EDDI: Explaining Data Drift using Influence

1st Nikolaos Myrtakis
Department of Computer Science & ETIS Lab
University of Crete & ENSEA
Cergy, France
nikos.myrtakis@ensea.fr

2nd Andrea Castellani
Honda Research Institute Europe GmbH
Offenbach, Germany
andrea.castellani@honda-ri.de

3rd Ioannis Tsamardinos
Department of Computer Science
University of Crete
Herakion, Crete
tsamard.it@gmail.com

4th Vassilis Christophides
ETIS Lab & IPAL
ENSEA, CYU, CNRS
Cergy, France & Singapore
vassilis.christophides@ensea.fr

***Abstract*—Data drift poses a significant challenge to the reliability of ML models in real-world applications, as the data distribution may change between training and inference phases. Existing drift monitoring systems have several limitations: (i) they detect either model drift or feature drift but not both; (ii) they rely on a plethora of detection methods, each making its own assumptions regarding the data and the ML models; (iii) they do not explain a drift in relation to the underlying probability distribution. To address these limitations, we propose EDDI, a novel influence-based drift detection and explanation framework that leverages the direct influence of samples on the decision boundary of the deployed predictive model. EDDI represents the first unified framework that utilizes Influence Functions to detect both model and feature drift. At the same time, EDDI provides a novel kind of explanation by revealing the probabilistic source of a drift. The key idea is that the influence distributions between drifted and non-drifted samples differ. Moreover, different drift types exhibit unique distributional influence characteristics, enabling their attribution. Based on a type-specific drift simulation, it is demonstrated that EDDI achieves statistically significant improvements on detection performance up to 15% over baselines across four drift types on 42 time-series classification datasets and reveals the drift type with up to 0.8 AUC. Moreover, we evaluated EDDI on two datasets with naturally occurring drift.**

***Index Terms*—Unified Drift Detection, Model Drift, Feature Drift, Data Drift Explanation, Influence Functions/Signals**

I. Introduction

Data drift [1] poses a critical challenge in the lifecycle of traditional machine learning (ML) models [2]. One reason supporting this observation is that ML models are trained assuming i.i.d. sampling. At the same time, in several real applications, the statistical properties of the serving data can differ from those of the training data, degrading the accuracy of the model and jeopardizing the reliability of its outcomes [3]. A data drift may be caused by non-stationary environments, seasonal patterns, or data collection issues related to sensors' tuning, aging, failures, etc. A drift monitoring system aims to address this problem; it typically operates in two phases: (i) drift detection and (ii) drift understanding (or explanation). For phase (i), numerous approaches have been proposed that detect either model drift, i.e., changes in the relationship of the input features and target, or feature drift, i.e., changes in the input features [4]–[7]. Phase (ii) aims to reveal the nature of the drift [8] to guide appropriate mitigation actions, such as model adaptation or re-training [1], [9]–[12]. The aforementioned pipeline is depicted in Fig. 1 (a). After the ML model training, each incoming window is assessed either based on its distributional difference with a reference set to detect feature drift or based on the accumulated prediction error (or loss), if labels for the window exist, to detect model drift if the error surpasses a certain threshold. Finally, the drift is usually explained via XAI methods [13].

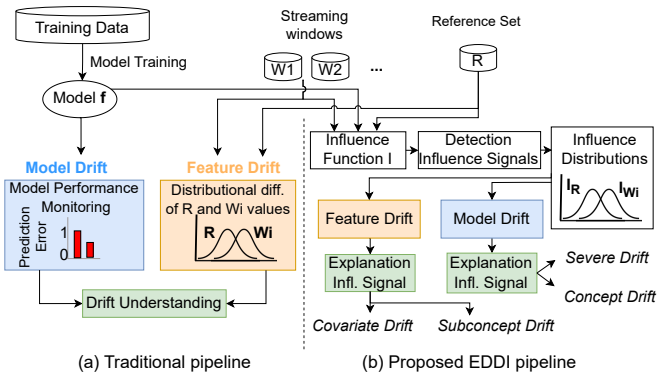


Fig. 1: Drift Monitoring Pipelines.

Motivating example. To motivate our work on why it is important to have a unified drift detection and explanation framework, consider a binary classification model f in Fig. 2, that is trained to map the input features X to a target variable Y, i.e., $f : X \rightarrow Y$. Then, the trained ML model is deployed to give label predictions for incoming samples in a streaming setting, arriving in the form of windows. At time t, the window W arrives, the decision boundary of f is used to predict samples drawn from a joint distribution $\mathcal{D}_W(X, Y)$, as depicted in Fig. 2. At time $t' > t$, a subsequent window W' with distribution $\mathcal{D}_{W'}$ arrives, and a drift occurs s.t. $\mathcal{D}_W \neq \mathcal{D}_{W'}$. However, different types of drift may occur as shown in Fig. 2 (a,b,c,d), requiring different mitigation actions.

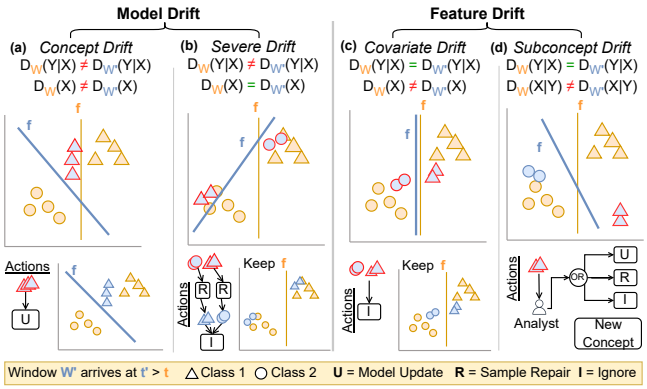


Fig. 2: Data drift types and actions. $D_W(Y|X)$ denotes the distribution in window W. Colors denote class labels, shapes input features, and red frames drift-triggering samples. The decision boundary is updated over time. Concept drift (a) needs model updates; severe drift (b) sample repair; covariate drift (c) can be ignored; subconcept drift (d) boundary adaptation.

Using a performance-based detector [10], a model drift can be detected based on the drop in inference performance. Changes in the input features may alter the posterior distribution for some of the target classes, e.g., due to sensors aging, lead to concept drift (a) [6], or directly alter the posterior distribution of the target classes without changes in the input features, e.g., due to potential label noise or corruption in the stream, leading to severe drift (b) [14]. Unlike (a), where the relevance of the previously learned decision boundary decreases for the incoming samples, and the model needs to be retrained [1], in (b), the data needs to be repaired, and the drift detector needs to be re-run to decide the next actions. Conversely, in the feature drift case, changes in the input features do not affect model performance [15]. However, distribution-based detectors [16] usually cannot distinguish between a covariate (c) and subconcept drift (d) [17], where changes of the latter concern mainly the class-conditioned distribution in a feature subspace. Although (c) may be tolerated, (d) requires updating the decision boundary so that the model learns the new data region that generalizes better to future samples. These drifting samples should be scrutinized by a human analyst to decide suitable mitigation actions (e.g., model adaptation using importance weighting on the training data [18]).

The Challenges and Gap. The first challenge is the unification of the drift detection task. As many approaches exist to detect feature and model drift, each with different assumptions regarding data distribution, modalities, or mechanisms to flag potential drift [6], [16], it is challenging to decide which approach to use for the data and model at hand. Second,*"blind" mitigation actions* without understanding the specific type of drifting samples are ineffective. More precisely, in the case of severe drift, adapting the model to corrupted drifting samples will harm its decision boundary and will subsequently deteriorate the predictive performance [19]. Inversely, in the case of subconcept drift, not adapting the model to a new feature area will negatively affect its decision boundary for future samples. To the best of our knowledge, existing detectors can detect either model or feature drift, but not both under a common framework. Moreover, the drift explanation mainly focuses on selecting the samples responsible for the drift [13]. This type of drift explanation aims to reveal *where* to adapt rather than *how* the model's decision boundary will be affected by the drift. In essence, the effect of a drift on the current and historical samples remains unknown, obscuring the choices of appropriate mitigation strategies for the data at hand [1], [9]–[12]. Note that change point detection [20] methods also aim to discover when or where (localization) a distribution shift occurred rather than attributing the probabilistic source of the drifted region, as studied in this work.

Our Solution. To address these challenges, we propose EDDI, an explainable data drift monitoring framework based on sample influence. EDDI leverages the influence of individual samples on a deployed predictive model as a unified mechanism for both drift detection and explanation. To efficiently quantify influence in a streaming setting, EDDI relies on a gradient-based influence estimator that measures loss-gradient alignment between pairs of samples. At the window level, influence magnitude is used to detect the presence of drift, while influence directionality captures how incoming samples affect historical data and the model, enabling drift type attribution. Unlike traditional drift pipelines that compare raw data or performance statistics (see Fig. 1.a) EDDI operates directly on influence distributions induced by the predictive (gradient-based) model, thereby linking drift patterns to decision-boundary behavior (see Fig. 1.b).

Novelty. EDDI introduces three novel aspects. First, it is the first unified framework that jointly detects and explains multiple forms of model and feature drift, which prior work has largely treated as separate problems [11]. Second, EDDI is the first approach to characterize drift explicitly through the influence of individual samples on the decision boundary of the deployed model. Third, EDDI provides a novel probabilistic explanation of drift by identifying its distributional source: distinct drift types manifest as characteristic changes in influence distributions, enabling principled attribution.

Existing drift detection methods primarily explain when drift occurs, while some focus on identifying affected samples (*what*), responsible features (*where*), or drift magnitude (*strength*). In contrast, EDDI addresses the largely unexplored how by revealing how changes in posterior, class-conditional, and marginal distributions translate into different drift behaviors at the model level. To this end, we introduce two influence-based signals that quantify drift magnitude and two complementary signals that capture influence directionality. Together, these signals allow EDDI to both detect drift and classify it into one of four canonical types. To the best of our knowledge, EDDI is the first influence-based framework that unifies drift detection and explanation. Prior work [21] has shown that influence signals are effective for sample repair. In this paper, we instead focus on demonstrating why drift explanations are essential, showing that different drift types require fundamentally different mitigation strategies, an aspect

that remains largely underexplored in the drift adaptation literature. In summary, our contributions are:

Unified drift detection framework. We propose EDDI, the first hyperparameter-free framework that jointly detects concept, severe, covariate, and subconcept drift using sample influence.

Influence-based drift detection and explanation. We introduce four influence signals that exploit distinct influence footprints of samples from drifted and non-drifted windows to both detect drift and explain its type.

Comprehensive experimental evaluation. We evaluate EDDI on 42 real-world time-series using a type-specific drift injection protocol, as well as on real-world datasets with naturally occurring concept drift. We compare EDDI against popular performance and distribution-based drift detectors, using ML models of different architectures, including convolutional, Transformer-based models, and foundation models.

Robust performance across drift types. We show that influence-based signals achieve up to 15% higher detection performance than existing methods across all drift types, and that drift type attribution reaches an AUC of up to 0.8 for both model and feature drift.

The remainder of the paper is organized as follows. Sections II–IV introduce background, definitions, and the proposed influence-based signals. Sections V and VI describe the experimental setup and results. Related work is discussed in Section VII, and Section VIII concludes the paper.

II. Preliminaries

In this section, we introduce the notation used throughout the paper and provide a background on the drift types.

Let $\mathcal{D}$ be the stationary joint distribution of the training data on $\mathcal{X} \times \mathcal{Y}$, where $\mathcal{X}$ is the input features and $\mathcal{Y}$ the labels. A classification model $f : \mathcal{X} \to \mathcal{Y}$ is trained using samples drawn i.i.d. from $\mathcal{D}$ where each sample z_i is a feature/label pair such that $z_i = (x_i, y_i) \sim \mathcal{D}$ called samples. The model f is parameterized by $\theta \in \mathbb{R}^p$, where p is the total number of parameters. These parameters are learned by optimizing a loss function $\mathcal{L} : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$. To ease the notation, we denote $\mathcal{L}(z_i; \theta) := \mathcal{L}(y_i, \hat{y}_i = f_\theta(x_i))$. The optimal parameters are obtained by minimizing the empirical risk (ERM): $\hat{\theta} = \operatorname{argmin}_\theta \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(z_i; \theta)$, where n denotes the number of observed training samples. The generalization performance of f is evaluated on a held-out reference set R, where $R \sim \mathcal{D}$.

In streaming settings, the data arrives consecutively over time. A data stream $\{z_t\}_{t=1}^{\infty}$, is an infinite sequence of samples where each $z_t \sim D_t$ is a new sample arriving at a different timestamp t. We follow the formal framework proposed by [22] for adapting the problem description by considering a model of time $\mathcal{T}$ and a collection of (possibly different) distributions $\mathcal{D}_t$ on $\mathcal{X} \times \mathcal{Y}$, indexed over $\mathcal{T}$. Assuming that there is a distribution P_T on $\mathcal{T}$ that describes the likelihood of observing a sample z_t at time t, and a collection of distributions $\mathcal{D}_t$ for all $t \in \mathcal{T}$ albeit, in practice, only a finite number of time points is observed. Together P_T and $\mathcal{D}_t$ form a distribution process (in the literature this is also referred to as *drift process* (P_T, $\mathcal{D}_t$) [8]). The time-enriched representation of the data $\mathcal{T} \times (\mathcal{X} \times \mathcal{Y})$ can be extended to a general statistical data interdependence setting by decomposing it into a distribution P_T on $\mathcal{T}$ and the conditional distribution $\mathcal{D}_t$ on $\mathcal{X} \times \mathcal{Y}$ such that $X, Y \mid T = t \sim \mathcal{D}_t$.

Typically, a data stream is processed using time windows $W \subset \mathcal{T}$ that are P_t non-null sets. A sample set of size l observed during W, denoted as $S = \{z_t\}_{t=i}^{l}$, is drawn i.i.d. from the mean distribution on W, that is $\mathcal{D}_W = \mathcal{D}(X, Y | T \in W)$. That resembles the practical procedure, where one obtains a sample set S during W from another sample set S' during W', by selecting those entries of S' that are observed during W. In particular, if $W = \{0, \ldots, t-1\}$, $W' = \{1, \ldots, t\}$, this corresponds to a sliding window that is updated with a single sample. Depending on the windowing strategy [11], other kinds of updates can be considered. In sliding windows, the overlap between W and W' could be any number of timestamps, i.e., $W \subset W'$, while in tumbling windows, the overlap is empty, i.e., $W \not\subset W'$.

In this context, we refer to *data drift* as the phenomenon where the data distribution $\mathcal{D}_W$ of a time window W differs from the distribution $\mathcal{D}_{W'}$ of window W' arriving later, such that $\mathcal{D}_W \neq \mathcal{D}_{W'}$. According to the Bayesian Decision Theory, the probabilistic source of a change in $\mathcal{D}_W(X, Y)$ may be attributed to shifts in the posterior distribution $\mathcal{D}_W(Y|X) \neq \mathcal{D}_{W'}(Y|X)$, the class-conditioned distribution $\mathcal{D}_W(X|Y) \neq \mathcal{D}_{W'}(X|Y)$, the marginal distribution $\mathcal{D}_W(X) \neq \mathcal{D}_{W'}(X)$ or even the prior distribution $\mathcal{D}_W(Y) \neq \mathcal{D}_{W'}(Y)$. We focus on changes in posterior, class-conditioned, and marginal distributions. Depending on the probabilistic source of a data drift may affect differently the performance of predictive models[1]. We refer to a performance-impacting drift as *model drift* and performance-invariant drift as *feature drift*.

We first consider fine-grained types of model drift by distinguishing between concept and severe drift. *Concept drift* (see Fig. 2-a) $\mathcal{F}_{\text{con}}$ [6], [12], [14], [24] causes a change in the input features altering the posterior distribution for some of the classes between a preceding W and incoming window W', as commonly observed in practice [25], such that $\mathcal{F}_{\text{con}} \coloneqq \mathcal{D}_W(Y|X) \neq \mathcal{D}_{W'}(Y|X) \wedge \mathcal{D}_W(X) \neq \mathcal{D}_{W'}(X)$. *Severe Drift* $\mathcal{F}_{\text{sev}}$ [6] causes a change directly to the posterior distribution of the target classes without affecting the input features such that $\mathcal{F}_{\text{sev}} \coloneqq \mathcal{D}_W(Y|X) \neq \mathcal{D}_{W'}(Y|X) \wedge \mathcal{D}_W(X) = \mathcal{D}_{W'}(X)$. This drift type is depicted in Fig. 2-b, also called full-concept drift according to prior works [14]. Plausible reasons for this change could be a sensor failure, a corruption in the data labelling process, or another phenomenon requiring further investigation. Note that severe drift can also be injected maliciously (adversarial drift) into the data stream to deliberately alter the decision boundary [19].

[1]Performance impacting vs invariant concept drift is usually referred to in the literature as real vs virtual, i.e., stating the conditional statistical dependence or independence of label y and time T given data X [23].

Unlike the previous types of drift, feature drift does not affect the model's performance, i.e., $\mathcal{D}_W(Y|X) = \mathcal{D}_{W'}(Y|X)$, and it is usually categorized as covariate and subconcept drift. *Covariate Drift* [6], [14], [26] depicted in Fig 2-c, is a process $\mathcal{F}_{\text{cov}}$ that change the marginal distribution such that $\mathcal{F}_{\text{cov}} := \mathcal{D}_W(X) \neq \mathcal{D}_{W'}(X)$. Finally, *Subconcept Drift* [17] concerns changes in the class-conditioned distribution in a feature subspace and it is formally defined as $\mathcal{F}_{\text{sub}} := \exists_{y \in Y} \forall_{x \in X} \mathcal{D}_W(x|y) \neq \mathcal{D}_{W'}(x|y)$. The challenging aspect of this drift type is that the model performance on the current window may not be reduced, as depicted in Fig. 2-d, yet the decision boundary may be substantially altered for predicting future samples in that data area if the model is updated.

III. Drift Detection and Explanation

In this section, we introduce the definitions for the proposed influence-based drift detection and explanation framework. The main idea lies in assessing the influence of an incoming window on the model's loss on current and historical samples. The influence quantification enables us to monitor *how* and in *what* ways a potential parameter update would affect historical and current samples' loss. Hence, based solely on the window's samples' impact on the decision boundary, the influence information allows us to state (a) sufficient conditions for detecting model and feature drift, and (b) necessary conditions for mapping model drift to severe and concept drift and feature drift to covariate and subconcept drift.

Influence-based drift detection requires the estimation of the impact of an incoming window on the model's decision boundary through loss before and after the parameter update. Updating the model's parameters with a sample $z' \sim \mathcal{D}_{W'}$ arriving at the current window W' can be defined as *Insert-One-Sample-In* (IOSI). IOSI is the symmetric evaluation protocol to the *Leave-One-Out-Retraining* (LOOR) [27] used to assess the influence on parameters if a sample is removed from the training set, followed by model retraining. IOSI has been used in prior works on sequential settings such as Active Learning [28], and it quantifies the change in the model's parameters $\hat{\theta}$ if a new sample z' is used in the learning process:

$$\hat{\theta}_{+z'} = \underset{\theta}{\operatorname{argmin}} \ \frac{1}{l} \sum_{i=1}^{l} \mathcal{L}(z_i; \theta) + \epsilon \mathcal{L}(z'; \theta), \tag{1}$$

where z_i is a sample of a preceding time window W, i.e., $z_i \sim \mathcal{D}_W$, l is the number of observations sampled from $\mathcal{D}_W$, and ϵ is the weight of z' to the parameter updates. In the case of LOOR parameter influence $\epsilon = -\frac{1}{l}$, while in IOSI, $\epsilon = \frac{1}{l}$. Note that prior samples may receive a different weight, i.e., importance on the parameter updates, to prioritize knowledge of new samples. For simplicity, we consider that past and incoming samples have equal weight in the loss. The IOSI of sample z' on the loss of another sample z is defined as:

$$\delta_{\mathcal{L}}(z, \hat{\theta}, \hat{\theta}_{+z'}) = \mathcal{L}(z; \hat{\theta}) - \mathcal{L}(z; \hat{\theta}_{+z'}).$$

The intuition for studying the influence of incoming samples stems from the fact that dissimilar data distributions from what the model has learned may substantially affect its decision boundary when predicting current or historical samples. To capture all possible updating effects, we analyze the influence of incoming data from three perspectives: *self-influence*, *intra-influence*, and *inter-influence*.

Self-influence. Self-influence quantifies how the update of model parameters caused by an incoming sample $z' \sim \mathcal{D}_{W'}$ affects its loss, defined as $\delta_{\mathcal{L}}(z', \hat{\theta}, \hat{\theta}_{+z'})$. Although self-influence has been proposed in prior works to discover errors in labels for i.i.d. samples [27], [29], to the best of our knowledge, this is the first work that applies self-influence in a drift detection task. High self-influence indicates that the feature/target combination of z' contradicts prior samples that the model learned from and will cause a large parameter update, attracting the decision boundary towards its area. This can be observed in Fig. 2-a,b for the blue samples that fall in the orange region. To reduce the loss for those samples, the model parameters need to change significantly. If many samples in the window exhibit high self-influence, then it is likely that a drift occurs.

Intra-influence. Intra-influence quantifies how the update of model parameters caused by a sample z'_i of a time window W' affects the loss of other samples z'_j observed in the same window. Formally, given two samples $z'_i, z'_j \sim \mathcal{D}_{W'}$ where $z'_i \neq z'_j$, the intra-influence of z'_i is defined as $\delta_{\mathcal{L}}(z'_j, \hat{\theta}, \hat{\theta}_{+z'_i})$. Although a sample may not cause large parameter updates, it may still exhibit high influence on its neighbors, especially on sparse or unexplored areas of the data space. This is illustrated in Fig. 2-c,d, where the incoming samples fall in sparse areas where the model lacks statistical information. Although these samples have low self-influence, as there is no label conflict with other samples in that region, their presence may significantly impact the loss of their neighbors.

Inter-influence. Inter-influence quantifies how the update of model parameters caused by a sample z' of a time window W' will affect the loss of samples z observed in a previous window W. Formally, given two samples $z \sim \mathcal{D}_W$ and $z' \sim \mathcal{D}_{W'}$ the inter-influence of z' is defined as $\delta_{\mathcal{L}}(z, \hat{\theta}, \hat{\theta}_{+z'})$. The inter-influence indicates how many decision boundary rules are expected to be broken if the model is updated using z'. For instance, in Fig. 2-a, the new samples do not increase the loss of historical data; however, this does not hold in Fig. 2-b. When many arriving samples have high inter-influence, significantly increasing or decreasing the loss of historical samples, then the window may exhibit a drift.

Together, the three influence perspectives allow full characterization of how an incoming window will affect the model's boundary. When the incoming samples are drawn from the same distribution that the model has been trained on, their IOSI influence distribution is expected to be the same as the historical (non-drifted) data. In other words, if a window is drawn from the original data distribution, then its influence on the model should resemble the influence of historical samples; otherwise, the window exhibits a drift.

Problem 3.1: ***Influence-based drift detection.*** Let S and S' be two sample sets drawn from $\mathcal{D}_W$ and $\mathcal{D}'_W$ respectively. An influence-based model or feature drift occurs when the

past and current self, intra, or inter-influence distributions are different, denoted as $P_S(\delta_{\mathcal{L}}) \neq P_{S'}(\delta_{\mathcal{L}})$.

Detecting a drift is the first step of a drift monitoring system. To decide the appropriate mitigation actions (i.e., model adaptation vs data repair), we also need to reveal the drift type at the window level (see Fig. 2). The main assumption for explaining a detected drift is that each probabilistic source of drift has a distinct influence footprint on the model's decision boundary. Hence, our explanation method aims to map the influence distribution of a window to the root-cause of the drifted samples.

Problem 3.2: ***Influence-based drift explanation.*** Let $\mathcal{F} = \{\mathcal{F}_{\text{con}}, \mathcal{F}_{\text{sev}}, \mathcal{F}_{\text{cov}}, \mathcal{F}_{\text{sub}}\}$ be the set of the possible drift types and S' some drifting observations drawn from an incoming window W'. An influence-based drift explanation is the mapping of the influence distribution $P_{S'}(\delta_{\mathcal{L}})$ to a drift type $\mathcal{F}$ such that $P_{S'}\big(\delta_{\mathcal{L}}\big) \rightarrow \mathcal{F}$.

IV. Influence-based Signals

IOSI (see Eq. 1) is an exact but computationally expensive method for measuring samples' influence on the decision boundary, as it requires incrementally retraining the model for every new sample. In addition, two model versions must be retained in the main memory, one version before parameter update and one after the update to compute the loss differences. The previous two limitations make IOSI impractical for large models in streaming settings. Hence, we rely on the TracIn Influence Function (IF) [29] to efficiently approximate IOSI. TracIn estimates the influence of sample z' on z by summing the loss gradients' alignment of the pair (z', z) across the training process such that $I(z', z) = \sum_{i=1}^{E} \eta_i \nabla \mathcal{L}(z, \theta_i)^T \cdot \nabla \mathcal{L}(z', \theta_i)$, where i denotes a particular epoch, E is the total number of epochs, θ_i denotes the model parameters updated using the sample z' and η_i is the learning rate at epoch i. As EDDI operates on a streaming setting, the only available information is the final model parameters $\hat{\theta}$ and a small non-drifted reference window $R \sim \mathcal{D}$. The TracIn influence in this setting becomes $I(z', z) = \nabla \mathcal{L}(z, \hat{\theta})^T \cdot \nabla \mathcal{L}(z', \hat{\theta})$. Note that the influence score is symmetric, i.e., $I(z', z) = I(z, z')$.

The loss gradient dot product essentially captures the magnitude and directionality of a sample's influence if the model is updated using its features and label combination. The influence magnitude quantifies how much other samples will be affected. Typically, larger values signify a larger impact on the decision boundary. The influence directionality (influence score sign) quantifies in what way the boundary will change to incorporate the new sample, showing how the loss of other observations will be affected. In this sense, $I(z', z)$ can be (i) negative, when two samples increase each other, (ii) positive, when two samples decrease each other, or (iii) neutral, when two samples do not affect each other. Subsequently, we build on the notions of magnitude and directionality to form signals that detect and explain the four types of drift.

A. Drift Detection

In this section, we propose influence-based signals to detect model and feature drift, i.e., aiming to address the Problem 3.1. A detection signal leverages the influence *magnitude* of the incoming samples to spot a model or feature drift. The main principle is that label conflicts, e.g., mislabeled samples, and input features with deviating patterns are expected to have a bigger impact on the construction of the decision boundary due to larger parameter updates. We propose two influence signals for drift detection, namely Self Influence for model drift and Window Influence for feature drift.

Self Influence (SI). SI aims to identify concept and severe drift, i.e., drifting windows that affect the model's performance (and loss). This signal measures the influence of a sample z on its loss. Formally, SI is defined as the gradient magnitude of z such that:

$$\text{SI}(z) = \nabla_\theta \mathcal{L}(z; \hat{\theta})^T \cdot \nabla_\theta \mathcal{L}(z; \hat{\theta}). \tag{2}$$

Intuitively, a sample exhibits an SI relative to its attraction force to the decision boundary. For instance, in Fig. 2-a,b, the incoming samples that are mispredicted will cause a large parameter update that would significantly alter the decision boundary to reduce the high loss of these samples. Hence, the loss gradient magnitude will be higher, causing a larger SI than samples predicted correctly by the model (i.e., with smaller error). SI has a linear time complexity w.r.t. the sample size of a window $O(ld)$, where l is the total observations in the window and d denotes the number of model parameters.

Window Influence (WI). WI aims to identify covariate and subconcept drift, i.e., drifting windows that do not affect the model's predictive performance (and loss). These types of drift are more challenging than model drift, as the labels may not be available; thus, the predicted labels are used to compute the influence. Even if the labels are available, predicting the samples exhibiting feature drift will not cause an increase in the model's loss, as defined in Section II. Thus, the gradient magnitude for such drift types is expected to be small as it is related to the model's error, resulting in an SI of lower magnitude. However, a low SI does not necessarily indicate low intra-influence, since predictions for samples in sparse regions can be affected by a few nearby samples. (see Fig. 2-c,d). To this end, we propose WI leveraging the intra-influence of the window's samples using their *predicted* labels. Given a window $\hat{S}'$ with l samples, where $\hat{S}' = \{\hat{z_i}' = \big(x_i, f(x_i; \hat{\theta})\big)\}_{i=1}^{l}$, the WI of $\hat{z}' \in \hat{S}'$ is defined as the average absolute influence of $\hat{z}'$ to $\hat{S}'$:

$$\text{WI}(\hat{z}' \rightarrow \hat{S}') = \frac{1}{l} \sum_{i=1}^{l} |\nabla_\theta \mathcal{L}(\hat{z}'; \hat{\theta})^T \cdot \nabla_\theta \mathcal{L}(\hat{z_i}'; \hat{\theta})|. \tag{3}$$

To avoid canceling effects in case of negative and positive influence scores during the sum aggregation, we consider the absolute pair-wise influence, capturing the magnitude of the inter-influence of each sample. The intuition behind WI stems from the fact that feature drifting samples will have a greater influence on each other as the model lacks statistical information for the low-density regions. Thus, every parameter

Algorithm 1: Model Drift Detection with Self Influence

Data: The window's samples S', a reference set R, the model's parameters $\hat{\theta}$, a drift detector DM, a threshold α

Result: A boolean flag

Function *SI_Detection*(S', R, $\hat{\theta}$, DM, a):
  $\text{SI}_{S'} \leftarrow \langle \text{SI}(z') \rangle_{z' \in S'}$ // Eq. 2
  $\text{SI}_R \leftarrow \langle \text{SI}(z) \rangle_{z \in R}$
  p-value $\leftarrow \text{DM}(\text{SI}_{S'}, \text{SI}_R)$
  if *p-value* $< \alpha$ **then**
    return True // Drift detected
  else
    return False // No drift detected
  end
end Function

Algorithm 2: Feature Drift Detection with Window Influence

Data: The window's samples $\hat{S}'$, a reference set R, the model's parameters $\hat{\theta}$, a drift detector DM, a threshold α

Result: A boolean flag

Function *WI_Detection*($\hat{S}'$, R, $\hat{\theta}$, DM, a):
  $\text{WI}_{\hat{S}'} \leftarrow \langle \text{WI}(\hat{z} \rightarrow \hat{S}') \rangle_{\hat{z}' \in \hat{S}'}$ // Eq. 3
  $\text{WI}_R \leftarrow \langle \text{WI}(z \rightarrow R) \rangle_{z \in R}$
  p-value $\leftarrow \text{DM}(\text{WI}_S, \text{WI}_R)$
  if *p-value* $< \alpha$ **then**
    return True // Drift detected
  else
    return False // No drift detected
  end
end Function

update in those areas will impact the model's decisions locally. WI time complexity is $O(l^2 d)$ as it computes the influence between each sample pair in the window.

Applying the detection signals on the window will result in l non-negative influence scores, i.e., each sample receives a score quantifying its influence magnitude on the decision boundary. A model or feature drift is detected when the influence distributions of incoming samples and the non-drifted reference set differ (see Def. 3.1). Hence, a univariate distribution-based drift detector [30] suffices to compare the two influence distributions. Note that the objective of this work is not to propose a method to quantify distributional distance but rather influence signals upon which the differences are measured by an existing detector. The values of the influence signals do not correspond to distances in the raw feature space; thus, Euclidean-based drift detectors must be avoided [31].

Alg. 1 and 2 illustrate the steps for detecting all types of drift using SI and WI, respectively. Recall that SI is measured using the observed labels of the window, while WI is measured solely on the predicted labels. Both algorithms require a significance threshold, and hence the detection method DM should output a p-value [30]. The null hypothesis on both algorithms is that the influence distributions are identical, i.e., the p-value is below the significance threshold, and thus, a drift occurs; otherwise, the window does not exhibit a drift. Note that the reference set R is assumed to be static and fixed-size with non-drifted historical samples. This is a standard practice when employing distribution-based detectors [16]. Although R can be updated using new non-drifting samples or using drifting samples where the model adapts (as in reservoir pooling [32]), modifying R is outside the scope of this work.

B. Drift Explanation

We turn our attention to the drift explanation, aiming to address the Problem 3.2. The previous detection signals identify a drift when the influence distributions between incoming and reference samples differ. However, windows with high influence on the model, although worth inspecting, do not reveal how new samples will affect historical or even future data. To this end, we propose two explanation signals that analyze the influence distributions of a detected drift to differentiate between severe and concept drift, as well as covariate and subconcept drift, at the window level. For mapping each drift to a probabilistic source of change according to Def. 3.2, we propose two drift explanation signals leveraging the influence *directionality*, namely Negative Influence to explain model drift and Balanced Influence to explain feature drift.

Negative Influence (NI). Given some drifting samples detected by SI, NI aims to differentiate between severe and concept drift. This is achieved by comparing the negative inter-influence distributions of the drifting samples on the reference set using their observed and predicted labels. For example, the influence of the misclassified samples of Fig. 2-b will resemble the influence of the non-drifted samples if the predicted (and correct) class is taken into account. Conversely, a window that exhibits concept drift is more likely to preserve its observed and predicted influence[2] on the reference set as the drifted samples fall in an unexplored area whose adaptation will not negatively affect historical samples significantly (see Fig. 2-a). Given an incoming sample $z' \in S'$ where $S' \sim \mathcal{D}_{W'}$ and a reference set $R \sim \mathcal{D}$ with m samples, NI of z' is computed as follows using its observed label on R:

$$\text{NI}(z' \rightarrow R) = \frac{1}{m} \sum_{i=1}^{m} \min\big(\nabla_\theta \mathcal{L}(z'; \hat{\theta})^T \cdot \nabla_\theta \mathcal{L}(z_i; \hat{\theta}),\ 0\big). \quad (4)$$

Similarly, the predicted influence of z' is denoted as $\text{NI}(\hat{z}' \rightarrow R)$, where $\hat{z}' = (x, f(x; \hat{\theta}))$. NI reveals a severe drift if the predictive and observed intra-influence distributions of a window S' statistically differ s.t. $P\big(\text{NI}(S' \rightarrow R)\big) \neq P\big(\text{NI}(\hat{S}' \rightarrow R)\big)$; otherwise, the window exhibits a concept drift. The time complexity of NI is $O(lmd)$, where l is the number of the

[2]We refer to observed/predicted influence when the influence is computed based on observed/predicted labels.

Algorithm 3: Negative Influence for Model Drift Explanation

Data: SI-detected window S' with observed and $\hat{S}'$ with predicted labels, reference set R, model's parameters $\hat{\theta}$, drift detector DM, threshold α
Result: The model drift type

Function *NI_Explanation*(S', $\hat{S}'$, R, $\hat{\theta}$, DM, a):
  $\text{NI}_{S'} \leftarrow \langle \text{NI}(z' \rightarrow R) \rangle_{z' \in S'}$ // Eq. 4
  $\text{NI}_{\hat{S}'} \leftarrow \langle \text{NI}(\hat{z}' \rightarrow R) \rangle_{\hat{z}' \in \hat{S}'}$
  p-value $\leftarrow \text{DM}(\text{NI}_{S'}, \text{NI}_{\hat{S}'})$
  if *p-value* $< \alpha$ **then**
    return "Severe Drift"
  else
    return "Concept Drift"
end Function

Algorithm 4: Balanced Influence for Feature Drift Explanation

Data: WI-detected window with predicted labels $\hat{S}'$, a reference set R, the model's parameters $\hat{\theta}$, a drift detector DM, a threshold α
Result: The feature drift type

Function *BI_Explanation*($\hat{S}'$, R, $\hat{\theta}$, DM, a):
  $\text{BI}_{\hat{S}'} \leftarrow \langle \text{BI}(\hat{z}' \rightarrow \hat{S}') \rangle_{\hat{z}' \in \hat{S}'}$ // Eq. 5
  $\text{BI}_{R} \leftarrow \langle \text{BI}(z \rightarrow R) \rangle_{z \in R}$
  p-value $\leftarrow \text{DM}(\text{BI}_{\hat{S}'}, \text{BI}_{R})$
  if *p-value* $< \alpha$ **then**
    return "Subconcept Drift"
  else
    return "Covariate Drift"
end Function

window's samples, m is the reference set size, and d is the number of model parameters.

Balanced Influence (BI). Given some drifting samples detected by WI, BI aims to discern covariate from subconcept drift by analyzing the balance of the positive over the negative intra-influence of drifting samples. For example, in the covariate drift of Fig. 2-c the new samples lie close to historical ones, causing a small disturbance on the boundary. Conversely, the subconcept drift of Fig. 2-d alters the boundary locally even though the model error remained unaltered; such a type of drift may lead to biased future predictions, especially close to the shifted region. Similar to WI, BI does not require observed labels and calculates the influence using the predicted labels. Given an incoming sample $\hat{z}' \in \hat{S}'$ where $\hat{S}' \sim \mathcal{D}_{W'}$ is the window's samples with their predicted labels, BI is defined as the positive over the negative influence ratio of a sample z' on $\hat{S}'$:

$$\text{BI}(\hat{z}' \rightarrow \hat{S}') = \frac{1}{l} \sum_{i=1}^{l} \frac{\max\big(\nabla_\theta \mathcal{L}(\hat{z}'; \hat{\theta})^T \cdot \nabla_\theta \mathcal{L}(\hat{z}_i'; \hat{\theta}),\ 0\big)}{\min\big(\nabla_\theta \mathcal{L}(\hat{z}'; \hat{\theta})^T \cdot \nabla_\theta \mathcal{L}(\hat{z}_i'; \hat{\theta}),\ \epsilon\big)}, \tag{5}$$

where $\epsilon > 0$ is a small number to prevent division by 0. BI reveals a subconcept drift if the positive over negative intra-influence distribution of $\hat{S}'$ and R samples statistically differ, i.e., $P\big(\text{BI}(\hat{S}' \rightarrow \hat{S}')\big) \neq P\big(\text{BI}(R \rightarrow R)\big)$. Similar to WI, the time complexity of BI is $O(l^2 d)$ as it computes the influence between each sample pair in the window.

The steps to apply NI and BI are illustrated in Alg. 3 and 4. To explain model drift, a drifting window detected using SI is assessed on the distributional overlap of the observed and predicted negative influence using a statistical drift detector DM that outputs a p-value, similar to the detection algorithms. The null hypothesis is that the two distributions are identical, i.e., the p-value is below the significance threshold in case of severe drift; otherwise, the window exhibits concept drift. Similar steps are followed for the feature drift explanation.

For completeness, the pipeline of the proposed EDDI framework is illustrated in Alg. 5. EDDI's computational complexity for detection is $O(l^2 d)$ in the case that there is no drift, i.e., both SI and WI run without triggering. It first detects model drift, if the window labels are available, using the *SI_Detection* function. This prevents EDDI from incurring a computational complexity of $O(l^2 d)$ in the presence of concept drift, as both the SI and WI signals would otherwise be triggered since the distribution of the input features also changes during concept drift. Note that, depending on the drift monitoring needs, one may want to run a combination of the explanation signals, if both SI and WI fire, to reveal whether the feature changes due to concept drift resemble covariate or subconcept drift. If there is no model drift or the window labels are not available (SI can not be used in this case), *WI_Detection* function runs to spot a potential feature drift; if feature distribution changes are detected, BI discerns covariate from subconcept drift. EDDI assumes one drift type per window, a common assumption of all prior works considering different drift types (e.g., on the temporal characteristics of a drift [12], [33]).

Algorithm 5: EDDI: Explainable Data Drift Monitoring using Influence

Data: A window S' with observed (if exists) and $\hat{S}'$ with predicted labels , a reference set R, the model's parameters $\hat{\theta}$, a drift detector DM, a threshold α
Result: Drift Type (if detected)

drift_type $\leftarrow \emptyset$ // $\emptyset$ signifies no drift
if *window_labels_exist* = *True* **then**
  if *SI_Detection*(S', R, $\hat{\theta}$, *DM*, a) = *True* **then**
    drift_type $\leftarrow$ *NI_Explanation*(S', $\hat{S}'$, R, $\hat{\theta}$, DM, a)
if *drift_type* $= \emptyset \wedge$ *WI_Detection*($\hat{S}'$, R, $\hat{\theta}$, *DM*, a) = *True* **then**
  drift_type $\leftarrow$ *BI_Explanation*($\hat{S}'$, R, $\hat{\theta}$, DM, a)
return drift_type

V. Experimental Setting

This section describes the experimental setting and the evaluation pipeline used in our work. All experiments were run on a Linux machine with a 12-core processor and 64 GB of main memory. The code is available in our GitHub repository[3]. EDDI is agnostic to the underlying data modality and applies to any streaming data, including tabular, image, and text data. We focus our empirical evaluation on time-series data, where the temporal nature of the data makes the notion of drift particularly well defined and naturally supports window-based streaming analysis. We evaluate EDDI using several state-of-the-art models from different architectural families, including convolutional networks, the Transformer-based TST [34], the foundation model MOMENT [35], and the convolutional model InceptionTime [36]. We employ 42 real-world time series (TS) classification datasets from the UCI repository [37] with various characteristics: sample size range $\in (442, 16647)$, classes $\in (2, 42)$, timestep range $\in (60, 1882)$. Moreover, EDDI is evaluated on the Electricity [38] and Insect [39] dataset, known to exhibit model drift [40].

EDDI performance is compared against the five model drift: EDDM [41], ADWIN [42], STEPD [43], HDDM-A and HDDM-W [44]; and three feature drift baselines: MMD [26], LSDD [45] and CVM. They are implemented in the libraries Alibi-Detect[4], and the Frouros[5]. For drift detection, EDDI relies on the Incremental-Kolmogorov-Smirnov [46] (IKS) Test due to its efficiency in streaming settings. Regarding the drift type explanation, the offline KS-Test was applied on the drifted window. To the best of our knowledge, there are no prior works that can reveal the type, i.e., the probabilistic source, of the drift in streaming settings. As described in Section VII-B, most prior works focus either on feature attribution to highlight a drift or on drifting sample selection and segmentation.

For each dataset, the model is trained on an initial 70% split. A disjoint 10% split is used as a fixed reference set against which incoming data are compared for drift detection. The remaining 20% of the samples are ordered by time and processed in consecutive windows to emulate a data stream.

Evaluating concept drift detection algorithms is a well-known challenge due to the scarcity of real-world datasets with reliable drift annotations [12], [47], [48]. Consequently, synthetic drift injection mechanisms are commonly adopted to construct controlled benchmark settings for systematic evaluation [15], [49]. In our experiments, training and reference sets remain unchanged, and only test samples are modified by injecting a sudden drift after 50% of the test stream. We repeat each experiment for different drift types. Concept drift is induced by perturbing input features with adversarial noise to increase model error while preserving true labels. Covariate drift follows a similar perturbation scheme but preserves model predictions, leaving accuracy unaffected. Subconcept drift is simulated by excluding one class during training and subsequently injecting its samples into the data stream with predicted labels, ensuring unchanged performance. Readers are referred to [50] for a comprehensive description of the synthetic drift injection scheme.

Since drift is injected abruptly, we adopt a small sliding window of size 10 with a stride of 1, which has been shown to be an optimal size in cases of abrupt changes [51]. Although window size is inherently problem-dependent, we fix it across all datasets and methods to ensure fair comparisons and to avoid dataset-specific tuning. EDDI and all baseline methods operate at the window level and assign a binary drift label to each sample within a window. Detection performance is evaluated at the sample level using Balanced Accuracy (BA) [52]. Unlike time-based metrics [53], which emphasize the prompt detection of drift onset, our evaluation focuses on classification accuracy. BA is robust to class imbalance and, unlike the F1-score, yields a constant random performance baseline of 0.5 across all datasets. Finally, we set the α hyperparameter to 0.05, as we have experimentally confirmed the robustness of EDDI detection performance reported in the next Section to different thresholds.

VI. Experimental Results

A. Detection of Synthetic Drift

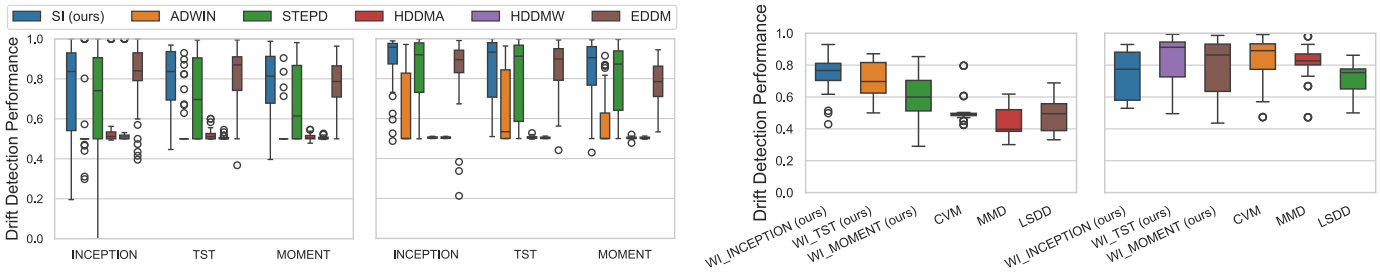
The objective of this experiment is to assess how the SI and WI influence signals coupled with the IKS-Test detector, and compare with existing detectors for model and feature drift. The detection performances for each drift type are illustrated in Fig. 3. Note that each bar corresponds to the result obtained from running a corresponding detector across the entire collection of datasets. In Fig. 3 (a), the performances for concept drift detection are depicted across the two ML models. Note that SI uses the model's gradients and the baselines use the model's label prediction vector across windows to detect a drift, also known as prequential error [54]. SI coupled with IKS is on par or better than the baselines at accurately detecting the samples under concept drift for all models. A similar trend is observed for severe drift as shown in Fig. 3 (b). The detection performance of SI has increased by 13% on average compared to concept drift. This is explained by the fact that under severe drift, the gradient magnitudes increase as the label conflicts would require substantial changes in the model's decision boundary. Under severe drift, SI outperforms the rest of the baselines. A notable strength of SI compared to the baselines is its reduced sensitivity to the lower predictive performance. This can be observed by analyzing the average predictive performance of the models; the average classification balanced accuracy of TST is (0.74 ± 0.22), for MOMENT is $0.58 \pm (0.17)$, and for INCEPTION (0.72 ± 0.20). As SI leverages the gradient information that leads to a smaller reduction in detection performance, SI is a more resilient signal to detect concept and severe drift in case of increasing model error rates.

For both covariate and subconcept drift, the class labels are assumed to be unknown. Thus, only distribution-based detectors can be employed. The detection performance of the

[3]https://github.com/myrtakis/EDDI

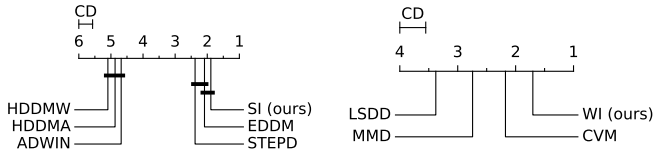
[4]https://github.com/SeldonIO/alibi-detect

[5]https://github.com/IFCA-Advanced-Computing/frouros



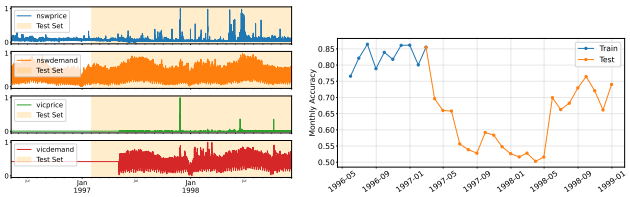
(a) Concept Drift (b) Severe Drift (c) Covariate Drift (d) Subconcept Drift

Fig. 3: Drift detection performance (the higher the better) for the four drift types on the 42 TS datasets. Leveraging the gradient information, the influence signals perform on par with or better than the dedicated detectors in all drift types.



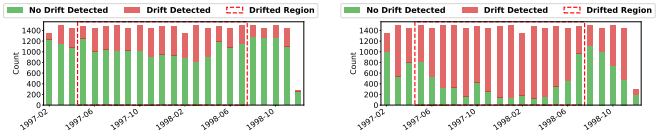
(a) Model Drift (b) Feature Drift

Fig. 4: Critical distance plots for model (concept and severe) and feature (covariate and subconcept) drift detection using all 42 TS datasets. SI and WI rank higher in terms of performance than the dedicated drift detectors.



(a) Numerical features. (b) TST accuracy over time.

Fig. 5: Electricity Dataset



(a) EDDM (b) SI

Fig. 6: Model Drift Detection on Electricity Dataset.

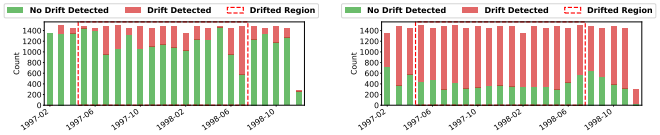
WI signal and the distribution-based detectors for covariate drift is depicted in Fig. 3 (c) and Fig. 3 (d). Note that for feature drifts, the baselines are model independent. As we can see in Fig. 3 (c), WI leads to higher detection performance for covariate drift than all baselines for all the models. WI is more sensitive even to slight changes in the input features distribution. In this scenario, existing detectors achieve detection performance close to or even lower than random. When detecting subconcept drift, shown in Fig. 3 (d), WI achieves 25% higher detection performance for both models on average than covariate drift; the performance of baselines is also substantially higher as input features are changing more drastically. WI performs on par with or better than the baselines, but exhibits a higher standard deviation due to noisier gradient estimates, as it represents the most challenging setting in our experiments.

Since the evaluation involves multiple methods and datasets, we assess their relative ranking using the critical distance diagram. This diagram determines if the difference in average ranks between methods is statistically significant. In Fig. 4, both signals are ranked first, and their rank is statistically significant. This result indicates that SI and WI consistently lead to better detection performance across multiple evaluation scenarios. Instead, it shows that model gradients, as leveraged by SI and WI, provide reliable and sensitive indicators that outperform dedicated detectors for model (Fig. 4a) or feature (Fig. 4b) drift.

B. Detection of Real-World Drift

In this experiment, we perform a qualitative drift detection analysis on the Electricity [38] dataset known to exhibit a drift [40]. This dataset contains 45,312 instances, dated from May 7, 1996, to December 5, 1998, and has 4 numerical features 5a and a binary outcome indicating whether the price was up or down. According to Fig. 5b, the accuracy of a TST model, trained up to 1997-01, substantially drops during the period 1997/05 - 1998/07, which defines the drifted region.

Starting from model drift, EDDM and SI run on sliding windows (of size 10 and stride 1), and the results are depicted in Fig. 6. We selected EDDM due to its performance based on the experimental results of the previous section. SI (Fig. 6b) is more accurate in identifying more windows during the period 1997/05-1998/08 as drifted than EDDM (6a). Regarding feature drift, half of the dataset's features after 1997/05 exhibit a substantial difference w.r.t. the previous values (Fig. 5a). Thus, the expected behavior is to detect the majority of the windows as drifted. As depicted in Fig. 7a, MMD is less sensitive than WI (Fig. 7b), detecting fewer drifted windows. Moreover, WI identifies more windows as drifted after the drifted region. This is aligned with the behavior of the vicdemand and nswrprice features (Fig. 5) whose distribution differs from the training set. The same evaluation methodology was performed for the Insects dataset [39]. The analysis yields a consistent trend with Electricity, hence, due to space limitations, we report in detail only for Electricity.



(a) MMD (b) WI

Fig. 7: Feature Drift Detection on Electricity Dataset.

TABLE I: Execution time to process a single window.

Method	Time (s)
SI	0.09
EDDM	0.064
HDDM-A	0.055
HDDM-W	0.058
ADWIN	0.065
STEPD	0.07

Method	Time (s)
WI	0.41
CVM	0.64
MMD	0.31
LSDD	0.12

C. Drift Explanation

This experiment evaluates the efficacy of the proposed influence-based explanation signals NI and BI to distinguish the different drift types. Since the drift is simulated for each dataset and its type is therefore known, the Kolmogorov–Smirnov (KS) test is applied to each drifted window to compute a p-value, testing the null hypothesis that the influence distributions of NI or BI are identical.

The first signal we evaluate is NI for explaining a model drift. NI is expected to assign lower p-values to severe than to concept drift, as the observed and predicted NI distributions are expected to significantly differ. NI can distinguish the two drift types with an AUC of 0.8 when the learner is TST. However, the ML model's performance impacts the efficacy of the explanation mapping. This is observed when using MOMENT as the learner, where its performance is substantially lower than TST. For MOMENT, the explanation efficacy is close to the random performance. The model's quality is strongly related to its internal representation of the data, as well as how informative the gradients are. In the case of model drift, MOMENT has lower performance than TST, leading to more misclassifications, hence making the *observed* NI distributions of both severe and concept drifting samples look similar and differ from the predicted NI distributions.

The second signal is BI for explaining feature drift, computed over the predicted samples' labels, i.e., the class labels assumed to be unknown. BI is expected to assign higher p-values to covariate than subconcept drift, as the positive-negative influence ratio distribution is expected to resemble the reference set's ratio. BI distinguishes covariate from subconcept drift with an AUC of 0.8 for TST and 0.76 for MOMENT. Even though MOMENT performs poorly, BI is more resilient, using the positive instead of only the negative influence used by NI. Positive influence is less affected by the model's performance for capturing feature changes.

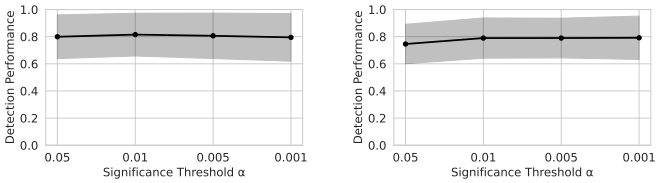
D. Computational Efficiency

The previous experiments verified that the proposed influence signals provide accurate detection and drift type identification. Apart from effectiveness, computational efficiency is equally important in a streaming setting. First, the execution time of each detection method to process a single window of size 10 is measured. We have included the execution time of all baselines and the proposed influence-based signals that are required to process a single window. More precisely, both EDDI and the model drift detectors require a forward pass to obtain the predicted labels. The main difference is that after the forward pass, SI computes the gradients based on the parameters of the last layer. According to Table I, this computation adds a small overhead of 0.03 seconds on average w.r.t. the baselines. WI is more computationally expensive than SI, exhibiting quadratic time complexity instead of linear. WI is 2x faster than CVM and 2.5x slower than MMD and LSDD on average. We should stress that MMD and LSDD share the sample computation complexity with SI. First, the implementation of the former methods is based on optimization tricks such as window subsampling. This optimization can also be employed to accelerate the execution time of WI. Second, to ensure a fair comparison between WI and the feature drift baselines, the forward pass and the gradients were computed on a CPU; employing a GPU is expected to reduce the required time by WI even further. Since the execution time heavily depends on the hardware architecture on which it is deployed, we compare the detection influence signals and IOSI in terms of GFLOPs. FLOPs count is a rough measure of how expensive a model can be, regardless of (i.e., GPUs vs CPUs). To measure the FLOPs, we used PyTorch profiler[6]. For calculating the exact influence, IOSI requires a forward pass over the given samples to obtain the window's loss values, a backward pass for the parameter updates, and then a forward pass to calculate the exact loss influence after updating. This process is repeated for every sample in the window. Instead, SI and WI require a forward pass and gradient computation over the last layer, without applying any parameter update. For a fair comparison with IOSI, the backward pass was performed for 1 epoch using only the window's 10 samples, i.e., not including historical samples. Influence-based signals offer substantial performance gains, reducing the computational overhead for TST by 2.5–5.1x and for MOMENT by 1.5-3x. These results show that influence signals constitute a practical choice for drift detection in streaming settings.

E. Hyper-parameter Ablation Study

EDDI relies on a statistical significance threshold α to determine whether an incoming window is flagged as drifted. To assess the robustness of the proposed method with respect to this parameter, we conduct an ablation study across multiple commonly used significance levels for both model drift and feature drift detection. In Fig. 8 we report the detection performance as a function of α, where the solid line denotes the mean performance and the shaded area indicates the standard deviation across datasets. Both SI (Fig. 8a) and WI (Fig. 8b) achieve consistently high detection performance (approxi-

[6]https://docs.pytorch.org/docs/stable/profiler.html



(a) Model Drift (b) Feature Drift

Fig. 8: Detection performance for model and feature drift detection on varying values of the significance threshold α.

mately 0.8 on average) across a wide range of significance thresholds. These results indicate that the proposed influence signals are robust to the choice of α and do not require careful threshold tuning, supporting the practical applicability of EDDI in real-world streaming scenarios.

F. Lessons Learned

Drift Detection Performance. Distributional differences in influence magnitude information leveraged by SI and WI lead to higher performance than the best error-rate/distribution-based baselines, increasing the performance by 5% for the model and 15% for the feature drift detection.

Drift Explanation Performance. Distributional differences in influence directionality leveraged by NI and BI can classify the type of drift up to 0.8 AUC. However, the negative influence of samples is affected either by label noise or misclassification. Both cases lead to high negative influence scores, making the observed NI distributions look similar for both concept and severe drifting samples. Positive influence is more robust to feature changes. By combining both influence directionalities, as in BI, we are able to better distinguish between covariate from subconcept drift.

Signals Efficiency. Computing the detection signals SI and WI per window requires substantially fewer computations than the exact IOSI influence. The additional cost incurred by the signals is the computation of the last layer's gradients instead of the full backward-forward pass of IOSI. The signals also reduce memory requirements as they do not change the parameters, thus only one model instance must be kept in memory. In contrast, IOSI must keep two model versions, one before and after the parameters update.

Factors Influencing Effectiveness. As influence signals rely on the model's loss gradient information, they depend on the quality of the model used to classify the data. The experiments showed that model and feature drift detection performances do not substantially differ on average between TST and MOMENT. However, as MOMENT misclassifies more samples, NI's performance drops.

VII. Related Work

Research work in the field of concept drift addresses mainly three problems: (i) how to accurately detect a drift (using error-rate-based, data distribution, and multiple hypothesis testing methods) [4]–[7] (ii) how to understand various aspects (what, when, where, how and strength) of a drift [8], and (iii) how to effectively react to drift by adapting related models and data [1], [9]–[12]. To the best of our knowledge, all drift detection algorithms provide a "*when*" explanation by detecting the occurrence time of the drift; other works identify the drifting samples ("*what*" explanation) that contributed to or were mostly affected by the drift. Other approaches aim to identify the features that led to the detected drift ("*where*" explanation) while distribution-based drift detectors can usually assess the magnitude of a drift ("*strength*" explanation). Existing literature is relatively sparse for uncovering how changes in the posterior, class-conditioned, and marginal distributions affect a drift ("*how*" explanations). In this section, we mainly position our work w.r.t. (i) closely related to EDDI drift detection methods exploiting model confidence and loss, and (ii) drift mapping and explanation methods. Readers are referred to various surveys on traditional drift detection methods [6], [53].

A. Confidence & Loss-based Drift Detection

Prior works have explored the detection of drifting samples by directly checking the prediction confidence of the original classifier [55]. A low confidence score could indicate that the incoming window exhibits a drift. However, this confidence score is a probability calculated based on the assumption that all the classes are known (closed-world). A drifting sample from an unknown class may be assigned to the wrong class with high confidence, as reported by previous works [55], [56]. For this reason, [57] proposes to compute a non-conformity measure between the incoming sample and each of the existing classes to determine fitness. This non-conformity measure is calculated based on a distance function to quantify the dissimilarity between samples. However, according to recent studies [58], such distance functions could lose effectiveness, especially with increasing data dimensionality and sparsity.

Loss-based approaches [15], [23], [58]–[60] aim to spot distributional differences between the loss values of current and historical samples. Loss-based approaches can be distinguished into model-unaware and model-aware categories. The former built a drift detector independently of the predictive model. Such drift detectors usually quantify (i) distributional differences on the reconstruction loss of an auto-encoder [60] or (ii) flag as drifted the windows having a large distance on cluster centers formed by minimizing a contrastive loss [15], [58]. The latter [23], [59] leverages the loss function of the predictive (actual) model to detect a drift when the distribution of the loss values between training and testing differs.

EDDI has three key differences from confidence and loss-based methods. First, rather than detecting only one drift family, EDDI can detect all types of drift under a single framework. Second, EDDI estimates the impact of updating the model with the arriving data, identifying distributional influence changes from current to past data, instead of capturing loss and confidence changes between current from past data. Third, unlike EDDI, none of the aforementioned approaches can provide drift-type explanations.

B. Drift Understanding & Explanation

Subsequently, the different explanation methods are organized by the type of explanation they provide to the drift.

"When" drift explanation. All drift detectors aim to identify the time point at which a drift occurred to reveal the temporal nature of a drift as sudden or abrupt, gradual, incremental, or recurrent [12]. In contrast to temporal explanations, EDDI reveals the drift type, i.e., attributing the changes in the posterior, class-conditioned, or marginal distributions.

"What" drift explanation. Recent works are focusing on sample subsets that the model has significantly worse (or better) performance [33], [61]–[63]. These methods aim to localize those parts of a model that need to be adapted, i.e., the parts of the data distribution that are undergoing drift. To retrieve non-drifting regions from suspended historical data, [33] proposes a local drift degree (LDD) measurement that can continuously monitor regional density changes. Instead of suspending all historical data after a drift, regional density discrepancies are synchronized according to LDD. [61] uses decision trees as a surrogate model to capture sub-parts of the input feature distribution where the accuracy of the actual model drops, called DDLAs. Then, the incoming windows are assessed on their similarity to the identified DDLAs. EDDI directly assesses the impact of a window on the predictive model's boundary, bypassing the need for using external models to assess harmful drifting windows. More importantly, it provides a natural ordering of samples within a window based on their influence scores.

Drift "strength" explanation. [14] was the first work that introduced the problem of drift understanding. Four quantitative distance measures of the joint probability distributions between different time points are proposed, including the key measure of drift magnitude. Coarse-grain drift summaries have been refined in order [40] to also account different subspaces of input data features. Unlike these descriptive statistics [64], EDDI explicitly accounts for the impact of drifting samples on the decision boundary of predictive models. Hence, the proposed approach can detect and characterize the type of a drift using exclusively the distributional characteristics of its influence. This enables accurate and efficient detection of fine-grained changes in the underlying distribution of data streams without using parametric tests and true drift labels.

"How" drift explanation. Attributing model performance drops to multiple factors of distributional changes has been introduced in [65]. By defining the value of a set of distributions to be the change in model performance when only this set has changed between (temporal or spatial) environments, the importance weighting method estimates a ratio of densities of an arbitrary set of distributions. The contribution of each distribution to the total performance change is then quantified as its Shapley value. To explain changes in model performance, the proposed attribution method considers model parameters (unlike similar root-cause methods, such as [66]) and attribution signs, which are particularly important since some shifts may decrease loss. However, this work assumes knowledge of the causal graph to factorize the joint distribution and obtain interpretable and succinct attributions. EDDI relaxes this assumption via influence estimators that approximate the extent to which each training sample alters the final model.

"Where" drift explanation. Finally, current eXplainable AI (XAI) methods are essentially limited to explaining static models [8]. In the case of dynamic models, explanations need to be computed from scratch each time the model changes to assess the changes in attribute value contributions based on the model predictions. For example, Shap-ADWIN [67] monitors the Shapley Loss (SL) value [68] for each sample to assess the contribution of each feature to the overall loss. By analyzing the distribution of SL values, the detector can identify when the loss is attributed more to a specific feature, indicating potential drift. [13] proposes an algorithmic scheme for drift explanations that relies on global (e.g., Global Feature Importance, or Discriminative Dimensionality Reduction) and local (e.g., Local Feature Importance, Characteristic Prototypes, Counterfactuals) explanation methods of predictive models trained suitably. However, (supervised) explanation methods require a sufficient sample size to estimate distributions, which is difficult to meet, given that the drifting samples are located in a sparse space outside of the data distribution. To the best of our knowledge, EDDI is the first influence-based method that can characterize the fine-grained drift nature for an informed model adaptation or data cleaning.

VIII. Conclusion

In this work, we introduced a unified influence-based framework for the detection and explanation of major drift types like concept, severe, covariate, and subconcept drift. Through type-specific simulation of drift on various real datasets, we show that the impact of the drifting samples on a model's decision boundary, captured by SI and WI, leads to a more effective detection of model and feature drift than error-rate and distribution-based methods. In addition, the effect of each drift type on the model, captured by NI and BI, allows us to form distinct influence signatures and hence, explain its type. The main limitations of EDDI are the following: (i) due to the TracIn influence, EDDI can work only with gradient-based optimized models; (ii) EDDI performance may be substantially affected by poor-performing models or noisy gradients; (iii) EDDI does not currently detect a shift in class labels.

As future research work, we plan to leverage sampling methods for improving the quadratic complexity of WI (in terms of the sample size of a window). Moreover, we plan to extend EDDI to other predictive tasks, such as regression and forecasting, and drift types (prior distributions). These extensions will potentially require devising novel influence signals. Finally, we plan to study how existing data or model adaptation methods can benefit from the influence-based explanations to automatically decide the best drift mitigation strategy.

Acknowledgment

The work of the first author was partially funded by Honda Research Institute Europe (HRI-EU). This research work has received funding from the Horizon Europe Framework Programme under Grant agreement No 101135775 (PANDORA) and ANR under the Grant agreement 24-CE23-6509 (AIDA).

AI-Generated Content Acknowledgement

Code generation for plotting via Chat GPT was used for figures 6 and 7.

References

[1] A. Mallick, K. Hsieh, B. Arzani, and G. Joshi, "Matchmaker: Data drift mitigation in machine learning for large-scale systems," *Proceedings of Machine Learning and Systems*, vol. 4, pp. 77–94, 2022.

[2] N. Polyzotis, S. Roy, S. E. Whang, and M. Zinkevich, "Data management challenges in production machine learning," in *Proceedings of the 2017 ACM International Conference on Management of Data*, 2017.

[3] A. Subbaswamy, B. Chen, and S. Saria, "A unifying causal framework for analyzing dataset shift-stable learning algorithms," *Journal of Causal Inference*, vol. 10, no. 1, pp. 64–89, 2022. [Online]. Available: https://doi.org/10.1515/jci-2021-0042

[4] F. Hinder, V. Vaquet, and B. Hammer, "One or two things we know about concept drift—a survey on monitoring in evolving environments. part a: detecting concept drift," *Frontiers in Artificial Intelligence*, vol. Volume 7 - 2024, 2024.

[5] E. Werner, N. Kumar, M. Lieber, S. Torge, S. Gumhold, and W. E. Nagel, "Towards computational performance engineering for unsupervised concept drift detection—complexities, benchmarking, performance analysis," *arXiv preprint arXiv:2304.08319*, 2024.

[6] F. Bayram, B. S. Ahmed, and A. Kassler, "From concept drift to model degradation: An overview on performance-aware drift detectors," *Knowledge-Based Systems*, vol. 245, p. 108632, 2022.

[7] S. Agrahari and A. K. Singh, "Concept drift detection in data stream mining: A literature review," *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 10, pp. 9523–9540, 2022.

[8] F. Hinder, V. Vaquet, and B. Hammer, "One or two things we know about concept drift-a survey on monitoring in evolving environments. part b: Locating and explaining concept drift," *Frontiers in Artificial Intelligence*, vol. 7, p. 1330258, 2024.

[9] J. Li, H. Yu, Z. Zhang, X. Luo, and S. Xie, "Concept drift adaptation by exploiting drift type," *ACM Trans. Knowl. Discov. Data*, vol. 18, no. 4, Feb. 2024.

[10] L. Yuan, H. Li, B. Xia, C. Gao, M. Liu, W. Yuan, and X. You, "Recent advances in concept drift adaptation methods for deep learning." in *IJCAI*, 2022, pp. 5654–5661.

[11] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang, "Learning under concept drift: A review," *IEEE transactions on knowledge and data engineering*, 2018.

[12] J. a. Gama, I. Žliobaitundefined, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Comput. Surv.*, vol. 46, no. 4, Mar. 2014.

[13] F. Hinder, V. Vaquet, J. Brinkrolf, and B. Hammer, "Model-based explanations of concept drift," *Neurocomputing*, vol. 555, p. 126640, 2023.

[14] G. I. Webb, R. Hyde, H. Cao, H. L. Nguyen, and F. Petitjean, "Characterizing concept drift," *Data Mining and Knowledge Discovery*, vol. 30, no. 4, pp. 964–994, 2016.

[15] A. Castellani, S. Schmitt, and B. Hammer, "Task-sensitive concept drift detector with constraint embedding," in *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2021, pp. 01–08.

[16] R. Müller, M. Abdelaal, and D. Stjelja, "Open-source drift detection tools in action: Insights from two use cases," in *Big Data Analytics and Knowledge Discovery: 26th International Conference, DaWaK 2024, Naples, Italy, August 26–28, 2024, Proceedings*. Berlin, Heidelberg: Springer-Verlag, 2024, p. 346–352.

[17] R. Cabañas, A. Cano, M. Gómez-Olmedo, A. R. Masegosa, and S. Moral, "Virtual subconcept drift detection in discrete data using probabilistic graphical models," in *Information Processing and Management of Uncertainty in Knowledge-Based Systems.* Springer, 2018, pp. 616–628.

[18] F. Maia Polo and R. Vicente, "Effective sample size, dimensionality, and generalization in covariate shift adaptation," *Neural Comput. Appl.*, 2022.

[19] Ł. Korycki and B. Krawczyk, "Adversarial concept drift detection under poisoning attacks for robust data stream mining," *Machine Learning*, vol. 112, no. 10, pp. 4013–4048, 2023.

[20] M. Olteanu, F. Rossi, and F. Yger, "Challenges in anomaly and change point detection," *arXiv preprint arXiv:2212.13520*, 2022.

[21] N. Myrtakis, I. Tsamardinos, and V. Christophides, "Data glitches discovery using influence-based model explanations," in *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.1*, 2025.

[22] F. Hinder, A. Artelt, and B. Hammer, "A probability theoretic approach to drifting data in continuous time domains," *arXiv preprint arXiv:1912.01969*, 2019.

[23] F. Hinder, V. Vaquet, J. Brinkrolf, and B. Hammer, "On the change of decision boundary and loss in learning with concept drift," in *International Symposium on Intelligent Data Analysis*. Springer, 2023, pp. 182–194.

[24] S. J. Delany, P. Cunningham, A. Tsymbal, and L. Coyle, "A case-based technique for tracking concept drift in spam filtering," in *International Conference on Innovative Techniques and Applications of Artificial Intelligence*. Springer, 2004, pp. 3–16.

[25] ——, "A case-based technique for tracking concept drift in spam filtering," *Knowledge-Based Systems*, vol. 18, no. 4, pp. 187–195, 2005, aI-2004, Cambridge, England, 13th-15th December 2004.

[26] A. Gretton, A. Smola, J. Huang, M. Schmittfull, K. Borgwardt, and B. Schölkopf, *Covariate shift and local learning by distribution matching*. Cambridge, MA, USA: MIT Press, 2009, pp. 131–160.

[27] P. W. Koh and P. Liang, "Understanding black-box predictions via influence functions," in *International conference on machine learning*. PMLR, 2017, pp. 1885–1894.

[28] M. Xia and R. Henao, "Reliable active learning via influence functions," *Transactions on Machine Learning Research*, 2023.

[29] G. Pruthi, F. Liu, S. Kale, and M. Sundararajan, "Estimating training data influence by tracing gradient descent," *Advances in Neural Information Processing Systems*, vol. 33, pp. 19 920–19 930, 2020.

[30] J. C. Sisniega, V. Rodríguez, G. Moltó, and Á. L. García, "Efficient and scalable covariate drift detection in machine learning systems with serverless computing," *Future Generation Computer Systems*, vol. 161, pp. 174–188, 2024.

[31] P. Sobhani and H. Beigy, "New drift detection method for data streams," in *Adaptive and Intelligent Systems: Second International Conference, ICAIS 2011, Klagenfurt, Austria, September 6-8, 2011. Proceedings*. Springer, 2011, pp. 88–97.

[32] L. Bu, C. Alippi, and D. Zhao, "A pdf-free change detection test based on density difference estimation," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 2, pp. 324–334, 2018.

[33] A. Liu, Y. Song, G. Zhang, and J. Lu, "Regional concept drift detection and density synchronized drift adaptation," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, ser. IJCAI'17. AAAI Press, 2017, p. 2280–2286.

[34] G. Zerveas, S. Jayaraman, D. Patel, A. Bhamidipaty, and C. Eickhoff, "A transformer-based framework for multivariate time series representation learning," in *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, 2021, pp. 2114–2124.

[35] M. Goswami, K. Szafer, A. Choudhry, Y. Cai, S. Li, and A. Dubrawski, "MOMENT: A family of open time-series foundation models," in *Proceedings of the 41st International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, R. Salakhutdinov, Z. Kolter, K. Heller, A. Weller, N. Oliver, J. Scarlett, and F. Berkenkamp, Eds., vol. 235. PMLR, 21–27 Jul 2024, pp. 16 115–16 152.

[36] H. Ismail Fawaz, B. Lucas, G. Forestier, C. Pelletier, D. F. Schmidt, J. Weber, G. I. Webb, L. Idoumghar, P.-A. Muller, and F. Petitjean, "Inceptiontime: Finding alexnet for time series classification," *Data Mining and Knowledge Discovery*, vol. 34, no. 6, pp. 1936–1962, 2020.

[37] D. Dheeru and E. Karra Taniskidou, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[38] M. Harries, N. S. Wales *et al.*, "Splice-2 comparative evaluation: Electricity pricing," 1999.

[39] V. M. Souza, D. M. dos Reis, A. G. Maletzke, and G. E. Batista, "Challenges in benchmarking stream learning algorithms with real-world data," *Data Mining and Knowledge Discovery*, vol. 34, no. 6, pp. 1805–1858, 2020.

[40] G. I. Webb, L. K. Lee, B. Goethals, and F. Petitjean, "Analyzing concept drift and shift from sample data," *Data Mining and Knowledge Discovery*, vol. 32, pp. 1179–1199, 2018.

[41] M. Baena-Garcıa, J. del Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavalda, and R. Morales-Bueno, "Early drift detection method," in *Fourth international workshop on knowledge discovery from data streams*, vol. 6. Citeseer, 2006, pp. 77–86.

[42] A. Bifet and R. Gavalda, "Learning from time-changing data with adaptive windowing," in *Proceedings of the 2007 SIAM international conference on data mining*. SIAM, 2007, pp. 443–448.

[43] K. Nishida and K. Yamauchi, "Detecting concept drift using statistical testing," in *International conference on discovery science*. Springer, 2007, pp. 264–269.

[44] I. Frias-Blanco, J. del Campo-Ávila, G. Ramos-Jimenez, R. Morales-Bueno, A. Ortiz-Diaz, and Y. Caballero-Mota, "Online and non-parametric drift detection methods based on hoeffding's bounds," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 3, pp. 810–823, 2014.

[45] L. Bu, C. Alippi, and D. Zhao, "A pdf-free change detection test based on density difference estimation," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 2, pp. 324–334, 2016.

[46] D. M. Dos Reis, P. Flach, S. Matwin, and G. Batista, "Fast unsupervised online drift detection using incremental kolmogorov-smirnov test," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1545–1554.

[47] E. Oikarinen, H. Tiittanen, A. Henelius, and K. Puolamäki, "Detecting virtual concept drift of regressors without ground truth values," *Data Mining and Knowledge Discovery*, vol. 35, no. 3, pp. 726–747, 2021.

[48] Z. Pang, J. Cen, and M. Yi, "Unsupervised concept drift detection method based on robust random cut forest," *International Journal of Machine Learning and Cybernetics*, vol. 14, no. 12, pp. 4207–4222, 2023.

[49] T. S. Sethi and M. Kantardzic, "On the reliable detection of concept drift from streaming unlabeled data," *Expert Systems with Applications*, vol. 82, pp. 77–99, 2017.

[50] N. Myrtakis, "Data debugging for machine learning," Ph.D. dissertation, 2025, thèse de doctorat dirigée par Christophides, Vassilis et Tsamardinos, Ioannis Sciences et Technologies de l'Information et de la Communication CY Cergy Paris Université 2025. [Online]. Available: http://www.theses.fr/2025CYUN1393

[51] P. M. Gonçalves Jr, S. G. de Carvalho Santos, R. S. Barros, and D. C. Vieira, "A comparative study on concept drift detectors," *Expert Systems with Applications*, vol. 41, no. 18, pp. 8144–8156, 2014.

[52] S. Greco, B. Vacchetti, D. Apiletti, and T. Cerquitelli, "Unsupervised concept drift detection from deep learning representations in real-time," *IEEE Transactions on Knowledge and Data Engineering*, 2025.

[53] D. Lukats, O. Zielinski, A. Hahn, and F. Stahl, "A benchmark and survey of fully unsupervised concept drift detectors on real-world data streams," *International Journal of Data Science and Analytics*, vol. 19, no. 1, pp. 1–31, 2025.

[54] J. Gama, R. Sebastiao, and P. P. Rodrigues, "On evaluating stream learning algorithms," *Machine learning*, vol. 90, pp. 317–346, 2013.

[55] D. Hendrycks and K. Gimpel, "A baseline for detecting misclassified and out-of-distribution examples in neural networks," in *International Conference on Learning Representations*, 2017. [Online]. Available: https://openreview.net/forum?id=Hkg4TI9xl

[56] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[57] R. Jordaney, K. Sharad, S. K. Dash, Z. Wang, D. Papini, I. Nouretdinov, and L. Cavallaro, "Transcend: detecting concept drift in malware classification models," in *Proceedings of the 26th USENIX Conference on Security Symposium*, ser. SEC'17. USA: USENIX Association, 2017, p. 625–642.

[58] L. Yang, W. Guo, Q. Hao, A. Ciptadi, A. Ahmadzadeh, X. Xing, and G. Wang, "CADE: Detecting and explaining concept drift samples for security applications," in *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, Aug. 2021, pp. 2327–2344.

[59] F. Hinder, V. Vaquet, J. Brinkrolf, and B. Hammer, "On the hardness and necessity of supervised concept drift detection," in *Proceedings of the 12th International Conference on Pattern Recognition Applications and Methods, ICPRAM*, 2023, pp. 164–175.

[60] U. Ali and T. Mahmood, "A novel framework for concept drift detection using autoencoders for classification problems in data streams," *International Journal of Machine Learning and Cybernetics*, vol. 16, no. 1, pp. 397–418, 2025.

[61] S. Dong, Q. Wang, S. Sahri, T. Palpanas, and D. Srivastava, "Efficiently mitigating the impact of data drift on machine learning pipelines," *Proceedings of the VLDB Endowment*, vol. 17, no. 11, pp. 3072–3081, 2024.

[62] S. Jain, H. Lawrence, A. Moitra, and A. Madry, "Distilling model failures as directions in latent space," in *The 11th International Conference on Learning Representations*, 2023.

[63] A. Ali, M. Cauchois, and J. C. Duchi, "The lifecycle of a statistical model: Model failure detection, identification, and refitting," 2022. [Online]. Available: https://arxiv.org/abs/2202.04166

[64] I. Goldenberg and G. I. Webb, "Survey of distance measures for quantifying concept drift and shift in numeric data," *Knowledge and Information Systems*, vol. 60, no. 2, pp. 591–615, 2019.

[65] H. Zhang, H. Singh, M. Ghassemi, and S. Joshi, ""why did the model fail?": attributing model performance changes to distribution shifts," in *Proceedings of the 40th International Conference on Machine Learning*, ser. ICML'23. JMLR.org, 2023.

[66] K. Budhathoki, D. Janzing, P. Bloebaum, and H. Ng, "Why did the distribution change?" in *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, A. Banerjee and K. Fukumizu, Eds., vol. 130. PMLR, 13–15 Apr 2021, pp. 1666–1674.

[67] B. Zimmermann and M. Boussard, "Improving drift detection by monitoring shapley loss values," in *ICPRAI*, 2022.

[68] H. Chen, S. M. Lundberg, and S. Lee, "Explaining models by propagating shapley values of local components," *CoRR*, vol. abs/1911.11888, 2019.