RegDriftKit: A Toolkit for Generating Data and Benchmarking Drift Detection in Regression Tasks

Oz Kilic
Carleton University
Canada
ozkilic@cmail.carleton.ca

Justin Charbonneau
IFS Canada Inc.
Canada
justin.charbonneau@ifs.com

Elio Velazquez
IFS Canada Inc.
Canada
elio.velazquez@ifs.com

Olga Baysal
Carleton University
Canada
olga.baysal@carleton.ca

Abstract

Production machine learning models rarely maintain their performance over time, requiring frequent retraining due to drifts. However, optimizing and benchmarking drift detectors is a challenging process, and drift detection is underexplored in regression scenarios compared to classification. We introduce our regression-focused toolkit, *RegDriftKit*, a modular and extensible Python package for drift detection experimentation. RegDriftKit provides utilities and pipelines for richly annotated dataset generation or emulation with controllable data and concept drifts, drift detection, and benchmarking. Our experiments show that RegDriftKit can reliably automate data drift tuning across a wide range of numerical distributions and diverse drift intensities in scenarios with consistent and sufficiently large data flows. We also propose additional rules to improve detection accuracy evaluation in edge cases or with non-sudden drifts, and share our source code and example drift-annotated datasets that can be used to benchmark data drift detection methods.

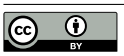
CCS Concepts

• **Software and its engineering** → **Object oriented frameworks**; • **General and reference** → **Evaluation**; **Experimentation**; • **Computing methodologies** → **Supervised learning by regression**; *Discrete-event simulation*; • **Information systems** → *Data stream mining*.

Keywords

Data drift, concept drift, drift detection, regression, synthetic data generation, toolkit, benchmark

ACM Reference Format:
Oz Kilic, Justin Charbonneau, Elio Velazquez, and Olga Baysal. 2026. RegDriftKit: A Toolkit for Generating Data and Benchmarking Drift Detection in Regression Tasks. In *2026 IEEE/ACM 5th International Conference on AI Engineering - Software Engineering for AI (CAIN '26), April 12–13, 2026, Rio de Janeiro, Brazil.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3793653.3793769



CAIN '26, Rio de Janeiro, Brazil

ACM ISBN 979-8-4007-2475-6/2026/04
https://doi.org/10.1145/3793653.3793769

1 Introduction

In supervised machine learning (ML), model performance is typically evaluated using a test set that is not used to train the model, allowing ML practitioners to infer the performance in the production environment. However, in real-life scenarios, it is uncommon for models to maintain their evaluated performance levels over longer periods of time. Baier et al. [6] found that only 9% of the ML practitioners interviewed indicated that their ML models adapt to data drifts, suggesting drift detection and adaptation are still a challenge in the industry.

Despite regression forming a significant portion of machine learning problems [5], drift detection and drift adaptation are underexplored for regression [5, 7, 38, 68]. Studies that focused on drift in regression commonly focused on adaptation without explicit detection, possibly due to the challenging and subjective nature of error quantification in regression (what is a significant error?) and the lack of regression datasets with drift annotations in the literature. However, the performance of the model itself is not a sufficient metric for drift detection [3], and evaluating the detection performance can produce more robust and explainable methods. Thus, there is a need for a study that focuses on regression, uses drift-annotated regression datasets, and evaluates drift detection independently of model performance.

Real datasets lack the drift ground truth that is needed to obtain drift detection metrics, which necessitates using synthetic datasets. Existing data synthesis methods are largely focused on realism and privacy preservation, with little consideration for drifts, especially for regression tasks. Moreover, relevant synthetic datasets and generators commonly focus on concept drift rather than data drift and classification rather than regression [38]. For drift detection and adaptation research, there is a need for dataset generation methods that allow highly controllable data and concept drift generation.

This research is conducted in collaboration with IFS Canada Inc. IFS Canada develops enterprise software solutions, such as resource planning, asset management, and field service management, for customers from diverse industries such as aerospace, defense, energy, construction, manufacturing, and telecommunications.

IFS Canada aimed to incorporate drift detection into its MLOps workflow to promote reliable and explainable machine learning, with a focus on data drifts in univariate regression scenarios and tabular data. Consequently, this study focuses on developing a toolkit

for ML practitioners to study, simulate, and automate drift detection for research and potential workflow integration. The toolkit is developed to be used within a no-code platform, enabling non-technical, internal domain experts to generate synthetic datasets resembling customer data and evaluate drift detection and retraining strategies to help decision-making. The *research objectives* are as follows:

- Developing a synthetic data generator capable of inducing data or concept drifts with various temporal patterns for regression problems.
- Generating synthetic regression datasets with annotated ground truth for drift occurrence and using them to evaluate and compare the effectiveness of different data drift detection methods under various conditions.
- Developing a drift detection rather than adaptation in regression settings.
- Design of a modular and extensible framework for drift detection and benchmarking.

In this study, we address the following *research questions*:

RQ1: To what extent can we generate synthetic datasets with highly controllable data drifts?
Since reliably evaluating detection methods requires (or assumes) that ground truth drifts are known, our approach depends on synthetic data generated with precise data drifts. At the same time, there is a need for a method that can be used without in-depth statistical knowledge that would facilitate automated dataset generation and data drift detection evaluation. We design an experiment to evaluate our smart data drift generation approach.

RQ2: How can the performance of data drift detection methods be reliably assessed without using proxy-based metrics?
Data drifts in regression scenarios are rarely explored in the literature and are mostly addressed by adaptation rather than explicit detection. With a focus on detection for reliability and explainability, there is a need for detection metrics and pipelines to evaluate the detection performance of different methods, instead of using proxy metrics such as prediction performance. Following our synthetic data and drift generation method and our data drift detection evaluation pipeline, we designed an experiment to demonstrate how different methods can be benchmarked.

To the best of our knowledge, RegDriftKit is the first drift-focused solution for regression scenarios that allows (1) fitting a generator to a reference dataset, (2) annotated dataset generation with drift ground truths and extensive metadata, (3) precise data and concept drift generation, and (4) reliable yet automated and declarative data drift tuning without deep statistical knowledge.

2 Background and Related Work

In this section, we provide background information regarding popular drift categories and briefly cover related work on drift detection tools, datasets/data generators, and data imitation methods.

2.1 Drift Types

Model drifts indicate a significant performance loss of a trained model, either due to encountering input values outside the known and modeled distributions or changes in the underlying concept.

Data drifts, also known as "covariate shifts" [62], correspond to changes in the newly observed input distribution compared to the training set, with some varying interpretations. Multiple studies [34, 35, 49] also used the term "*virtual drift*" or "virtual concept drift" for a change in the data, while some others explicitly referred to changes in the data without a change in the concept [53]. Tsymbal [78] referred to changes in the data that require a change in the model, requiring a model drift for both data and concept drifts. Microsoft used "virtual drifts" to indicate data or target drifts that do not cause a change in the concept [69]. Lesort et al. [48] also explained that "virtual drifts" can be interpreted as data drifts with neither a concept nor a target drift, also known as "domain drifts." While some studies explicitly included the term "concept" in "virtual *concept* drift" [35, 68], other studies simply used the term "virtual drift" [23, 34, 48, 49, 53, 88], without a clear pattern.

Concept drift (or "dataset/concept shift" [53, 62, 87]) is a change in the joint probability distribution of a set of inputs and the target variable [5], altering the underlying input-to-target mapping. There is a lack of consensus on what counts as a concept drift [7]. Some studies categorized data drifts under concept drift using the term "virtual concept drift" [34, 52–54, 90].

We define any input variable change as **data drift**, and changes in the input-target relationship as **concept drift**.

2.2 Temporal Patterns of Drifts

Drift occurrences are commonly categorized based on their temporal characteristics. *Sudden/abrupt drifts* indicate a change that occurs immediately, which is usually easier to detect [4]. *Gradual drifts* occur when the frequency of observing instances with drift increases over time and ultimately completely takes over the previous concept or distribution, while *incremental drifts* occur when the new distribution or concept drifts away with intermediate steps [34]. Gradual and incremental drifts can be harder to detect over a short period of time and can cause significant model drift [70]. *Reoccurring drifts*[1] occur when the initial concept or distribution is observed after some time from drift occurrence, potentially indicating seasonality effects.

2.3 Detection Tools

There are multiple tools for drift detection, originating from both industry and academia, such as *Evidently* [30], *Frouros* [21], *Alibi Detect* [84], *Deepchecks* [18], *Menelaus* [59], *NannyML* [63], *River* [61], and *Google Cloud*'s new model monitoring service [36].

In our study, we implemented detection classes that work with Evidently and Frouros libraries due to their maturity, active development, adoption, and permissive license. Evidently is a mature framework for evaluating, testing, and monitoring machine

[1] In the literature, reoccurring drifts are also commonly referred to as "recurrent" and "recurring" drifts. While multiple studies used both terms interchangeably [2, 27, 60], we explicitly preferred the term "reoccurring" due to "recurrent" implying seasonality and predictability, which are not actually guaranteed for drifts. To clarify the difference, Costa et al. [19] differentiated the two types by defining them as "recurrent cyclical" and "recurrent acyclical" drifts.

learning models, offering extensive visualization and data-quality metrics. However, its drift detection is limited to univariate methods applied per feature, and it favors custom test functions over extensible monitor classes. Frouros focuses solely on drift detection algorithms, covering both univariate and multivariate data and concept drifts. It provides a wide range of implementations and has been used in major European Union research projects [1, 42].

2.4 Datasets and Data Generators

Most of the popular real and synthetic datasets used in drift detection are classification-oriented [10, 12, 24, 29, 33, 37, 40, 43–45, 66, 71–73, 76, 79, 80]. The state-of-the-art offers fewer real, regression-oriented datasets [22, 31, 46, 50, 57, 65, 81, 82], featuring a mixture of numerical, categorical, temporal, and Boolean input variables. However, real datasets naturally do not include or allow the extraction of drift ground truth. *Turing Change Point Detection Benchmark* [83] provides 37 different time-series from different domains that can be used to evaluate change point detection algorithms. To address the lack of ground truth related to when changes occur, manual change point annotations for each time series for each dataset. However, only four of its datasets are multivariate, and its emphasis on time-series data does not suit our study's focus on general, non-time-series regression problems.

While there are some synthetic regression data generators that have drift generation capabilities, customizability is very limited. Friedman data generator [41] changes a portion or the whole function used to obtain the target variable in different manners to produce sudden/gradual local/global concept drifts. Wang et al. [86] use a linear regression formula with some added noise for target variable generation, and switch the active preset (coefficients of variables) to generate sudden and reoccurring drifts. While [61] provides popular real and synthetic datasets that can be used to evaluate drift detection methods, only one of its data generators has a regression focus and explicit drift generation capabilities (implemented based on Ikonomovska et al. [41]). None of the mentioned examples provides support for customizable data or concept drift generation. A recently proposed tool, *SiD²Re* [38], is designed for the generation of synthetic datasets with data and concept drifts, and model-dependent features through Cholesky decomposition [8]. However, its data generation is limited to uniform, constant, Gaussian, and periodic features. Furthermore, its data drift approach focuses on seasonality-like temporal behaviors, not allowing precise and parameter-level drift generation.

Drift-enabled synthetic dataset generation tools commonly induce drifts by manipulating instance classes rather than inducing data drifts through feature manipulations [55]. Lewis et al. [49] proposed a toolset, Augur, that can be used to simulate and detect drifts with modules for dataset generation, model training, and inference. Its dataset generation module allows for inducing data drifts in different temporal patterns. However, Augur was designed for and tested with classification problems, creating drifts by manipulating target class occurrences, and requires significant changes to be used for regression. While Lukats et al. [55] addressed the limitations of class distribution manipulation by swapping the underlying function used to generate data, they did not provide the drift ground truth, their simulated drifts were always abrupt, and they focused on classification, like many other studies.

2.5 Data Imitation

Synthetic dataset generation from a reference dataset (data imitation) largely focuses on realism or privacy-preservation, and can be summarized under *deep-learning-based* and *standard* (non-deep-learning-based) methods [32]. While deep-learning-based approaches (especially Generative Adversarial Networks) are popular and found to yield more realistic results, they are also more complex and less interpretable due to their black-box nature, which hinders controlled drift generation.

Among standard methods, Zhang et al. [89] employed Bayesian networks to model a given dataset as a set of conditional distributions, focusing on privacy preservation. Nowok et al. [67] proposed sequentially generating variables to preserve dependencies, but its order dependence reduces controllability and offers limited realism when modeling complex feature relationships. Copula-based methods [75] model features and feature dependencies separately, but traditional copulas struggle with high-dimensional relationships [39], and elliptical copulas have limited flexibility [77]. Vine copulas improve scalability by modeling dependencies as cascades of bivariate copulas [13], but their exponentially growing complexity requires optimized configuration [77] and manipulating one marginal cascades the entire joint distribution, making it difficult to induce controlled and isolated drifts. Overall, existing methods trade univariate drift ground truth at the marginal level for added realism and do not consider drift generation.

3 Toolkit Overview

RegDriftKit's primary functions are to generate annotated regression data sets, provide a unified interface for drift monitoring using different algorithms and libraries, and benchmark drift monitors. Its top-level packages are defined as follows:

- **Generator**: Classes and functions for dataset and drift generation, statistical distribution classes, drift tuning, and dataset fitting.
- **Data Monitor**: Data drift detection classes (also called "data monitors") and functions to instantiate or optimize them.
- **Model Monitor**: A model drift detection class (also called "model monitor") and functions to instantiate it (or future model monitor implementations).
- **Benchmark**: A benchmark dataset class that encapsulates objects and metadata relevant to a generated dataset and functions to evaluate data monitors using such datasets.
- **Pipeline**: Pipeline classes that can be run from a terminal.
- **Utilities**: Various utility modules grouped under different subpackages (collections, file operations, statistical distributions, data generation, monitoring, persistence, and visualization). These are used by the core packages.
- **ML**: Helper functions to train and optimize different types of regression models that are used in benchmarking.

Figure 1 illustrates two key pipelines of the toolkit: one for generating benchmark datasets (that can mirror a reference dataset), and another one for evaluating data monitors with configurable settings.

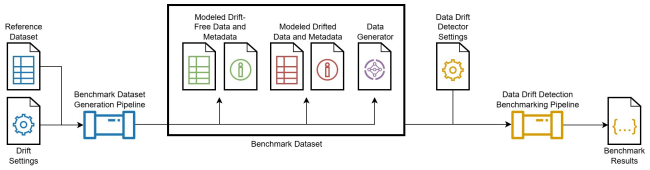


Figure 1: Dataset generation and data drift detector benchmarking pipelines

3.1 Synthetic Dataset Generation

Evaluating drift detection is more difficult in regression compared to classification, where a prediction can be deemed correct or not. Obtaining ground truths for the drift occurrences is therefore crucial, especially with non-sudden drifts where the shift occurs over a period of time. Real datasets naturally do not include the drift ground truth due to the complex and unknown relationships. While sudden changes can be retroactively identified, identifying the ground truth becomes particularly challenging for gradual and non-sudden drifts [26].

Due to not having the complications observed in real data and all drifts occurring under control, synthetic datasets are more suitable for drift detection evaluation. However, many existing synthetic generators focus on classification or have limited customizability. Considering these limitations, we developed a generalizable synthetic data generator with highly customizable drift synthesis capabilities.

We implemented a data generator, *LinearRegressionDrifter*, that acts as a discrete event simulator to sample input and target variables and induce data and/or concept drifts on demand or a defined schedule, generating new data in batches or as a stream. Each feature is modeled as an independent marginal distribution, allowing controlled feature-level drifts. The toolkit supports SciPy [74] distribution and custom distribution classes (such as the included example Gaussian mixture custom class) that follow the SciPy distributions' API. The target variable is modeled through a linear regression formula, where each input variable is an independent variable, and optional noise is added. To avoid feature range imbalance, feature values are standardized under the hood before their target variables are obtained.

Drift options are specified through specific data classes, which define drift timing (manual, automatic, or semi-automatic), occurrence interval, pattern (sudden, gradual, incremental, or reoccurring), and amount. Concept drifts modify regression coefficients or the intercept, and coefficients can be explicitly specified through absolute or relative drift expressions, along with other options such as sampling drifting variables or allowing cumulative and random-direction drifts. Data drifts change feature distributions or parameters used to create the distributions using drift expressions. Due to the discrete nature of the simulation, gradual and incremental drifts are converted into a series of intermediary sudden drifts (using the drift expression, pattern steps, and the drift amount) and put into drift queues to be processed over time. Drift options can be specified during or after initialization of the data generator.

To reduce the statistical knowledge required for specifying drift magnitude and to support no-code usability of the toolkit, we implemented a smart data drift generation feature that automatically

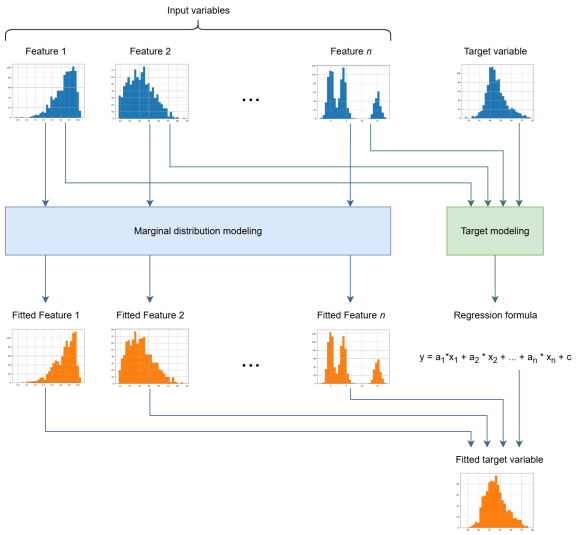


Figure 2: Dataset fitting process that models marginal distributions and the target variable to generate a dataset that mimics the given reference.

tunes distribution parameter changes. Users provide the desired p-value between two batches generated before and after the drift, and the toolkit optimizes the new parameter values by first heuristically defining bounded search spaces based on the distribution parameter types. Different rules are applied for location, standard deviation, rate, scale, shape, and probability. For some distributions, such as binomial and negative binomial, the number of trial/success parameters is kept intact to preserve the originally observed range of discrete values. The toolkit then applies Bayesian optimization to match the target p-value using the Cramér–von Mises test [20, 85].

The generator can also fit itself to a reference dataset, estimating marginal distributions and a regression model to reproduce realistic characteristics. A set of suitable candidate distribution objects is fitted to each marginal distribution and selected using AIC, AIC_c, or BIC. It currently supports 20 SciPy distributions and the custom Gaussian mixture distribution for fitting.

Generated datasets and metadata are encapsulated in benchmark dataset objects that store inputs, targets, metadata, and the generator object for reproducibility and visualization. A benchmark dataset can be saved as a folder and loaded back from it. It also includes some visualization utilities to see the value, parameter, or concept changes in the dataset over time.

3.2 A Unifying Approach for Drift Detection

Our toolkit has a unifying approach to drift detection in two ways. While there are multiple drift detection libraries (some having a considerable overlap of methods), there is no API standard, which prevents drop-in replacement or easily combining methods from different libraries. In RegDriftKit, we implemented wrapper data drift detectors (data monitors) that support batch-based detection methods from Evidently and Frouros libraries (35 methods in total). These wrappers standardize data ingestion, detection output,

and initialization (with some necessary diversion) under a unified interface, allowing users to switch between detector instances or combine methods from different libraries using the same API. They encapsulate preprocessing, drift checking, and postprocessing through common methods while delegating the detection logic to the wrapped library. Both monitors support Boolean signaling with custom thresholds, preprocessing/postprocessing, comparison context management, and metadata output, enabling consistent integration with the toolkit and other pipelines. Our Evidently wrapper also supports using multiple univariate detection methods and combining their signals, and drift detection report generation.

A common design pattern in drift detection frameworks is the separation between offline (batch) and online (streaming) monitoring. For instance, Alibi Detect and Frouros implement the same methods (e.g., Kolmogorov–Smirnov, Maximum Mean Discrepancy, Cramér–von Mises) multiple times, one per paradigm. However, both compare new and historical data, and the two approaches start to converge in drift detection. It is still necessary to periodically update or prune the reference dataset in batch processing to ensure its relevance, and some stream processing methods batch incoming data first. A unified, hybrid architecture could therefore support both scheduled and ad-hoc detection, accommodating diverse ingestion modes within a single framework. We implemented an abstract class that manages data ingestion, comparison window (cache), detection scheduling, and other operations that can be decoupled from detection logic. The cache is an adaptive window inspired by ADWIN [9] due to its popularity and influence in the literature [53], robustness to non-sudden drifts [9], and ease of implementation. Each newly observed instance is appended to the cache, and the cache grows ("adaptive windowing") until a drift is detected or the cache limit is reached. When the cache hits a user-defined length or space limitation after a new observation, the oldest instances (or the oldest group of instances if group atomicity is preferred) are dropped. Having a fixed reference set or a fixed moving window for comparison is also supported. This approach facilitates stream compatibility for batch-oriented methods while allowing focus on detection logic. Our wrappers inherit these base functionalities and also provide a heuristic candidate cut-off date selection method from Bifet and Gavalda [9], which can be included in custom data monitors through a reusable utility function.

We currently provide one model monitor class for model drift detection. It evaluates a given model on a validation dataset using a user-defined performance metric and returns a drift signal. If specified, the signal can be Boolean based on the user-defined metric threshold. The model monitor shares the same configuration and initialization logic as data monitors. It also has a similar API, but does not have some of the more advanced features, such as context management.

3.3 Detection Evaluation

Previous studies, even the ones that focused on drift detection rather than drift adaptation, commonly used proxy metrics such as MAE, RMSE, SMAPE, for regression [5, 25] and average class accuracy, average classification error rate, average classification accuracy, for classification [26, 51, 64]. These metrics evaluate not the detection performance itself but rather the prediction performance of an ML model or ensemble that is altered by the proposed method, providing a comparison between the proposed and alternative methods or the baseline. However, such metrics offer only a partial perspective, as they typically assess overall drift-handling performance without isolating individual components of the process. As a result, a model may appear to perform well while still lacking clarity or accuracy in explaining the underlying causes of the drift. For example, in a high-volume data scenario, a detection method with high recall but low precision can improve model accuracy while producing frequent false alarms, reducing interpretability and efficiency in practice.

For detection evaluation, we use the *Mean Time Ratio* (MTR) metric adapted from Bifet et al. [11] and used by other studies [15–17, 28], defined in Equation 1 as

$$MTR = \frac{TFP}{TTD} \times DR \tag{1}$$

where MTR is calculated by the mean time between false positives (TFP) divided by the mean time to detect (TTD), multiplied by the detection rate (DR). Since an ideal detection method is both fast and accurate — two goals often in trade-off — MTR evaluates both dimensions given a sequence of ground-truth drift events and corresponding monitor signals. TTD is calculated using the average difference between the times a ground truth drift occurs and the monitor detects it, while DR is calculated by dividing the true positive count by the number of ground-truth drift events. While this metric was originally proposed for concept drift detection, it is also suitable for data drift ground truths and Boolean signals. We also report TTD, TFP, DR, TPR (true positive rate), FPR (false positive rate), and F1 score metrics.

Similar to Lukats et al. [55], we observed that MTR was undefined in some edge cases due to zero-division problems. To ensure a reliable comparison between monitors using all metrics, we added the following considerations:

- If no ground-truth drift event is present, the evaluation is omitted, as the absence of drift provides no valid basis for assessing the monitor's performance.
- If a monitor exhibits no false positives, we assume that false positives occurred before and after their signals were captured. Therefore, its mean time between false positives (TFP) is defined as the total sequence length + 1. Bifet et al. [11] similarly use the sequence length in such cases.
- Since a detection is defined as a true positive for a given drift if it occurs before the next drift event, a monitor with no detections is optimistically assumed to need one additional opportunity to detect each ground-truth drift. Therefore, in such cases, TTD is calculated from the maximum possible detection time (right before the following drift occurs) + 1 for each ground-truth drift except the last one. The last drift is excluded, since its maximum detection window is unknown. Bifet et al. [11] do not consider this because they only report the MTR metric, which can be simply set to 0 in this case. If only a single drift is present, a similar assumption used in the FPR calculation is applied: drifts occur immediately before the first step and immediately after the last step, and the distance between these two points is used to obtain the maximum possible time to detect.

- Since a monitor can detect a drift the moment it is induced, their TTD can be 0, which can cause a zero-division problem for MTR. Therefore, for the MTR calculation, TTD is clamped to a minimum of $10^{-min(3,\ detection\ count)}$, providing numerical stability while still rewarding fast detectors with an adjusted confidence.
- Other zero-division cases are replaced with 0.

An additional limitation of the proposed metrics, also explained by Lukats et al. [55], is that they use single points in time that correspond to a drift occurrence. However, gradual and incremental drifts take multiple time steps to stabilize. To address this limitation, we define the following specifications for the detection evaluation:

- A positive drift signal shall be considered a true positive if it was observed between the initial induction of a drift and the induction of the next one. A false negative is recorded when no detection occurs between two drift inductions. In other words, a valid detection may take place either during the progression of the drift or after it has stabilized, as long as it is observed before the induction of the next drift event.
- When a gradual or incremental drift is detected while it is still ongoing, any additional positive drift signal before the next drift induction is treated as a false positive.
- TTD is calculated using the induction time of the drift and the first positive signal after induction.

In our experiments, data monitors that obtained perfect or near-perfect performance without generating any false positives led to highly skewed MTR results. To mitigate the skewness, we propose applying a log1p transformation and analyzing Log MTR instead ($Log\ MTR = ln(MTR+1)$). This transformation allows compressing larger numbers while preserving the MTR's monotonic relationship with other metrics, such as the F1 score.

As an alternative to using the ground truth from the metadata, we defined two empirical methods to obtain ground truths for detection evaluation. In the first one, for a given marginal distribution, whether there is a significant change between two time steps is decided using the observed values, while the second one uses the distributions stored in the metadata and does sampling from those distributions with the requested sample size before making the comparison. These alternative methods are useful when drifts are not reasonably ensured to create a significant difference. However, their usefulness is limited depending on the sample size. We did not leverage the alternatives, as we created our own datasets with tuned drift significance and relied on the tuned ground-truth drifts.

3.4 Pipelines

To support benchmarking and experimentation, we developed three pipelines that extend a base class, support YAML or JSON configuration files (which are validated against the pipeline schema), and can be executed through Python or a command-line interface.

Our first pipeline, *SmartDriftGenerator*, is a wrapper for our dataset modeling and smart data drift generation features mentioned in Section 3.1.

Our second pipeline, *UnivariateDataMonitorBenchmark*, facilitates optimizing, benchmarking, and selecting data monitors. If a reference dataset is provided, the pipeline first generates a benchmark dataset using *SmartDriftGenerator*. If optimization is needed, two datasets are needed (or generated using different seeds) to prevent overfitting. The result of this pipeline indicates the best configuration for each data monitor, user-specified detection metrics, and which data monitors perform best for each input variable.

Our third pipeline, *RetrainingStrategyBenchmark*, facilitates comparing different retraining strategies (no retraining, periodic retraining, retraining based on model drift detection, and retraining based on data drift detection) using a set of performance and consumption metrics calculated over time using a benchmarking dataset. In this pipeline, the given set of univariate data drift detectors is weighted based on the feature importance of the features they track, and their consolidated drift signal is compared with a user-defined threshold to trigger retraining along with additional parameters. Apart from returning benchmark results, the pipeline can also generate an HTML report that visualizes data drifts, retraining trigger times and time ranges, predictor performance, and resource consumption over time, along with comparative plots across different strategies.

For modularity and reusability, we designed the classes and their configuration schema to facilitate easy subtask delegation between pipelines. For example, if the user passes a reference dataset to *RetrainingStrategyBenchmark* pipeline and provides data monitors that have parameters set to be optimized, data monitor optimization and benchmarking are delegated to the *UnivariateDataMonitorBenchmark* pipeline, which delegates benchmark dataset creation to the *SmartDriftGenerator* pipeline.

4 Experiment

Our toolkit features smart drift generation that enables users to create data drifts without requiring in-depth statistical knowledge by providing the desired p-value between two sample distributions — before and after drift generation. We hypothesized that smart drift generation can produce drifts that closely match the target p-value, ensuring controlled and statistically meaningful drift simulation.

4.1 Setup

To test our smart data drift generation, we created a base set of 15 distributions consisting of different SciPy distributions and our Gaussian mixture class (all supported for dataset fitting). Figure 3 illustrates the distribution histograms using a sample size of 1,000, demonstrating that the set is curated to include diverse distribution types with respect to shape, range, skewness, continuity, tailedness, cyclicity, discreteness, and modality.

For each distribution, we performed drift tuning using the default configuration that heuristically determines a bounded search range for each distribution parameter type and using an optimization budget of 100 iterations. Since our drift tuning evaluates only the final outcome rather than individual steps — and steps (for gradual and incremental drifts) are calculated from the desired final state — we limited this experiment to sudden drifts. To mimic a real scenario in which distributions evolve over time through multiple drifts, we performed 100 consecutive drifts for each feature to evaluate the robustness of our method. We repeated this experiment with four different target p-values to evaluate our approach's capability to generate drifts of varying statistical strength: 0.001 (highly significant), 0.01 (very significant), 0.05 (significant), and 0.5 (insignificant, effectively noise). For parameter optimization,

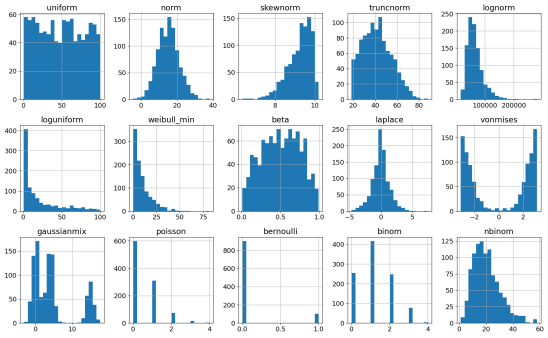


Figure 3: Distributions used to simulate a diverse set of features.

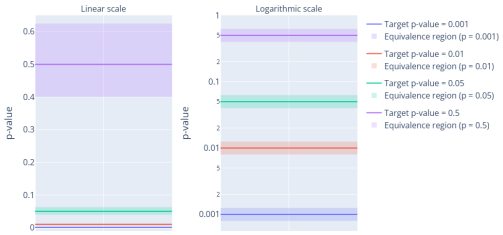


Figure 4: Target p-values and their equivalence regions in linear and logarithmic scales.

we applied the absolute error of the observed p-value as the loss to minimize and a sample size of 1,000 with different seeds for optimization and evaluation to statistically compare the original and drifted distributions.

4.2 Evaluation

For statistical testing, we considered multiple candidate tests from SciPy. We wanted to use a statistical test that can be applied to both continuous and discrete distributions, therefore eliminating the Kolmogorov–Smirnov test. We excluded the Eppstein–Singleton test because it exhibits numerical instability in some discrete distributions. We also excluded the Anderson–Darling test because of its p-value cap. We did not adopt a permutation test approach due to computational complexity. Instead, we used the Cramér–von Mises test due to its applicability to both continuous and discrete distributions, ease of interpretation, computational simplicity, and balanced sensitivity throughout the distribution.

For each target significance and feature combination, we analyzed the residuals (the difference between the observed and target p-values) and performed an equivalence test using two one-sided t-tests (TOST) with the residuals. Since p-values exhibit some log-normal characteristics, where the difference in p corresponds to a larger change in the lower extremes and p-value thresholds are traditionally log-spaced, we defined a multiplicative factor f. Following Lužar-Stiffler and Stiffler [56], we set f to 1.25 due to margins obtained from f being logarithmically symmetrical around zero, as $ln(1.25^{-1}) = -0.223$ and $ln(1.25) = +0.223$. Therefore, for a target p-value p, we defined the region of equivalence $\mathcal{E}$ as $[p \cdot 1.25^{-1}, p \cdot 1.25]$. Using this formula, we obtained the following regions of equivalence for each target p-value:

- 0.001: [0.0008, 0.00125] ([-0.0002, 0.00025] for the residuals)
- 0.01: [0.008, 0.0125] ([-0.002, 0.0025] for the residuals)
- 0.05: [0.04, 0.0625] ([-0.01, 0.0125] for the residuals)
- 0.5: [0.4, 0.625] ([-0.1, 0.125] for the residuals)

Figure 4 presents target p-values and their equivalence regions using both linear and logarithmic scales. We observe that the regions are both symmetrical and uniform, while the target p-values are approximately equidistant on the logarithmic scale. These margins also allowed us to have non-overlapping regions that also grow with the growing target p-value. Using our multiplicative factor, we defined our two null hypotheses as

$$H_{01} : \frac{\mu}{p} \leq \frac{1}{f}, \quad H_{02} : \frac{\mu}{p} \geq f \tag{2}$$

where μ represents the mean of the residual population. Therefore, if the p-values are significantly higher than the lower bound (rejecting H_{01}) and significantly lower than the upper bound (rejecting H_{02}), the p-values are considered to be within the equivalence region and *close enough* to the target value. We used residuals instead of raw p-values for easier comparison and error direction analysis across different significance levels.

It was important that drift tuning does not yield monotonic trends in samples unless the user intentionally specifies a very slow incremental drift, as a trend over longer periods of time could be *interpreted* as an incremental drift itself, reducing the reliability of the ground truth. A monotonic trend in drift tuning performance was also potentially dangerous, as it would suggest that drift tuning is not reliable in longer runs. It was also important that drift tuning does not produce autocorrelations. An autocorrelation in sample means can reduce the statistical power of tests, and over longer periods of time, it can have negative impacts on drift detection algorithms. An autocorrelation in drift tuning performance can suggest that drift tuning may stagnate after arriving at a distribution state. For these reasons, we tested both drift tuning performance (using p-value residuals) and distribution sample means.

For drift tuning performance, we checked for monotonic trends using the Spearman rank correlation between the residuals and the time indices. To check for autocorrelations, using residuals and time indices again, we employed the Ljung–Box test with an alpha of 0.05 and a maximum lag of 5, following the recommended percentage (5% of the time-series length) by Burns [14]. To account for the increased false discovery rate (FDR), we applied the Benjamini–Hochberg correction using the number of features for each p-value (separately for each target p-value). For sample means, we followed the same Spearman and Ljung–Box procedures using the sample means and the time indices.

4.3 Findings

By subtracting the target p-value from the observed p-values for each tuned drift, we obtained residuals showing how far each drift was from the target (0.001 for highly significant, 0.01 for very significant, 0.05 for significant, and 0.5 for insignificant). Figure 5

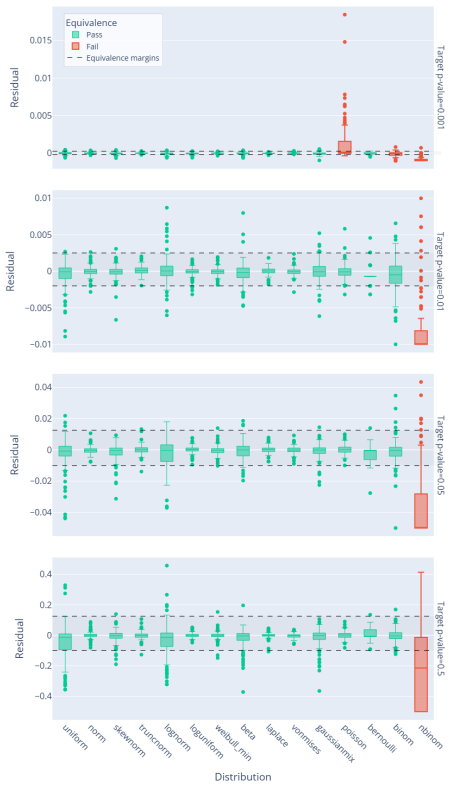
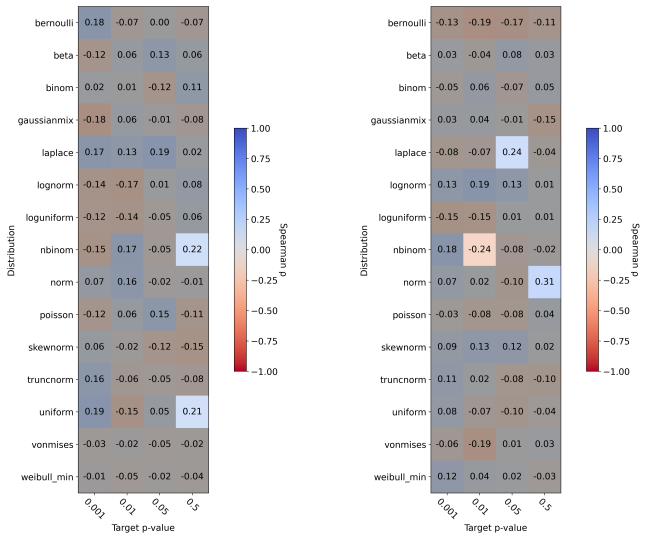


Figure 5: Residuals obtained from p-values and target p-values for each distribution. Dashed lines represent the equivalence margins for each target p-value.



(a) Residuals **(b) Sample means**

Figure 6: Spearman correlations between time indices and residuals (left) or sample means (right). Darker shading indicates statistically insignificant relationships ($p \geq 0.05$).

illustrates the residuals for each feature and target p-value combination, along with target-specific equivalence margins. The residuals had an extremely narrow range for drifts that target a p-value of 0.001 (highly significant), while their range and variance increased progressively with higher target p-values.

According to the TOST criteria, equivalence regions also indicate the 90% confidence interval into which each distribution's residuals must fall at each target p-value. Distributions that do not satisfy this condition are considered to exhibit drift levels that deviate from the desired intensity; these cases are highlighted in red in Figure 5. With the exception of the negative binomial distribution, smart drift generation achieved results within equivalence margins for very significant, significant, and insignificant drift levels. With highly significant drifts, the negative binomial, binomial, and Poisson distributions exhibited residuals statistically different from zero. In particular, the Poisson distribution showed the greatest deviation, displaying a notably wider confidence interval with less significant drifts compared to the other distributions. Across all target p-values, the negative binomial distribution had a positive skewness, especially exhibiting greater variance in very significant and other less significant target p-values.

Figure 6 shows the Spearman rank correlations between the time indices and either (a) the residuals or (b) sample means for each feature and target p-value, where the insignificant correlations are shaded darker (alpha = 0.05). For residuals, most correlations were found negligible and insignificant. With insignificant drifts, we noticed that negative binomial and uniform distributions had significant yet weak positive correlations (0.22 and 0.21, respectively), suggesting tuned drift p-value has a slight positive trend over time for such distributions. For sample means, no significant correlation existed in highly significant drifts. With very significant drifts, *nbinom* had a weak negative correlation (-0.24). With significant drifts, *laplace* had a weak positive correlation (-0.24). With insignificant drifts, *norm* had a moderate positive correlation (0.31).

Our Ljung–Box test results for residuals and sample means for each feature and target p-value combination are reported in Table 1. With an alpha of 0.05 and a maximum lag of 5, the critical cut-off $\chi^2_{5,0.95}$ for the Ljung–Box statistic (Q) was 11.07. With residuals, most distributions' raw Q was fairly below this threshold. As an exception, *skewnorm* and *vonmises* in very significant drifts and *gaussianmix* in significant drifts had a Q higher than the critical cut-off. However, due to testing for 15 features, we used the Benjamini–Hochberg correction within each target p-value group and reported the adjusted p-values. Obtaining the minimum adjusted p-value of 0.23, we did not reject the null hypothesis that there was no autocorrelation. We obtained similar results with sample means; after applying the Benjamini–Hochberg correction, the minimum adjusted p-value was 0.175, not finding a significant autocorrelation.

We also visually examined sample means over time by plotting line charts, color-coded by their target p-value. The results showed no consistent directional trend in the means. However, we observed a clear pattern that distributions with more significant drifts demonstrated consistently stronger oscillations, while those with less significant drifts displayed narrower oscillation ranges and were more densely concentrated around the central region. An example is presented in Figure 7.

	Adjusted p-value per target drift level							
	Residual				Sample mean			
Feature	0.001	0.01	0.05	0.5	0.001	0.01	0.05	0.5
uniform	0.98	0.43	0.97	0.88	0.68	0.66	0.72	0.84
norm	0.98	0.77	0.97	0.97	0.72	0.59	0.48	0.73
skewnorm	0.98	0.29	0.40	0.97	0.72	0.18	0.91	0.80
truncnorm	0.98	0.58	0.97	0.88	0.68	0.66	0.94	0.73
lognorm	0.98	0.85	0.97	0.97	0.91	0.95	0.71	0.73
loguniform	0.98	0.58	0.97	0.97	0.68	0.66	0.48	0.73
weibull_min	0.98	0.51	0.97	0.88	0.68	0.95	0.91	0.73
beta	0.98	0.36	0.97	0.97	0.72	0.95	0.94	0.73
laplace	0.98	0.71	0.97	0.97	0.68	0.66	0.48	0.73
vonmises	0.98	0.36	0.97	0.88	0.68	0.18	0.51	0.73
gaussianmix	0.98	0.43	0.23	0.97	0.87	0.72	0.91	0.84
poisson	0.98	0.68	0.97	0.88	0.74	0.66	0.91	0.73
bernoulli	0.98	0.58	0.97	0.97	0.72	0.66	0.48	0.73
binom	0.98	0.58	0.97	0.97	0.89	0.66	0.91	0.73
nbinom	0.98	0.51	0.97	0.88	0.92	0.38	0.72	0.73

Table 1: Ljung-Box test results with residuals and sample means denoting the autocorrelation up to lag=5 with Benjamini-Hochberg correction for each target drift level.

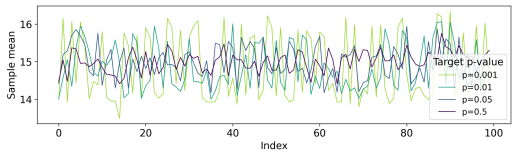


Figure 7: *norm* sample mean over time, color-coded by target p-value.

We observed a similar pattern with distribution parameter values over time, particularly for location parameters. The pattern was less visible in parameters that had a competing effect with other parameters or had a non-linear impact on the sample, which also had greater residuals. Given the complex relationship between parameter values and the samples, and since our primary focus was on drift tuning performance and sample means, we did not conduct further analysis of parameter values.

Temporal sample means and parameter value plots, and other additional materials are available in our replication package [47].

5 Discussion

In this section, we discuss our experiment results, implications, and limitations.

5.1 Smart Drift Generation

We hypothesized that our smart data drift generation approach can reliably induce drifts that are *sufficiently* close to the desired levels without manual specification of distribution parameters. We defined reasonable regions of equivalence using multiplicatively symmetrical margins from the literature [56]. We then performed two one-sided t-tests to evaluate if the observed drift magnitudes fell within these predefined equivalence bounds. Our results demonstrated that, except for the negative binomial distribution, our smart data drift generation can reliably tune drifts to desired p-values of 0.01 (very significant), 0.05 (significant), and 0.5 (insignificant).

Our results were less reliable for highly significant drifts, in particular, for Poisson, binomial, and negative binomial distributions. Considering all three distributions are discrete, we argue that drift tuning is less reliable with discrete distributions. This is supported by the fact that small changes can cause a larger shift more easily in discrete distributions compared to continuous ones, in particular with heavy tails. Moreover, all three distributions had only one tunable parameter (p for the binomial and negative binomial distributions, *mu* for the Poisson distribution) that also had a non-linear effect on the shape of the distribution.

For discrete distributions or distributions with fewer parameters, increasing the number of iterations can address the parameter's nonlinear effect on the distribution. While the drift tuning functionality allows specifying the iteration count, our pipeline currently uses a single configuration parameter for all features.

We note that the region of equivalence for the target p-value of 0.001 was very narrow (between 0.0008 and 0.00125). Furthermore, we observed that most distributions with residuals that were statistically too different from zero had a central tendency below the target significance. Therefore, for use cases where the user has an upper bound but not a lower bound for the significance, it might be possible to reliably use even a negative binomial distribution. In other cases, for distributions with known below-target tendencies, using the provided asymmetrical penalty option may be helpful.

Across all target p-values and distributions, Spearman rank correlations between p-value residuals and time steps yielded significant correlations with only negative binomial and uniform distributions when combined with insignificant drifts, and their correlation was weak (0.22 and 0.21, respectively). Spearman rank correlations between sample means and time indices displayed a similar pattern, where most distributions exhibit no significant correlation, except for three distributions, and the largest correlation was moderate (0.31) with insignificant drifts.

While the correlations suggest that our drift tuning and data shifts generally neither stagnate nor show a trend over time, significant correlations were more frequent and intense for insignificant drifts. Due to a likely larger parameter space satisfying insignificance compared to significance, insignificant drift generation may be less prone to the oscillation behavior observed with significant drifts, potentially exhibiting a weak monotonic trend. However, additional tests are needed to identify if and why insignificant drifts are statistically more likely to have an overall trend.

Through Ljung–Box tests, we found no autocorrelation in drift tuning performance or sample means over time, suggesting that our drift tuning not only does not stagnate over time, but it is also more likely to recover from underoptimisation.

RQ1: To what extent can we generate synthetic datasets with highly controllable data drifts?

Answer: Our toolkit enables precise control over data drift generation by allowing users to specify changes in distribution parameters. It also supports declarative and automated drift generation, reliably adjusting drift intensity across a wide range of continuous and discrete distributions.

5.2 Drift Detection Evaluation

We proposed a set of additional rules that addresses edge cases and non-sudden drifts, which were not covered by Bifet et al. [11].

Using the drift ground truth annotations of our data generator and Boolean data monitor signals, we facilitated the use of popular classification-oriented metrics, such as the F1 score, and drift detection metrics, such as MTR and TTD, that were originally not possible to use in this domain due to a lack of ground truth.

We developed a configurable benchmarking pipeline that facilitates optimizing and evaluating detection speed and accuracy of data monitors using different metrics.

There are multiple possible ways to consider whether a drift signal is correct or not. Our one-signal-per-induction rule provides a somewhat flexible but straightforward approach and coincides with our drift tuning, which focuses only on the final difference rather than each step. However, it may also be unfavorable for gradual drifts, especially of longer durations, due to their oscillating nature. It also favors configurations that signal a drift more sparingly, likely towards their stabilization point, which is potentially more suitable if labeling of newer instances is costly or infeasible.

RQ2: To what extent can we generate synthetic datasets with highly controllable data drifts?

Answer: We generated synthetic datasets with well-defined data drift ground truths and established evaluation rules to classify detection signals as true positives, true negatives, false positives, and false negatives. We developed a data drift detection benchmarking pipeline that computes a variety of metrics, including F1 score, Mean Time Ratio (MTR), Time to Detect (TTD), and others. Furthermore, we extended existing evaluation and computation rules to ensure that these metrics are applicable across all evaluation scenarios.

5.3 Implications

Our toolkit offers a tool support for practitioners such as data scientists and ML engineers to conduct drift detection experiments. It facilitates dataset fitting, data generation with and without data or concept drifts, and evaluation of different detection methods. A unified data monitor interface facilitates switching between data monitors, combining libraries, and adapting batch-based methods to streaming. ML or data engineers can integrate the toolkit into their production workflows.

By integrating it with a graphical user interface, abstracting statistical details, and using configuration presets, domain experts can be supported for decision-making regarding data monitor choice or other ML lifecycle decisions.

5.4 Limitations

A major limitation of our data generation method is that, while it preserves marginal distribution characteristics, it does not capture correlations and complex relationships between features due to independent feature sampling. However, this limitation is a trade-off that ensures marginal distribution isolation, which is necessary to know if there is a shift in a given distribution at a given time. Moreover, univariate approaches are inherently more explainable than multivariate approaches. Furthermore, almost all batch-based data drift detection methods in relevant libraries are either univariate by design or only applicable in a univariate manner (16/17 for Frouos and 18/18 for Evidently); thus, our study is limited to univariate drift detection methods, which inherently cannot detect certain multivariate changes (such as individual distributions being stable while their correlations drift).

To keep concept drifts easy to generate and interpret, we chose to model the target variable using an OLS linear regression model, which is fitted to the original data and used to generate the target variable. However, this limits our data generator to numerical distributions, which can be partially addressed by one-hot-encoding and the use of ordinal representation. It likely also limits the target variable representativeness of our benchmark datasets, since linear regression is limited to linear relationships. While our study focuses on data drifts, concept drift simulation remains a limitation.

We used a sample size of 1,000 for statistical convenience. Small sample sizes reduce the statistical power and can result in exaggerated drifts to ensure the desired significance. We observed that drift tuning stability suffers under a sample lower than a few hundred. To address this limitation, the drift tuning feature allows overriding the sample size to be used during tuning, or the user can use empirical evaluation alternatives from the toolkit (albeit provided as functions and not as a pipeline). Additional studies are needed to understand the effect of sample size on drift tuning.

6 Conclusion

In this work, we proposed a modular and extensible toolkit, RegDriftKit, to support data generation, drift detection, and drift detection benchmarking in regression scenarios, and we evaluated the toolkit's smart data drift generation functionality. Our drift generation experiment demonstrates that RegDriftKit allows reliable, precise, and declarative data drift generation without requiring deep statistical knowledge.

We extended the previously proposed drift detection metric, MTR, by proposing a set of complementary rules to allow its use in edge cases and with non-sudden drifts. 12 benchmark datasets with metadata, the toolkit's source code, UML diagrams, supporting materials used to obtain our experimental results, and a document with additional figures and tables are publicly available [47].

Future work: RegDriftKit is an open-source prototype and will be refactored by IFS Canada before it is released as a fully-fledged open-source project. Further work may evaluate its *dataset fitting performance*, which affects data monitor optimization and predictor performance. Future developments may explore *smart concept drift generation* and dataset fitting alternatives that preserve *feature correlations* at the expense of univariate data drift ground truth.

Acknowledgments

This work was supported by the Mitacs Accelerate program (application ref. IT43780) [58]. We thank the IFS Canada team members, Olivier Miguel and Jayson Crasto, for their feedback and discussions during the research project.

References

[1] AI4EOSC Consortium. 2025. AI4EOSC. https://ai4eosc.eu/

[2] Marie Al-Ghossein, Pierre-Alexandre Murena, Antoine Cornuéjols, and Talel Abdessalem. 2018. Online Learning with Reoccurring Drifts: The Perspective of Case-Based Reasoning. In *3rd Workshop on Synergies between CBR and Data Mining, ICCBR. Stockholm, Sweden.*

[3] Shruti Arora, Rinkle Rani, and Nitin Saxena. 2024. A systematic review on detection and adaptation of concept drift in streaming data using machine learning techniques. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* (2024), e1536.

[4] Manuel Baena-García, José del Campo-Ávila, Raul Fidalgo, Albert Bifet, Ricard Gavalda, and Rafael Morales-Bueno. 2006. Early drift detection method. In *Fourth international workshop on knowledge discovery from data streams*, Vol. 6. Citeseer, 77–86.

[5] Lucas Baier, Marcel Hofmann, Niklas Kühl, Marisa Mohr, and Gerhard Satzger. 2020. Handling Concept Drifts in Regression Problems: the Error Intersection Approach. *arXiv preprint arXiv:2004.00438* (2020).

[6] Lucas Baier, Fabian Jöhren, and Stefan Seebacher. 2019. Challenges in the Deployment and Operation of Machine Learning in Practice.. In *ECIS*, Vol. 1.

[7] Firas Bayram, Bestoun S Ahmed, and Andreas Kassler. 2022. From concept drift to model degradation: An overview on performance-aware drift detectors. *Knowledge-Based Systems* 245 (2022), 108632.

[8] Commandant Benoit. 1924. Note sur une méthode de résolution des équations normales provenant de l'application de la méthode des moindres carrés à un système d'équations linéaires en nombre inférieur à celui des inconnues (Procédé du Commandant Cholesky). *Bulletin géodésique* 2, 1 (1924), 67–77.

[9] Albert Bifet and Ricard Gavalda. 2007. Learning from time-changing data with adaptive windowing. In *Proceedings of the 2007 SIAM international conference on data mining*. SIAM, 443–448.

[10] Albert Bifet and Elena Ikonomovska. 2009. Airlines Dataset. https://www.openml.org/search?type=data&sort=version&status=any&order=asc&exact_name=airlines&id=1169

[11] Albert Bifet, Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Indrė Žliobaitė. 2013. CD-MOA: Change detection framework for massive online analysis. In *Advances in Intelligent Data Analysis XII: 12th International Symposium, IDA 2013, London, UK, October 17-19, 2013. Proceedings 12*. Springer, 92–103.

[12] Jock A Blackard. 1998. *Comparison of neural networks and discriminant analysis in predicting forest cover types.* Colorado State University.

[13] Eike Christian Brechmann and Ulf Schepsmeier. 2013. Modeling dependence with C-and D-vine copulas: the R package CDVine. *Journal of statistical software* 52 (2013), 1–27.

[14] Patrick Burns. 2002. Robustness of the Ljung–Box test and its rank equivalent. *Available at SSRN 443560* (2002).

[15] Ross Callister, Mihai M Lazarescu, and Duc-Son Pham. 2015. Detection of Structural Changes in Data Streams.. In *AusDM*. 79–88.

[16] Rohgi Toshio Meneses Chikushi, Roberto Souto Maior de Barros, Marilu Gomes N Monte da Silva, and Bruno Iran Ferreira Maciel. 2021. Using spectral entropy and Bernoulli map to handle concept drift. *Expert Systems with Applications* 167 (2021), 114114.

[17] Zeynab Chitsazian and Saeed Sedighian Kashi. 2023. Detecting Concept Drift in Just-In-Time Software Defect Prediction Using Model Interpretation. (2023).

[18] Shir Chorev, Philip Tannor, Dan Ben Israel, Noam Bressler, Itay Gabbay, Nir Hutnik, Jonatan Liberman, Matan Perlmutter, Yurii Romanyshyn, and Lior Rokach. 2022. Deepchecks: A Library for Testing and Validating Machine Learning Models and Data. *Journal of Machine Learning Research* 23 (2022), 1–6. http://jmlr.org/papers/v23/22-0281.html

[19] Albert Costa, Rafael Giusti, and Eulanda M dos Santos. 2025. Analysis of descriptors of concept drift and their impacts. In *Informatics*, Vol. 12. MDPI, 13.

[20] Harald Cramér. 1928. On the composition of elementary errors: First paper: Mathematical deductions. *Scandinavian Actuarial Journal* 1928, 1 (1928), 13–74.

[21] Jaime Céspedes Sisniega and Álvaro López García. 2024. Frouros: An open-source Python library for drift detection in machine learning systems. *SoftwareX* 26 (2024), 101733. doi:10.1016/j.softx.2024.101733

[22] Saverio De Vito, Ettore Massera, Marco Piga, Luca Martinotto, and Girolamo Di Francia. 2008. On field calibration of an electronic nose for benzene estimation in an urban pollution monitoring scenario. *Sensors and Actuators B: Chemical* 129, 2 (2008), 750–757.

[23] Gregory Ditzler, Manuel Roveri, Cesare Alippi, and Robi Polikar. 2015. Learning in nonstationary environments: A survey. *IEEE Computational Intelligence Magazine* 10, 4 (2015), 12–25.

[24] Pedro Domingos and Geoff Hulten. 2000. Mining high-speed data streams. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. 71–80.

[25] Bo Dong, Yifan Li, Yang Gao, Ahsanul Haque, Latifur Khan, and Mohammad M Masud. 2017. Multistream regression with asynchronous concept drift detection. In *2017 IEEE International Conference on Big Data (Big Data)*. IEEE, 596–605.

[26] Lei Du, Qinbao Song, Lei Zhu, and Xiaoyan Zhu. 2015. A selective detector ensemble for concept drift detection. *Comput. J.* 58, 3 (2015), 457–471.

[27] Piotr Duda, Maciej Jaworski, and Leszek Rutkowski. 2017. On ensemble components selection in data streams scenario with reoccurring concept-drift. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 1–7.

[28] Sarah D'Ettorre, Herna L Viktor, and Eric Paquet. 2017. Context-based abrupt change detection and adaptation for categorical data streams. In *Discovery Science: 20th International Conference, DS 2017, Kyoto, Japan, October 15–17, 2017, Proceedings 20*. Springer, 3–17.

[29] Ryan Elwell and Robi Polikar. 2011. Incremental learning of concept drift in nonstationary environments. *IEEE transactions on neural networks* 22, 10 (2011), 1517–1531.

[30] Evidently AI. 2020. Evidently. https://github.com/evidentlyai/evidently/

[31] Hadi Fanaee-T and Joao Gama. 2014. Event labeling combining ensemble detectors and background knowledge. *Progress in Artificial Intelligence* 2 (2014), 113–127.

[32] Alvaro Figueira and Bruno Vaz. 2022. Survey on synthetic data generation, evaluation methods and GANs. *Mathematics* 10, 15 (2022), 2733.

[33] Joao Gama, Pedro Medas, Gladys Castillo, and Pedro Rodrigues. 2004. Learning with drift detection. In *Advances in Artificial Intelligence–SBIA 2004: 17th Brazilian Symposium on Artificial Intelligence, Sao Luis, Maranhao, Brazil, September 29-Ocotber 1, 2004. Proceedings 17*. Springer, 286–295.

[34] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. 2014. A survey on concept drift adaptation. *ACM computing surveys (CSUR)* 46, 4 (2014), 1–37.

[35] Manuel L González, Javier Sedano, Ángel M García-Vico, and José R Villar. 2022. A comparison of techniques for virtual concept drift detection. In *16th International Conference on Soft Computing Models in Industrial and Environmental Applications (SOCO 2021)*. Springer, 3–13.

[36] Google. [n. d.]. Introduction to Vertex AI model monitoring | Google Cloud. https://cloud.google.com/vertex-ai/docs/model-monitoring/overview#v2

[37] M Harries. 1999. Splice-2 comparative evaluation: electricity pricing (technical report unsw-cse-tr-9905). *Artificial Intelligence Group, School of Computer Science and Engineering, The University of New South Wales, Sydney* 2052 (1999).

[38] Constanze Hasterok, Jan Hermes, and Benedikt Stratmann. 2023. SiD^{2}Re - A novel simulation framework for drifting regression data. In *2023 IEEE 21st International Conference on Industrial Informatics (INDIN)*. IEEE, 1–8.

[39] Marius Hofert, Martin Mächler, and Alexander J McNeil. 2013. Archimedean copulas in high dimensions: Estimators and numerical challenges motivated by financial applications. *Journal de la Société Française de Statistique* 154, 1 (2013), 25–63.

[40] Geoff Hulten, Laurie Spencer, and Pedro Domingos. 2001. Mining time-changing data streams. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. 97–106.

[41] Elena Ikonomovska, Joao Gama, and Sašo Džeroski. 2011. Learning model trees from evolving data streams. *Data mining and knowledge discovery* 23 (2011), 128–168.

[42] iMagine Project Consortium. 2025. iMagine Project. https://www.imagine-ai.eu/

[43] Ioannis Katakis, Grigorios Tsoumakas, Evangelos Banos, Nick Bassiliades, and Ioannis Vlahavas. 2009. An adaptive personalized news dissemination system. *Journal of intelligent information systems* 32 (2009), 191–212.

[44] Ioannis Katakis, Grigorios Tsoumakas, and Ioannis Vlahavas. 2008. An ensemble of classifiers for coping with recurring contexts in data streams. In *ECAI 2008*. IOS Press, 763–764.

[45] Ioannis Katakis, Grigorios Tsoumakas, and Ioannis Vlahavas. 2010. Tracking recurring contexts using ensemble classifiers: an application to email filtering. *Knowledge and Information Systems* 22 (2010), 371–391.

[46] Heysem Kaya, Pinar Tüfekci, and Fikret S Gürgen. 2012. Local and global learning methods for predicting power of a combined gas & steam turbine. In *Proceedings of the international conference on emerging trends in computer and electronics engineering ICETCEE*. 13–18.

[47] Oz Kilic. 2025. RegDriftKit - CAIN '26. https://osf.io/g3ne5/overview?view_only=46a1caea285d45e5970edbd172c3f4a4

[48] Timothée Lesort, Massimo Caccia, and Irina Rish. 2021. Understanding continual learning settings with data distribution drift analysis. *arXiv preprint arXiv:2104.01678* (2021).

[49] Grace A Lewis, Sebastián Echeverría, Lena Pons, and Jeffrey Chrabaszcz. 2022. Augur: A step towards realistic drift detection in production ML systems. In *Proceedings of the 1st Workshop on Software Engineering for Responsible AI*. 37–44.

[50] Xuan Liang, Tao Zou, Bin Guo, Shuo Li, Haozhe Zhang, Shuyi Zhang, Hui Huang, and Song Xi Chen. 2015. Assessing Beijing's PM2. 5 pollution: severity, weather impact, APEC and winter heating. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 471, 2182 (2015), 20150257.

[51] Patrick Lindstrom, Brian Mac Namee, and Sarah Jane Delany. 2013. Drift detection using uncertainty distribution divergence. *Evolving Systems* 4 (2013), 13–25.

[52] Viktor Losing, Barbara Hammer, and Heiko Wersing. 2016. KNN classifier with self adjusting memory for heterogeneous concept drift. In *2016 IEEE 16th international conference on data mining (ICDM)*. IEEE, 291–300.

[53] Jie Lu, Anjin Liu, Fan Dong, Feng Gu, Joao Gama, and Guangquan Zhang. 2018. Learning under concept drift: A review. *IEEE transactions on knowledge and data engineering* 31, 12 (2018), 2346–2363.

[54] Ning Lu, Jie Lu, Guangquan Zhang, and Ramon Lopez De Mantaras. 2016. A concept drift-tolerant case-base editing technique. *Artificial Intelligence* 230 (2016), 108–133.

[55] Daniel Lukats, Oliver Zielinski, Axel Hahn, and Frederic Stahl. 2024. A benchmark and survey of fully unsupervised concept drift detectors on real-world data streams. *International Journal of Data Science and Analytics* (Aug. 2024). doi:10.1007/s41060-024-00620-y

[56] Vesna Lužar-Stiffler and Charles Stiffler. 2002. Equivalence testing the easy way. *Journal of computing and information technology* 10, 3 (2002), 233–239.

[57] Hristo Mavrodiev. 2019. London Bike Sharing Dataset. https://www.kaggle.com/datasets/hmavrodiev/london-bike-sharing-dataset

[58] Mitacs. 2025. Boost Innovation with Research-Based Internships - Mitacs Accelerate. https://www.mitacs.ca/our-programs/accelerate/

[59] MITRE. [n. d.]. Menelaus: Online and batch-based concept and data drift detection algorithms to monitor and maintain ML performance. https://github.com/mitre/menelaus

[60] Veena Mittal and Indu Kashyap. 2015. Online methods of learning in occurrence of concept drift. *International Journal of Computer Applications* 117, 13 (2015).

[61] Jacob Montiel, Max Halford, Saulo Martiello Mastelini, Geoffrey Bolmier, Raphael Sourty, Robin Vaysse, Adil Zouitine, Heitor Murilo Gomes, Jesse Read, Talel Abdessalem, et al. 2021. River: Machine learning for streaming data in Python. *Journal of Machine Learning Research* 22, 110 (2021), 1–8.

[62] Nimisha G Nair, Pallavi Satpathy, Jabez Christopher, et al. 2019. Covariate shift: A review and analysis on classifiers. In *2019 Global Conference for Advancement in Technology (GCAT)*. IEEE, 1–6.

[63] NannyML. 2023. NannyML (release 0.13.0). https://github.com/NannyML/nannyml

[64] Antonino Feitosa Neto and Anne MP Canuto. 2021. EOCD: An ensemble optimization approach for concept drift applications. *Information Sciences* 561 (2021), 81–100.

[65] New York City Taxi and Limousine Commission. 2015. TLC Trip Record Data: Yellow, Green, For-Hire Vehicle, High-Volume FHV. https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page Accessed: 2025-06-14.

[66] NOAA National Centers for Environmental Information. 1999. Global Surface Summary of the Day - GSOD. 1.0. https://www.ncei.noaa.gov/access/metadata/landing-page/bin/iso?id=gov.noaa.ncdc:C00516

[67] Beata Nowok, Gillian M Raab, and Chris Dibben. 2016. synthpop: Bespoke creation of synthetic data in R. *Journal of statistical software* 74 (2016), 1–26.

[68] Emilia Oikarinen, Henri Tiittanen, Andreas Henelius, and Kai Puolamäki. 2021. Detecting virtual concept drift of regressors without ground truth values. *Data Mining and Knowledge Discovery* 35, 3 (2021), 726–747.

[69] Shane Peckham and Bryan J. Smith. 2024. AI Drift — learn.microsoft.com. https://web.archive.org/web/20241217193430/https://learn.microsoft.com/en-us/ai/playbook/technology-guidance/mlops/drift-overview

[70] Siqi Ren, Bo Liao, Wen Zhu, and Keqin Li. 2018. Knowledge-maximized ensemble algorithm for different types of concept drift. *Information Sciences* 430 (2018), 261–281.

[71] River Contributors. 2018. Waveform Synthetic Dataset. https://riverml.xyz/0.18.0/api/datasets/synth/Waveform/

[72] River Contributors. 2020. Random RBF Drift Synthetic Dataset. https://riverml.xyz/0.18.0/api/datasets/synth/RandomRBFDrift/

[73] Jeffrey C Schlimmer and Richard H Granger. 1986. Incremental learning from noisy data. *Machine learning* 1 (1986), 317–354.

[74] SciPy. [n. d.]. SciPy. https://scipy.org/

[75] Abe Sklar. 1973. Random variables, joint distribution functions, and copulas. *Kybernetika* 9, 6 (1973), 449–460.

[76] W Nick Street and YongSeog Kim. 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*. 377–382.

[77] Yi Sun, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. 2019. Learning vine copula models for synthetic data generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 5049–5057.

[78] Alexey Tsymbal. 2004. The problem of concept drift: definitions and related work. *Computer Science Department, Trinity College Dublin* 106, 2 (2004), 58.

[79] UCI Machine Learning Repository. 1994. LED Display Domain Synthetic Dataset. https://archive.ics.uci.edu/dataset/57/led+display+domain

[80] UCI Machine Learning Repository. 2002. Poker Hand Dataset. https://archive.ics.uci.edu/dataset/158/poker+hand

[81] UCI Machine Learning Repository. 2015. Taxi Service Trajectory Prediction Challenge (ECML/PKDD 2015) Dataset. https://archive.ics.uci.edu/dataset/339/taxi+service+trajectory+prediction+challenge+ecml+pkdd+2015

[82] U.S. Department of Transportation, Bureau of Transportation Statistics. 2025. Airline On-Time Performance Data (EFD). https://transtats.bts.gov/DatabaseInfo.asp?QO_VQ=EFD Accessed: 2025-06-14.

[83] G. J. J. Van den Burg and C. K. I. Williams. 2020. An Evaluation of Change Point Detection Algorithms. *arXiv preprint arXiv:2003.06222* (2020).

[84] Arnaud Van Looveren, Janis Klaise, Giovanni Vacanti, Oliver Cobb, Ashley Scillitoe, Robert Samoilescu, and Alex Athorne. 2024. *Alibi Detect: Algorithms for outlier, adversarial and drift detection.* https://github.com/SeldonIO/alibi-detect

[85] Richard Von Mises. 1936. Wahrscheinlichkeit Statistik und Wahrheit: Einführung in die neue Wahrscheinlichkeitslehre und ihre Anwendung. (1936).

[86] Li-Yu Wang, Cheolwoo Park, Kyupil Yeon, and Hosik Choi. 2017. Tracking concept drift using a constrained penalized regression combiner. *Computational Statistics & Data Analysis* 108 (2017), 52–69.

[87] Gerhard Widmer and Miroslav Kubat. 1996. Learning in the presence of concept drift and hidden contexts. *Machine learning* 23 (1996), 69–101.

[88] Shujian Yu, Xiaoyang Wang, and José C Príncipe. 2018. Request-and-reverify: Hierarchical hypothesis testing for concept drift detection with expensive labels. *arXiv preprint arXiv:1806.10131* (2018).

[89] Jun Zhang, Graham Cormode, Cecilia M Procopiuc, Divesh Srivastava, and Xiaokui Xiao. 2017. PrivBayes: Private data release via Bayesian networks. *ACM Transactions on Database Systems (TODS)* 42, 4 (2017), 1–41.

[90] Indrė Žliobaitė and Jaakko Hollmen. 2015. Optimizing regression models for data streams with missing values. *Machine learning* 99 (2015), 47–73.