# Processes Acquiring Explainability from Data Drift Detection in IoT Applications

Thao Phung, Vinh Nguyen, Hong-Linh Truong
*Department of Computer Science, Aalto University, Finland*
{*thao.phungduc, vinh.nguyen@aalto.fi, linh.truong*}*@aalto.fi*

***Abstract*—In this study, we will examine current methods and tools for supporting the explainability of data-related drifts. We will focus on different types of drift (data drift, concept drift and quality of data) in the end-to-end ML development with IoT data. The evaluation will examine existing methods and tools from multiple aspects, such as inputs, outputs, APIs, and algorithms to identify issues in utilizing these methods and tools for extracting explanation in the end-to-end ML process. We will carry out experiments with real datasets and ML solutions for Base Transceiver Stations (BTS).**

## 1. Introduction

With increase in information from various sources of social networking sites and online news during COVID-19 pandemic, the number of IoT devices, mobile phones, and smart systems has grown significantly [1]. This leads to the need for IoT analytics of streaming data for multiple purposes of data processing including information retrieval, user reviews, customer records, etc. Methods for handling, organizing, and analyzing meaningful patterns from such overwhelming data capacity becomes significantly important and more challenging when inference results from machine learning (ML) is required in real-time with streaming data [2]. Traditional ML methods cannot handle streaming data in the dynamic and non-stationary environment such as concept drift. Thus, it is mandatory to have a systematic process and effective method that can perform adaptive analytics to every-changing nature of IoT data and capture unpredictable changes of the information to interpret the patterns, make decisions to update the models, or handle IoT maintenance periodically.

In recent papers, the research for explainability of data drift has been a prominent concern [3]. The challenges have shifted from how to detect data drift from machine learning lifecycle into how to extract explanations and understanding of data drift based on underlying distribution changes [4]. The field of explainability has emerged due to the requirements for understanding and extracting explanations from ML systems which require a higher degree of model complexity called as black box models.

As the traditional problems related to ML scalability and monitoring has been facilitated by the help of open-source tools such as Seldon Core, WhyLabs, NeptuneAI, KF-Serving, Kubeflow, and MLFlow [5], it is important that the designed systems could be monitored in real time and keep track of potential alerts from data drift in term of relationship between model performance and drift observations. To achieve that, we are going to address different questions regarding the type of explanations for data drift [4], the requirements for collecting sufficient data from drift detection [6], severity of data drifts related model performance [7], and target audience for these explanations, especially in case where explainability are hard to achieve with the tools focusing in model explanations such as LIME [8] and SHAP [9].

**Motivating example:** the BTS dataset is considered as an example to test the explainability of data drift for end-to-end ML development with the purpose of predictive maintenance. The dataset involves collections of various background information from sensors for monitoring and alerting. They are monitored to keep the operating temperature and voltage stable to facilitate the wireless communication. Previous work has discussed requirements for explainability of end-to-end ML for IoT systems [6]. This paper aims to provide more context and propose a systematic process that can assist ML developers and stakeholders to choose possible techniques and tools related to both data and concept drift detection for the entire system including data collection, training, model development and inference phases. In the data collection process, it is important to develop adaptive systems to monitor the changes of data sources. The underlying distribution changes and data drift are analyzed to form baselines for detecting drift and defining thresholds for false alarms. To ensure compatibility of different tools measuring drifts in separate phases, it is possible to combine drift-detected results from collection phase to support drift detection in model development processes where the drop in accuracy could be resulted from model or training data. Thus, explanations from ML pipelines are combined to offer valid references to face challenging questions of drift categorization, causation, interpretation and immediate actions to handle detected drifts. Before developing ML solutions or data pipelines, it is necessary to collect requirements from stakeholders and constraints from equipment and infrastructure to develop an integration system that can work compatible to tackle mentioned questions and provide explanations with well-defined criteria, accurate drift detection and time-bound manner.

The above example highlights complex processes for gaining explainability of data drift in ML services. We focus on combination of different viewpoints from stakeholders, developers, users to produce explainable pipelines for ML

services.

## 2. Background

### 2.1. Types of concept drifts

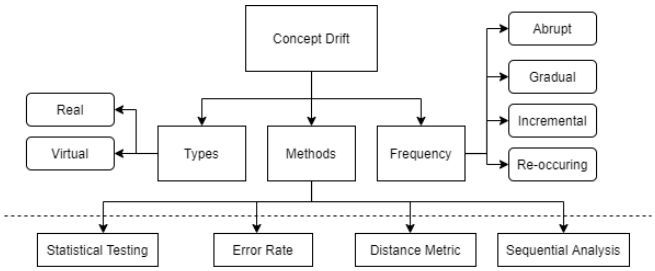


Figure 1: Taxonomy of data drift definition, drift frequency, and methods for detection

This paper also attempts to summarize different drift-related issues and their definition from existing literature, and they are given in abstract layer shown in Figure 1 to present overview of concept drift learning before proposing the systematic process for extracting explanations of data drifts to contribute to this knowledge area. From the last decade, there has been an increase in the number of papers [4], [10] that discuss different kinds of concept drift and divides them into various categories based on rate frequency of changes [11]. This paper and taxonomy from Figure 1 follows the definitions of concept drift framework from the work in [12], detection algorithms [13], and frequency patterns [14]. Related works [11], [15] provides the clear classification for many sources of drifts from related works such as data, covariate, population, real and virtual concept drift.

The remaining of this paper will use the term virtual concept drift and real concept drift to refer to shift in feature distribution and feature-label relationship, respectively.

### 2.2. Drift patterns

The results from changes of real and concept drifts lead to different types of patterns in term of speed presented in following cases:

- **Abrupt type:** this occurs when there is sudden change in the output label which is the new concept that is completely different with defined labels from training data.
- **Gradual type:** the shift is not abrupt but it goes back and forth between new and previous patterns. This happens due to gradual change of a label/concept overtime and the transition is not trivial when some changes are periodical or appear only once.
- **Incremental type:** there is a gradual transition to new distribution where the probability of sampled data in training data decreases and the one from the production data increases.
- **Re-occurring type:** when new that appears with repeated patterns and different drifting speed.

For our work, we need to understand and capture the drift patterns based on the above description as they are one of explainability aspects that helps user analyze the root cause of drift problems. For example, by deriving drift frequency, we categorize them into the similar groups of patterns and identify the corresponding causes, so the next time drift happens we can learn the characteristics or related issues resulted from the drifts. Depending on certain applications, various types of drifts could happen at the same time and many algorithms are designed to capture them based on frequency pattern, and these learning algorithms are discussed in the next section.

### 2.3. Drift handling methods

Before analyzing causes for the drift, the ML developers needs to design algorithms or functions to detect it first based on the nature of arriving information. There are many popular methods mainly used for classifying concept drift or performing data drift detection, and they are classified into four methods as described below.

**2.3.1. Error rate.** Drift Detection Method (DDM) is among the most cited papers within this category [16]. The method tries to detect if within a window of observations the online classification error rate increases drastically. This approach is enabled by the PAC learning model, which assumes the error rate of a learning model will decrease when the number of data increases given that the data generating distribution is stationary. In contrast, if the error rate of the algorithm increases significantly while the number of examples increases, then the underlying distribution is not necessarily stationary anymore. Binomial distribution with probability of misclassifying a data point $p_i$ and the standard deviation $s_i = \sqrt{p_i(1-p_i)/i}$ are used to set the warning level and drift level.

To further improve DDM, which only considers the number of examples, the authors of [17] incorporate the distance between two classification errors to better handle the gradual drift. Many other methods also extend the idea from DDM, include Learning with Local Drift Detection (LDDM) [18], Fuzzy time Windowing for Gradual Concept Drift Adaptation (FW-DDM) [19], and Heoffding's inequality based Drift Detection Method (HDDM) [20].

In the context of leveraging windowing strategies to detect concept drift, the authors of Adaptive Windowing (ADWIN) propose a dynamic scheme that lets the window expand when the examples are stationary, and shrink automatically when changes are detected. The method is guaranteed with a performance bounds on false negative and false positive rates. ADWIN2 version with the utilization of data stream algorithms is also proposed to tackle the space and time complexity issues presented in ADWIN [21].

One limitation of error rate based methods is that they are developed under the assumption that there is an instant

availability of label information. This assumption might not be realistic as in many real-world scenarios, there is a delay in the flow of label information, and in some extreme cases, the labels are very scarce or not available at all. This limitation poses a serious problem in the feasibility of detecting real concept drift in supervised learning settings [22].

**2.3.2. Statistical testing.** Non-parametric Kolmogorov-Smirnov (KS) hypothesis test [23] is widely used to test if two samples of data come from the same underlying distribution. One disadvantage of KS test is that it takes $O(NlogN)$ time to be applied on a pair of samples. In [23], the authors introduce the incremental variant of KS test (IKS) to reduce the time complexity to $O(logN))$ for adding/removing samples to the sliding window, and constant time for performing actual IKS test without label information.

Recently, there is an emerging interest in hierarchical drift detection. Proposed methods in this category usually consist of two components: **detection layer**, which continuously analyzes the incoming data stream using a low-complexity online sequential change-point test to detect if there is any change in the data generating process, and **validation layer**, which performs offline hypothesis testing on whether the trigger from detection layer corresponds to an actual data change or just false positive alarm [24].
To further extend the hierarchical hypothesis testing (HHT) approach, [25] contribute two methods, namely Hierarchical Hypothesis Testing with Classification Uncertainty (HHT-CU) and Hierarchical Hypothesis Testing with Attribute-wise "Goodness-of-fit" (HHT-AG). In both of these methods, layer I utilize test statistic to capture most the properties of the input stream in a fully unsupervised manner, then layer II will validate the detection from layer I (to reduce false alarm rate) in combination of labels **only when necessary** (to mitigate the increase in miss detection induced by the validation step).

**2.3.3. Distance metric.** The statistical distance method can be used to estimate the similarity between distributions of training and serving data, and they are useful when debugging failures appear in model inputs, outputs and hyperparameters during the ML processes. This involves monitoring feature levels between referenced distribution compared with serving data in production environment by various measures such as Population Stability Index (PSI) [26], Kullback-Leibler (KL) divergence, Jensen-Shannon (JS) divergence [27], Earth Mover's Distance (EMD) [28], Euclidean [29], etc.

**2.3.4. Sequential Analysis.** Another approach to for concept drift detectors is sequential analysis with the basis of Sequential Probability Ratio Test (SPRT) [30]. Algorithms in this category often concern with change detection problem, which try to identify whether there is a point separating the sequence of data into two subsequences having different underlying distribution. The Cumulative Sum (CUSUM) [31] is a sequential analysis method that triggers an alarm when the cumulative sum of the average of the incoming data stream deviates drastically from its mean. A variant of CUSUM is Page-Hingkley (PH) test that tracks the cumulative difference between the observed samples and their mean up until the current time.

## 2.4. Drift explanation methods

Thus, there are 4 additional areas that support explainable ML services in term of data drift and different processes:

- **Monitoring statistics of incoming data**: in production environment, it is difficult to acquire truth labels immediately for the incoming data and these statistics could be served as surrogated models for detecting model degradation
- **Detecting data drift**: by accounting the moment of data changes regarding both data and predictive distribution the shift is not abrupt but it goes back and forth between new and previous patterns. This happens due to gradual change of a label/concept overtime and the transition is not trivial when some changes are periodical or appear only once.
- **Mapping drift with model performance** there is a gradual transition to new distribution where the probability of sampled data in training data decreases and the one from the production data increases. Knowing the relationship between drift and model quality can enhance the predictability in handling events early when model accuracy drops or drifts occur.
- **Building transparent drift detection model**: Having transparent algorithms or methods in the system is critical for decision makers as they are informed by more reliable drift alarms and information analysis from detection that can benefit the system by validating the analysis results, giving the feedback on improving the accuracy of models and reducing technical errors [32].

## 2.5. Existing Tools for Drift Detection

There are some available tools for drift detection such as EvidentlyAI[1], scikit-multiflow[2], alibi-detect[3], Google VertexAI[4], Microsoft Azure Datashift[5], and Amazon SageMaker[6]. However, they only focus in one specific phase inside ML processes, and the explanation is deliver in process level that are difficult to understand root cause of drifts for the whole picture. In the experiment section, we

1. https://github.com/evidentlyai/evidently
2. https://scikit-multiflow.github.io/
3. https://github.com/SeldonIO/alibi-detect
4. https://cloud.google.com/vertex-ai
5. https://docs.microsoft.com/en-us/azure/machine-learning/how-to-monitor-datasets
6. https://aws.amazon.com/pm/sagemaker

will explore available tools and related works to combine them for producing inferences resulted from drift detection, and this will provide an holistic view of processes extracting explanation from drifts and facilitate the development of experiments to prove our concept.

There are few research topics [33]–[38] that describe different types of drift mapping with explanation contexts, root causes to certain drift types, and selected frameworks to handle and find causes for specific drift problems. This leads to a need for comprehensive summary on different state-of-the-art technologies that can investigate the nature or insights of provided drifts. Thus, the frameworks should provide an explainability approach that can deliver the results in understandable manners for humans [39]. On the other hand, there are some reviews related to drifts available but they are scattered in different domains, and they may have various concepts and focus on specific phases in ML projects [40].

## 3. Related Work

The recent topics for concept drift have focused on data stream applications, and they addressed problems of finding valuable information from infinite continuous data streams with resource constraints, preprocessing data automatically for continuous input arrivals, and time processing for unbounded size [41]. These problems cannot be solved by the traditional offline machine learning algorithms as they cannot extract useful information from continuous data or provide real-time processing on sequence of data samples with high speed. Thus, the concept drift should be handled and examined in different methods to determine the immediate change effects on accuracy of data inference as well as the gradual degradation of model performance on predicting unlabeled data streams [42]. Not every system has the input data with associated labels, so multiple frameworks are proposed to examine the effect of concept drift in big data streaming applications.

Using clustering algorithms to capture from streaming data is dissimilar with those from the static data or training datasets. As predicting a chunk of incoming data could not capture the whole data population, the concepts of underlying data may evolve and the clustering groups are changed in temporally relevant manners, which leads to the need for system learning data drift throughout processes to detect change and produce explainable predicted results in real time.

There have been many works attempting to give a detailed data related drift survey in the literature. A highly cited work is [15], which concentrates on concept drift adaptation. Within the survey, the authors examined various adaptive methods for supervised learning algorithms in environments having unpredictable dynamics. The authors also proposed numerous taxonomies of adaptive learning systems, including ”memory”, ”change detection”, ”learning”, and ”loss estimation”. Nevertheless, the paper didn't incorporate interpretation of drift detection into the system, and put more emphasis on algorithm, type, and capability of different concept drift adaptation methods.

In [4], the authors went a step further that besides studying concept drift detection and adaptation components, they proposed an additional concept drift understanding component as a bridge between the other two components. Information retrieved from the concept drift understanding component such as ”when”, ”how”, and ”where” are valuable for interpreting the dynamics of how the drift occurred, and from there facilitating the concept drift adaptation process. However, the authors only focused on examining whether the drift detection algorithms are intrinsically capable of returning explainability information.

To the best of our knowledge, there is no paper dedicated to exploration of end-to-end frameworks to enable drift explainability. The lack of existing literature inspires this paper to explore further into this area, as well as conduct comparison among existing drift detection tools against several criteria regarding explainability.

## 4. Data Drift Explainability Processes

In this section, we introduce the workflow and graph knowledge that incorporate different components of ML services to explore the relationship between explainability and data drift, and the designed system is illustrated in Figure 2

### 4.1. Workflow Processes for Drift Detection

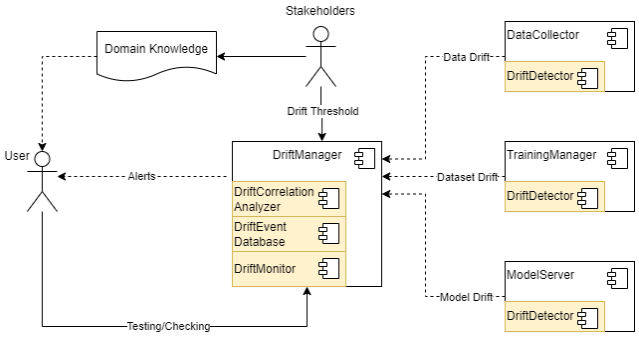


Figure 2: Workflow Process for Explaining Data Drift

The diagram from Figure 2 describes the end-to-end processes for ML systems that cooperate with potential drift detectors located Data Collection and Data Cleaning phase of ML process to extract drift-related information which is then stored in DriftManager component located in Model Development, Testing and Prediction phases in making decisions of updating ML models or debugging the data pipelines, and try to monitor data drifts via DriftMintor component based on the request of stakeholders or displaying key metrics of model performing based on monitoring components. Furthermore, the DriftEventDatabase stores historical information and provides necessary metadata and drift values to DriftCorrelationAnalyzer for analyzing the underlying drift cause and relationships between

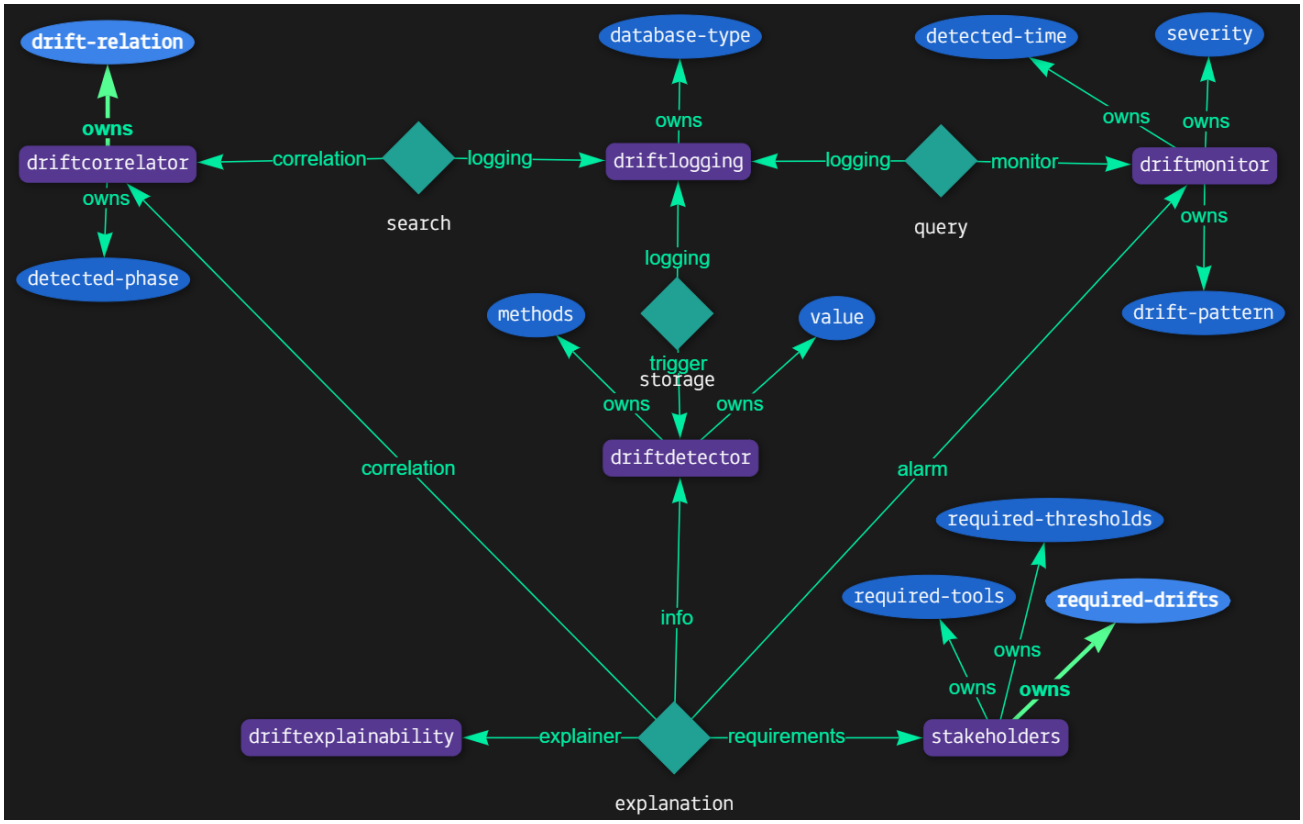


Figure 3: Graph knowledge beteen Explainability, Stakeholder, Process and Drift Information

drifts occured in different phases of data processes. These components presented in workflow process are integrated in different phases of ML systems to address the root causes from data drifts and the results from analysis by combination of drift events to answer the questions from stakeholders in term of explainability.

`DriftDetector`: Components of ML systems can be categorized into a set of abstraction layers. In this paper, only three layers are considered for incorporating drift detection. `DataCollector` module is responsible for handling the incoming streams of data to the system, including data cleansing, data validation, feature engineering, data loading, etc. The outputs from the data collection module will be the inputs to other modules. Therefore, detecting changes in the distribution of the data play a crucial role in anticipating future decay in model performance. On the other hand, it is also crucial to have drift detection in training and model serving phase as the quality of collected data sometimes remains the same but the input features may be skewed there would be skew in the input features, which implies the training data and serving data does not comply with similar data format or feature space. `TrainingManager` manages processes related to model training, validating, and testing. Within this module, possible drift between the training data distribution and validating/testing data distribution need to be detected to ensure model quality when deployed to production. In `ModelServer` module, functional and operational monitoring have been shown to be key in maintaining model quality in production-level ML systems [5]. Significant changes in feature attribution, model evaluation metrics, infrastructure health, system loads, etc. are indicators of possible serious system errors or model deterioration. In each of those modules, dedicated drift detectors can be incorporated to capture significant changes.

`DriftEventDatabase`: Only knowing about the drifting event is insufficient regarding the explainability aspect. In order to interpret the drift detection and manage drift adaptation, a comprehensive set of metadata needs to be collected and stored in the drift event database so that engineers or scientists can retrieve and analyze to understand the causes of the drift incidents. As discussed in the previous sections, information about where, how, and when the drift occurred should at least be captured in the database. One note is that there are two aspects regarding where the drift occurred. Real-world ML systems can possibly monitor hundreds of data streams simultaneously, so the identifier of the data stream experiencing drift should be recorded for fast query. From there, if the drift detection algorithm deployed in the system is capable of localizing the particular drifted region, that location information should also be recorded in the database. If this is not the case, additional software logics can be utilized to gather location information whenever the drift detector triggers the alarm.

`DriftMonitor`: is another critical module in the ex-

plainability framework. Visualizing information from the drift event database intuitively can help engineers and scientists quickly identify drifting events, explain their causes, and adapt the models to the change. Moreover, the module should also support sending notifications to the users about detected drift incidents according to the policies. This raises the necessary awareness of the current issues in the system.

`DriftCorrelationAnalyzer`: is another necessary module within the drift explainability framework. Some drift events exhibit either statistical or casual dependencies, and capturing those them can greatly facilitate the drift explanation process. This could be established by examining the drift events occured from different phases in the whole process. For example, there are associated drifts collected in data streams and these incoming data are grouped and inputted into statistical testing to observe the difference between the test and training data. At the same time, the predictions of ML model making inference on these inputs are stored to analyze the correlation between drift occurrences on data collection, ML training and serving phases. This brings valuable aspects for stakeholders to trace back the root cause of the problems and for developers to check the system performance if there is a rising link between.

`Stakeholders`: are also valuable sources of information to facilitate the drift explainability. They often hold a comprehensive domain knowledge that is not necessarily known by the developers and administrators of the ML system. Thus, they often provide useful rules to make inferences based on unobserved patterns. As a result, they are service customers that could provide insightful evaluation and provide feedback on the accuracy of change point detection for drifts.

## 4.2. Graph Knowledge for Drift Information

The graph knowledge is designed in way that can illustrate the ML processes and embedded requirements from stakeholders and developers to deliver expected explanations. Furthermore, they are combined as graph database to answer following questions that users would like to understand systems accuracy and inferences based on existing information.

**Which drift type?**: The drift type is crucial in supporting the explainability aspects. When the kind of drift is observed, the possible explanations could be extracted from it based on the characteristics of drift frequency. For this reason, the entity `DriftDetector` stores the drift types of detection based on requirements provided by stakeholders.

**When drifts happened?**: The observed time from drifts plays an important role in ML prediction, especially for time series format data. The model may gradually become outdated resulting from gradual or incremental drift, so exact time should be detected and informed to users for making decisions using the entity `DriftMonitor`. These captured times could be analyzed in `DriftCorrelator` to extract the drift pattern and support decision in frequency of updating the models or discarding the certain drifts when detected.

**Where drifts occurred?**: When a tool triggers an alarm for drift, the information about the features at which the drift occurred can be utilized to quickly locate the set of data streams that are experiencing drift. The `DriftDetector` transfer this information to `DriftLogging`, and they are queries when stakeholders required explanation or observation via `DriftMonitor`. That information might also give good hints about the root cause of the problem.

**How severe are drifts?** With the information about the severity of the drift stored in `DriftLogging`, the user can design the protocol to respond based on different drift scenarios. Thus, drift severity measures can also indicate evidence in the change of performance of machine learning models in production, and `DriftCorrelationAnalyzer` examine the relationships between the model accuracy and drift coefficients to support performance mapping.

**Transparency** The knowledge about the algorithm used within the tool can be added in `Stakeholders` as they provide valuable source of information when dealing with explainability, as they can reveal how the tool makes a particular detection in `DriftDetector`. Most of the black-box methods do not explain their inferences in the way that human can interpret [43], and the lack of explanations could lead to severe effects of misused algorithms, misleading actions to handle drifts, or troubleshooting algorithms when errors in system are discovered. Thus, interpretable algorithms are suitable for measuring drift applications as they are based on well-understood mathematical and statistical theory. Open-source tools usually show their algorithms, while commercial tools often hide away their algorithms to protect their intellectual properties. These mentioned factors should be put into `DriftExplainablity` when choosing the methods, tools, or algorithms in solving the problems.

**Who are target users?**: this value characterizes to whom the explanation is interpretable and chooses the suitable targets for the system as explainability methods could be defined in different terms based on their type, capacity and complexity. Using the entity `Stakeholders`, we can understanding users' behaviors, views, goals of interpretation and particular interactions with system's explanations, the levels of explanations are identified and this selects appropriate groups or target users based on their requirements and explanations displayed in `DriftMonitor` for explainability.

Putting them together, we can explain the whole picture when solving the ML problems related with data drifts. The process workflow in Figure 2 clarify the processes in handling IoT applications. To illustrate, when an alarm is triggered on the power usage at a BTS station, the ML developer might have to collaborate closely with the telecommunication engineer to discover the issue with the newer generation of solar panels. Those additional inputs and conditional requirements from the stakeholders are definitely beneficial for explainability and identification of the root cause of the drift event. Furthermore, it is crucial to collect these determined requirements from stakeholders into the drift

management system. From that, when problems happen, users can explain and derive necessary outputs from drift detection to relevant stakeholders regarding expectation for types of explanations extracting from data drift. On the other hand, in Figure 3, this shows detail implementation for basic structure of graph knowledge that is required to support both stakeholders and developers to make decision on selected tools for drift detection, required thresholds for features in the application, and the type of output explanations from the inferences. This influences the decision of how tools choose to display specific output types to related stakeholders such as JSON data, visualized graphs, HTML reports, or tabular data. The same applies for input data where specifications on configuration parameters and inspected elements are presented in input metadata. For instance, before the system performs drift detection, it acquires parameter settings for tools, threshold values on particular features presented in datasets for triggering alarms, or preferred attributes which will be monitored in real-time.

### 4.3. Experimental Selection Design

After introducing the workflow processes and graph knowledge for gaining explanations for detected drifts, the selection criteria for experiments are designed based on the objectives of BTS applications. Thus, the main focus of this illustration is to capture anomalous behavior of data collected from many BTS stations, which mostly involves time series data collected from sensors installed at BTS for predictive maintenance purposes. This requires the drift tools and processes to support functionalities for time series data and capture drift in an early time manner to validate the change point detection and trigger alarms if necessary. Another aspect is that different sensor data streams can exhibit correlation to each other [44], which required the tools to offer detection algorithms capable of working on not only univariate data but also multivariate data.

**4.3.1. Tool and Database Selection.** To demonstrate the workflow processes mentioned above, we perform experiments using `scikit-multiflow` [45], `alibi-detect` [46], `EvidentlyAI` which are mentioned in Background section. These tools are chosen due to their well-written documentation on tool usage, transparency in the algorithms used for capturing drifts and they are suitable for building the experiments on examining the explainability contexts from data drift detection. On the other hand, database system is also a necessary component to record detected drift events, their related metadata, as well as human inputs to the system for visualization and analysis. The support of visualization and notification modules are great criteria because they serve as the first insight into the problems in the system. Intuitive interfaces and efficient notifications will ease the workflow of the users. We use `Neo4j` [47] database as potential tool that support graph database to provide insight in term of graph visualization and query language with cause-and-effect relationships.

**4.3.2. Criteria Selection.** The framework proposed in Figure 2 will be used as the basis for multiple criteria for evaluating explainability of the tools. The tools of interest are expected to support detection algorithms capable of answering the ”where”, ”how”, and ”when” questions. Documentations about the underlying algorithms are also important for drift detection interpretation. Each algorithm has its own characteristics which can be utilized to support the investigation of the cause and effect of the detected drifts. Statistical testing based algorithms are usually easy to explain their predictions, while ML based algorithms, especially deep learning, might require extra care to reason about their prediction. Open-source drift detection tools usually publish their implementation, which promotes the transparency of the tool. In contrast, commercial tools often conceal their algorithms to protect their intellectual properties, which can hinder the explanation processes or trust of stakeholders in tool evaluation.

Other aspects of the system should also be considered to further contribute to the adoption of the tools as well as the understanding of the detected drift incidents. For example, ease of the integration of the tool to the system, time and space complexity of the detection algorithms, and input/output format specifications are beneficial sources of information. From the discussion above, here is the list of criteria that we would like to focus in our experiment:

- Tools provide support for various data types
- Tools contain different methods for drift detection
- Tools provide easy interface with database system
- Tools have meaningful graphical results
- Tools provide reasons for drift prediction

## 5. Experimental Contexts

Given such a framework for an end-to-end system for data drift and concept drift explainability, the context for choosing a set of criteria for evaluating existing tools for drift monitoring is based on monitoring drifts for BTS predictive maintenance data. In this experiment, we presents the designed workflow processes mentioned in the previous section to detect, evaluate, and visualize the drifts based on the proposed system to observe their abilities to explain the drift situations. On the contrary, the graph knowledge has not been implemented due to the lack of time and resources. Thus, the next focus on future experiments is exploring and analyzing the relationship between drifts in multiple places via graph databases.

### 5.1. Experiments

For this experiment, the current scope of our work is to cover the explainability aspects extracted from workflow processes in Figure 2 by using the tools (EvidentlyAI and alibi-detect) implemented as `DriftDetector` component to evaluate the explanations from the drift detection between two BTS locations which collect various sensor values. The types of sensors are used for detecting drifts are

`Temperature of Air Conditioner`, `Humidity`, `Voltage of Power Grid`, `Load of Power Grid`, `Load of Battery`, and `Capacity`. Thus, they are valuable information and captured at the end of the day for estimating the data distribution for each sensor parameter. From these certain properties, the drift detector can collect drifts and provide information that can distinguish the characteristics between two stations, and the ranged operations of sensor properties can be examined to decide if the same LSTM model could benefit for both stations.

**5.1.1. Experiment with alibi-detect.** For the alibi-detect tool, we are using the spot-the-diff detector on training the drift classifier on reference dataset from one station and distinguish with the test samples from different station [48]. This method is very useful for controlling the learned function that discriminates between reference and test sets using a kernel with appropriate coefficients that could be optimized with the help from domain experts for high accuracy detection. Furthermore, the method can cooperate with ML frameworks such as PyTorch [49] and TensorFlow [50], and the method also allows the GPU to speed up the computation process. In the experiment, we are using the default radial basis function Gaussian [51] to train the classifier and detect drift on the test set. Having a simple kernel can support the interpretability of the kernel function and this could be used as a baseline when training drift models for different models. To experiment the method, we perform a training process on the dataset containing information about one station and validate with unseen samples from the same underlying distribution.

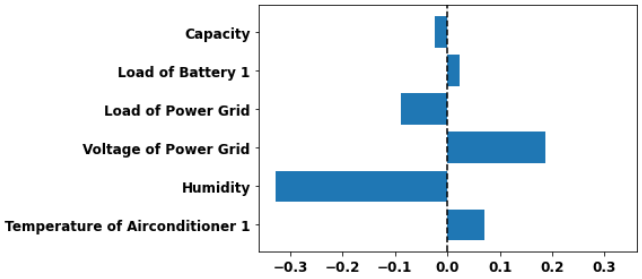


Figure 4: Drift coefficient of sensor parameters from spot-the-diff detector of alibi-detect tool

Thus, the model could be used as a baseline after several experiments with configuration parameters on learned kernels until drift is not detected on both training and validation sets. Then, this baseline model applies on the test set from different stations and the settings for p-value conditions are also tested for triggering the drift alerts based on user requests. On the other hand, the method offers information about the differences of feature levels using the drift coefficient as shown in Figure 4. This is useful for customers where they can understand how much difference between two stations given in numerical values and they can provide assessments for developers on how configured parameters should be adjusted to meet their requirements. With the result from drift coefficients, the customers can analyze their characteristics for two stations as reference points and extend their estimates on different stations on how to improve quality of sensors or decide suitable ML models on each station.

**5.1.2. Experiment with EvidentlyAI.** We are using the two-sample Kolmogorov-Smirnov (KS) test which is suitable for numerical features in IoT applications. The purpose is to detect the data distribution on different parameters if drastic change occurs. If the data contain different types of data such as categorical features or binary classes, additional methods could be considered such as chi-squared test or test on z-scores. The results shown in Figure 5 are captured from the tool detecting databases between two stations. Thus, the output is provided in the visualization report in the HTML file, which includes data drift by individual feature, feature distribution, drift table, and summary of the sare between drifting features. The tool also provides different configurations on setting components of the report including statistical test threshold, bin width of histograms, or report generated as JSON for triggering different applications.


Figure 5: Drift detection based on KS test on `Capacity` parameter

There are 2 drift parameters detected with p-value for similarity less than 0.05 out of 6 features and they are `Temperature of Air Conditioner` and `Capacity` that share similar distributions with minor differences in magnitude values, while `Voltage of Power Grid` have the same distribution structure but with value ranges such as 170-210 and 220-240 voltage unit respectively, and the tool cannot detect this gap using statistical testing shown in Figure 6. However, the difference can be clearly seen in the visualization report where the reference dataset's range value is far from the test dataset as described

in the same figure. This provides customers on how the drift detection makes an explanation based on a predefined threshold for p-value test and allows them to monitor the drift if the tool is making the correct judgment. Based on that, the customers can feedback to developers for adding, removing, or improving the conditional changes to the system. In addition to this, this has clear and precise visualized report and it can be integrated into reporting system such as daily email to customers shown in Figure 5.


Figure 6: Drift detection based on KS test on `Voltage of Power Grid` parameter

## 5.2. Results and Discussion

**Using graph database to extract relationships between drifts**: We have not experiment with collected drift using defined graph knowledge in Figure 3 due to time constraint, but we have established the base model for nodes and edges involved in explainability aspects that could use directly in database design. This could be our next target for experiments in drift causation, drift-related performance in ML models and relationship establishments between drifts in multiple phases of ML processes.

**Using tools to process different data types and volumes**: With our defined workflow in 2, we can easily work with different data types and frequency to detect drift. While EvidentlyAI shows its capability in detecting historical data frames, alibi-detect shows more extensive methods to process both online streaming data and past records with diversity and range of data types such as images, texts and data tables. Thus, this is useful to stakeholders' requirements relevant to computer vision applications. The process can capture what types of data being processed and the tools for drift detection, and this provides users an explanatory process.

**Using tools to compute drifts with different and interpretive results**: From two experiments using EvidentlyAI and alibi-detect tools with BTS data, they did not share the exact same method for processing the drifts, and they also show both disadvantages and advantages in terms of drift explanation. The spot-the-diff from alibi-detect is useful to stakeholders that want to examine the relationship between two stations given in numerical results. Thus, the results could be extended to multiple stations to examine characteristics of sensors on individual levels by comparing with other stations having the similar geographical locations or using the same communication technologies. On the other hand, experiments from Evidently tool offer visualization on how explanations are analyzed and this could give stakeholders further insights about how to improve the system. With our process, we can examine the visualizations from these tools and this could be applied to other drift detection frameworks to smooth the path of experiments and gaining interpretation from different tools. In addition to this, this experiment can help users have understand different drift detection methods, more flexibility in choosing the right tools for their applications and select principal keys of our process design to support both tool, method, and operation in developing drift detection systems.

**Using tools to interact with logging database systems for extracting drift information**: For both experiments, they are designed in the way that we are capable of capturing the information into the graph knowledge given in Figure 3. The stakeholders provide different threshold values regarding p-value on JSON files that can be input into database. Then the drift detectors extract requirements from database and use them as parameters for drift methods. At the same time, the drift monitor extracts drifts with wrapper functions depending on detecting approaches to extract available details regarding detected time, location, threshold drift, drift value, and detector method. The workflow process shown in the Figure 2 are important as it keeps track of relationships between components and requirements from stakeholders. Thus, when stakeholders request an explanation, they are informed about the location of severe drifts.

**Using tools to examine reasons and relationships of drift prediction happened on different phases of ML serivces**: In this paper, we do not focus on determining the complex relationships between drifts occured in different phases of ML systems. In our view, the stakeholders may have deep understanding of these relationships and they could define constraints for the attributes or complicated rules for their relationships. Furthermore, the threshold values defined for drift attributes can be complex as they may be context independent and they should be selected by service stakeholders or domain experts to optimize the constraints for drift detection. Both dependencies between drift attributes and specific constraints are out of our work but they play a crucial role for drift explainability.

**Using tools as complementary components to verify the drift movement**: Many researches on Drift detection so far focused on one phase of applications such as collection, training, or inference phase, but not cover the whole system to capture context of data and detect ambiguities in different phases. Hence, there is a possibility of using these tools to complement the explanation results in different manners such as visualization, interpretable results, and monitoring. For the scikit-multiflow tool, we are not able to experiment

| Tools | Inputs | | | | RequiredLabel | Outputs | Execution Environments | DeliveryMethod |
|---|---|---|---|---|---|---|---|---|
| | Types | Univariate | Multivariate | Streaming/Batch | | | | |
| scikit-multiflow | numerical | Yes | No | Streaming | Yes | Boolean | Single Machine, CPU | library |
| EvidentlyAI | tabular | No | Yes | Batch | No | JSON, HTML, Graph | Single Machine, CPU | library |
| alibi-detect | text, tabular, image | Yes | Yes | Batch, Streaming | No | JSON | Single Machine, CPU/GPU | library |
| AWS SageMaker | tabular | No | Yes | Batch, Streaming | Yes/No | JSON, Dashboard | — | online service |
| Azure Datashift | time-series tabular | No | Yes | Batch, Streaming | No | JSON, Dashboard | — | online service |
| Google Vertex AI | tabular | No | Yes | Batch, Streaming | Yes/No | JSON, Dashboard | — | online service |

TABLE 1: Tool interfaces for data drift detection

| Tools | Methods | | | | Algorithms | Detected Patterns |
|---|---|---|---|---|---|---|
| | Statistical Testing | Error-based | Distance Metrics | Sequential Analysis | | |
| scikit-multiflow | Yes | Yes | No | Yes | ADWIN, DDM, EDDM, HDDM_A, HDDM_W, KSWIN, PageHinkley [52] | Concept shift [45], [53] |
| EvidentlyAI | Yes | No | No | No | Chi-square, KS tests [23], [54] | Label, Covariate shift |
| alibi-detect | Yes | No | No | No | Chi-square, KS , CVM, FET, MMD, LSDD, Learned Kernel, Classifier, Spot-the-diff, Model Uncertainty [46] | Label, Covariate shift [5] |
| AWS SageMaker | Yes | No | Yes | No | Feature Attribution Drift, Model Bias based on DPPL [55] | Label, Covariate, Concept shift [55] |
| Azure Datashift | No | No | Yes | No | Azure ML models, Wasserstein, Euclidian, Min, Mean, Max [56] | Covariate shift [57] |
| Google Vertex AI | No | No | Yes | No | JS divergence, L-infinity [58], [59] | Covariate shift |

TABLE 2: Methodologies of tools using for data drift detection

with unsupervised data as the tool provides methods that are suitable for classification problems in which it requires the prediction results available before detecting the drifts [52]. In some IoT applications, it is difficult to get the truth labels or predicted values available in fast evolving data streams. However, this tool can perform an additional alarm to trigger a change point when the prediction model starts to reduce in accuracy or become outdated.

**Examing possible potentials from tools based on its specifications**: After experiments with some methods from mentioned tools, we believe that there are other methods from available tools that could provide various aspects regarding the drifts. In this section, we also summarize the capabilities of different frameworks and their methodology on detecting drifts to support stakeholders an overview on selecting tools for their applications.

Table 1 summarizes the interfaces related to tool implementation and usage that could be further explored to gain explanation from drift based on specific applications. This demonstrates what types of inputs are suitable for certain tools and how their inputs are computed to get the explanations. For example, alibi-detect tool handles various kinds of input types, while the scikit-multiflow tool performs drift detection on continuous values on one feature, and some methods of this library requires decision model to make a classification prediction such as EDDM and DDM [52]. The output of these functions from this tool results in boolean variable which indicate a shift appeared on data streams, whereas other tools can assess drift with batch datasets or window inputs and produces additional metrics such as p-value scores or drift scores based on feature levels, and they are included in JSON, HTML or graph reports. Additionally, Google Vertex AI and AWS SageMaker both have feature attribution drifts which compute differences of feature-level contributions based on the predictions. On the other hand, the Azure Datashift tool needs input datasets including timestamp column which is used to monitor the drift from the model's serving data, and the result appears on dashboard of Azure studio, similarly, the drift results could be observed in dashboard of AWS CloudWatch and Google Cloud Console when using AWS SageMaker and Google VertexAI respectively.

Table 2 shows the methods of tools using different algorithms to detect the drift. This supports stakeholders in choosing appropriate tools for their applications and covers some of the techniques used to identify certain types of drifts based on their needs. From the technical perspective, these tools provide various solutions to drift problems and they can be combined to provide holistic explanations to the users. For example, the KS test [23] works better with numerical datasets to distinguish the difference between subsets of two samples, while the chi-squared test [54] focuses on the differences between observations and expected results of categorical values. In addition to this, the feature attribution method applies on model perspective to monitor dissimilarities for categorical or numerical input features given predictions. Thus, by studying these tools, many aspects regarding input, method, output requirements can be understood clearly to make a decision on selecting the right tool based on requirements or using them as complementary tools to provide more explainable aspects in the system design.

## 6. Conclusions and Future Work

Gaining explainability from drift detection in ML processes is crucial for many businesses and stakeholders as this requires domain knowledge on the applications for specifying conditions for drift detectors and the deployment of drift systems to deliver understandable results that could support customers to make correct decisions. In this paper, we address different approaches for detecting drifts and systematic ways to extract valuable information from these drifts by having processes to capture certain attributes from stakeholders' requirements, methods for detection, and explanation results.

We experimented different tools regarding the BTS dataset based on demonstrated pipelines and provide evaluations for the explainability that the system could achieve. For the next experiments, we will perform experiment with scikit-multiflow tool by detecting drifts on prediction results from LSTM model applied to certain sensors, which could

help elaborate situations where the applications have trained ML models that are required to be restrained if severe drifts occur. The future work will focus on examining the relationships of drifts from detectors placed on different ML processes to extract the causality of the system for providing cause-effect relationship on explanations.

## References


[1] S. Ornes, "The internet of things and the explosion of interconnectivity," *Proceedings of the National Academy of Sciences*, vol. 113, no. 40, pp. 11 059–11 060, 2016. [Online]. Available: https://www.pnas.org/doi/abs/10.1073/pnas.1613921113

[2] V. Chandola, V. Mithal, and V. Kumar, "Comparative evaluation of anomaly detection techniques for sequence data," 01 2009, pp. 743 – 748.

[3] A. Lavin and S. Ahmad, "Evaluating real-time anomaly detection algorithms - the numenta anomaly benchmark," Nov 2015.

[4] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang, "Learning under concept drift: A review," Apr 2020. [Online]. Available: https://arxiv.org/abs/2004.05785

[5] J. Klaise, A. Van Looveren, C. Cox, G. Vacanti, and A. Coca, "Monitoring and explainability of models in production," Jul 2020. [Online]. Available: https://arxiv.org/abs/2007.06299

[6] M.-L. Nguyen, T. Phung, D.-H. Ly, and H.-L. Truong, "Holistic explainability requirements for end-to-end machine learning in iot cloud systems," in *2021 IEEE 29th International Requirements Engineering Conference Workshops (REW)*, 2021, pp. 188–194.

[7] P. Kosina, J. Gama, and R. Sebastião, "Drift severity metric," vol. 215, 01 2010, pp. 1119–1120.

[8] S. Shi, X. Zhang, and W. Fan, "A modified perturbed sampling method for local interpretable model-agnostic explanation," Feb 2020. [Online]. Available: https://arxiv.org/abs/2002.07434

[9] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 4768–4777.

[10] A. Tsymbal, "The problem of concept drift: Definitions and related work," 05 2004.

[11] 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*, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1319157821003062

[12] I. Khamassi, M. Sayed Mouchaweh, M. Hammami, and K. Ghedira, "Discussion and review on evolving data streams and concept drift adapting," *Evolving Systems*, vol. 9, 03 2018.

[13] S. Ackerman, E. Farchi, O. Raz, M. Zalmanovici, and P. Dube, "Detection of data drift and outliers affecting machine learning model performance over time," 12 2020.

[14] L. Minku, A. White, and X. Yao, "The impact of diversity on online ensemble learning in the presence of concept drift," *IEEE Trans. Knowl. Data Eng.*, vol. 22, pp. 730–742, 05 2010.

[15] 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. [Online]. Available: https://doi.org/10.1145/2523813

[16] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection," in *Brazilian symposium on artificial intelligence*. Springer, 2004, pp. 286–295.

[17] 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, 2006, pp. 77–86.

[18] J. Gama and G. Castillo, "Learning with local drift detection," in *International conference on advanced data mining and applications*. Springer, 2006, pp. 42–55.

[19] A. Liu, G. Zhang, and J. Lu, "Fuzzy time windowing for gradual concept drift adaptation," in *2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 2017, pp. 1–6.

[20] 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.

[21] 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.

[22] I. Žliobaité, "Change with delayed labeling: When is it detectable?" in *2010 IEEE International Conference on Data Mining Workshops*, 2010, pp. 843–850.

[23] 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.

[24] C. Alippi, G. Boracchi, and M. Roveri, "Hierarchical change-detection tests," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 2, pp. 246–258, 2017.

[25] S. Yu, X. Wang, and J. C. Príncipe, "Request-and-reverify: Hierarchical hypothesis testing for concept drift detection with expensive labels," *arXiv preprint arXiv:1806.10131*, 2018.

[26] B. Yurdakul and J. Naranjo, "Statistical properties of the population stability index," Feb 2021. [Online]. Available: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3783305

[27] J. Haug and G. Kasneci, "Learning parameter distributions to detect concept drift in data streams," Oct 2020. [Online]. Available: https://arxiv.org/abs/2010.09388

[28] T. Brockhoff, M. S. Uysal, and W. Aalst, "Time-aware concept drift detection using the earth mover's distance," 10 2020, pp. 33–40.

[29] D.-H. Tran, "Automated change detection and reactive clustering in multivariate streaming data," in *2019 IEEE-RIVF International Conference on Computing and Communication Technologies (RIVF)*, 2019, pp. 1–6.

[30] A. Wald, *Sequential analysis*. Courier Corporation, 2004.

[31] E. S. Page, "Continuous inspection schemes," *Biometrika*, vol. 41, no. 1/2, pp. 100–115, 1954.

[32] C. Sampson, R. Arnold, S. Bryan, P. Clarke, S. Ekins, A. Hatswell, N. Hawkins, S. Langham, D. Marshall, M. Sadatsafavi, W. Sullivan, E. Wilson, and T. Wrightson, "Transparency in decision modelling: What, why, who and how?" *PharmacoEconomics*, vol. 37, 06 2019.

[33] J. Demšar and Z. Bosnic, "Detecting concept drift in data streams using model explanation," *Expert Systems with Applications*, vol. 92, 10 2017.

[34] S. Micevska, A. Awad, and S. Sakr, "Sddm: an interpretable statistical concept drift detection method for data streams," *Journal of Intelligent Information Systems*, vol. 56, 06 2021.

[35] F. Bayram, B. S. Ahmed, and A. Kassler, "From concept drift to model degradation: An overview on performance-aware drift detectors," Mar 2022.

[36] ——, "From concept drift to model degradation: An overview on performance-aware drift detectors," *Knowledge-Based Systems*, vol. 245, p. 108632, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950705122002854

[37] L. Baier, M. Hofmann, N. Kühl, M. Mohr, and G. Satzger, "Handling concept drifts in regression problems – the error intersection approach," Apr 2020.

[38] C. Duckworth, F. P. Chmiel, D. K. Burns, Z. D. Zlatev, N. M. White, T. W. V. Daniels, M. Kiuber, and M. J. Boniface, "Emergency department admissions during covid-19: explainable machine learning to characterise data drift and detect emergent health risks," *medRxiv*, 2021. [Online]. Available: https://www.medrxiv.org/content/early/2021/06/09/2021.05.27.21257713

[39] F. Doshi-Velez and B. Kim, "Towards a rigorous science of interpretable machine learning," Mar 2017. [Online]. Available: https://arxiv.org/abs/1702.08608

[40] J. Moreno-Torres, T. Raeder, R. Alaiz, N. Chawla, and F. Herrera, "A unifying view on dataset shift in classification," *Pattern Recognition*, vol. 45, pp. 521–530, 01 2012.

[41] I. Khamassi, M. Sayed Mouchaweh, M. Hammami, and K. Ghédira, "Discussion and review on evolving data streams and concept drift adapting," *Evolving Systems*, vol. 9, 03 2018.

[42] M. Gaber, "Advances in data stream mining," *Wiley Interdisc. Rew.: Data Mining and Knowledge Discovery*, vol. 2, pp. 79–85, 01 2012.

[43] C. Rudin, "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead," Sep 2019. [Online]. Available: https://arxiv.org/abs/1811.10154

[44] S. Su, Y. Sun, X. Gao, J. Qiu, and Z. Tian, "A correlation-change based feature selection method for iot equipment anomaly detection," *Applied Sciences*, vol. 9, p. 437, 01 2019.

[45] J. Montiel, "Learning from evolving data streams," 01 2020, pp. 70–77.

[46] A. Van Looveren, J. Klaise, G. Vacanti, O. Cobb, A. Scillitoe, and R. Samoilescu, "Alibi detect: Algorithms for outlier, adversarial and drift detection," 2019. [Online]. Available: https://github.com/SeldonIO/alibi-detect

[47] M. Rodriguez and P. Neubauer, "The graph traversal pattern," 04 2010.

[48] W. Jitkrittum, Z. Szabo, K. Chwialkowski, and A. Gretton, "Interpretable distribution features with maximum testing power," Oct 2016. [Online]. Available: https://arxiv.org/abs/1605.06796

[49] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, and et al., "Pytorch: An imperative style, high-performance deep learning library," Dec 2019. [Online]. Available: https://arxiv.org/abs/1912.01703

[50] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: A system for large-scale machine learning," in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, 2016, pp. 265–283. [Online]. Available: https://www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf

[51] K. Thurnhofer-Hemsi, E. López-Rubio, M. A. Molina-Cabello, and K. Najarian, "Radial basis function kernel optimization for support vector machine classifiers," Jul 2020. [Online]. Available: https://arxiv.org/abs/2007.08233

[52] A. Pesaranghader and H. L. Viktor, "Fast hoeffding drift detection method for evolving data streams," in *Joint European conference on machine learning and knowledge discovery in databases*. Springer, 2016, pp. 96–111.

[53] J. Montiel, J. Read, A. Bifet, and T. Abdessalem, "Scikit-multiflow: A multi-output streaming framework," Jul 2018. [Online]. Available: https://arxiv.org/abs/1807.04662

[54] R. Singhal and R. Rana, "Chi-square test and its application in hypothesis testing," *Journal of the Practice of Cardiovascular Sciences*, vol. 1, 01 2015.

[55] D. Nigenda, Z. Karnin, M. B. Zafar, R. Ramesha, A. Tan, M. Donini, and K. Kenthapadi, "Amazon sagemaker model monitor: A system for real-time insights into deployed machine learning models," Dec 2021. [Online]. Available: https://arxiv.org/abs/2111.13657

[56] D. Alvarez-Melis and N. Fusi, "Gradient flows in dataset space," arXiv preprint, October 2020. [Online]. Available: https://www.microsoft.com/en-us/research/publication/gradient-flows-in-dataset-space/

[57] P. Kourouklidis, D. Kolovos, N. Matragkas, and J. Noppen, *Towards a Low-Code Solution for Monitoring Machine Learning Model Performance*. New York, NY, USA: Association for Computing Machinery, 2020. [Online]. Available: https://doi.org/10.1145/3417990.3420196

[58] B. Fuglede and F. Topsoe, "Jensen-shannon divergence and hilbert space embedding," 01 2004, p. 31.

[59] O. Lipsky and E. Porat, "Approximate matching in the l-infinity metric," *Information Processing Letters*, vol. 105, pp. 138–140, 02 2008.