IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 25 December 2023, accepted 14 January 2024, date of publication 19 January 2024, date of current version 30 January 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3355959

RESEARCH ARTICLE

Concept Drift Detection Based on Typicality and Eccentricity

YURI THOMAS P. NUNES AND LUIZ AFFONSO GUEDES

Post-Graduate Program in Electrical and Computing Engineering, Technology Center, Federal University of Rio Grande do Norte, Natal 59078-970, Brazil

Corresponding author: Yuri Thomas P. Nunes (yuri.thomas.053@ufrn.br)

This work was supported in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior—Brazil (CAPES)—Finance Code 001.

ABSTRACT Many applications and fields produce a vast quantity of time-relevant or continuously changing data which may represent new phenomena. This data stream behavior is known as Concept Drift. The need to efficiently and accurately process online data streams is a current need in many areas. Concept drift is a cause of performance degradation of classical machine learning approaches. It is necessary to address the concept drift to deploy real-world applications fed by data streams. This work presents a perspective of Concept Drift Detector (CDD) application to empower a data stream classifier in a real-world scenario followed by the proposal of Concept Drift Detector based on Typicality and Eccentricity Data Analytics (TEDA-CDD). Our method employs two models in monitoring the data stream in order to keep the information of a previous concept whereas monitoring the emergence of a new concept. The models are considered to represent two distinct concepts when the intersection of data samples are significantly low, described by the Jaccard Index. TEDA-CDD is compared to known methods from literature in experiments using synthetic and real-world datasets simulating real-world applications. In these experiments, TEDA-CDD performs comparably in terms of accuracy against well-established algorithms whereas presenting higher memory efficiency.

INDEX TERMS Classification on data stream, concept drift detector, data stream, supervised learning, TEDA.

I. INTRODUCTION

Data stream processing is a growing area where applications and fields produce a vast quantity of data. These data are time-relevant or continuously changing, representing new phenomena. Therefore, it is essential to use evolving techniques to process the data stream and these techniques must be wary of concept drifts. In this scenario, determining when the data stream has changed enough to demand readjustments for a real-time application is essential.

A common approach is to define a Concept Drift Detector (CDD) to monitor the data stream and determine when concept drift occurs to prevent inappropriate data processing. There are three major categories of CDD: supervised, unsupervised and semi-supervised [1]. Supervised CDD techniques assume that the ground-truth label of a data sample is known immediately after a prediction. In unsupervised CDD, the prediction feedback is delayed or does not exist. A semi-supervised CDD has access to a small number of ground-truth labels enabling the combination of supervised and unsupervised approaches.

A CDD can also be categorized multidimensional or unidimensional. The multidimensional approaches can capture more detailed information on each concept by leveraging the model's complexity. Unidimensional methods captures less detailed information on each data concept whereas it is far simpler and less resource-demanding. The trade-off between these two approaches is relevant to the application performance in terms of computation time, memory usage and model complexity.

In recent years, several approaches have been proposed for unsupervised concept drift detection. One such method, NN-DVI, utilizes the nearest neighborhood concept to compare densities between the reference and detection

The associate editor coordinating the review of this manuscript and approving it for publication was Berdakh Abibullaev.

windows [2]. Another approach, FAAD, is designed to detect sequence-anomalies in multidimensional sequence data streams that are prone to concept drift [3]. It employs information theory concepts to select features and reduce redundancy, reducing the workload associated with high dimensionality. Anomaly detection uses models built with random feature sampling to generate scores, which are then compared to a user-defined threshold. A comparison between the proportion of anomalies in the reference and detection windows indicates the occurrence of concept drift.

Furthermore, UDetect detects concept drift in activity data streams using a supervised classifier model as a reference to compare unlabeled data instances [4]. Similarly, the algorithm SQSI-IS [5] is based on SQSI [6]. SQSI relies on a supervised offline trained model to provide scores to unlabeled data (detection window) and training data (reference window). Divergence in score densities indicates a concept drift. SQSI-IS introduces an additional step of instance selection for the unlabeled data using the Kolmogorov-Smirnov Test.

MD3 is an approach that monitors the decision boundaries of an SVM classifier to detect concept drift [7]. It looks for density changes within the boundary region, which indicates concept drift, triggering a classifier retrain. On the other hand, the algorithm DDAL utilizes active learning to determine relevant instances in the incoming data stream [8]. DDAL uses these instances to calculate the maximum and minimum densities and constructs a range with the same values from the reference window. This range is then compared to a user-defined threshold to determine if a concept drift has occurred.

Despite the discussed methods, the literature has a higher concentration of supervised concept drift detection papers [9] and unsupervised concept drift detection research area is considered unexplored [10]. In this context, this work focuses on efficient unsupervised unidimensional concept drift detection approaches for its performance and realistic approach. It is essential to have a performative method to process data stream due to its sample-wise nature. A sample must be processed before the next arrives. Also, the unsupervised approach addresses the realistic fact of concept drift detection: the true label of a data sample is unknown and maybe never becomes available.

Concept Drift Detector based on Typicality and Eccentricity Data Analytics (TEDA-CDD) is a concept drift detector based on TEDA, a framework for data analytic leveraging on typicality and eccentricity [11]. It is devised for unsupervised scenarios where the ground-truth label of the incoming samples from the data stream are unavailable. Furthermore, TEDA-CDD is an unidimensional CDD, which means that each feature of the data stream must be individually monitored by distinct TEDA-CDD instance. It monitors a data stream feature by using two TEDA-based models and comparing them at each sample arrival. One model is more resistant to change whereas the other models the most recent data samples. The models are compared using the Jaccard Index based on characteristics intrinsic to TEDA. The principal novelties are the use of TEDA for modeling the concepts and Jaccard Index for comparing the models. The proposed CDD presents a competitive performance compared to other unsupervised CDDs and more memory efficiency using up to three times less memory.

In this paper, we present a novel unidimensional unsupervised Concept Drift Detector, which utilizes TEDA as a concept modeling tool, offering an efficient balance between time and memory consumption. To enhance its capability to detect new concepts, we introduce a forgetting factor associated with the TEDA model. The detection of emerging concepts is achieved using the Jaccard index. Our novel model is thoroughly evaluated through experiments simulating real-world data stream processing scenarios using a data stream classifier. Additionally, we delve into an in-depth discussion of the essential characteristics and behavior of the data stream classifier under consideration. In this paper, we present a novel unidimensional unsupervised Concept Drift Detector, which utilizes TEDA as a concept modeling tool, offering an efficient balance between time and memory consumption. To enhance its capability to detect new concepts, we introduce a forgetting factor associated with the TEDA model. The detection of emerging concepts is achieved using the Jaccard index. Our novel model is thoroughly evaluated through experiments simulating real-world data stream processing scenarios using a data stream classifier. Additionally, we delve into an in-depth discussion of the essential characteristics and behavior of the data stream classifier under consideration.

The following section, Section II, presents a realistic description of concept drift detection on data stream processing. The description of the TEDA-CDD is in Section III. The discussion on experiments and results using the proposed CDD is in Section IV. Finally, the paper concludes in Section V with arguments defending the proposed approaches, limitations regarding the TEDA-CDD applications, and future works exploring the concepts in this paper.

II. PROBLEM DESCRIPTION

Data Streams are sources of non-stationary and theoretically infinite data. They are present in many real-world applications across many areas, for example, network monitoring [12], [13], [14], [15] environment monitoring [12], [16], [17], health monitoring [18], [19], [20], [21], finance [15], [18], sensoring [15], [19], [22], [23], [24], social network [22], [25], [26], scientific production [18], [22], [27], management [12], [18], [28], telecomunications [15], cyber security [19], [29], [30], [31], to name a few. These applications are known for evolving through seasonality and tendency. Atypical events may also affect how these data sources change over time. In the context of data stream processing, the changes over time of the data streams are concept drift.

These potential applications have restrictions regarding the use of data available regarding size and aging. Some applications present challenges to storing and processing

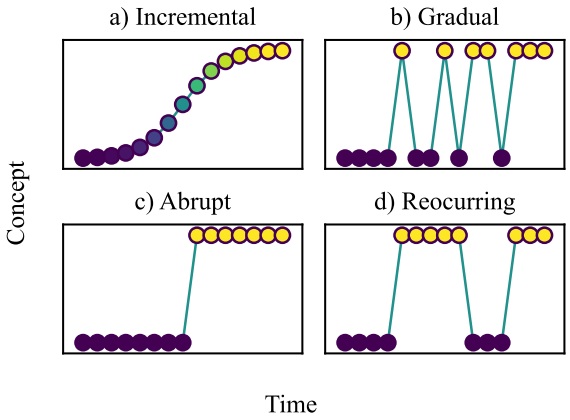

FIGURE 1. Concept drift types: a) Incremental; b) Gradual; c) Abrupt; d) reoccurring.

data due to time constraints and data growth rate. In some applications, even if storing data is mandatory, older data becomes irrelevant as newer data changes and represents new concepts. Generally, data stream processing techniques must quickly and efficiently process data samples and adapt to any concept drifts. Another common characteristic of data stream processing techniques is to process each data sample once. This last characteristic is known as single pass and is a consequence of data availability restrictions on data streams [32].

Concept drift is a relevant aspect of data stream processing. Since a concept drift represents a change in the data stream, it can no longer be considered stationary [33]. Therefore, algorithms and techniques that assume stationarity in data are ineffective as the data stream evolves, demanding new algorithms to process data streams effectively.

There are two known taxonomies regarding concept drift. One classifies the concept drift depending on how the concept changes over time. The second classifies the concept drift based on the relation between features and target. Regarding change over time, concept drifts can occur in four ways: abrupt, gradual, incremental, and reoccurring. Figure 1 illustrates concept drift types. Abrupt concept drift is an abrupt change in the occurrence of concepts in consecutive samples. Gradual concept drift is composed of abrupt concept drifts that gradually increase the new concept occurrence probability while decreasing the old concept occurrence probability. In contrast, an incremental concept drift is a smooth transition between two persons where a previous person transforms into a newer person going through many intermediate states. Finally, in the reoccurring concept drift, concepts keep reappearing in the evolving data stream over time.

The learning process of a machine learning model is analogous to estimating a conditional probability density function between a target variable, y, and a feature vector, $\mathbf{X}$, as in Equation (1) [34].

$$P(y|\mathbf{X}) = \frac{P(y)P(\mathbf{X}|y)}{\sum_y P(y)P(\mathbf{X}|y)} \tag{1}$$

In this context, Equation (2) represents a concept drift in a data stream between two time instants.

$$P_{t_0}(\mathbf{X}, y) \neq P_{t_1}(\mathbf{X}, y) \tag{2}$$

where $P_{t_0}(\mathbf{X}, y)$ denotes the joint distribution at time t_0 between features, $\mathbf{X}$, and target, y, variables. A concept drift can be classified as real or virtual based on the relation between features and target. In a real concept drift the way to represent a concept changes over time even if the features remain stationary. Equation (3) represents the condition for a real concept drift.

$$P_{t_0}(y|\mathbf{X}) \neq P_{t_1}(y|mathbf) \tag{3}$$

whereas a virtual concept drift occurs when the incoming data distribution change without affecting the $P(y|\mathbf{X})$. The condition for a virtual requires $P(y|\mathbf{X})$ remains stationary whereas $P(\mathbf{X})$ changes as in Equation (4).

$$\left[P_{t_0}(y|\mathbf{X}) = P_{t_1}(y|\mathbf{X})\right] \wedge \left[P_{t_0}(\mathbf{X}) \neq P_{t_1}(\mathbf{X})\right] \tag{4}$$

The distinction of real and virtual concept drift is meaningful in supervised learning whereas in unsupervised learning one can assume that any concept drift is virtual.

Due to data stream processing restrictions, a classical Machine Learning technique faces obstacles in training and maintaining good performance. For instance, data can not be considered stationary, and concept drift degrades the model performance making them obsolete over time. Specifically, in supervised techniques, data streams do not provide the correct label at the prediction time, delaying the model adaptation for when and if the data stream provides a class. However, they are usable if it is possible to mimic the stationary condition on data stream segments or if incremental learning approaches are employed.

Therefore, any machine learning model must be adaptable as the concepts evolve when processing data streams. A sample, batch, or concept drift can update the model. When updating sample-wise or batch-wise, the model update occurs as soon as data is available. It is blind to performance degradation. Using concept drifts to determine when to update the model is more efficient. Concept drift is a cause of performance degradation. To enable update by concept drift, a CDD is needed.

Using a CDD enables the deployment of Machine Learning models in real applications. Considering the classification task, a Data Stream Classifier may use a CDD to prevent performance degradation. This strategy allows the model to efficiently and quickly update internal parameters to adapt to new concepts. The following subsections deepen the discussion on Data Stream Classifier and CDD concepts while highlighting the arguments to develop the TEDA-CDD.

A. DATA STREAM CLASSIFIER

A Data Stream Classifier must have at least three components to perform the task in a real-world application.

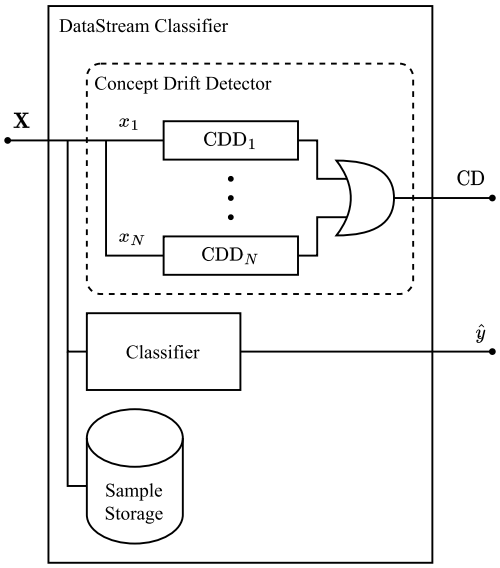

FIGURE 2. Data stream application perception.

- Classifier model
- Concept Drift Detector
- Data Sample Storage

The first component is a classifier model to provide a label based on the known features. As discussed earlier, a CDD is necessary to efficiently and effectively update the classifier model. Finally, a data sample storage for retraining. Figure 2 illustrates a real-world Data Stream Classifier.

A realistic assumption on data stream application is that a data stream exists before the application. It implies that there are available data from the data stream (pre-deploy data), which enables offline training. Once online, the model can be used for prediction and retrained when needed. Therefore it is possible to avoid cold starts of Data Stream Classifiers and leverage this strategy in updating the model. Figure 3 illustrates this idea.

To enable the classifier update, a CDD monitors the available data. It is important to note that not the whole data sample may be available at the same time. When processing a data stream in a supervised task, the ground-truth label is unavailable at prediction time. There is no guarantee that the data will receive the actual labels within the time to update the model. Therefore, a supervised concept drift detector is not appropriate for many cases. Waiting for the ground-truth label to perform a concept drift detection enables the model to make predictions on a data stream that suffered a concept drift without raising the alarm. Therefore an unsupervised online strategy must be applied as the first line of defense against model aging.

B. CONCEPT DRIFT DETECTOR

A CDD determines when a data stream has suffered a significant change. These changes are relevant when they invalidate any assumption on the data or invalidate known descriptors. In these cases, it is necessary to take action and correct the assumptions or update the descriptors.

A CDD must monitor the incoming data and process it adequately to determine if a change has occurred. CDDs are classified as supervised, semi-supervised, and unsupervised, depending on the available data and processing strategy. The supervised approach assumes that the ground truth of the task it is assisting is known. A semi-supervised CDD has access to a limited ground truth labels. An unsupervised approach only considers the feature variables.

In a supervised approach, when the ground truth of the task is known, the CDD may monitor descriptors for the target and feature variables. Although, a common strategy is to monitor the model performance in terms of hits.

When the target variables are unconsidered, the CDD is considered unsupervised. It is only possible to monitor the descriptors in these cases. This approach has two main benefits: reduced time to detect a concept drift, and the underlying descriptors are frequently updated to represent the concept instead of increasing an arbitrary performance metric. The major downside is that some concept drifts are invisible for unsupervised approaches, for example, switching two known classes.

Another relevant aspect of a CDD is if it is unidimensional or multidimensional. A unidimensional CDD monitors each feature individually, needing multiple CDDs to monitor a multidimensional feature space. Whereas multidimensional CDDs can monitor the entire multidimensional feature space with only one CDD instance. The main point of a trade-off between unidimensional and bold CDDs is the complexity. Whereas the unidimensional CDD is less complex, the multidimensional CDD may be more accurate. The complexity affects performance in terms of memory, time, and concepts.

Usually, supervised CDDs are unidimensional and only process the hits of a classifier. Unsupervised CDDs are more common in unidimensional approaches by their simplicity and efficiency.

Using a classifier and a CDD to monitor a data stream, a Data Stream Classifier may run online indefinitely with low-performance degradation. The present work focuses on CDD and assumes that any data stream classifier can take advantage of an unsupervised concept drift detector. This work focuses on unidimensional unsupervised CDD to detect concept drifts in incoming data streams.

III. TEDA-BASED CONCEPT DRIFT DETECTOR

Concept Drift Detection is essential for evolving models and traditional deployed models. In realistic scenarios, the detection of virtual concept drifts can occur before real concept drifts since a virtual concept drift does not directly depends on the target feature. Therefore, an unsupervised concept drift detector can detect early changes in the distribution of the input features. It is a principal motivator to propose an unsupervised concept drift detector. Another factor that enables early detection is sample-wise processing, and using

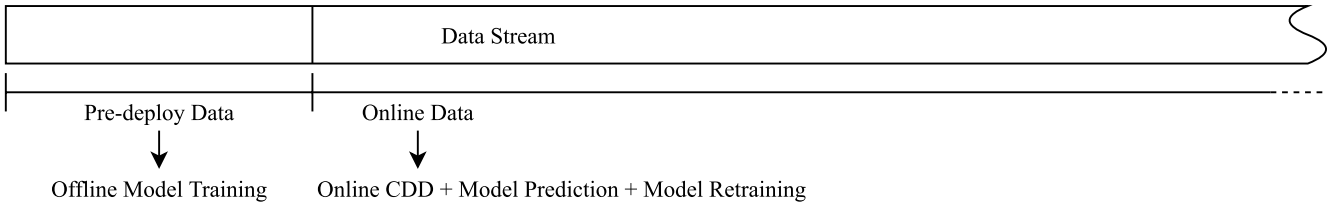

FIGURE 3. Stream Classifier System model.

TEDA as the base model enables sample-wise processing. Therefore, TEDA-CDD is an unsupervised concept drift detector based on TEDA.

TEDA is a framework for data analytics and defines a way to measure how typical or eccentric (atypical) a data sample is to a data set [11]. Typicality measures how representative a sample is to a data set. Inversely, eccentricity measures how abnormal a data sample is to a data set. Both metrics are defined based on a distance or similarity measure to all other data in the dataset. The framework is theoretically independent of any specific distance measure. Although, the implementation can be optimized and problem-specific based on the distance measure utilized.

In this work, the Euclidean distance is used and implicated in a series of definition formulations and specific limitations. For instance, the implementation of TEDA using Euclidean distance defines a hyper-spherical pertinence threshold. A hyper-sphere does not capture covariances in a multidimensional space. It provides an improper pertinence threshold in most cases. The spherical implication indirectly limits TEDA-CDD as a unidimensional concept drift detector. Therefore, in a multidimensional machine learning task, each input feature has an instance of TEDA-CDD.

TEDA-CDD has four essential components: reference data model, evolving data model, detection metric, and reset strategy. Both the reference and evolving models are TEDA based and referred as concept models. The detection metric is based on the Jaccard Index to compare the model ranges. Finally, the reset strategy considers the relevancy of past models and avoids cold restarts.

A. REFERENCE MODEL

The reference model uses the classical definition of TEDA to represent the concept known by the classifier. Whereas the evolving model uses an adaptation to describe the current features state. Therefore, the reference model uses typical data samples to update the internal parameters. The reference model disregards any atypical data sample. This approach considers the reference model incomplete and can change within a tolerable range.

In general, to determine if a data sample is atypical, the eccentricity of the data sample is calculated and compared to the threshold. When the eccentricity is lower than the threshold, the data sample is considered typical, and the reference model is updated. The reference model is not updated if the data sample is atypical to retain the original concept representation.

The eccentricity can be estimated as in (5) when considering the Euclidean distance. Which is based on a recursive estimation of the mean, μ_k, and the variance, σ_k^2, where k is the time index of the data sample, and n is the number of data samples processed.

$$\xi(n) = \frac{1}{n} + \frac{||\mu(n) - x(n)||^2}{n\sigma^2(n)} \tag{5}$$

The values of mean and variance can be recursively estimated by (6) and (7).

$$\mu_R(n) = \frac{n-1}{n}\mu_R(n-1) + \frac{x(n)}{n} \tag{6}$$

$$\sigma_R^2(n) = \frac{n-1}{n}\sigma_R^2(n-1) + \frac{\|\mu_R(n) - x(n)\|^2}{n} \tag{7}$$

The threshold devised to determine if a data sample is eccentric derives from the Chebyshev inequality. The eccentric threshold is a function of a sensitivity parameter and the total number of processed typical samples. Equation (8) describes this threshold where n is the number of samples and m is a parameter to control the threshold sensibility [11]. The sensitivity parameter m has a similar effect of a multiplicative factor of *sigma* ($m\sigma$). Higher values of m make difficult to encounter atypical data samples whereas lower values causes more data samples to be considered atypical. The parameter m is a positive real number and it is indicated values between 1 and 3.

$$\xi(n) \leq \frac{m^2 + 1}{2n} \tag{8}$$

Finally, the reference model is composed of three internal parameters: n, μ, and σ^2. The parameter n is the number of samples considered typical for the model. Parameters μ and σ^2 are updated using Equations (6) and (7), respectively.

B. EVOLVING MODEL

A key difference between the reference and evolving model is that the evolving model is updated by each new data stream sample whereas the reference only by the typical samples. Although, the reference model weighs each sample equally making new concepts hard to detect. For this reason the evolving model uses an adaptation of TEDA.

The adaptation used for constructing the evolving model of TEDA is regarding the ability to focus on recent data samples.

That means older data samples have less contribution to estimating eccentricity. This effect is achieved by introducing a forgetting factor in the update formulas for mean and variance, effectively creating an exponential window.

The forgetting factor enables TEDA to forget past concepts by recursive applying an exponentially decreasing weight as the number of samples grows. The ability to forget an outdated concept enables TEDA to model the current state of a data stream. Equations (9) and (10) describe the update strategy using the forgetting factor, α. The parameter α affects the evolving model as a sentivity parameter. The higher the value of α more important is a data sample to the current concept and the evolving model is more sensible to noise. Conversely, the lower the parameter α more less important is the incoming data sample to the current concept whereas the model is more robust against noise. Acceptable values for α are between 0 and 1. Ideally, closer to 0. It is important to note that using Equations (6) and (7) avoids weighting inconsistency while the value of $1-\alpha$ is higher than $(n-1)/n$.

$$\mu_F(n) = \alpha\mu_F(n-1) + (1-\alpha)x(n) \tag{9}$$

$$\sigma_F^2(n) = \alpha\sigma_F^2(n-1) + (1-\alpha)\|\mu_F(n) - x(n)\|^2 \tag{10}$$

The evolving model is compatible with Equations (5) and (8). This fact makes the concept models comparable in terms of internal parameters, over time.

C. DETECTION STRATEGY

TEDA-CDD indicates a concept drift when the reference and evolving models are sufficiently distinct. In this context, a detection strategy based on the Jaccard Index is proposed. If the strategy indicates a dissimilarity then a concept drift is detected.

The detection strategy for concept drift considers the concept models as representations of sets. Each set is equivalent to a subspace inside the hypersphere defined as the consequence of using Euclidean distance for TEDA. It enables using the Jaccard Index as a similarity measure between the concept models.

The Jaccard Index (JI) is a measure of similarity between sets. Equation (11) defines the JI for two arbitrary sets, A and B. Using JI as similarity measures for the concept models is an obvious solution considering that they represent datasets from the data stream.

$$\text{JI} = \frac{A \cap B}{A \cup B} \tag{11}$$

Effectively, the two sets are similar as their intersection is closer to the union. They are identical if the intersection of the sets is equal to the sets union. Any two sets are distinct when there is no intersection.

The geometric shape in the feature space is the reference to estimate the intersection between the concept models. Ideally, the volume equivalent in the n-dimensional feature space would the estimated for the intersection and union. In this work, for simplicity, the radius and center of each concept model are used to represent the subspace delimiting the set.

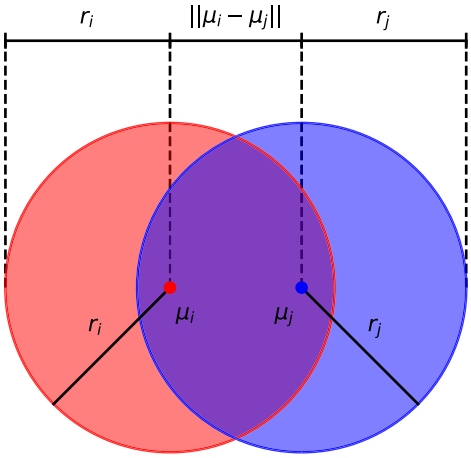

FIGURE 4. Jaccard Index illustrative example.

The center is analogous to the mean. The radius derives from the Chebyshev inequality as the maximum distance a data sample can be from the center in the Equation (12).

$$r(n) = m\sigma^2(n) \tag{12}$$

where $r(n)$ is the radius of a given concept model at instant k, m is the sensitivity parameter, and $\sigma^2(n)$ is the variance.

Combining the fact that TEDA effectively defines hyper-spherical decision boundaries when using the Euclidean distance and Equation (8) it is possible to simplify the JI based on the radius of the hyper-spheres as in Equation (13).

$$\text{JI} = \frac{r_R + r_F - d_{RF}}{r_R + r_F + d_{RF}} \tag{13}$$

This enables to detect a concept drift regardless the number of data samples. A concept drift occurs if the JI is lower than a given threshold, JT. The JT defines a limit of similarity between similar and dissimilar and the higher the value of JT more sensible to divergence is the CDD. A lower JT makes the CDD tolerate higher levels of divergence between reference model and evolving model. Possible values of JT are between 0 and 1. Considering the JI, a JT of 0.5 delimits that two models that share less the half of their data samples are dissimilar.

The main drawback is that the model should have a strategy to avoid nuisance while too few data samples are processed. In the other hand, the main advantage, is that if two models share a great number of samples before diverging in parameters, the decision is not weighted to not raise the alarm of concept drift. Figure 4 illustrates the concepts to devise the detection strategy.

This parameter is crucial for the detection. Ideally, it would be tuned used using an optimization search. It can also be adjusted dynamically using concept drift detection metrics [35]. Unfortunately, it is out of the scope of this initial work with TEDA-CDD.

D. TEDA-CDD RESET

When a concept drift is detected TEDA-CDD is reset to start monitoring the current concept. To avoid a cold restart, the information on the evolving model is used to update the reference model while a new evolving model is created from scratch. It is achieved by implementing Equations (15), (16), and (14).

$$n_R = \begin{cases} n_F, & \frac{n-1}{n} \leq \alpha \\ \left\lfloor \frac{1}{1-\alpha} \right\rfloor, & \frac{n-1}{n} > \alpha \end{cases} \tag{14}$$

$$\mu_R(n_R) = \mu_F(n_F) \tag{15}$$

$$\sigma_R^2(n_R) = \sigma_F^2(n_F) \tag{16}$$

After the reset, the detection will not trigger, while the evolving model does not process enough new data samples. This behavior avoids nuisance detection due to noise and high variance in the early data samples.

E. OVERALL ALGORITHM

In Figure 5 is presented the flowchart of a data stream classifier for a generic CDD. Considering the application of TEDA-CDD, the steps of updating the reference and evolving model are performed in the Run CDD process. The detection strategy is applied in the CD Detected decision and TEDA-CDD is reset if a concept drift is detected at Reset CDD process. This flowchart connects each main component of TEDA-CDD to clearly present the whole functionality of the detector into a data stream classifier. In Algorithm 1, we also provided an algorithm in pseudocode implementing the proposed strategy.

Regarding memory usage, the proposed algorithm does not apply a time window in the same way as the comparison algorithms. TEDA-CDD only stores a fixed-size set of descriptors (n, μ, α, σ) from the data stream, so for an n-dimensional data stream, it would use n sets of descriptors. As a result, the memory complexity is $O(n)$.

Algorithm 1 Data Stream Classifier Algorithm

Input: Pre-deploy dataset, m, α, JT

```
model = new Classifier().fit(preDeployDataset)
CDD.setup(preDeployDataset, m, α, JT)
dataStorage = new DataStorage()
N = 1/(1−α)
X = yield
while true do
  ŷ = model.predict(X)
  for i = 0 to |X| do
    detected = false
    x = X[i]; μ, σ, n = CDD[i].reference
    ξ = 1/n + ||μ−x||²/(nσ²)
    if ξ ≤ (m²+1)/(2n) then
      n = n + 1
      Use Eq. (6) and (7) using μ, σ, n and x
    end if
    μ, σ, n_F, α = CDD[i].evolving
    if 1 − α > (n_F − 1)/n_F then
      Use Eq. (6) and (7) using μ, σ, n_F and x
    else
      Use Eq. (9) and (10) using μ, σ, α and x
    end if
    Calculate JI using Eq. (13)
    detected = detected or (JI > JT and n ≥ N)
  end for
  if detected then
    dataStorage.addOnlineLabels()
    model = new Classifier().fit(dataStorage)
    for i = 0 to |X| do
      Update CDD[i].reference (Eq. (14), (15), (16))
      CDD[i].evolving = new EvolvingModel()
    end for
    dataStorage.flush()
  end if
  dataStorage.push(X)
  X = yield ŷ, detected
end while
```

IV. EXPERIMENTS

In this section, we discuss the experiments and results which indicate that TEDA-CDD is viable and efficient in terms of memory and performance. Initially, the methods used for comparison are presented. Following, the metrics and simulation strategy are discussed. Then, the sythetic datasets experiments are discussed in Subsection IV-A followed by real-world datasets discussed in Subsection IV-B.

The experiment simulates using a concept drift detector processing a data stream in real-time. Besides TEDA-CDD, ADWIN, KSWIN and Page-Hinkley are used to compare the performance. ADWIN maintains a variable-length window of recent data stream samples and detects drift by comparing the distributions of two sub-windows within the window [36]. The KSWIN [37] concept drift detector uses a sliding window divided into two sub-windows: reference and detection windows. The r most recent data samples compose the detection window, whereas sampling from the remaining sliding windows composes the reference window. Then, the Kolmogorov-Smirnoff test indicates when sub-windows data distributions are statistically different. Page-Hinkley [38] monitors CUMSUM metrics of each feature to measure the expected increase or decrease of feature value. For an increasing scenario, a drift is detected when the difference between expected increase and minimum expected increase exceeds a given threshold. All methods must process a minimum number of data samples prior concept drift test after each reset. The selected methods are unsupervised strategies to detect concept drift to ensure comparison fairness. Also, the River python package [39] has implementations of the selected algorithms. The parameters used for the benchmark methods are listed below according to the used library.

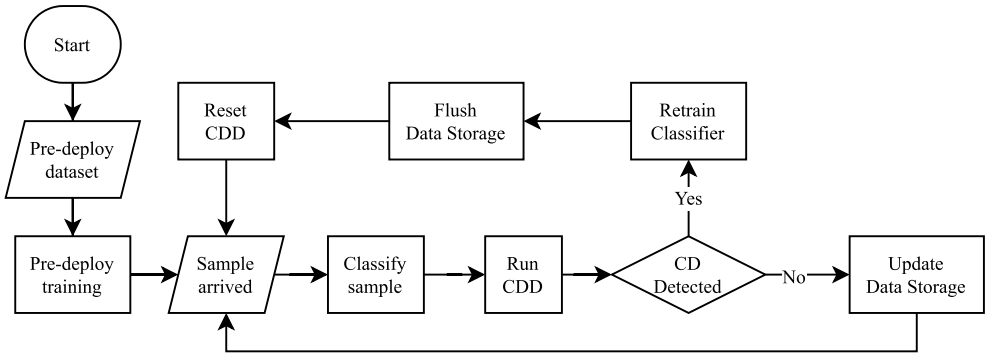

FIGURE 5. Data stream classifier running flowchart.

- ADWIN:
  - delta: 0.002
- KSWIN:
  - alpha: 0.005
  - window_size: 100
  - stat_size: 30
- Page-Hinkley:
  - min_instances: 30
  - delta: 0.005
  - threshold: 50.0
  - alpha: 0.9999
  - mode: both

The evaluation process basis are the ideas of Figures 2, 3 and 5. The Naive Bayes classifier is used as the model and trained with the pre-deploy data as an initial batch. Then the CDD and the classifier start processing the data stream. The sample storage stores the streaming data for future model updates. Whenever concept drift occurs, the model is updated using the data available in the sample storage, the sample storage dumps the previous data, and the concept drift detector resets.

In this experiment, CDD performance and model performance are related. When the CDD correctly detects concept drifts, the model does not suffer performance degradation due to frequent training or false alarms. Therefore, the model performance through the data stream is also a metric of the employed CDD performance. The experiment uses three strategies for performance measuring: prequential, sliding, and holdout [40]. The prequential and sliding strategies measure the classifier performance, whereas the holdout provides a reference baseline.

There are two key points in using prequential scoring. First, it allows using all data stream samples as test and train samples. When a new data sample arrives, the model makes a prediction for scoring and then a model update using the data sample. Second, the most recent score is the accumulated score for all data samples processed, differently to holdout and sliding, which use a subset of the data stream samples. Equation (17) defines the estimation of the prequential accuracy at instant k of the data stream where the function $1(x)$ is the indicator function, $\hat{y}$ is the predicted label, and y is the true label.

$$P_{\text{acc}}(k) = \frac{1}{k}\sum_{i=0}^{k} 1(\hat{y}_i = y_i) \tag{17}$$

In sliding scoring, the performance metrics only consider the data samples into a sliding window of size w. It provides a performance score unbiased by previous data samples (data samples out of the sliding window). Equation (18) defines the estimation of the sliding accuracy for a window of size w at instant k.

$$S_{\text{acc}}(k, w) = \frac{1}{w}\sum_{i=k-w}^{k} 1(\hat{y}_i = y_i) \tag{18}$$

The holdout scoring strategy divides the incoming data samples into two groups: train and test. The model updates using the train set and the test set to evaluate the model's performance. It provides a strong separation between train and test samples in contrast to prequential scoring but reduces the number of training samples. Similarly to the sliding scoring, the holdout scoring provides a performance score not biased by older data samples (data samples out of the current holdout batch). Equation (19) defines the estimation of the holdout accuracy of a given test batch, b, of size w_b where $\hat{y}_i^b$ and y_i^b are, respectively, the i-th predicted label and i-th true label of the b-th test batch.

$$H_{\text{acc}}(b) = \frac{1}{w_b}\sum_{i=0}^{k} 1(\hat{y}_i^b = y_i^b) \tag{19}$$

To use the holdout reference it is need to split the data into batches and for this the drift length of the synthetic datasets is used. In this context, the holdout accuracy was only calculated for the synthetic experiments. Since the classifier used for calculating the holdout accuracy uses the drift length as batch size the models are not presented with concept drift and therefore do not present performance degradation by concept drift or use a CDD. It is important to note that the holdout accuracy is not used to estimate the performance of the methods but to give a baseline for reference and is also

expected that the holdout accuracy is consistently higher than the prequential or sliding accuracy.

In order to provide a meaningful reference for prequential accuracy, the holdout reference was accumulated. Accumulating the holdout accuracy makes it comparable with prequential accuracy in the sense of all previous data sample affect the current score. The accumulated holdout accuracy for the i-th batch uses all test subset from the first batch up to the i-th, inclusive. In this context, the accumulated holdout accuracy is also biased by older data samples presenting the global accuracy for data stream.

The last metric of interest of a CDD is the memory usage along the data stream processing measured in bytes (B). In this context, memory usage is measured at each data sample processing. Memory usage only considers the CDD since the models use the same classifier, and the stored data does not directly affect the detection. The less memory usage, the better.

A. SYNTHETIC DATASETS

The synthetic benchmark used is the Non-stationary Environments Archives (NEA) [41]. It is a collection of non-stationary datasets. The majority is of synthetic data composed of moving Gaussian distributions. There are a variety of concept drift patterns that the majority can be considered incremental.

The NEA Benchmark is relevant because of three characteristics: extensive, complex, and determined. It is extensive as it presents various datasets with multiple behaviors. NEA expresses its complexity through the evolution and interaction between concepts and datasets dimensionality. And finally, it is well-defined as it describes the data and behavior patterns between concepts.

Each dataset presents a unique pattern and a drift duration as listed in Table 1. The drift duration is used to setup the first pre-deploy training of the classifier and to determine the holdout batches. In the pre-deploy data only half of the drift duration is used to build the batch whereas the remaining drift duration is used as online data. To determine the holdout batches the data stream is divided into segments with the drift duration length. Each segment is divided in half for training batch and testing batch (holdout batch) producing a score value for each segment.

The paramaters used for TEDA-CDD in this experiment are: $m = 3$, $\alpha = 0.9655$ and $JT = 0.93$. The other methods use the default parameters from the library.

Figure 6 present the prequential accuracy for data-sets 2CDT and UG_2C_3D. The holdout reference shows the accuracy metric for a classifier with the knowledge of drift duration and therefore performs better than any methods and indicates the upper limit in performance. The remaining algorithms present consistent performance among them. In the case of dataset 2CDT, the algorithms had a significantly inferior performance than the reference, whereas for dataset UG_2C_3D all algorithms' performances were similar.

TABLE 1. NEA datasets descriptions.

Dataset	Classes	Features	Samples	Drift length
1CDT	2	2	16,000	400
2CDT	2	2	16,000	400
1CHT	2	2	16,000	400
2CHT	2	2	16,000	400
4CR	4	2	144,400	400
4CRE-V1	4	2	125,000	1,000
4CRE-V2	4	2	183,000	1,000
5CVT	5	2	40,000	1,000
1CSurr	2	2	55,283	600
4CE1CF	5	2	173,250	750
UG_2C_2D	2	2	100,000	1,000
MG_2C_2D	2	2	200,000	2,000
FG_2C_2D	2	2	200,000	2,000
UG_2C_3D	2	3	200,000	2,000
UG_2C_5D	2	5	200,000	2,000
GEARS_2C_2D	2	2	200,000	2,000

TABLE 2. Final accuracy of model.

Dataset	ADWIN	KSWIN	PH	TEDA-CDD	Reference
1CDT	0.9954	0.9982	**0.9991**	0.9990	0.9986
2CDT	0.8427	0.8942	0.9456	**0.9504**	0.9694
1CHT	0.9863	0.9935	0.9939	**0.9946**	0.9947
2CHT	0.7881	0.8344	0.8848	**0.8889**	0.9095
4CR	0.2510	**0.9998**	0.9942	**0.9998**	0.9999
4CRE-V1	0.2596	0.9660	0.9377	**0.9794**	0.9870
4CRE-V2	0.2665	**0.9223**	**0.9223**	0.9201	0.9287
5CVT	0.8204	0.8366	0.8775	**0.8968**	0.8880
1CSurr	0.7997	0.9735	0.9796	**0.9819**	0.9851
4CE1CF	0.9592	**0.9773**	0.9274	0.9737	0.9784
UG_2C_2D	0.4535	**0.9609**	0.9606	0.9602	0.9621
MG_2C_2D	0.9020	**0.9174**	0.9173	0.9145	0.9202
FG_2C_2D	0.8777	**0.9371**	0.9336	0.9320	0.9413
UG_2C_3D	0.9420	0.9508	**0.9515**	0.9492	0.9538
UG_2C_5D	0.9313	**0.9353**	0.9342	0.9305	0.9389
GEARS_2C_2D	**0.9583**	0.9575	0.9582	0.9563	0.9587

The same behavior is noticeable in Figure 7, which show the sliding accuracy for datasets 2CDT and UG_2C_3D. The 2CDT presents a higher variance and a tendency of the compared methods to be averagely below the reference. The UG_2C_3D has a lower variance and a tendency to follow the reference.

These remarks are confirmed by Table 2. This table lists the final accuracy achieved by each method for all datasets. In bold is the highest accuracy value for the dataset disregarding the reference accuracy. In this experiment, TEDA-CDD and KSWIN have the highest accuracy in 7 datasets. This performance shows that TEDA-CDD is competitive in detecting concept drifts.

In terms of memory, TEDA-CDD presents a consistent superiority using less memory for all datasets. Although, it is important to highlight two points. First, ADWIN and KSWIN are window-based techniques, making this an unfair comparison. Second, the Page-Hinkley technique is unidirectional, only capable of detecting a concept drift that increases the values of a feature. Therefore, its implementation needs two instances to monitor a data stream feature. The second instance monitors the decrease in values of the data stream.

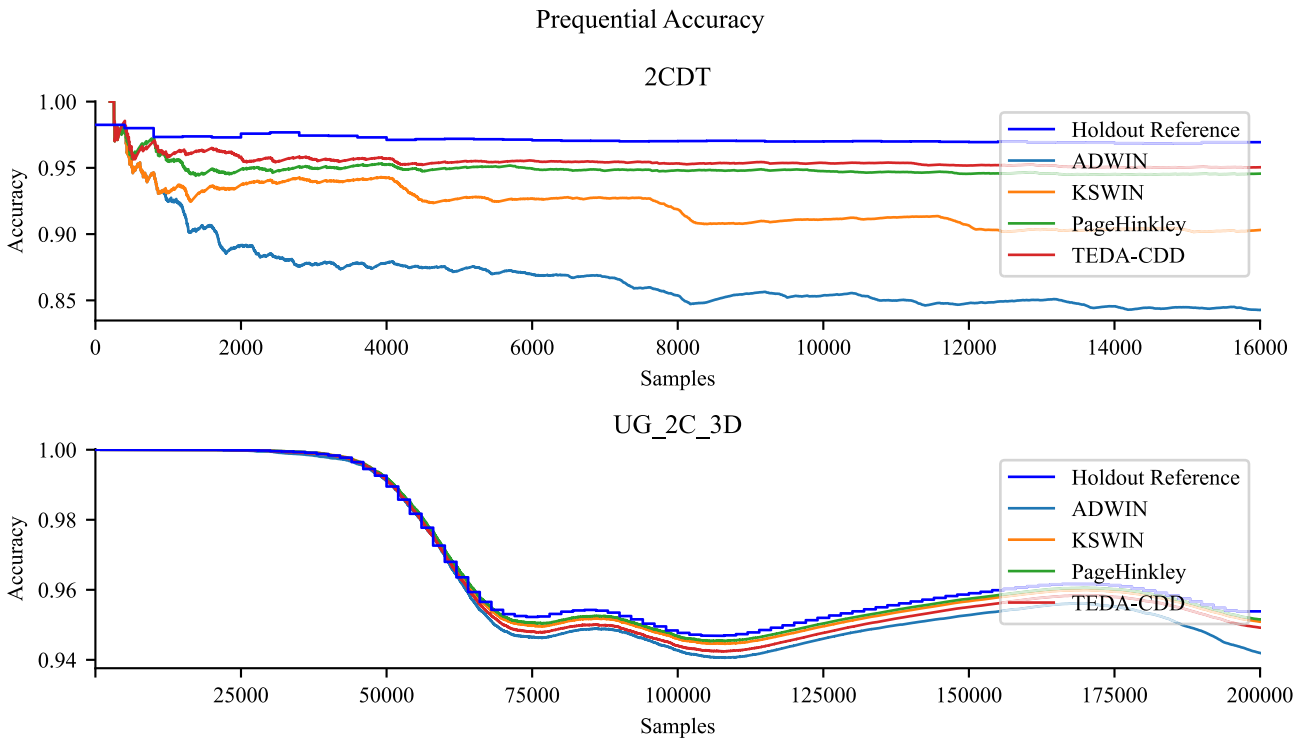

FIGURE 6. Prequential accuracy for synthetic datasets.

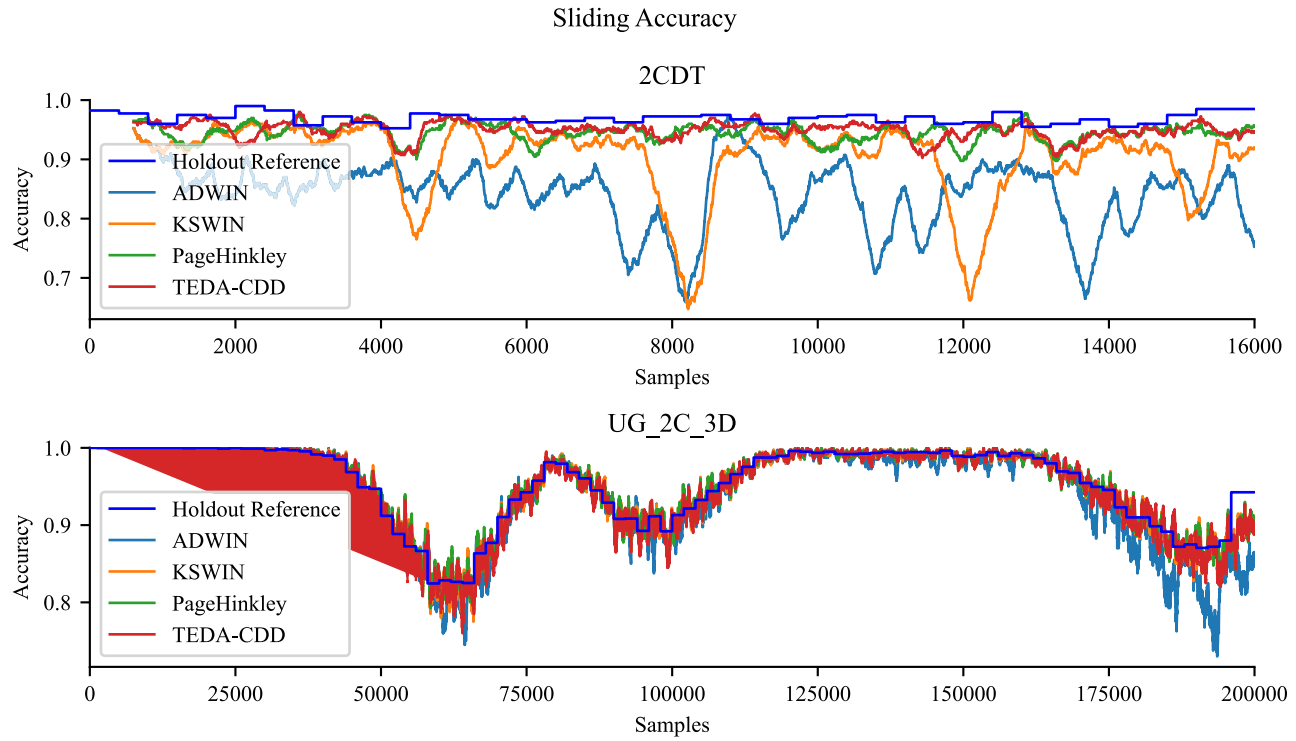

FIGURE 7. Sliding accuracy for synthetic datasets.

B. REAL-WORLD DATASETS

The real-world benchmark is composed by three datasets obtaining from the River package and one the the NEA. From River, the datasets **credit_card**, **electricity** and **phishing** are selected. From NEA, the only real-world dataset provided is selected, **keystrokes**.

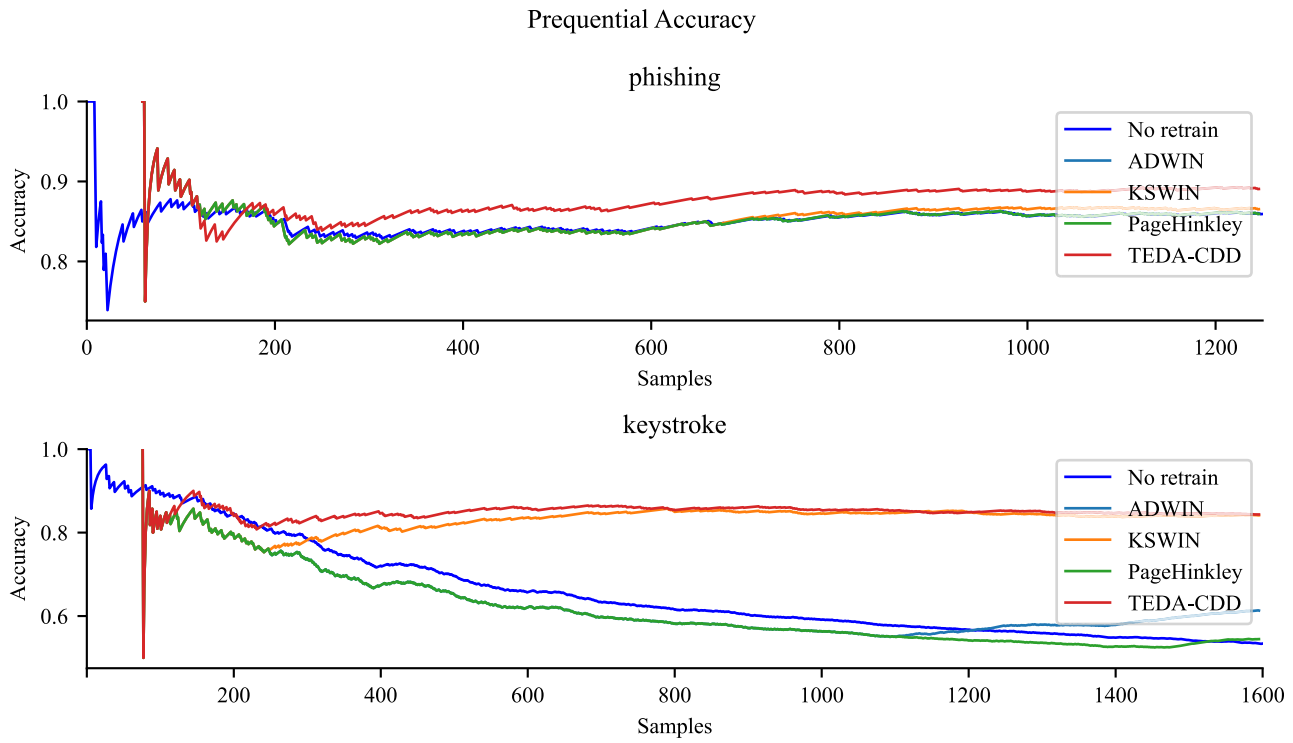

FIGURE 8. Prequential accuracy for real datasets.

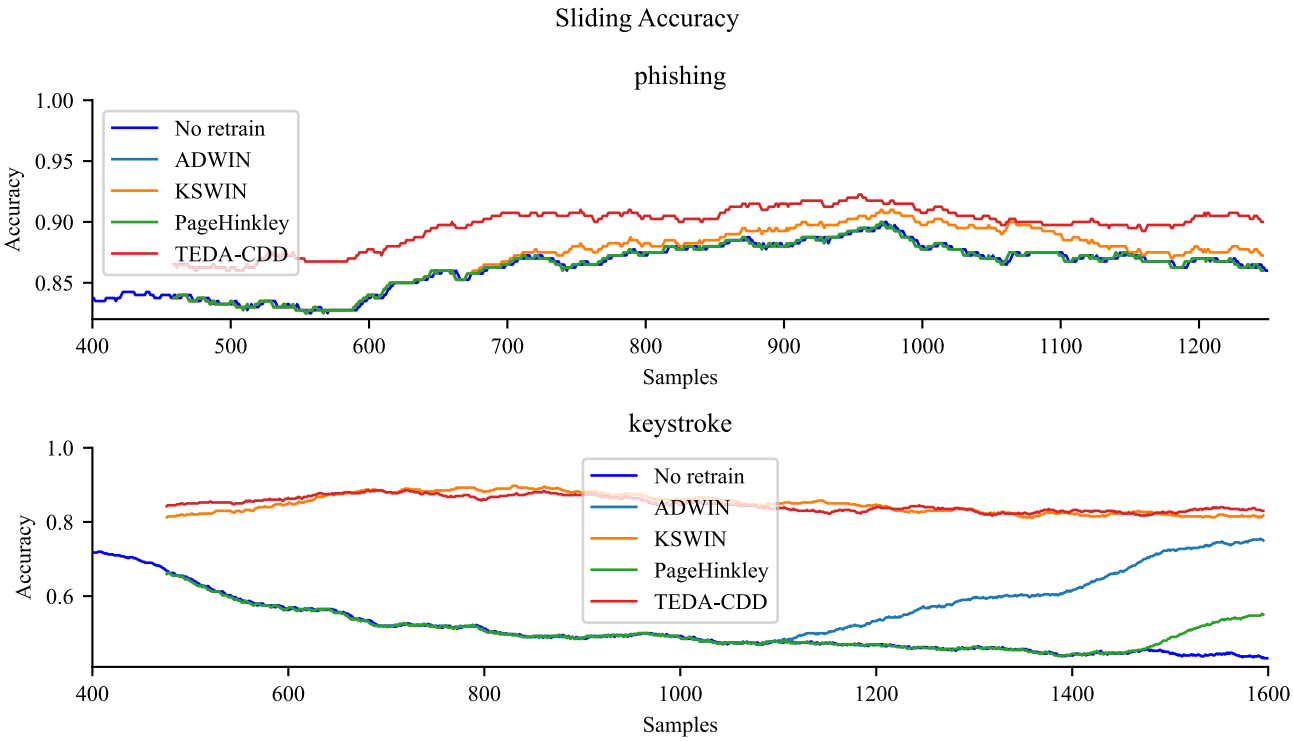

FIGURE 9. Sliding accuracy for real datasets.

- **credit_card** is an anonymized and preprocessed dataset with data from credit card transactions. The goal of this dataset is to detect fraud. It has 28 PCA extracted features, a time index feature, the value of the transaction and target feature. The time index is not used since it presents a monotonic

TABLE 3. Mean memory usage in B (bytes).

Dataset	ADWIN	KSWIN	Page-Hinkley	TEDA-CDD
1CDT	3852.20	15618.36	3447.18	**3296.11**
2CDT	3611.50	15135.91	3458.55	**3295.56**
1CHT	3836.43	15365.19	3447.98	**3295.99**
2CHT	3628.72	15315.21	3458.13	**3295.74**
4CR	4612.64	15529.61	3443.39	**3296.15**
4CRE-V1	4589.64	15456.80	3462.03	**3296.18**
4CRE-V2	4656.23	15531.34	6923.61	**3296.19**
5CVT	3611.43	15315.00	3460.57	**3295.99**
1CSurr	3912.80	15583.78	6909.76	**3296.20**
4CE1CF	4643.44	15460.23	3441.76	**3296.20**
UG_2C_2D	4342.09	15588.45	3460.24	**3296.12**
MG_2C_2D	4217.18	15601.36	3460.40	**3296.20**
FG_2C_2D	4439.74	15480.80	3458.82	**3296.20**
UG_2C_3D	5975.81	22759.68	5192.24	**4944.20**
UG_2C_5D	9956.80	36933.42	8647.19	**8240.27**
GEARS_2C_2D	4675.29	15550.91	3467.88	**3296.13**

TABLE 4. Real datasets descriptions.

Dataset	Classes	Features	Samples
credit_card	2	29	284,807
electricity	2	6	41,437
keystroke	2	10	1,600
phising	2	9	1,250

increasing feature and does not represent concepts directly.

- **electricity** is a dataset for an Australian Electricity market. The task of this dataset is to determine if the electricity cost wil increase or decrease. It has 8 features besides the target feature, two beign time related and, therefore, not used in the experiment.
- **keystroke** is a subset from a dataset from volunteers typing a password. It has 10 features and a 4 class target feature. This version of the dataset was obtained from NEA.
- **phishing** is a dataset composed from data of websites which are classified and phishing or not. It has 9 features and a two classes target feature.

Table 4 presents the number of classes, features and samples of each dataset.

In this experiment the pre-deploy data is composed by 5% of the samples of each dataset. The paramaters used for TEDA-CDD are: $m = 3.3$, $\alpha = 0.9666$ and $JT = 0.85$. The other methods use the default parameters from the library.

Since they are real-world datasets, it is unreasonable to consider a known periodicity or concept drift time instants. In this context, the evaluation based on the classifier performance from the synthetic scenario is valid. Table 5 lists the final accuracy of each method for each real-world dataset. Table 6 lists the mean memory usage in bytes for the real-world dataset experiment. Again, TEDA-CDD has similar performance in terms of accuracy and lower memory usage.

Using data from all experiments, we constructed a critical difference diagram for accuracy and one for memory use, Figures 10 and 11, respectively. We can extract from them

TABLE 5. Final model accuracy on real-world datasets.

Dataset	ADWIN	KSWIN	PH	TEDA-CDD	No retrain
credit_card	0.9945	**0.9985**	**0.9985**	0.9984	0.9816
electricity	**0.7638**	0.7534	0.7399	0.7169	0.6844
keystroke	0.6132	0.8414	0.5447	**0.8434**	0.5337
phising	0.8594	0.8653	0.8594	**0.8905**	0.8592

TABLE 6. Mean memory usage in B (bytes) on real-world datasets.

Dataset	ADWIN	KSWIN	Page-Hinkley	TEDA-CDD
credit_card	58280.87	189964.58	49875.77	**47675.98**
electricity	10764.11	37238.82	10262.09	**9802.79**
keystroke	18328.52	67439.48	17235.32	**16436.43**
phising	16780.21	69499.74	15586.49	**14785.56**

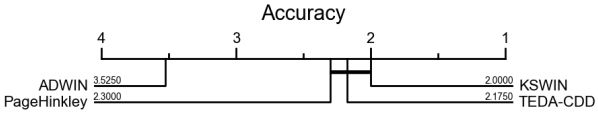

FIGURE 10. Critical difference diagram for accuracy.

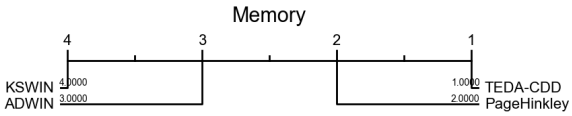

FIGURE 11. Critical difference diagram for memory.

TABLE 7. Parameters sensitivity space.

Parameter	Start	End	Step
m	1	3	0.2
α	0.5	0.9	0.05
JT	0.5	0.95	0.05

that TEDA-CDD performance is equivalent to PageHinkley and KWSIN in terms of accuracy and is the isolated best when considering memory usage.

A straightforward experiment illustrates the sensitivity of the parameters wherein Table 7 designates sets of values corresponding to each parameter. Subsequently, all datasets underwent processing and measuring of the overall accuracy for every parameter combination. The outcomes of this experimentation were employed to construct a parallel coordinates graph. The findings reveal a discernible level of parameter tolerance. Predominantly, the results manifest an accuracy hovering around 0.82, with infrequent instances exhibiting values below 0.8. Importantly, no distinct parameter range was identified as causing any decline in performance.

Finally, the accuracy and memory usage indicate that TEDA-CDD is efficient in terms of memory and accuracy for synthetic and real-world datasets. TEDA-CDD presents the same level of accuracy with much less memory consumption compared to KSWIN. Whereas Page-Hinkley performs closer to TEDA-CDD in terms of memory and accuracy-wise, the implementation complexity (managing two instances with

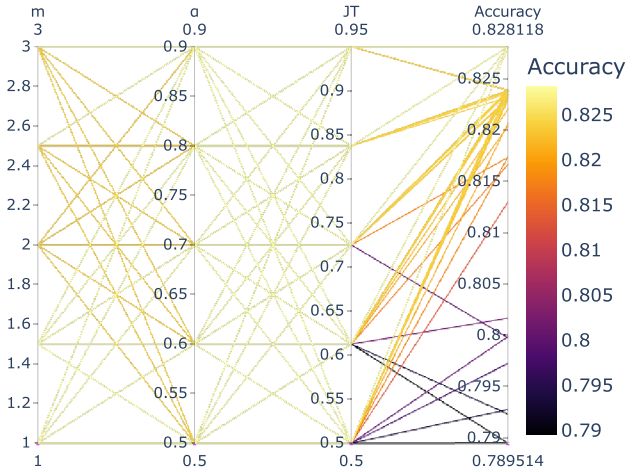

FIGURE 12. Parameter sensitivity analysis.

data sample preprocessing) and lack of generality in the base algorithm subjectively place TEDA-CDD in a better position.

V. CONCLUSION

Our assumptions on data stream processing make our simulations closer to real machine learning applications. In this scenario, unsupervised concept drift detection enables early detection and safer handling. And in this scenario, TEDA-CDD presents a state-of-the-art performance comparable to a consistent method as KSWIN while having lower complexity (in terms of parameters and theoretical concepts) and lower memory usage. Therefore, TEDA-CDD is a competitive approach to concept drift detection.

The known limitations of TEDA-CDD are related to the use of Euclidean distance as a metric of similarity and the fully unsupervised approach. Use Euclidean distance forces spherical models that are conceptually simple because they depend heavily on the mean and variance parameters. The fully unsupervised approach ignores the offline setup of the data stream processing algorithm and online labeling.

In future work, we plan to expand TEDA-CDD to process multidimensional data with the same performance as the one-dimensional approach. Using Mahalanobis distance makes it possible to create ellipsoidal models instead of spherical models to describe concepts. And in a parallel effort, we will propose a more realistic approach where exists known labels at training time and unknown at prediction time. In our understanding, this approach is semi-supervised and enables the modeling and monitoring of known concepts in an unsupervised data stream. Also, we intend to investigate the effects of dynamically adjust TEDA-CDD parameters in response to data stream changes.

REFERENCES

[1] I. Khamassi, M. Sayed-Mouchaweh, M. Hammami, and K. Ghédira, "Discussion and review on evolving data streams and concept drift adapting," *Evolving Syst.*, vol. 9, no. 1, pp. 1–23, Mar. 2018.

[2] A. Liu, J. Lu, F. Liu, and G. Zhang, "Accumulating regional density dissimilarity for concept drift detection in data streams," *Pattern Recognit.*, vol. 76, pp. 256–272, Apr. 2018.

[3] B. Li, Y.-J. Wang, D.-S. Yang, Y.-M. Li, and X.-K. Ma, "FAAD: An unsupervised fast and accurate anomaly detection method for a multi-dimensional sequence over data stream," *Frontiers Inf. Technol. Electron. Eng.*, vol. 20, no. 3, pp. 388–404, Mar. 2019.

[4] S. A. Bashir, A. Petrovski, and D. Doolan, "A framework for unsupervised change detection in activity recognition," *Int. J. Pervasive Comput. Commun.*, vol. 13, no. 2, pp. 157–175, Jun. 2017.

[5] A. G. Maletzke, D. M. dos Reis, and G. E. A. P. A. Batista, "Combining instance selection and self-training to improve data stream quantification," *J. Brazilian Comput. Soc.*, vol. 24, no. 1, pp. 1–17, Dec. 2018.

[6] A. G. Maletzke, D. M. dos Reis, and G. E. A. P. A. Batista, "Quantification in data streams: Initial results," in *Proc. Brazilian Conf. Intell. Syst. (BRACIS)*, Brazil, Oct. 2017, pp. 43–48.

[7] T. S. Sethi and M. Kantardzic, "Don't pay for validation: Detecting drifts from unlabeled data using margin density," *Proc. Comput. Sci.*, vol. 53, pp. 103–112, Jan. 2015.

[8] A. F. J. Costa, R. A. S. Albuquerque, and E. M. dos Santos, "A drift detection method based on active learning," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2018, pp. 1–8.

[9] A. S. Iwashita and J. P. Papa, "An overview on concept drift learning," *IEEE Access*, vol. 7, pp. 1532–1547, 2019.

[10] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang, "Learning under concept drift: A review," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 12, pp. 2346–2363, Dec. 2019.

[11] P. Angelov, "Anomaly detection based on eccentricity analysis," in *Proc. IEEE Symp. Evolving Auto. Learn. Syst. (EALS)*, Dec. 2014, pp. 1–8.

[12] L. Zhang, J. Zhao, and W. Li, "Online and unsupervised anomaly detection for streaming data using an array of sliding windows and PDDs," *IEEE Trans. Cybern.*, vol. 51, no. 4, pp. 2284–2289, Apr. 2021.

[13] V. L. Cao, M. Nicolau, and J. McDermott, "Learning neural representations for network anomaly detection," *IEEE Trans. Cybern.*, vol. 49, no. 8, pp. 3074–3087, Aug. 2019.

[14] X. Miao, Y. Liu, H. Zhao, and C. Li, "Distributed online one-class support vector machine for anomaly detection over networks," *IEEE Trans. Cybern.*, vol. 49, no. 4, pp. 1475–1488, Apr. 2019.

[15] M. U. Togbe, Y. Chabchoub, A. Boly, M. Barry, R. Chiky, and M. Bahri, "Anomalies detection using isolation in concept-drifting data streams," *Computers*, vol. 10, no. 1, p. 13, Jan. 2021.

[16] H. Mehmood, P. Kostakos, M. Cortes, T. Anagnostopoulos, S. Pirttikangas, and E. Gilman, "Concept drift adaptation techniques in distributed environment for real-world data streams," *Smart Cities*, vol. 4, no. 1, pp. 349–371, Mar. 2021.

[17] D. J. Hill and B. S. Minsker, "Anomaly detection in streaming environmental sensor data: A data-driven modeling approach," *Environ. Model. Softw.*, vol. 25, no. 9, pp. 1014–1022, Sep. 2010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1364815209002321

[18] J. Lu, A. Liu, Y. Song, and G. Zhang, "Data-driven decision support under concept drift in streamed big data," *Complex Intell. Syst.*, vol. 6, no. 1, pp. 157–163, Apr. 2020, doi: 10.1007/s40747-019-00124-4.

[19] R. N. Gemaque, A. F. J. Costa, R. Giusti, and E. M. dos Santos, "An overview of unsupervised drift detection methods," *WIREs Data Mining Knowl. Discovery*, vol. 10, no. 6, pp. 1–18, Nov. 2020.

[20] A. A. Beyene, T. Welemariam, M. Persson, and N. Lavesson, "Improved concept drift handling in surgery prediction and other applications," *Knowl. Inf. Syst.*, vol. 44, no. 1, pp. 177–196, Jul. 2015, doi: 10.1007/s10115-014-0756-9.

[21] H.-S. Chiang and Z.-W. Wu, "Online incremental learning for sleep quality assessment using associative Petri net," *Appl. Soft Comput.*, vol. 68, pp. 774–783, Jul. 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1568494617304696

[22] A. L. D. Rossi, C. Soares, B. F. D. Souza, and A. C. P. de Leon Ferreira de Carvalho, "Micro-MetaStream: Algorithm selection for time-changing data," *Inf. Sci.*, vol. 565, pp. 262–277, Jul. 2021.

[23] J. Gama and P. P. Rodrigues, "Data stream processing," in *Learning from Data Streams*. Berlin, Germany: Springer, 2007, pp. 25–39, doi: 10.1007/3-540-73679-4_3.

[24] P. Kadlec, R. Grbić, and B. Gabrys, "Review of adaptation mechanisms for data-driven soft sensors," *Comput. Chem. Eng.*, vol. 35, no. 1, pp. 1–24, Jan. 2011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0098135410002838

[25] R. A. Rios, P. A. Pagliosa, R. P. Ishii, and R. F. de Mello, "TSViz: A data stream architecture to online collect, analyze, and visualize tweets," in *Proc. Symp. Appl. Comput. (SAC)*. New York, NY, USA: Association for Computing Machinery, Apr. 2017, pp. 1031–1036, doi: 10.1145/3019612.3019811.
[26] M. Heusinger, C. Raab, and F.-M. Schleif, "Dimensionality reduction in the context of dynamic social media data streams," *Evolving Syst.*, vol. 13, no. 3, pp. 387–401, Jun. 2022, doi: 10.1007/s12530-021-09396-z.
[27] I. Fister, I. Fister, and M. Perc, "Toward the discovery of citation cartels in citation networks," *Frontiers Phys.*, vol. 4, p. 49, Dec. 2016. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fphy.2016.00049
[28] D. Sun, M. Fu, L. Zhu, G. Li, and Q. Lu, "Non-intrusive anomaly detection with streaming performance metrics and logs for DevOps in public clouds: A case study in AWS," *IEEE Trans. Emerg. Topics Comput.*, vol. 4, no. 2, pp. 278–289, Apr. 2016.
[29] T. S. Guzella and W. M. Caminhas, "A review of machine learning approaches to spam filtering," *Expert Syst. Appl.*, vol. 36, no. 7, pp. 10206–10222, Sep. 2009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S09574174 0900181X
[30] A. Abdallah, M. A. Maarof, and A. Zainal, "Fraud detection system: A survey," *J. Netw. Comput. Appl.*, vol. 68, pp. 90–113, Jun. 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804516300571
[31] B. Bobowska, M. Choras, and M. Wozniak, "Advanced analysis of data streams for critical infrastructures protection and cybersecurity," *J. Univers. Comput. Sci.*, vol. 24, no. 5, pp. 622–633, 2018.
[32] S. Mansalis, E. Ntoutsi, N. Pelekis, and Y. Theodoridis, "An evaluation of data stream clustering algorithms," *Stat. Anal. Data Mining, ASA Data Sci. J.*, vol. 11, no. 4, pp. 167–187, Aug. 2018. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/sam.11380
[33] H. Wang and Z. Abraham, "Concept drift detection for streaming data," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2015, pp. 1–9.
[34] J. Gama, I. Žliobaité, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Comput. Surv.*, vol. 46, no. 4, pp. 1–37, Apr. 2014, doi: 10.1145/2523813.
[35] A. Liu, J. Lu, Y. Song, J. Xuan, and G. Zhang, "Concept drift detection delay index," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 5, pp. 4585–4597, May 2023.
[36] A. Bifet and R. Gavaldà, "Learning from time-changing data with adaptive windowing," in *Proc. 7th SIAM Int. Conf. Data Mining*, Apr. 2007, pp. 443–448.
[37] C. Raab, M. Heusinger, and F.-M. Schleif, "Reactive soft prototype computing for concept drift streams," *Neurocomputing*, vol. 416, pp. 340–351, Nov. 2020, doi: 10.1016/j.neucom.2019.11.111.
[38] E. S. Page, "Continuous inspection schemes," *Biometrika*, vol. 41, no. 1, pp. 100–115, Jun. 1954. [Online]. Available: http://www.jstor.org/stable/2333009
[39] J. Montiel, M. Halford, S. M. Mastelini, G. Bolmier, R. Sourty, R. Vaysse, A. Zouitine, H. M. Gomes, J. Read, T. Abdessalem, and A. Bifet, "River: Machine learning for streaming data in Python," 2020, *arXiv:2012.04740*.
[40] J. Gama, R. Sebastião, and P. P. Rodrigues, "On evaluating stream learning algorithms," *Mach. Learn.*, vol. 90, no. 3, pp. 317–346, Mar. 2013.
[41] V. M. A. Souza, D. F. Silva, J. Gama, and G. E. Batista, "Data stream classification guided by clustering on nonstationary environments and extreme verification latency," in *Proc. SIAM Int. Conf. Data Mining*, Vancouver, BC, Canada, Jun. 2015, pp. 873–881.

YURI THOMAS P. NUNES received the B.S. degree in science and technology, the B.S. degree in computing engineering, and the Master of Science degree in electrical and computing engineering from the Federal University of Rio Grande do Norte (UFRN), Natal, Rio Grande do Norte, Brazil, in 2015, 2017, and 2019, respectively. He is currently pursuing the Ph.D. degree in electrical and computing engineering. His previous research interests include the fields of data science, data analysis, and industrial applications.

LUIZ AFFONSO GUEDES received the B.Sc. degree in electrical engineering from the Federal University of Pará (UFPA), Brazil, in 1988, the M.Sc. degree from the Institute Technological of Aeronautic (ITA), Brazil, in 1991, and the Ph.D. degree from Unicamp, Brazil, in 1999. Currently, he is a Full Professor with the Department of Computer Engineering and Automation (DCA), Federal University of Rio Grande do Norte (UFRN), Brazil. He has expertise in fault diagnostic and operational reliability of industrial processes. He has supervised 14 doctoral theses and 39 master dissertations, and he has more than 15 years of experience in coordination of research and development projects, which have resulted in various software solutions in use in industry.