A Probabilistic Approach for Detecting Real Concept Drift

Sirvan Parasteh[a] and Samira Sadaoui[b]
Computer Science Department, University of Regina, Regina, Canada

Keywords: Concept Drift Detection, Real Concept Drift, Data Stream, Synthetic Datasets, Probability Distributions.

Abstract: Concept Drift (CD) is a significant challenge in real-world data stream applications, as its presence requires predictive models to adapt to data-distribution changes over time. Our paper introduces a new algorithm, Probabilistic Real-Drift Detection (PRDD), designed to track and respond to CD based on its probabilistic definitions. PRDD utilizes the classifier's prediction errors and confidence levels to detect specifically the Real CD. In an exhaustive empirical study involving 16 synthetic datasets with Abrupt and Gradual drifts, PRDD is compared to well-known CD detection methods. PRDD is highly performing and shows a time complexity of O(1) per datapoint, ensuring its computational efficiency in high-velocity environments.

1 INTRODUCTION

Nowadays, many real-world applications come with high-velocity and high-volume data, spanning sectors such as e-commerce, healthcare, and finance. The impossibility of storing the entire data for processing necessitates that Machine Learning (ML) algorithms can only view samples once. Ml algorithms assume that both training and unseen samples follow the same distribution. However, the underlying data distributions may shift over time in today's evolving data-generating sources. For example, user purchasing behavior may change due to unpredictable events like the COVID pandemic or new types of products being introduced in the e-commerce market over time. This discrepancy between training and testing data distributions, called Concept Drift (CD), is a significant challenge for researchers (Gama et al., 2014; Webb et al., 2016), because CD degrades prediction quality substantially. The ML models may have learned patterns no longer relevant to the new incoming data. Therefore, in these non-stationary environments, continuous monitoring of the ML models' performance is essential, along with frequent updates to accommodate the newly detected concept. Indeed, the presence of CD in the data stream will majorly impact the predictive models and decision-making tasks. Consequently, understanding and detecting those unpredictable data changes is vital to develop robust adaptation mechanisms. The CD is a complex notion involving several features (Lu et al., 2018), such as the type of change (Real or Virtual) and the transition speed (Abrupt, Gradual, or Incremental). This paper focuses on the Real CD and considers both Abrupt and Gradual shifts.

Several studies examined the performance of numerous CD detection algorithms, such as those conducted in (Gonçalves Jr et al., 2014) and (Barros and Santos, 2018). These studies showed that no single approach consistently excels in all scenarios. Selecting a CD detection method is tied to the application's requirements, including the datasets' characteristics and the ML models being used. Moreover, these studies outlined several limitations, such as sensitivity to the tuning of the parameters, a considerable computational cost, and challenges with complex data. Consequently, there is still a need for more efficient methods capable of handling diverse data types and dynamically adapting to rapidly evolving concepts.

Our paper proposes an efficient solution for detecting Real CD in data streams, named the Probabilistic Real-Drift Detection (PRDD) algorithm. The latter capitalizes on two key aspects of a classifier's performance: (1) the prediction errors and (2) the confidence level in these predictions. The PRDD algorithm is grounded in the formal definition of Real CD, tracking changes in the posterior probability distribution $P(y|x)$ over time to detect instances of drift promptly. PRDD retains a fixed-size moving window of the most recent data, enabling continuous data stream monitoring and updating critical statistical metrics. These metrics include the real drift

[a] https://orcid.org/0000-0001-7642-2654
[b] https://orcid.org/0000-0002-9887-1570

Parasteh, S. and Sadaoui, S.
A Probabilistic Approach for Detecting Real Concept Drift.
DOI: 10.5220/0012378800003636
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 16th International Conference on Agents and Artificial Intelligence (ICAART 2024) - Volume 2*, pages 301-311
ISBN: 978-989-758-680-4; ISSN: 2184-433X
Proceedings Copyright © 2024 by SCITEPRESS – Science and Technology Publications, Lda.

threshold (T_{real}) and the ratio of real drift instances, empowering PRDD to track and adapt to evolving data streams. Our empirical study highlights the high performance of PRDD when compared to five well-established CD detection methods across 16 diverse synthetic stream datasets. Furthermore, PRDD stands out for its computational efficiency. With a time complexity of O(1) per data point, PRDD's computational cost remains invariant with the size of the data stream, making it especially suited for real-time applications that deal with vast volumes of data. Our study's contributions are three-fold:

- We developed the PRDD algorithm based on the formal definition of Real CD. In Real CD scenarios, classifiers often make incorrect predictions, yet exhibit high confidence in these predictions. This behavior aligns with the Bayesian definition of Real CD, where the decision boundary becomes ineffective even though the input data distribution, $P(x)$, remains unchanged. PRDD harnesses the classifier's prediction probabilities as an indicator of its confidence, ensuring a rapid response to CD. The main parameters of the PRDD algorithm were fine-tuned through rigorous experimental testing. Importantly, PRDD has a consistent execution time of O(1) per data sample, emphasizing its suitability for real-time stream processing scenarios.

- Performing a comprehensive empirical study to compare PRDD's performance against traditional CD detection methods (in total five) across 16 diverse synthetic stream datasets, handling Gradual and Abrupt drifts. We also assess the base learner (devoid of any CD detection mechanism) to demonstrate the necessity of detecting CD. The evaluation is carried out as follows: (1) Performance (Accuracy and F1-score) of the seven drift detection algorithms on eight Abrupt datasets, (2) Performance of the seven drift detection algorithms on eight Gradual drift datasets, (3) Aggregated performance analysis using a rank-based statistical test, and (4) Analysis of the average execution time of each CD detection algorithm.

- We conducted an extensive empirical study comparing PRDD with five conventional CD detection methods across 16 synthetic stream datasets, which include both Gradual and Abrupt drifts. Furthermore, we evaluated a base learner without any CD detection to underscore the importance of CD detection. Our evaluation comprised: (1) Analyzing the performance (Accuracy and F1-score) of the seven algorithms on eight Abrupt datasets, (2) Assessing these algorithms on eight Gradual drift datasets, (3) Undertaking an aggregated performance analysis through a rank-based statistics, and (4) Evaluating the average execution time for each CD detection method.

This paper is organized as follows. Section 1 provides a formal definition of the Real CD with Gradual and Abrupt shifts. Section 2 describes well-established CD detection algorithms. Section 3 introduces our Probabilistic Real-Drift Detection (PRDD) algorithm, elaborating on its design and underlying principles. Section 4 conducts an extensive empirical study to validate the performance of PRDD, comparing it with existing CD detection methods across various synthetic stream datasets.

2 REAL CONCEPT DRIFT

We utilize the probabilistic definitions of CD given in (Gama et al., 2014; Hoens et al., 2012; Webb et al., 2017). Based on these definitions, two different CD types have been recognized in the literature: (1) Real drift, where only the learner's decision boundary changes, and (2) Virtual drift, where only the input-feature distribution changes. The Bayesian approach is a popular choice for developing CD detection methods, as it captures the changes in the joint distribution of the features and class labels (Hoens et al., 2012). In this paper, we focus only on the Real CD, indicating that the statistical properties of the target variable change over time. More precisely, Real drift means that the conditional distribution of the target variable $P(y \mid \mathbf{x})$ changed, while there is no change in the distribution of the input features P(x) (Lu et al., 2018):

$$P_t(y \mid \mathbf{x}) \neq P_u(y \mid \mathbf{x}) \quad and\ P_t(\mathbf{x}) = P_u(\mathbf{x}) \tag{1}$$

where x represents a set of feature vectors and y its corresponding target variable, and time u, which is after t, denotes when the data distribution has changed.

The posterior probability distribution is computed using Bayes' theorem as follows:

$$P(y|\mathbf{x}) = \frac{P(\mathbf{x}|y)P(y)}{P(\mathbf{x})} \tag{2}$$

where $P(\mathbf{x}|y)$ is the likelihood of the features given the target class, $P(y)$ is the prior probability of the class label and $P(\mathbf{x})$ is the marginal probability of the features. In the presence of CD, the prior distribution $P(y)$ and likelihood distribution $P(\mathbf{x}|y)$ have changed, leading to a change to the posterior distribution (Gama and Castillo, 2006).

Considering the Real CD definition, a learned concept can remain stable for a period of time and

Table 1: Error-based Concept-Drift Detection Methods.

Method	**Implementation**	**Year**	**#citation**
PH (Page, 1954)	MultiFlow, River	1954	6563
DDM (Gama et al., 2004)	MultiFlow, River	2004	1596
EDDM (Baena-Garcıa et al., 2006)	MultiFlow, River	2006	868
STEPD(Nishida and Yamauchi, 2007)	Git	2007	286
ADWIN (Bifet and Gavalda, 2007)	MultiFlow, River	2007	1549
ECDD (Ross et al., 2012a)	Git	2012	341
EWMA(Ross et al., 2012b)	Git	2012	382
SeqDrift(Pears et al., 2014)	Git	2013	46
HDDM (Frias-Blanco et al., 2014)	MultiFlow, River, Git	2014	271
SEED(Huang et al., 2014)	Link	2014	68
FHDDM(Pesaranghader and Viktor, 2016)	Git	2016	134
RDDM(Barros et al., 2017)	Git	2017	130
FTDD(de Lima Cabral and de Barros, 2018)	Git (Fisher test)	2018	90
MDDM(Pesaranghader et al., 2018)	Git	2018	55
KSWIN(Raab et al., 2020)	MultiFlow	2020	58

then change into another concept in the data stream:

$$Concept_t \neq Concept_u <=> P_t(x|y) \neq P_u(x|y) \quad (3)$$

In addition to the drift types (Real vs. Virtual), CD is characterized by its transition (i.e., speed of change), which has often been categorized as Abrupt, Gradual, and Incremental to express whether the change levels are small or significant. These aspects carry essential information that can be utilized to develop drift-handling mechanisms. Our study focuses on the Abrupt and Gradual changes for Real CD:

Abrupt Drift. In Abrupt drifts, a known concept C_t switches suddenly to another concept C_u, and the progression of change is very rapid. This shift can happen for several reasons, such as the outage of service or failure of a sensor and equipment.

Gradual Drift. The transition between concepts happens slowly, following a Gradual progression of tiny changes. For instance, a slowly degrading part of factory equipment can result in a Gradual drift in the quality of the output parts, or inflation through a period of time can impact models dealing with pricing (Tsymbal, 2004). (Gama et al., 2014) introduced "intermediate concepts" that help to illustrate the transition between the old concept C_i and the new concept C_{i+1}. The intermediate concept can be one of C_t or C_u, which means each sample appearing during the drift period belongs to one of the concepts involved.

3 RELATED WORKS

Since the mid-1990s, researchers have shown great interest in CD (Widmer and Kubat, 1996; Klinkenberg and Renz, 1998), and several methods for detecting changes in the data stream have been developed. Among these methods, error-based methods (supervised) have gained significant attention. These methods utilize the predictive performance as input and apply statistical distribution tests to capture any significant change in the learner's performance. Thus, any fluctuation in the error rate can signal a drift. These methods return the CD locations/timestamps in the data stream. We explored numerous CD detectors and report in Table 1, the most utilized ones. Table 1 presents various CD detection algorithms, including their (1) implementation, (2) year of publication, and (3) popularity based on the citation number. We investigated many methods and reported their implementation using three primary sources: (1) **GitHub** for the public repositories, (2) **Link** for the provided link to the source code, and (3) **Multiflow or River** for the implementation on Scikit-multiflow or River, which are Python-based packages for ML for streaming environments.

Examining various error-based CD detection methods reveals a chronological progression in their development. The Page-Hinkley (PH) method (Page, 1954) marked the beginning of this progression, which has since evolved to include increasingly advanced approaches, such as DDM (Gama et al., 2004), EDDM (Baena-Garcıa et al., 2006), and ADWIN (Bifet and Gavalda, 2007). While some techniques have garnered significant attention regarding the citation count, others have remained less influential. Nonetheless, the diverse landscape of error-based methods offers researchers a wide array of techniques to choose from based on their specific application needs.

Recent trends in the field of CD detection have seen an increased interest in taking advantage of probabilistic methods. For instance, the study (Parasteh and Sadaoui, 2023) introduced a new supervised probabilistic CD detection algorithm called SPNCD. The latter utilizes the Sum-Product Network to learn the joint probability distribution of incoming data in a tractable way. More specifically, SPNCD leverages the predicted probabilities from the SPN model and combines them with the base ML model's predictions to effectively detect drifts (Real and Virtual). However, the SPNCD's dependence on the SPN as an additional model added computational demands.

In this paper, for comparison purposes, we choose the following five methods:

- *ADWIN (Adaptive Windowing)*: This is a detector and estimator that efficiently adapts the length of a window of observations to detect changes in the observable process. The adaptation is based on an online algorithm that maintains the statistical properties of the data stream, allowing a prompt reaction to changes. ADWIN is more efficient for Gradual drifts (Bifet and Gavalda, 2007).
- *EDDM (Early Drift Detection Method)*: This is a supervised detection method that monitors the distribution of distances between consecutive classification errors. EDDM can detect Gradual and Abrupt changes while maintaining low false positive rates and is particularly designed to detect early signs of drifts. (Baena-Garcıa et al., 2006).
- *KSWIN (Kolmogorov-Smirnov Windowing)*: This is a drift detection technique based on the Kolmogorov-Smirnov statistical test. It compares the distributions of two samples from a window of recent observations and triggers alarms upon significant distribution changes, thereby indicating CD.
- *HDDM (Hellinger Distance Drift Detection)*: This detector measures the dissimilarity between two probability distributions using the Hellinger distance, aiming to detect changes in data streams. The method offers two variations, HDDM_A and HDDM_W, with the former more sensitive to Abrupt changes and the latter designed to identify Gradual changes (Frias-Blanco et al., 2014).

4 PROBABILISTIC REAL-DRIFT DETECTION APPROACH

The new approach for detecting real drift operates based on an adaptive probabilistic mechanism that continuously monitors the incoming data stream, capturing and reacting to drift. This mechanism aligns with the formal definition of CD, which signifies changes in the posterior probability distribution, $P(y|x)$, while the data distribution in the input space,($P(x)$, remains consistent over time. The approach, termed *Probabilistic Real-Drift Detection (PRDD)*, emphasizes two key aspects of a classifier's performance: (1) prediction error and (2) confidence in its predictions. The underlying rationale is that during real drift, a classifier's decision boundary may become outdated, leading to an increase in prediction errors. Thus, even with consistent input features, the classifier, confident in its predictions, may misclassify drifted samples due to an irrelevant decision boundary. This misclassification arises when there's a shift in the target variable distribution, despite the input distribution remaining unchanged.

While processing the data stream sample by sample (online learning), PRDD maintains a fixed-size moving window for calculating the drift detection factors, including an adaptive real-drift threshold (T_{real}), the real drift rate computed as the proportion of drifting candidate within the moving window, and a comparison of the real drift rate with a drift alarm threshold (T_{alarm}). The pseudocode of the PRDD approach is given in Algorithm 1. Below, we explain the PRDD parameters and variables:

- $T_{real} \in [0,1]$: T_{real} serves as a threshold marking the learner's confidence in misclassified samples. Any sample misclassified with a confidence exceeding T_{real} is flagged as a potential drift candidate. To dynamically adjust this threshold in response to the most recent changes in the data stream, we employ the Exponential Moving Average of the classifier's prediction probabilities. This method strikes a balance between the classifier's recent and historical prediction performance. For every prediction error, the threshold T_{real} is recalibrated according to the formula $T_{real} = 0.7 \times P_{avg} + 0.3 \times T_{real}$, where P_{avg} denotes the average prediction probability within the current window. Such an adaptive approach ensures that T_{real} consistently mirrors the data stream's prevailing conditions.
- $T_{alarm} \in [0,1]$: T_{alarm} is a predefined alarming threshold utilized to ascertain if a drift is manifesting within the moving window. Grounded on our empirical assessments across 16 datasets,

T_{alarm} is established at 0.47. This threshold assists in pinpointing situations where the real drift rate within the window exceeds a limit that necessitates the recognition of an actual CD. T_{alarm} guarantees a balance, ensuring sensitivity to legitimate drifts while maintaining resilience against incidental noise and minor variations.

- Warmup Threshold: This parameter is introduced to allow the model to warm up or acclimate to the recent state of the data stream. The warmup threshold is been set empirically to let the model observe sufficient data and calibrate its parameters without actively triggering any drift detection.
- Sample Count is used to count the number of processed samples since the beginning of the new concept. When the sample count passes the warmup threshold, the detection mechanism is activated, and When drift is detected, this variable will be reset to 0.
- Window Size: The window size determines the number of recent samples included in the calculations for T_{real} and the drift rate. Given a fixed drift alarm threshold, such as 0.5, the window size has a direct influence on the algorithm's ability to detect changes. A smaller window can make the model overly sensitive, reacting to slight changes or noise as if they were significant drifts. Conversely, a larger window might result in slower detection of rapid drifts. Therefore, selecting the right window size is crucial to ensure timely drift detection while avoiding false alarms caused by noise.

In conclusion, the PRDD method presents a robust solution for real drift detection in streaming data. It capitalizes on the classifier's prediction probabilities and maintains a quick adaptation rate to CD, thereby ensuring reliable performance in dynamic environments. Utilizing an adaptive learner's confidence threshold (T_{real}), a static drift alarm threshold (T_{alarm}), a warm-up period, and an optimal window size, PRDD forms a high-quality mechanism for real drift detection.

5 VALIDATION

To validate the detection capability of our approach, we conduct experiments on a comprehensive set of synthetic stream datasets designed explicitly for evaluating CD detection algorithms. These public datasets were generated using the scikit-multiflow framework to simulate the occurrence of various types of drifts. As these datasets precisely mark the locations of the induced drifts, they serve as ground truth for performance measurement of different CD detection metrics. In the following subsections, we describe the experimental setup, the synthetic datasets, and the obtained performance results in detail.

Algorithm 1: Real CD Detection Algorithm.

```
Require: dataStream (continuous), windowSize = 20, $T_{alarm}$= 0.47, warmupThreshold = 20
Ensure: Drift detection and classifier update
1: Initialize classifier and window parameters
2: sampleCount = 0, $T_{real}$ = 0, realDriftCount = 0
3: for each sample in dataStream do
4:     sampleCount +=1
5:     (*Perform prequential prediction and training*)
6:     Predict the label $y_{pred}$ for the current sample $x$
7:     Calculate the probability $P(y_{pred}|x)$ associated with the prediction
8:     Update the classifier using the true label $y$
9:     if sampleCount > warmupThreshold then
10:        Update window of probabilities with $P(y_{pred}|x)$
11:        if label is incorrect and $P(y_{pred}|x) \geq T_{real}$ then
12:            realDriftCount +=1
13:            Update window of real drifts
14:        end if
15:        (*Calculate average probability*)
16:        $P_{avg} = \frac{1}{windowSize} \sum P(y_{pred}|x)$
17:        if label is incorrect then
18:            (*Update $T_{real}$*)
19:            $T_{real} = 0.7 \times P_{avg} + 0.3 \times T_{real}$
20:        end if
21:        (*Calculate real drift rate*)
22:        $RD_{rate} = \frac{realDriftCount}{windowSize}$
23:        if window is full and $RD_{rate} > T_{alarm}$ then
24:            Signal drift
25:            Reset stats and re-initialize classifier
26:            sampleCount =0
27:        end if
28:    end if
29: end for
```

5.1 Diverse Drift Datasets

To ensure the robustness of our findings, we employ 16 synthetic datasets from the publicly available collection hosted by Harvard Dataverse(López Lobo, 2020). These datasets are designed specifically for CD detection research, encapsulating Abrupt and Gradual drift scenarios. Each dataset consists of 40,000 observations with a balanced binary class distribution and devoid of any noise. Specifically, these datasets manifest four distinct concepts separated by three drifts at predetermined time steps. These transitions span over 1000 instances for Gradual drift datasets, simulating a slow adaptation to the new concepts. The datasets originated from four different stream generators: Sine, Random Tree, Mixed, and Stagger. Each generator contributes to Abrupt and Gradual drifts, enhancing the diversity of our dataset collection. The datasets bear unique characteristics regarding drift types, the number of features, and underlying generating functions.

- **Mixed:** Constructs datasets with four numerical features and adopts two distinct function orders defined as F1 = [0, 1, 0, 1] and F2 = [1, 0, 1, 0].
- **Sine:** Generates datasets with two numerical features using two function orders: F1 = [0, 1, 2, 3] and F2 = [3, 2, 1, 0].
- **Stagger:** Produces datasets with three numerical features based on two function orders: F1 = [0, 1, 2, 0] and F2 = [2, 1, 0, 2].
- **Random Tree (RT):** Creates datasets with two numerical features using two function orders: F1 = [8873, 9856, 7896, 2563] and F2 = [2563, 7896, 9856, 8873].

Each generator produces four different datasets. For example, the Mixed generator constructs MixedF1Abrupt, MixedF2Abrupt, MixedF1Gradual and MixedF2Gradual. Using this diverse range of datasets, our evaluation thoroughly evaluates the proposed algorithm across varying types of CD scenarios.

5.2 Experiment Setup

Our evaluation strategy encompasses a comparative study with five contemporary drift detection algorithms, namely ADWIN, EDDM, KSWIN, HDDM_A, and HDDM_W. We benchmark these models against our real drift detection algorithm on the 16 synthetic datasets. Performance is quantified using three key metrics: Accuracy, F1-score, and execution time, providing a comprehensive overview of each model's predictive capability, the balance between precision and recall, and computational efficiency. With regard to KSWIN, it possesses a degree of nondeterminism stemming from its built-in sampling process. To accommodate this variability, we conduct a series of 10 independent runs for each dataset when testing with KSWIN. The reported results for this method represent the average outcomes of these multiple runs, offering a more reliable measure of its performance.

We adopt the Hoeffding Tree Classifier as the underlying learner for the six drift detection algorithms. This classifier, renowned for its adaptability to high-speed data streams, is a decision tree designed specifically to process data items arriving at fast rates. It serves as an appropriate choice given the dynamic nature of CD and the real-time processing requirements of streaming data.

To better understand the contribution of the drift detection component, we also include a baseline scenario in our experimental setup. This scenario consists solely of the Hoeffding Tree Classifier, devoid of any drift detection mechanism. This baseline allows us to gauge the added value of integrating a drift detector with the classifier. While we expect that combining a classifier with a drift detector generally outperforms a standalone classifier, we focus on analyzing how effectively the proposed method enhances the performance. The performance evaluation of the seven algorithms is structured along four distinct segments:

1. Performance metrics on the eight Abrupt drift datasets.
2. Performance metrics on the eight Gradual drift datasets.
3. Aggregated performance analysis using rank-based statistics.
4. Analysis of the average execution time to evaluate the computational efficiency.

Our comprehensive approach to performance evaluation sheds light on the algorithm's behavior under both Abrupt and Gradual drift scenarios and provides a comparative view of its performance against other algorithms, including computational aspects. The choice to employ a rank-based analysis alongside traditional performance metrics stems from the desire for a well-rounded assessment of the algorithm's capabilities. While direct metrics such as accuracy or F1-score can offer valuable insights, they might sometimes be swayed by the unique properties of individual datasets. For instance, an algorithm could excel on specific datasets because those datasets inherently match the algorithm's assumptions. Moreover, the variability in performance outcomes across diverse datasets can make averaged comparisons less statistically relevant. In contrast, a rank-based analysis provides a better comparative perspective by measuring how often our proposed algorithm outperforms other algorithms, regardless of the absolute performance metrics.

By exploring these different evaluation scenarios, we aim to provide a comprehensive and robust assessment of our proposed algorithm's effectiveness and utility across various drift dynamics and computational constraints.

5.3 Performance Evaluation on Abrupt Drift

Our proposed real-drift detection algorithm underwent an extensive comparative evaluation across eight datasets that exhibit Abrupt drift. The evaluation was set up to ensure a level playing field where all participating algorithms, including our proposed model, were coupled with the Hoeffding Tree Classifier.

5.3.1 Performance Across Abrupt Datasets

Table 2 offers a comparative analysis of several algorithms, focusing on their Accuracy and F1-score metrics across Abrupt drift datasets. At a glance, our proposed PRDD algorithm emerges as a top performer, leading in 6 out of the 8 datasets including MixedF1, MixedF2, RTF2, SineF1, SineF2, and StaggerF2. This consistent performance underscores its strength and reliability in real-drift detection. However, it faced challenges on the other two datasets. On the RTF1 dataset, PRDD is closely rivaled by HDDM_A. While PRDD posts an Accuracy of 80.15%, HDDM_A slightly surpasses it with 80.34%. Also, The StaggerF1 dataset presents a more pronounced deviation. PRDD's Accuracy dips to 91.44%, placing it fifth among the algorithms. Such a result underscores that while PRDD is generally robust, there exist scenarios where its assumptions might not align perfectly with the dataset's characteristics. It's also noteworthy that on this dataset, EDDM shines brightest with an impressive Accuracy of 96.12%, higher than HDDM_A which showed higher competency with PRDD. Other methods like HDDM_W, KSWIN, EDDM and ADWIN showed better performance compare to the base learner, with no detection algorithm, HoeffdingTree. In many cases, HoeffdingTree often showd significantly lower Accuracy compared to other methods which highlithed the importance of implying CD detection algorithms.

Conclusively, while PRDD delivers dominant performance in most settings, there are specific instances requiring further investigation. The overall results, combined with its computational efficiency, position PRDD as a qualified choice. However, understanding the nuances of each dataset and scenario will further enhance its applicability and effectiveness.

5.3.2 Rank Statistics for Abrupt Drifts

In addition to the learner performance metrics, we also analyzed the rank statistics of each algorithm's performance in terms of Accuracy and F1-score across the Abrupt drift datasets. Rank-based evaluation can offer a different perspective, as it aggregates model performance across multiple datasets and illustrates algorithmic consistency and relative performance across diverse datasets. The rank statistics for Accuracy and F1-score across the Abrupt drift datasets are shown in Table 3. For each model, the mean rank and standard deviation (Std. Dev.) are computed. The rank of an algorithm on a dataset is determined by its position in the sorted list of algorithm performances, with rank 1 being the best.

Our proposed algorithm stands out in this rank-based evaluation, securing the top position with a mean rank of 1.75 in terms of Accuracy and 1.69 for F1-score-based ranking. The low standard deviations of 1.39 for both metrics further emphasize PRDD's consistently high performance across the board. In the ranking hierarchy, HDDM_A and HDDM_W trail closely with mean ranks of 2.13 and 3.25, respectively. Following them are KSWIN and EDDM. HoeffdingTree, on the other hand, consistently ranks at the bottom with a mean rank of 7 for both metrics, highlighting the evident advantage of specialized drift detection techniques over more generic methods in stream learning scenarios.

Overall, these results highlight the robustness and superior performance of our proposed real-drift detection algorithm in handling Abrupt CDs. The ensuing sections will further explore the algorithm's performance on Gradual drift datasets and its computational efficiency.

5.4 Performance Evaluation on Gradual Drift

Within this section, we delve into the comparative analysis of algorithmic performance on datasets characterized by gradual drifts, which is a complex and subtle challenge in the domain of data stream learning.

5.4.1 Performance on Gradual Drifts

The performance of the algorithms, when confronted with Gradual real drifts, is a crucial aspect to consider, given the subtlety of these drifts and the consequent difficulty in their detection. To that end, we assessed the seven algorithms across the eight Gradual drift datasets, with respect to Accuracy and F1-

Table 2: Model Performance Metrics (Accuracy and F1-score) Across **Abrupt** Drift Datasets.

	PRDD		**ADWIN**		**KSWIN**		**EDDM**		**HDDM_W**		**HDDM_A**		**HoeffdingTree**	
	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1
MixedF1	**91.19**	**91.20**	89.51	89.54	90.65	90.65	90.43	90.43	91.03	91.03	91.18	91.19	83.35	84.37
MixedF2	**91.22**	**91.22**	89.00	89.00	90.82	90.83	89.91	89.94	91.18	91.17	90.94	90.96	80.84	79.30
RTF1	80.15	77.97	77.44	75.09	79.85	77.76	76.67	74.54	78.15	75.65	**80.34**	**78.56**	73.46	72.05
RTF2	**82.70**	**81.10**	78.45	75.77	80.18	78.28	75.68	73.46	79.30	77.38	81.79	79.89	72.16	71.09
SineF1	**93.04**	**93.05**	91.96	91.96	92.95	92.94	90.58	90.55	93.00	93.00	**93.05**	**93.05**	63.85	59.68
SineF2	**93.19**	**93.19**	91.31	91.34	92.90	92.90	89.93	90.01	93.17	93.15	92.94	92.95	56.29	58.39
StaggerF1	91.44	91.78	93.19	93.50	91.13	91.56	**96.12**	**96.23**	92.02	92.56	94.34	94.59	88.82	89.47
StaggerF2	**97.70**	**97.73**	94.95	94.99	92.65	93.10	97.32	97.35	95.36	95.52	96.44	96.51	92.62	92.57

Table 3: Rank Statistics for Accuracy and F1-score Across Abrupt Drift Datasets.

	Accuracy Rank		**F1-score Rank**	
Model	Mean	Std. Dev.	Mean	Std. Dev.
HoeffdingTree	7.00	0.00	7.00	0.00
ADWIN	5.00	0.93	5.00	0.93
EDDM	4.63	2.00	4.63	2.00
KSWIN	4.25	1.17	4.25	1.17
HDDM_W	3.25	0.89	3.25	0.89
HDDM_A	2.13	0.84	2.19	0.75
PRDD	**1.75**	1.39	**1.69**	1.39

score metrics compiled in Table 4. Our proposed algorithm maintained its superior performance, again dominating in 6 out of 8 datasets including, MixedF1, MixedF2, RTF2, SineF1, SineF2, and StaggerF2. These results provide compelling evidence of the proposed model's proficiency at detecting and responding to Gradual real drifts.

While **PRDD** exhibited superior performance across most datasets, there were instances when other algorithms surged ahead. For instance, in the RTF1 dataset, HDDM_A marginally surpassed **PRDD** in both Accuracy and F1-score. In the StaggerF1 dataset, although HDDM_A achieved the highest accuracy and F1-score, it trailed behind **PRDD**. As observed in the abrupt scenario, no single algorithm consistently dominates in all scenarios, underscoring the potential for future research to delve into algorithmic intricacies and potential areas of enhancement.

HDDM_W, on the other hand, delivered commendable results, securing a solid third position. However, the performance of other methods such as EDDM, ADWIN, and KSWIN was more variable on Gradual datasets. As anticipated, the model without a detector lagged significantly behind its counterparts.

To sum up, our proposed **PRDD** method demonstrated remarkable adeptness at managing Gradual drifts. Yet, the nuanced variations in performance across different datasets emphasize the significance of tailoring algorithm selection to the specific dataset in question. Subsequent sections will provide a more detailed, rank-based comparative analysis to further illuminate these observations.

5.4.2 Rank Statistics for Gradual Drifts

In our efforts to offer a consolidated perspective on the performance of each algorithm under Gradual real drifts, rank statistics were computed across the eight Gradual drift datasets. Rankings were ascertained based on Accuracy and F1-score, where a lower rank implies enhanced performance. The findings are encapsulated in Table 5.

In alignment with our previous observations, **PRDD**, our proposed model firmly holds the top position. It secures an admirable mean rank of 1.63 (with a standard deviation of 1.07) for both Accuracy and F1-score metrics. These data further solidify **PRDD**'s prowess in detecting and adeptly handling Gradual real drifts. The second-best in accuracy is HDDM_A, which registers a mean rank of 2.13. Following closely, HDDM_W clinches the third spot with an average rank of 2.75 for Accuracy. The other algorithms, EDDM, ADWIN, and KSWIN, display a more varied rank distribution, echoing their inconsistent performance across different datasets. Predictably, the HoeffdingTree model, devoid of any detector, languishes at the bottom with a mean rank of 7 for both Accuracy and F1-score.

The results of our rank-based analysis align with and underscore our previous discussions, emphasizing the effectiveness and robustness of PRDD against gradual real drifts in data streams. Figure 1 depicts the rank distribution for each method in the 16 datasets. As highlighted, PRDD consistently achieves the top rank, closely followed by HDDM_A.

5.5 Execution Time Analysis

The processing speed and efficiency of a real-drift detection algorithm are fundamental, equating in importance to its predictive accuracy. Swiftly addressing and adjusting to ongoing data stream alterations are essential features of a leading CD detection algorithm.

Figure 2 reveals that our proposed model consistently demonstrates computational efficiency, clock-

Table 4: Model Performance Metrics (Accuracy and F1-score) Across **Gradual** Drift Datasets.

	PRDD		ADWIN		KSWIN		HDDM_W		EDDM		HDDM_A		HoeffdingTree	
	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1
MixedF1	**88.92**	**88.90**	86.42	86.43	86.35	86.41	88.14	88.17	87.97	87.97	88.61	88.61	83.53	83.96
MixedF2	**88.79**	**88.83**	86.53	86.61	88.01	88.06	88.12	88.09	87.90	87.94	88.37	88.40	81.94	81.41
RTF1	79.33	76.61	77.18	74.25	76.01	74.34	78.61	75.97	78.43	75.89	**79.35**	**76.80**	72.11	70.60
RTF2	**80.19**	**77.98**	78.02	74.76	76.79	75.05	79.04	76.52	76.35	74.59	79.39	76.34	71.45	71.65
SineF1	**90.38**	**90.34**	88.97	88.91	77.82	77.21	90.33	90.26	88.32	88.33	90.03	90.01	64.68	60.06
SineF2	**91.00**	**91.01**	88.70	88.69	74.72	75.35	90.51	90.52	89.57	89.60	90.49	90.50	58.40	58.90
StaggerF1	93.13	93.34	91.55	92.04	92.97	93.22	94.68	94.83	94.46	94.63	**94.78**	**94.96**	87.13	88.14
StaggerF2	**97.66**	**97.67**	92.01	92.34	84.88	83.75	96.86	96.87	95.29	95.37	96.93	96.93	91.08	91.15

Table 5: Rank Statistics for Accuracy and F1-score Across Gradual Drift Datasets.

	Accuracy Rank		**F1-score Rank**	
Model	Mean	Std. Dev.	Mean	Std. Dev.
HoeffdingTree	7.00	0.00	7.00	0.00
KSWIN	5.13	0.84	5.38	0.75
ADWIN	4.88	1.73	4.63	1.67
EDDM	4.5	0.92	4.50	0.92
HDDM_W	2.75	0.71	2.63	0.74
HDDM_A	2.13	0.83	2.25	0.89
PRDD	**1.63**	1.07	**1.63**	1.07

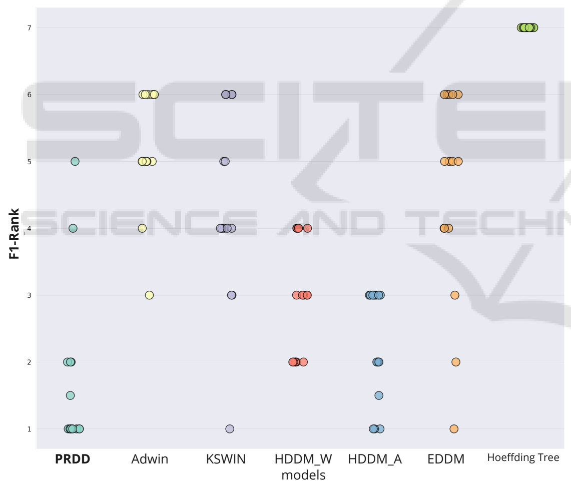

Figure 1: Ranking distribution of algorithms across multiple datasets.

ing an average execution time of approximately 5 seconds across the tested datasets. This level of efficiency is maintained even when compared to a baseline model devoid of a drift detection feature. The precisely built design of our model, with a $O(1)$ time complexity per data point, is essential to its resilient efficiency. PRDD employs a moving window to calculate essential metrics such as mean probabilities and drift ratios. A naive approach would require recalculating these measures for every point within the window with each new data addition, incurring significant computing costs.

In contrast, our model includes an optimization in the form of a ”running sum” technique. The operational dynamics of the ”running sum” technique are as follows:

- **Data Ingestion:** As new data points are received, they are immediately incorporated into the running sum, ensuring real-time updates.
- **Window Saturation:** Once the moving window reaches its capacity, for every subsequent data point, the model seamlessly updates the running sum. This is achieved by subtracting the value of the oldest data point (the one that exits the window) and adding the value of the incoming data point.

This methodological approach eliminates the need for recalculating the sum for the entire window with each incoming data point, considerably mitigating computational overhead. Such an optimized mechanism not only assures prompt updates but also positions our model as a standout choice for real-time applications necessitating immediate responsiveness.

Additionally, updating base learners, such as the HoeffdingTree, to accommodate data changes can be computationally burdensome, especially in the face of drift. Detecting drift early and swiftly adapting to the emerging data patterns is, therefore, paramount to achieving improved computational efficiency. Our model's ability to proactively detect and manage drifts is a significant advantage. It cuts down the high computational costs that come with constantly adjusting to evolving data. Since our model has an $O(1)$ time complexity it responds quickly to changes and maintains consistent performance. This efficiency is retained regardless of how large the incoming data stream becomes.

A comparative analysis with benchmark models is also instructive. For instance, models like HDDM and ADWIN, despite their constant time complexity, register average execution times close to 6 seconds. The EDDM model clocks in at about 7 seconds on average. In contrast, the KSWIN model, bearing a time complexity of $O(w)$ where w is the window size,

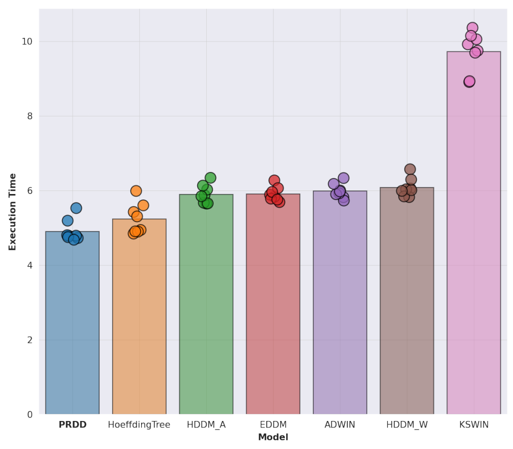

Figure 2: Execution time of models on all the datasets.

records an average execution time of nearly 10 seconds, almost twice that of our proposition.

The differences in these execution times, when all models use a consistent base learner, offer a clear efficiency contrast among the drift detection algorithms. This comparison underscores our method's dual strength in both prediction and computation. Consequently, our proposed model stands out as an optimal choice for handling high-velocity data streams.

6 CONCLUSION AND FUTURE WORK

Predictive models based on historical patterns are susceptible to performance degradation in non-stationary environments where the underlying data distributions shift over time. Therefore, devising algorithms that can effectively capture and adapt to Concept Drift (CD) is crucial. Our proposed Probabilistic Real-Drift Detection (PRDD) algorithm demonstrates excellent performance in identifying real CD with high sensitivity, rendering it a practical and reliable tool for real-world data-stream applications. The PRDD's robustness and adaptability are further evidenced by its consistent performance under various drift dynamics, including Gradual drifts.

Future work presents numerous research directions. Firstly, we plan to investigate CD in real-world applications, an area that currently lacks sufficient exploration in the literature. Specifically, we will focus on credit card fraud, an ever-evolving field. Our aim is to understand the nature and characteristics of CD in this application, considering that CD can occur in both normal data (changes in users' spending habits or e-payment channels) and fraud data (fraudsters updating their strategies in response to new technologies). Such insights will be invaluable in devising even more effective predictive models to tackle CD. Also, we aim to compare our active adaptive learning method to the passive learning method (Sadreddin and Sadaoui, 2022).

REFERENCES

Baena-Garcıa, M., del Campo-Ávila, J., Fidalgo, R., Bifet, A., Gavalda, R., and Morales-Bueno, R. (2006). Early drift detection method. In *Fourth international workshop on knowledge discovery from data streams*, volume 6, pages 77–86.

Barros, R. S., Cabral, D. R., Gonçalves Jr, P. M., and Santos, S. G. (2017). Rddm: Reactive drift detection method. *Expert Systems with Applications*, 90:344–355.

Barros, R. S. M. and Santos, S. G. T. C. (2018). A large-scale comparison of concept drift detectors. *Information Sciences*, 451:348–370.

Bifet, A. and Gavalda, R. (2007). Learning from time-changing data with adaptive windowing. In *Proceedings of the 2007 SIAM international conference on data mining*, pages 443–448. SIAM.

de Lima Cabral, D. R. and de Barros, R. S. M. (2018). Concept drift detection based on fisher's exact test. *Information Sciences*, 442:220–234.

Frias-Blanco, I., del Campo-Ávila, J., Ramos-Jimenez, G., Morales-Bueno, R., Ortiz-Diaz, A., and Caballero-Mota, Y. (2014). Online and non-parametric drift detection methods based on hoeffding's bounds. *IEEE Transactions on Knowledge and Data Engineering*, 27(3):810–823.

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

Gama, J., Medas, P., Castillo, G., and Rodrigues, P. (2004). Learning with drift detection. In *Brazilian symposium on artificial intelligence*, pages 286–295. Springer.

Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., and Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM computing surveys (CSUR)*, 46(4):1–37.

Gonçalves Jr, P. M., de Carvalho Santos, S. G., Barros, R. S., and Vieira, D. C. (2014). A comparative study on concept drift detectors. *Expert Systems with Applications*, 41(18):8144–8156.

Hoens, T. R., Polikar, R., and Chawla, N. V. (2012). Learning from streaming data with concept drift and imbalance: an overview. *Progress in Artificial Intelligence*, 1:89–101.

Huang, D. T. J., Koh, Y. S., Dobbie, G., and Pears, R. (2014). Detecting volatility shift in data streams. In *2014 IEEE International Conference on Data Mining*, pages 863–868. IEEE.

Klinkenberg, R. and Renz, I. (1998). Adaptive information filtering: Learning in the presence of concept drifts. *Learning for text categorization*, pages 33–40.

López Lobo, J. (2020). Synthetic datasets for concept drift detection purposes. *Harv. Dataverse*.

Lu, J., Liu, A., Dong, F., Gu, F., Gama, J., and Zhang, G. (2018). Learning under concept drift: A review. *IEEE transactions on knowledge and data engineering*, 31(12):2346–2363.

Nishida, K. and Yamauchi, K. (2007). Detecting concept drift using statistical testing. In *International conference on discovery science*, pages 264–269. Springer.

Page, E. S. (1954). Continuous inspection schemes. *Biometrika*, 41(1/2):100–115.

Parasteh, S. and Sadaoui, S. (2023). A Novel Probabilistic Approach for Detecting Concept Drift in Streaming Data. In *International Conference on Deep Learning Theory and Applications, Delta, Springer Nature*, pages 173–188.

Pears, R., Sakthithasan, S., and Koh, Y. S. (2014). Detecting concept change in dynamic data streams. *Machine Learning*, 97(3):259–293.

Pesaranghader, A. and Viktor, H. L. (2016). Fast hoeffding drift detection method for evolving data streams. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 96–111. Springer.

Pesaranghader, A., Viktor, H. L., and Paquet, E. (2018). Mcdiarmid drift detection methods for evolving data streams. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–9. IEEE.

Raab, C., Heusinger, M., and Schleif, F.-M. (2020). Reactive soft prototype computing for concept drift streams. *Neurocomputing*, 416:340–351.

Ross, G. J., Adams, N. M., Tasoulis, D. K., and Hand, D. J. (2012a). Exponentially weighted moving average charts for detecting concept drift. *Pattern recognition letters*, 33(2):191–198.

Ross, G. J., Adams, N. M., Tasoulis, D. K., and Hand, D. J. (2012b). Exponentially weighted moving average charts for detecting concept drift. *Pattern recognition letters*, 33(2):191–198.

Sadreddin, A. and Sadaoui, S. (2022). Chunk-based incremental feature learning for credit-card fraud data stream. *Journal of Experimental & Theoretical Artificial Intelligence*, 0(0):1–19.

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

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

Webb, G. I., Lee, L. K., Petitjean, F., and Goethals, B. (2017). Understanding concept drift. *arXiv preprint arXiv:1704.00362*.

Widmer, G. and Kubat, M. (1996). Learning in the presence of concept drift and hidden contexts. *Machine learning*, 23(1):69–101.