A Self-Tuning ensemble approach for drift detection *

Guilherme Y. Sakurai[1], **Bruno B. Zarpelão**[1], **Sylvio Barbon Junior**[2]

[1] Departamento de Computação – Universidade Estadual de Londrina (UEL)
Caixa Postal 10.011 – 86.057-970 – Paraná – PR – Brazil

[2]Department of Engineering and Architecture – University of Trieste (UNITS)
34127 – Trieste – Italy

guilherme.yukio@uel.br, brunozarpelao@uel.br, sylvio.barbonjunior@units.it

Abstract. *Processing data streams is challenging due to the need for mining algorithms to adapt to real-time drifts. Ensemble strategies for concept drift detection show promise, yet gaps in flexibility and detection remain. We propose the Self-tuning Drift Ensemble (StDE) method, which dynamically adapts ensemble structure to stream changes while maintaining a lightweight solution. StDE adjusts the number of base learners through a self-regulating voting system, achieving high detection accuracy. Experiments across various drift scenarios demonstrate the superior performance of our method compared to established baselines.*

1. Introduction

The rapid advances in wireless sensor networks, cloud computing, and big data have led to the widespread use of devices that gather, transmit, and process online data across various domains, such as weather forecasting, network traffic analysis, power grid control, and stock market trading. These applications generate data streams with challenging characteristics, including massive size and high velocity, requiring mining algorithms to meet strict response latency and storage constraints. Moreover, as these data streams evolve over time, algorithms must adapt to new knowledge as it emerges [Cano and Krawczyk 2022, Han et al. 2022].

The dynamic and infinite nature of real-world data streams often leads to concept drifts—changes in data behavior due to seasonal events, system reconfigurations, or unexpected situations. Detecting and handling these drifts is crucial for algorithms to adapt effectively [Han et al. 2022, Komorniczak et al. 2022, Martins et al. 2023]. Concept drifts can be abrupt, gradual, incremental, recurring, sudden, mixed, or blip, each requiring specific detection techniques or hyperparameter configurations.

Designing detectors for multiple drift types is challenging. Simple, independent detectors may fail with complex drifts, as noted by Korycki et al. [Korycki and Krawczyk 2019]. Ensemble-based approaches offer potential solutions, but they also pose challenges. For instance, while ensemble methods can handle various drift types, they may introduce computational overhead and scalability issues [Pérez et al. 2020, Du et al. 2015, Komorniczak et al. 2022]. Abbasi et al.

*This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) – Finance Code 001.

[Abbasi et al. 2021] also pointed out that offline classifiers might slow down real-time adaptation. Consequently, new solutions must be adaptive, maintaining limited memory usage and high accuracy over unbounded, heterogeneous streams.

This work proposes the Self-tuning Drift Ensemble (StDE), an ensemble-based method for concept drift detection. StDE adapts to data stream changes by dynamically adjusting the number and configuration of its base learners based on voting behavior. A drift is detected through a hard voting scheme when a simple majority of base learners agree. StDE operates in two phases: a Warm Start phase, where an ensemble of three detectors self-tunes over a labeled stream, and an Online phase, where StDE monitors real-time streams, adjusting the ensemble based on voting results. Unanimous decisions reduce detector numbers to save resources, while divided decisions increase detectors with modified hyperparameters. Tests on homogeneous and heterogeneous drifts demonstrated StDE's robustness and efficiency in maintaining a lightweight structure.

The remainder of this work is organized as follows: Section 2 presents the related work. Section 3 describes StDE. The experimental results are presented and discussed in Section 4. Finally, Section 5 provides the concluding remarks.

2. Related Work

The identification and adaptation to concept drifts in data streams have been explored by single and ensemble-based solutions. This section reviews various techniques, particularly focusing on the identification of concept drifts using ensembles. Algorithms for prediction and adaptation are outside the scope of our proposal, as well as single detectors.

Most ensemble drift detectors rely on supervised learning. [Deckert 2011] introduced the Batch Weighted Ensemble framework, which enhances performance by integrating multiple classifiers with a weighted voting scheme. [Abbasi et al. 2021] used an ensemble of classifiers like K-Nearest Neighbor, Random Forest, and Multilayer Perceptron to efficiently handle concept drift in dynamic social big data streams. [Ramane and Gnanasekar 2022] proposed an ensemble framework combining two detectors and a classifier, using dissimilarity-based and performance-based drift detection methods for effective drift management. [Mavromatis et al. 2023] presented LE3D, an ensemble framework that identifies irregularities in sensor streams using ADaptatinve WINdowing (ADWIN), Page-Hinkley Test (PHT), and Kolmogorov-Smirnov Windowing (KSWIN).

It is important to mention that relying on machine learning algorithms require implementing offline and online phases, creating computationally costly solutions, and even violating stream mining constraints. For example, [Abbasi et al. 2021] noted that reliance on offline classifiers may slow down the adaptation to new data and [Deckert 2011] does not address potential challenges in scalability and computational overhead when using traditional machine learning.

For this reason, we consider that solutions based on stream detectors fit stream constraints more properly. The first work grounded on an ensemble of drift detectors was proposed by [Khamassi et al. 2013]. The authors discussed integrating drift detection methods and proposed the Error Distance-based Approach for Drift Detection (EDIST), a new method that monitors the distance between two consecutive classification errors.

This proposal differs from ours due to the use of a fixed number of detectors with fixed hyperparameters and the combination of classification procedures.

[Du et al. 2015] proposed a selective detector ensemble using methods like Drift Detection Method (DDM), Early Drift Detection Method (EDDM), ADWIN, and Statistical Test of Equal Proportions (STEPD) to identify concept drifts. While this approach maintains model accuracy through supervised learning, it suffers from a high level of false positives due to its early-find-early-report strategy. [Samant et al.] introduced a selective ensemble method that dynamically chooses the most relevant detectors (ADWIN, DDM, and EDDM) based on a machine learning classifier's performance, combining detectors and classifiers to adaptively respond to drifts.

[Korycki and Krawczyk 2019] discussed the Ensemble Drift Detection with Feature Subspaces (EDFS), which independently monitors and detects drifts in data streams, emphasizing simplicity and efficiency. [Pérez et al. 2020] introduced the Statistical Tests Ensemble Detector (STED), which combines Brown-Forsythe, O'Brien, and ANOVA statistical tests with voting strategies to detect changing data distributions. [Komorniczak et al. 2022] proposed the Statistical Drift Detection Ensemble (SDDE), which uses a fixed set of detectors, including a Gaussian Naïve Bayes classifier, to manage sudden, incremental, and gradual drifts.

Supported by the current literature review regarding ensembles of drift detectors, we observed that the flexibility to detect a wide range of drifts while maintaining a lightweight structure requires further research and new proposals. We projected StDE to fulfill these gaps.

3. StDE: Self-tuning Drift Ensemble

In this section, we describe our proposed approach for enhancing drift detectors predictive performance through a voting-based and windowed ensemble method. Our approach, illustrated in Figure 1, consists of two main phases: the Warm Start phase and the Online phase. During the Warm Start phase, the proposed approach performs hyperparameter tuning for DDM and EDDM. Following this, the Online phase involves dynamically managing the drift detectors based on data stream behavior, allowing the ensemble to adapt and perform self-tuning in response to changes. This dynamic and self-tuning capability ensures that the model remains robust and effective in varying conditions.

3.1. Voting and Windowing Mechanisms

We propose a voting ensemble method that incorporates multiple instances of DDM and EDDM. Each detector independently monitors the data stream and votes on whether a concept drift has occurred. The final decision is based on the majority vote among these detectors. By leveraging the strengths of both DDM and EDDM within the ensemble, we can improve the robustness and accuracy of our drift detection process [Lagman et al. 2020].

Our ensemble algorithm for drift detection includes a windowing mechanism to determine if successive bits in a data stream represent the same drift or different ones. The window size is a key parameter that influences the sensitivity and robustness of drift detection. A smaller window detects abrupt changes quickly, enhancing responsiveness but increasing the risk of false positives. A larger window provides more stable detection,

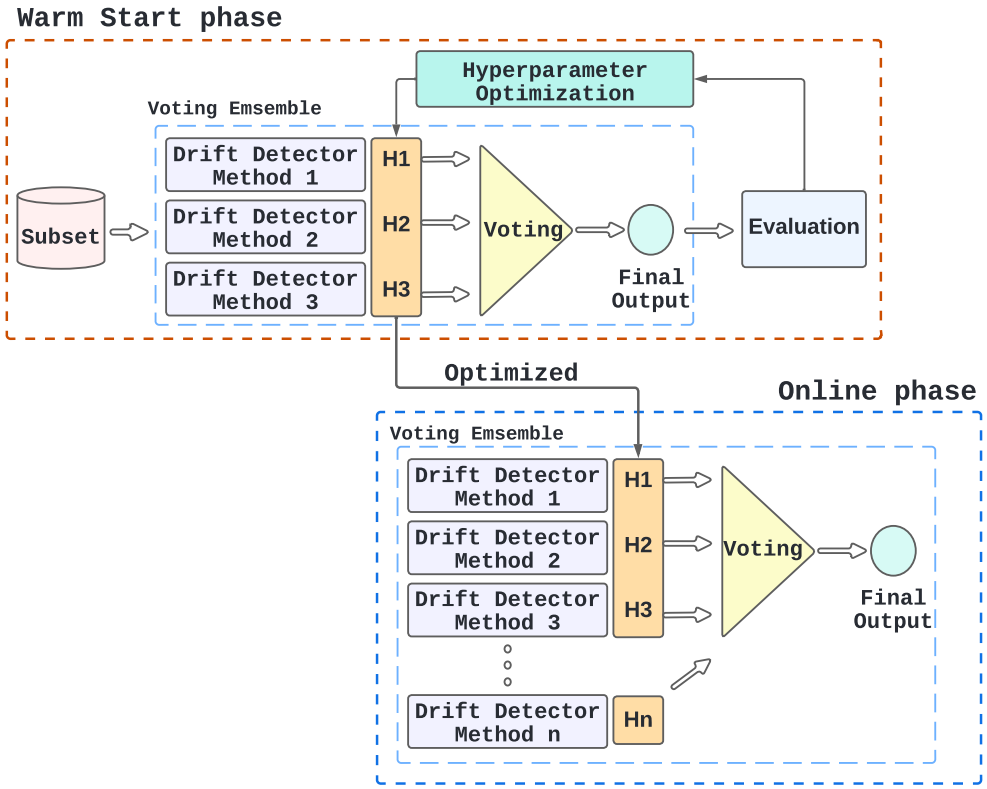

Figure 1. Self-tuning Drift Ensemble (StDE) overview.

reducing false alarms but potentially delaying drift recognition or increasing false negatives. When the first detector in the ensemble detects a drift, the window is initiated, and all drifts identified within this window are considered the same. For example, as shown in Figure 2, with a window size of 3 bits, any detector signaling a drift within this window is considered to indicate the same drift, while detections outside the window indicate different drifts.

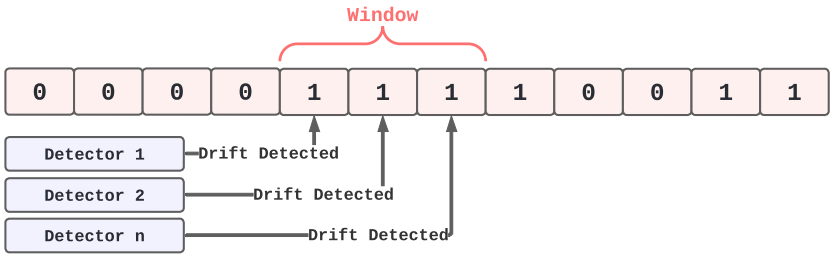

Figure 2. Windowing mechanism overview.

3.2. Warm Start Phase

The Warm Start phase initializes the ensemble operation by tuning the parameters of the drift detection algorithms (base learners). This offline process utilizes a finite data stream. By employing algorithms that have been tuned based on this specific subset of

data streams relevant to the problem, we ensure a robust and well-adapted model for the subsequent online learning phase. The Warm Start phase consists of 4 steps, which are presented as follows:

1. **Hyperparameter Space Definition**: the search space for hyperparameters is defined to explore various configurations for the ensemble components. For both DDM and EDDM, the tuned hyperparameter is the threshold for triggering a warning (α). The value ranges are determined based on the specific characteristics and requirements of the problem at hand.
2. **Hyperparameter Optimization**: this step involves exploring the hyperparameter space to find the optimal settings for the ensemble. We use a Random Search algorithm, which randomly samples and evaluates multiple hyperparameter combinations on a validation subset of the data streams. Random Search is chosen for its simplicity and effectiveness, offering a good balance between performance and computational efficiency. However, other hyperparameter optimization methods could also be employed [Silva et al. 2024].
3. **Performance Evaluation**: each hyperparameter combination is assessed based on its performance in detecting concept drifts, measured through the F1-score metric.
4. **Selection of Hyperparameter Values**: the combination of hyperparameters that achieves the best performance across the data streams is selected. These parameters are then used to initialize DDM and EDDM instances for the subsequent online phase.

Algorithm 1 implements the Warm Start phase, formalizing the process of iteratively optimizing hyperparameters for a voting ensemble of drift detectors.

3.3. Online Phase

The Online phase is executed after the Warm Start phase. Its first step is to configure an odd number of detectors, DDM or EDDM, with the hyperparameter values obtained in the previous phase. After configuring the initial detector set, this phase moves on to the data stream monitoring, during which drifts are detected and the number of detectors is managed in real-time. This detector dynamic management relies on two main methods:

1. **Increasing Number of Detectors**: in a divided vote scenario, where a simple minority of detectors agree on the presence of a concept drift, it suggests potential uncertainty or the emergence of new patterns in the data stream. To address this, and to maintain an odd total number of detectors, two additional detectors are added to the ensemble. These new detectors, either DDM or EDDM, are selected randomly and instantiated with hyperparameters perturbed by Gaussian noise centered around the optimal values defined in the Warm Start phase. This controlled randomness enhances the ensemble's ability to adapt to diverse drift patterns while avoiding over-specialization. The mean-centered nature of the Gaussian distribution ensures that while variability is introduced, the detectors remain close to their optimal configurations.
2. **Decreasing Number of Detectors**: when there is an unanimity vote scenario, the detectors indicate strong confidence in the presence of a drift, which suggests a stable concept. In this case, the ensemble reduces its complexity by removing two randomly selected detectors. This reduction helps maintain computational efficiency and prevents over-fitting.

Algorithm 1 Warm Start Phase for Self-tuning Drift Ensemble (StDE)

Input:
- D: Initial finite data stream
- M: List of drift detectors
- H: Dictionary of hyperparameter ranges for each detector in M
- n_iter: Number of random hyperparameter configurations

Output:
- $best\_params$: Dictionary of best hyperparameters for each detector in M

begin $best\_params \leftarrow \emptyset$
$best\_score \leftarrow -\infty$
for $i \in [1, \ldots, n\_iter]$ **do**
  $current\_detectors \leftarrow \emptyset$
  for $m \in M$ **do**
    - Generate random hyperparameter h_i for detector m from $H[m]$
    - Configure detector d_i of type m with hyperparameters h_i
    - Add d_i to $current\_detectors$
  end
  - Create hard voting ensemble E_i using $current\_detectors$
  - Evaluate E_i on subset D using F1 Score metric
  - $score_i \leftarrow$ Evaluation metric for E_i
  if $score_i > best\_score$ **then**
    $best\_score \leftarrow score_i$
    $best\_params \leftarrow \{h_i \mid m \in M\}$
  end
end
return $best\_params$

Algorithm 2 implements the Online phase to dynamically update the ensemble by changing the number of detectors based on data stream behavior.

4. Exiperiments

4.1. Automatic Generation of Streams

The StDE's performance was evaluated on abrupt, gradual, and incremental drifts using synthetic data streams generated with the Synthetic Datastream Database Generator[1] [Sakurai et al. 2023]. This tool allows the creation of homogeneous streams (containing one drift type) and heterogeneous streams (mixing various drift types). It systematically introduced different drifts, providing a robust test bed for assessing the algorithm's adaptability and accuracy.

We generated 1800 data streams—900 homogeneous and 900 heterogeneous. Homogeneous streams each contain 300 samples of a single drift type, while heterogeneous streams mix all three drift types in varying sequences. Stream sizes and drift locations were randomly defined to introduce variability and ensure unbiased algorithm evaluation.

[1] Available on `https://github.com/gysakurai/datastream-synthetic`

Algorithm 2 Online Phase for Self-tuning Drift Ensemble (StDE)

Input:

- $Ensemble$: Set of initialized detectors $D_1, D_2, \ldots, D_n$ with optimal hyperparameters
- Streaming data: Incoming data stream
- k: Threshold for Minority Sequence Count
- m: Threshold for Unanimity Sequence Count

begin

1. **Set** MinorityThreshold $\leftarrow \frac{|\text{Ensemble}|}{2}$
2. **Set** UnanimityThreshold $\leftarrow |\text{Ensemble}|$
3. **Initialize** MinoritySequenceCount $\leftarrow 0$
4. **Initialize** UnanimitySequenceCount $\leftarrow 0$
5. **While** streaming data is available:
   (a) Collect votes from all detectors in Ensemble if warning detected
   (b) Count votes indicating drift ($driftCount$)
   (c) **If** $driftCount <$ MinorityThreshold **then**
      i. MinoritySequenceCount $\leftarrow$ MinoritySequenceCount $+ 1$
      ii. **If** MinoritySequenceCount $= k$ **then**
         - Add two new random detectors $D_{\text{new1}}, D_{\text{new2}}$
         - Perturb hyperparameters of new detectors with Gaussian noise
         - Ensemble $\leftarrow$ Ensemble $\cup\, D_{\text{new1}}, D_{\text{new2}}$
         - Reset MinoritySequenceCount to 0
      iii. Reset UnanimitySequenceCount to 0
   (d) **Else**
      i. **StDE detects the drift**
      ii. Reset MinoritySequenceCount to 0
      iii. **If** $driftCount =$ UnanimityThreshold **then**
         A. UnanimitySequenceCount $\leftarrow$ UnanimitySequenceCount $+ 1$
         B. **If** UnanimitySequenceCount $= m$ **then**
            - Remove two random detectors $D_{\text{remove1}}, D_{\text{remove2}}$
            - Ensemble $\leftarrow$ Ensemble $\setminus\, D_{\text{remove1}}, D_{\text{remove2}}$
            - Reset UnanimitySequenceCount to 0
   (e) Update detectors in Ensemble with new data

end

4.2. Evaluation Metrics

StDE was assessed according to three main criteria: detection performance, delay to detect drifts and memory cost. In this section, we provide further details on how the metrics for these criteria were calculated.

F1 score is derived from precision and recall metrics [DeVries et al. 2021]. Precision measures the algorithm's ability to avoid false positives, while recall measures its ability to detect as many drifts as possible. By combining these two metrics, F1 score offers a comprehensive evaluation of the algorithm's performance.

Also is used Detection Delay as one of evaluation metrics, it represents the time interval between the occurrence of a concept drift and its detection by the detector. Low detection delays enable prompt reactions to drifts, meeting real-time constraints, while high delays can lead to missed opportunities, false alarms, or incorrect decisions. This delay is influenced by factors such as data complexity, drift frequency and magnitude, the detector's behavior, and the decision threshold used. Balancing detection delay and accuracy is vital for optimizing data stream drift detectors according to the specific requirements of the application. In this study, we calculate detection delay by counting the number of bits in the data stream between the known start of drifts and the point where they were detected by the assessed detector.

Memory cost in drift detection algorithms refers to the memory allocated and utilized during execution, which impacts efficiency and scalability in resource-limited environments. It includes average memory usage, reflecting the typical memory footprint. Efficient memory management is crucial for handling large data streams and adapting to changes without causing overflow or slowdowns, ensuring real-time performance and reliability. In our study, we use Python 3.10 and the tracemalloc library[2] to measure memory usage, reporting it in megabytes (MB) for comparisons.

4.3. Results

For comparison and benchmarking against StDE, we will use the following baselines: a DDM instance with optimal hyperparameters from the Warm Start Phase, an EDDM instance with its optimal hyperparameters, and a modified version of our ensemble method, Fixed-StDE, which maintains a constant number of base learners equivalent to the maximum used by StDE during its online phase. These benchmarks will be tested on both homogeneous and heterogeneous data streams to evaluate model performance across different environments and drift behaviors. For StDE testing, we set the following parameters based on preliminary results: Threshold for Minority Sequence Count = 1000, Threshold for Unanimity Sequence Count = 5, and Window Size = 5.

For detection performance in homogeneous data streams, StDE substantially outperforms the baselines and achieves the highest F1 score, indicating superior performance in handling drifts of the same type within a stream, as shown in Figure 3. StDE's F1-score exceeded Fixed-StDE's by 20 percentage points, which suggests that dynamically adjusting the number of detectors can be more effective than merely maximizing their quantity in this scenario.

When it comes to heterogeneous data streams, StDE also outperformed the baselines. DDM and Fixed-StDE also achieved satisfactory results. Conversely, EDDM produced significantly poor results, as shown in Figure 3. This outcome indicates that the balanced integration of different methods within the proposed StDE method, along with the hyperparameters tuning — making some detectors more sensitive and others less sensitive — results in superior detection performance even when multiple drift types are present in the same stream. By carefully adjusting these hyperparameters, we are able to enhance the overall performance, achieving more satisfactory results compared to the other methods.

[2]Available on `https://docs.python.org/3/library/tracemalloc.html`

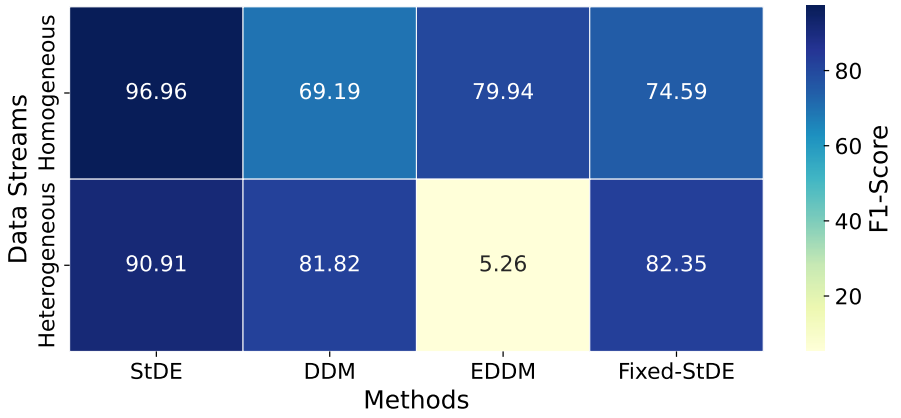

Figure 3. Heat-map of F1 score in detection methods. DDM and EDDM have had their hyperparameters tuned. Fixed-StDE was constructed with the maximum number of base learners.

For detection delay in homogeneous data streams, StDE performed slightly better than DDM and Fixed-StDE, as shown in Figure 4. Conversely, EDDM yielded significantly worse results when considering delay. For detection delay in heterogeneous data streams, StDE is also slightly better than DDM and Fixed-StDE, with StDE exhibiting a higher delay when compared to homogeneous streams, though. EDDM presented a very rapid reaction, as observed in Figure 4, but performed poorly in terms of predictive capacity.

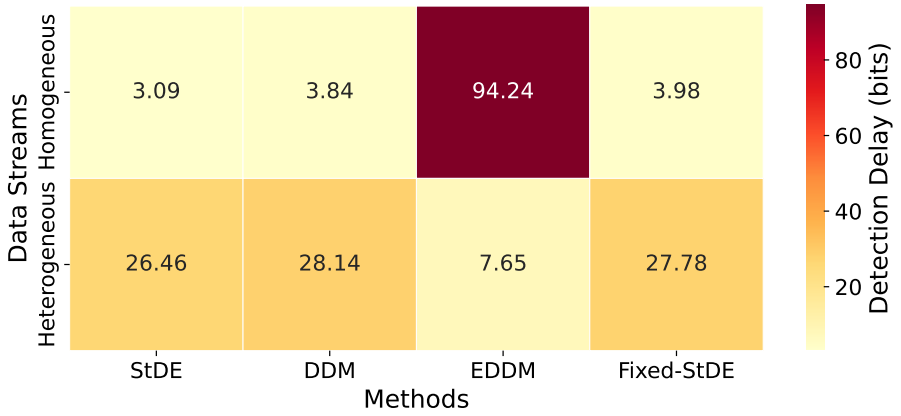

Figure 4. Heat-map of delay in detection methods. DDM and EDDM have had their hyperparameters tuned. Fixed-StDE was constructed with the maximum number of base learners.

About the memory cost metric, Figure 5 presents the results for this metric. As expected, StDE required more memory than the baselines based on DDM and EDDM. StDE utilizes a minimum of three base learner instances but only consumes up to twice the memory of a single base learner (DDM or EDDM), which can be deemed satisfactory. Another positive aspect here is that StDE's memory consumption is much lower than that of Fixed-StDE. This demonstrates that the dynamic adjustment of the number of base learners allowed for good predictive performance alongside a significant reduction in memory footprint.

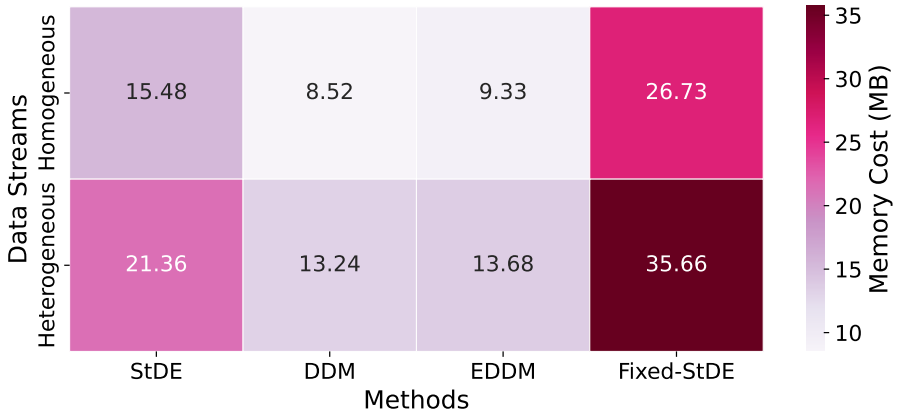

Figure 5. Heat-map of Memory Cost in detection methods. DDM and EDDM have had their hyperparameters tuned. Fixed-StDE was constructed with the maximum number of base learners.

4.3.1. Number of Detectors Fluctuation

As presented in Figure 6, the blue line indicates the number of detectors over time, while the red dashed lines mark when potential drifts were signaled by the detectors. Additionally, the figure indicates the actual drifts in the data stream, highlighted by the intervals in gray. The moments where the number of detectors increases or decreases are based on the number of votes, with each detector signaling a potential drift. Initially, the number of detectors remains constant at its initial value of three detectors. However, as potential drift points are signaled by the detectors, there are visible increments and decrements in the number of detectors, reflecting the system's adaptive response to changes in the data distribution. As exemplified in Figure 6, at the point with 2 votes out of 5 detectors, when the threshold for minority is reached, the number of detectors increases. Conversely, the point with 9 votes out of 9 detectors represents a moment where the threshold for unanimous votes is reached and the number of detectors decreases.

5. Conclusion

StDE presents an advancement in the field of concept drift detection for data streams. By dynamically adjusting the number and configuration of its base learners based on their voting behavior, StDE effectively addresses the challenges posed by the diverse and evolving nature of data streams. The dual-phase operation — initial Warm start phase with offline data followed by real-time monitoring — ensures that StDE remains adaptable to various drift types, whether abrupt, gradual, or incremental. Our tests demonstrate that StDE maintains high efficiency and accuracy across different drift scenarios, validating its robustness and versatility. Overall, StDE represents a promising solution for real-time data stream mining, capable of maintaining performance and reliability in the face of continuous and unpredictable changes. Future work could explore further optimization of the ensemble's hyperparameters and investigate the integration of additional types of base learners to enhance the system's performance in even more complex data environments.

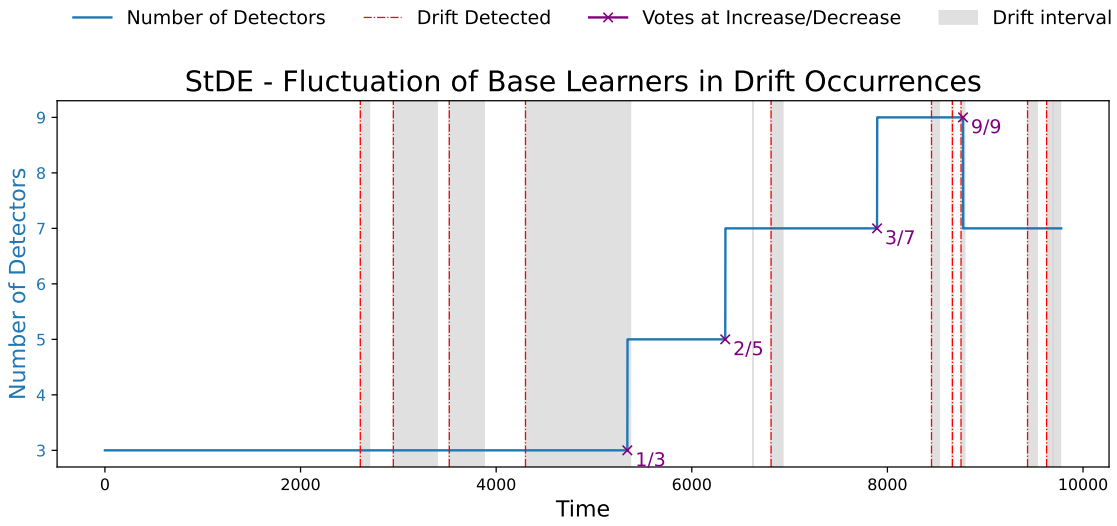

Figure 6. Fluctuation in the number of detectors in response to the signaling of potential drifts by the detector

References

Abbasi, A., Javed, A. R., Chakraborty, C., Nebhen, J., Zehra, W., and Jalil, Z. (2021). Elstream: An ensemble learning approach for concept drift detection in dynamic social big data stream learning. *IEEE Access*, 9:66408–66419.

Cano, A. and Krawczyk, B. (2022). Rose: robust online self-adjusting ensemble for continual learning on imbalanced drifting data streams. *Machine Learning*, 111(7):2561–2599.

Deckert, M. (2011). Batch weighted ensemble for mining data streams with concept drift. In *Foundations of Intelligent Systems: 19th International Symposium, ISMIS 2011, Warsaw, Poland, June 28-30, 2011. Proceedings 19*, pages 290–299. Springer.

DeVries, Z., Locke, E., Hoda, M., Moravek, D., Phan, K., Stratton, A., Kingwell, S., Wai, E. K., and Phan, P. (2021). Using a national surgical database to predict complications following posterior lumbar surgery and comparing the area under the curve and f1-score for the assessment of prognostic capability. *The Spine Journal*, 21(7):1135–1142.

Du, L., Song, Q., Zhu, L., and Zhu, X. (2015). A selective detector ensemble for concept drift detection. *The Computer Journal*, 58(3):457–471.

Han, M., Chen, Z., Li, M., Wu, H., and Zhang, X. (2022). A survey of active and passive concept drift handling methods. *Computational Intelligence*, 38(4):1492–1535.

Khamassi, I., Sayed-Mouchaweh, M., Hammami, M., and Ghédira, K. (2013). Ensemble classifiers for drift detection and monitoring in dynamical environments. In *Annual Conference of the PHM Society*, volume 5.

Komorniczak, J., Zyblewski, P., and Ksieniewicz, P. (2022). Statistical drift detection ensemble for batch processing of data streams. *Knowledge-Based Systems*, 252:109380.

Korycki, L. and Krawczyk, B. (2019). Unsupervised drift detector ensembles for data stream mining. In *2019 IEEE international conference on data science and advanced analytics (DSAA)*, pages 317–325. IEEE.

Lagman, A., Alfonso, L., Goh, M., Lalata, J.-A., Magcuyao, J. P., and Vicente, H. (2020). Classification algorithm accuracy improvement for student graduation prediction using ensemble model. *International Journal of Information and Education Technology*, 10:723–727.

Martins, V. E., Cano, A., and Junior, S. B. (2023). Meta-learning for dynamic tuning of active learning on stream classification. *Pattern Recognition*, 138:109359.

Mavromatis, I., Sanchez-Mompo, A., Raimondo, F., Pope, J., Bullo, M., Weeks, I., Kumar, V., Carnelli, P., Oikonomou, G., Spyridopoulos, T., et al. (2023). Le3d: a lightweight ensemble framework of data drift detectors for resource-constrained devices. In *2023 IEEE 20th Consumer Communications & Networking Conference (CCNC)*, pages 611–619. IEEE.

Pérez, J. L. M., Barros, R. S., and Santos, S. G. (2020). Statistical tests ensemble drift detector. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1021–1028. IEEE.

Ramane, M. and Gnanasekar, J. (2022). A novel hybrid concept drift detector ensemble for handling pure and hybrid drift in non-stationary data streams. *NeuroQuantology*, 20(8):6170.

Sakurai, G. Y., Lopes, J. F., Zarpelão, B. B., and Barbon Junior, S. (2023). Benchmarking change detector algorithms from different concept drift perspectives. *Future Internet*, 15(5).

Samant, R. C., Patil, S. H., Sinha, R. N., and Kadam, A. K. A systematic ensemble approach for concept drift detector selection in data stream classifiers.

Silva, R. P., Junior, S. B., Zarpelão, B. B., and De Melo, L. F. (2024). Unsupervised tuning for drift detectors using change detector segmentation. *IEEE Access*, 12:54256–54271.