Utilization of Adaptive Machine Learning for Streaming Sentiment Analysis: The Effects of Batch and Drift Types

Sudianto Sudianto
Informatics Engineering Study Program, Telkom University, Purwokerto, Indonesia
sudianto@telkomuniversity.ac.id (corresponding author)

Aminatus Sa'adah
Informatics Engineering Study Program, Telkom University, Purwokerto, Indonesia
aminatuss@telkomuniversity.ac.id

Brian Farrel Arkana
Informatics Engineering Study Program, Telkom University, Purwokerto, Indonesia
brianfarrel@student.telkomuniversity.ac.id

Received: 19 November 2025 | Revised: 13 December 2025, 30 December 2025, 1 January 2026, and 3 January 2026 | Accepted: 4 January 2026

 DOI: https://doi.org/10.48084/etasr.16379

ABSTRACT

The changing data patterns that continue to emerge from user reviews on digital platforms demand machine learning models to be able to adapt sustainably. This phenomenon, known as concept drift, can degrade accuracy if not handled appropriately. The study explores two adaptive learning approaches—Adaptive Random Forest (ARF) and Adaptive XGBoost (AXGB)—to classify sentiment on real-time data streams. The dataset used includes OVO reviews (2016–2025) and GoPay reviews (2023–2025) from the Google Play Store. After performing text cleanup, tokenization, stemming, and Term Frequency–Inverse Document Frequency (TF-IDF) representation, the data are streamed into two learning schemes: Big Batch Small Batch (BBSB), which combines large and small batches to detect changes more subtly, and N Drift Types Batch (NDTB), which adjusts batch sizes based on the number of drift types considered. The evaluation was carried out using accuracy metrics, computational time, and true positive rate in each sentiment category. The results showed that ARF provided the most stable performance with an average accuracy of 82%, whereas AXGB performed better in the BBSB configuration with an accuracy of 81% and faster training time. These findings confirm that both configurations can serve as practical frameworks for text stream learning and adaptive concept drift management.

Keywords-adaptive learning; concept drift; data stream; batch streaming; sentiment analysis

I. INTRODUCTION

The surge of user-generated content on digital platforms has produced continuously evolving data, where the relationship between features and target labels shifts over time, known as concept drift [1]. When concept drift occurs, traditional static models fail to maintain performance since they cannot adapt without full retraining, which is costly and impractical in real-time environments [2]. Adaptive learning models such as Adaptive Random Forest (ARF) and Adaptive XGBoost (AXGB) have been proposed to overcome this challenge by updating their internal structure dynamically during streaming [3, 4]. These models have demonstrated strong adaptability in diverse domains, including intrusion detection [5], data stream processing in fog-cloud environments [6], and large-scale anomaly recognition [7]. In addition, ARF and related ensemble frameworks have shown robustness in handling imbalanced and drifting data [8, 9]. Studies in text classification and stream ensembles further highlight the importance of adaptive models for handling evolving data [10]. However, when applied to text-based sentiment streams, their comparative behavior—especially under varying batch configurations—remains inadequately studied. Therefore, this study aims to evaluate ARF and AXGB in real-world financial application reviews, focusing on their ability to handle evolving sentiment data streams with varying drift intensities and runtime constraints.

Despite significant progress, current adaptive stream learning methods still face several limitations. Many studies have focused on improving drift adaptation through meta-learning [11], hybrid multi-stream frameworks [12], and

lightweight adaptation mechanisms [13], whereas others have explored federated and distributed solutions to ensure fairness in model retraining [14, 15]. Techniques such as dynamic drift detection [16], unsupervised adaptation using robust cut forests [17], and incremental genetic programming [18] have achieved improved responsiveness to drift yet are mostly validated on numerical or sensor datasets. Research on text-based or sentiment-rich data streams—where linguistic context shifts rapidly—remains sparse [19]. Furthermore, studies have optimized ensemble learning using adaptive detectors [4], drift-type classification [20], and locality-aware adaptation [21], but comparative analysis between ARF and AXGB under real-world streaming textual data is still lacking [22]. This reveals a research gap in understanding how adaptive models behave under textual stream drift, particularly in multilingual, time-evolving contexts where class distributions are highly imbalanced.

Another critical gap lies in stream configuration strategies, which directly influence model adaptation speed and detection precision. Several researchers have introduced adaptive windowing for drift type classification [20], hierarchical batch schemes [23], and ensemble frameworks to balance accuracy with computation [24]. Other studies focused on improving drift resilience and energy efficiency in adaptive ensembles [25, 26], or enhancing ARF performance through dimensionality reduction [27] and continual experience replay [28]. Complementary approaches such as anomaly-aware adaptive ensembles [29], comprehensive classification strategies for imbalanced streams [30], and systematic reviews on drift adaptation [31], have contributed valuable insights, yet few address the trade-off between batch granularity and runtime performance. Recent ensemble models like Robust Online Self-Adjusting Ensemble (ROSE) demonstrate self-calibration in imbalanced drifting data [32], suggesting that data stream configuration and retraining frequency remain decisive factors for stable adaptation. To bridge these gaps, this study introduces two stream configurations—Big Batch Small Batch (BBSB) and N Drift Types Batch (NDTB)—to analyze their effects on runtime efficiency, adaptability, and accuracy of ARF and AXGB in evolving sentiment classification tasks.

II. METHODOLOGY

Figure 1 illustrates the overall workflow of the proposed research, consisting of three main phases: data collection, data preprocessing, and model implementation and evaluation.

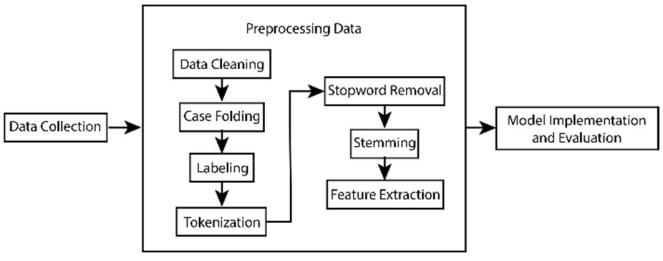

Fig. 1. Research *w*orkflow.

A. Data Collection

The first phase involves data collection using the Google Play Scraper library in Python to extract user reviews from mobile payment applications. The dataset consists of 406,282 OVO reviews collected between 2016 and 2025, and 240,406 GoPay reviews collected between 2023 and 2025. These reviews serve as the source of streaming data for training and testing the adaptive learning models.

B. Data Preprocessing

After the data were collected, several preprocessing steps were applied to ensure textual consistency and improve the quality of input features. The preprocessing pipeline includes:

- Data cleaning: Removal of non-alphabetic characters, redundant spaces, and unnecessary symbols.
- Case folding: Conversion of all text to lowercase.
- Sentiment labeling: Automatic sentiment labeling using the IndoBERT/RoBERTa model to assign each review as positive, neutral, or negative.
- Tokenization: Splitting text into individual word tokens.
- Stopword removal: Eliminating common words with low semantic value.
- Stemming: Reducing words to their base or root form.
- Feature extraction: Transforming the cleaned text into numerical vectors using the Term Frequency–Inverse Document Frequency (TF-IDF) representation.

After preprocessing, the resulting feature matrices were streamed into two different data streaming configurations, designed to simulate real-world continuous data arrival for adaptive model evaluation.

C. Big Batch Small Batch Configuration

In the first configuration, called BBSB, the data stream is divided hierarchically—each big batch consists of 968 data instances, which are further subdivided into eight small batches of 121 instances each. A practical analogy of this setup can be seen in temporal data, such as days within a week, hours within a day, minutes within an hour, or even months within weeks, where smaller units naturally cluster inside larger units. Figure 2 shows the hierarchical batch configuration for the BBSB setup.

The ARF and AXGB models process data incrementally at the small-batch level, whereas drift detection and retraining occur at the end of each big batch. This structure enables the observation of drift patterns by analyzing changes in the data distribution between consecutive small batches within a single big batch.

D. N Drift Types Batch Configuration

The second configuration, referred to as NDTB, determines the batch size based on the formula:

$$batch\_size = \frac{total\_data}{number\ of\ drift\ types * n} \quad (1)$$

where n represents the desired number of samples per drift type. In this study, $n = 100$ and the $number\ of\ drift\ types = 5$—namely no drift, sudden drift, gradual drift, incremental drift, and recurrent drift.

Using this configuration, the resulting batch sizes are 813 data instances per batch for training and 481 data instances per batch for testing. This approach aims to provide balanced sample representation across multiple drift types, although in practice, the actual data distribution may not always be perfectly uniform across all drift categories. Figure 3 shows the hierarchical batch configuration for the NDTB setup.

E. Model Implementation and Evaluation

Both configurations were implemented using two adaptive ensemble models: ARF and AXGB. Each model employed 20 estimators (trees) with max_features='sqrt'. The OVO review dataset was used as the training data stream, whereas the GoPay review dataset was employed as the testing stream to simulate a realistic cross-domain and temporal concept drift scenario. OVO reviews span a longer historical period (2016–2025), capturing gradual changes in user sentiment over time, whereas GoPay reviews represent a more recent platform context (2023–2025) with potentially different user behavior, linguistic patterns, and sentiment distribution. This separation allows the evaluation of model generalization and adaptability when confronted with domain shift and evolving sentiment characteristics, reflecting real-world deployment conditions where models trained on historical data must adapt to newer platforms or services. After the streaming process was completed, the model performance was evaluated based on:

- Accuracy: Overall classification performance per configuration.
- Runtime: Total processing time for each streaming setup.
- True positive rate: Performance consistency across different sentiment classes.

These evaluations aimed to determine how each adaptive model handled concept drift under the two streaming scenarios and how effectively they maintained classification performance over time. All experiments were conducted on a laptop equipped with an NVIDIA GeForce RTX 3050 Laptop GPU, an Intel processor, and a Python-based machine learning environment. GPU acceleration was utilized to support efficient model training and streaming evaluation, ensuring reproducibility of runtime performance across adaptive learning configurations.

III. RESULTS AND DISCUSSION

This section presents the experimental results and analysis of two adaptive learning models, ARF and AXGB, implemented under two different data stream configurations: the BBSB configuration and the NDTB configuration. The evaluation focuses on the models' classification performance, runtime efficiency, and stability in handling evolving data distributions.

A. Big Batch Small Batch Configuration

Tables I and II and Figures 4 and 5 present the classification reports and confusion matrices for ARF and AXGB, showing that the BBSB configuration achieved an overall accuracy of 82% for ARF and 81% for AXGB. Both models performed well in predicting positive sentiments, which dominate both the training and testing datasets.

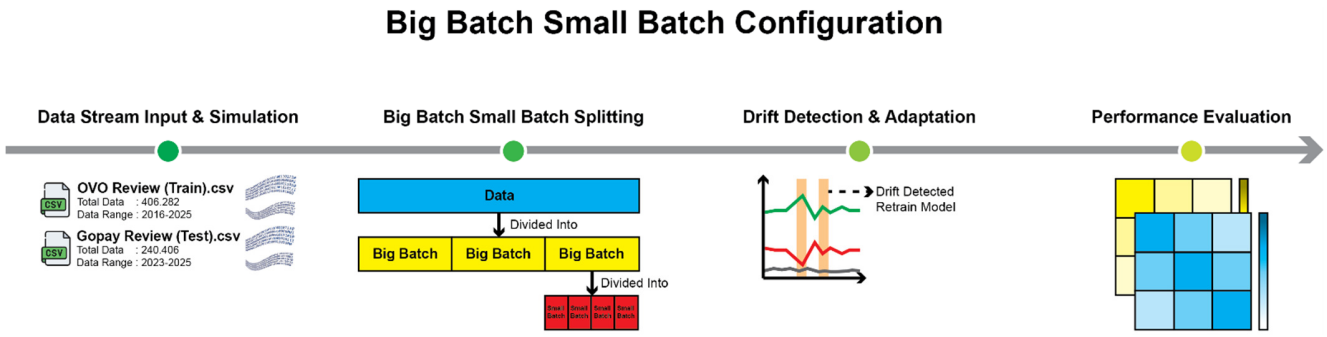

Fig. 2. Illustration of the BBSB configuration.

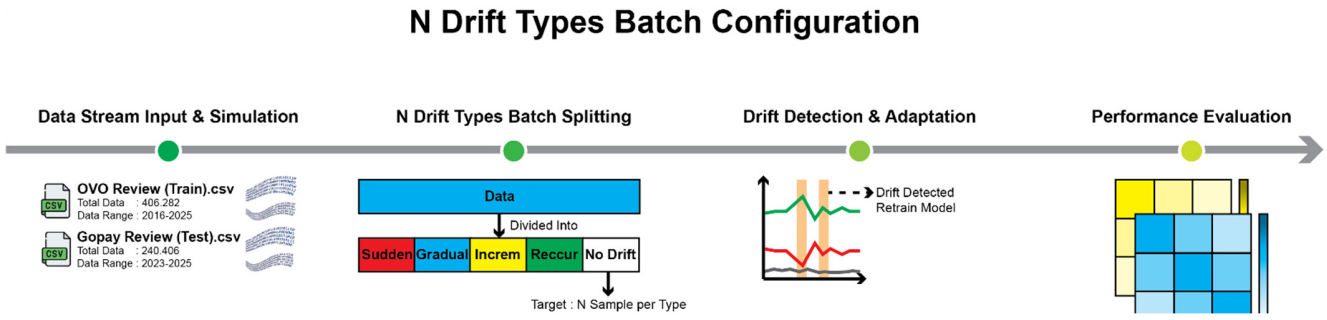

Fig. 3. Illustration of the NDTB configuration.

TABLE I. CLASSIFICATION REPORT OF ARF – BBSB CONFIGURATION

	Precision	Recall	F1-score	Support
Negative	0.66	0.53	0.59	46,913
Neutral	0.69	0.04	0.07	19,188
Positive	0.84	0.98	0.91	174,305
Accuracy			0.82	240,406
Macro Avg	0.73	0.52	0.52	240,406
Weighted Avg	0.80	0.82	0.78	240,406

TABLE II. CLASSIFICATION REPORT OF AXGB – BBSB CONFIGURATION

	Precision	Recall	F1-score	Support
Negative	0.72	0.45	0.55	46,913
Neutral	0.54	0.21	0.30	19,188
Positive	0.84	0.98	0.90	174,305
Accuracy			0.81	240,406
Macro Avg	0.70	0.55	0.59	240,406
Weighted Avg	0.79	0.81	0.79	240,406

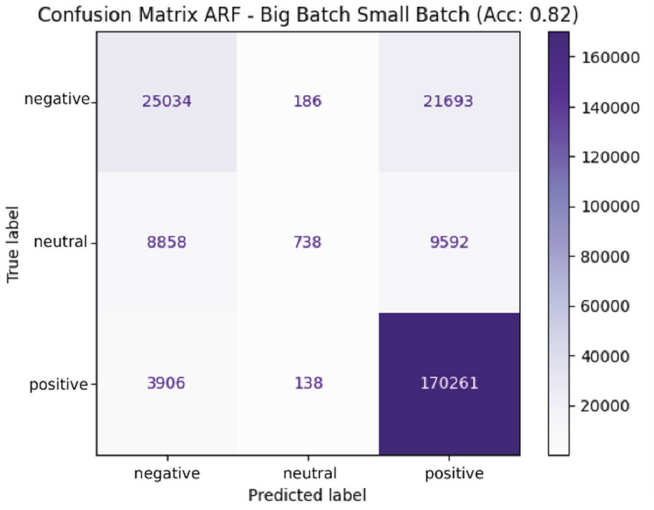

Fig. 4. Confusion matrix of ARF – BBSB configuration.

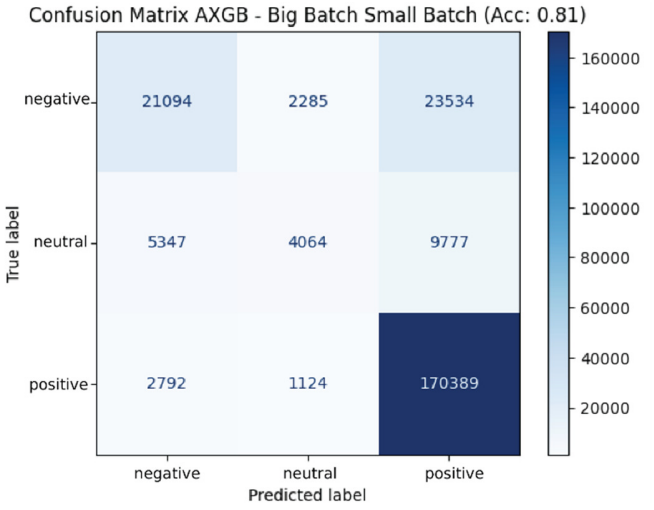

Fig. 5. Confusion matrix of AXGB – BBSB configuration.

The total runtime for the BBSB configuration was 25 h, 18 min, and 41 s, with a 10 s delay per small batch and a 60 s delay per big batch. The detailed runtime analysis is as follows:

- Training time: 8 h, 4 min, 20 s.
- Average time per small batch: 19–20 s.
- Average time per big batch: 3 min 30 s – 4 min.

When executed without any time delay, the runtime significantly decreased to 9–10 s per small batch and 2 min 30 s – 3 min per big batch.

B. N Drift Types Batch Configuration

Tables III and IV and Figures 6 and 7 present the classification reports and confusion matrices for ARF and AXGB, showing that the NDTB configuration achieved an overall accuracy of 82% for ARF and 80% for AXGB. This indicates that ARF maintained its performance level across configurations, whereas AXGB experienced a slight decrease (–1%) in accuracy under the NDTB setup.

TABLE III. CLASSIFICATION REPORT OF ARF – NDTB CONFIGURATION

	Precision	Recall	F1-score	Support
Negative	0.66	0.53	0.59	46,913
Neutral	0.70	0.04	0.07	19,188
Positive	0.85	0.98	0.91	174,305
Accuracy			0.82	240,406
Macro Avg	0.74	0.52	0.52	240,406
Weighted Avg	0.80	0.82	0.78	240,406

TABLE IV. CLASSIFICATION REPORT OF AXGB – NDTB CONFIGURATION

	Precision	Recall	F1-score	Support
Negative	0.69	0.41	0.52	46,913
Neutral	0.48	0.18	0.26	19,188
Positive	0.83	0.97	0.89	174,305
Accuracy			0.80	240,406
Macro Avg	0.67	0.52	0.56	240,406
Weighted Avg	0.77	0.80	0.77	240,406

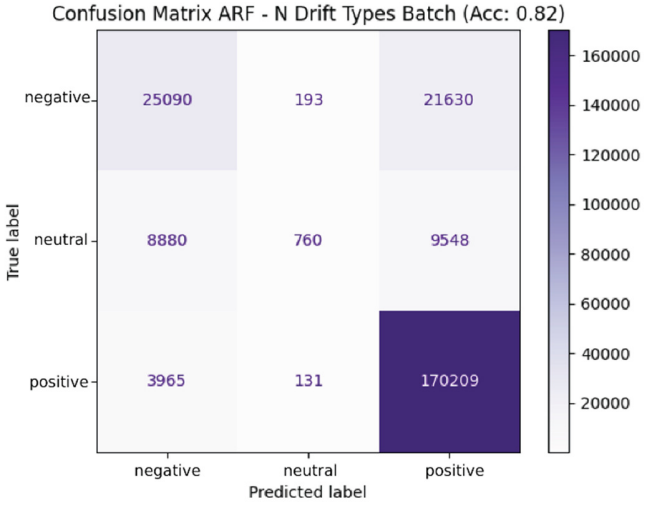

Fig. 6. Confusion matrix of ARF – NDTB configuration.

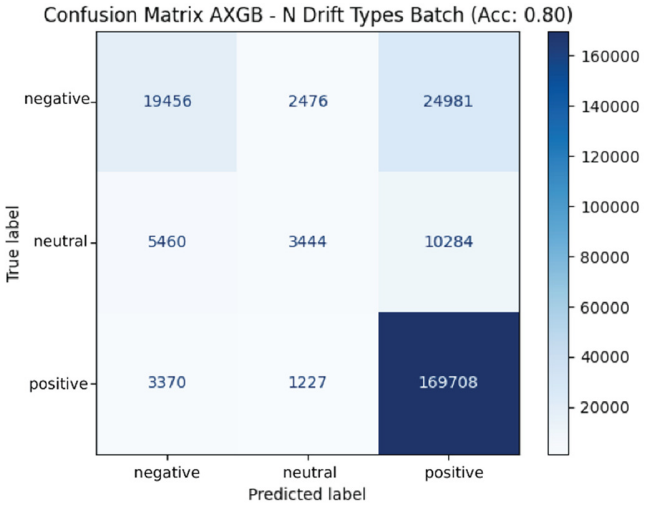

Fig. 7. Confusion matrix of AXGB – NDTB configuration.

The total runtime for the NDTB configuration was 23 h, 33 min, and 32 s, with a 60 s delay per batch. The detailed runtime breakdown is as follows:

- Training time: 8 h, 19 min, 10 s.
- Average time per batch: 1 min 30 s – 2 min.

When executed without any batch delay, the average runtime per batch was reduced to 30–60 s. This reduction highlights the computational efficiency of the NDTB configuration.

In addition, Figures 8 and 9 visualize the daily accuracy trends and learning curves for both adaptive models throughout the streaming process using GoPay review data collected between 2023 and 2025.

The daily accuracy visualization (Figure 8) indicates that both models are able to maintain relatively stable performance across the streaming period. However, ARF demonstrates higher consistency, with smaller fluctuations during periods of concept drift. This stability suggests that ARF is more robust to abrupt or gradual changes in sentiment patterns.

Meanwhile, the learning curve analysis (Figure 9) shows that both models exhibit initial variability in accuracy during early stages of streaming, followed by gradual improvement and eventual stabilization. This trend confirms their adaptability under evolving data distributions.

C. Comparative Analysis

Based on the experimental results obtained under both data stream configurations, several key observations can be summarized:

- Model performance: AXGB achieved marginally higher performance under the BBSB configuration, with an accuracy improvement of approximately 1% compared to the NDTB configuration. Conversely, ARF delivered consistently stable performance across both configurations, maintaining an accuracy of 82%. Notably, ARF produced a higher number of true positives per sentiment in the NDTB configuration.
- Runtime efficiency: From a computational perspective, the BBSB configuration resulted in faster training time, reducing model training duration by approximately 14 min and 50 s. However, the NDTB configuration achieved a shorter overall runtime (1 h and 45 min faster), mainly due to its single-level batch design.

The reported performance differences between models and configurations are interpreted from a practical streaming perspective rather than strict statistical hypothesis testing. Given the large-scale and continuous nature of the data stream, accuracy, true positive rate trends, and runtime behavior were analyzed descriptively to assess model stability and adaptability. Observed accuracy differences of 1–2% are considered operationally meaningful in streaming sentiment classification, particularly when evaluated alongside consistency across sentiment classes and computational efficiency.

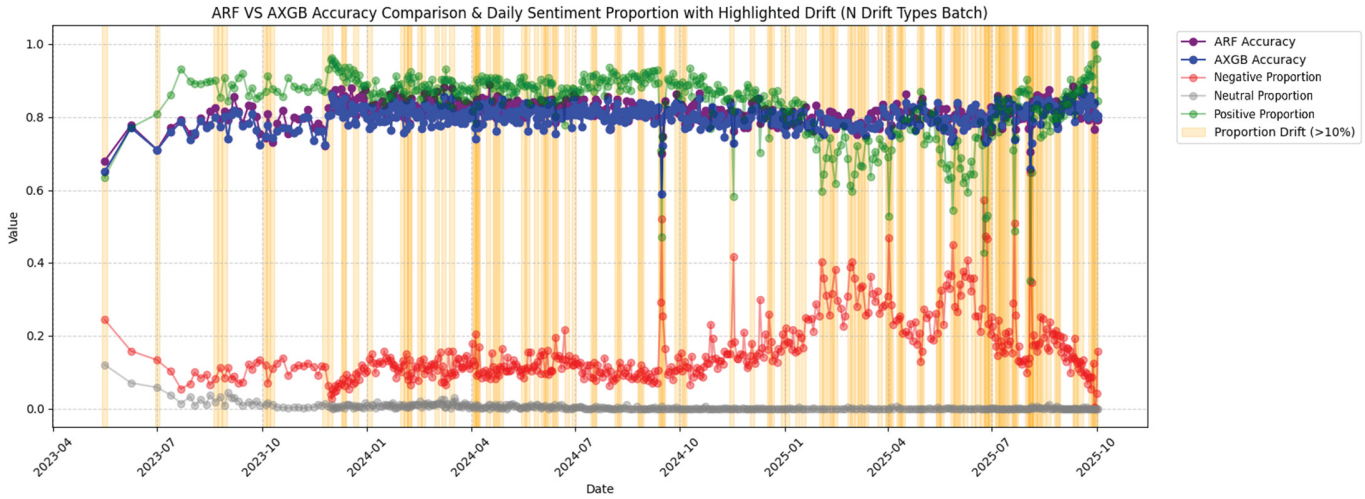

Fig. 8. Daily accuracy of both models with sentiment proportion from GoPay review data – NDTB configuration.

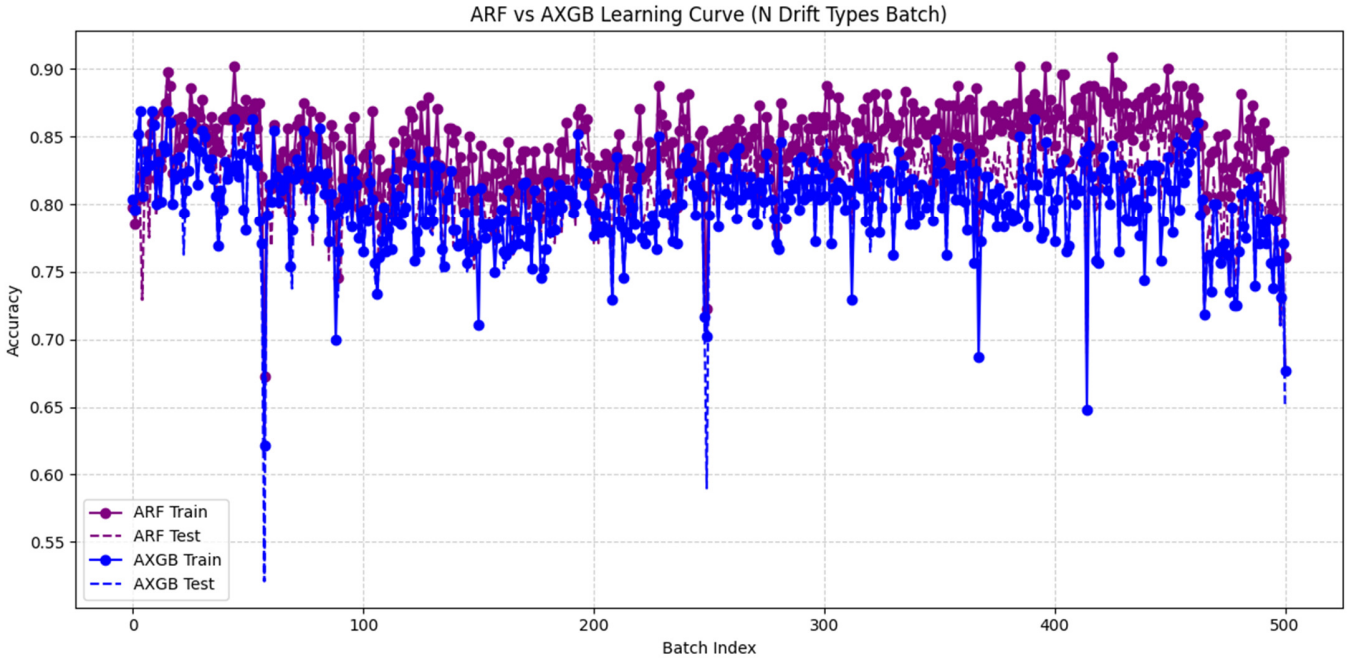

Fig. 9. Learning curve of both models – NDTB configuration.

IV. CONCLUSION

This study investigated the handling of concept drift in data stream sentiment classification using two adaptive ensemble models: Adaptive Random Forest (ARF) and Adaptive XGBoost (AXGB). Two streaming configurations were proposed—the Big Batch Small Batch (BBSB) configuration and the N Drift Types Batch (NDTB) configuration—to evaluate model adaptability and performance consistency under dynamic streaming conditions. The experimental results showed that both models performed competitively, with ARF achieving stable accuracy across both configurations (82%), whereas AXGB showed a slight improvement under the BBSB configuration with faster training time. The BBSB configuration demonstrated better adaptation efficiency, whereas the NDTB configuration provided more balanced performance in terms of accuracy and true positive distribution across sentiment classes.

From a practical standpoint, the findings of this study provide guidance for designing adaptive sentiment monitoring systems in real-world applications such as digital payment platforms, online marketplaces, and social feedback analytics. The BBSB configuration is suitable for environments requiring finer-grained drift detection with hierarchical temporal structure, whereas the NDTB configuration offers a simpler and more computationally efficient alternative when balanced drift representation is desired. The demonstrated stability of ARF suggests its applicability in long-running systems prioritizing robustness, whereas AXGB is advantageous in scenarios demanding faster retraining and responsiveness. These insights can assist practitioners in selecting appropriate adaptive models and streaming configurations based on operational constraints and application requirements.

REFERENCES

[1] G. J. Aguiar and A. Cano, "Enhancing Concept Drift Detection in Drifting and Imbalanced Data Streams through Meta-Learning," in *2023 IEEE International Conference on Big Data*, Sorrento, Italy, 2023, pp. 2648–2657, https://doi.org/10.1109/BigData59044.2023.10386364.

[2] M. A. Shyaa, N. F. Ibrahim, Z. Zainol, R. Abdullah, M. Anbar, and L. Alzubaidi, "Evolving cybersecurity frontiers: A comprehensive survey on concept drift and feature dynamics aware machine and deep learning in intrusion detection systems," *Engineering Applications of Artificial Intelligence*, vol. 137, Nov. 2024, Art. no. 109143, https://doi.org/10.1016/j.engappai.2024.109143.

[3] B. Halstead *et al.*, "Analyzing and repairing concept drift adaptation in data stream classification," *Machine Learning*, vol. 111, no. 10, pp. 3489–3523, Oct. 2022, https://doi.org/10.1007/s10994-021-05993-w.

[4] X. Jin and Y. Zhang, "Adaptive Random Forest with Dynamic Detectors for Evolving Data Stream Classification," in *Proceedings of the 2023 9th International Conference on Computing and Artificial Intelligence*, Tianjin, China, 2023, pp. 678–684, https://doi.org/10.1145/3594315.3594390.

[5] N. Abdulla, M. Demirci, and S. Özdemir, "Adaptive Learning on Fog-Cloud Collaborative Architecture for Stream Data Processing," in *2021 International Symposium on Networks, Computers and Communications*, Dubai, United Arab Emirates, 2021, pp. 1–6, https://doi.org/10.1109/ISNCC52172.2021.9615824.

[6] A. O. AlQabbany and A. M. Azmi, "Measuring the Effectiveness of Adaptive Random Forest for Handling Concept Drift in Big Data Streams," *Entropy*, vol. 23, no. 7, July 2021, Art. no. 859, https://doi.org/10.3390/e23070859.

[7] Ł. Korycki and B. Krawczyk, "Adaptive Deep Forest for Online Learning from Drifting Data Streams." arXiv, Oct. 14, 2020, https://doi.org/10.48550/arXiv.2010.07340.

[8] M. G. Rahman and M. Z. Islam, "Adaptive Decision Forest: An incremental machine learning framework," *Pattern Recognition*, vol. 122, Feb. 2022, Art. no. 108345, https://doi.org/10.1016/j.patcog.2021.108345.

[9] L. Yang and A. Shami, "A Lightweight Concept Drift Detection and Adaptation Framework for IoT Data Streams," *IEEE Internet of Things Magazine*, vol. 4, no. 2, pp. 96–101, June 2021, https://doi.org/10.1109/IOTM.0001.2100012.

[10] D. Joshi and M. Shukla, "An Ensemble Approach to Improve the Performance of Real Time Data Stream Classification," *Engineering, Technology & Applied Science Research*, vol. 14, no. 6, pp. 17749–17754, Dec. 2024, https://doi.org/10.48084/etasr.8563.

[11] F. Ceschin, M. Botacin, H. M. Gomes, F. Pinagé, L. S. Oliveira, and A. Grégio, "Fast & Furious: On the modelling of malware detection as an evolving data stream," *Expert Systems with Applications*, vol. 212, Feb. 2023, Art. no. 118590, https://doi.org/10.1016/j.eswa.2022.118590.

[12] M. Badar, W. Nejdl, and M. Fisichella, "FAC-fed: Federated adaptation for fairness and concept drift aware stream classification," *Machine Learning*, vol. 112, no. 8, pp. 2761–2786, Aug. 2023, https://doi.org/10.1007/s10994-023-06360-7.

[13] E. Yu, J. Lu, B. Zhang, and G. Zhang, "Online Boosting Adaptive Learning under Concept Drift for Multistream Classification," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 15, pp. 16522–16530, Mar. 2024, https://doi.org/10.1609/aaai.v38i15.29590.

[14] E. Yu, Y. Song, G. Zhang, and J. Lu, "Learn-to-adapt: Concept drift adaptation for hybrid multiple streams," *Neurocomputing*, vol. 496, pp. 121–130, July 2022, https://doi.org/10.1016/j.neucom.2022.05.025.

[15] V. Yelleti, "ROSFD: Robust Online Streaming Fraud Detection with Resilience to Concept Drift in Data Streams." arXiv, Apr. 14, 2025, https://doi.org/10.48550/arXiv.2504.10229.

[16] Y. Zhong, H. Yang, Y. Zhang, P. Li, and C. Ren, "Long short-term memory self-adapting online random forests for evolving data stream regression," *Neurocomputing*, vol. 457, pp. 265–276, Oct. 2021, https://doi.org/10.1016/j.neucom.2021.05.026.

[17] Z. Pang, J. Cen, and M. Yi, "Unsupervised concept drift detection method based on robust random cut forest," *International Journal of Machine Learning and Cybernetics*, vol. 14, no. 12, pp. 4207–4222, Dec. 2023, https://doi.org/10.1007/s13042-023-01890-x.

[18] F. Ridder, K.-H. Chen, and N. Alachiotis, "Accelerated Real-Time Classification of Evolving Data Streams using Adaptive Random Forests," in *2023 International Conference on Field Programmable Technology*, Yokohama, Japan, 2023, pp. 232–237, https://doi.org/10.1109/ICFPT59805.2023.00031.

[19] M. A. Shyaa *et al.*, "Enhanced Intrusion Detection with Data Stream Classification and Concept Drift Guided by the Incremental Learning Genetic Programming Combiner," *Sensors*, vol. 23, no. 7, Apr. 2023, Art. no. 3736, https://doi.org/10.3390/s23073736.

[20] K.-T. Nguyen, Q.-T. Ha, and X.-H. Phan, "Dynamic Windowing Strategies for Concept Drift Type Classification in Data Streams," in *2024 IEEE International Conference on Progress in Informatics and Computing*, Shanghai, China, 2024, pp. 70–74, https://doi.org/10.1109/PIC62406.2024.10892652.

[21] G. J. Aguiar and A. Cano, "A comprehensive analysis of concept drift locality in data streams," *Knowledge-Based Systems*, vol. 289, Apr. 2024, Art. no. 111535, https://doi.org/10.1016/j.knosys.2024.111535.

[22] A. L. Suárez-Cetrulo, D. Quintana, and A. Cervantes, "A survey on machine learning for recurring concept drifting data streams," *Expert Systems with Applications*, vol. 213, Mar. 2023, Art. no. 118934, https://doi.org/10.1016/j.eswa.2022.118934.

[23] K. Goel and S. Batra, "Adaptive online learning for classification under concept drift," *International Journal of Computational Science and Engineering*, vol. 24, no. 2, pp. 128–135, Jan. 2021, https://doi.org/10.1504/IJCSE.2021.115099.

[24] P. S and A. U. R, "Ensemble framework for concept drift detection and class imbalance in data streams," *Multimedia Tools and Applications*, vol. 84, no. 11, pp. 8823–8837, Mar. 2025, https://doi.org/10.1007/s11042-024-18349-y.

[25] J. Montiel, R. Mitchell, E. Frank, B. Pfahringer, T. Abdessalem, and A. Bifet, "Adaptive XGBoost for Evolving Data Streams," in *2020 International Joint Conference on Neural Networks*, Glasgow, UK, 2020, pp. 1–8, https://doi.org/10.1109/IJCNN48605.2020.9207555.

[26] A. M. Paim and F. Enembreck, "Adaptive random tree ensemble for evolving data stream classification," *Knowledge-Based Systems*, vol. 309, Jan. 2025, Art. no. 112830, https://doi.org/10.1016/j.knosys.2024.112830.

[27] H. K. Fatlawi and A. Kiss, "Efficiency improvement of adaptive random forest using principal component analysis for mining data stream," *Annales Universitatis Scientiarum Budapestinensis de Rolando Eötvös Nominatae. Sectio computatorica*, vol. 55, pp. 39–48, Jan. 2023, https://doi.org/10.71352/ac.55.039.

[28] Ł. Korycki and B. Krawczyk, "Class-Incremental Experience Replay for Continual Learning under Concept Drift," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, Nashville, TN, USA, 2021, pp. 3644–3653, https://doi.org/10.1109/CVPRW53098.2021.00404.

[29] Y. Wu, L. Liu, Y. Yu, G. Chen, and J. Hu, "Online Adaptive Ensemble Enhanced Anomaly Detection for Addressing Concept Drift in IoT Systems." TechRxiv, Jan. 02, 2024, https://doi.org/10.36227/techrxiv.23304461.v2.

[30] K. A. M. Junaid, D. Paulraj, and T. Sethukarasi, "A comprehensive ensemble classification techniques detecting and managing concept drift in dynamic imbalanced data streams," *Wireless Networks*, vol. 31, no. 1, pp. 19–30, Jan. 2025, https://doi.org/10.1007/s11276-024-03742-0.

[31] S. Arora, R. Rani, and N. Saxena, "A systematic review on detection and adaptation of concept drift in streaming data using machine learning techniques," *WIREs Data Mining and Knowledge Discovery*, vol. 14, no. 4, July 2024, Art. no. e1536, https://doi.org/10.1002/widm.1536.

[32] A. Cano and B. Krawczyk, "ROSE: robust online self-adjusting ensemble for continual learning on imbalanced drifting data streams," *Machine Learning*, vol. 111, no. 7, pp. 2561–2599, July 2022, https://doi.org/10.1007/s10994-022-06168-x.