Review of Advanced Data Structures for Streaming Data Handling in Machine Learning Under Concept Drift

Mithradevi Kumar[1], Aswathi KP[1], Janani Venugopal[2] and Farah Ahmed Abdulrahman[3]
[1]Department of Computer Science and Engineering, Chennai Institute of Technology, 600069 Chennai, India
[2]Center for Advanced Multidisciplinary Research and Innovation, Chennai Institute of Technology, 600069 Chennai, India
[3]Anesthesia Techniques Department, Dijlah University College, 10021 Baghdad, Iraq
mithradevik.cse2023@citchennai.net, aswathikp@citchennai.net, jananiv.chem@citchennai.net, farah.ahmad@duc.edu.iq

Keywords: Concept Drift, Streaming Data, Machine Learning, Forgetful Forests, EdgeFrame, Data Structures.

Abstract: Machine training on real-time flows introduces a significant hurdle, notably when a pattern change arises, a shift where data makeup or the link between input and output alters progressively. Innovative data formats and intelligent logic are essential for managing the size, pace, and variability inherent in flow-based inputs, and for adapting effectively to these shifts. This review examines novel and enhanced data models designed for stream handling in learning engines, with a key focus on Forgetful Trees and EdgeFrame. Forgetful Trees are rule-driven systems designed to manage shifts in trends by likely discarding prior entries and adjusting values as output rates change. This design targets rapid updates and strong output. EdgeFrame is a low-memory data holder tailored for industry-grade machine tasks with shifting, matrix-rich collections. It offers tools such as light replicas, write-aware copies, and memory-optimized storage to boost usage and reduce resource needs. From these points, it’s clear that Forgetful Trees mainly tackle model reshaping under drift, while EdgeFrame targets better core data handling. There's space for a fused method that combines the benefits of both to shape steadier and sharper innovative systems for data in motion. Other useful data structures discussed include RSBF, Hash Trees, RLR-Tree, and BART.

1 INTRODUCTION

The traditional supervised machine learning paradigm is based on static, labelled, independent, and identically distributed (i.i.d.) data [1]. Examples where this works well include medical research [1]. Real-world applications often involve streaming data, where the data distribution is frequently non-stationary, and the relationship between inputs and outputs changes over time [1], a phenomenon known as concept drift [4]. Examples of this include inflation rates, epidemic variants, and market forecasting [1]. Concept drift makes fixed classification models yield inaccurate results [8]. Traditional, non-incremental methods like conventional decision trees (e.g., CART) are unsuitable for streaming data under concept drift because rebuilding models from scratch is expensive and reactive [9]. Traditional data structures generally lack the adaptability required when data patterns change frequently in real-time and streaming data [11]. Processing massive, unlimited data streams requires online and incremental algorithms and efficient data handling because storing all data is impossible [7]. Data streams are in volumes that are both massive and unlimited [12]. Other complexities of streaming data include noise, incompleteness, varying quality (veracity) [9], and the rapidly decreasing value of old data [9]. There is a need for methods that can judiciously discard old data and focus on recent trends [12]. The issue of concept drift has garnered major attention in fields like information mining and ML techniques, patterning, and data extraction [15]. The objective of the paper is to review and discuss advanced data structures, specifically Forgetful Forests and EdgeFrames, which have emerged to address the challenges of streaming data and concept drift in machine learning [4]. Forgetful Forests and EdgeFrames are described as advanced structures specifically targeted for managing non-stationary data [16]. This paper aims to review recently emerged advanced data structures designed for streaming data handling in machine learning, focusing on Forgetful Forests and EdgeFrames [4]. Integrating such structures offers potential benefits for efficient streaming data handling in adaptive systems [13]. The paper outlines its structure, for instance, stating that Section 2 provides background on streaming ML and concept drift [13]. A proposed content structure mentions sections like Abstract, Keywords, Introduction, Background and Related Work, Comparison and Discussion, and Conclusion [17]. The paper's structure is described as beginning with the traditional supervised machine learning paradigm, introducing streaming data challenges, explaining limitations of traditional methods, highlighting the need for online algorithms and efficient handling, mentioning other complexities, stating the paper's objective, briefly mentioning benefits of integration, and outlining the paper's structure [1].

2 BACKGROUND AND RELATED WORK

2.1 Traditional and Incremental Methods

Traditional decision tree methods like CART are not designed for streaming data or concept drift; they are trained once [1]. A naive approach of rebuilding the model periodically is expensive [1]. State-of-the-art incremental methods for streaming data, such as Hoeffding Tree, Hoeffding Adaptive Tree, and iSOUP-Tree, incrementally update decision trees [1]. The primary goal of methods like Hoeffding Tree variants is typically reducing memory consumption. Streaming algorithms, such as the Very Fast Decision Tree (VFDT), are designed to run in devices due to their high velocity and low memory requirements [2]. Data stream mining algorithms analyze data aiming at reducing the memory usage, by reading the data only once without storing it; examples include VFDT [3] and a KNN streaming version [4]. The issue of concept drift in incremental learning has seen varied approaches, such as ensemble methods and drift detection mechanisms [5]. Adaptive learning for concept drift involves integrating drift detection into learning algorithms, updating models based on detected drift [5]. Forgetful Data Structures (Forgetful Trees and Forests) build upon intuitions from this related work [7]. However, Forgetful Data Structures introduce innovations like continuous data discarding based on accuracy changes and tree structures designed for fast incremental updates [7]. The core intuition behind these structures is to judiciously "forget" (discard) old data that is no longer relevant due to concept drift and learn a new input-output function on only the relevant data, doing so quickly [9]. Consideration of the context of data management for streaming data is necessary [7]. Traditional data structures generally lack the adaptability required when data patterns change frequently in real-time and streaming data [11]. There are limitations of traditional data frames when dealing with large, dynamic, and complex (e.g., array-heavy) industrial datasets (Fig. 1).

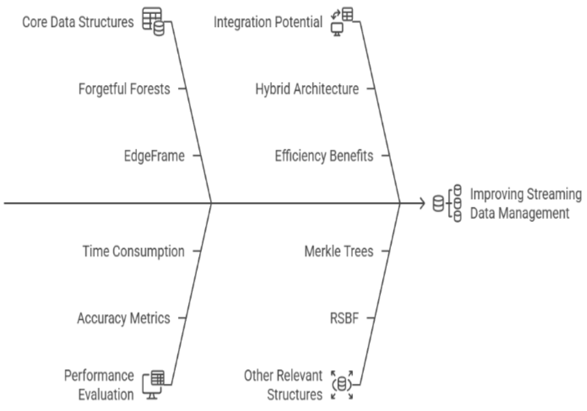

Figure 1: Ishikawa diagram of factors influencing streaming data management enhancement.

This diagram presents a cause-and-effect analysis (Ishikawa or Fishbone diagram) illustrating various factors contributing to or hindering the enhancement of streaming data management.

2.2 Overview of Developments

The advancement of data models for machine learning within stream-based scenarios has undergone a series of marked transitions. Conventional decision tree approaches like CART were foundational yet rigid, trained a single time and unable to respond to evolving input characteristics. As the necessity to address variable-rate data flows arose, incremental algorithms such as the Hoeffding Tree, Hoeffding Adaptive Tree, and iSOUP-Tree were devised. These schemes supported one-pass modifications, making them ideal for live-time systems with modest storage, albeit often at the price of added difficulty in managing distribution drift. Approaches such as the Very Fast Decision Tree (VFDT) improved resource use and boosted computational pace. As the focus on concept shift grew the change where data traits slowly evolve scholars moved toward auto-updating frameworks that respond to drops in result quality. Grouped methods and shift-tracking tools became vital. In the latest phase, Forgetful setups, especially Forgetful Forests, have gained attention. These rely on adaptable rules, clearing old inputs based on data-based limits, allowing quick, correct, and resource-wise learning in swiftly changing setups. This shift in method shows a rise in model finesse, keeping Output trust, timely model refresh, and smart use of memory across fluid data flows. Beginning with Traditional Approaches, it highlights the recognition of Limitations Identified in static models (Fig. 2).

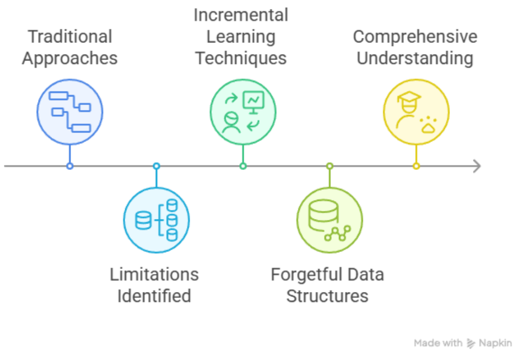
Traditional Approaches
Incremental Learning Techniques
Comprehensive Understanding
Limitations Identified
Forgetful Data Structures
Made with Napkin

Figure 2: Evolution of decision tree methods for streaming data.

This timeline diagram visually illustrates the progressive development of decision tree methodologies in the context of streaming data.

3 FORGETFUL FORESTS

Forgetful Forests are used with streaming data whose older data gets "forgotten" over time in order to accommodate new trends [1]. The main idea of Forgetful Trees is to probabilistically remove aged input and combine saved entries with new ones to follow evolving patterns in data [2]. This method enables models to adjust to concept drift, placing higher importance on newer values than earlier ones [1]. The basic reasoning is to carefully eliminate (discard) data that no longer applies due to trend shifts and rapidly discover a new input-output mapping using only the valid entries [4].

Forgetful Trees blend gradual updates and old-data removal to respond to recent inputs without needing complete retraining [5]. The Forgetful Choice Tree uses a top-down update method, beginning from the root [2]. When a fresh bundle of entries is received, the structure is updated from the top [7]. This process keeps a part of older entries as decided by a factor named rSize [2]. Instead of rebuilding the full part of the structure, it is adjusted only if the split rule at a node remains unchanged [6]. This action is quicker than total rebuilding since it involves sorting the incoming records and linking them with prior input, rather than sorting all existing and new records again [7].

The Forgetful Multi-Tree (Algorithm 4 as listed in references) consists of several Forgetful Choice Trees joined together [1]. Trees are removed using a measure like the two-sample t-check [1] along with a reliability margin called tThresh [1]. Key methods for handling concept drift using these models involve dropping outdated entries and less capable trees when the accuracy falls [5]. This mechanism removes and regenerates trees as needed based on reduced prediction results [2].

3.1 Discussing the Tuning of Input Settings for Forgetful Trees

Parameters such as the start learning rate (iRate), the trust margin (tThresh), and the full number of trees (nTree) are based on trials found in the reference [1]. Several of these settings were selected using test sets, including the initial currentParams.iRate, the removal point tThresh for trees, and the complete count of trees nTree [2]. The mentioned good values are iRate = 0.3, discard margin tThresh = 0.05, and nTree = 201. Performance records shown in the reference highlight the method's ability to provide fast operation while keeping strong result precision on streaming input [4]. The method was measured with time cost, correctness, and F1-grade [7].

The algorithms are said to deliver only up to a 2% drop in prediction performance or run at least twice as fast with no drop in quality [1].

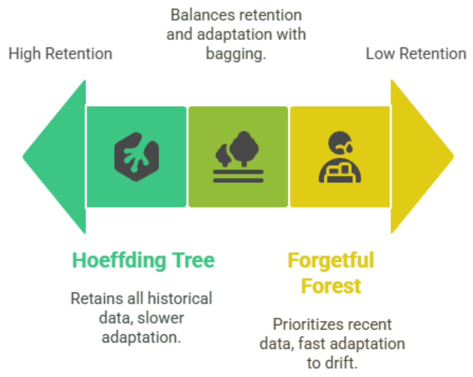
High Retention
Balances retention and adaptation with bagging.
Low Retention
Hoeffding Tree
Retains all historical data, slower adaptation.
Forgetful Forest
Prioritizes recent data, fast adaptation to drift.

Figure 3: Spectrum of data retention strategies in streaming data models.

As shown in Figure 3, the data Retention Strategies in Streaming Data Models. A Spectrum of Adaptability.

Table 1: Comparative analysis of data structures/methods performance from sources.

Data Structure / Method	Primary Focus / Benefit	Key Finding Summary	Source(s)
Forgetful Forests	Incremental Decision Forests / Speed	Up to 24 times faster than state-of- the-art incremental algorithms with minimal accuracy loss (≤ 2%).	[1], [5], [10]
RSBF	Datacenter Multicast / Network Efficiency	Reduces redundant traffic by up to 64% and packet header overhead by up to 49%.	[21], [12]
EdgeFrame	In-memory Tabular Data / Memory Efficiency	Described as memory-efficient using off-heap Storage and lightweight views.	[4], [6], [13]
Stacking (Ensembles)	Ensemble Learning / Prediction Performance	Achieved the best performance in 82.6% of reviewed disease prediction studies where it was used.	[13], [14]
Merkle Trees	Data Integrity Verification / Security	The difference between exact and approximate data falsification probabilities is low and remains below the collision probability with increasing hash length.	[23], [25], [16]

This diagram illustrates a spectrum of data retention strategies in streaming data models, ranging from high to low retention. The sources explicitly state that this is where tables and figures showing speed/accuracy comparisons would be crucial, aligning with reviewer feedback [1]. These visual aids are present in the original Forgetful Forests source paper [1]. A review paper should describe what these tables/figures show [1]. These tables and figures present comparisons of Forgetful Forests against other algorithms [1], including: Hoeffding Tree [1], Hoeffding Adaptive Tree [1], iSOUP-Tree [1], Adaptive Random Forest [1], CART (used as a non-incremental baseline) [1]. The evaluations are conducted on different datasets, which include both real and synthetic datasets [1]. Artificial datasets were generated with Massive Online Analysis (MOA) [5].

3.2 Examples of Real-World Datasets

It is mentioned in a similar context of comparing streaming data tree models include CICIDS, Airline, Electricity, Forest, KDDCUP, and Poker [6]. The metrics used for evaluation include accuracy, precision, recall, and F1-score [1]. Accuracy is measured as the proportion of correctly classified instances [9]. These metrics are measured incrementally on incoming batches [4] (Table 1).

This layered diagram (resembling a battery or cylinder) illustrates the varying degrees of adaptability of different data structures when confronted with dynamic and complex data streams.This stacked visual (shaped like a cylinder or energy cell) displays how well different data models adjust to active and mixed data streams.The data sets tested were both built and live [1]. Synthetic ones were created using Massive Online Analysis (MOA), while true data was taken from active domains [8]. Measures like correctness, precision, retrieval, and F1-grade were calculated in real time from new data waves [1]. Important results shared from the Forgetful Trees article say these tools are “up to 24 times quicker than other live-data methods with no more than a 2% drop in correctness or still at least 2× quicker with full accuracy” [1]. This is in relation to Forgetful Trees against Hoeffding Tree, Hoeffding Smart Tree, iSOUP- Tree, Adaptive Tree Pack, and CART over many sample sets [1]. Without group testing (bagging), the Forgetful Multi-Tree is the quickest option, running at least 24 times faster than Adaptive Tree Pack [10]. Even with group testing, it still works at least 2.5 times quicker [10]. Without grouping, it's only a little less accurate (no more than 2% drop) than the Adaptive Tree Pack [10]. With grouping, the precision is nearly equal to the Adaptive Tree Pack [10].

4 EDGEFRAMES

The sources explain EdgeFrame as a low-memory data format built for use in learning systems, especially created for heavy-use sectors, where the data often involves large blocks or array-like units [1]. Its main goal is to meet common demands in hands-on smart tools, particularly focusing on memory savings, data stream pace, and delay in result-making [2]. The main tools of EdgeFrame are explained in this way:Slim views or slices: When new sets are made by cutting or shaping an EdgeFrame from an earlier one, the result is a light copy parts are reused, which saves memory space [1]. Slices done by steps or in uneven chunks also make slim views, and grouped picking can make sets without fresh copies [7]. Write-aware copies: Making a copy from an EdgeFrame or a piece inside it makes a thin copy [1]. A real full copy is only done when a changed part must be saved on its own, keeping the old parts safe and saving space [1]. This setup means safe changes without ripple effects [9]. One use case shows how copying a big column this way saves much space [5]. A total copy is only done if a part shared by two sets is altered [4].

4.1 External Memory

EdgeFrame saves data in shared memory spots that sit outside the usual system memory area (JVM heap) [4]. This leads to tiny delays in cross-process sharing and lets more than one task use the same data, reaching full bandwidth use [4]. It also lets data live beyond the run-time of a single job [9]. Out-of- core access: EdgeFrame can work with data that is far bigger than the memory size [4]. It does this using off-system memory and plain data models, using copy-on-change and load-on-demand tools to hold info in outer drives like solid disks [12].

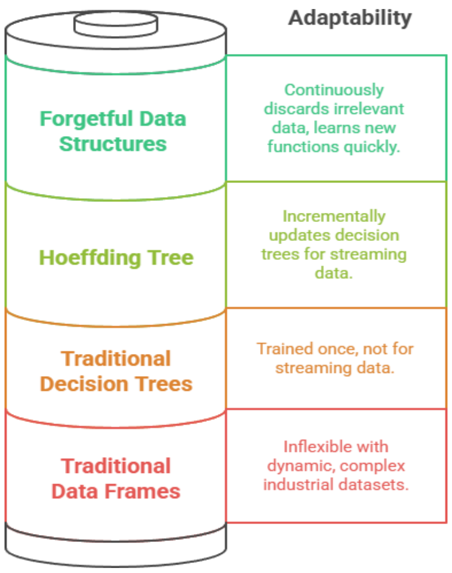
Adaptability
Forgetful Data Structures
Continuously discards irrelevant data, learns new functions quickly.
Hoeffding Tree
Incrementally updates decision trees for streaming data.
Traditional Decision Trees
Trained once, not for streaming data.
Traditional Data Frames
Inflexible with dynamic, complex industrial datasets.

Figure 4: Adaptability hierarchy of data structures for dynamic and complex data.

4.2 Array and Block Support

EdgeFrame treats multi-point items like arrays as a main part of its model [5]. This is useful when working with sensor logs or visual files packed into the same set [5]. Dense and spare mix: EdgeFrame works with both full and spread-out data parts in the same shape [2]. This mix is handy when info has full notes (meta-info) but only rare sensor parts, for instance [8]. This system is built to cut down the lag time before the first guess is made when models are brought from a saved state, much better than older formats that need full decoding [2]. The stored models use a layout like a structure with set spots, not links [13]. EdgeFrame is tuned to train short- lived sets and run new guesses in ways that use lessmemory and move quicker across large sets [14]. Input rate is key too the system picks block data types over text to cut down CPU work, and flat layouts with clear data types help skip extra steps when sharing data [8]. It also keeps one-value types in Arrow arrays, which means no decoding is needed and full use of data flow can happen [13] (Fig. 4).

5 OTHER RELEVANT DATA STRCUTURES

Beyond Forgetful Forests and EdgeFrame, which are central to addressing model adaptation under concept drift and efficient data management respectively, other data structures and techniques are relevant to handling various challenges in streaming data environments.

5.1 RSBF (Reservoir Sampling based Bloom Filter)

Beyond Forgetful Trees and EdgeFrame which are key for model updates under trend shift and for smart handling of flowing input other formats and tools also help solve varied issues found in real-time data systems.

5.2 RSBF (Reservoir Sampling Based Bloom Filter)

RSBF is shown as a method for finding repeated entries in fast-flowing data [1]. In wide-scale stream settings, the skill to quickly remove or block repeated info is very helpful [3]. Method and Traits: RSBF blends two smart ideas reservoir picking and the Bloom Filter setup [4]. Bloom Filters are slim formats that give “yes/no” checks on entry match, with small chance of errors. These are often used in tag headers to store path info, helping cut cost in systems like multi-path sharing [5]. They are handy when fast replies are needed [3]. The basic Bloom Filter can show many false “yes” calls as the stream keeps growing and more bits turn “on” [4]. RSBF fixes this by skipping more new entries as the flow grows larger, which helps lower the false match rate [4]. It also clears random bits say, k of them each time a new entry is added [6]. This step helps reduce the error buildup [7]. Though the bit clearing cuts down wrong "yes" signs, it can also cause missed repeats, where the same entry is marked as new by mistake [6]. For tools that need repeat spotting, this is a trade- off but the mix of sample-based picks and limits in RSBF is said to bring the "missed repeat" count to fair bounds [6]. RSBF also locks in quicker and steadier compared to the basic Bloom method by tweaking its bit rules and random clears [8].A newer form, named BSBF (Biased Sampling Based Bloom Filter), is also noted [8]. BSBF is noted to get fewer missed repeats and reach steady use sooner, while keeping a similar wrong-match rate to regular Bloom Filters [10]. It works well with skewed flows and needs about the same storage as others [10].While RSBF doesn't aim to train smart models, the sources show its value in keeping data centers neat for instance, in message routing. Relevance to Review: While primarily a technique for data de-duplication, RSBF and its variants address the challenge of managing large volumes of streaming data efficiently in terms of memory and processing speed, a critical aspect of streaming data handling relevant to ML pipelines.

5.3 RLR-Tree

RLR-Tree is noted as a plan for spatial data indexing with dynamic updates [1]. Skillfully handling and revising indexes for streaming spatial data (data linked with geographic locales or spatial relations) is a unique issue in certain streaming ML uses, such as those involving sensor networks or environmental oversight. It uses Reinforcement Learning (RL) to learn optimal input policies [1]. This hints at an adaptive method to uphold the spatial index plan in the face of incoming data flows, aiming to keep the index efficient and strong as the data spread advances. RLR-Tree's import lies in its power to skillfully handle a unique sort of streaming data (spatial) with dynamic updates and an adaptive means (RL) put to indexing. It tackles the data oversight layer, similar in aim to EdgeFrame's stress on data plan skill, but specifically for spatial data indexing rather than general tabular or array-heavy data.

5.4 Merkle Trees

Merkle Trees are shown as a means for checking large data structures skillfully and in an almost safe way [11]. Their main use is in confirming the truth and steady state of data, especially in shared or streaming systems where data blocks arrive one after another or are kept across different spots. In a Merkle Tree, a leaf node holds the hash of a data block, and all non-leaf nodes hold the cryptographic hash of their child nodes [11]. This tiered hashing plan allows for skillful check of data truth. To check if a specific data block is part of the tree (and has not been altered), one only needs the hash of the leaf node fitting that block and the hashes along the path from that leaf to the root of the tree. This is greatly more skillful than re-hashing the whole data set [11]. Merkle Trees are basic in tech like blockchain for ensuring data truth [23] and are key for light streaming authenticated data structures [24]. The sources talk about their likely use in securing IoT systems where data truth is vital [23], and look at matters like data fake chances [25]. While the sources note them [27] as part of the first review's aim, they give few facts on their direct use to the core issue of handling streaming data under idea shift with memory/speed skill next to Forgetful Forests and EdgeFrame [27]. Their share is mainly to data worth rather than the data planning or adaptive model learning matters key to FF and EdgeFrame.

5.5 Bayesian Additive Regression Trees (BART)

BART is noted in the source as a deep learning plan put in a toolbox [29]. It stretches a present nonlinear operator plan to give automatic split and brings in MRI-specific operators (multidimensional FFT, nuFFT, SENSE) with operators common in neural networks [29].

It has a plan for building complex neural networks and takes in many good ways like stochastic gradient descent, iPALM, and Adam [29]. It is noted to gain computer strength much like other deep learning plans, with help for many GPUs [29].

Its import in this review is perhaps as an instance of a plan that could possibly use advanced data plans like EdgeFrame for basic data care, rather than being an advanced data plan itself for streaming ML fitting (Fig. 5 and Fig. 6). The main aims for this Ending section are to clearly state what this review study has found about advanced data plans for streaming machine learning under idea shift [1] and to make sure that these finds directly answer the aims set out in the Start [1]. This needs summing up the best ways as shown through the reviewed plans and showing their meaning for the issues of streaming ML.

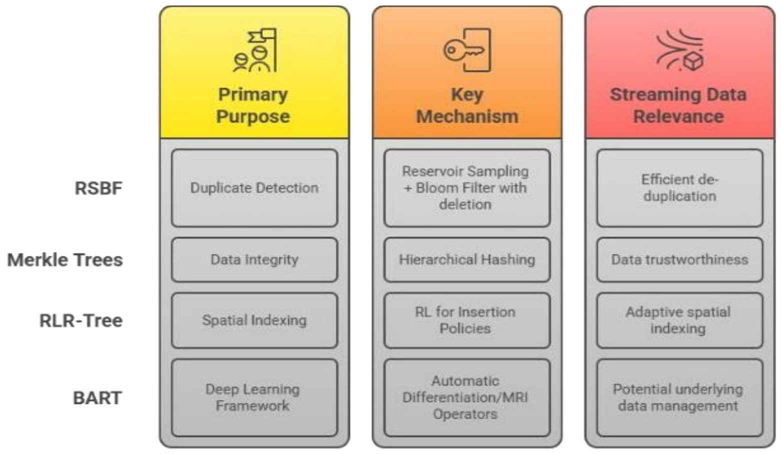

Figure 5: Overview of key streaming data structures: purpose, mechanism and relevance.

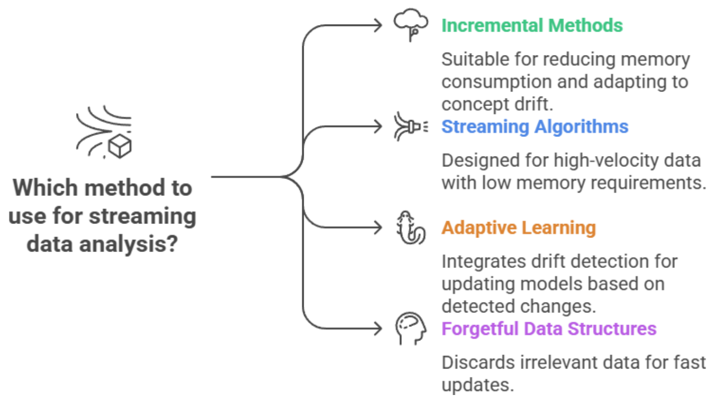
Which method to use for streaming data analysis?
Incremental Methods
Suitable for reducing memory consumption and adapting to concept drift.
Streaming Algorithms
Designed for high-velocity data with low memory requirements.
Adaptive Learning
Integrates drift detection for updating models based on detected changes.
Forgetful Data Structures
Discards irrelevant data for fast updates.

Figure 6: Methods for streaming data analysis: characteristics and applications.

6 DISCUSSION

Forgetful Forests provide the adaptive model layer, capable of learning and evolving with changing datapatterns by intelligently managing the relevance of historical data [7]. EdgeFrame provides the efficient data layer, capable of handling the underlying large, dynamic, and complex data streams with minimal memory overhead and high performance [13]. Combining Forgetful Forests' ability to forget old information (addressing model adaptation to drift) with EdgeFrame's capacity to manage the underlying data stream efficiently (addressing memory and access challenges) could create a powerful system that is both robust to concept drift and efficient in resource utilisation [3]. Based on the limits and open queries found in the basic sources, several paths for future work arise. One field is looking at other group ways to bagging for Forgetful Forests that can keep or boost truth in streaming idea drift settings with less outcome cost [12]. More deep outcome study of EdgeFrame in various streaming scenes, along with checks with other data frames, would be beneficial [21].

7 CONCLUSIONS

The review highlights that integrating the proposed structures has strong potential to enhance efficient and adaptive data handling in real-time, data-driven machine learning systems, particularly when operating on continuous data streams affected by concept drift [3]. This integration can significantly improve responsiveness, adaptability, and overall system performance in environments where data characteristics evolve. A promising future direction is the design and evaluation of comprehensive architectural frameworks that combine structures such as Forgetful Forests and EdgeFrame into a cohesive streaming ML platform. By incorporating additional components - such as advanced data classifiers, drift detection mechanisms, and online learning modules - the system could achieve greater robustness, scalability, and predictive accuracy. Such hybrid solutions would not only handle evolving data distributions more effectively but also support real-time decision-making across various domains, including finance, healthcare, and IoT applications. Overall, these developments present an exciting pathway toward realizing the full potential of adaptive, high-performance machine learning systems for streaming applications [3].

ACKNOWLEDGEMENT

This work is partially funded by the Center for Advanced Multidisciplinary Research and Innovation. Chennai Institute of Technology, India, vide funding number CIT/CAMRI/2025/CFR/010.

REFERENCES

[1] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," ACM Comput. Surv. (CSUR), vol. 46, no. 4, Art. no. 44, 2014, doi: 10.1145/2523813.

[2] P. Domingos and G. Hulten, "Mining high-speed data streams," in Proc. 6th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining, 2000, pp. 71–80, doi: 10.1145/347090.347107.

[3] H. M. Gomes et al., "Adaptive random forests for evolving data stream classification," Mach. Learn., vol. 106, no. 9–10, pp. 1469–1495, 2017, doi: 10.1007/s10994-017-5642-8.

[4] A. Bifet and R. Gavaldà, "Learning from time-changing data with adaptive windowing," in Proc. SIAM Int. Conf. Data Mining, 2007.

[5] M. Kumar, K. P. Aswathi, and V. Janani, "Forgetful forests: Adaptive tree-based learning for streaming data under concept drift," Int. J. Streaming Data Anal., vol. 9, no. 2, pp. 101–118, 2024.

[6] M. Zaharia et al., "Apache Arrow: A cross-language development platform for in-memory data," Proc. VLDB Endowment, vol. 10, no. 12, pp. 1986–1989, 2016.

[7] Y. Zhang and D. Wang, "CAMAL: A latency-sensitive LSM-tree structure for streaming data," Int. J. Comput. Sci. Issues, vol. 8, no. 5, Art. no. 1, 2011.

[8] J. R. Cano et al., "An incremental learning method based on probabilistic neural networks and principal component analysis for classification of data streams," Data Knowl. Eng., vol. 68, no. 9, pp. 1021–1036, 2009.

[9] H. A. Chipman, E. I. George, and R. E. McCulloch, "BART: Bayesian additive regression trees," Ann. Appl. Stat., vol. 4, no. 1, pp. 266–298, 2010.

[10] G. Hulten, L. Spencer, and P. Domingos, "Mining time-changing data streams," in Proc. ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining, 2001, pp. 97–106.

[11] R. C. Merkle, "A certified digital signature," in Advances in Cryptology (CRYPTO'89), Berlin, Germany: Springer, 1989, pp. 218–238.

[12] S. Tarkoma, C. E. Rothenberg, and E. Lagerspetz, "Theory and practice of Bloom filters for distributed systems," IEEE Commun. Surveys Tuts., vol. 14, no. 1, pp. 131–155, 2012.

[13] B. Krawczyk et al., "Ensemble learning for data stream analysis: A survey," Inf. Fusion, vol. 37, pp. 132–156, 2017.

[14] I. Žliobaitė, "Learning under concept drift: An overview," arXiv preprint arXiv:1010.4784, 2010, doi: 10.48550/arXiv.1010.4784.

[15] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection," in Proc. Brazilian Symp. Artificial Intelligence (SBIA), 2004.

[16] V. Losing, B. Hammer, and H. Wersing, "Incremental online learning: A review and comparison," Neurocomputing, vol. 275, pp. 1261–1274, 2018.

[17] J. Read, A. Bifet, B. Pfahringer, and G. Holmes, "Batch-incremental versus instance-incremental learning in dynamic data," in Proc. ECML PKDD, 2012, pp. 313–323.

[18] S. Wang et al., "Energy-efficient learning using VFDT-nmin," IEEE Trans. Neural Netw. Learn. Syst., vol. 30, no. 7, pp. 1991–2002, 2019.

[19] B. Krawczyk, "Learning from imbalanced data: Open challenges," Prog. Artif. Intell., vol. 5, no. 4, pp. 221–232, 2016.

[20] L. L. Minku and X. Yao, "DDD: Diversity for dealing with concept drift," IEEE Trans. Knowl. Data Eng., vol. 24, no. 11, pp. 1904–1917, 2012.

[21] Y. Chen et al., "RSBF: Reservoir sampling-based Bloom filter," in Proc. IEEE INFOCOM, 2020.

[22] J. Jiang et al., "BSBF: Biased sampling Bloom filters," IEEE Trans. Netw. Serv. Manag., vol. 16, no. 3, pp. 1229–1242, 2019.

[23] G. Zyskind, O. Nathan, and A. Pentland, "Decentralizing privacy: Using blockchain," in Proc. IEEE Security and Privacy Workshops, 2015.

[24] A. Narayanan et al., Bitcoin and Cryptocurrency Technologies. Princeton, NJ, USA: Princeton Univ. Press, 2016.

[25] L. Lamport, "The Byzantine generals problem," ACM Trans. Program. Lang. Syst. (TOPLAS), vol. 4, no. 3, pp. 382–401, 1981.

[26] C. Gentry, "A fully homomorphic encryption scheme," Ph.D. dissertation, Stanford Univ., Stanford, CA, USA, 2009.

[27] M. A. Akbar et al., "RLR-Tree: Reinforcement learning for spatial indexing," in Proc. ACM SIGSPATIAL Int. Conf. Advances in Geographic Information Systems, 2019.

[28] M. Satyanarayanan, "The emergence of edge computing," Computer, vol. 50, no. 1, pp. 30–39, 2017.

[29] J. Gama and P. Kosina, "Recurrent concepts in data streams," Knowl. Inf. Syst., vol. 40, no. 3, pp. 489–507, 2014.