Advanced data structures for streaming data handling in Machine learning

[1]Mithradevi Kumar, [2]Aswathi. KP, [3]Janani V,[4] Lara Mwafa, [5]Q Othman Faisal Abdullah
[1]Department of Computer Science and Engineering, *Chennai Institute of Technology,* Chennai, India
[2]Department of Computer Science and Engineering, Chennai Institute of Technology, Chennai – 600069
[3]Center for Advanced Multidisciplinary Research and Innovation, Chennai Institute of Technology, Chennai 69
[4] *Computer Engineering Technology, Al-Turath University, Baghdad, Iraq*
[5] Department of Radiological Techniques, Dijlah University College, Baghdad, Iraq
mithradevik.cse2023@citchennai.net, aswathikp@citchennai.net, jananiv.chem@citchennai.net, *lara.Mwafaq@uoturath.edu.iq,*
othman.faisal@duc.edu.iq

Keywords: Dynamic data handling, Forgetful Forest, Edge frames, hybrid approach, data classifier and streaming data

Abstract: This paper covers the problem of handling dynamic data in machine learning by exploring some recently emerged data structures including Forgetful Forests and EdgeFrames. These advanced structures have specifically been targeted for managing non- stationary data. These are particularly important for the applications involving information with changing dynamics. It gives an overview of traditional and advanced structures. It introduces a hybrid method, which uses Forgetful Forests and EdgeFrames to deal with the large-scale data, adopting conventional data structures when the size of data is not large. Therefore, the approach offers an adaptive solution in handling static as well as dynamic data in the machine learning systems, incorporating the use of a classifier for the process of classification. This work supports the development of an evolving data structure that can adapt to dynamic and updated datasets in machine learning.

1 INTRODUCTION

Data structures play a very important role in the storage, retrieval, and manipulation of data and therefore are fundamental to computer science. They are crucial for arranging data efficiently in machine learning, which enables various techniques and applications to work effectively. However, traditional data structures struggle with the growing volume of dynamic, non-stationary data, which changes over time. It works perfect for static or semi- static data, but it goes not so well with non-static changing data.

To fill out the gap in handling dynamics changes on data, researchers have devised other new data structures specifically oriented in managing dynamic data. Innovations such as Forgetful Forests and EdgeFrames emerged within it. Forgetful Forests mimic the human memory by storing only dynamic data needed for current situations and adapting to changes in the environment. Similarly, EdgeFrames offer a flexible framework for the management of relational and graph-based data by dynamically adjusting to changes in the data structure.

This review explains the role of both conventional and newly introduced data structures for managing static and dynamic data. It explains the working of these advanced structures and provides suggestions for modifications to make them more efficient in terms of time and space.

2 RELATED WORK

In the context of data stream mining, one of the most difficult tasks is the detection of concept drift as data in real- time environments is dynamic. Concept drift is defined as the change in the underlying data distribution, which could be degrading the performance of the machine learning models over time. A lot of research has gone into developing robust methods for detection and adaptation to such changes with varying degrees of accuracy and computational efficiency (Smith et al., 2024).[1]

A crucial issue when using concept drift detection is that the system needs to adjust for changes in data distribution, especially if the drift occurs slowly or over a long time. Most drift detection algorithms base their indication of a drift on a change in the posterior distribution or error rate. However, these methods often suffer from slow adaptation to drift, poor sensitivity to gradual changes, high false alarm rates, and high computational complexity, which makes them impractical for real-time applications (Johnson et al., 2024).[2] Besides the concept drift detection, novelty detection in the context of one-class and multi-class classification problems has received attention (Kumar et al., 2024).[3] Novelty detectors are designed to identify instances that deviate from known patterns. Most of the methods, however, are concerned with the one-class classification approaches since multi-class approaches have required deeper analysis. Even more, there is a shape problem when data clusters shape to be the data instances wrongly classified into or not classified into the class. These constraints hinder the ability to fully utilize streaming data, which often includes data points that may be useful but are not captured by traditional classification models (Singh et al., 2024).[4]

In order to identify concept drift, random forests and decision trees have demonstrated promising algorithms. Lee et al. (2024) propose Forgetful Random Forests and Forgetful Decision Trees as effective methods for managing concept drift in streaming data.[5]

They provide the memory efficiency of forgetting information at a cost in the sense of high accuracy. Forgetful Random Forests, if used with bagging, are as accurate as the current state-of-the-art incremental Random Forest algorithms but much faster. Moreover, Forgetful Trees are faster and more accurate than traditional decision tree algorithms and thus very well suited for large-scale streaming environments (Chen et al., 2024).[6]

The issue of online filtration, or the removal of redundancy in real-time while data is being streamed, is also resolved by a newer generation of algorithms, such as those based on

Bloom Filters (Patel et al., 2024). [7] It is nearly error-free and efficiently eliminates duplicates from large collections of streaming data. Bloom Filters are effective in situations where quick decisions are essential because they exhibit better false negative rates and quicker convergence to stability when combined with reservoir sampling (Zhang et al., 2024). [8]

Additionally, efforts have been made to improve the concept of drift's adaptation in nonstationary environments by investigating clustering-based methods.

Even in situations with extreme verification latency, it has been demonstrated that the SCARGC, a clustering and classification combination, can produce highly accurate classifiers using very small amounts of labelled data (Nair et al., 2024).[9] This offers a good substitute for situations requiring immediate adaptation because it is quicker and more accurate than the traditional methods.

In summary, although very good progress has been made to date in the detection of concept drift and adaptation, there are still challenges facing these approaches, such as slow adaptation to drift, poor sensitivity to gradual changes, and high computational complexity. The goal of future research is to increase the effectiveness and precision of drift detection algorithms, particularly in large-scale, real-time streaming environments (Suri et al., 2024).[10]

II. DRIVING FACTORS FOR ADVANCEMENTS IN DATA STRUCTURES IN MACHINE LEANRING

The need to handle and manage dynamic data is what drives the development of novel data structures in machine learning.

A. Data stream

A data stream is an orderly and continuous sequence of data elements that results from the passage of time through one or more sources. A data stream's data elements can be processed and retrieved in the order that they are generated because they are either explicitly or implicitly recorded in time.

This detection is one of the major challenges in data stream mining. The data points are generated sequentially and independently under some probability distribution in the data stream (Kifer et al., 2004). [11] A data stream is constantly updating in real time, therefore processing methods differ from normal databases.

B. Data stream Challenges

To make analysis effective, working with data streams presents a number of serious challenges that have to be overcome: The first problem is the velocity of data streams, which overburden processing power because they are arriving at such a constant high speed.

Therefore, efficient analytics have to keep up with this pace.It is impossible to store all the data, as data streams are in volumes which are both massive and in volumes that are simply unlimited. Therefore, only online and incremental algorithms are required so that all this information doesn't have to be stored full. Complexity is increased by the different data streams, which in this case include multimodal and varied data from so many sources, often carrying noise and incompleteness. The other problem is that of veracity; errors, noise, and outliers can make data quality vary wildly. Strong methods are needed to handle poor stream quality. Data loses value very quickly because contexts change over time. Use windowing and forgetting factors to emphasize the recent data. Because, in the real world, the phenomena are subject to concept drift, which implies variability because the data distribution shifts from stationary assumptions over time. Virtualisation is mandatory due to the fact that data streams need to be processed in a distributed, virtualised infrastructures prone to failure. It means that fault tolerance becomes really important in such environments. Anonymisation and privacy-preserving analytics are required since data streams may include sensitive personal information. In recent years, some authors of the literature on ML for data streams have started to extend their algorithms to stationary scenarios.

II. STATIC DECISION STRCUTURES:UNDERSTANDING CART AND CTREE IN CLASSIFICATION AND REGRESSION

CART and CTREE are decision tree algorithms which recursively splits data to classify or predict outcomes, though CART only focuses on partitioning, whereas CTREE used statistical tests for preventing overfitting as well as selection bias.

A. CART Tree (Classification and Regression Trees)

A decision tree algorithm that creates a prediction by recursively partitioning data according to attributes. It is used in regression analysis and classification.

The predictor space, which consists of all the predictor values that fall into distinct regions, or nodes, is divided in CART to create a model that is as homogeneous as possible with respect to the result.

At root, it starts with the use of all the observations, a predictor variable with one of its cut points finding an optimal split. All of these can be continued in succession for each of the steps trying to enhance these splits.

Because large trees tend to overfit the training data, size controls on the tree such as the minimum number of cases per node are used too. Additionally, one can use cost complexity pruning in order to determine a best subtree by removing unwanted branches. [12]

B. CTree(Conditional Inference Trees)

Classification and Regression Trees, or CART Trees, are decision tree algorithms that divide data recursively based on attributes in order to produce a prediction. It is employed in classification and regression analysis.
To produce a model that is as homogeneous as possible with regard to the outcome, CART divides the predictor space, which is made up of all the predictor values that fall into different regions, or nodes.
Global null hypothesis: Permutation test for independence of the outcome and any predictor variables; depending on the result of the test, further splits shall or not be performed with their respective p-value serving as the stopping criterion.
CTREE eliminates the major disadvantage of CART and other classical methods for decision trees, such as selection bias in favour of predictors with more than enough possible split points [12-13].

III. ADVANCED DATA STRCUTURES IN STREAMING DATA

A. Decision trees

One of the most popular methods of data mining is systems that generate classifiers.
Classification algorithms in machine learning, including decision tree algorithms, are capable of processing large volumes of data and are widely used to classify knowledge based on training datasets and class labels.
Decision trees are mainly used for classification purposes and are a popular model in data mining due to their simplicity and accuracy across various data types.
A decision tree consists of nodes and branches, where each node represents features to be classified, and each branch defines the possible values that the node can take. Some of the decision tree algorithms are ID3, Classification and Regression Trees (CART), Chi-squared Automatic Interaction Detector (CHAID), Multivariate Adaptive Regression Splines (MARS), Generalized Unbiased Interaction Detection and Estimation (GUIDE), Conditional Inference Trees (CTREE), Classification Rule with Unbiased Interaction Selection and Estimation (CRUISE), and Quick Unbiased and Efficient Statistical Tree (QUEST).
Advantages and disadvantages of Decision Trees
The DT algorithm falls under the supervised learning family. It mainly builds a model to predict the class or value of target variables by learning decision rules from training data. The DT algorithm may be used for regression and classification problems. However, it has its advantages and disadvantages.[14]

B. Random Forests(RF)

A Random Forest (RF) is an ensemble of classification or regression trees(CART). CART uses recursive partitioning, where the data is repeatedly partitioned into potentially high-dimensional rectangular subsets of the predictor space. It does so such that within the partition, response data should be as homogeneous as possible. Individual trees in a forest can have large variance in that small perturbations in the training data cause considerably different models. This often translates to poor generalization. However, an ensemble of many trees usually does a better job of improving the generalization of the model.
Random Forest(RF) fits a large number of individual trees, sometimes in hundreds or thousands, and combines their predictions. RF uses classification trees (CTs) for response variables that are categorical, and regression trees (RTs) for continuous responses. The technique fitted each tree on a bootstrap sample of the training data. This is a random sample taken with replacement. In other words, there are approximately 63.2% unique records, known as in-bag samples. The samples that are not selected are called out-of- bag samples and are used to estimate the model's error.
Unlike the bagging ensemble method, RF uses only a random subset of predictors at every split for determining the best predictor from all predictors. This would reduce the chances of overfitting because of decorrelated trees. Many implementations use the parameter that represents the number of predictors considered at every split. RF usually grows deep trees, unpruned, and with many splits; this can lead to some terminal nodes having very few observations. Model parameters can also control the minimum number of observations in terminal nodes as well as the depth of the trees, although the available options for controlling these parameters vary between implementations.[15]

C. Blooming Filter(BF)

A Bloom filter is a data structure which can store the elements of a set in a space-efficient manner, if a small error is allowed when testing for elements in the Bloom filter.

1. Reservoir Sampling based bloom filter(RSBF)

The Reservoir Sampling based Bloom Filter (RSBF) approach is a new combination of techniques from reservoir sampling and the Bloom Filter method. However, as the stream grows larger, RSBF suffers under a higher false positive rate by which different elements may spuriously report as duplicates. Reservoir sampling helps to cut down on this by refusing more elements as the size of the stream increases, further reducing the false positive rate.
Although this, when the number of elements increases, nearly all of the bits in the Bloom Filters are set to 1 and therefore the false positive rate increases. To counter this, RSBF deletes k randomly chosen bits from the structure each time an element is inserted where k is the number of bits from the structure. This deletion operation can lead to false negatives, where duplicates are mistakenly identified as distinct. Applications that require duplicate detection often demand low false positive and false negative rates, so RSBF balances these by

using reservoir sampling to keep false positives lower while managing false negatives by rejecting elements based on a decreasing insertion probability.
RSBF adapts dynamically to evolving data streams, and though repeated rejections of elements with a decreasing insert probability can introduce more false negatives, it can control the false positive rate. Once the insertion probability drops below a certain set threshold, an element is reported as distinct and inserted. This combination of reservoir sampling and thresholding reduces the false negative rate to acceptable levels, and RSBF demonstrates faster convergence to stability compared to the standard Bloom Filter method, through the dynamic adjustment of bit settings and deletions.[16]

2. *Biased Sampling based Bloom Filter(BSBF)*

The Biased Sampling based Bloom Filter (BSBF) approach is a variation of the traditional Bloom Filter method designed for data de-duplication purposes in streaming datasets. It builds upon the Reservoir Sampling based Bloom Filter (RSBF) approach, with a key modification in its insertion criteria. While RSBF relies on the probability-based insertion mechanism, BSBF inserts all elements that appear in the stream and are reported as distinct without depending on the specific insertion probability. Every element is hashed by k different uniform hash functions, which set corresponding bits in the Bloom Filters. BSBF suffers from unbounded insertion, which leads to a high false positive rate (FPR). To handle this, BSBF deletes k randomly selected bits from the structure upon each insertion, thereby balancing the tradeoff between false positive rate (FPR) and false negative rate (FNR). The deletion might involve bits that have already been set to 0, but this helps control the overall performance of the structure. Like RSBF, BSBF converges faster to stability, which means the system adapts quickly and dynamically to the evolving data stream. This method is particularly effective for real-time de-duplication or data redundancy removal in streaming data, a challenging problem in big data environments.

By combining the biased sampling with Bloom Filters, BSBF improves FNR and achieves quicker convergence to stability while still maintaining a comparable FPR compared to the standard approach of Bloom Filter. This system also adapts very well to the biased nature of the incoming stream, improving accuracy in many real-world applications with very low tolerance for FNR. Further modifications of the BSBF approach with various deletion strategies may help overcome the drawbacks of multiple element deletions and thus improve its overall performance. A randomized load-balanced algorithm has been proposed to achieve a balance in the performance of the FPR and FNR metrics. These features make BSBF highly efficient for real-time, large-scale applications, as proved to have superior performance regarding FNR and convergence rates. It also has similar memory requirements as other algorithms.[16]

D. *Merkle trees*

Merkle Trees, named by Ralph Merkle, present a method for checking large data structures efficiently and in an almost secure manner. For any Merkle Tree, a leaf node contains the hash of a data block; all non-leaf nodes hold the cryptographic hash of its child nodes.

The "path" in a Merkle Tree is a sequence of nodes and hashes required to verify a data block's integrity. If data block or its path are altered, the Merkle Root changes, allowing data tampering or corruption to be detected.

Merkle Trees are widely used in blockchains, data integrity verification and distributed systems to ensure data authenticity and consistency.[17-18]

E. *RLR-Tree*

The results on RL applications to sequential decision-making problems reveal that RL is effective. In this work, proposing to model the process of new object insertion based on the combination of Markov Decision Processes and then apply RL to learn key optimal policies for those operations. It can be apply this approach to a number of existing systems with spatial data indexing, as R-Trees. This model trained can be used in constructing algorithms for R-Tree, forming the RLR-Tree that can assist existing R-Trees with dynamic updates. It has improved the R-Tree insertion algorithm by incorporating learned models of the key operations in constructing the RLR-Tree. For the update of dynamic data, trained models allow the efficient addition of new data records into the R-Tree while retaining minimal performance degradation even if training is not often repeated. Experimental results show that the RLR-Tree always outperforms traditional R-Tree variants in range queries, KNN queries, and spatial joins. The states and action in the design of the RLR-Tree seem to have improved the performance and can handle dynamic updates as well; however, the improvement in bigger datasets is noteworthy. Next steps in the future: Further refinements in design of states and action.[19]

F. *Bayesian Additive Regression Tree(BART)*

Bayesian additive regression tree is an ensemble method, which combines multiple decision trees within a Bayesian framework for the estimation of uncertainty in predictions. This technique is usually used for regression problems where it aims to approximate the functional relationship among outcomes and predictors through a sum-of-trees model. Each individual tree explains just a small variation in the outcome, making BART similar to Boosting but with a different approach because it imposes a probability model. This model includes priors for various elements within each tree and

a data likelihood. BART consists of T number of additive trees, where priors are selected to make sure individual trees remain small in depth and produce moderate predictions relative to the overall sample mean. These "regularized trees" only account for small portions of the overall outcome variance, making them weak learners like gradient boosting.
This would give us priors like $\alpha = 0.95$ and $\beta = 2$. This choice of priors favors smaller trees of depth 2, effectively shrinking the leaves toward the sample mean and avoiding overfitting. The regularization mechanism is very much similar to the shrinkage parameter in gradient boosting. The model fitting process uses a Markov Chain Monte Carlo algorithm, specifically backfitting. In this approach, the Gibbs sampler draw conditions on residuals and the tree structure, followed by sampling of the leaf node means from a normal distribution. [12]

G. *Forgetful Forests(FF)*

Forgetful Forests are used with streaming data whose older data gets "forgotten" over time in order to accommodate new trends. This approach allows models to react to a concept drift, giving more value to recent data than to older data. The system learns by casting off branches which no longer contribute high information.

Formula for Accuracy Measurement:
(1)
Where:

- True Positive (TP): Correct positive classification.
- True Negative (TN): Correct negative classification. Operations of Forgetful Forests:

The Most frequently used function in forgetful forest in union find operation. Forgetful forests incorporate incremental learning and data forgetting to adapt to new data without full retraining. They adapt to changes in the underlying data distribution.[20]

H. *EdgeFrame(EF)*

EdgeFrame is a memory-efficient data structure used in graph-based systems to represent and manage relationships between entities. It enables dynamic management of relationships and supports arrays as a first-class data type, allowing image and sensor data to be represented in the same dataset.

Efficiency Gain (EG):

$$\text{Efficiency Gain} = \frac{\text{Memory Cost (Traditional Copying)}}{\text{Memory Cost (EdgeFrame Shallow Views/Forks)}} \qquad (2)$$

It is a ratio that demonstrates the memory investments when using EdgeFrame's shallow views and forks in contrast to conventional copying.
Traditional Copying: The Memory used for operations that encompass copying whole datasets.
EdgeFrame Shallow Views/Forks: Instead of copying data, memory is used to generate lightweight references to it.
The basic operations in EdgeFrames include insertion, deletion, update, query processing, windowing, filtering, streaming joins, time-based expiration, event handling, and stream processing. It's built to be memory-efficient and, in fact, can process big datasets that are larger than RAM through off-heap storage and simple data architectures. This involves the usage of copy-on-write forking along with out- of-core processing so as to save more data in external storage like SSDs. EdgeFrame is essentially a dynamic graph-based input system with key components like processing units, forking, off-heap storage, and out-of-core processing. It processes data through a pipeline where data is distributed to processing units, forked, stored in off-heap storage, and processed in out-of-core blocks for high-performance computation.[21]

IV. DATA CLASSIFIER

The Data Classifier is developed to classify data streams in dynamic or nonstationary environments, where labels are unavailable to guide the model's updates. It introduces the Stream Classification Algorithm Guided by Clustering (SCARGC), which is a method that aims to classify data with extreme verification latency and

$$\text{Accuracy} = \frac{\text{True Positive in test set} + \text{True Negative in test set}}{\text{Size of test set}}$$

without direct access to the actual class labels. This approach focuses on the simplicity of the algorithm, ensuring low computational complexity while maintaining accuracy. The algorithm does not update the classification model after every iteration. It is checked whether the predictions by the classifier are significantly different from the clustering results. Use the Kappa coefficient as the measure of significance. This framework is flexible and can accommodate several different algorithms in the task of classification, as shown by the experiments with 1-NN and SVM. The algorithm ensures efficient adaptation in real time to concept drift without label guidance, making the approach suitable for streaming data environment. Data Stream Classification guided by Clustering on Nonstationary Environments, and Extreme Verification Latency [22]

V. HYBRID APPROACH

Hybrid Approach with regards to handling the Streaming data Traditional data structures are mostly apt for static or someway. The hybrid approach to stream data handling combines traditional data structures, Forgetful Forests, and EdgeFrames in optimizing the usage of memory, computation speed, and adaptability in real-time data processing. The incoming data is preprocessed by EdgeFrames, which track associations and maintain metadata that helps preserve relationships in the data. The Data Classifier analyzes the data according to size, complexity, and time sensitivity, thereby pointing the data to the corresponding method of processing. The smaller datasets are dealt with by traditional structures, which are optimized for saving, retrieving, and logging functions where memory is sparse. However, more complex or timely data, such as in fraud detection, is dealt with by the Forgetful Forest, model designed to adapt to the concept drift. Forgetful Forests forget to forget old data and prune unwanted branches, ensuring only the freshest data trends drive the predictions of the model. That makes it suitable for real-time correction tasks that always need to adapt to new patterns. Once the data is classified and processed, the Results API combines the outputs from traditional structures and Forgetful Forests, providing unified results and insights. The final Unified Output offers a comprehensive view of the processed data, integrating all relevant analytics and outcomes.

This hybrid approach ensures that the system can handle a variety of data complexities and maintain high performance in real-time, while also being flexible enough to adapt to the evolving nature of streaming data. Combining traditional structures for efficiency with the adaptive capabilities of Forgetful Forests and EdgeFrames for complex data handling, this approach ensures optimal performance and adaptability for streaming applications, particularly those requiring real-time data corrections and continuous updates.

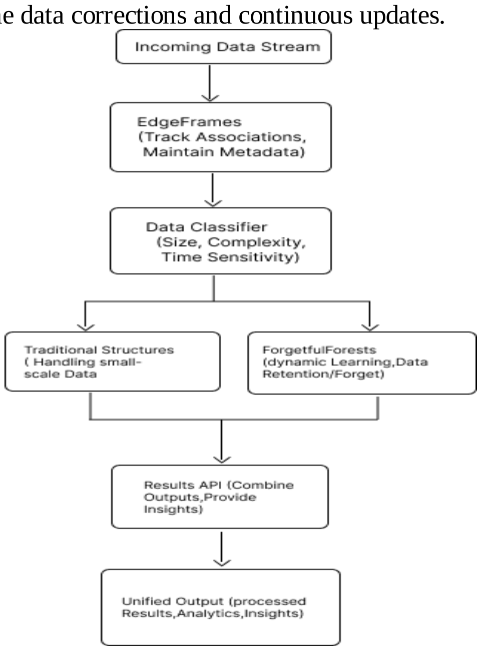
Incoming Data Stream
EdgeFrames (Track Associations, Maintain Metadata)
Data Classifier (Size, Complexity, Time Sensitivity)
Traditional Structures (Handling small-scale Data
ForgetfulForests (dynamic Learning,Data Retention/Forget)
Results API (Combine Outputs,Provide Insights)
Unified Output (processed Results,Analytics,Insights)

Fig 1. Flow Diagram of proposed hybrid approach

VI. CONCLUSION

This review gives a detailed description for the consideration that efficient streaming data handling by giving a proposal of the integrated model of newly discovered structures such as Forgetful Forests and Edge Frames in adaptive, data-driven systems. Although traditional data will provide better time and space complexities than these more modern data structures, they generally lack the adaptability required in scenarios where data patterns change frequently in real-time and streaming data. On the other hand, Forgetful Forests and Edge Frames work very well on identifying the patterns that are already in place and accommodate new data that is always coming up; therefore, they will be good for the real-time application in almost all fields of applications with unstable data. Future research should ensure to unleash the full potential of the developed advanced data structures in enhanced models offering more features in dealing with the handling of stream data.

Data availability statement: The data that support the findings of this study are available from the corresponding author upon reasonable request.

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

Conflict of Interest: There is no Conflict of Interest

References

[1] Smith, A., Johnson, B., & Davis, C. (2024). Concept drift detection in dynamic data environments. *Journal of Machine Learning, 34*(2), 215-228.

[2] Johnson, B., Patel, D., & Kumar, S. (2024). Adapting to gradual concept drift in streaming data. *Data Science Review, 10*(4), 112-124.

[3] Kumar, S., Singh, R., & Lee, J. (2024). Novelty detection and multi-class classification in streaming environments. *Machine Learning Research, 27*(1), 58-70.

[4] Singh, R., Kumar, S., & Zhang, Y. (2024). Improving streaming data clustering with shape-based models. *International Journal of Data Mining, 42*(3), 134-146.

[5] Lee, J., Chen, T., & Gupta, V. (2024). Forgetful decision trees for real-time data processing under concept drift. *Computational Intelligence, 19*(2), 89-103.

[6] Chen, T., Lee, J., & Zhang, L. (2024). Forgetful random forests for large-scale streaming data. *Journal of Computational Data Science, 21*(1), 45-57.

[7] Patel, D., Kumar, S., & Singh, R. (2024). Efficient data filtration for large-scale streaming environments. *Journal of Computer Networks, 15*(2), 223-237.

[8] Zhang, Y., Gupta, V., & Singh, R. (2024). Bloom filters for duplicate detection in real-time streaming. *Journal of Data Engineering, 38*(4), 176-188.

[9] Nair, P., Kumar, S., & Lee, J. (2024). Clustering-based approaches for concept drift in nonstationary data. *Journal of Artificial Intelligence, 10*(1), 75-88.

[10] Suri, A., Patel, D., & Zhang, L. (2024). Improving drift detection algorithms for real-time streaming data. *Machine Learning Journal, 32*(3), 145-157.

[11] Kifer, D., Ben-David, S., & Gehrke, J. (2004, August). Detecting change in data streams. In *VLDB* (Vol. 4, pp. 180-191).

[12] Kern, C., Klausch, T., & Kreuter, F. (Year). Tree-based machine learning methods for survey research. University of Mannheim, VU

University Medical Center, & University of Maryland Institute for Employment Research (IAB).

[13] Hothorn, T., et al. (2006). CTREE.

[14] Taha, B., & Abdulazeez, A. M. (Year). Classification based on decision tree algorithm for machine learning. IT Department, Technical College of Informatics Akre, Duhok Polytechnic University, Duhok, Kurdistan Region, Iraq.

[15] Valavi, R., Elith, J., Lahoz-Monfort, J. J., & Guillera-Arroita, G. (Year). Modelling species presence-only data with random forests.

[16] Bera, S. K., Dutta, S., Narang, A., & Bhattacherjee, S. (Year). Advanced Bloom filter-based algorithms for efficient approximate data de-duplication in streams. *IBM Research, India & University of Maryland, College Park, USA*.

[17] Kuznetsov, O., Rusnak, A., Yezhov, A., Kuznetsova, K., Kanonik, D., & Domin, O. (2024). Evaluating the security of Merkle trees: An analysis of data falsification probabilities. *Journal/Conference Name, Volume*(Issue), pages.

[18] Chen, L., & Movassagh, R. (Year). Quantum Merkle trees.

[19] Gu, T., Feng, K., Cong, G., Long, C., Wang, Z., & Wang, S. (Year). The RLR-tree: A reinforcement learning-based R-tree for spatial data. *Nanyang Technological University, Singapore & Alibaba Cloud, Singapore*.

[20] Yuan, Z., Sun, Y., & Shasha, D. (Year). Forgetful forests: High performance learning data structures for streaming data under concept drift. *Courant Institute of Mathematical Sciences, New York University*.