TMLAI TRANSACTIONS ON MACHINE LEARNING AND ARTIFICIAL INTELLIGENCE
SOCIETY FOR SCIENCE AND EDUCATION UNITED KINGDOM
Volume 8 No 4
ISSN 2054-7390

A Survey of Challenges Facing Streaming Data

[1]Katie Jin and [1]Sikha Bagui
[1]Department of Computer Science, University of West Florida, Pensacola, FL 32514, USA
yiranjin6@163.com; bagui@uwf.edu

ABSTRACT

This survey performs a thorough enumeration and analysis of existing methods for data stream processing. It is a survey of the challenges facing streaming data. The challenges addressed are preprocessing of streaming data, detection and dealing with concept drifts in streaming data, data reduction in the face of data streams, approximate queries and blocking operations in streaming data.

Keywords: Streaming Data, Sliding Windows, Concept Drift, Data Preprocessing, Data Reduction, Data Streams

1 Introduction

In connection-oriented communication, data streams are a sequence of digitally encoded coherent signals (packets of data) used to transmit or receive information that is in the process of being transmitted [1]. With the flooding emergence of high-speed web, social media, Internet of Things (IoT) devices, sensors and the Cloud, data streams or streaming data is becoming pervasive. The growing demand for extracting valuable information from real-time data streams and faster analytics requirements accompany the huge growth of data.

Streaming data is especially challenging to deal with because it is continuously generated by a set of devices and sources in heterogeneous formats. It is difficult to get uniformly processed real-time information from streaming data. For example, one dominating example of complex data streams is IoT. Data of IoT devices is always flowing. Traditional batch processing methods are not capable of dealing with these continuous streams of data and the plentiful different data types.

Secondly, streaming data requires infinite memory. The size of storage required for computing a precise result of data stream queries may grow without an upper bound. External and distributed storage approaches will have an impact on real-time response to continuous queries. And, the computation time of each data element needs to be short due to the ceaseless arrival of new data.

Hence, data processing needs to adapt to the changing environment. There are numbers of possible problems related to data streaming:

- Noise reduction on arriving data instances.
- Detection of concept drift.
- Data storage issues.
- Data confidentiality, integrity and availability in the distributed environment.

DOI: 10.14738/tmlai.84.8579
Publication Date: 12th July, 2020
URL: http://dx.doi.org/10.14738/tmlai.84.8579

- Model simplification and evaluation.
- Fast data processing algorithms and heuristics developments.
- Residual analysis and error bounds evaluation.
- Decision support reliability.

This survey aims to perform a thorough enumeration and analysis of existing methods for data stream processing. It is a survey of the challenges facing streaming data. The structure of this work is as follows. Section 2 discusses data stream preprocessing; section 3 discusses concept drift, a very important concept in streaming data; section 4 discusses data reduction in the face of streaming data; section 5 presents approximate query on data streams; section 6 discusses blocking operations on data streams; and section 7 concludes the paper.

2 Data Preprocessing for Streaming Data

Data preprocessing [2][3] is one of the most significant stages in the knowledge discovery process. Raw data usually has many defects, such as default values, inconsistencies, noise, and redundancy. Lower data quality will reduce the performance of a processing algorithm. Proper pre-processing steps will help in improving the quality and reliability of automated discovery and decision making. Data preprocessing [2] includes strategies such as integration, standardization, cleaning and conversion.

Reducing complexity is an essential step when dealing with data streams using traditional algorithms. Data reduction [4] includes selecting and removing redundant and noisy features and/or instances, discretizing complex continuous data, aggregating and developing data cubes [41]. At the same time, data streams may be unstable, leading to conceptual drift. That is, statistical properties of incoming data may change over time. Therefore, learning algorithms should allow online learning from new instances which can quickly change the underlying decision mechanism [5].

Despite the importance of reducing data, most papers focus on incremental algorithms that manage limited datasets. Improvements in reduction techniques for static data also have certain limitations. This is because most techniques assume that the entire training set is available, and that the properties of the data do not change over time. For example, most static instance selectors need to pass data multiple times, which will not be suitable for processing high-speed data streams [2].

Therefore, preprocessing streaming data is a main concern in the field of data mining.

3 Concept Drifts in Streaming Data

A data stream is an unbounded and ordered sequence of instances that arrive over time [6]. Data streams impose specific constraints on learning systems. Static learning schemes and stream learning schemes have different characteristics.

Cases of static data are available in advance. However, cases in data streams are not available in advance and arrive sequentially in the form of streams. Data in a stream may arrive in the form of elements or blocks, where there is a certain time interval between the arrival of the data. Furthermore, the size of a data stream accumulates over time. Therefore, the data size may be infinite, and it is impossible to store it all in memory.

Due to storage space limitations, each instance of a data stream can only be accessed a limited number of times, and then the instance has to be discarded. Also, instances of data streams must be processed

within a given amount of time in order to provide real-time responses and avoid data cache loss or data queuing. Because of the high cost of tag queries for each incoming instance, it is not possible to remodel and accurately classify each arriving instance. Therefore, for the sake of providing reliable tags, data streams take a long time to be classified or predicted.

While statistical characteristics of static data is determined by the complete data set obtained, statistical characteristics of data streams may change over time, based on the data flow instances. In many real-life situations, the nature of the data may evolve over time due to various conditions. This phenomenon, called concept drift [5][6], can be defined as the distribution and definition of changes in learning concepts over time. Drift affects the underlying properties of a class that a learning system wants to discover. As the change progresses, the concept drift reduces the correlation of the classifiers used based on the old data, even deviating from the classification or prediction of the original target. The method of dealing with drift in data streams is critical to this area of research.

3.1 Types of Concept Drift

Based on the nature and behavior of the data, there can be many types of concept drifts: True concept drifts, virtual concept drifts, rate of change based concept drift, recursive concept drift, mixed concept drift, local concept drift, overall concept drift, and feature concept drift. Below we briefly discuss each of these types of concept drift.

3.1.1 True Concept Drift versus Virtual Concept Drift

Based on the impact on the classification boundary of learning, concept drift can be divided into the two categories: true concept drift and virtual concept drift. True concept drift affects the decision boundary (posterior probability) and may affect the unconditional probability density function, posing a threat to a learning system. Virtual concept drift does not affect the decision boundary (posterior probability), but it affects the conditional probability density function and therefore does not affect the currently used learning model. However, it can still be detected.

3.1.2 Rate of change based Concept Drift

The rate of change based concept drift gives rise to three types of concept drift: (i) Sudden concept drift, the fastest concept drift; (ii) gradual concept drift, which has a medium rate of change; and (iii) incremental concept drift, which can be considered the slowest concept drift. A feature of sudden concept drift is that features are replaced by newly acquired corresponding features. Gradual conceptual drift can be viewed as a transition phase in which the proportion of new and old data in the data set available for analysis is constantly changing. The rate of change of incremental concept drift is much slower, and the change between old and new data is not obvious, and it is usually not statistically significant.

3.1.3 Recursive Concept Drift

Recursive concept drift means that a concept in a previous iteration may reappear later and may occur periodically. Outliers should be ignored because the changes represented by this value are random [7]. Noise, which represents a negligible fluctuation in the concept of environmental or other factors, should be filtered out [8].

Copyright © Society for Science and Education United Kingdom

3.1.4 Local Concept Drift versus Overall Concept Drift

[9] proposed a severity criterion that allows for the distinction between local concept drift and overall concept drift. Local concept drift means that the change only affects a small part of the feature space, while global concept drift affects the entire feature space, which makes it easier to detect than a local concept drift [10].

3.1.5 Feature Concept Drift

A "feature drift" [11] would be where the changes only affect selected attributes in the data. And, it should be noted that concept drifts can also be a combination of many of the above concept drifts.

3.1.6 Mixed Concept Drift

There may be more than one type of concept drift in the flow of a data mining process, hence mixed concept drift.

3.2 Handling Concept Drift

To handle the concept drift that can occur in a data stream, the following three solutions can be considered:

- Whenever a new instance or block arrives, it can be integrated into the training set and the classification model can be retrained.
- Retrain the classifier only if the degree of change in the instance is significant. A threshold can be set to judge the degree of change.
- Use an adaptive learning approach that can itself follow changes and drifts in the stream.

With the above three solutions, no additional control code or manual intervention would be required. Obviously, the computational cost of the first method would be high, since the system would need to calculate a new model at all time. Therefore, a focus on the other two solutions might be more practical.

3.3 Concept Drift Detectors

External tools have been developed to detect certain types of concept drifts. These tools measure various statistical characteristics of the data streams, such as standard deviation [12], prediction error [13] or instance distribution to detect concept drift. Most drift detectors use a two-stage setup.

- The detector issues a warning signal when a change begins. This warning signal is from a new classifier trained based on latest cases.
- When the current change is very serious, the old classifier is replaced with the new classifier. This solution is also known as explicit drift processing.

3.4 Sliding Windows

Sliding windows assume that a fixed-size buffer is kept for the most recent instance [14]. The stored instance is replaced based on the arrival of new data stream instances. This allows for the reduction in the number of model updates and tracks the current state of the data stream by storing the latest data in memory [15]. This can be achieved by discarding the oldest instance or by dynamically weighting the correlation of old and new data [16].

However, the size of a window has a crucial impact on performance. Smaller windows can accommodate small and fast changes, but may lose the general condition of the problem being analyzed, deviate from

the classification purpose of the model, and are prone to overfitting. Large windows can store more information efficiently, but the impact of new data on the model is greatly reduced. For tradeoffs of sliding window sizes, recent research has focused on dynamic resizing [17] or using multiple windows [18] at the same time. The ideal sliding window should be able to adapt to changes in the flow. That is, implicit drift processing.

3.5 Online learners

Online learners update instances one at a time, so they can be adjusted as soon as the flow changes. Online learners must meet the following requirements [19]:

- Each instance must be processed only once during the training process.
- The computational complexity of processing must be as small as possible.
- The accuracy should not be lower than the accuracy of the batch classifier.

A standard set of classification algorithms can work in online mode, for example, neural networks [20] or naive Bayes. However, many existing algorithms have also been modified to provide an efficient online mode of operation [21, 22]. These methods also provide implicit drift processing.

3.6 Ensemble learner

Ensemble Learner is a popular data stream mining method [23, 24]. Due to its composite structure, an ensemble learner can easily adapt to changes in a flow, thereby increasing the flexibility and predictability of a model.

The two main models here assume that the lineup of sets [25, 26, 27] change or update the basic classifier [28, 29].

- In the first model, the ensemble learner trains a new classifier to process recently arrived data (usually collected in blocks) and adds it to a collection. Pruning is used to control the size of the model and training data and to remove unrelated or older models. The weighting scheme allows the highest importance to be assigned to the latest or most frequent integration components. Some of the more complex solutions allow for an increase in the weight of the most recently performed classifier. Static classifiers are suitable for this approach because dynamic lineups can track the progress of the stream.
- Another kind of ensemble learner assumes that a fixed-size collection is kept, but each component is updated as new data becomes available. Managing the diversity of collections is critical to achieving good predictive power [30]. At the same time, the method must include a classifier that works based on incremental or online mode.

4 Data Reduction in the face of Streaming Data

Data reduction [3] is an important step in data preprocessing. In data reduction, an accurate, fast, and adaptable model is expected, which would aim at quickly responding to incoming objects and modifying the model. Therefore, dynamically reducing the complexity of the data is critical. In addition, when maintaining and updating online models, instances, training and relevance may change over time. Different forms of data reduction are discussed next.

Copyright © Society for Science and Education United Kingdom

4.1 Feature Reduction

Techniques for reducing features are widely used, including feature selection (FS), feature extraction (FE), and local retention projection [31, 32, 33]. FS [34] eliminates irrelevant or redundant features/columns, but keeping the original features makes it easier to interpret the model. Feature extraction (FE) generates a simpler feature space by transforming the original features. Based on the relationship between a selector and predictor algorithm, the FS algorithm may use one of the following three methods:

- Filter, that acts before the learning process and has nothing to do with the learning process
- Wrapper, that uses a specified learning algorithm to evaluate the feature subgroup
- Embedded, in which search is part of the learning process.

Wrapper methods tend to be more accurate than filters, but more complex. Embedded methods are cheaper than wrappers, but require a direct modification of the learning process.

4.2 Instance Selection (IS)

IS selects the most representative example to reduce the number of training instances. The instance generation (IG) [34] methods generate new instances to fill the gaps in the concept. Sampling is usually chosen randomly, while IS classifies instances based on the problem. The types of searches implemented based on the IS algorithm can be divided into three categories:

- Condensation, which eliminates outliers away from the boundary
- Edition, which eliminates noisy points near the class boundary
- Hybrid, combines noise and redundancy elimination

4.3 Feature space simplification

Feature space simplification includes normalization and discretization. Discretization [35] summarizes a set of continuous values into a finite discrete interval. Although most algorithms use continuous data, many algorithms can only handle nominal continuous features, especially those based on statistics (e.g., Naive Bayes (NB)) [36]. Tree-based classifiers [37] produce more accurate and compact results when using discrete values. A good discretizer will try to get the best predictive performance from discrete data while minimizing the number of intervals [38, 39]. Based on how the discretizers generate intervals, two main categories can be developed:

- Splitting method, which divides the most promising interval in each iteration into two partitions
- Mergence method, which combines the best two adjacent intervals in each iteration.

In streaming solutions, data reduction techniques are needed to optimize the online or batch mode to process elements as quickly as possible without any prior assumptions about data allocation.

5 Approximate Queries

When memory is limited, accurate answers cannot be provided to data flow queries. And high-quality approximate answers are often accepted instead of precise answers. Hence, approximation algorithms in the data flow problem have achieved fruitful results. Some general techniques for data reduction and feed construction include: sketches, random sampling, histograms, and wavelets.

5.1 Queries referencing past data

In a data flow scenario, once a data element passes, the model cannot revisit it. This means that queries issued after some data has been discarded may not be answered accurately. This problem can be solved

by specifying that temporary queries only allow future data to be referenced. Any past stream elements will be ignored. In addition, a method of processing a temporary query referencing past data may be employed to maintain a summary of the data stream. These summaries can be used to provide rough answers to future ad hoc queries, which can be used for forecasting. Adopting this approach requires pre-determining the best way to use memory resources to give a good approximation of the various possible future queries.

In a data flow application, most queries are long-term continuous queries, rather than short-lived one-time queries, and the benefits that can be achieved through multi-query optimization may be much greater than those used in traditional database systems. The existence of a temporary query will turn the problem of finding the best query plan for a set of queries from an offline problem to an online one.

5.2 Sliding Windows

A sliding window does not evaluate the entire historical data of the data stream, but only the specified window with the most recent data for the data stream. The sliding window is deterministic, so there is no random selection but poses a risk of poor approximation.

Sliding windows can also emphasize the role of the latest data, which usually has more relevant practical significance than older data. For example, when trying to monitor network traffic patterns in real time, website transaction records, or smart home sensor data, more and useful information is usually available based on recent observations.

Processing continuous queries over sliding windows introduces two issues that affect the design of windowed algorithms: re-execution strategies and tuple invalidation procedures [40]. [40] proposed and analyzed the incremental, multipath algorithm for sliding windows on data streams. Based on the unit time cost model, a joint order heuristic method was developed to find a good join order without traversing the entire search space.

5.3 Batch Processing, Sampling, and Synopses

Abandoning the processing of each data element that arrives yields an approximate answer. Some types of sampling or batching processing technique can be used to speed up query execution. When dealing with continuous queries, the best case is where both operations are fast relative to the speed of arrival of elements in the data stream. However, if even one or two data structures are slow, it is not possible to produce an accurate answer to continuous updates.

5.3.1 Batch Processing

In batch processing, newly arrived data elements are buffered and periodically used to provide responses to queries. Since responses are not completely real-time, the responses can be considered approximate. That is, responses represent exact responses at a certain point in time, not the exact response at the current moment. Approximating batch responses does not cause uncertainty, but sacrifices timeliness. Batch processing is also useful when algorithms cannot be consistent at peak data stream rates. At this time, batch processing will be able to buffer data at a higher flow rate and catch up in a slow cycle, thus load balancing.

Copyright © Society for Science and Education United Kingdom

5.3.2 Sampling

Since the size of the data is unlimited, and the data may arrive faster than the processing speed, and it may be unrealistic to use all the data when determining an output. Therefore, application queries may evaluate the data stream samples, not the entire data stream. A confidence range can be used for the degree of error that can be introduced or accepted by a sampling process. Unfortunately, in some cases, for example, in most queries involving joins, the sample-based approach will not give a reliable approximation guarantee.

5.3.3 Synopsis data structures

For data flow query classes that do not exist with the exact data structure of the required attributes, an approximate data structure can usually be designed. This structure will preserve the sketch or outline of the data, not the exact representation. Therefore, the amount of calculation of each data stream can be minimized. Data reduction by substituting batch data or sampling with a summary data structure is a fruitful area of research which is particularly relevant to data flow calculation models.

6 Blocking operations

A blocking query operator is a query operator that is unable to produce the first tuple of its output until it has seen its entire input. Since continuous data streams may be infinite, a blocking operator that has a data stream as one of its inputs will never see its entire input, and therefore it will never be able to produce any output. Hence, blocking operators are not very suitable for data stream computational models. But entirely doing away with blocking operators would be problematic, hence dealing with them effectively is one of the more challenging aspects of data stream computations.

Another problem is that the results produced by blocking operators may continue to change over time until all the data has been seen. Oftentimes reliable decisions cannot be made based on an intermediary stage of query execution.

7 Conclusion

This paper has presented a thorough survey of the challenges facing data stream mining. It is a thorough enumeration and analysis of existing methods for data stream processing. The basic concepts are analyzed, existing works are presented, and present challenges facing mining data streams are presented and discussed. The concepts of concept drift, data reduction, approximate query and blocking operations on data streams are presented and discussed.

Acknowledgements: This work has been partially funded by the Askew Institute of the University of West Florida.

REFERENCES

[1]. "Federal Standard 1037C *data stream*" April 4, 2007.

[2]. García, S., Luengo, J., & Herrera, F. (2015). Data preprocessing in data mining. New York: Springer.

[3]. García, S., Luengo, J., & Herrera, F. (2016). Tutorial on practical tips of the most influential data preprocessing algorithms in data mining. *Knowledge-Based Systems*, *98*, 1-29.

[4]. Ramírez-Gallego, S., Krawczyk, B., García, S., Woźniak, M., & Herrera, F. (2017). A survey on data preprocessing for data stream mining: Current status and future directions. *Neurocomputing*, *239*, 39-57.

[5]. Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM computing surveys (CSUR)*, *46*(4), 44.

[6]. Gaber, M. M. (2012). Advances in data stream mining. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *2*(1), 79-85.

[7]. Kuncheva, L. I. (2008, July). Classifier ensembles for detecting concept change in streaming data: Overview and perspectives. *2nd Workshop SUEMA*, 5-10.

[8]. Brzezinski, D., & Stefanowski, J. (2014). Combining block-based and online methods in learning ensembles from concept drifting data streams. *Information Sciences*, *265*, 50-67.

[9]. Minku, L. L., White, A. P., & Yao, X. (2009). The impact of diversity on online ensemble learning in the presence of concept drift. *IEEE Transactions on knowledge and Data Engineering*, *22*(5), 730-742.

[10]. Khamassi, I., Sayed-Mouchaweh, M., Hammami, M., & Ghédira, K. (2015). Self-adaptive windowing approach for handling complex concept drift. *Cognitive Computation*, *7*(6), 772-790.

[11]. Barddal, J. P., Gomes, H. M., Enembreck, F., Pfahringer, B., & Bifet, A. (2016, September). On dynamic feature weighting for feature drifting data streams. In *Joint european conference on machine learning and knowledge discovery in databases* (pp. 129-144). Springer, Cham.

[12]. Gama, J., Medas, P., Castillo, G., & Rodrigues, P. (2004, September). Learning with drift detection. *Brazilian symposium on artificial* i*ntelligence*, Springer, Berlin, Heidelberg, 286-295.

[13]. Bifet, A., & Gavalda, R. (2007, April). Learning from time-changing data with adaptive windowing. In *Proceedings of the 2007 SIAM international conference on data mining*, 443-448. Society for Industrial and Applied Mathematics.

[14]. Hulten, G., Spencer, L., & Domingos, P. (2001, August). Mining time-changing data streams. *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, 97-106.

[15]. Shan, J., Luo, J., Ni, G., Wu, Z., & Duan, W. (2016). CVS: fast cardinality estimation for large-scale data streams over sliding windows. *Neurocomputing*, *194*, 107-116.

[16]. Krawczyk, B., & Woźniak, M. (2015). One-class classifiers with incremental learning and forgetting for data streams with concept drift. *Soft Computing*, *19*(12), 3387-3400.

[17]. Du, L., Song, Q., & Jia, X. (2014). Detecting concept drift: an information entropy based method using an adaptive sliding window. *Intelligent Data Analysis*, *18*(3), 337-364.

[18]. Mimran, O., & Even, A. (2014). Data stream mining with multiple sliding windows for continuous prediction.

[19]. Domingos, P., & Hulten, G. (2000, August). Mining high-speed data streams. *Kdd* (2), 4.

[20]. Liu, W., Wang, Z., Liu, X., Zeng, N., Liu, Y., & Alsaadi, F. E. (2017). A survey of deep neural network architectures and their applications. *Neurocomputing*, *234*, 11-26.

Copyright © Society for Science and Education United Kingdom

[21]. Czarnecki, W. M., & Tabor, J. (2016). Online extreme entropy machines for streams classification and active learning. *Proceedings of the 9th International Conference on Computer Recognition Systems CORES 2015* (pp. 371-381). Springer, Cham.

[22]. Lakshminarayanan, B., Roy, D. M., & Teh, Y. W. (2014). Mondrian forests: Efficient online random forests. *Advances in neural information processing systems*, 3140-3148.

[23]. Woźniak, M. (2013, September). Application of combined classifiers to data stream classification. In *IFIP International Conference on Computer Information Systems and Industrial Management* (pp. 13-23). Springer, Berlin, Heidelberg.

[24]. Woźniak, M., Graña, M., & Corchado, E. (2014). A survey of multiple classifier systems as hybrid systems. *Information Fusion*, *16*, 3-17.

[25]. Elwell, R., & Polikar, R. (2011). Incremental learning of concept drift in nonstationary environments. *IEEE Transactions on Neural Networks*, *22*(10), 1517-1531.

[26]. Sun, Y., Tang, K., Minku, L. L., Wang, S., & Yao, X. (2016). Online ensemble learning of data streams with gradually evolved classes. *IEEE Transactions on Knowledge and Data Engineering*, *28*(6), 1532-1545.

[27]. Song, G., Ye, Y., Zhang, H., Xu, X., Lau, R. Y., & Liu, F. (2016). Dynamic clustering forest: an ensemble framework to efficiently classify textual data stream with concept drift. *Information Sciences*, *357*, 125-143.

[28]. Canzian, L., Zhang, Y., & van der Schaar, M. (2015). Ensemble of distributed learners for online classification of dynamic data streams. *IEEE Transactions on Signal and Information Processing over Networks*, *1*(3), 180-194.

[29]. Minku, L. L., & Yao, X. (2011). DDD: A new ensemble approach for dealing with concept drift. *IEEE Transactions on Knowledge and Data Engineering*, *24*(4), 619-633.

[30]. Minku, L. L., White, A. P., & Yao, X. (2009). The impact of diversity on online ensemble learning in the presence of concept drift. *IEEE Transactions on Knowledge and Data Engineering*, *22*(5), 730-742.

[31]. Shikkenawis, G., & Mitra, S. K. (2015). 2d orthogonal locality preserving projection for image denoising. *IEEE Transactions on Image Processing*, 25(1), 262-273.

[32]. Al-Shiha, A. A. M., Woo, W. L., & Dlay, S. S. (2014). Multi-linear neighborhood preserving projection for face recognition. *Pattern Recognition*, *47*(2), 544-555.

[33]. Zhang, H., Wu, Q. J., Chow, T. W., & Zhao, M. (2012). A two-dimensional neighborhood preserving projection for appearance-based face recognition. *Pattern Recognition*, *45*(5), 1866-1876.

[34]. Doquire, G., & Verleysen, M. (2012). Feature selection with missing data using mutual information estimators. *Neurocomputing*, *90*, 3-11.

[35]. Ferreira, A. J., & Figueiredo, M. A. (2014). Incremental filter and wrapper approaches for feature discretization. *Neurocomputing*, *123*, 60-74.

[36]. Yang, Y., & Webb, G. I. (2009). Discretization for naive-Bayes learning: managing discretization bias and variance. *Machine learning*, *74*(1), 39-74.

[37]. Hu, H. W., Chen, Y. L., & Tang, K. (2009). A dynamic discretization approach for constructing decision trees with a continuous label. *IEEE Transactions on Knowledge and Data Engineering*, *21*(11), 1505-1514.

[38]. Cano, A., Nguyen, D. T., Ventura, S., & Cios, K. J. (2016). ur-CAIM: improved CAIM discretization for unbalanced and balanced data. *Soft Computing*, *20*(1), 173-188.

[39]. Cano, A., Luna, J. M., Gibaja, E. L., & Ventura, S. (2016). LAIM discretization for multi-label data. *Information Sciences*, *330*, 370-384.

[40]. Golab, L., & Özsu, M. T. (2003). Processing sliding window multi-joins in continuous queries over data streams. *Proceedings of the 29th international conference on Very large data bases-Volume 29,* 500-511.

[41]. Han, J., Kamber, M. & Pei, J. (2012). *Data Mining: Concepts and Techniques,* 3rd Edition, Waltham: Elsevier.

Copyright © Society for Science and Education United Kingdom