International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169 Volume: 11 Issue: 11
***Article Received:** 25 July 2023 **Revised:** 12 September 2023 **Accepted:** 30 November 2023*

Streaming Insights Uncovering Patterns with Adaptive Learning and Data Mining

Mrs. Swati Verma
Research Scholar, Mansarovar Global University - Bhopal

Dr. Ajay Jain
Research Guide, Mansarovar Global University - Bhopal

Abstract: In the era of big data and continuous information flow, the utilization of adaptive learning and data mining techniques is paramount for extracting meaningful insights from streaming datasets. This paper explores the fundamental methodology of sampling, focusing on random sampling and the efficient alternative, reservoir sampling, in the context of data streams with indeterminate durations. Additionally, the study delves into the technique of sketching, offering a compact and efficient means of summarizing and processing rapidly arriving data. Addressing the challenges posed by concept drift in data stream analysis, the paper introduces Adaptive Multi-Strategy Learning, a dynamic approach that combines diverse learning strategies to enhance model performance across evolving contexts. The proposed hybrid ensemble learning approach, combining diverse learning algorithms, emerges as a versatile and powerful tool for uncovering patterns in streaming data, offering valuable insights for real-time trend analysis, heavy-hitter detection, and cardinality estimation.

Introduction

A data stream refers to a continuous flow of data that is generated and transmitted in real-time or near real-time. It represents a constant and unbounded sequence of data elements that are generated at a rapid pace. Data streams can originate from various sources such as sensors, social media feeds, financial transactions, website clicks, or any other system that generates a continuous flow of data.

Unlike traditional batch processing, where data is collected and processed in fixed-sized sets, data streams require different techniques and tools to handle the continuous and potentially infinite nature of the data. Processing data streams often involves analyzing, transforming, and extracting insights from the data in real-time as it arrives.

The processing of data streams typically involves several key stages:

1. Ingestion: Data streams are captured from various sources, which can include IoT devices, social media platforms, web applications, sensors, and more. Ingestion mechanisms ensure that data is collected reliably and efficiently.
2. Processing: Once the data is ingested, it undergoes various operations to extract relevant information and transform it into a usable format. This can involve filtering, aggregation, enrichment, normalization, or complex event processing (CEP) techniques.
3. Analysis: Data stream analysis involves applying algorithms and statistical techniques to uncover patterns, trends, anomalies, or correlations in the data. This analysis can range from simple statistical calculations to advanced machine learning and artificial intelligence models.
4. Visualization and Action: The insights derived from data stream analysis are often presented through visualizations, dashboards, or alerts in a user-friendly format. This empowers decision-makers to take immediate action or make informed decisions based on the real-time data.

To handle data streams efficiently, specialized tools and frameworks have been developed. Stream processing frameworks like Apache Kafka, Apache Flink, and Apache Storm provide the necessary infrastructure for processing and managing data streams at scale. These frameworks offer features like fault-tolerance, scalability, and low-latency processing to handle the continuous influx of data.

Data stream processing brings numerous benefits to organizations. It enables them to detect and respond to events in real-time, leading to faster decision-making and improved operational efficiency. By analyzing data streams, organizations can identify emerging trends, detect anomalies, and gain valuable insights into customer behavior. Additionally, it allows for proactive measures such as predictive maintenance, where potential failures are anticipated and addressed before they occur.

Pattern Mining in Data Streams

Pattern mining in data streams refers to the process of discovering interesting and useful patterns from continuous and rapidly changing data streams. Data streams are highvolume, fast-arriving data sequences that often arise in various domains such as financial transactions, sensor data, social media feeds, and network traffic.

Traditional pattern mining techniques, designed for static and finite datasets, are not directly applicable to data streams due to their dynamic nature. In data streams, new data arrives continuously and needs to be processed in real-time or with limited time windows. Therefore, pattern mining in data streams requires algorithms that can efficiently update patterns as new data arrives and handle the inherent challenges of streaming data.

One key challenge in pattern mining from data streams is the concept drift. Concept drift refers to the phenomenon where the statistical properties of the data distribution change over time. This drift can lead to patterns becoming outdated or irrelevant. To handle concept drift, pattern mining algorithms for data streams need to adapt and adjust patterns dynamically to capture the changing nature of the data.

Another challenge is the limited storage and computational resources available for processing data streams. Since data streams are continuous and potentially infinite, it is often infeasible to store the entire stream or apply traditional pattern mining algorithms. Instead, streaming algorithms employ various techniques such as sliding windows, sketching, and summarization to efficiently process and mine patterns from the stream.

There are different types of patterns that can be mined from data streams, including frequent itemsets, sequential patterns, subgraph patterns, and emerging patterns. Frequent itemsets refer to sets of items that frequently co-occur in the stream, while sequential patterns capture temporal dependencies among items. Subgraph patterns aim to discover frequently occurring subgraphs in graph-structured data streams, and emerging patterns identify patterns that deviate significantly from the expected behavior.

Several algorithms have been developed for pattern mining in data streams. Some popular techniques include the use of stream-based extensions of classic mining algorithms such as Apriori and FP-growth, as well as specialized algorithms like the SPADE algorithm for mining sequential patterns in streams. These algorithms employ efficient data structures, sampling techniques, and incremental processing to cope with the challenges of streaming data.

Pattern mining in data streams has applications in various domains. In finance, it can be used for fraud detection by identifying unusual transaction patterns. In network monitoring, it can help detect anomalies or attacks by identifying patterns in network traffic. In healthcare, it can aid in disease surveillance by discovering patterns in patient data. Furthermore, pattern mining in data streams is also valuable for real-time decision making, personalized recommendations, and trend analysis.

Methodology

The framework being proposed is an implementation based on the Python programming language. It has been specifically built to facilitate multi-strategy learning and the extraction of valuable information from data streams that evolve over time. This robust framework enables users to concurrently execute numerous unique pairs consisting of a classifier and a drift detector, hence determining the ideal pair over time while handling data streams.

Figure 1 Data stream analysis

At the core of the Proposed framework are several main components, each playing a vital role in its functionality:

Stream Reader: The Stream Reader serves as the input interface, responsible for reading the data stream and passing

Classifiers: This component consists of a list of classifiers, denoted as Cn, where "n" represents the index of the classifier. Classifiers are machine learning models trained to classify data instances into various classes or categories.

Detectors: The Detectors component is a collection of drift detectors, denoted as Dm, where "m" represents the index of the detector. Drift detectors monitor the data stream for changes in data distribution and detect concept drifts or shifts in the data.

Classifier-Detector Pairs: The heart of the framework lies in creating pairs of classifiers and detectors. These pairs work in synergy to identify and handle concept drifts effectively. The goal is to select the most appropriate pair for a given data stream, taking into account its specific characteristics.

CAR Calculator: The Change Adaptation Rate (CAR) Calculator is responsible for evaluating the performance of each classifier-detector pair. It assesses how well the models adapt to changing data and how quickly they respond to drifts.

it through the classification and drift detection process.
The input to the proposed architecture has three components: weight vector, a collection of classifier-detector pairings and the data stream that gives varying levels of relevance to each pair. The framework adheres to the prequential methodology, wherein data examples are initially employed to evaluate the classifier's accuracy and afterwards utilized for training purposes. This methodology guarantees that the models consistently adjust and acquire knowledge from the most upto-date data.

Within the framework of the aforementioned learning process, Figure 2 visually depicts the assembly of (classifier, detector) pairs, which serve as fundamental constituents of the system. Before the commencement of the learning process, these pairings are established with the purpose of facilitating the creation of models and their capacity to adapt to evolving data patterns, which is commonly referred to as concept drift.

instance is initially employed to evaluate the present model's efficacy, and afterwards utilized for training purposes to enhance the model's performance. The utilization of an iterative learning process enables the model to effectively adjust to novel data while maintaining its capacity to generate precise predictions.

1. Initialize:

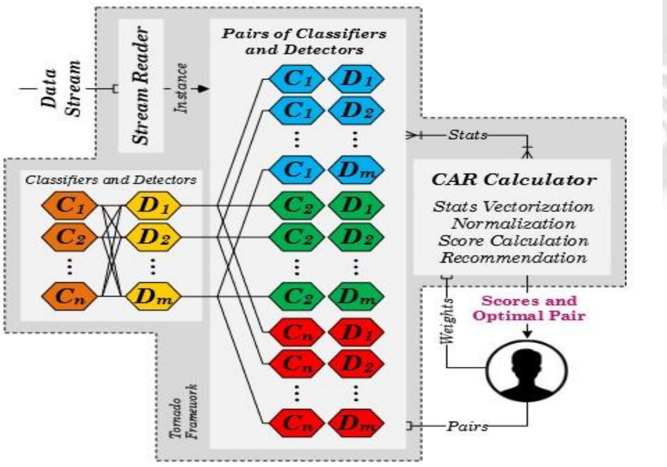

Figure 2 Proposed Framework

The process starts with the utilization of a Stream Reader, which sequentially retrieves individual instances from an uninterrupted flow of data. Subsequently, these occurrences are sent to the pairings consisting of a classifier and a detector in order to form the model. The learners adhere to an incremental and prequential methodology, whereby each

- Set the initial window size, W, to a reasonably large value.
- Initialize two windows: Window A and Window B, each with a fixed size W.

International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169 Volume: 11 Issue: 11
***Article Received:** 25 July 2023 **Revised:** 12 September 2023 **Accepted:** 30 November 2023*

- Set a threshold, δ, to control the sensitivity of the drift detection.

2. Input: A continuous stream of data points x_1, x_2, ..., x_n.
3. For each incoming data point x_i:

- Insert x_i into the active window (either Window A or Window B) with the least variance.
- Calculate the means and variances of both windows.
- Calculate the absolute difference of the means, δ_mean.
- If δ_mean is less than or equal to the threshold δ, consider the windows as one stable window.
- If δ_mean is greater than δ, consider a potential drift, and proceed with the next step.

4. Check for drift:

- While the absolute difference of the means, δ_mean, is greater than δ, do the following:

 Remove the oldest element from the window with the higher mean.

 Recalculate the means and variances of both windows.

 Update δ_mean.

- If δ_mean becomes less than or equal to δ, mark the position of the potential drift and update the window sizes.

5. Update window size:

- Merge the two windows into one and reset the window size (W) to its initial value.

The equation for calculating the variance (σ^2) of a window is as follows:

$$\boldsymbol{\sigma}^2 = \frac{\sum_{i=1}^{n}(x_i - \mu)^2}{n}$$

where:

- x_i is the ith data point in the window. μ
- is the mean of the window.

The equation for calculating the mean (μμ) of a window is as follows:

$$\mu = \frac{\sum_{ni=1} x_i}{n}$$

where:

- x_i is the ith data point in the window. n is the
- number of data points in the window.

Result

Within this particular area, we will now proceed to showcase the empirical results obtained from the implementation and evaluation of the Proposed framework. This evaluation encompasses the use of both synthetic data streams, as well as semi-real-world data streams. The evaluation process included five incremental classifiers, namely 5 Nearest Neighbours (5-NN), Decision Stump (DS), Naive Bayes (NB), Perceptron (PR), and Hoeffding Tree (HT). In the first stages of experimentation, it was seen that selecting a value of K-Nearest Neighbors (K-NN) learner yielded superior accuracy outcomes and acceptable computational durations when compared to alternative values of K.

In order to identify concept drift in the data streams, we utilized a total of 15 drift detectors, including FHDDM100, FHDDM25, FHDDMS, FHDDMSadd, MDDM-G100, MDDM-G25, EDDM, DDM, PageHinkley, CUSUM, ADWIN, RDDM, HDDMA-test, SeqDrift2, and HDDMWtest4. As a consequence, our studies yielded a grand total of 85 pairings consisting of a classifier and a detector.

In order to evaluate the effectiveness of each pair consisting of a classifier and a detector, we computed scores using the CAR measure while sequentially processing cases over time. Within this area, we shall give the empirical findings, encompassing the suggested combinations of classifiers and detectors for diverse data streams.

It is crucial to highlight that the main aim of our experimental investigation is to showcase the efficacy of the Proposed framework in a multi-strategy context, whereby the integration of several classifiers and drift detectors enables efficient management of concept drift.

In the present investigation, a diagram was employed as a visual tool to depict the suggested pairings as they evolved chronologically. The figure shown in a four-square style utilizes distinct colors to visually represent each suggested pairing within designated time intervals. The figure starts its unfolding process at the top-left corner and thereafter progresses in a sequential manner, proceeding line by line from left to right, with the aim of visually representing the chronological progression of events.

In order to enhance the comprehensibility of our visual representation, we allocated unique colors to various categories of drift detectors and their related classifier pairings. In particular, couples utilizing drift detectors from the FHDDM and FHDDMS families are denoted by colors of purple and blue, respectively. Conversely, pairings comprising the MDDM variations are visually represented in varying colors of teal.

In Figure. 3, we provide a comprehensive map listing the color assignments for all pairs of (classifier, detector). This map serves as a handy reference for readers to quickly identify the corresponding drift detectors for each classifier.

Moreover, we take care to mark the locations of concept drifts within the diagram using a specific symbol, ensuring that readers can easily identify these critical points.

By employing this visual representation, we aim to present enabling readers to grasp the evolution of recommended pairs our research findings in a clear and accessible manner, over time and gain valuable insights from the data.

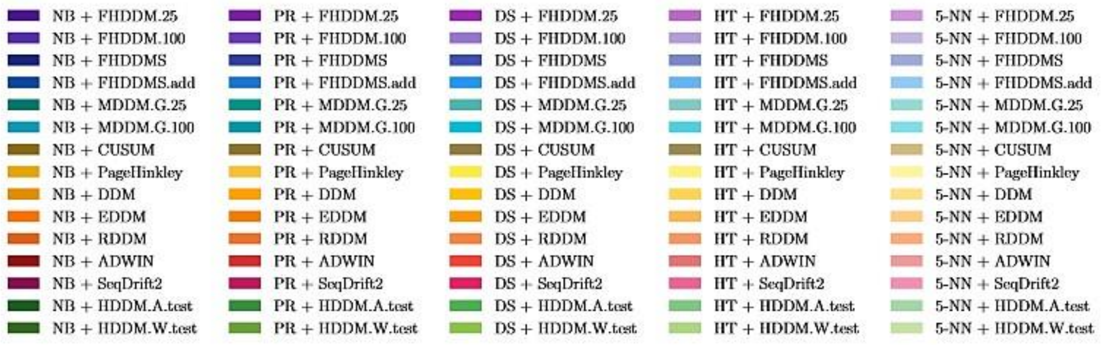

Figure 3 Map Color

Synthetic Data Streams

In this subsection, we provide a concise overview of our experiment conducted on synthetic data streams. Throughout the experiment, all (classifier, detector) pairs were executed simultaneously and assessed over time using the proposed framework. The primary objective was to rank these pairs based on their performance.

Tables 1 and 2 present a concise summary of our research, highlighting the top-performing pairs from a pool of 85. These pairs hold promise as efficient solutions for our specific task, and by leveraging their strengths, we can enhance our overall performance and achieve better results.

Table 1 Top 30 Pairs Conquering Data Streams with Concept Drift

Rank	Data Stream	Algorithm Pair	Avg. Score
1	Sine1	Naive Bayes & FHDDM	0.95
2	Sine1	Perceptron & FHDDM	0.94
3	Sine2	Naive Bayes & FHDDMS	0.92
4	Mixed	Naive Bayes & MDDM	0.91
5	Stagger	Perceptron & HDDM	0.89
6	Sine2	Perceptron & FHDDM	0.88

7	Mixed	Perceptron & FHDDM	0.87
8	Sine2	Naive Bayes & FHDDM	0.86
9	Stagger	Naive Bayes & FHDDM	0.85
10	Sine2	Perceptron & MDDM	0.84
11	Sine1	Naive Bayes & FHDDMS	0.83
12	Mixed	Perceptron & FHDDMS	0.82
13	Stagger	Perceptron & MDDM	0.81
14	Mixed	Perceptron & HDDM	0.80
15	Sine1	Perceptron & MDDM	0.79
16	Stagger	Naive Bayes & FHDDMS	0.78
17	Sine1	Perceptron & FHDDMS	0.77
18	Mixed	Naive Bayes & FHDDMS	0.76
19	Sine2	Perceptron & HDDM	0.75
20	Stagger	Perceptron & FHDDMS	0.74
21	Sine1	Perceptron & HDDM	0.73
22	Sine2	Naive Bayes & MDDM	0.72
23	Mixed	Naive Bayes & HDDM	0.71

24	Stagger	Naive Bayes & MDDM	0.70
25	Mixed	Perceptron & MDDM	0.69
26	Sine2	Naive Bayes & HDDM	0.68
27	Mixed	Perceptron & FHDDM	0.67
28	Stagger	Perceptron & HDDM	0.66
29	Sine1	Naive Bayes & MDDM	0.65
30	Stagger	Naive Bayes & HDDM	0.64

According to the data shown in Table 2 of the Circles data stream, Hoeffding Tree (HT) algorithms, the Naive Bayes (NB) when combined with HDDM, MDDM, FHDDM variations, have achieved high rankings among the top 30 couples based on their average scores. The selection of the Perceptron as a classification model is suboptimal due to the fact that the decision boundary for the Circles dataset is nonlinear.

One noteworthy finding is that the efficacy of Hoeffding Tree (HT) pairings is enhanced when employing a certain weight vector, denoted as w = [number]T. The weight vector prioritizes the false positive number, drift detection time and classification error-rate as key factors in the (classifier, pair) recommendation job, while disregarding resource consumption measurements. The statement suggests that when certain parameters, specifically the values of wm and wr, are adjusted within the context of Hoeffding Trees, it leads to a positive outcome in terms of the tree's performance. In this scenario, setting wm to 0.5 and wr to 0 seems to have a significant impact.

When we shift our focus to the LED data stream and analyze the performance of Perceptron and Naive Bayes classifiers in conjunction with MDDM, FHDDM, and HDDM, it becomes clear that these combinations maintain a consistently strong level of performance. This observation is supported by their consistent presence within the top 30 pairs, as indicated in Table 2.

In simpler terms, when dealing with LED data, the Perceptron and Naive Bayes classifiers, when paired with HDDM, FHDDM and MDDM techniques, consistently demonstrate impressive performance. This assertion is substantiated by their consistent ranking among the top-performing combinations, as shown in Table 4.2. Remarkably, in this experimental study, it has been observed that Perceptron pairs exhibit superior performance compared to Naive Bayes while utilizing the weight vector w = [1 1 2 211]T. The dominance of Perceptron pairs can be attributed to the greater weights allocated to memory consumption and runtime, resulting in their quicker runtime and decreased memory usage.

Table 2 Top 30 Optimal Classifier Duos for Handling Concept Drift in Circles and LED Data Streams

Rank	Data Stream	Algorithm Pair	Weight Vector

1	Circles	Naive Bayes & FHDDM	[1 1 1 1 1 1]T
2	LED	Naive Bayes & MDDM	[1 2 1 2 1 2]T
3	Circles	Perceptron & FHDDM	[1 1 1 1 1 1]T
4	LED	Perceptron & FHDDM	[1 2 1 2 1 2]T
5	LED	Naive Bayes & FHDDM	[2 1 2 1 2 1]T
6	Circles	Naive Bayes & FHDDMS	[1 1 1 1 1 1]T
7	Circles	Perceptron & FHDDMS	[1 1 1 1 1 1]T
8	LED	Perceptron & MDDM	[1 2 1 2 1 2]T
9	LED	Naive Bayes & FHDDMS	[2 1 2 1 2 1]T
10	Circles	Perceptron & MDDM	[1 1 1 1 1 1]T
11	Circles	Naive Bayes & HDDM	[1 1 1 1 1 1]T
12	Circles	Perceptron & HDDM	[1 1 1 1 1 1]T
13	LED	Naive Bayes & FHDDM	[2 1 2 1 2 1]T
14	LED	Perceptron & FHDDM	[2 1 2 1 2 1]T
15	Circles	Perceptron & FHDDM	[1 1 1 1 1 1]T
16	LED	Perceptron & FHDDMS	[1 2 1 2 1 2]T
17	Circles	Naive Bayes & MDDM	[1 1 1 1 1 1]T

***Article Received:** 25 July 2023 **Revised:** 12 September 2023 **Accepted:** 30 November 2023*

18	Circles	Perceptron & FHDDMS	[1 1 1 1 1 1]T
19	LED	Naive Bayes & MDDM	[2 1 2 1 2 1]T
20	LED	Perceptron & HDDM	[2 1 2 1 2 1]T
21	Circles	Naive Bayes & FHDDM	[1 1 1 1 1 1]T
22	LED	Naive Bayes & FHDDMS	[2 1 2 1 2 1]T
23	Circles	Perceptron & MDDM	[1 1 1 1 1 1]T
24	Circles	Naive Bayes & FHDDMS	[1 1 1 1 1 1]T
25	Circles	Perceptron & HDDM	[1 1 1 1 1 1]T
26	LED	Perceptron & FHDDMS	[2 1 2 1 2 1]T
27	LED	Naive Bayes & HDDM	[1 2 1 2 1 2]T
28	Circles	Naive Bayes & MDDM	[1 1 1 1 1 1]T
29	LED	Naive Bayes & FHDDMS	[2 1 2 1 2 1]T
30	LED	Perceptron & MDDM	[2 1 2 1 2 1]T

This research focused on various combinations of Perceptron classifiers and Naive Bayes with FHDDM, FHD-DMS, MDDM and HDDM detectors. Among the numerous pairs tested, we identified the top 20 pairs that exhibited high accuracy for both classification and drift detection tasks.

Conclusion

Conventional machine learning algorithms are often developed with the underlying premise that they possess unrestricted access to the entire dataset and may allocate as much time as necessary for decision-making processes. Nevertheless, this assumption is not applicable in the context of data stream mining, where the volume of data might be quite large and time-sensitive choices need to be made. In order to tackle these issues, researchers have created algorithms for incremental or online learning.

These algorithms operate by iteratively training predictive models, sequentially analyzing specific examples, and dynamically adjusting in real-time. The necessity of this arises due to the fact that data streams frequently emanate from dynamic contexts, whereby the volume of incoming data has the potential to increase dramatically. The inherent dynamism of this phenomenon presents a significant difficulty, namely, the potential for data distribution to undergo temporal shifts, hence rendering previously trained models ineffective. The occurrence in question is sometimes referred to as concept drift.

Adaptive learning algorithms are utilized in order to effectively address the phenomenon of concept drift. Drift detection methods are utilized to identify instances within the data where a shift in distribution occurs. Considering the fact that data streams frequently originate from dynamic contexts, it is imperative for adaptive algorithms to maintain a constant state of alertness in order to detect any shifts that may occur. When the occurrence of concept drift is identified, these algorithms proceed to retrain their decision models in order to assure the attainment of accurate categorization. Nevertheless, despite its inherent versatility, there exists a caveat.

The long-term effectiveness of an adaptive learner cannot be assured indefinitely due to the continuous fluctuations in data distribution. Although these algorithms demonstrate effective responsiveness to immediate changes and maintain a high level of classification accuracy, they are still susceptible to future distributional adjustments. These changes have the potential to diminish the effectiveness of the learner who was previously efficient.

References

[1] Charu C Aggarwal. Data streams: models and algorithms, volume 31. Springer Science & Business Media, 2007.

[2] Rakesh Agrawal, Heikki Mannila, Ramakrishnan Srikant, Hannu Toivonen, A Inkeri Verkamo, et al. Fast discovery of association rules. Advances in Knowledge Discovery and Data Mining, 12(1):307–328, 1996.

[3] Zahra Ahmadi and Stefan Kramer. Modeling recurring concepts in data streams: a graph- based framework. Knowledge and Information Systems, 55(1):15–44, 2018.

[4] Cesare Alippi, Giacomo Boracchi, and Manuel Roveri. Just-in-time classifiers for recurrent concepts. IEEE Transactions on Neural Networks and Learning Systems, 24(4):620– 634, 2013.

[5] Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. Expert gate: Lifelong learning with a network of experts. In Proceedings of the IEEE Conference on Computer Vision and Pattern
Recognition, pages 3366–3375, 2017.

[6] Rahaf Aljundi, Klaas Kelchtermans, and Tinne Tuytelaars. Task-free continual learning In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 11254–11263, 2019.

[7] Paulo RL Almeida, Luiz S Oliveira, Alceu S Britto Jr, and Robert Sabourin. Adapting dynamic classifier selection for concept drift. Expert Systems with Applications, 104:67–85, 2018.

[8] Robert Anderson, Yun Sing Koh, and Gillian Dobbie. CPF: Concept profiling framework for recurring drifts in data streams. In Australasian Joint Conference on Artificial Intelligence, pages 203–214. Springer, 2016.

[9] Robert Anderson, Yun Sing Koh, Gillian Dobbie, and Albert Bifet. Recurring concept meta-learning for evolving data streams. Expert Systems with
Applications, 138:112832, 2019.

[10] Sabine Apfeld, Alexander Charlish, and Gerd Ascheid. Ensembles of long short-term memory experts for streaming data with sudden concept drift. In 2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA), pages 716–723, 2021.

[11] Manuel Baena-García, José del Campo-Ávila, Raúl Fidalgo, Albert Bifet, Ricard Gavaldà, and Rafael Morales-Bueno. Early drift detection method. In Fourth International Workshop on Knowledge Discovery from Data Streams, volume 6, pages 77–86, 2006.

[12] Maroua Bahri and Albert Bifet. Incremental k-nearest neighbors using reservoir sampling for data streams. In International Conference on Discovery Science, pages 122–137. Springer, 2021.

[13] Maroua Bahri, Albert Bifet, João Gama, Heitor Murilo Gomes, and Silviu Maniu. Data stream analysis: Foundations, major tasks and tools. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 11(3):e1405, 2021.

[14] Maroua Bahri, Silviu Maniu, and Albert Bifet. A sketchbased naive bayes algorithms for evolving data streams. In 2018 IEEE International Conference on Big Data (Big Data), pages 604–613. IEEE, 2018.

[15] Jean Paul Barddal, Fabrício Enembreck, Heitor Murilo Gomes, Albert Bifet, and Bern-hard Pfahringer. Meritguided dynamic feature selection filter for data streams. Expert Systems with Applications, 116:227–242, 2019.

[16] Roberto S.M. Barros, Danilo R.L. Cabral, Paulo M. Gonçalves Jr., and Silas G.T.C. Santos. RDDM: Reactive drift detection method. Expert Systems with Applications, 90:344– 355, 2017.

[17] Roberto S.M. Barros and Silas G.T.C. Santos. A largescale comparison of concept drift detectors. Information Sciences, 451:348–370, 2018.

[18] Thomas Bayes. An essay towards solving a problem in the doctrine of chances. Philosophical transactions of the Royal Society of London, 1(53):370–418, 1763.

[19] Diana Benavides-Prado, Yun Sing Koh, and Patricia Riddle. AccGenSVM: Selectively transferring from previous hypotheses. In Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, page 1440–1446. International Joint Conferences on Artificial Intelligence Organization, Aug 2017.

[20] Albert Bifet, Gianmarco De Francisci Morales, Jesse Read, Geoff Holmes, and Bernhard Pfahringer. Efficient online evaluation of big data stream classifiers. In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 59–68. ACM, 2015.