IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 23, Issue 3, Ser. II (May – June 2021), PP 55-67
www.iosrjournals.org

A Comprehensive –Theoretical Review of Data Stream Mining: methods and challenges

Ms. Shailaja B. Jadhav †
Research Scholar , Shivaji University, Kolhapur , Maharashtra- India

Dr. D. V. Kodawade
Department of Computer Science and Engineering , DKTE Society's
Textile and Engineering Institute, Ichalkaranji, (MAHARASHTRA) INDIA.

Abstract
In many applications of today's information systems, Mining and analyzing data streams is inevitable in order to find the useful insights from the data streams . At present , many researchers have extensively worked on this area of research, However, there are specific problems that challenge the ongoing algorithmic solutions . Hence, this area of research is evolving and gaining extensive attention as the work progresses . The aim of this paper is to present a comprehensive spectrum of various approaches used for data streams through systematic and deep dive into the relevant literature . The Paper also discusses the advanced learning concepts, novel and adapted algorithms, summarizing major characteristics and highlighting important drawbacks. The Paper concludes with key challenges faced by algorithmic solutions, discussion of open research problems and lines of future research.

Keywords : *Data streams, stream mining ,real time big data, incremental learning*

Date of Submission: 20-06-2021 Date of Acceptance: 05-07-2021

I. Introduction:

Recently, data stream mining became a very important and challenging issue of computer science research. The reason is the enormous growth of data amounts generated in various areas of human activities. Data streams [1–3] are potentially of infinite size and often arrive at the system with very high rates. Therefore, it is not possible to store all the data in memory. Appropriate algorithms should use some synopsis structures to compress the information gathered from the past data. Moreover, data stream mining algorithms should be fast enough.

Most often they have an incremental nature, i.e. each data element is processed at most once. Alternatively, the data stream can be analyzed in a block-based manner. Another feature of data streams is that the underlying data distribution may change over time. It is known in the literature as 'concept drift' . A good data stream mining method should be able to react to different types of changes.

Data streams pose new challenges for machine learning and data mining as the traditional methods have been designed for static datasets and are not capable of efficiently analyzing fast growing amounts of data and taking into consideration characteristics such as:

- Limited computational resources as memory and time, as well as tight needs to make predictions in reasonable time.
- The phenomenon called concept drift , i.e., changes in distribution of data which occur in the stream over time. This could dramatically deteriorate performance of the used model.
- Data may come so quickly in some applications that labeling all items may be delayed or sometimes even impossible.

1.1 Data stream characteristics :

A majority of machine learning methods which have been presented in the literature worldwide are designed for static datasets. Unfortunately, they cannot be directly applied to data streams because of their specific nature. Data streams are potentially unbounded ordered sequences of data elements -which arrive over time, i.e. $S = (s_1, \ldots, s_{n,} S)$), where n(S) can potentially tend to infinity. The time intervals between the arrival of each data item may vary. It should be noted that in standard data mining, the order of data elements in a training dataset is

of no meaning .These data items can be simple attribute-value pairs like relational database tuples, or more complex structures such as graphs.
The main features of data streams include [4-5] :

- Data items in the stream appear sequentially over time.
- The processing system should be able to react at any time, as there is no control over the order in which data items arrive.
- Streams are possibly of infinite length-huge data size,usually impossible to store all the data from the data stream in memory.
- Due to the rapid rate of data arrival (relatively high with respect to the processing power of the system) usually when the item is processed it is discarded or aggregated statistics or synopses; are calculated for storage if necessary.
- Data streams are susceptible to change (data distributions generating examples may change on the fly)
- The data labeling may be very costly and can be delayed .

These data stream characteristics pose the need for algorithms, capable of handling the data stream requirements viz. constraints of memory usage, restricted processing time, and one scan of incoming examples [13] . Of course some algorithms, like instance based learning, Naive Bayes, or neural networks are naturally incremental ones and can be experimented for the above said purpose , However, simple incremental learning is typically insufficient, as it neither takes into account evolving nature of data sources , nor does it meets tight computational demands .
The main contributions of this paper are outlined as follows:
1. Critically and extensively review existing literature, including both well established and recent techniques for handling data streams.
2. Identify and discuss clear gaps in the literature areas for improvement.
3. Provide possible avenues for future research that address current research gaps in the existing literature.

1.2 Organization of Paper:

This paper is organized as follows.
Section 1 – Gives Introduction about the general idea of Data stream mining.
Section 2 - provides a general overview of stream mining, its key differences from traditional batch learning and design requirements for stream mining algorithms.
Section 3 - focuses on Preprocessing Procedures for Data Streams , providing definitions and discussing algorithmic requirements
Section 4 - categories and critically reviews existing techniques and State-of-the-Art of Data Stream Mining Methods.
Section 5 - discusses advanced issues in data stream analysis, evaluation methods, as well as new frameworks for evaluating data stream classifiers(SOFTWARE PACKAGES AND FRAMEWORKS) .
Section 6 - Offers Concluding Remarks and Challenging Problems for future research direction.

II. Stream mining overview

Data stream mining, as its name suggests, is connected with two basic fields of computer science, i.e. data mining and data streams. Data mining is an inter- disciplinary subfield of computer science whose main aim is to develop tools and methods for exploring knowledge from large datasets. Data mining is strictly related to statistics, pattern recognition and machine learning , using methods like neural networks, decision trees, Bayesian networks or support vector machines. Neural networks are often considered a method belonging to soft computing or computational intelligence . Another soft computing concept used in data mining is fuzzy logic. It should be noted that all mentioned above subfields of computer science are not strictly defined and overlap in many issues. Moreover, data mining is something more than learning or extracting knowledge, and includes, among others, database systems or data visualization as well.

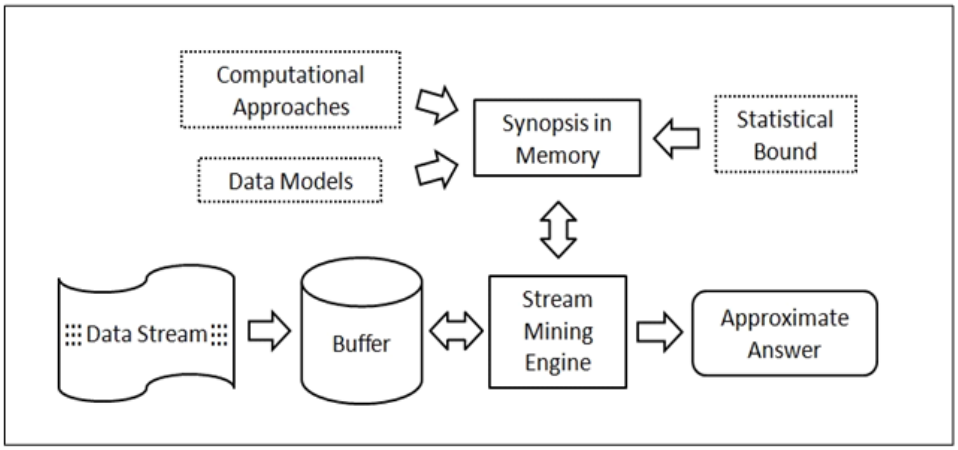

Fig. 2.1 System architecture for data stream mining

Data streams possess specific and unique characteristics that differentiate them from other forms of data. A majority of machine learning methods which have been presented in the literature worldwide are designed for static datasets. Unfortunately, they cannot be directly applied to data streams because of their specific nature. The difference between traditional methods and data streams.

1. Data elements in the stream arrive on line.
2. The system has no control over the order in which data elements arrive to be processed, either within a data stream or across data streams.
3. Data streams are potentially unbound in size.
4. Once an element from a data stream has been processed, it is discarded or archived. It cannot be retrieved easily unless it is explicitly stored in memory, which is small relative to the size of data streams.

The unique characteristics of a data stream contribute to the challenges in processing its arriving elements.
The key differences between processing traditional batch data and stream data are shown Table 1.

Table 2.1: key differences between processing traditional batch data and stream data

Traditional Batch Data Mining	**Stream Data Mining**
Data is persistent	Transient
can be queried once in its entirety	must be queried continuously
Accessed at random	can only be accessed in the sequence of their arrival
Static data distribution that does not shift over time	distribution for an evolving data stream may change over time
Batch and multi-pass learning	with one pass of the data only
	computationally fast and lightweight

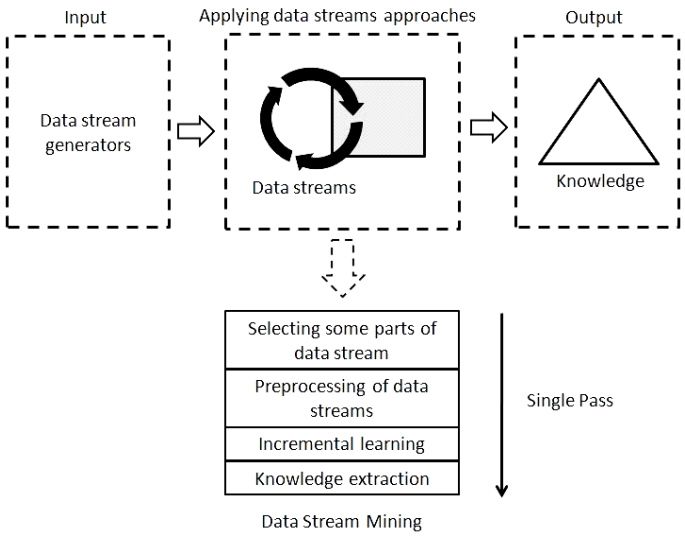

Fig 2.2 Data stream mining outline

Since batch data is persistent, it can be queried once in its entirety and individual data elements can be accessed at random.

Data streams however, since transient, must be queried continuously by the algorithm. The data elements in the stream cannot be accessed at random; they can only be accessed in the sequence in which they arrive from the stream.

(1) very large, potentially infinite number of data elements,

(2) high rate of data arrival at the system,

(3) potential changes of various kind in data distribution during data stream processing (a phenomena known in the literature under the name of concept drift).

In the light of the first feature, it is assumed that in any device used in data stream mining the amount of available memory might be not enough to store all data elements. Therefore, data stream mining algorithms should be able to store information about past data elements in a compact form, e.g. using some synopsis structures. Sufficient statistics used in the VFDT algorithm [23] are an example of such a structure.

The second feature, i.e. the high rate of a data stream, forces a data stream mining algorithm to process each data element and update current results as fast as possible. It means that appropriate algorithms should have an incremental nature. If the rate of a data stream is too high with respect to the computational power of a computing device, then some data elements have to be discarded without taking them into account at all.

A data stream can be processed by an algorithm mainly in two ways.

The first group consists of online algorithms which perform computations and update the output after each data element is read from the stream.

The second type of algorithms **partition the data stream into blocks** (chunks) of data B_l, $l = 1, \ldots, L$, of size $n(B_l)$. Subsequent blocks do not have to have the same sizes. Algorithms perform computations on block B_l and simultaneously subsequent data elements are collected in a buffer to form block B_{l+1}

The solutions and properties of data mining algorithms restrict themselves to these two features and actually develop algorithms designed for big data. However, probably the most significant difference between static and stream data lies in the third feature, i.e. the concept drift.

It means that, unlike in static datasets, probability density may change over time in the case of data streams. The probability density function of a data element depends on time t, for example in a supervised learning problem it is given by $\rho(x_1, \ldots, x_D, y, t)$. Data stream element s_i is a random variable drawn from the probability distribution described by $\rho(x_1, \ldots, xD, y, t_i)$, where ti is a time moment at which data element s i is read from the stream. The concept drift may or may not take place. However, a proper data stream mining algorithm should be able to deal with it in the case of its occurrence.

While standard data mining algorithms provide a single output using training dataset S, algorithms for data streams produce a sequence of outputs.[24]

In reality, data streams produce plenty amounts of data at a near constant rate. Examples of such streams are sensor networks ,surveillance systems,and telecommunication systems. However, with recent modern hardware developments, these streams are produced by new devices such as smart household appliances and car navigation systems. Algorithms must be able to accurately model the underlying distribution in order to learn from data streams. The ability to detect and adapt to changes in the distribution of examples is of supreme importance for data stream mining algorithms.

Thus,stream mining algorithms bring with them their own set of challenging algorithmic requirements, as stated above. Any classifier, operating with stream data must contain mechanisms to meet these requirements;otherwise their predictive performance will diminish over time. The predictive model must be capable of updating with new data as it arrives, or even replacing itself entirely if required.

III. Preprocessing Procedures for Data Streams

Data preparation is an essential part of a machine learning solution. Real-world problems require transformations to raw data, preprocessing steps and usually a further selection of 'relevant' data before it can be used to build machine learning models. Typical preprocessing, such as normalizing a feature, can be complicated in a streaming setting as statistics about the data are unknown a priori.

There are mainly two reasons for performing data preprocessing before training a machine learning model:

1. To make it easy for the learning algorithms to handle the data;

2. To enhance the learning performance by extracting or keeping only the most relevant data.

Some data stream mining algorithms contain appropriate preprocessing mechanisms embedded in their structure. However, the preprocessing is often performed externally as well.

A preprocessing technique which permits dealing with large-sized datasets is sampling.

In this technique, only a small sample of the whole dataset is selected and then used for computation. The selected sample should be as representative of the dataset as possible. The simplest method is to assign the equal probability of being chosen to each data element.
Another method is a reservoir sampling [31–33]. In this method reservoir of F data elements is created. Subsequent data elements have probability equal to F/n(S) of being selected and they replace the so-far oldest element in the reservoir.
A specific type of data sampling is a technique called load shedding [34, 35]. Load shedding is helpful in high-speed data stream processing when the computational power of a computing device is insufficient to process every data element and some of them have to be discarded. Discarding data elements may be performed either completely randomly or the knowledge about correlations between subsequent elements may be used to perform load shedding in more intelligent way [36].
In [37] the application of the Particle Swarm Optimization was proposed to perform the load shedding. Since the amount of available memory is not enough in any device processing data streams, algorithms should create synopsis structures and summaries to store information about past data. Vivid examples of such synopsis structure are wavelets. In [38] a one-pass algorithm for estimating the coefficients of Haar wavelets from data was proposed. Another group of data summaries is formed by histograms [39, 40]. Several data stream mining algorithms contain their own synopsis structures which help to store information about the past. The idea of microclusters was presented in the CluStream [41] and the HPStream [42] clustering algorithms. In the VFDT algorithm [18], the information about data elements in each decision tree node is stored in the form of so-called sufficient statistics. This three-dimensional structure stores numbers of data elements for each class, each attribute, and each possible attribute value. Hence, it requires a constant amount of memory during the processing of data stream.
Since the stream may change its concept over time, it is desired to operate mainly on the newest data elements rather than on the oldest ones. To achieve this goal structures called sliding windows are used [43, 44]. They store only a part of data elements from the nearest past and all data mining procedures are performed only on the windowed data. In [45] two kinds of windows are distinguished:
•sequence based windows
• time-stamp based windows
The former stores constant number W of the newest data elements. In the latter data elements from the last period of time of length T are collected. In [46] another specific type of sliding windows called the tilted time window is considered. In this structure, data elements from the nearest past are stored quite densely while it contains only several data elements from the distant past. In [47] a window of a variable size, i.e. the Adaptive Window (ADWIN), was proposed. If there is no drift in the stream the ADWIN collects data elements with no limit. Periodically the window is divided into two parts and for each part the mean value is calculated. If the difference between the two means is significant, according to a statistical test, then the part of the window with older elements is discarded.
In [48] the FISH and the FISH2 sliding window algorithms were presented. In these windows, not only the temporal distance but the spatial distance between stream data elements is taken into account as well. The distance between two data elements is a linear combination of temporal and spatial distances. The most recent algorithm of this kind, FISH3, was presented in[56]

Table 3.1 Data stream preprocessing techniques used

Preprocessing Technique Used	**Subprocess Details if any**
Sampling	Load Shedding - Particle Swarm Optimization
Reservoir sampling	
Synopsis structures	Haar wavelets, histograms
Microclusters	CluStream, HPStream,
Sliding windows	sequence based windows time-stamp based windows Adaptive Window (ADWIN) FISH1,FISH2,FISH3

Preprocessing, fitting and testing a model are distinct phases in a batch learning pipeline. They are applied in order, and the output of this process is a fitted model that can be used for future data.
Streaming data preprocessing is much different from that of stationary data preprocessing as it needs the continuous application of the whole pipeline while it is being used. Consequently, all these phases need to be interleaved in an on line process, through the pipeline. Ideally it does not rely on performing some tasks off line.

3.1 Feature Engineering Accurate machine learning solutions often are based on a well-thought feature transformation, selection or reduction of the raw data (Feature Engineering).

3.1.1 Summarization Sketches:

Working with limited memory in streaming data is crucial,since data streams produce enormous quantities of raw data, which are often not useful as individual instances, but essential when aggregated. The solution to this problem arises in the form of sketches .The summary created to avoid storing and maintaining a large amount of data is often referred to as a 'sketch' of the data. Sketches are probabilistic data structures that summarize streams of data, such that any two sketches of individual streams can be combined into the sketch of the combined stream in a space-efficient way [25-27] . Using sketches requires a number of compromises: sketches must be created by using a constrained amount of memory and processing time, and if the sketches are not accurate, then the information derived from them may be misleading.

3.1.2 Feature Scaling :

Feature scaling consists of transforming the features domain in a way that they are on a similar scale. Commonly, scaling refers to normalizing, i.e., transform features such that their mean x < = 0 and standard deviation σ= 1. In batch learning, feature scaling is both an important and an uninteresting topic, which is often added to the data transformation pipeline without much thought of the process.

It is infeasible to perform feature scale for data streams as aggregate calculations must be estimated throughout the execution. For landmark window approaches there are exact solutions to incrementally computing the mean and standard deviation without storing all the points .

We need to maintain only three numbers:

The number of data points seen so far - n, the sum of the data points Σ (x), and the sum of the squares of the data points $\Sigma(x^2)$. These statistics are easy to compute incrementally.

The mean is given by

$$\Sigma x/n$$

and the variance is given by

$$\frac{\Sigma x^2 - (\Sigma(x)^2)/n}{n-1}$$

In landmark windows, its easy and fast to maintain exact statistics by storing few numbers. However, in time changing streams the adaptation is too slow. To deal with change, sliding window models are more appropriate. The problem is that exact computation of the mean or variance over a stream in a sliding window model requires to store all data points inside the window. Approximate solutions, using logarithmic space, are the exponential histograms [31].

Exponential histograms store data in buckets of exponential growing size: 20, 21, 22, 23, ... For a window size W, only log(W) space is required. Recent data are stored in fine granularity, while past data are stored in an aggregated form. The statistics computed using exponential histograms are approximate, with error bounds. The error comes from the last bucket, where it is not possible to guarantee all data is inside the window.

3.1.3 Feature Discretization

Discretization is a process that divides numeric features into categorical ones using intervals. Depending on the application and predictive model being used, discretization can bring several benefits, including faster computation time as discrete variables are usually easier to handle compared to numeric ones; and decreases the chances of overfitting since the feature space becomes less complex.

3.2 Dimensionality reduction

Dimensionality reduction takes care of retention of patterns that are relevent to the learning task in the input data . Here are some works and gaps on dimensionality reduction techniques that apply transformations to the input data. Principal Component Analysis (PCA), and Random Projections;being prominent in them.

Table 3.2 Dimensionality reduction methods

Sr. No.	Techniques Used	Details
1	Memory-limited approximation of PCA (calculated under reasonable error bounds)	based on sampling and sketches
2	Single-pass randomized PCA method	Evaluated on a single image dataset
3	An incremental feature learning algorithm to determine the optimal model complexity	online data based on the denoising autoencoder.

4	An Ensemble that combines different random projection techniques	with Hoeffding Trees,applied to realworld data.

3.3 Feature Scaling

Incrementally identifying which features are important is a relevant subject in context of data streams. hoeffding (decision) trees [28] and decision rules [29] are the major representatives of classification and regression systems that can incrementally identify important features .

Table 3.3 : Feature Scaling methods

Sr. no.	Major Issues	Parameters used
1	Evaluation of feature selectors	Accuracy and scalability, ease of use-efficiency of feature selectors
2	Contradiction between feature stability and selection accuracy	
3	Feature Drifts	Compromise between feature stability and selection accuracy

New methods must be designed so that the feature selection process can identify and adapt to changes in the relevance of features, a phenomenon called feature drift . Another critical gap of feature selection in streaming scenarios regards the evaluation of feature selectors. There are different factors to account for when evaluating feature selection proposals. Many times different quantitative measures, such as accuracy and scalability; and ease of use, have been used to highlight the efficiency of feature selectors [30]. As Each learner builds its own predictive model differently although it is fed with the same subset of features, it is of great importance to assess the performance of feature selection algorithms when combined with different learners . On the other hand, it is also important to see that the selected subset of features matches the features that are indeed relevant (i.e. the accuracy).

An open challenge in the streaming setting is the contradiction between feature stability and selection accuracy. If the features' importance shifts over time (feature drifts) then the feature selection method will need to compromise either stability or accuracy.

IV. State-of-the-Art of Data Stream Mining Methods

The data stream mining algorithms proposed so far for supervised and unsupervised learning, are mostly for the classification and clustering tasks . Most of them are based on the standard methods used for learning in static data, with significant modifications so as to be able to deal with data streams.[31,32]

Among clustering algorithms, the most algorithms which shown significant performance are the CluStream algorithm [33], the HPStream algorithm [34], the DenStream algorithm [35], the Very Fast K-Means (VFKM) algorithm [36]. A data stream clustering algorithm as given in [58], is designed for detection of clusters with arbitrary shapes . Standard machine learning methods which stand as a basis for data stream classification algorithms are for example instance-based classifiers, neural networks, Bayesian classifiers or decision trees.

4.1 Instance-Based Classifiers

4.1.1 CluStream clustering algorithm-In this

Step 1 classifier data elements are aggregated in constant number q of microclusters. This ensures the constant amount of memory needed for learning.

Step 2 Each microcluster is described by the three-element tuple: center, radius and weight representing the number of data elements collected in that microcluster.

Step 3 New data elements are included into the closest microclusters,modifying slightly their parameters. This procedure can be performed incrementally.

After each step, the weights of microclusters can be multiplied by a positive real number λ lower than 1. This introduces a forgetting mechanism for older data elements and ensures greater impact on the classifier for the newer ones.[33]

4.1.2 Adaptive Nearest Neighbor Classification Algorithm for Data Streams (ANNCAD)-

In this algorithm,

Step 1 - The space of attribute values is divided into hypercubes at many levels of resolution.

Step 2 - In the finest level, a weight is assigned to each hypercube, which is equal to the number of data elements collected in it.
Step 3 - At the coarser levels, weights are the arithmetic averages of corresponding weights from the finer level.
Step 4 - To each hypercube either a majority class is assigned or it is tagged as 'mixed' if the difference between numbers of the most frequent and the second most frequent classes is below some threshold value.

In the classification process, the distance between the unlabeled data element and centers of hypercubes is taken into account. The classification begins at the finest level. If it does not provide a solution, the coarser level is used.
The algorithm uses a constant amount of memory and can be learned incrementally. The forgetting mechanism [37] can be used to reduce the impact of old data elements in this case as well.

4.2 Bayesian Classifiers

Bayesian classifiers are based on the Bayesian theorem, which express the relations between the prior probabilities. The aim of learning is to estimate the necessary probabilities using the training dataset. A learned classifier is then used to classify new data. The class which maximizes the posterior probability is assigned to an unlabeled element. A Bayesian classifier is designed for data with nominal attributes. Numerical attributes can also be used, however, they first need appropriate preparation. Domains of numerical attributes shouldbe partitioned into bins. The most commonly used learning algorithm in this group is the Naive Bayes Classifier. In this case all attributes are independent is a simplifying assumption. This enablesthe replacement of jointprobabilities with products of probabilities for individual attributes. This assumption significantly simplifies calculations. [38]

Merits:
(a) Naive Bayes Classifiers can be learned in an incremental manner.
(b) They require a constant amount of memory.
(c) By applying the sliding window Naive Bayes Classifiers can react to changes in data concepts.
(d) Pattern-based Bayesian classifiers are effective for evolving data streams .
(e) The idea of applying the forgetting factor to discard the old data in Bayesian classifiers is potentially suitable for the said purpose.
(f) Naive Bayes Classifiers are often used in decision trees as one of the possible classification procedures in leaves .

Demerits:
(a) The main drawback of the Naive Bayes classifier is the omission of potential correlations between attributes.

4.3 Artificial Neural Networks

Artificial Neural Networks , inspired by biological neural networks forming the nervous system in animals are another interesting type of learning model. An artificial neural network consists of neurons and connections between them i.e. synapses. Neurons form layers and connections are described by real number weights. Multi Layer Perceptron (MLP) network is the most common neural network used for classification .
It consists of an input layer, an output layer and one or more hidden neuron layers between them. The network can be trained using the backward propagation of errors (backpropagation) learning method.
For each element from a training dataset, connection weights are slightly adjusted in order to get the predicted output closer to the actual value.
Each data element from a training set is entered to the input layer once in an 'epoch'. In a standard scenario, learning of the MLP network consists of many epochs.
If the number of training data elements is large, as in the case of data streams, learning of a neural network can be in a natural way converted to one-pass incremental algorithm .
Subsequent stream data elements are applied to the input layer; hence, the network automatically and continuously reacts to potential concept drifts.
If the structure of the network, i.e. the number of neurons and synapses, is kept unchanged during learning, then the required memory is constant.
The above features of neural networks make them applicable for data streams.
Recently, the researchers of the computer science community attracted much attention from deep neural networks. Several authors tried toapply deep learning methods to data stream processing. Some of them mentionable are
a) In evolving granular neural networks fuzzy data streams are formed into a number of fuzzy sets, i.e. granules of data. These granules are updated incrementally and provide an input for a neural network.
b) The evolving deep neural network was combined with the Least Squares Support Vector Machine.[39]

c) As given by [40] the deep neural networks were applied to the semi-supervised learning task.
d) Also the different approach with Deep Hybrid Boltzmann Machines and Denoising Autoencoders is used for online learning from data streams.
e) In [41] the Deep Belief Network is learned in an unsupervised manner based on the data stream elements. The rarely occurring labeled elements are used to fine-tune the model to the current data concept.

4.4 Decision Trees

Other important machine learning tools which can be applied to datastream mining are decision trees. The most prominent method of this type is probably the VFDT algorithm [42], based on the idea of Hoeffding trees. As in the case of decision trees for static data, the most crucial point of a Hoeffding tree induction is the choice of an appropriate attribute to split each node of the tree. For each attribute a value of some split measure function S collected in a considered tree node. Split measures are often based on impurity measures, like the information gain in the ID3 algorithm [43] or the Gini gain in the CART algorithm [44].

Therefore, in Hoeffding trees information about past data is stored in the form of so-called sufficient statistics. It is a three-dimensional structure containing the number of data elements for each class and each value of each attribute (or each bin of each attribute in the case of numerical features). Therefore, the amount of memory required for Hoeffding trees is proportional to the number of leaves. Therefore, in the VFDT algorithm a memory management mechanism is applied. Only a constant number M of leaves is activated during learning and collects information to sufficient statistics. The error rate of each leaf is monitored. At a given moment M, leaves with the highest value of error rate are set to the active mode. In Hoeffding trees a node is split according to the attribute providing the highest value of split measure only if the number of data elements collected in the considered node is large enough. This number should be as large as to ensure that with high probability $1-\delta$ $= (1-\delta)^{(D-1)}$ the chosen attribute would also provide the highest expected value of split measure among all attributes.

Table 4.1 : Different approaches used for Decision Trees

Sr. No.	Approach used for Decision Trees	Method
1	VFDT algorithm and Hoeffding Trees	A constant number M of leaves is activated during learning and collects information to sufficient statistics
2	Concept adaptive Very Fast Decision Tree (CVFDT)	Able to react to changes in data distribution. In this method, a sliding window is applied. The sufficient statistics in nodes represent only the data kept in the window at a given moment.Additionally, the possibility of creating alternative subtrees was introduced. Sufficient statistics are kept in all nodes of the tree, and not only in leaves as it is in the case of the VFDT algorithm
3	Hoeffding Option Tree	It adapts the idea of option trees to the Hoeffding trees. As in the case of the CVFDT algorithm, sufficient statistics are stored in both
4	StreamDM-C++	Decision trees for data streams in C++ and allows inducing trees which are able to adapt to changes in the distribution of data stream elements.
5	Fast Incremental Model Trees with Drift Detector (FIMT-DD)	It is designed for data with numerical attributes. The variance reduction was chosen as a split measure function.
6	FlexDT algorithm	It is a combination of decision trees with fuzzy logic

4.5 Ensemble Classifiers

Methods of this kind are actually meta-algorithms in which various approaches for combining many single classifiers into a group are proposed. The learners to be used as component classifiers are not imposed by these methods in advance.

Streaming Ensemble Algorithm(SEA): In [45] the Streaming Ensemble Algorithm (SEA) was proposed which processes , the stream in a block-based manner.New classifiers are learned on subsequent blocks of data.,keeping the maximum number of classifiers fixed. The accuracy of new classifiers is verified on new blocks. Based on these results the quality of each classifier is evaluated. If the new classifier demonstrates a higher quality than some classifiers from a given ensemble, then the worst classifier is replaced by a new one. If an unlabeled data element needs to be classified, then each component of the ensemble predicts its own class for it (it gives a 'vote'). A class which obtains the most of votes is assigned to the element.

Accuracy Weighted Ensemble (AWE): A similar algorithm, the Accuracy Weighted Ensemble (AWE), was proposed in [47]. In this approach, the votes of classifiers are weighted by their average accuracy. The authors of this algorithm proved a theorem which states that an ensemble built from L data blocks provides a smaller classification error thana single classifier built on all data from these L blocks if the components of the ensemble are weighted by their expected classification accuracy.

Adaptive Classifier Ensemble (ACE): In [48] the Adaptive Classifier Ensemble (ACE) algorithm was presented. In this approach, a single online classifier is learned in parallel with an ensemble of block-based ones. This online learner facilitates a reaction to concept drift. In [54] the authors instead of relying on a traditional voting, classifiers are allowed to abstain from contributing to the final decision. The Data stream processing with the idea of an iterative boosting-based ensemble was proposed in [55]. Another important ensemble classifier meta-algorithms are the Learn++.NSE algorithm [53], the Dynamic Weighted Majority (DWM) algorithm [52], the Diversity for Dealing with Drifts (DDD) algorithm [51], the Accuracy Based Weighted Aging Ensemble Algorithm [45].

Table 4.2 Ensemble classifiers for Data Stream

Sr. No.	Ref .	Ensemble Classifier	Methodology
1	49	Streaming Ensemble Algorithm (SEA)	New classifiers are learned on subsequent blocks of data. Their accuracy is verified on new blocks.
2	47	Accuracy Weighted Ensemble (AWE)	the votes of classifiers are weighted by their average accuracy
3	48	Adaptive Classifier Ensemble (ACE)	a single online classifier is learned in parallel with an ensemble of block-based ones. This online learner facilitates a reaction to concept drift.
4	53	Learn++.NSE	Learn++ utilizes ensemble of classifiers by generating multiple hypotheses using training data sampled according to carefully tailored distributions. The outputs of the resulting classifiers are combined using a weighted majority voting procedure
5	52	The Dynamic Weighted Majority (DWM)algorithm	Dynamic weighted majority (DWM) maintains an ensemble of base learners, predicts using a weighted-majority vote of these "experts", and dynamically creates and deletes experts in response to changes in performance.
6	51	The Diversity for Dealing with Drifts (DDD)	DDD maintains ensembles with different diversity levels and is able to attain better accuracy than other approaches. Furthermore, it is very robust, outperforming other drift handling approaches in terms of accuracy when there are false positive drift detections.
7	45	Accuracy Based Weighted Aging Ensemble Algorithm	easily adapt to the probability characteristic changes caused by so-called concept drift. one individual classifier is trained on the basis of the each incoming data chunk
8	46	Accuracy Updated Ensemble (AUE)	able to react to various types of concept drift

V. Advanced issues in data stream analysis:

5.1) Evaluation Measures :

Machine Learning is to be assisted with Proper evaluation measures [10][11]. In the context of data stream mining, especially in non-stationary environments, new solutions are needed. While evaluating predictive ability, it is inevitable to take into account incremental processing,evolving data characteristics and the in-response actions of the classifier to changes. New classes may appear, feature space may change and decision rules lose relevance over time. Moreover, computational aspects such as processing time, recovery of the model after the change, and memory usage should also be taken into account. So, instead of gathering data for a longer period of time and trying to rebuild the model in a single time consuming step, It ismore reasonable to achieve speedy updating of a learning model and gradual recovery. Usually one is more interested in tracking working characteristics over the course of stream progression than examining point or average prediction measures of the classifier, .

following performance metrics play a vital role in Data stream Evaluation .

Accuracy : the proportion of all correct predictions to the total number of examples, or its corresponding measure classification error , are the most commonly used for classification.

Mean square error or absolute error is a typical measure for regression.

Sensitivity of the class of interest (also called Recall or True Positive Rate) is accuracy of a given class.

G-Mean : the geometric mean of sensitivity and specificity is often applied on class-imbalanced data streams to avoid the bias of the overall accuracy.

Kappa Statistic : K = p0−pc/ 1−p c , where p 0 is accuracy of the classifier and pc is the probability of a random classifier making a correct prediction. [19]
Generalized Kappa Statistics such as Kappa M proposed in[10,11], which should be more appropriate than the standard Kappa Statistics for dealing with imbalanced data streams.
Apart from the predictive accuracy or error, the following performance metrics should be monitored and taken into account during properly executed evaluation of streaming algorithms [20]
Memory Consumption : It is necessary to monitor the average memory requirements of each algorithm, changes in memory requirement with respect to actions being taken.
Update Time : This is the amount of time that an algorithm requires to update its structure and accommodate new data from the stream. Ideally, the update time should be lower than the arrival time of a new chunk of data.
Decision Time : Amount of time that a model needs to make a decision regarding new instances from the stream. So, any decision latency may result in creating a bottleneck in the stream processing.
A complete framework for evaluating the recovery rate of the algorithm ,once a change has occurred in the stream ;has been proposed by Shaker and Hãlermeier [21] .The authors have considered the issues like how well the model reduced its error in the new decision space,and what was the time necessary to achieve this. Zliobaite et al. [22] introduced the notion of cost-sensitive update in order to evaluate the potential gain from the cost (understood as time and computational resources) put into adapting the model tothe current change. The authors argue that this allows to check if the actual update of the model was a worthwhile investment.

5.2) Software Packages And Frameworks
There are some existing tools available for applying machine learning to data streams for research as well as practical applications. These can also be adapted to deploy models in practice based on the problem requirements.[15-18]
Following table summarizes the software packages and frameworks.

Table 6.1 Software Packages and Frameworks

Sr. No.	Framework/Package	Details
1	Massive Online Analysis (MOA)[15]	several algorithms for classification, regression, multi-label, multi-target, clustering
		Provides Data generators,evaluation methods ,statistics
2	Advanced Data mining And Machine learning System (ADAMS)[16]	workflow engine can be combined with MOA, and other frameworks like SAMOA and WEKA
3	Scalable Advanced Massive Online Analysis (SAMOA)[17]	combines stream mining and distributed computing (i.e., MapReduce)
4	StreamDM [18]	Open source framework for big data stream mining

This will facilitate collaboration among research groups and for providing the evaluation and test platform .

5.3) Datasets:
5.3.1 Handling Real Time Streaming Data :
An important and significant aspect of streaming data is how the data is made available to the system. It is difficult for learning algorithm as High latency data sources will 'hold' the whole system. The data source for streaming data is often harder to grasp for beginners, due to the major differences like it is not merely a self-contained file or a well defined database. In fact, it has to allow the appearance of new data with low latency so that the learner is updated as and when new data is available.
Stream Processing frameworks, such as Apache Spark and Flink ,recently, introduced a novel API to handle streaming data, namely the Structured Streaming API which overshadows the previous Spark Streaming API. Similar to Spark Streaming API , Structured Streaming is mainly based on micro-batches(the rows are combined into small logical batches) , instead of immediately presenting new rows of data to the user code. This facilitates manipulating the data. The problem with truly incremental systems is difficulty in handling and efficient implementations respectively for users and the framework developers. The new Structured Streaming API assumes that there is a structure to the streaming data, which makes it possible to manipulate data using SQL and uses the abstraction of an unbounded table.

A thorough evaluation of predictive models in non-stationary environments typically requires the use of not only real world data streams, but also data streams with artificially generated concept drifts.

The number of real-world publicly available datasets for testing stream classifiers is still too small. Moreover, some popular data electricity data [12] is questioned to represent sufficiently real characteristics of streaming data as real drifts . This is a more difficult situation compared to the state of available static datasets such as the UCI Machine Learning Repository. The example datasets obtained from UCI Machine Learning Repository are as described below [14].

5.3.1.1 UCI Machine Learning Repository:

a) Covertype Data Set - This dataset consists of data denoting the parameters of forest cover, with the objective of obtaining a prediction of the cover type only from cartographic variables. This dataset consists of 581012 data elements, each of which is a 54-dimensional vector (10 quantitative variables, 4 binary wilderness areas, and 40 binary soil type variables). Each data element is labeled by one of seven classes describing the types of forest cover.

b) Abalone Data Set - This dataset was created to predict the age of abalone from physical measurements. Each data element in this set consists of the 8 attribute values (one nominal and 7 numerical) and one integer value (class) describing the abalone age. There are 29 classes and 4177 instances in this dataset.

c) Connect-4 Data Set - This dataset consists of data describing the possible moves in game connect-4. Only not forced moves are taken into account. In this dataset there are 42 attributes with possible values 'x', 'o', 'b' denoting 'player x has taken', 'player o has taken' and 'blank', accordingly, where 'x' is the first player and 'o' i the second player. There are three class labels (win, loss, draw) denoting the outcome regarding the first player. There are 67557 instances in this dataset.

Table 6.2 summarizes the information of these datasets.

Table 5.2 Datasets from UCI Machine Learning Repository

Sr. No.	Dataset	Metadata
1	Cover type Data Set	581012 data elements- each of which is 54-dimensional vector. (10 quantitative variables, 4 binary wilderness areas, and 40 binary soil type variables)
2	Abalone Data Set	29 classes and 4177 instances in this dataset. Each data element consists of the 8 attribute values (one nominal and 7 numerical) and one integer value (class) describing the abalone age.
3	Connect-4 Data Set	67557 instances ,42 attributes in this dataset.

VI. Concluding Remarks and Challenging Problems for future research:

Stream mining is a challenging area of research but has valuable yields, especially in industry and commercial applications. Data streams offer qualitative and quantitative information that is of enormous use in different ways to boost business efficiency. However the unbound size, unknown speed and varying characteristics of data streams make applying machine learning techniques a complex task.

Through an in-depth and critical review of existing literature, this paper has identified the following shortcomings:

- need to incorporate per-processing techniques that continuously transform the raw data;
- Explore the relationships between other AI developments (e.g., recurrent neural networks, reinforcement learning, etc.) and adaptive stream mining algorithms;
- Characterize and detect drifts in the absence of immediately labeled data;
- Develop adaptive learning methods that take into account verification latency;

we briefly reviewed challenges and approaches common in the field of streaming data analysis,

Future trends in streaming data analysis, are based on

1) how to **develop data streaming methods that scale to Big Data like large deep neural networks, but work well in all domains.** In the future, the quantity of data generated in real- time is going to grow, so there will be need to develop new methods using large distributed systems.

2) Deep Learning has become a very extreme successful use case for Machine Learning and Artificial Intelligence, due to the **availability of massive quantities of data to build data models, and large computational resources** . Standard deep learning techniques needs to do several passes over the data. How to build models only doing one pass over the data, without storing the data, will be an important future area of research .

3) Finally, when dealing with large quantities of data, **an important trend will be how to do online learning using distributed streaming engines, as Apache Spark, Apache Flink, Apache Storm and others.** Algorithms have to be distributed in an efficient way, so that the performance of the distributed algorithms does not suffer from the network cost of distributing the data.

Ms. Shailaja B. Jadhav, et. al. "A Comprehensive –Theoretical Review of Data Stream Mining: methods and challenges." *IOSR Journal of Computer Engineering (IOSR-JCE)*, 23(3), 2021, pp. 55-67.