applied sciences

MDPI

Review

Process-Oriented Stream Classification Pipeline: A Literature Review

Lena Clever [1,*], Janina Susanne Pohl [1], Jakob Bossek [2], Pascal Kerschke [3] and Heike Trautmann [1,4]

[1] Department of Information Systems, University of Münster, 48149 Münster, Germany
[2] Department of Computer Science, RWTH Aachen University, 52062 Aachen, Germany
[3] "Friedrich List" Faculty of Transport and Traffic Sciences, TU Dresden, 01062 Dresden, Germany
[4] Data Management & Biometrics Group, University of Twente, 7522 NB Enschede, The Netherlands
* Correspondence: lena.clever@wi.uni-muenster.de

Featured Application: Nowadays, many applications and disciplines work on the basis of stream data. Common examples are the IoT sector (e.g., sensor data analysis), or video, image, and text analysis applications (e.g., in social media analytics or astronomy). With our work, we gather different approaches and terminology, and give a broad overview over the topic. Our main target groups are practitioners and newcomers to the field of data stream classification.

Abstract: Due to the rise of continuous data-generating applications, analyzing data streams has gained increasing attention over the past decades. A core research area in stream data is stream classification, which categorizes or detects data points within an evolving stream of observations. Areas of stream classification are diverse—ranging, e.g., from monitoring sensor data to analyzing a wide range of (social) media applications. Research in stream classification is related to developing methods that adapt to the changing and potentially volatile data stream. It focuses on individual aspects of the stream classification pipeline, e.g., designing suitable algorithm architectures, an efficient train and test procedure, or detecting so-called concept drifts. As a result of the many different research questions and strands, the field is challenging to grasp, especially for beginners. This survey explores, summarizes, and categorizes work within the domain of stream classification and identifies core research threads over the past few years. It is structured based on the stream classification process to facilitate coordination within this complex topic, including common application scenarios and benchmarking data sets. Thus, both newcomers to the field and experts who want to widen their scope can gain (additional) insight into this research area and find starting points and pointers to more in-depth literature on specific issues and research directions in the field.

Keywords: data mining; big data; stream classification; data stream analysis; supervised learning; machine learning

Citation: Clever, L.; Pohl, J.S.; Bossek, J.; Kerschke, P.; Trautmann, H. Process-Oriented Stream Classification Pipeline: A Literature Review. *Appl. Sci.* **2022**, *12*, 9094. https://doi.org/10.3390/app12189094

Academic Editor: Luoyi Fu

Received: 1 August 2022
Accepted: 6 September 2022
Published: 9 September 2022

1. Introduction

The analysis of stream data has become increasingly important in recent years. For instance, applications based on real-time sensor data or images have become indispensable in the Internet of Things (IoT). The characteristic of stream data is that it comes in large quantities and at a high speed. Moreover, this kind of data often is of relatively large dimensionality, i.e., it contains numerous features and observations [1]. Manual data processing is infeasible; thus, a broad field of data stream mining methods has emerged over the years. As in the static environment, the core tasks include unsupervised and supervised learning. In this work, we address research questions and challenges of supervised learning methods that are exposed to data streams—concretely, we focus on stream classification. Data stream classification finds several application areas. These include, for example, the detection of specific gestures or elements in video data, the interpretation of a device's sensor data, or the analysis of telescope data [2,3].

Regardless of the specific application, the process of data stream classification consists of several steps and modification options. Next to typical classification steps, such as data preprocessing and training and testing the classifier model, other essential tasks open up in the streaming scenario. While the initial training can occur in an offline mode, the validation needs to be performed during the whole lifetime of an online process. Changes in the data, such as the shift of feature distributions, or the emergence of new classes, can weaken the algorithm's quality over time. If a change in the data is detected and the performance of classification results decreases substantially, the model needs to be updated immediately. For good model quality, update procedures must guarantee continuous adaptation to the data stream without neglecting the actual classification task. During the entire usage cycle, fast processing of new incoming data points must be ensured.

Due to these numerous challenges, a broad field of work on stream classification has emerged over the years. Methodologically, the field is vast. On the one hand, there is work on developing and further refining (classification) algorithms. On the other hand, researchers examine closely related topics such as drift detection and adaption techniques. Diverse research fields produced problem-tailored solution approaches as well as terminology and taxonomies. Thus, the field also grew in summary papers and comparative work. Benchmark studies, for example, deal with the comparison of different algorithms concerning efficiency and speed. Survey papers structure the field of data stream mining and bring the notations and wordings of the different disciplines to a common understanding. Due to the size of the research field, it is striking that the works often deal only with a partial aspect of the stream classification process. For example, they focus on change detection or algorithm structure [4,5]. Newcomers or interested researchers thus have to navigate through various papers to gain a holistic overview. Our main contribution is to provide a road map in stream classification literature. Thus, for better comprehensibility, we structure the research field from a process-centered point of view along a stream classification pipeline (see Figure 1). For each process step, we refer to relevant, comparative, and summary works and future research directions. Our underlying literature base spans approximately 320 papers from the stream classification discipline between 1984 and 2021. We specifically searched for summary papers and recent publications in the field.

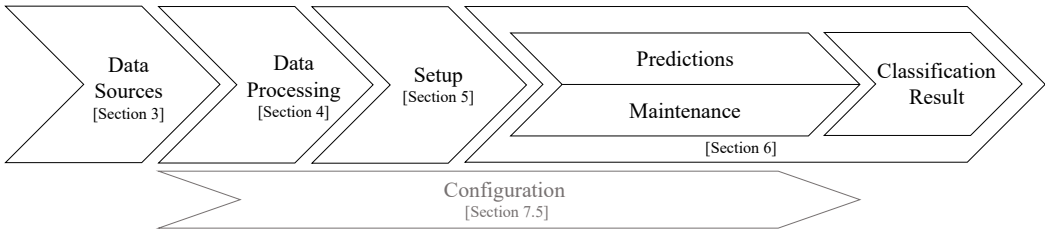

Figure 1. Scheme of the process that is employed by most stream classification algorithms.

The remainder of this work is structured as follows: In Section 2, we provide an overview of the underlying problem description, the literature base as well as the stream classification pipeline, which is the underlying structure of this work. Following the stream classification pipeline, the subsequent sections are organized accordingly: Data Sources and Application Scenarios (Section 3), Process Data (Section 4), Classifier Algorithms and Architectures (Section 5), as well as Classifier Maintenance (Section 6). We group the discussed topics in a decision tree to support the reader's understanding in each section. In Section 7, we bring the individual components together and highlight the latest and future research directions.

2. Background and Stream Classification Pipeline

In the following, we will first define stream classification and its requirements. Second, we will introduce our stream classification pipeline, which serves as a structure for this work. Finally, we report on our literature search procedure.

2.1. Definition and Requirements of Data Stream Classification

Classification is a supervised learning technique for solving decision problems [6]. To avoid confusion with non-stream classification scenarios, we use the terms static or offline for traditional classification, while we use the terms stream-based or online when talking about the in-time processing of data streams.

The overall goal of classification is to assign one or more (see Section 5.2) discrete classes to an observation. For this purpose, a model is trained on a set $S = \{(\vec{x}_1, y_1), \ldots, (\vec{x}_n, y_n)\}$ of n labeled observations. Such as in common literature, $\vec{x}_i = (x_{i1}, \ldots, x_{ip}) \in \mathcal{X}$ defines the feature vector of the i-th observation, $i \in \{1, \ldots, n\}$, with $\mathcal{X}$ being the p-dimensional feature space. Depending on the space, the features are said to be either numerical ($x_{ij} \in \mathbb{R}$), ordinal or categorical ($\vec{x}_{ij} \in \{r_1, \ldots, r_m\}$ for some $m > 0$), respectively. Moreover, $y_i \in \mathcal{Y}$ indicates the corresponding class label from the finite set $\mathcal{Y} := \{C_1, \ldots, C_l\}$ of l classes.

This labeled—so-called training data—is used to construct a classification model or discriminator $D : \mathcal{X} \times \Theta \to \mathcal{Y}$, which ideally predicts the correct class label $D(\vec{x}) \in \mathcal{Y}$ for a previously unseen observation $\vec{x} \in \mathcal{X}$ with a high likelihood. Here, Θ defines a parameter space that is specific to the considered learning algorithm. The performance of a classification model is usually evaluated by applying the model to and assessing it on a set of labeled instances, which has not been used during the training procedure itself.

In contrast to *static* classification problems, the goal of a stream classification algorithm is to classify data points that are continuously arriving. Thus, the algorithm consists of an online phase, in which it operates on the incoming data stream $S = \{(\vec{x}_1, y_1), (\vec{x}_2, y_2), \ldots\}$. Observations in the data stream follow the formal definition of the offline scenario but come in a temporal order. According to the literature on basic stream (classification) algorithms, a data stream is a continuous, ordered, and potentially unbounded sequence of observations. Data points can arrive at high speed [1].

Since data streams are assumed too large to fit entirely into the main memory, new observations (or batches) have to be processed and discarded afterward [1]. All operations on data thus must be highly efficient, since only a limited amount of computational memory is available. The model must be feasible to predict labels of new incoming observations at any point in time without a high temporal delay, even when the data stream changes, e.g., caused by new incoming classes or changing feature space.

In literature, it is assumed that only a limited number of labeled data points are available for training. This goes with practical application scenarios, where human coders label a subset of the data to build an automated classification model. Coming to the stream setting, it is often assumed that newly labeled observations occasionally arrive, such that the model needs to adapt itself to changes in the data stream in a continuous manner. This is in contrast to the classical static classification scenarios, in which models are set up once at the beginning (based on a batch of training data) and remain fixed. Thus, stream classification aims to train a continuous sequence of classification models.

2.2. Process-Oriented Stream Classification Pipeline

In general, stream classification follows the pipeline sketched in Figure 1. First, data are integrated from one or many sources depending on the respective streaming scenario. Researchers and practitioners can use suitable benchmarking datasets to evaluate their algorithms and solutions. Datasets are as versatile as the application areas of stream classification, as discussed in Section 3.2. Together with respective frameworks in Section 3.3 for automated stream data classification, they provide a working environment for experiments.

Independent of which dataset has been used, most probably, the data must be prepared before the actual classification. Data processing could include, e.g., segmentation of various data sources, labeling, or data type-dependent preprocessing, e.g., feature extraction or feature space simplification.

Coming to the classification model itself, we can differentiate between two states. Commonly, in the first phase, an initial model is set up. In Section 5.2, we give a detailed overview of influential algorithms in the field. We structure algorithms in terms of their

architecture and type. After the setup phase, the classification algorithm can be used continuously to predict the class of incoming data records.

During its lifetime, a stream classifier needs to be maintained. To understand the challenges of classifier maintenance, we introduce basic concept drift analysis literature in Section 6. Therein, we will give insights into the types of changes, as well as into their detection. Further, we give an overview of stream classification models' evaluation and update procedures. Next to the continuous evaluation and update of the model, the algorithm's parameters should be configured continuously during the whole lifetime of the model.

2.3. Literature Base

We first searched for various summary and comparative works in the field as a basis for this work. We conducted our search on Google Scholar (see www.scholar.google.com (accessed on 5 September 2022)), Web of Science (see www.clarivate.com/webofsciencegroup/solutions/web-of-science (accessed on 5 September 2022)), and ArXiv (see www.arxiv.org (accessed on 5 September 2022)). We combined terms of the individual process steps with the keywords "survey", "review", and "benchmarking". An overview of the search terms used for each step of the stream classification pipeline is depicted in Table 1. For example, we collected papers on the drift detection process step by searching for "stream classification" + ("drift detection", "change detection", "drift adaptation") + ("survey", "benchmark", "review"). We mainly focused on the last five years and included influential work via a backward search.

Table 1. Overview over search terms of the structured literature search.

Step	Search Terms
Data sources	Application, dataset, repository, framework
Data processing	Input, window, segmentation, preprocessing, labeling
Setup	Algorithm
Maintenance	Drift detection, change detection, drift adaptation, evaluation, monitoring, assessment, drift detection evaluation, ensemble evaluation
Configuration	Configuration, tuning, parameter

We further searched for the most recent work of the last three years by searching for the term "stream classification" on all three platforms. In addition, we have checked all papers to observe whether they fit qualitatively and thematically. Papers without a recognizable conference, journal affiliation, or quality issues (e.g., apparent lingual deficits) were sorted out. All in all, our search ended up with 329 papers. Within our literature base, we found no work which goes along with the idea of structuring papers employing a stream classification pipeline. Scouted papers mainly focused on specific steps of the stream classification pipeline or took a more high-level view of data stream mining.

3. Data Sources and Benchmarking

The most relevant difference between data stream classification and batch processing is that data comes in a steady, endless stream of large quantities, which must be processed continuously and immediately. Many application areas require this kind of processing, as observed in Figure 2, where we give an overview of the topics covered in this section.

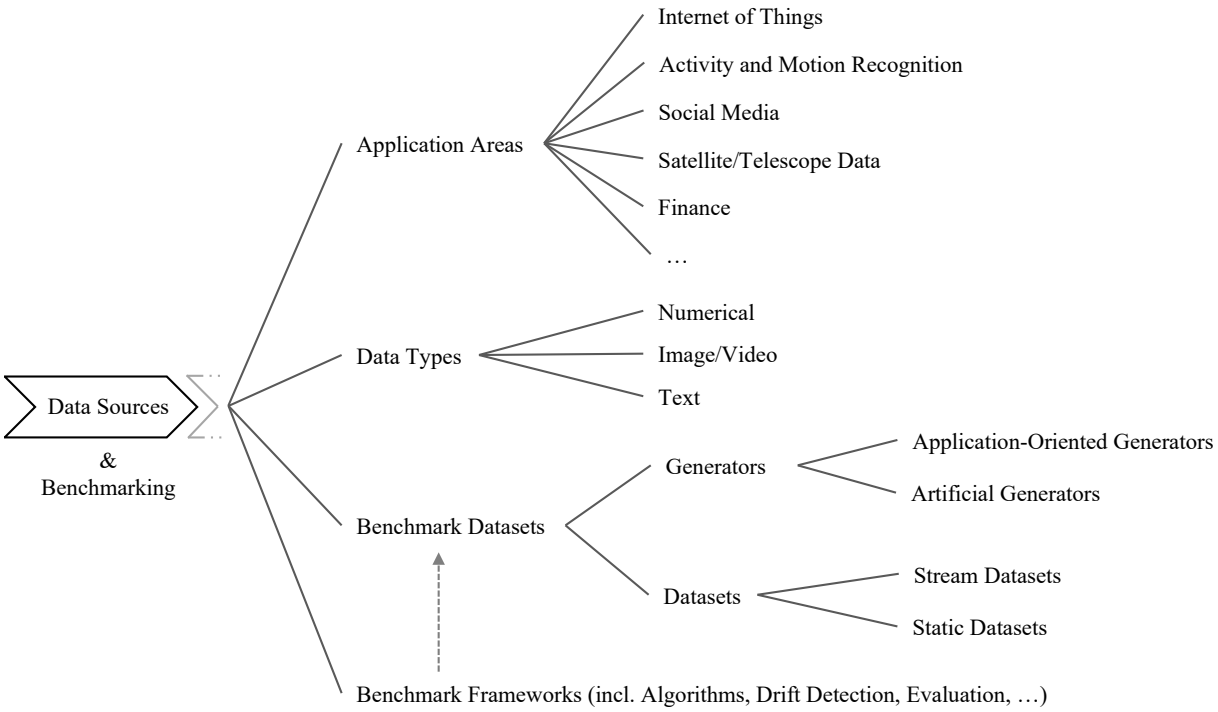

Figure 2. Typical application areas and data types, as well as a categorization of benchmarking datasets and frameworks, stand at the beginning of a stream classification pipeline.

3.1. Application Areas and Data Types

The most relevant difference between data stream classification and batch processing is that data comes in a steady, endless stream of large quantities, which must be processed continuously and immediately. Many application areas require this kind of processing, as observed in Figure 2, where we give an overview of the topics covered in this section. Stream classification applications can be found in almost all areas where data streams are generated, but they differ depending on the respective conditions and problems, such as the underlying data types. Typical stream data types are numerical or categorical data of sensor environments, video or image data, and text data, for example, from social media sources. In the following, we give an overview of the largest application areas found during our literature search. Additionally, an overview of common stream classification application areas is given in the works of [7,8].

A frequently mentioned field where stream classification algorithms are applied is the IoT sector. IoT applications rely on sensor sets or intelligent devices, which constantly gather data and monitor the surroundings [9]; an exemplary use case is smart vehicles [10]. Intelligent decisions must be made in real time by data stream algorithms based on sensor data input for automated driving maneuvers. In addition to driver support, vehicle data can also be used to assist public transport, e.g., to detect traffic jams or delays [11].

In the field of activity recognition, video data can be analyzed with the help of stream classification algorithms [12–14]. Typical application scenarios are, e.g., human motion capturing for the automatic recognition of sign language or accident detection in home surveillance applications for older adults, or vehicle detection [2,15,16]. In addition to surveillance applications in public and private spaces, stream classification algorithms are also used in the entertainment sector. For instance, human gesture recognition is utilized in video game development [13].

With the rise of social media, a new application area for stream-based algorithms was born. Participatory online platforms such as Facebook or Twitter invite users to publish their content in texts, photos, and videos. Over time, several algorithms have been developed to analyze the development of discussions within these platforms. Tasks and problems here are, for example, the automated detection of hate speech or cyber-bullying [17]. There

are also efforts to detect texts that indicate health problems such as depression or even emergency event detection [18–20]. Social media researchers are dealing with concise texts compared to other applications on (stream) text data, such as spam email detection. Therefore, an algorithm has less contextual information to learn the true meaning of texts for classifying, e.g., sentiments [21,22].

Another application area for stream algorithms is the analysis of telescope data. Detecting stellar behavior or other astronomical events requires algorithms that can deal with large amounts of image data [3,23–25]. In contrast, ref. [26] uses satellite data to monitor forest development, and ref. [27] uses sensor and satellite data to detect abnormal events (fires, animal movement) in forest areas. Another application of image data is the health industry, in which medical image data is used to detect abnormalities or signs of illness [28]. Recently, images related to COVID-19 patients were also in the focus of research, i.e., facial mask detection and ultrasound images [29,30].

Stream algorithms are also used in the financial sector to detect financial distress periods or abnormal developments and intrusion detection [31]. In the left part of Figure 3, we show the distribution of papers in our literature base distinguished by the disciplines in which they were published.

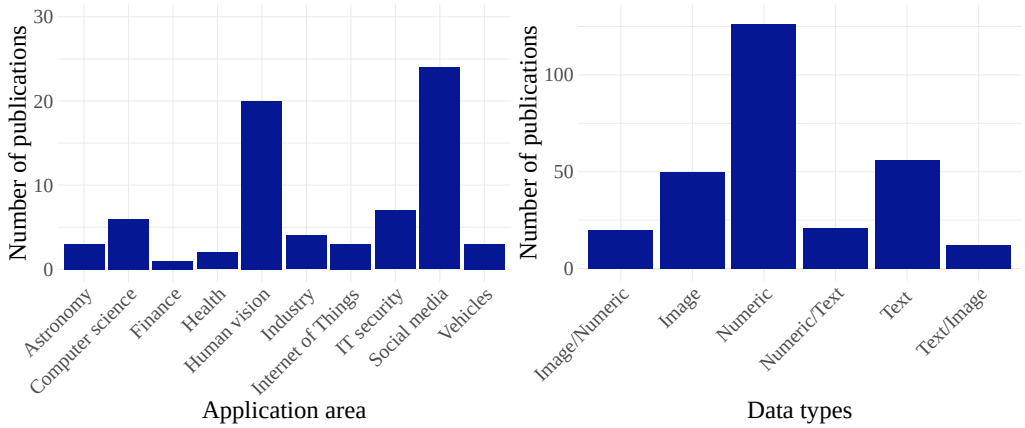

Figure 3. Distribution of outlet disciplines and data types in our literature base.

A comprehensive analysis often requires the combination of different data sources and types, such as the fusion of different sensor data or merging of image and text data in the social media area [9]. In our literature base, approximately 10% of the publications incorporate more than one data type. The results can be observed in the right part of Figure 3.

3.2. Benchmark Datasets

As shown in the previous section, many application areas for stream classification exist. To compare the performances of algorithms across different scenarios, one usually utilizes synthetic, or real-world benchmark datasets [13]. Tables 2 and 3 provide an overview of general-purpose datasets for both categories. For an extended overview, publicly accessible platforms and repositories such as OpenML (see www.openml.org (accessed on 5 September 2022)) by [32] or the UCI Machine Learning Repository (see www.archive.ics.uci.edu (accessed on 5 September 2022)) can be used. It should be noted, though, that the datasets available on these platforms are primarily used in static supervised learning. A recent survey by [33] includes many of the datasets listed below, as well as a comparison of which dataset was used most often in previous studies by other researchers.

The content of Tables 2 and 3 has been derived from the surveys of [4,34–39]. Furthermore, the works of [40,41] about multilabel stream classification as well as semi-supervised problems instances have been included, respectively. In addition, ref. [42] proposes a process for generating artificial problem instances and thus presents several synthetic generators and datasets in our tables. These summarizing works serve as a basis for our tables, since they generally focus on stream classification, covering a broad range of application areas. Other works focus on more specific application scenarios, and thus, they are briefly discussed below:

Refs. [43,44] focus on the task of intrusion detection, while ref. [28] give an overview of datasets including medical images. Here, for example, ultrasound images can be used to detect anomalies early. For human activity recognition problems, refs. [13,14] provide an extensive overview. Activity recognition problems encompass images of cameras that capture people's movements to, for example, monitor patients in hospitals or detect threats via video surveillance in public spaces [14]. For activity recognition problems, the Center for Advanced Studies in Adaptive Systems (CASAS) (see http://casas.wsu.edu/ (accessed on 5 September 2022)) also provides a collection of datasets. Ref. [8] list datasets in the IoT domain, while ref. [45] describe datasets for text detection and recognition in images. Although social media is another essential application for text recognition in stream classification, no benchmark exists. This is partly due to the network's strict legal data-sharing restrictions. Ref. [46] propose not to share the data but the algorithms to solve this lack of opportunities for algorithm performance comparison. Then, the researcher holding the data can run all algorithms on one machine based on his or her data set and can publish the comparative results.

3.2.1. Generators

In their work, ref. [4] differentiate between two different kinds of generators: artificial generators return a carefully structured and parametrized problem landscape, while application-oriented generators produce instances for a real-world problem with artificial drifts. Generators of the first group implement an artificial mathematical concept, e.g., a hyperplane, that defines the problem landscape. The function returns an endless stream of numerical data that can be used as input data for stream classification. By adjusting the function parameters, the problem landscape can be shaped differently. One advantage of using this kind of dataset is that we have a ground truth a priori, i.e., we know about changes in the underlying data distribution. Additionally, unlike publicly available real-world datasets, the stream is potentially unbounded, since new instances can always be generated. In Table 2, we present standard generators of synthetic stream classification datasets created via predefined functions. One example is the Sine generator that returns points above or below a sine curve [47].

Generators of the second group produce data records based on application-oriented scenarios. An example of such a generator is the Ocean Waveform Generator by [48], which simulates three different wave shapes that a stream classification algorithm has to identify. The prediction of different wave shapes is, e.g., required for the operation of wave energy converters for sustainable energy generation [48].

Table 2 depicts each generator's (abbreviated) name and a short description of its application domain. More than two references imply that other researchers significantly enhanced or altered the dataset. Furthermore, we included the number of attributes of the generated dataset, the number of classes, and, if so, which concept drift (for detailed information about different types of concept drifts, see Section 6). The authors included:

Table 2. Synthetic data generators from different application scenarios of stream classification. Above the horizontal line, we summarized artificial generators, while we considered application-oriented generators below.

	Generator	Description	Attributes	No. Classes	Concept Drift
Artificial	BG-FD [49]	Binary Generator with Feature Drift	2 categorical	3	Feature
	Circle [47]	Four contexts defined by four circles	2 numerical	2	Gradual
	Gauss [47]	Normally distributed data	2 numerical	2	Abrupt
	Mixed [47]	Different functions used	2 numerical, 2 categorical	2	Abrupt
	RandomRBF [50]	Random Radial Basis Function	Simple: 10 numerical Complex: 50 numerical	Arbitrary	None
	R-RBF with drift [50]	Random RBF with drift	Simple: 10 numerical Complex: 50 numerical	Arbitrary	Gradual
	RHG [51]	Rotating Hyperplane Generator	Arbitrary	Arbitrary	Gradual, Incremental
	RHG with drift [50]	RHG with drift for each attribute	Arbitrary	Arbitrary	Incremental
	RTG [51]	Random Tree Generator	Simple: 10 numerical, 10 categorical Complex: 50 numerical, 50 categorical	Arbitrary	Abrupt
	RTG-FD [49]	RTG with Feature Drift	Simple: 20/Complex: 100	2	Features
	SEA [52]	Streaming Ensemble Algorithm	3 numerical	2	Abrupt
	SEA-FD [49]	SEA with Feature Drift	3 numerical	2	Feature
	Sine [47]	Points below or above sinus curve	2 numerical	2	Abrupt
	STAGGER [53]	Boolean function	3 categorical	2	Abrupt
Application-oriented	Agrawal [54]	Loan applications approval	6 numerical, 2 categorical	2	None
	LED [55,56]	Digit prediction on LED display	24 categorical	2	Abrupt
	LED with drift [50,56]	LED generator with drift	24 categorical	2	Arbitrary
	Rotating checkerboard [57]	Generates virtual drift of checkerboard	2 numerical	2	Gradual
	WaveForm [48,55]	Detect Wave Form	Simple: 21 numerical Complex: 40 numerical	3	Abrupt
	WaveForm-FD [48,55]	Detect Wave Form with Feature Drift	Simple: 21 numerical Complex: 40 numerical	3	Feature

Table 3. Real-world datasets from different application scenarios of stream classification.

Dataset	Description	Observations	Attributes	No. Classes	Discipline
Adult [58]	Predict income	48,842	14 mixed types	2	Miscellaneous
Airline [59]	Flight delays in USA	116×10^6	13 mixed types	2	Vehicles
AWS Prices [60]	Bids on server capacity	27.5×10^6	6 mixed types	Arbitrary	Industry
CIFAR [61,62]	Tiny colour images	60,000	32×32 pixels	10	Computer Vision
COCO [45]	Text recognition in images	173,589	5 mixed types	Arbitrary	Computer Vision
Electricity [47,63]	Relative price changes	45,312	8 mixed types	2	Industry
ECUE 1 [64]	E-mail spam filtering	10,983	287,034 tokens	2	Miscellaneous
ECUE 2 [64]	E-mail spam filtering	11,905	166,047 tokens	2	Miscellaneous
E-Mail data [65]	E-mail headings	1500	913 words	2	Social Media
Forest Covertype [66]	Covertype for quadrants	581,012	54 mixed types	7	Miscellaneous
Gas Sensor Array [67,68]	Gas type identification	13,910	8 mixed types	6	Industry
Kddcup99 [69]	Intrusion detection	494,021	41 mixed types	23	IT-Security
Keystroke [70]	Detect imposter keystrokes	20,400	31 mixed types	10	IT-Security
MNIST [71]	Handwritten digits	70,000	28×28 pixel	10	Computer Vision
Luxembourg [72]	Classify internet usage	1901	20 mixed types	2	Social Media
NOAA Rain [73]	Predict rain	18,159	8 mixed types	2	Miscellaneous
Nursery [74]	Rank applications for nursery schools	12,960	8 numerc	5	Miscellaneous
Ozon [75]	Preidct ozone levels	2534	72 mixed types	2	Miscellaneous
Outdoor objects [76]	Recognize objects in garden	4000	21 mixed types	40	Computer Vision
Poker Hand [77]	Hand of five cards	1×10^6	11 mixed types	9	Miscellaneous
Powersupply [69]	Predict hour on basis of power supply	29,928	2 numeric	24	Industry
Rialto Timelaps [78]	Identify buildings	82,250	27 numeric	10	Computer Vision
Sensor [69]	Identify sensor ID	2,219,803	5 numeric	54	Miscellaneous
Usenet [79]	Messages sent in groups	1500	99 attributes	2	Social Media
Spam Assassin Collection [80]	Spam E-Mails	9324	39,917 words	2	Social Media
Weather [57]	Predict occurrence of rain	18,159	8 mixed types	2	Miscellaneous

3.2.2. Datasets

Artificially generated datasets are not the only type of dataset for stream classification. Often, datasets collected in real-world settings are used as well, as observed in Table 3. The underlying classification tasks range from predicting numerical price changes

in the electricity markets of Australia to object recognition by using thousands of tiny images [47,61]. A common practice regarding real-world datasets is the concatenation of static smaller sets to mimic streaming scenarios, i.e., to connect them in a series and create consecutive artificial timestamps. An example of this approach is the famous MNIST dataset, consisting of images of handwritten numbers, which is used across various machine learning domains [62,81]. In Table 3, the name of the dataset, as well as a short description of its content, is given. We referenced the original authors of the dataset, sometimes alongside other researchers who conducted significant enhancements. In addition, the number of records, attributes, assigned classes, and the general discipline from which the dataset was collected are given.

Although today, more data is produced than ever in the past, it is generally criticized that only a few large datasets exist for benchmarking purposes [34]. Additionally, the existing datasets have flaws: for example, refs. [39,82] detect irregularities in the Electricity dataset provided by [63], since it incorporates dependent labels. Prices succumb to long periods of ups and downs. Moreover, delays often occur in real-world problems, and labels are not directly available [83]. Thus, many researchers identified producing more suitable, high-quality datasets as an important future work endeavor [34,35,84–86].

3.3. Benchmarking Frameworks

We define a framework as an open-source application that enables the execution of a stream classification pipeline. Over the years, multiple frameworks have been proposed to help users perform certain parts of the stream classification process, as given in Figure 1. Frameworks facilitate the interaction of different stream classification tasks, enhance interoperability and thus contain all required building blocks for algorithm design [87]. They follow a defined program flow consisting of unmodifiable code. A framework may, for example, include data generators or real-world datasets for benchmarking, algorithms, and metrics that can be computed for monitoring purposes. That is, it should contain all principal parts of our stream classification pipeline. In contrast, a loose or arbitrary combination of different algorithms for a specific application scenario is not considered a framework.

During our literature search, we encountered several works listing software frameworks. A general overview can be found in the work of [5,35,37,88,89]. We excluded frameworks such as Vowpal Wabbit (see www.vowpalwabbit.org (accessed on 5 September 2022)) or RapidMiner (www.rapidminer.com (accessed on 5 September 2022)), since they—although they can handle a large amount of data for batch processing—were not specifically designed for stream classification purposes and did not incorporate, e.g., essential stream classification algorithms. In the studies of [90,91], the authors compare stream processing frameworks that facilitate stream mining tasks but cannot be used for classification.

In the following, we give an overview of relevant frameworks, which are also summarized in Table 4. Below, we provide the framework's name and an overview of which process step of our pipeline can be executed via which framework. Additionally, each framework will be discussed briefly:

The Very Fast Machine Learning toolkit (`VFML`) concentrates on high-speed data stream mining and very large datasets [92]. It incorporates a collection of datasets from the UCI repository. Moreover, it includes functions for feature space simplification. Its core comprises a collection of algorithms: primarily tree-based, *k*-nearest neighbor, and deep neural network approaches. It contains some functions for monitoring, e.g., the development of tree-based algorithms, but has no focus on either evaluation or continuous monitoring.

`Jubatus` is a framework developed during a Japanese research project [93]. The focus lies on distributed processing. Notably, it provides a model-sharing architecture for practical training and collaboration of the classification models. Thus, Jubatus aims to reduce the high networking costs and high latency as inevitable consequences of distributed environments. It includes nineteen test datasets from various sources, e.g., Twitter or malware classification. Further, it incorporates functions for feature space simplification

next to preprocessing algorithms specially designed for textual data, i.e., vectorization and removal of links. All basic stream classification algorithms, such as, for example, Hoeffding trees (see Section 5.2.1), are integrated into the framework [93]. Nevertheless, it contains only essential evaluation and monitoring functionality, such as a confidence value for each assigned class label. In any related project, it can be executed via the analytics engine Spark [94] or the Python sci-kit-learn library.

Table 4. Overview of relevant stream classification frameworks.

	VFML	Jubatus	streamDM	River	MOA
Artificial datasets	✓	✓	✓	✓	✓
Real-world datasets		✓	✓	✓	✓
Preprocessing	✓	✓	✓	✓	✓
Data Segmentation	✓	✓	✓	✓	✓
Labeling				✓	✓
Trees	✓	✓	✓	✓	✓
Neural networks	✓	✓	✓	✓	✓
Neighborhood-based			✓	✓	✓
Frequency-based	✓	✓	✓	✓	✓
Rule-based	✓	✓	✓	✓	✓
SVM			✓	✓	✓
Ensemble		✓	✓	✓	✓
Concept drift detection			✓	✓	✓
Update Mechanisms				✓	✓
Evaluation	✓	✓	✓	✓	✓

The Stream Data Mining library (`streamDM`) developed by Huawei is an extension of Spark [95]. Like Jubatus, it is an extensible and programmable framework focusing on the distributed processing of datasets. The input data stream is divided into processable batches, then forwarded into the Spark engine to generate classification results. Since it has connections to the MOA framework discussed below, it is possible to import all datasets available in MOA. Nevertheless, the authors claim that due to the implementation in C++, it is faster than MOA. streamDM only incorporates different approaches to read the data stream but does not offer explicit preprocessing steps. However, it offers more available learners than Jubatus, such as various stochastic gradient descent approaches, bagging, different Hoeffding trees, and ensemble learners. Likewise, it offers a broader evaluation and monitoring functionality, such as prequential evaluation and metrics based on a contingency table.

`River` is the recent result of the fusion of the sci-kit-multiflow and the creme Python libraries [96]. It grants access to the whole scikit-learn (see www.scikit-learn.org/stable/index.html (accessed on 5 September 2022)) library and the streaming background of the python package creme (see www.pypi.org/project/creme/ (accessed on 5 September 2022)). River includes nearly all benchmark datasets in Table 2 and various preprocessing approaches such as feature selection and normalization. Additionally, numerous classification algorithms and a broad range of monitoring and evaluation methods were incorporated, e.g., for concept drift handling. It is faster than the classical machine learning libraries PyTorch and Tensorflow if one observation at a time must be labeled, but ref. [88] claim that it is slower than MOA, since Java code is expected to run faster than Python code.

Finally, we discuss the most frequently cited framework, Massive Online Analysis (`MOA`) [97]. It merges numerous functionalities of the previously mentioned frameworks, especially related to River and streamDM. It is based on WEKA, the Waikato Environment for Knowledge Analysis framework, a toolkit for batch machine learning algorithms [98]. The framework is written in Java and can be executed via a built-in graphical user interface (GUI) or the command line. Additionally, it holds interfaces to the statistical programming language R (see www.jwijffels.github.io/RMOA/, www.cran.r-project.org/web/packages/streamMOA/index.html (accessed on 5 September 2022)).

MOA was initially developed to facilitate stream classification performance measurement to manage the stream algorithm's speed, memory usage, and accuracy. Thus, the maximum amount of available memory space in MOA is fixed, while the other two dimensions can be adjusted to the user's needs. Currently, its limits are handling up to ten different class labels. It includes many datasets we identified as relevant benchmark datasets in Table 2 and some from Table 3. It holds many preprocessing approaches such as feature selection and discretization. MOA includes many stream classification-related algorithms, such as decision trees and Naïve Bayes classifiers, as well as several evaluation methods such as the prequential holdout. Advanced application scenarios include extensions such as a Tweet-reader to collect Twitter tweets directly, a framework for sentiment analysis, and data reduction techniques on streams without drift. Additionally, it contains all the functionalities of the Scalable Advanced Massive Online Analysis (`SAMOA`) framework, which is no longer maintained but has been integrated into MOA. SAMOA was an Apache project to create a new platform that performs stream mining in a distributed environment. Lastly, different applications and platforms, such as OpenML [32], provide easy access to MOA. For further information, the interested reader is referred to [50].

4. Data Processing

When data is consolidated from different sources, as the first step (see Figure 1), numerous integration steps are required. In this section, possible ways of input segmentation, preprocessing methods, and labeling techniques are discussed. Again, an overview of all the covered subtopics is given in Figure 4.

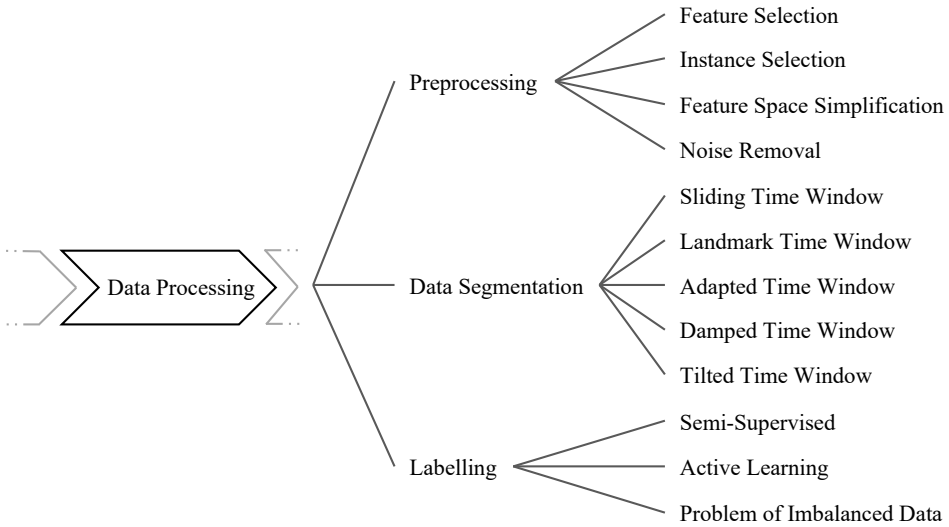

Figure 4. Overview of data preparation methods applicable for data stream classification purposes. Here, the handling of data of the respective *sources* is specified.

4.1. Preprocessing

Data preprocessing is nearly all knowledge discovery processes' first and most critical step. Frequently, this step takes more time than the actual analysis step [99]. Raw data is usually low quality and full of inconsistencies, missing values, noise, or redundancies. This challenge could cause the performance of classification models to fall short of their capabilities. By proper preprocessing, the quality and reliability of automated classification can be improved significantly. Working with stream data is accompanied by specific properties, e.g., that it cannot be stored in its entirety and data points often arrive at high speed. The typical preprocessing steps have to be treated differently than in classical offline settings [99]. Time is a critical component of stream classification problems compared to the offline setting. Typical offline methods benefit efficiency because some approaches can iterate over the data multiple times to identify essential features or align feature spaces for better comparability with enough processing time. Regarding the requirements of online

learning, one iteration should be enough. However, this is not always precisely handled in practice. For example, multiple iterations over a selection of data points are also possible. Nevertheless, not all data points can be stored permanently, and preprocessing steps should be kept in time and short. In addition, the emergence of new classes and features or the change of feature value spaces makes accurate preprocessing difficult. An example is an image or video data, as in surveillance applications or gesture recognition scenarios. Usually, processing pixels is a complex task due to a large data quantity and the fact that only a small amount of pixels captures essential information. The data needs to be reduced before the analysis for efficient computations.

While most summarizing papers in the field of stream data preprocessing cope with sub-aspects, the work of [91,99] give a broad overview of tasks and techniques. The authors elaborate on the current status, such as challenges and research directions. Within their extensive survey, ref. [99] review the mentioned preprocessing algorithms and evaluate them against several synthetic and real-world benchmarking datasets by comparing the accuracy of classification results. As a basis, they use the datasets, as well as generators of the MOA framework. In the following, we will structure typical preprocessing steps according to the work of [99].

4.1.1. Feature Selection

According to [91,99] dimensionality reduction can be achieved by feature selection. Here, methods are divided into filters (applied prior to the actual learning algorithm), wrappers (built upon an evaluation of features), and embedded methods (where the selection is a part of the learning algorithm itself). For stream scenarios, especially the first category is of interest, since the other two approaches are usually too time-intensive or require multiple iterations over data points. Further, filter methods are robust concerning changes in the data streams. Classical filtering methods are, for example, the evaluation of the features based on the information gain or the χ^2 index [99].

A well-known filter approach is the `DXMiner`, which can react to drifts and changes in feature space. The algorithm uses the deviation weight measure to rank features during the classification phase [100]. DXMiner then uses the most discriminative features of the latest labeled instances to classify new, unseen observations. Within their exhaustive benchmarking study, ref. [99] demonstrate that DXMiner is the only algorithm that addresses the problem of concept drifts in stream data. The authors point out that feature selection never improved classification accuracy, but some algorithms yield comparable results with lower model complexity. Again, DXMiner is proposed as the best-performing method here.

4.1.2. Instance Selection

According to [99], the selection of instances is another preprocessing step that usually differs from the traditional offline setting. Selecting appropriate data points is vital for the training phase of an algorithm. Especially after the appearance of a drift in the data stream, it is necessary to select new instances for training and discard data points that belong to outdated concepts in the stream. While drift detection methods are discussed in detail in Section 6, here, the focus is only on the training instance selection methods. The goal is to evaluate which data points should be used for the training, or in other words, which data records represent the current status of the data stream. One way to identify a change in the data stream is to analyze data points regarding their distance to other stream data records. Nearest neighbor methods, for example, bring data points into spatial dependencies. In addition, the temporal component can be considered by sliding a fixed-size window over a data stream or by weighing instances according to their age (see Section 4.2).

An influential work in this context is the `IBL-DS` (Instance-based learning on data streams) algorithm by [101]. The algorithm follows instance-based learning, which means that predictions are made ad hoc based on current data—called the case base—rather than using a previously created model. Thus, the temporal component and possible changes in

the data stream are not disregarded, which is beneficial, especially in scenarios with concept drift. The case base is the pivotal point in this approach. Since not all data points can be stored, the selection must be made carefully. The instances should reflect the characteristics of the current data stream, and the feature space should be represented as balanced as possible. Thus, neither regions are over- or under-represented. Furthermore, only instances that belong to the current concept in the data stream should be included. The approach is implemented via the WEKA interface. It has been evaluated on many synthetic datasets such as STAGGER, Gauss, Sine2, or Hyperplane, as well as on real-world stream datasets (the stream datasets were artificially generated based on static datasets of the UCI package. IBL-DS is evaluated on the car, nursery, and balance dataset). A drawback of nearest neighbor approaches is the computational complexity [99].

Ref. [64] proposed an instance selection method with two levels, called Competence-Based Editing (CBE). The first stage is noise removal and the second cope with redundancy reduction. The authors present the algorithm in the context of spam detection. Within their benchmarking study, ref. [99] points out that CBE is evaluated as the best option, for instance, in model accuracy and reduction.

4.1.3. Feature Space Simplification and Noise Removal

Following the taxonomy of [99], feature space simplification is the third category of data preprocessing methods. An example here is the discretization of feature values. This step is crucial for some classification algorithms (e.g., Naive Bayes) because they only work, or operate better, on discrete data [99]. Ref. [102] present a two-step approach, which thrives on finding the optimal discretization of the data. First, the Partition Incremental Discretization algorithm (PiD) summarizes data and creates preliminary intervals, e.g., of equal width. These intervals are then optimized by splitting whenever the number of elements in an interval is above a predefined threshold. Second, the intervals are merged using a discretization metric, e.g., a minimum description length discretizer. The approach's performance depends not least on the choice of the initial intervals. Another example of feature space simplification is topic modeling in terms of textual data. Social media data is usually relatively short, creating a sparse feature space. This makes traditional NLP methods more error-prone. Ref. [22] perform topic modeling on the sparse text data and then represent the texts based on contained topics rather than just on the words present.

Another challenging task in preprocessing is the step of noise removal. Within their extensive study, ref. [103] elaborate on the different noise scenarios in stream data. They distinguish between stationary noise in evolving data streams, evolving noise in stationary data streams, and evolving noise in evolving data streams. A data stream is named "evolving" when some drift (see Section 6) occurs. In contrast, "evolving noise" means that the noise changes over time, such as, for example, a sensor record when the sensor performance decreases with its lifetime. The authors conclude that there are numerous challenges in noise removal within stream scenarios. At the point of their study, noise is often tackled by building robust classifier models (e.g., [104]) instead of removing noisy data records. Further, the scenarios of evolving noise are not investigated so far [103].

In line with noise removal, the problem of outlier detection is discussed in the literature. Ref. [105] give a structured overview of outlier detection algorithms in the stream data environment. As this is commonly used, they focus on the Local Outlier Factor technique (LOF). The initial idea is that density is measured, i.e., the number of data points located in the immediate environment. Then, the density of one data point is compared to the densities of its neighbors. Data points with a significantly lower density than their neighbors are considered outliers. Ref. [105] conclude that the LOF technique is only partially valuable in stream scenarios. The main drawback is the extensive computational cost and memory usage needed to calculate the distances. To overcome this, they propose the idea of LOF based on a subset of data points. Nevertheless, further work and benchmarking studies should be conducted on outlier detection. Ref. [99] conclude that there is room for improvement in all preprocessing tasks, especially when it comes to high dimensional

and specifically imbalanced or multi-labeled data. Most methods are in the first stage of adaption from the offline setting and have issues regarding efficiency. Future research should focus on proper feature selection methods in stream scenarios with concept drift. This goes especially for reoccurring concepts in the data. The authors pose the need for better algorithms to handle the discretization of features online. Ref. [91] additionally refer to the future challenge of missing values imputation, which is a complex task in data stream mining, as here, relationships among features and observation have to be taken into account.

Next to the extensive work of [91,99], there exists other summarizing work in the field, which, however, is more context-specific. In [106], preprocessing is elaborated for the activity recognition domain. The authors introduce different approaches to extract meaningful sensor data and evaluate the approaches by the CASAS intelligent home dataset.

4.2. Data Segmentation

Since the data stream is possibly unbounded in stream classification, it is often segmented into batches called windows when it comes to the actual processing, e.g., drift detection or updating. An overview of the most often described time window types in our literature base can be observed in Figure 5.

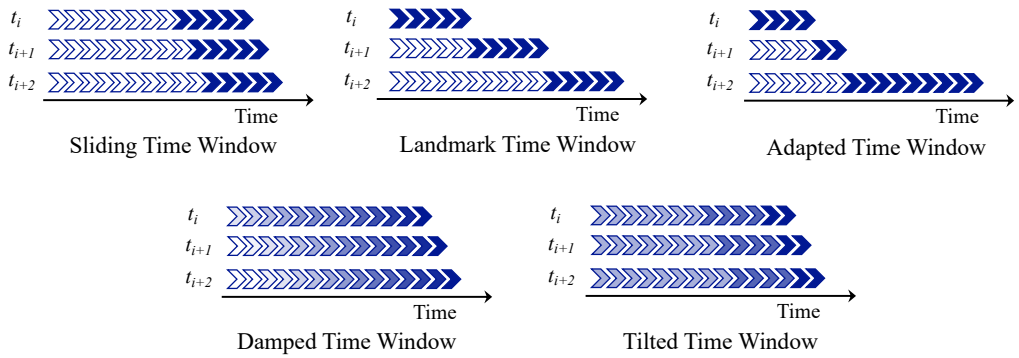

Figure 5. Different ways to segment the input data. The more concentrated the blue, the more are the observations considered in retraining the model.

Several works discuss the different types of time windows. Commonly, sliding, damped, and landmark time windows are distinguished [49,88]. In [89,107] damped time windows are denoted *fading time windows*; however, they refer to the same concepts.

In sliding time windows, the specific window size is fixed so that the algorithm discards the oldest observation from consideration for each new incoming observation. Thus, the same number of observations is always used to retrain the model. Similarly, the landmark time window keeps a steady number of observations, but the data is portioned into disjoint chunks. Whenever a landmark is reached, all observations previously considered are released. Landmarks can be defined differently, e.g., a threshold for the maximum number of observations or a certain amount of time passed. This approach should not be used when it is known a priori that the data stream contains drifts with long periods of stability in between [5]. In such cases, no window size can capture the distribution shift. An alternative is the damped time window. Unlike the approaches discussed, a function is used to decay older observations over time. In literature, usually exponential functions are used to weight observations and generate the decay [108].

Additionally, refs. [89,107,109] present a tilted time window as a variant of the damped time window. It applies different levels of granularity to the data records according to their recency. Thus, old data points are summarized in larger, less important windows, while new observations are considered in smaller, heavier-weighted windows.

Similarly, ref. [5] introduce the adapted time window as a variant of the landmark time window. Here, the window size is not fixed but can be adjusted dynamically. In prac-

tice, adapted time windows may lead to significant data chunk sizes and computational overhead. For example, the Adaptive Windowing algorithm (ADWIN) flushes the considered observations every time a concept drift is detected. Thus, it automatically selects the appropriate window size. The user specifies a sufficiently long time, within ADWIN automatically generates all possible windows and compares them by inspecting the mean value difference [110]. An optimal cut is found when the mean difference exceeds a predefined tolerance level that the user can set. ADWIN forms the basis for many automated update algorithms through its adaptive windows' capability [36].

In the works of [88,111], general advantages and disadvantages of windowing are discussed. Refs. [5,88,112] mention the processing of each incoming observation incrementally as it arrives as a possible way to process incoming data. Ref. [111] points out that by using the windows approach instead of incrementally adding new observations, the expiration of the old observations has to be controlled. Setting the correct window size is crucial for drift detection for all these approaches. However, some types of drifts cannot be detected by some window types. Using different input segmentation methods in parallel [111] is reasonable.

Note here that the denomination of window types are those most commonly used in the literature. However, for different types of stream classification tasks, they can vary. For example, ref. [106] described window types for activity detection with sensors that incorporate the same approaches discussed here but are called differently. E.g., sliding time windows are called time-based windows, while landmark time windows are called sensor-based windows [106].

4.3. Labeling

Another essential step of data processing, which is not part of the actual classification procedure, is labeling data points. While in traditional classification scenarios, a certain amount of data is often available as ground truth initially, the stream classification process thrives on new labeled data points. Two problems go along with this: on the one hand, labeled data are rare (as in the traditional setting), and on the other hand, it can often arrive with an inevitable delay (or never) [113]. An example is the problem of predicting a plane's delay at its departure. The true label—the plane's actual delay—is available hours after the model predicted it. For implementing a stream classification algorithm, labels are often lagging and have to be generated occasionally, either manually or automatically.

An essential strategy in this context is active learning, by which meaningful data points are labeled and then used for training [114]. This approach is suitable for stream data scenarios, as it focuses on only a limited number of instances. Potentially significant data points can be extracted by using instance selection (see Section 4) approaches. Commonly, labeling of new data points is triggered once a concept drift has been detected, and the labeling itself is conducted by experts or in an automated fashion.

However, in practice, instance selection for (manual) data labeling is a very costly and almost infeasible process. Therefore, automated generation of labeled data is currently the focus of research. One approach for automated labeling is semi-supervised learning, which conceptually ranges between supervised and unsupervised learning [41]. Typically, those algorithms attempt to improve performance in one of these two tasks. Regarding classification, semi-supervised techniques can be distinguished between creating classification models (see Section 5.2) and labeling strategies of new data points, which typically require clustering of instances [41]. Here, the idea is that unlabeled data can be used to construct a more accurate classifier under certain assumptions about the data distribution. In general, labeled data is enriched by originally unlabeled records, assuming that data points within a cluster belong to the same class. A prominent example of automated labeling of new instances from a few labeled data is self-organizing maps (also known as Kohonen maps) [115].

Next to the labeling problem itself, another highly discussed topic is the problem of imbalanced data and label sparsity. This topic is often considered in preprocessing

literature [91]. In evolving data streams, classes may be highly unbalanced. As in offline scenarios, methods of under-or oversampling are considered to balance data and enhance prediction performance [91]. An influential work in the field of imbalanced data is the Synthetic Minority Oversampling Technique (SMOTE) proposed by [116]. The algorithm chooses a random example of the minority class and finds the k nearest neighbors. Then, a new data point is created by randomly choosing one of the neighbors and then randomly specifying a point between the latter and the chosen example. Numerous advancements of the initial algorithm exist, which can deal with large amounts of data as standard in the stream scenario [91]. Nevertheless, the problem of imbalanced data is one of the most prominent challenges in stream data analysis [40,91,117,118].

5. Stream Classification Algorithms and Architectures

There are plenty of options for choosing the actual classification algorithm as the core of the classification pipeline (Figure 1). The algorithm needs to handle (potentially unbounded) data streams, where instances might arrive at high speed and can not be stored completely, and predictions need to be made in time [1]. In addition to the stream data-specific requirements, there may be other specific demands on the algorithms. For example, multi-label or multi-class algorithms may be required depending on the scenario. Based on the various application areas (see Section 3), scientists have been engaged in developing and advancing individual stream classification approaches over the years. Moreover, a large number of reviews, as well as benchmarking papers, exist. Based on this literature, we give an overview of stream classification algorithms together with unique features and challenges. Moreover, we summarize the review and benchmarking studies by indicating which algorithm types and structures they cover (see Table 5 and Figure 6 for the structure of the section).

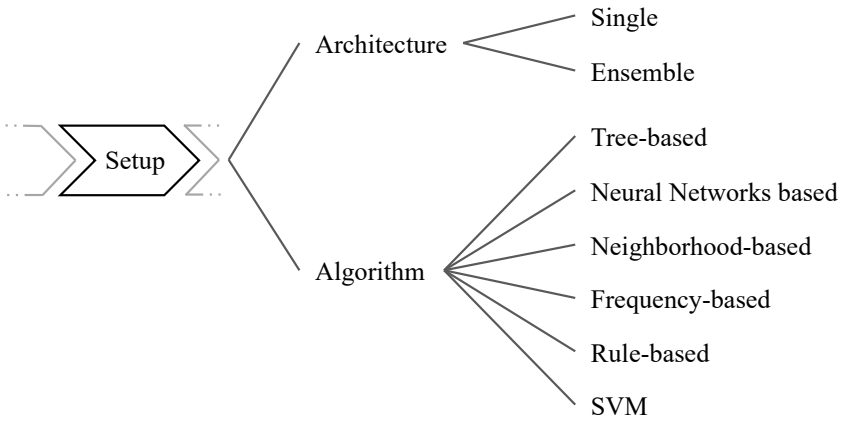

Figure 6. Overview of classification algorithms. Depending on the data sources and initial *processing*, the choice of a suitable algorithm has to made.

5.1. Architecture

Commonly, the classifiers are divided regarding their architecture into single classifiers, or classifier structures, called ensembles [1,89]. To deal with the large amounts of data and the changing data under certain circumstances, combining several models has proven beneficial rather than performing the classification based on a single classification model. A classifier ensemble is a collection of several weak learners, i.e., simple algorithms, whose accuracy is at least slightly above chance. The main focus is on algorithm efficiency, and the basic idea is that combining several weak learners, which only work on a few features, leads to better global classification accuracy. Research on the composition of the ensembles and the combination or weighting of the weak learners is controversial. In their work, Ref. [5] propose different aspects, which should be considered when creating an ensemble.

The combination of classifiers describes how the individual classifiers interact with each other. Several approaches and voting strategies exist for weak learners [5]. A flat

architecture (also found in literature as bagging strategy) describes a setting where weak learners are trained in parallel on different data points and/or features subsets, and their classification results are combined, e.g., via majority votes. Thereby, the most frequently predicted class is assigned to the respective instance. Alternatively, classification can be based on a single selected ensemble classifier or by weighting the different models [119].

Another possibility is to use the results of the weak learner to train a meta learner (called stacking) [5]. Instances formed by the predictions of the weak learners are denoted as meta-dataset. Together with the initial labels of the training data, a meta-classifier can be induced, which is trained on predictions of weak learners as features. This model is then used to predict the label of a new observation.

Table 5. Overview of relevant stream classification algorithm review and benchmarking studies.

	Ensemble	Tree	Neural	Neighbor	Rule	Frequency	SVM	Benchmark
[120]	✓	✓		✓	✓			
[121]	✓	✓		✓	✓	✓	✓	
[111]		✓	✓	✓	✓		✓	
[89]	✓	✓	✓	✓		✓	✓	✓
[122]	✓	✓		✓		✓		✓
[5]	✓							
[37]	✓	✓		✓	✓			
[1]	✓	✓	✓	✓				
[84]	✓							
[123]	✓	✓	✓			✓		✓
[124]	✓							✓
[33]		✓	✓	✓			✓	
[125]			✓					
[13]		✓	✓	✓		✓		✓
[88]	✓	✓	✓	✓		✓		
[7]	✓	✓		✓				✓
[107]				✓				✓

Another important configuration aspect of classifier ensemble approaches is classifier diversity [5]. On the one hand, it is possible to train the individual classifiers on different features (vertical partitioning) or data points (horizontal partitioning). On the other hand, different hyperparameters of classifiers or even different base classifiers on the whole training data can be used. With all these possibilities, the diversity of ensembles is a multifaceted topic. Many studies, strategies, and metrics exist to achieve and monitor the appropriate level of diversity for ensembles [5]. Unlike offline, maintaining diversity and performance in the online setting are considerably more complex. This is especially the case for changing data streams (see Section 6 for more details of changes in data streams). Ref. [84] gives a broad overview of ensemble algorithms in stream classification. They elaborate on the ensemble diversity and combination of base learners. Further, they give an overview of the suitability of algorithms in the case of imbalanced data, novelty class detection, active and semi-supervised learning, and high-dimensional data. The authors define future research directions as handling high dimensional data, the combination of multiple streams, and self-tuning ensembles. However, more work could be conducted to analyze the type of drift and work with imbalanced data and delayed label information. Ref. [126] analyze the impact of changes in the data stream on diversity measures in streaming scenarios. The analyses are based on the MOA framework and the artificial and real-world datasets provided in the MOA framework. This study demonstrates that diversity metrics can be used for stream classification. Further, the monitoring of ensemble diversity over time can be used to elaborate the type of concept drift (see Section 6).

5.2. Algorithms

Various algorithms can be found in the literature suitable for a stream classification problem. In line with the review papers in the field, the most common solutions are based on the ideas of trees, neural networks-, frequency-, and neighborhood-based approaches [13,88]. In addition, Support Vector Machines and rule-based algorithms are fre-

quently mentioned, as they can be adapted to the stream setting [89,111]. In the following, we will shortly give an overview of the most common techniques (see Figure 7).

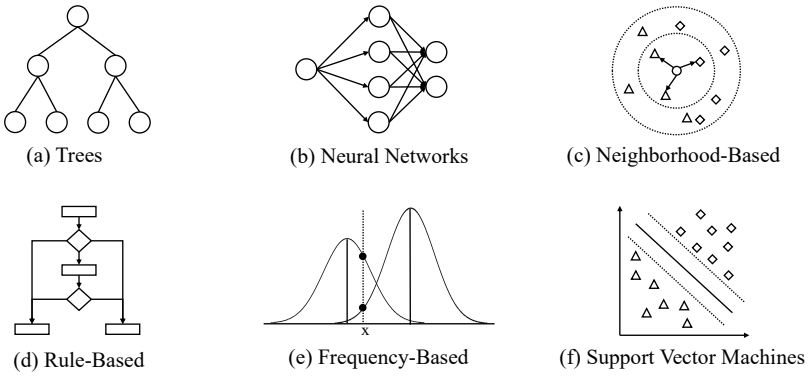

Figure 7. Overview of the most common algorithm types used in the reviewed literature in a stream classification scenario. The point shapes in (**c**) and (**f**) represent data of different classes.

5.2.1. Trees

Tree-based algorithms (see Figure 7a) are suitable for stream classification scenarios because of their low computational costs, their ability to deal with redundant features, and their robustness in terms of noisy data [89]. Hoeffding Trees, as a particular class of decision trees, are one of the most fundamental algorithms in the field [127]. The underlying concept is a very fast decision tree (VFDT) algorithm. With this algorithm, it is not necessary to consider an exhaustive training database but a sufficiently large subset at each splitting point of the tree. Moreover, instances are not stored as such but with the help of counters. Counters enumerate how many instances are classified according to a particular tree branch. Thus, the tree is generated incompletely but with a sufficiently large amount of information. By this, the algorithm perfectly fits the stream scenario, where training data might be incomplete at the beginning, not all instances can be stored, and predictions need to be made in time. For example, within our search, we found the application of Hoeffding trees in intrusion detection [128].

For experimental studies, the SAMOA framework can be used to parallelize the vertical Hoeffding Tree (VHT) to cope with the high data volumes in stream classification scenarios [129]. VHT splits observations into feature subsets (vertical parallelism) of an instance. The algorithm treats all feature subsets as an independent. Thus, computations can be conducted in parallel. Ref. [124] evaluate ensemble methods on MOA real-world datasets. Specifically, they compared ensembles, which are trained online and window-based. They conclude that the adaptive window online ensemble (AWOE), introduced by [130], which operates in a hybrid fashion and includes an internal change detector, produced the best results in terms of accuracy, memory usage, and processing time.

Trees in terms of ensemble classifiers are called forests. The best-known variant—random forests—consists of several randomly generated decision trees. Each tree is trained on a random sample of the data and a subset of features (bagging). The classification results of the trees are merged into a prediction, e.g., via a majority vote, within the combination step.

To cope with the circumstances of the streaming scenario, the Adaptive Random Forest algorithm (ARF) includes a dynamic update method to react to possible changes in the data stream [131]. Each tree is provided with a change detection functionality so that new and more suitable trees can be built in the background at any time. In a large-scale study, the authors show the performance of ARF algorithms on different synthetic and real-world datasets with concept drift. For their experiments, they use the MOA framework. The evaluation criteria within their experiments are classification accuracy, CPU Time, and Memory usage. The authors show that adaptive random forests perform exceptionally well on real-world data sets with label delay. Further, the algorithm is suitable for settings with many features, e.g., in a Natural Language Processing setting such as

spam email detection. Here, the authors demonstrate good results even when using a small number of decision trees. Additionally, within the ARF approach, it is possible to train the trees in parallel without harming the model's classification performance. The algorithm is especially suitable for classification problems, where a subset of features is sufficient to create a decisive explanatory model. A drawback of tree-based classification models is that rebuilding, e.g., changing data, is a time-consuming task [121]. Further, trees tend to over-fit in case they create a high number of branches with only small leafs.

5.2.2. Neural Network

Due to their generalizability and ability to solve non-linear and dynamic decision problems, neural network-based methods (see Figure 7b) are standard in the field [89]. One of the best-known neural network-based methods in the field of stream classification is the extreme learning machine (ELM) [125]. The classical ELM is a learning algorithm conducted on feed-forward neural network(s) with a single hidden layer [132]. The algorithm provides high generalization performance and is very efficient because it does not require gradient-based back-propagation to work. The hidden nodes, as well as their weights and biases, are chosen randomly. Only the weights of the hidden nodes to the output layer must be learned by a least squares method.

Since the invention of the ELM in 2004, it has been used and further developed in many applications. In their review paper, ref. [125] describe the development of the algorithm and show the different works and modifications that have emerged around the basic idea of this efficient and straightforward approach. While the original ELM was designed for batch learning, the Online Sequential ELM (OS-ELM) algorithm can be trained online on single new data points or flexible batch sizes [133]. Based on the OS-ELM, many extensions and solutions have been developed. The individual works deal with the problems of concept drift, unbalanced distribution of classes, and uncertainty of data [125].

Of course, using ELM classifiers as base learners in an ensemble setting is also conceivable [134]. Ref. [134] use the result of an ensemble of ELM models with randomly assigned activation functions to apply a weighted voting strategy for classifying new observations. For a broad overview of the individual approaches and their interrelationships, we recommend the work of [125].

Next to the ELM, other neural network-based approaches were proposed in the literature. Ref. [135] shows a wide-ranging benchmarking study for neural network algorithms on stream data. Concretely, they compare a basic Multi-Layer Perceptron (MLP) model with a Long-Short Term Memory network (LSTM), a Convolutional Neural Network (CNN), and a Temporal Convolutional Network (TCN). The models are evaluated in terms of accuracy and computational efficiency on various UCI repository datasets, including image, sensor, and motion datasets. The authors conclude that CNNs reach the best accuracy by allowing fast processing of the incoming data stream. LSTM and TCN lag behind the capabilities in terms of higher processing time and worse performance.

CNNs are broadly used in high-dimensional data, as in image and video analysis. Intelligent filtering and reduction of the input data make it possible to classify images or detect objects in the images efficiently. Application scenarios are, e.g., gesture recognition procedures, surveillance applications, or the automatic analysis of ultrasound [15,27,136,137].

More complex neural networks are at a disadvantage in the stream setting because most methods cannot make predictions in real-time [88]. Nevertheless, researchers thrive on using state-of-the-art methods' advantages on data streams. A specific example is a generative adversarial network (GAN) proposal by [138]. The deep learning architecture can regenerate training data and overcome data storing limitations. Due to the high computational costs of training and their need for extensive parameter tuning, deep learning methods are not commonly approved in evolving data streams [88]. Next to their high complexity, neural network-based methods are often criticized as black boxes, as the decisions are not interpretable [89].

5.2.3. Neighborhood Based

Neighborhood-based methods (see Figure 7c) are frequently used in the stream classification setting. As one of the most common approaches, the *k*-nearest Neighbor (*k*NN) algorithm predicts labels of new data records by selecting the class label of the closest training instance [121]. For $k > 1$, the class label is defined as the majority of all k nearest neighbors of the instance.

Nearest neighbor classifiers can be naturally transformed to the incremental setting by selecting a limited subset of the most "useful" examples (also called reservoir sampling or lazy learning [1,121]. The sampling process can be biased or unbiased [111]. In biased methods, the time component of data streams is considered. Thus, more recent data points are weighted higher than earlier data records, especially suited for streams with concept drifts. Although the algorithm is intuitive, determining the appropriate k, especially when considering evolving data streams, is not trivial. The ANNCAD algorithm is one of the earliest *k*NN algorithms designed explicitly for data streams [139]. Here, an adaptive choice of the number of k nearest neighbors is possible.

Similar to traditional nearest neighbor approaches, classification can also be conducted based on clusters. Original training data is represented as, e.g., the centroid of supervised microclusters. Thus, the data, as well as computational costs, are reduced [111]. A class consists of one or more micro-clusters, and predictions are derived by measuring the distance of newly arriving instances to the centroids of clusters. Classification of instances based on micro-clusters is also reported by [120] as an on-demand classification. Another common technique, aligned with the one above, is representing data points as density regions in a grid [140,141]. The idea of grid-based clusters is that only counters of the grid regions need to be stored instead of complete observations. Ref. [142] propose an ensemble of k-means classifiers—called Semi-Supervised Adaptive Novel Class Detection and Classification (SAND). Each classifier is trained on a different batch of data. The data of each batch is partially labeled and can be determined dynamically. New labeled instances are gained using records with a high classification confidence score to handle concept drifts. Thus, the classifier is updated. Further, a dense area of outliers is interpreted as a new class. Within their study, the authors compared their algorithms with, e.g., Adaptive Hoeffding Trees in terms of its classification performance and ECSMiner, (ECSMiner is a novel class detection algorithm, proposed by [143].) for the novel class detection component. The performance of SAND in both tasks is at least comparable to the considered algorithms, with only a small number of labels.

The challenges of neighborhood-based approaches are the reduction of processing time, the development of meaningful cluster splitting methods, and the curse of dimensionality [88].

5.2.4. Rule-Based

An intuitive classification approach are rule-based systems [37,120,122]. The idea is that combinations of different features indicate a class label (see Figure 7d). For instance, specific intervals of numerical attributes are associated with a respective class. Thus, traditional rule-based methods cannot be used in stream scenarios with evolving feature spaces or other changes in the data. Exemplary algorithms extending the idea of rule-based systems to the streaming scenario are, e.g., STAGGER, FLORA, and AQ-PM [121]. STAGGER consists of two stages to handle shifts in the distribution: in the first step, the weights of the features, which lead to the applied rules, are adjusted. Second, new features are added to existing rules [53]. FLORA (as well as the extension FLORA3) is based on the idea of temporal windowing [144]. A collection of temporal contexts can be disabled or enabled over time, e.g., to handle seasonal drifts. The AQ-PM approach keeps only the training data close to the rules' decision boundaries [145]. Thus, AQ-PM can learn new concepts by forgetting older examples. However, the major drawbacks of rule-based systems are their inflexibility and low learning speed [121].

5.2.5. Frequency-Based

The Naïve Bayes algorithm (NB) is an incremental algorithm, which is by nature able to handle data streams [121]. Decisions are made based on a frequency scheme (see Figure 7e). Concretely, NB is based on the Bayes Theorem, which assumes that the explanatory variables are independent conditionally of the target variable [146]. Thus, Naïve Bayes works intrinsically incremental by updating relevant entries in its probability table and deriving predictions based on posterior probabilities whenever a new training instance arrives. The algorithm is well-known for its simplicity and can predict class memberships with low computational costs. The traditional algorithm deals with a discretized feature space. However, numerical feature spaces occur often, e.g., in sensor data. Bayesian classification can be combined with an initial discretization step as preprocessing, e.g., via the PiD method [102]. An option of using the algorithm on continuous data arises from the Gaussian Bayes models [123]. Nevertheless, the two main drawbacks of Bayes classifiers are the independence assumption of the features, such as the inability to handle multimodal distributions.

5.2.6. Support Vector Machines

The Support Vector Machine (SVM) [89] is a well-known offline classification algorithm. SVMs are based on finding the maximum marginal hyperplane that separates training data of different classes (see Figure 7f). Finding the hyperplane is a convex optimization problem, which can be solved using the Lagrangian formulation. Using SVM algorithms in big data settings comes with complex calculations, making the traditional SVM unsuitable for the stream scenario. An approach that adapts the idea of the SVM algorithm to the stream classification problem is the Core Vector Machine (CVM) [147]. Here, a minimum enclosing hypersphere is used, which represents the divisive data records. Calculations are then based on the representative hypersphere instead of the original data records, making calculations more efficient. This idea forms the basis for other SVM-based algorithms such as StreamSVM [148]. Here, the CVM is extended so that one pass over the data instances is sufficient. The main drawback of the SVM is the low learning and prediction speed, such as the inability to react to changes in the data stream. Further, SVM models are extremely difficult to interpret [121]. Further, the performance of SVM-based classification models depends on a proper parametrization, e.g., choosing a suitable Kernel [89].

5.3. Specific Classification Problems

Although classification scenarios are traditionally often introduced as binary decision problems, we frequently encounter more than two classes and/or labels, both in practice and in theoretical application scenarios [40]. More precisely, a distinction has to be made between multi-class and multi-label problems.

In the first case, an observation $\vec{x} \in \mathcal{X}$ is classified into an l-dimensional space $\mathcal{Y} = \{C_1, \ldots, C_l\}$ of $l \geq 3$ classes. A common way to deal with multi-class problems is the one-vs-all principle, which is based on l binary classification models [149] (one model per class). Each of them is generated using labeled instances of the respective class of interest on the one hand (1) and the instances of all remaining classes on the other (0). It should be noted that nowadays, various classification algorithms—e.g., neighborhood-based methods, or trees—can deal with more than two classes by nature.

In the case of multi-label problems, instances will not only be assigned one but several classes simultaneously. That is, the feature vector of the i-th observation remains a p-dimensional vector $\vec{x}_i = (x_{i1}, \ldots, x_{ip}) \in \mathcal{X}$. However, extending the previous notation, the corresponding outcome will no longer be a single class label $y_i \in \mathcal{Y}$, but instead a k-dimensional vector $\vec{y}_i = (y_{i1}, \ldots, y_{ik}) \in (\mathcal{Y}_1 \times \ldots \times \mathcal{Y}_k)$, where each element $y_{ij} \in \mathcal{Y}_j$, $j = 1, \ldots, k$, maps to its own finite set $\mathcal{Y}_j$ of l_j class labels.

The following approaches are common when dealing with Multi-Label problems: First, multi-label problems can be separated into several binary problems (called binary relevance), meta labels of label combinations (called label power sets), and a set of label

pairs (called pairwise). These approaches are referred to as problem transformation. The appropriate strategy depends on the data basis. Binary relevance is an easy-to-understand method, which is often implemented in the context of ensemble models. A disadvantage of this method is that no label correlations are considered here, i.e., each label is assigned independent of all other labels. On the other hand, label correlations are considered within the label powerset approach. Here, however, the focus is on essential interactions to counteract the sparsity of data points and achieve a meaningful classification, e.g., by selecting important label correlations [150]. The pairwise approach is rarely used in online settings due to its high computational complexity in large-scale multi-label problems.

Another possibility for dealing with non-binary problems is extending the classification algorithms (such as decision trees). One example is the extension of the Hoeffding Tree algorithm [151]. Here, leaf nodes exist for every necessary label combination in the label powerset.

For a comprehensive overview of multi-label stream classification, we refer to the survey paper by [40]. They discuss a wide range of multi-label stream classification algorithms, their ability to adapt to changes in the data stream, and standard multi-label datasets. As an evaluation criterion, the authors analyzed the algorithm performance in terms of the F1 metric and the running time. By benchmarking multiple algorithms on different real-world and synthetic data sets, they give a good overview of the strengths and weaknesses of the individual approaches. The authors conclude that a Multi-Label ensemble with Adaptive Windowing (MLAW), proposed by, shows the best classification performance and efficiency results. The authors conclude that there is room for further research on handling and detecting different types of concept drift, imbalanced classes, and temporal dependencies of labels.

6. Classifier Maintenance

Using a stream classifier is always aligned with its maintenance. The stream classification model's (hyper-)parameters might have to be adjusted in case of changing data distributions, which are checked by evaluating current classification performance. Thus, maintenance includes the reliable detection of changes (drift detection) and adequate testing and updating of the existing classification model. An overview of the discussed topics is displayed in Figure 8.

6.1. Concept Drift

The topic of changes in streaming data is an integral part of stream classification applications. It should be noted that in addition to the term concept drift, alternative formulations can also be found in the literature. Other terms are, e.g., concept change, distribution, or data shift [36].

Different types of concept drift can be distinguished. Due to the many strands of research, various terminologies have developed over the years. Table 6 lists the work cited in drift detection and the distinctions of concept drifts. Combining the findings within those works, four different concept drift types can be differentiated. Within Figure 9, we present a graphical comparison of them. Within a sudden (also known as abrupt) concept drift, the data distribution changes suddenly at a certain point in time. In other words, the new concept replaces the old one completely. Whereas during an incremental drift, the data distribution changes progressively over a period of time. In contrast to the sudden drift, the old concept vanishes smoothly within an incremental drift. A sudden drift occurs, for example, when a new sensor type is included in an IoT application. An incremental one is comparable to a setting where a sensor becomes slightly more and more inaccurate over time [4]. Some researchers also mention the category of gradual concept drift, where first a new concept appears within the old one for some periods in time [4,152]. At some point, the new concept replaces the old one completely. Please note that the terms sudden/abrupt and gradual are sometimes related to the speed or duration of a drift [153].

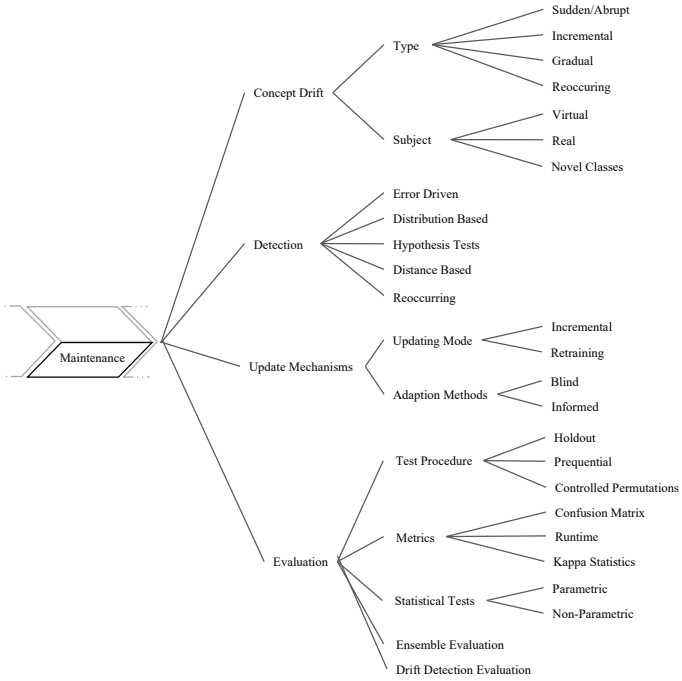

Figure 8. Stream classification model maintenance consits of several building blocks. After an initial setup of the algorithm, the model is maintained, while predictions are made simultaneously.

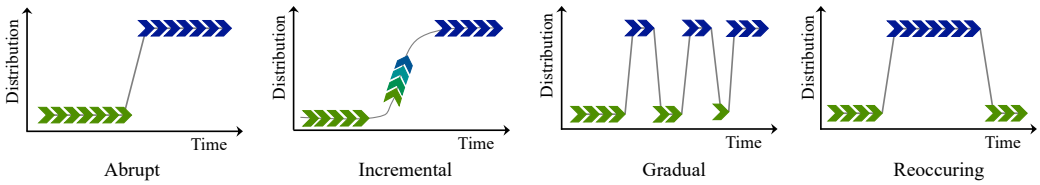

Figure 9. Different types of concept drifts that display possible data distribution changes over time (based on [4,36]).

So-called reoccurring (or seasonal) concept drifts, follow a periodic pattern, where the old and the new concept alternate [4,36].

Researchers also differentiate regarding the subject of the respective drift [153]. The first category is the class changes, called class drift, prior probability shift, or more prominent real drift. Here, Class probabilities $P(\mathcal{Y}|\mathcal{X})$ change over time. The second category is called covariate drift, or virtual concept drift, where the distribution of features $P(X)$ shifts. Of course, a combination of both, where the distribution of features and classes occur, is possible [36].

A particular case of drift is the appearance of novel classes [153]. Within the scenario of the novel or emerging classes, a new and previously unknown class is added to the stream scenario. Sometimes, this is also denoted as a concept evolution [154]. Some works make other distinctions in the context of concept drifts. These, e.g., cover the severity, predictability, and frequency of changes in the data stream [4]. We recommend the work of [153] for the interested reader. The authors discuss the different terminologies extensively and relate them to each other in a broad taxonomy.

6.2. Drift Detection Algorithms

Since concept drifts can occur in various forms, numerous detection approaches exist. Concept drift detectors are either part of the actual classification algorithm itself or run parallel to the classification task and send an alarm to the actual classifier [1]. Summarizing works give an overview of algorithms for concept drift detection [4,36,85,112,118,155–157]. Depending on the specific work, taxonomies and terms, as well as the focus of concept drift

detection algorithms, differ. Table 6 gives a quick overview of the covered aspects of the works. In the following, an overview of the most prominent techniques mentioned in most of the examined papers is given.

Table 6. Overview of summarizing drift detection work and used terminologies of drift types, as well as detection algorithm categories.

Work	Drift Types	Drift Detection Catgeories
[4]	Sudden/abrupt, incremental Gradual, reoccurring	Sequential analysis, control charts, Distribution, contextual
[153]	Real, virtual, abrupt, incremental, Gradual, reoccurring, cyclical, full, sub	
[154]	Sudden, incremental, gradual, Recurring, concept evolution	Offline, online, number of classifiers, Supervised, unsupervised
[155]		Statistical test, error, distribution
[36]	Sudden,gradual, Incremental, reoccurring	Error, distribution, multiple hypothesis
[112]	Real, virtual, class prior, Abrupt, gradual, local, Global, cyclic, acyclic, Predictable, unpredictable	Sequential, window-based
[118]	Sudden,gradual, Incremental, reoccurring	
[85]	Real, virtual	Statistical methods, window-based, Block-based ensembles, incremental
[156]	Local, global	Unsupervised batch, unsupervised online
[34]	Abrupt, gradual, Incremental, real, virtual	Distribution, sequential analysis, Statistical process control

6.2.1. Error Driven

According to [36], most methods are based on the misclassification error rates of the actual classification algorithm. If the error rate increases, the learned algorithmic model no longer satisfyingly fits the data. One of the first error-based methods is the drift detection method (DDM) [47]. (Please note, that [85] categorizes DDM related methods as statistical-based techniques.) The error rates of the classification models are compared based on landmark windows as soon as new labeled data are available. If the error increases significantly, the method triggers the update of the classification model.

Over the years, further developments have been made based on DDM. The extensions address different adjusting screws of the algorithm [36]. For instance, the *Early drift detection method* (EDDM) monitors the distance between correctly classified instances in addition to the error rate in order to detect changes already at the beginning of a drift [158]. However, most of the algorithms listed in [36] and the methods discussed here can only detect the point at which time drift occurs. Please note that, e.g., in [85], the DDM approach is listed under statistical methods-based drift detection.

To detect the critical region of the substantial change in the distribution, an extension of the algorithm based on the Hoeffdings Inequality (HDDM) can be used [159]. Manual specification of window sizes is naturally error-prone. The fuzzy window DDM (FWDDM) uses a fuzzy time window instead of the rigid one to address gradual drifts [160]. Also well-known is the Adaptive Windowing (ADWIN) method proposed by [110] (see Section 4.2). In the summarizing work of [85], the ADWIN algorithm is categorized as a window-based drift detection.

6.2.2. Distribution Based

The second-largest class of drift detection methods is distribution based [36], i.e., the distribution of newly arriving instances is compared to the historical distribution by a suitable distance metric. For comparing two distributions, well-known and often used metrics are, e.g., the Kullback-Leibler or the Jensen-Shannon divergence. An example of the first is the *Information Theoretic Approach* (ITA). Historical and new data records are divided into bins using a k-dimensional tree, and then the frequencies within the bins are compared using the distance metric [161]. Large deviations indicate a change in the data stream. The data-driven approach pursues the idea of detecting drifts directly at the source and does not suffer from model intrinsic errors. A drawback of comparing large datasets is

the high computational complexity [36]. Nevertheless, these algorithms are generally also capable of detecting the severity of a drift. Similar to error-driven approaches, two samples, namely the historical and the new dataset, must be carefully defined. Typically, this is implemented using two sliding windows. Again, choosing the right size is, of course, decisive. For example, recurring concepts must be recognizable in the data and not be ignored by the wrong choice of window size.

6.2.3. Statistical Test Based

Within concept drift detection literature, authors often refer to statistical test-based approaches [36,85,155,157]. Please note that the cited summarizing papers of this section differ in their taxonomies. Within some works, the methods listed previously also fall under statistical methods. For example, ref. [4] refer to statistical tests, which compare two windows of a data stream (see Distribution Based). Ref. [157] count statistical tests as subclasses of distribution and performance (or error)-based approaches. Ref. [85] states that numerous drift detection methods are based on Sequential Probability Ratio Tests. The idea of tests from that category is that if a distribution shifts, then the probability of observing elements of this new Distribution should be higher than the probability of observing elements of the "old" Distribution. An influential approach, following this idea, is the cumulative sum (CUSUM) test [36,85,155]. CUSUM, first proposed by [162], raises the alarm when the mean of the input data—e.g., the prediction error of the classification model (as input data of the CUSUM drift detection method researchers propose the residuals of the Kalman filter (see [36,85,155]))—is significantly different from a threshold value. Ref. [36] refer to multiple hypothesis testing, where parallel and hierarchical testing can be distinguished. In parallel hypothesis testing, several statistical tests are applied simultaneously to the input data stream. The algorithm Linear Four Rate drift detection (LFR) proposed by Heng and Abraham tracks changes in the True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) rate in parallel [163]. For each error rate, a statistical test is applied. The assumption is that error rates remain stable within a reference interval until a change occurs. Considering all kinds of error rates, the method also applies to imbalanced data sets. Additionally, different errors can be weighted higher by corresponding reference intervals.

Hierarchical testing usually follows a two-step approach [36]. First, drifts are detected by the detection layer, and another hypothesis test is applied as a validation layer. The algorithms in the field differ in terms of the used test statistics and detection strategies. Returning to the idea of LFR, hierarchical Linear Four Rate (HLFR) is a recently proposed hierarchical drift detection algorithm. In the first step, the drift detection algorithm LFR is used as the detection layer. When a potential drift is detected, a zero-one loss test (the Zero One loss test gives the proportion of misclassified instances, where the loss of zero indicates no prediction errors) is performed on training data splits to confirm (or deny) the validity of the suspected drift. Therefore, sequentially ordered training data are split into batches. The idea is that the concept drift is confirmed when the prediction loss on the sequentially ordered training data batches deviates significantly from that of the shuffled. If the concept drift is confirmed, the upgrade of the classification model is triggered.

6.2.4. Semi- and Unsupervised

Especially for novel class detection, we find several clustering-based algorithms. Within the OLINDA method, a *k*-means algorithm is executed as soon as incoming data points cannot be assigned to one of the existing clusters of the classes [164]. The produced clusters are then verified. A valid cluster must be cohesive, meaning that instances must meet a similarity threshold. Further, the number of instances within a cluster must exceed a minimum number. Depending on the location of the new cluster, an existing class (macro level) is extended by this cluster (micro-level), or a concept drift is recognized. The MINAS algorithm works similarly. Here, a cluster is connected to an existing class's cluster if the distance of the centroids does not exceed a predefined threshold [165].

Accordingly, a grid-based clustering approach can handle changes in the data stream and emerging classes [166]. The Evolving Micro Clusters (EMC) algorithm determines class memberships by grid-based evolving microclusters. For this method, a grid is applied over the entire data search space. Each observation is assigned to a cell within the grid. New classes can be detected by measuring the density of data points in the grid. Accordingly, grid density-based clustering approaches can be used to detect changes in the input data stream [140]. The grid density information is used to sample the training instances of the actual classifier. When a grid reaches a predefined density threshold, the grid is considered for training. Areas that do not reach this threshold indicate changes in the stream or outliers. Depending on the individual situation, clusters are re-formed, split, or combined when a new dense grid emerges. Following the same idea, the SAND classification algorithm also uses the grid's density to detect new classes [142]. Additionally, the algorithm contains a drift detection method based on the confidence values of the classification algorithm.

6.2.5. Reoccurring Concept Detection

A unique form of concept drift is that of reoccurring concepts. For example, one can think of a measurement series subject to seasonal variations. In this case, standard methods would forget essential information, which may become necessary again later. Therefore, particular interest is given to detecting reoccurring concepts in data streams. The aim is to keep models which have become unusable in reserve for reuse at a later point in time. Conventional for this scenario is the use of classifier ensembles. For example, the Concept Profiling Framework (CPF) uses a collection of different classification models [167]. A drift detection mechanism triggers the archiving of new labeled instances when a threshold for the classification error is reached. Based on this archived set, all existing classifiers are tested. If no suitable model is found, a new model is trained. Likewise, the similarity of the classifiers is evaluated so that the collection remains diverse, and the new model replaces the most previously similar classifier. In order to prevent the erroneous deletion of a classifier, in the extended version, both models initially remain. Only if it could be ensured over an extended period that both represent the same concept is the old model discarded [168]. While Anderson et al. based the drift detection on the model error, the detection of reoccurring concepts could also be conducted by comparing data distributions in the stream (see data-based methods).

Promising research directions are the handling of imbalanced data [118], drift detection in case of missing values [112], as well as the inclusion of temporal dependencies [155]. Ref. [36] states that drift detection methods should be enhanced in terms of their ability to provide information about drift types and regions. Further, ref. [157] see potential in the automated parameter tuning of drift detection and stream classification algorithms.

6.3. Update Mechanisms

A model update follows the evaluation and detection of a change in the data stream to maintain classification accuracy. The following update strategies can be distinguished: it can be differentiated between whether a single algorithm or an ensemble needs to be updated [154]. While in the former case, the single algorithm is adapted along its usage, the update of ensembles is often proceeded by introducing new and deleting old ensemble members. Usually, updating is conducted after external feedback about the actual labels has become available. Labeling is performed either by an expert or automatically, e.g., via an active learning approach on a temporary recurring basis (see Section 4.3). Updating stream classification models is a widely discussed topic. It needs to be timely and efficient and should not hinder the actual use of the classification algorithm. Note here that the data segmentation strategies discussed in Section 4.2 can also be used to determine the point in time and the amount of data used for updating the model. The following sections provide links to the different window types used in various updating modes if necessary (for reference, see Section 4.2).

6.3.1. Updating Mode

Generally, the learning of data stream models can be differentiated between the incremental adaptation of the old, or the complete retraining of new models [4]. In the latter case, new training instances must be collected over time. Like the typical batch learning approach, the old model is discarded and succeeded by a new one, which is similar to the landmark time window approach. Complete retraining is often conducted by models adapted from the traditional batch setting.

In the incremental case, the labeled data points are fed into training one by one. Well-known examples are the VFDT, where new examples are added and updated leaf statistics, and the STAGGER algorithm [53,127].

The update itself can affect the complete model (called global replacement), or just a part of the model (called local replacement) [4]. Whereas decision trees or decision rules can be updated partially, other algorithms, like Naive Bayes, must be retrained as a whole.

6.3.2. Adaptation Methods

The adaptation of the model can be conducted in an informed, or blind manner. Note here that these adaption methods are also called active or passive learning in other works, e.g., by [1]. In the former case, the update happens after a triggering event. This event can be, e.g., the deflection of a change detector or the excess of a threshold (e.g., model error). In blind model adaptation, the update happens without any special trigger event on a recurrent time basis. Incremental classification models are updated blindly by nature since they evolve with data. Again, the VFDT is an exemplary algorithm. Without any strategy to explicitly detect concept drift, the model adapts to the most recent data, e.g., compare Section 4.2 for references on segmentation. It updates itself as soon as a new labeled instance arrives and thus reacts to the current concept without triggers. In this case, it may also make sense to degrade older data records with a suitable fading factor. The idea is to weigh old labeled data points less over time in the training process. Informed (or active) classification methods can be implemented by including a drift detection algorithm.

For more detailed information, several surveys or extensive studies present different aspects of drift detection and update mechanisms in data streams [4,36,154]. The authors structure groundwork algorithms based on their possibilities for drift detection and maintenance. In addition, they discuss various synthetic as well as real-world datasets [36]. Researchers interested in stream data with specific drift characteristics can find a comprehensive overview of relevant datasets in their study.

6.4. Evaluation

Stream classification is an adaptive learning approach. When the data distribution changes, the model has to be adjusted to maintain a high model performance. Thus, thoroughly monitoring, displaying, and interpreting performance measurements over time is a primary concern. The kind of performance measure must be chosen regarding the goal of the learning task and the dataset at hand. More specifically, the following four aspects are discussed: (a) the procedure when and on which data basis the model is tested, (b) the performance metrics used for this purpose, (c) the *statistical test* that can be conducted to compare different performance measurements, (d) evaluation of ensembles, as well as (e), how to evaluate concept drift detection algorithms.

6.4.1. Test Procedure

First of all, users must determine what portion of data they will use for training and testing. In stream classification, it is assumed that the instances appear in fixed sequential intervals and that the corresponding ground truth is directly available. However, standard techniques of the offline classification setting, e.g., cross-validation, do not always meet the requirements of the online setting since they would destroy the temporal order of data [4]. Thus, problem-tailored methods for the streaming scenario evolved.

In many works, e.g., by [4,7,36,84,88,169] three different validation methods are discussed: holdout, prequential evaluation (also named test-then-train), and controlled permutations. Within the holdout strategy, one data subset is selected for training, the rest for testing. In the stream setting, the latter subset should represent the exact distribution currently present in the data. Thus, it can only be applied in artificial settings when the user can define (a) the occurrence of concept drifts and (b) the exact training and testing subsets in advance [88].

Prequential schemata apply an interleaved test-then-train approach: each data record is used to test model performance and afterward to (re-)train the algorithm. Thus, this method maximizes the data's usability. In contrast to holdout, the concept of drifts does not have to be known in advance. However, it may produce biased results, since the data stream is first used for testing and then for training. Controlled permutations overcome this issue. This approach runs multiple test runs with randomized portions of the data stream. Then, the portions are permuted in a sophisticated controlled way to preserve the local data distribution. More precisely, observations initially close concerning time remain close after a permutation. Accordingly, users who expect sudden concept drifts are recommended to use this method [7].

Refs. [83,170], criticize that, in all of these approaches, immediate availability of the labels is assumed. However, in real-world settings, a verification latency may occur. Ref. [170] discusses the example of the prediction of flight delays. A plane's delay is calculated after its departure, i.e., the actual delay is available hours after the model predicted the possible delay. This time can be divided into bins until the ground truth is available. For each bin, for every observation, a prediction is made. Then, for each sub-period of time, a performance metric is calculated. Thus, the change in the performance metric over time can be analyzed. Finally, when the current data instance's accurate label arrives, each bin's performance metric can be compared to the final label, and an overall model evaluation takes place.

6.4.2. Evaluation Metrics

After the (delayed) labels arrive, they can be compared to the model predictions. Metrics can be calculated to assess the performance of the stream classifier. Generally, many works differentiate between three different types of metrics: indicators based on the confusion matrix, kappa statistics, and other criteria related to speed and memory usage [7,36,37,84,88].

As can be observed in the comparative work of [107], metrics based on the confusion matrix, displayed in Table 7, are the most often-used performance measures. As such, commonly used metrics for benchmarking are Accuracy $Acc = (TP + TN)/(TP + FP + FN + TN)$, Recall $Rec = TP/(TP + FN)$, and Precision $Pre = TP/(TP + FP)$ [88,112,131]. Further, the $F1$-score $F1 = 2 \cdot (Pre \cdot Rec)/(Pre + Rec)$ aggregates two of the previous metrics in one score, which is particularly useful in the case of imbalanced data. Next to numerical performance indicators, refs. [36,86] list the Receiver Operating Characteristic (ROC) curve as a standard graphical representation of the trade-off between class assignments. The points spanning the ROC curve depend on the threshold the classifier uses for assigning observations to the positive or negative class. By calculating (and displaying) a convex hull of the points along the ROC curve, the best classifier or threshold can easily be identified as the one with the largest convex hull and thus covered area. Similarly, the trade-off between recall and precision can be displayed in the Precision-Recall curve [86]. ROC and Precision-Recall curves can be summarized in a single value, the Area Under the curve (AUC), called AU-ROC or PR-AUC. To adapt these methods to the streaming context, they are calculated incrementally after the arrival of new observations. For example, ref. [171] presents the Prequential AU-ROC. In contrast to the conventional AU-ROC, it is invariant to changes in the class distribution, as it is incrementally updated.

Table 7. Schematic overview of a binary confusion matrix.

	Actually Positive	**Actually Negative**	
Predicted positive	True positive (TP)	False positive (FP)	$TP + FP$
Predicted negative	False negative (FN)	True negative (TN)	$FN + TN$
	$TP + FN$	$FP + TN$	$TP + FP + FN + TN$

Although the metrics based on the confusion matrix are often used in stream settings, they are also criticized in the literature. For instance, ref. [169] mentions that the classes are frequently imbalanced in the stream setting. A potential approach for processing imbalanced data is aggregating as many fields as possible from the confusion matrix, as it provides a more accurate assessment of the actual classification abilities. Still, ref. [169] introduces the Kappa statistic κ, ranging in the interval $[0, 1]$. In its basic form, the Kappa statistic relates the accuracy of a trained classifier with the accuracy of a random classifier.

A variant of this metric with a dedicated focus on the stream setting is κ_{per}, which can handle data with temporal dependence [83]. In such scenarios—e.g., when investigating the recordings of surveillance data—the chronological order of observations is of particular relevance. In contrast to its classical counterpart, κ_{per} compares the trained classifier's accuracy with that of a persistent classifier, i.e., a classifier that always predicts according to the label of the most recently observed instance. Thus, this metric helps to identify misleading classifier performance over time once these two performances diverge. Related to this line of thought, ref. [84] presents the generalized or combined Kappa statistic by computing the geometric mean of κ and κ_{per}.

Since stream classification deals with a huge amount of data that must be processed and kept in memory, other criteria, such as algorithm speed, memory usage, and CPU capacity utilization, are also relevant for evaluation. Here, especially the RAM-hours are discussed [4,36]. RAM hours indicate the number of gigabytes of RAM deployed per hour, which originally stems from the cost of renting cloud computing services [172]. Ref. [40] presents the running time, which is the time an algorithm needs to run on a dataset, i.e., a predefined number of data records, and is generally measured in seconds. Ref. [88] emphasizes that the duration of all preprocessing steps must also be considered.

Based on our literature search, the methods presented above are conventional in the general stream classification setting. However, other metrics might be relevant for particular purposes or well-defined specific research areas. For evaluating multi-label classification problems, ref. [40] differentiate between example-based and label-based evaluation metrics. Example-based metrics assess the classifier's performance for all instances by combining and averaging the performance metric over the whole dataset, while in the label-based case, the performance for each label is assessed before it is averaged.

Although advancements have been made toward stream classification-specific performance measures, authors still criticize the lack of a holistic performance metric. Current evaluation metrics for stream classification measure individual or local aspects along the stream but do not assess all problem dimensions for the complete stream [36,86,112]. Especially problem settings that complicate the standard settings for stream classification—such as the emergence of novel classes, multi-label classification, or highly imbalanced classes—lack appropriate measures [7,86].

6.4.3. Statistical Tests

Statistical tests can be used to assess the significance of classifier performance and serve as a comparison of different classifiers. Generally, the test can be differentiated in parametric and non-parametric tests. In parametric statistical tests, the sample data are assumed to be derived from a population that can be adequately modeled by a probability distribution with a fixed set of parameters. In a non-parametric model, no explicit mathematical form is assumed for the distribution when modeling the data, but assumptions are made about that distribution, such as continuity or symmetry.

The McNemar test is non-parametric and evaluates whether two given classifiers perform equally well [173]. Specifically, it compares whether the marginal frequencies, i.e., the total row and column values of the confusion matrix, of the predictions of the two classifiers are equal. Similarly, the sign test compares the number of data instances misclassified by the first classifier and correctly classified by the second classifier. A one-sided test is performed to determine which classifier has fewer misclassified data instances based on these differences. Next to these two tests, a Wilcoxon signed-rank test can be used [174]. The difference between the classifier results is calculated and increasingly ordered to build ranks for the test statistic. One classifier outperforms the other if the population ranks differ substantially, since, in this case, the corresponding test statistic exceeds the critical value [36]. The Nemenyi parametric test can be used to compare pairwise performances [175]. The average rank of all classifiers over multiple datasets is calculated and compared to determine the best classifier.

Although these tests are discussed in several stream classification surveys, they are often criticized, since they are inherently static and do not represent the temporal evolution of stream classifier performance [36,84,169]. In consequence, since there has been no advancement in replacing them with more appropriate statistical tests, statistical tests, in general, are not discussed in more recent studies [7,88]. Additionally, no comparative studies exist that evaluate the usefulness and suitability of different tests.

6.4.4. Ensemble Evaluation

In the case of algorithm ensembles, their algorithms need to be assessed on an individual basis and in terms of their contribution to the portfolio. More specifically, most works differentiate between measuring group composition and group performance. The accuracy of single classifiers and classifier differences are crucial for increasing the ensemble's performance as a whole [5,84,126].

Group composition can be measured by evaluating the ensemble's diversity and, thus, complementarity. Intuitively, the diversity of the ensemble measures the heterogeneity of its members, i.e., the benefit which can potentially be obtained by using several classifiers. High classifier heterogeneity leads to uncorrelated votes of the ensemble's members, increasing performance. Nevertheless, note here that no correlation between ensemble diversity and accuracy exists. It might happen that a highly diversified ensemble still performs poorly. The relation between diversity and accuracy in this setting is further discussed by [176].

In many works, the ensemble diversity in the streaming scenario is measured using batch processing metrics [176–178]. For example, ref. [126] calculated standard diversity measures and visualized them over time to meet streaming scenario challenges. Standard measures are, for example, Yule's Q and the disagreement measure D. Both take values in the range of $[-1, 1]$. Yule's Q assesses whether the number of equal predictions of two ensemble members matches the number of divergent predictions. The measure of disagreement D is the ratio of the number of cases where one classifier is correct, and the other is incorrect relative to the total number of cases. Ref. [179] published diversity measures for streaming scenarios. He introduced two new trend diversity measures for data stream classification ensembles, the pair and pool error trend diversity measures. These diversity measures were developed based on the direction and magnitude of changes in the error trends of the base classifiers as they process subsequent samples and cope with stream changes.

In the end, the ensemble's performance is measured with the same metrics as for the single classifier case. In their respective studies, for example, refs. [177,178] used accuracy, the $F1$-score, and the prequential AU-ROC next to other standard evaluation measures to assess ensemble performance. A more detailed overview of these evaluation metrics is given in Section 6.4.2.

6.4.5. Drift Detection Evaluation

If concept drift detectors are incorporated separately in the overall stream classification pipeline, they must also be evaluated. Their performance directly influences the assessment of the model, because only in case of drifts are correctly detected can the classifier make accurate predictions. The principles of assessing classifier and concept drift detector performance are similar: the exactness of the detection and the resilience against false alarms should be taken into account. Thus, the probability of detecting actual changes should be high, while the probability of raising a false alarm should be low. For an overview of requirements for drift detection algorithms, we refer to the works of [4,112,155,180].

Generally, concept drift detectors are tested either on synthetic data or a dataset without concept drift. Drifts must be known in advance to assess whether the detector can detect them accurately or if they do not occur so that the false alarm rate can be tested. Referring to Table 2, all generators could potentially be used for this purpose, since the researcher can determine when and how a drift occurs. Additionally, the RandomRBF by [50] or the Agrawal generator by [54] do not incorporate drifts at all, which thus could be used to test the False Alarm Rate. This rate, as well as the misdetection rate, are discussed by [180]. They measure how often the algorithm falsely assumes a concept drift or a drift that comes unnoticed by the detector.

In contrast to that, refs. [4,155] present measures based on the time until the drift is detected instead of concentrating on error rates. The mean time between false alarms—or also called average run length—measures the expected time between false alarms when no change occurs. The mean time detection measures how fast the detector recognizes a drift after it occurs. Derived from that, the missed detection rate displays the probability of not detecting the drift when it occurred. Additionally, the Mahalanobis distance can be used to measure the span between the moments of change occurrence and change detection [181].

In most of these studies, either newly developed drift detection algorithms are tested all at once, or an overview of current drift detection evaluation metrics is given. A comparative study of these metrics, i.e., in which setting which kind of metrics should be used, unfortunately, has not been conducted so far.

Evaluation of stream classification models is a broad topic and bears significant differences from standard batch processing. The temporal dependence of the data, as well as possible verification latencies, impedes traditional evaluation methods. Which method is used depends on the type of dataset at hand and the computational resources. For example, on the one hand, keeping track of time between drift detection can be a computational overload for some use cases. On the other hand, comparing different metrics may yield a more accurate picture of drift detection performance than just a single metric. Thus, the continuous development of stream-tailored procedures and metrics remains an essential research task.

7. Current and Future Research Directions

We have provided comprehensive insights into the literature regarding the Stream Classification process. For each process step (see Figure 1), the most influential and fundamental works were presented. Next, an overview of current research directions is given by using a global list of identified challenges as a basis. Finally, we will look at recent works and briefly summarize the research directions addressed therein.

7.1. Central List of Future Research Directions

Based on the works presented in Sections 3–6; we identified the following open challenges in data stream classification:

- The lack of appropriate real-world benchmark datasets is a problem in data stream classification, especially in areas where privacy or other regulations are applied, e.g., in health care or when using social media data. A safe and secure way to share the datasets must be identified and developed.

- Most preprocessing methods (e.g., discretization) still are in preliminary stages, as they have issues with efficiency in stream scenarios.
- Dealing with missing data is a challenging task, especially in evolving data streams.
- The combination of several data stream sources, as well as handling high-dimensional data, is in the early stages.
- Imbalanced data is a problem in data stream classification scenarios. This challenge touches on preprocessing tasks such as under- and oversampling and the actual classification and drift detection methods.
- Neural network-based approaches are often inferior in efficiency.
- A unified metric or concept for evaluation of stream classifiers is still missing. Due to the data's time dependency and possible verification latencies, current evaluation measures derived from the batch setting are not suitable for stream classification.
- More research regarding the detection of the type of concept drifts is needed.
- Promising potential lies in automated (hyper-)parameter configuration of classification and drift detection algorithms.

7.2. Current Work in the Field

In this section, we elaborate on current research directions in the field. To this end, we have summarized the topics of all papers from 2020 and 2021 (from our literature database) in a word cloud showing the most prominent words in each title and abstract. We also included articles only available on ArXiv to analyze current research strands. The result can be observed in Figure 10.

Figure 10. Wordcloud of titles and abstracts of papers published in 2020 and 2021.

7.2.1. Data Sources and Benchmarking

Some words such as "real-world", "dataset", and "synthetic", indicate that research is conducted in the field of benchmarking data sets. At the moment, especially the IoT sector and activity recognition problems are prevalent in the literature [9,13,15]. Recent works related to the COVID-19 pandemic have been published, in which stream classification methods are used, e.g., for facial mask detection or the processing of ultrasound images [29,30]. Considering the mentioned examples, it becomes clear that a special interest is observed regarding the issue of privacy protection. Since data sources, such as IoT devices or recordings of individuals, are often based on sensitive user data, this is an additional challenge for stream classification algorithms [182]. In the future, researchers must test modern methods of preserving data privacy, such as decentralized data storage,

for their suitability for stream classification. In decentralized storage, data are not stored in a central location such as a cloud but in individually encrypted blocks [183,184].

A similar problem exists in the social media domain, since social networks restrict data sharing among researchers for the same reasons. Thus, finding benchmark solutions for social media stream data—an image and textual-based—remains a challenge for future works [46].

As shown in Figure 3 as well as Table 3, more numerical data sources are available than image and textual data sources. Generally, the generation and sharing of more real-world benchmark datasets remain an open task for the future. Existing datasets are often too small or have other flaws such as dependent labels [34,86]. Recently, published works incorporate multi-object detection and tracking, as well as multi-stream input [28,185,186]. Thus, more challenging datasets are also required, which reflect the current state of technology, e.g., for processing images of highly sensitive cameras that produce numerous images per second [187].

To incorporate data sources and other parts of the stream classification pipeline, *frameworks* can be used. The literature review demonstrated that few frameworks exist for stream classification purposes, while many more exist for handling stream data but cannot be used for classification. River and MOA are the most comprehensive frameworks under active development. Until now, MOA is the most cited framework in recent literature [13,124,188]. However, River also provides a broad range of stream classification data sets, algorithms, drift detection, and evaluation methods for smaller tasks or situations where researchers want to execute the stream classification pipeline in Python.

7.2.2. Data Processing

Words such as “feature”, “high”, “big”, “label”, and “outlier” demonstrate that researchers work on challenges of (pre-)processing tasks such as outlier detection, labeling, feature selection, and high dimensional data. A current example is feature selection for multi-dimensional time series data. Within [189], relevant features are filtered by calculating their correlation with classes of the decision problem. The authors evaluated the algorithm on six (mainly motion-capturing related) data sets and reported significantly lower computational costs in contrast to other methods in the field. Ref. [190], on the other hand, take the approach of splitting up data streams in a meaningful way and processing them further based on this. Their main contribution is that the date’s origin (here called entity) contains information useful for solving the decision problem. IoT data is split per sensor into so-called sub-streams, for example. The approach also proposes a sampling approach called k random entities, which selects suitable sub-streams for the classification problem. The authors evaluate the approach on three data sets of different domains. They conclude that the approach has potential in both the medical and sensor data fields, but the sub-stream selection part needs further investigation.

The cleaning of highly noisy data streams is another important task for future work. Ref. [191] enhance online stream classification algorithms by using prediction uncertainty and active learning. Conflicting opinions of the classifiers of the ensemble are stored as features, and instances are sent to an expert oracle from time to time. Evaluations of IoT, cloud task and face recognition data sets show a significant increase in classification accuracy and the cross-domain suitability of the algorithm.

7.2.3. Stream Classification Algorithms and Architectures

The word cloud offers words such as “deep”, “neural”, “network”, “ensemble”, and “models”, which indicate that researchers investigate classification algorithm development. Examples are further development on dynamic ensemble selection methods under evolving data streams [192], or adaptions of the *ARF* model [193]. The modified ARF algorithm combines the original adaptive forest approach with the compressed sensing method. By this, the algorithm tackles the problem of high-dimensional data. Compressed sensing is used as an internal preprocessing step and ensures that sparse data is com-

pressed and only a smaller number of crucial and low coherent features is used for further actions. The authors evaluated the approach using the MOA framework and the contained datasets. They also created three Twitter datasets with the MOA Tweet reader to analyze the algorithm's performance, specifically on high-dimensional data. They tested the approach regarding its classification accuracy and performance indicators such as memory utilization and computation time. Using the complete feature information and ARF with the dimension reduction of compressed sensing showed comparable performance with lower computational costs. It outperforms state-of-the-art algorithms such as Hoeffding Adaptive Trees, Self-Adjusting Memory kNN, and Naïve Bayes [193]. Ref. [194] propose an algorithm to handle high-dimensional data with seasonal concept drift. They adapt to the AODE classifier based on the idea of a classical Naïve Bayes, relaxing the independence assumption of features. They include time as a powerful feature that calculates correlation to classes and other features. The authors evaluated the algorithm on two text-based real-world data sets against nine state-of-the-art tree and frequency-based algorithms and achieved consistently better results.

Aligned with the increase in computational resources, various works explore the use of deep neural networks in the context of data stream classification. Of course, CNN's are constantly being developed and improved. For instance, in image analysis, it can be helpful to combine several network structures to process different information sources of an image. For example, ref. [15] uses a two-strand CNN approach, where one network operates on two extracted feature bases that can be obtained from an RGB color-coded image. The authors evaluated their approach to 3D-sign recognition data sets and demonstrated an improvement in classification performance compared to state-of-the-art models in the field.

Ref. [195] introduces a BERT-based deep learning model for streaming NLP tasks. Concretely, their approach is based on an adaptive network that outperforms state-of-the-art approaches, especially regarding imbalanced data distributions. Moreover, ref. [196] introduces a Recurrent Neural Network for astronomical lightcurve classification with a focus on imbalanced data streams. The authors do not apply sampling strategies to the telescope data but modify the algorithm so that the bias towards the majority class is down-weighted. Further, they include additional contextual information—e.g., distance to other galaxies—into the classification setting. They demonstrate that the approach outperforms algorithms without a weighted function and additional contextual information.

7.2.4. Classifier Maintenance

As depicted in Figure 10, a large number of research papers of the last two years include the topic of "concept drift", "detection", or the appearance of "novel" classes. For example, a state-of-the-art approach to finding novel classes is to use neighborhood-based approaches [197]. Concretely, they focus on the cohesiveness and separation index of the Mahalanobis distance. Similarly, ref. [166] base their emerging class detection and classification on microclusters. All data instances not immediately assigned to a close microcluster are clustered with a *k*-means algorithm, and if the clusters are pure, a novel class is detected. Ref. [198] introduces a density-based clustering approach where they monitor sub-cluster regions to detect and adapt to concept drifts and detect novel classes.

Moreover, words related to classifier assessment such as "performance" and "accuracy", as well as "evaluation", can be observed. Assessing the performance of stream classification models is a multi-dimensional problem requiring many steps. Some are already adjusted to the requirements of stream classification, while others still use methods derived from the static setting. First, the training and testing data have to be determined. Prequential evaluation and controlled permutations are state-of-the-art methods for the stream setting, as can be observed in the recent comparative works of [7,88], since they take the volatile nature of stream classification data into account.

In contrast, most works on metrics focus on time-tested measurements derived from the static setting. For example, measures such as accuracy, precision, and recall are used in recent studies, as can be observed in the comparative work of [30,107,199,200]. However,

this kind of metric does not reflect the volatile nature of stream classification data and algorithms. While there are metrics that have been adapted for the stream classification scenario—such as the κ statistics [83,169], no further progress has been made recently. Nevertheless, other relevant metrics such as the duration of all preprocessing steps, RAM hours, or memory usage are used in recent works to assess different dimensions of the stream classifier's performance [88]. However, a unified metric or concept is still missing. The data's time dependency, as well as possible verification latencies, complicate conventional assessment methods. Particularly in the case of problems that complicate the standard classification of data streams—e.g., the emergence of novel classes, multiple label classification, or highly imbalanced classes—appropriate measurement frameworks are still missing, as pointed out in the works of [7,86].

The same issue applies to statistical tests for comparing classifier performances. These tests were developed for comparing classifier results in static settings and are not for stream-specific requirements. Although they are not included in recent surveys, e.g., by [88] or [7], they are still being used in novel studies, such as, for example, by [201,202]. Since they are not developed to test for the variable stream setting and take more than one dimension of the problem into account, they only provide a very limited view of the classifier's performance. Ensemble evaluation methods also mostly rely on batch processing metrics, although advancements for stream purposes are being made. Ref. [179] introduced two new trend diversity measures based on the direction and magnitude of changes in the error trends of classifiers. This measure is used in recent studies, e.g., by [203,204].

Finally, the stream classifier and the drift detection algorithm must be evaluated. Ideally, the algorithm should indicate a data distribution shift at the right point in time in the right direction. A standard drift detection method that is being used in recent studies is, e.g., the False Alarm rate [205]—however, a comparative study as a recommendation on when to use which metric was not conducted until now.

Overall, advancements have been made to adapt classical evaluation procedures and metrics to the stream setting. Nevertheless, especially when considering metrics, more holistic indicators are required to display the actual performance of stream classifiers in more than one dimension.

8. Discussion

Since data stream classification can be applied to a multitude of different application areas, there are many diverse research directions. Although this yields the benefit of using potential synergies with other domains, it also complicates finding standard concepts and terms. Especially in the area of drift detection and the classification algorithms themselves, it is noticeable that different taxonomies exist. This diversity makes it difficult for newcomers to the field to get started. In this work, we give a broad overview of work in the field of data stream classification. Thus, the main contribution of this work is to assist users in this research field by structuring the literature according to the classical stream classification process—the so-called stream classification pipeline. We provide an overview of the most influential works for each process step and link appropriate summaries and benchmarking papers. Finally, we also highlight future research directions, as well as current research in the field. Our literature search encompassed 320 papers in the end.

While it becomes clear from earlier works that models have been adopted from the offline setting, current research directions in the field are increasingly focusing on specific solutions to data stream classification. Here, predominant are the topics of concept drift detection and adaptation, evaluation, and issues around imperfect data, such as class imbalance or missing values. In addition, high-dimensional data and complex classification models such as deep neural networks offer room for improvement. Another research strand that gets almost no attention is the automated (hyper-)parameter configuration of stream classification algorithms. Since the underlying data stream is potentially volatile in real-life, automatically adjusting the configuration would considerably increase the classifiers' quality.

All in all, it can be said that the scenario of stream classification opens a diverse field of research. Today, there is still no common sense in some steps of the pipeline. Therefore, scientists in the field should always keep in mind the different research focuses and statuses of the disciplines.

Author Contributions: Conceptualization, L.C. and J.S.P.; Methodology, L.C. and J.S.P.; Formal Analysis, L.C. and J.S.P.; Data Curation L.C.; Investigation, L.C. and J.S.P.; Writing—Original Draft Preparation, L.C. and J.S.P. and J.B.; Writing—Review and Editing, H.T. and P.K.; Visualization, J.S.P. and L.C.; Supervision, H.T. and P.K.; Project Administration, L.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Project DemoResil (FKZ 005-1709-0001, EFRE-0801431) funded by the German ministry of culture and science and the BMBF-funded project Hybrid (funding ref. 16KIS1531K). The authors acknowledge support from the European Research Center for Information Systems (ERCIS), and the Topical Program "Algorithmization and Social Interaction" of the University of Münster.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Stefanowski, J.; Brzezinski, D. Stream Classification. In *Encyclopedia of Machine Learning and Data Mining*; Springer: Boston, MA, USA, 2017; pp. 1191–1199. [CrossRef]
2. Gracewell, J.J.; Pavalarajan, S. Fall Detection Based on Posture Classification for Smart Home Environment. *J. Ambient Intell. Humaniz. Comput.* **2019**, *12*, 3581–3588. [CrossRef]
3. Zorich, L.; Pichara, K.; Protopapas, P. Streaming Classification of Variable Stars. *Mon. Not. R. Astron. Soc.* **2020**, *492*, 2897–2909. [CrossRef]
4. Gama, J.A.; Žliobait, I.; Bifet, A.; Pechenizkiy, M.; Bouchachia, A. A Survey on Concept Drift Adaptation. *ACM Comput. Surv.* **2014**, *46*, 44. [CrossRef]
5. Gomes, H.M.; Barddal, J.P.; Enembreck, F.; Bifet, A. A Survey on Ensemble Learning for Data Stream Classification. *ACM Comput. Surv. (CSUR)* **2017**, *50*, 23. [CrossRef]
6. Bishop, C.M. *Pattern Recognition and Machine Learning*; Springer: New York, NY, USA, 2006.
7. Din, S.U.; Shao, J.; Kumar, J.; Mawuli, C.B.; Mahmud, S.M.H.; Zhang, W.; Yang, Q. Data Stream Classification with Novel Class Detection: A Review, Comparison and Challenges. *Knowl. Inf. Syst.* **2021**, *63*, 2231–2276. [CrossRef]
8. Mohammadi, M.; Al-Fuqaha, A.; Sorour, S.; Guizani, M. Deep Learning for IoT Big Data and Streaming Analytics: A Survey. *IEEE Commun. Surv. Tutor.* **2018**, *20*, 2923–2960. [CrossRef]
9. Al-Osta, M.; Bali, A.; Gherbi, A. Event Driven and Semantic Based Approach for Data Processing on IoT Gateway Devices. *J. Ambient Intell. Humaniz. Comput.* **2019**, *10*, 4663–4678. [CrossRef]
10. Yu, L.; Gao, Y.; Zhang, Y.; Guo, L. A Framework for Classification of Data Stream Application in Vehicular Network Computing. In Proceedings of the Green Energy and Networking, Dalian, China, 4 May 2019; Jin, J., Li, P., Fan, L., Eds.; Springer: Cham, Switzerland, 2019; pp. 57–67.
11. Grzenda, M.; Kwasiborska, K.; Zaremba, T. Combining Stream Mining and Neural Networks for Short Term Delay Prediction. In Proceedings of the International Joint Conference SOCO'17-CISIS'17-ICEUTE'17, León, Spain, 6–8 September 2017; Springer: Cham, Switzerland, 2017; pp. 188–197.
12. Wang, Q.; Chen, K. Multi-Label Zero-Shot Human Action Recognition Via Joint Latent Ranking Embedding. *Neural Netw.* **2020**, *122*, 1–23. [CrossRef]
13. Khannouz, M.; Glatard, T. A Benchmark of Data Stream Classification for Human Activity Recognition on Connected Objects. *Sensors* **2020**, *20*, 6486. [CrossRef]
14. Singh, T.; Vishwakarma, D. Video Benchmarks of Human Action Datasets: A Review. *Artif. Intell. Rev.* **2019**, *52*, 1107–1154. [CrossRef]
15. Kumar, E.K.; Kishore, P.; Kumar, M.T.K.; Kumar, D.A. 3D Sign Language Recognition with Joint Distance and Angular Coded Color Topographical Descriptor on a 2–Stream CNN. *Neurocomputing* **2020**, *372*, 40–54. [CrossRef]
16. Anjum, A.; Abdullah, T.; Tariq, M.F.; Baltaci, Y.; Antonopoulos, N. Video Stream Analysis in Clouds: An Object Detection and Classification Framework for High Performance Video Analytics. *IEEE Trans. Cloud Comput.* **2019**, *7*, 1152–1167. [CrossRef]
17. Nahar, V.; Li, X.; Zhang, H.L.; Pang, C. Detecting Cyberbullying in Social Networks using Multi-Agent System. *Web Intell. Agent Syst. Int. J.* **2014**, *12*, 375–388. [CrossRef]

18. Tuarob, S.; Tucker, C.S.; Salathe, M.; Ram, N. An Ensemble Heterogeneous Classification Methodology for Discovering Health-Related Knowledge in Social Media Messages. *J. Biomed. Inform.* **2014**, *49*, 255–268. [CrossRef]
19. Burdisso, S.G.; Errecalde, M.; Montes-y Gómez, M. A Text Classification Framework for Simple and Effective Early Depression Detection over Social Media Streams. *Expert Syst. Appl.* **2019**, *133*, 182–197. [CrossRef]
20. Deviatkin, D.; Shelmanov, A.; Larionov, D. Discovering, Classification, and Localization of Emergency Events via Analyzing of Social Network Text Streams. In Proceedings of the International Conference on Data Analytics and Management in Data Intensive Domains, Moscow, Russia, 9–12 October 2018; Springer: Cham, Switzerland, 2018; pp. 180–196.
21. Taninpong, P.; Ngamsuriyaroj, S. Tree-Based Text Stream Clustering with Application to Spam Mail Classification. *Int. J. Data Min. Model. Manag.* **2018**, *10*, 353–370. [CrossRef]
22. Hu, X.; Wang, H.; Li, P. Online Biterm Topic Model Based Short Text Stream Classification Using Short Text Expansion and Concept Drifting Detection. *Pattern Recognit. Lett.* **2018**, *116*, 187–194. [CrossRef]
23. Carrasco-Davis, R.; Cabrera-Vives, G.; Förster, F.; Estévez, P.A.; Huijse, P.; Protopapas, P.; Reyes, I.; Martínez-Palomera, J.; Donoso, C. Deep Learning for Image Sequence Classification of Astronomical Events. *Publ. Astron. Soc. Pac.* **2019**, *131*, 108006. [CrossRef]
24. Lyon, R.; Brooke, J.; Knowles, J.; Stappers, B. A Study on Classification in Imbalanced and Partially-Labelled Data Streams. In Proceedings of the 2013 IEEE International Conference on Systems, Man, and Cybernetics, Manchester, UK, 13–16 October 2013; pp. 1506–1511.
25. Huijse, P.; Estevez, P.A.; Protopapas, P.; Principe, J.C.; Zegers, P. Computational Intelligence Challenges and Applications on Large-Scale Astronomical Time Series Databases. *IEEE Comput. Intell. Mag.* **2014**, *9*, 27–39. [CrossRef]
26. Brandt, M.; Tucker, C.; Kariryaa, A.; Rasmussen, K.; Abel, C.; Small, J.; Chave, J.; Rasmussen, L.; Hiernaux, P.; Diouf, A.; et al. An Unexpectedly Large Count of Trees in the West African Sahara and Sahel. *Nature* **2020**, *587*, 78–82. [CrossRef]
27. Krishnaveni, P.; Sutha, J. Novel Deep Learning Framework for Broadcasting Abnormal Events Obtained From Surveillance Applications. *J. Ambient Intell. Humaniz. Comput.* **2020**, *11*, 4123. [CrossRef]
28. Ali, M.; Ali, R.; Hussain, N. Improved Medical Image Classification Accuracy on Heterogeneous and Imbalanced Data using Multiple Streams Network. *Int. J. Adv. Comput. Sci. Appl.* **2021**, *12*, 617–622. [CrossRef]
29. Ding, Y.; Li, Z.; Yastremsky, D. Real-time Face Mask Detection in Video Data. *arXiv* **2021**, arXiv:2105.01816.
30. Liu, L.; Lei, W.; Wan, X.; Liu, L.; Luo, Y.; Feng, C. Semi-Supervised Active Learning for COVID-19 Lung Ultrasound Multi-symptom Classification. In Proceedings of the 2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI), Baltimore, MD, USA, 9–11 November 2020; pp. 1268–1273. [CrossRef]
31. Sun, J.; Li, H.; Fujita, H.; Fu, B.; Ai, W. Class-Imbalanced Dynamic Financial Distress Prediction Based on Adaboost-SVM Ensemble Combined with SMOTE and Time Weighting. *Inf. Fusion* **2020**, *54*, 128–144. [CrossRef]
32. Vanschoren, J.; van Rijn, J.N.; Bischl, B.; Torgo, L. OpenML: Networked Science in Machine Learning. *SIGKDD Explor. Newsl.* **2014**, *15*, 49–60. [CrossRef]
33. Srivani, B.; Sandhya, N.; Padmaja Rani, B. Literature review and analysis on big data stream classification techniques. *Int. J. Knowl.-Based Intell. Eng. Syst.* **2020**, *24*, 205–215. [CrossRef]
34. Souza, V.M.A.; dos Reis, D.M.; Maletzke, A.G.; Batista, G.E.A.P.A. Challenges in Benchmarking Stream Learning Algorithms with Real-World Data. *Data Min. Knowl. Discov.* **2020**, *34*, 1805–1858. [CrossRef]
35. Gomes, H.M.; Read, J.; Bifet, A.; Barddal, J.P.; Gama, J.a. Machine Learning for Streaming Data: State of the Art, Challenges, and Opportunities. *SIGKDD Explor. Newsl.* **2019**, *21*, 6–22. [CrossRef]
36. Lu, J.; Liu, A.; Dong, F.; Gu, F.; Gama, J.; Zhang, G. Learning Under Concept Drift: A Review. *IEEE Trans. Knowl. Data Eng.* **2018**, *31*, 2346–2363. [CrossRef]
37. Janardan; Mehta, S. Concept drift in Streaming Data Classification: Algorithms, Platforms and Issues. *Procedia Comput. Sci.* **2017**, *122*, 804–811. [CrossRef]
38. Heywood, M. Evolutionary model building under streaming data for classification tasks: Opportunities and challenges. *Genet. Program. Evolvable Mach.* **2014**, *16*, 283–326. [CrossRef]
39. Bifet, A.; Read, J.; Žliobaitė, I.; Pfahringer, B.; Holmes, G. Pitfalls in Benchmarking Data Stream Classification and How to Avoid Them. In Proceedings of the Machine Learning and Knowledge Discovery in Databases, Prague, Czech Republic, 23–27 September 2013; Blockeel, H., Kersting, K., Nijssen, S., Železný, F., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2013; pp. 465–479. [CrossRef]
40. Zheng, X.; Li, P.; Chu, Z.; Hu, X. A Survey on Multi-Label Data Stream Classification. *IEEE Access* **2019**, *8*, 1249–1275. [CrossRef]
41. Engelen, J.; Hoos, H. A survey on semi-supervised learning. *Mach. Learn.* **2020**, *109*, 373–440. [CrossRef]
42. Narasimhamurthy, A.; Kuncheva, L.I. A Framework for Generating Data to Simulate Changing Environments. In Proceedings of the 25th Conference on IASTED International Multi-Conference: Artificial Intelligence and Applications, Innsbruck, Austria, 12–14 February 2007; ACTA Press: Anaheim, CA, USA, 2007; pp. 384–389.
43. Zhao, J.; Jing, X.; Yan, Z.; Pedrycz, W. Network traffic classification for data fusion: A survey. *Inf. Fusion* **2021**, *72*, 22–47. [CrossRef]
44. Tidjon, L.N.; Frappier, M.; Mammar, A. Intrusion Detection Systems: A Cross-Domain Overview. *IEEE Commun. Surv. Tutor.* **2019**, *21*, 3639–3681. [CrossRef]

45. Veit, A.; Matera, T.; Neumann, L.; Matas, J.; Belongie, S. COCO-Text: Dataset and Benchmark for Text Detection and Recognition in Natural Images. *arXiv* **2016**, arXiv:cs.CV/1601.07140.
46. Assenmacher, D.; Weber, D.; Preuss, M.; Calero Valdez, A.; Bradshaw, A.; Ross, B.; Cresci, S.; Trautmann, H.; Neumann, F.; Grimme, C. Benchmarking Crisis in Social Media Analytics: A Solution for the Data Sharing Problem. *Soc. Sci. Comput. Rev. (SSCR) J.* **2021**, *39*. [CrossRef]
47. Gama, J.; Medas, P.; Castillo, G.; Rodrigues, P. Learning with Drift Detection. In *Proceedings of the Brazilian Symposium on Artificial Intelligence*; Springer: Sao Luis, Maranhao, Brazil, 2004; pp. 286–295.
48. Aha, D. Waveform Database Generator Data Set. 2021. Available online: http://archive.ics.uci.edu/ml/datasets/waveform+database+generator+%28version+1%29 (accessed on 5 September 2022).
49. Barddal, J.P.; Murilo Gomes, H.; Enembreck, F. A Survey on Feature Drift Adaptation. In Proceedings of the 27th International Conference on Tools with Artificial Intelligence, Vietri sul Mare, Italy, 9–11 November 2015; Volume 127, pp. 1053–1060. [CrossRef]
50. Bifet, A.; Gavaldà, R.; Holmes, G.; Pfahringer, B. *Machine Learning for Data Streams: With Practical Examples in MOA*; The MIT Press: Cambridge, MA, USA, 2018. [CrossRef]
51. Hulten, G.; Spencer, L.; Domingos, P. Mining Time-Changing Data Streams. In Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 26–29 August 2001; ACM: New York, NY, USA, 2001; pp. 97–106. [CrossRef]
52. Street, W.N.; Kim, Y. A Streaming Ensemble Algorithm (SEA) for Large-Scale Classification. In Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 26–29 August 2001; ACM: New York, NY, USA, 2001; pp. 377–382. [CrossRef]
53. Schlimmer, J.C.; Granger, R.H. Incremental Learning from Noisy Data. *Mach. Learn.* **1986**, *1*, 317–354. [CrossRef]
54. Agrawal, R.; Imielinski, T.; Swami, A. Database Mining: A Performance Perspective. *IEEE Trans. Knowl. Data Eng.* **1993**, *5*, 914–925. [CrossRef]
55. Breiman, L.; Friedman, J.H.; Olshen, R.A.; Stone, C.J. *Classification and Regression Trees*; Brooks/Cole Publishing: Monterey, CA, USA, 1984.
56. Aha, D. LED Display Domain Data Set. 2021. Available online: https://archive.ics.uci.edu/ml/datasets/LED+Display+Domain (accessed on 5 September 2022).
57. Elwell, R.; Polikar, R. Incremental Learning of Concept Drift in Nonstationary Environments. *IEEE Trans. Neural Netw.* **2011**, *22*, 1517–1531. [CrossRef]
58. Kohavi, R. Scaling up the accuracy of naive-Bayes classifiers: A decision-tree hybrid. In Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, Portland, OR, USA, 2–4 August 1996.
59. Data Expo. Airline On-Time Performance. 2018. Available online: http://stat-computing.org/dataexpo/2009/ (accessed on 5 September 2022).
60. Visser, B.; Gouk, H. AWS Spot Pricing Market. 2018. Available online: https://www.openml.org/d/41424 (accessed on 5 September 2022).
61. Krizhevsky, A. *Learning Multiple Layers of Features from Tiny Images*; Technical Report; University of Toronto: Toronto, ON, Canada, 2009.
62. Li, H. CIFAR10-DVS: An event-stream dataset for object classification. *Front. Neurosci.* **2017**, *11*, 309. [CrossRef] [PubMed]
63. Harries, M. *SPLICE-2 Comparative Evaluation: Electricity Pricing*; Technical Report; University of South Wales: South Wales, UK, 1999.
64. Delany, S.J.; Cunningham, P.; Tsymbal, A.; Coyle, L. A case-based technique for tracking concept drift in spam filtering. *Knowl. Based Syst.* **2005**, *18*, 187–195.
65. Katakis, I.; Tsoumakas, G.; Vlahavas, I. Tracking Recurring Contexts Using Ensemble Classifiers: An Application to Email Filtering. *Knowl. Inf. Syst.* **2010**, *22*, 371–391. [CrossRef]
66. Blackard, J.; Dean, D. Comparative Accuracies of Artificial Neural Networks and Discriminant Analysis in Predicting Forest Cover Types from Cartographic Variables. *Comput. Electron. Agric.* **1999**, *24*, 131–151. [CrossRef]
67. Vergara, A.; Vembu, S.; Ayhan, T.; Ryan, M.A.; Homer, M.L.; Huerta, R. Chemical gas sensor drift compensation using classifier ensembles. *Sens. Actuators B Chem.* **2012**, *166–167*, 320–329. [CrossRef]
68. Rodriguez-Lujan, I.; Fonollosa, J.; Vergara, A.; Homer, M.; Huerta, R. On the calibration of sensor arrays for pattern recognition using the minimal number of experiments. *Chemom. Intell. Lab. Syst.* **2014**, *130*, 123–134. [CrossRef]
69. Zhu, X. Stream Data Mining Repository. 2010. Available online: https://www.cse.fau.edu/~xqzhu/stream.html (accessed on 5 September 2022).
70. Killourhy, K.; Maxion, R. Why Did My Detector Do That?! In Proceedings of the Recent Advances in Intrusion Detection, Ottawa, ON, Canada, 15–17 September 2010; Jha, S., Sommer, R., Kreibich, C., Eds.; Springer: Berlin/Heidelberg, Germany, 2010; pp. 256–276.
71. Lecun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-Based Learning Applied to Document Recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [CrossRef]
72. Žliobaitė, I. Combining Similarity in Time and Space for Training Set Formation Under Concept Drift. *Intell. Data Anal.* **2011**, *15*, 589–611. [CrossRef]

73. Ditzler, G.; Polikar, R. Incremental Learning of Concept Drift from Streaming Imbalanced Data. *IEEE Trans. Knowl. Data Eng.* **2013**, *25*, 2283–2301. [CrossRef]
74. Zupan, B.; Bohanec, M.; Bratko, I.; Demsar, J. Machine Learning by Function Decomposition. In Proceedings of the Fourteenth International Conference on Machine Learning; Morgan Kaufmann, Nashville, TN, USA, 8–12 July 1997.
75. Zhang, K.; Fan, W. Forecasting Skewed Biased Stochastic Ozone Days: Analyses, Solutions and Beyond. *Knowl. Inf. Syst.* **2008**, *14*, 299–326. [CrossRef]
76. Losing, V.; Hammer, B.; Wersing, H. Interactive online learning for obstacle classification on a mobile robot. In Proceedings of the International Joint Conference on Neural Networks, Killarney, Ireland, 12–17 July 2015; pp. 1–8. [CrossRef]
77. Cattral, R.; Oppacher, F.; Deugo, D. Supervised and Unsupervised Data Mining with an Evolutionary Algorithm. *Recent Adv. Comput. Comput. Commun.* **2002**, *2*, 296–300. [CrossRef]
78. Losing, V.; Hammer, B.; Wersing, H. KNN Classifier with Self Adjusting Memory for Heterogeneous Concept Drift. In Proceedings of the 2016 IEEE 16th International Conference on Data Mining (ICDM), Barcelona, Spain, 12–15 December 2016; Volume 1, pp. 291–300. [CrossRef]
79. Katakis, I.; Tsoumakas, G.; Vlahavas, I. An Ensemble of Classifiers for coping with Recurring Contexts in Data Streams. In Proceedings of the 18th European Conference Artificial Intelligence, European Coordinating Committee for Artificial Intelligence, Patras, Greece, 21 July 2008; pp. 763–764. [CrossRef]
80. Katakis, I.; Tsoumakas, G.; Vlahavas, I. Dynamic Feature Space and Incremental Feature Selection for the Classification of Textual Data Streams. In Proceedings of the ECML/PKDD-2006 International Workshop on Knowledge Discovery from Data Streams, Berlin, Germany, 18–22 September 2006; Springer: Berlin/Heidelberg, Germany, 2006; p. 107.
81. He, Y.; Sick, B. CLeaR: An adaptive continual learning framework for regression tasks. *AI Perspect* **2021**, *3*, 2. [CrossRef]
82. Zliobaite, I. How good is the Electricity benchmark for evaluating concept drift adaptation. *arXiv* **2013**, arXiv:cs.LG/1301.3524.
83. Žliobaitė, I.; Bifet, A.; Read, J.; Pfahringer, B.; Holmes, G. Evaluation Methods and Decision Theory for Classification of Streaming Data with Temporal Dependence. *Mach. Learn.* **2015**, *98*, 455–482. [CrossRef]
84. Krawczyk, B.; Minku, L.L.; Gama, J.; Stefanowski, J.; Woźniak, M. Ensemble learning for data stream analysis: A survey. *Inf. Fusion* **2017**, *37*, 132–156. [CrossRef]
85. Wares, S.; Isaacs, J.; Elyan, E. Data Stream Mining: Methods and Challenges for Handling Concept Drift. *SN Appl. Sci.* **2019**, *1*, 1412. [CrossRef]
86. Wankhade, K.; Dongre, S.; Jondhale, K. Data stream classification: A review. *Iran J. Comput. Sci.* **2020**, *3*, 239–260. [CrossRef]
87. Gartner IT Glossary. Frameworks. 2021. Available online: https://www.gartner.com/en/information-technology/glossary/framework (accessed on 5 September 2022).
88. Bahri, M.; Bifet, A.; Gama, J.; Gomes, H.M.; Maniu, S. Data stream analysis: Foundations, major tasks and tools. *WIREs Data Min. Knowl. Discov.* **2021**, *11*, e1405. [CrossRef]
89. Nguyen, H.L.; Woon, Y.K.; Ng, W.K. A Survey on Data Stream Clustering and Classification. *Knowl. Inf. Syst.* **2015**, *45*, 535–569. [CrossRef]
90. Inoubli, W.; Aridhi, S.; Mezni, H.; Maddouri, M.; Nguifo, E. A comparative study on streaming frameworks for big data. In Proceedings of the Very Large Data Bases (VLDB), Rio de Janeiro, Brazil, 27–31 August 2018; Springer: Berlin/Heidelberg, Germany, 2018.
91. García, S.; Ramírez-Gallego, S.; Luengo, J.; Benítez, J.M.; Herrera, F. Big data preprocessing: Methods and prospects. *Big Data Anal.* **2016**, *1*, 9. [CrossRef]
92. Hulten, G.; Domingos, P. VFML: Very Fast Machine Learning Toolkit for Mining High-Speed Data Streams. 2004. Available online: https://www.cs.washington.edu/dm/vfml/ (accessed on 5 September 2022).
93. Jubatus Team. Framework and Library for Distributed Online Machine Learning. 2019. Available online: http://jubat.us/en/ (accessed on 5 September 2022).
94. Apache Software Foundation. Apache Spark–Unified Analytics Engine for Big Data. 2021. Available online: https://spark.apache.org (accessed on 5 September 2022).
95. Noah's Ark Lab. streamDM: Data Mining for Spark Streaming. 2016. Available online: http://huawei-noah.github.io/streamDM/ (accessed on 5 September 2022).
96. Montiel, J.; Halford, M.; Mastelini, S.M.; Bolmier, G.; Sourty, R.; Vaysse, R.; Zouitine, A.; Gomes, H.M.; Read, J.; Abdessalem, T.; et al. River: Machine Learning for Streaming Data in Python. *arXiv* **2020**, arXiv:cs.LG/2012.04740.
97. Bifet, A.; Holmes, G.; Pfahringer, B.; Kranen, P.; Kremer, H.; Jansen, T.; Seidl, T. MOA: Massive Online Analysis. A Framework for Stream Classification and Clustering. In Proceedings of the First Workshop on Applications of Pattern Analysis, Windsor, UK, 1–3 September 2010; PMLR: 2010; pp. 44–50.
98. Hall, M.; Frank, E.; Holmes, G.; Pfahringer, B.; Reutemann, P.; Witten, I.H. The WEKA Data Mining Software: An Update. *SIGKDD Explor. Newsl.* **2009**, *11*, 10–18. [CrossRef]
99. Ramírez-Gallego, S.; Krawczyk, B.; García, S.; Woźniak, M.; Herrera, F. A Survey on Data Preprocessing for Data Stream Mining: Current Status and Future Directions. *Neurocomputing* **2017**, *239*, 39–57. [CrossRef]

100. Masud, M.M.; Chen, Q.; Gao, J.; Khan, L.; Han, J.; Thuraisingham, B. Classification and Novel Class Detection of Data Streams in a Dynamic Feature Space. In Proceedings of the Machine Learning and Knowledge Discovery in Databases, Athens, Greece, 5–9 September 2011; Balcázar, J.L., Bonchi, F., Gionis, A., Sebag, M., Eds.; Springer: Berlin/Heidelberg, Germany, 2010; pp. 337–352.
101. Beringer, J.; Hüllermeier, E. Efficient Instance-based Learning on Data Streams. *Intell. Data Anal.* **2007**, *11*, 627–650. [CrossRef]
102. Gama, J.A.; Pinto, C. Discretization from Data Streams: Applications to Histograms and Data Mining. In Proceedings of the 2006 ACM Symposium on Applied Computing, Dijon, France, 23–27 April 2006; ACM: New York, NY, USA, 2006; pp. 662–667. [CrossRef]
103. Prati, R.C.; Luengo, J.; Herrera, F. Emerging topics and challenges of learning from noisy data in nonstandard classification: A survey beyond binary class noise. *Knowl. Inf. Syst.* **2019**, *60*, 63–97. [CrossRef]
104. Sun, B.; Chen, S.; Wang, J.; Chen, H. A Robust Multi-Class AdaBoost Algorithm for Mislabeled Noisy Data. *Knowl.-Based Syst.* **2016**, *102*, 87–102. [CrossRef]
105. Alghushairy, O.; Alsini, R.; Soule, T.; Ma, X. A Review of Local Outlier Factor Algorithms for Outlier Detection in Big Data Streams. *Big Data Cogn. Comput.* **2020**, *5*, 1. [CrossRef]
106. Yala, N.; Fergani, B.; Fleury, A. Towards Improving Feature Extraction and Classification for Activity Recognition on Streaming Data. *J. Ambient Intell. Humaniz. Comput.* **2017**, *8*, 177–189. [CrossRef]
107. Tieppo, E.; Santos, R.R.d.; Barddal, J.P.; Nievola, J.C. Hierarchical classification of data streams: A systematic literature review. *Artif. Intell. Rev.* **2021**, *54*, 1–40. [CrossRef]
108. Zhu, Y.; Shasha, D. StatStream: Statistical Monitoring of Thousands of Data Streams in Real Time. In *Proceedings of the 28th International Conference on Very Large Databases*; Bernstein, P.A., Ioannidis, Y.E., Ramakrishnan, R., Papadias, D., Eds.; Morgan Kaufmann: San Francisco, CA, USA, 2002; Chapter 32, pp. 358–369. [CrossRef]
109. Ng, W.; Dash, M. Discovery of Frequent Patterns in Transactional Data Streams. In *Transactions on Large-Scale Data- and Knowledge-Centered Systems II*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 1–30. [CrossRef]
110. Bifet, A.; Gavalda, R. Learning from Time-Changing Data with Adaptive Windowing. In *Proceedings of the 2007 SIAM International Conference on Data Mining*; Society for Industrial and Applied Mathematics (SIAM): Philadelphia, PA, USA, 2007; pp. 443–448.
111. Aggarwal, C.C. A Survey of Stream Classification Algorithms. In *Data Classification: Algorithms and Applications*; Charu, C., Aggarwal, V.K., Eds.; CRC Press: New York, NY, USA, 2014; Chapter 9, pp. 245–274.
112. Khamassi, I.; Sayed Mouchaweh, M.; Hammami, M.; Ghédira, K. Discussion and review on evolving data streams and concept drift adapting. *Evol. Syst.* **2018**, *9*, 1–23. [CrossRef]
113. Masud, M.M.; Woolam, C.; Gao, J.; Khan, L.; Han, J.; Hamlen, K.W.; Oza, N.C. Facing the Reality of Data Stream Classification: Coping with Scarcity of Labeled Data. *Knowl. Inf. Syst.* **2012**, *33*, 213–244. [CrossRef]
114. Žliobaitė, I.; Bifet, A.; Pfahringer, B.; Holmes, G. Active Learning with Drifting Streaming Data. *IEEE Trans. Neural Netw. Learn. Syst.* **2013**, *25*, 27–39. [CrossRef] [PubMed]
115. Arabmakki, E.; Kantardzic, M. SOM-Based Partial Labeling of Imbalanced Data Stream. *Neurocomputing* **2017**, *262*, 120–133. [CrossRef]
116. Chawla, N.V.; Bowyer, K.W.; Hall, L.O.; Kegelmeyer, W.P. SMOTE: Synthetic Minority Over-sampling Technique. *J. Artif. Intell. Res.* **2002**, *16*, 321–357. [CrossRef]
117. Krawczyk, B.; Stefanowski, J.; Wozniak, M. Data Stream Classification and Big Data Analytics. *Neurocomputing* **2015**, *150*, 238–239. [CrossRef]
118. Iwashita, A.S.; Papa, J.P. An Overview on Concept Drift Learning. *IEEE Access* **2019**, *7*, 1532–1547. [CrossRef]
119. Pan, S.; Zhang, Y.; Li, X. Dynamic Classifier Ensemble for Positive Unlabeled Text Stream Classification. *Knowl. Inf. Syst.* **2012**, *33*, 267–287. [CrossRef]
120. Gaber, M.M.; Zaslavsky, A.; Krishnaswamy, S. A Survey of Classification Methods in Data Streams. In *Data Streams*; Advances in Database Systems; Aggarwal, C.C., Ed.; Springer: Boston, MA, USA, 2007; Volume 31, pp. 39–59. [CrossRef]
121. Lemaire, V.; Salperwyck, C.; Bondu, A. A Survey on Supervised Classification on Data Streams. *Bus. Intell.* **2014**, *4*, 88–125.
122. Barddal, J.P.; Gomes, H.M.; de Souza Britto, A.; Enembreck, F. A benchmark of classifiers on feature drifting data streams. In Proceedings of the 2016 23rd International Conference on Pattern Recognition (ICPR), Cancun, Mexico, 4–8 December 2016; pp. 2180–2185. [CrossRef]
123. Losing, V.; Hammer, B.; Wersing, H. Incremental on-line learning: A review and comparison of state of the art algorithms. *Neurocomputing* **2018**, *275*, 1261–1274. [CrossRef]
124. Nagendran, N.; Sultana, H.P.; Sarkar, A. A Comparative Analysis on Ensemble Classifiers for Concept Drifting Data Streams. In *Soft Computing and Medical Bioinformatics*; SpringerBriefs in Applied Sciences and Technology; Springer: Singapore, 2019; pp. 55–62. [CrossRef]
125. Li, L.; Sun, R.; Cai, S.; Zhao, K.; Zhang, Q. A Review of Improved Extreme Learning Machine Methods for Data Stream Classification. *Multimed. Tools Appl.* **2019**, *78*, 33375–33400. [CrossRef]
126. Brzezinski, D.; Stefanowski, J. Ensemble Diversity in Evolving Data Streams. In Proceedings of the International Conference on Discovery Science, Bari, Italy, 19–21 October 2016; Springer: Cham, Switzerland, 2016; pp. 229–244.
127. Domingos, P.; Hulten, G. Mining High-Speed Data Streams. In Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Boston, MA, USA, 20–23 August 2000; ACM: New York, NY, USA, 2000; pp. 71–80.

128. Yin, C.; Feng, L.; Ma, L. An Improved Hoeffding-ID Data-Stream Classification Algorithm. *J. Supercomput.* **2016**, *72*, 2670–2681. [CrossRef]
129. Kourtellis, N.; Morales, G.D.F.; Bifet, A.; Murdopo, A. VHT: Vertical Hoeffding Tree. In Proceedings of the International Conference on Big Data, Washington, DC, USA, 5–8 December 2016; pp. 915–922.
130. Sun, Y.; Wang, Z.; Liu, H.; Du, C.; Yuan, J. Online Ensemble Using Adaptive Windowing for Data Streams with Concept Drift. *Int. J. Distrib. Sens. Netw.* **2016**, *12*, 4218973. [CrossRef]
131. Gomes, H.M.; Bifet, A.; Read, J.; Barddal, J.P.; Enembreck, F.; Pfharinger, B.; Holmes, G.; Abdessalem, T. Adaptive Random Forests for Evolving Data Stream Classification. *Mach. Learn.* **2017**, *106*, 1469–1495. [CrossRef]
132. Huang, G.B.; Zhu, Q.Y.; Siew, C.K. Extreme Learning Machine: A New Learning Scheme of Feedforward Neural Networks. In Proceedings of the International Joint Conference on Neural Networks, Budapest, Hungary, 25–29 July 2004; Volume 2, pp. 985–990.
133. Liang, N.Y.; Huang, G.B.; Saratchandran, P.; Sundararajan, N. A Fast and Accurate Online Sequential Learning Algorithm for Feedforward Networks. *IEEE Trans. Neural Netw.* **2006**, *17*, 1411–1423. [CrossRef]
134. Xu, S.; Wang, J. A Fast Incremental Extreme Learning Machine Algorithm for Data Streams Classification. *Expert Syst. Appl.* **2016**, *65*, 332–344. [CrossRef]
135. Lara-Benítez, P.; Carranza-García, M.; Martínez-Álvarez, F.; Santos, J.C.R. On the Performance of Deep Learning Models for Time Series Classification in Streaming. In Proceedings of the 15th International Conference on Soft Computing Models in Industrial and Environmental Applications, Burgos, Spain, 16–18 September 2020; Springer: Cham, Switzerland, 2020; pp. 144–154.
136. Elboushaki, A.; Hannane, R.; Afdel, K.; Koutti, L. xMultiD-CNN: A Multi-Dimensional Feature Learning Approach Based on Deep Convolutional Networks for Gesture Recognition in RGB-D Image Sequences. *Expert Syst. Appl.* **2020**, *139*, 112829. [CrossRef]
137. Lin, Z.; Li, S.; Ni, D.; Liao, Y.; Wen, H.; Du, J.; Chen, S.; Wang, T.; Lei, B. Multi-Task Learning for Quality Assessment of Fetal Head Ultrasound Images. *Med. Image Anal.* **2019**, *58*, 101548. [CrossRef]
138. Besedin, A.; Blanchart, P.; Crucianu, M.; Ferecatu, M. Deep Online Classification Using Pseudo-Generative Models. *Comput. Vis. Image Underst.* **2020**, *201*, 103048. [CrossRef]
139. Law, Y.N.; Zaniolo, C. An Adaptive Nearest Neighbor Classification Algorithm for Data Streams. In Proceedings of the European Conference on Principles of Data Mining and Knowledge Discovery, Porto, Portugal, 3–7 October 2005; Jorge, A.M., Torgo, L., Brazdil, P., Camacho, R., Gama, J., Eds.; Springer: Berlin/Heidelberg, Germany, 2005; pp. 108–120. [CrossRef]
140. Sethi, T.S.; Kantardzic, M.; Hu, H. A Grid Density Based Framework for Classifying Streaming Data in the Presence of Concept Drift. *J. Intell. Inf. Syst.* **2016**, *46*, 179–211. [CrossRef]
141. Tennant, M.; Stahl, F.; Rana, O.; Gomes, J.B. Scalable Real-Time Classification of Data Streams with Concept Drift. *Future Gener. Comput. Syst.* **2017**, *75*, 187–199. [CrossRef]
142. Haque, A.; Khan, L.; Baron, M. SAND: Semi-Supervised Adaptive Novel Class Detection and Classification over Data Stream. In Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 February 2016; pp. 1652–1658.
143. Masud, M.M.; Gao, J.; Khan, L.; Han, J.; Thuraisingham, B. Classification and Novel Class Detection in Data Streams with Active Mining. In Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining, Hyderabad, India, 21–24 June 2010; Springer: Berlin/Heidelberg, Germany, 2010; pp. 311–324.
144. Widmer, G.; Kubat, M. Learning in the presence of concept drift and hidden contexts. *Mach. Learn.* **1996**, *23*, 69–101. [CrossRef]
145. Maloof, M.A.; Michalski, R.S. Selecting examples for partial memory learning. *Mach. Learn.* **2000**, *41*, 27–52. [CrossRef]
146. Bayes, T. LII. An essay towards solving a problem in the doctrine of chances. By the late Rev. Mr. Bayes, FRS communicated by Mr. Price, in a letter to John Canton, AMFR S. *Philos. Trans. R. Soc. Lond.* **1763**, *53*, 370–418.
147. Tsang, I.W.; Kocsor, A.; Kwok, J.T. Simpler Core Vector Machines with Enclosing Balls. In Proceedings of the 24th International Conference on Machine Learning, Corvalis, OR, USA, 20–24 June 2007; ACM: New York, NY, USA, 2007; pp. 911–918. [CrossRef]
148. Rai, P.; Daumé, H.; Venkatasubramanian, S. Streamed Learning: One-Pass SVMs. In Proceedings of the 21st International Jont Conference on Artifical Intelligence, Pasadena, CA, USA, 11–17 July 2009; Morgan Kaufmann Publishers Inc.: San Francisco, CA, USA, 2009; pp. 1211–1216. [CrossRef]
149. Hashemi, S.; Yang, Y.; Mirzamomen, Z.; Kangavari, M. Adapted One-Versus-All Decision Trees for Data Stream Classification. *IEEE Trans. Knowl. Data Eng.* **2008**, *21*, 624–637. [CrossRef]
150. Read, J.; Pfahringer, B.; Holmes, G. Multi-Label Classification Using Ensembles of Pruned Sets. In Proceedings of the 2008 Eighth IEEE International Conference on Data Mining, Pisa, Italy, 15–19 December 2008; pp. 995–1000.
151. Read, J.; Bifet, A.; Holmes, G.; Pfahringer, B. Scalable and Efficient Multi-Label Classification for Evolving Data Streams. *Mach. Learn.* **2012**, *88*, 243–272. [CrossRef]
152. Lu, J.; Yang, Y.; Webb, G.I. Incremental discretization for naïve-bayes classifier. In Proceedings of the International Conference on Advanced Data Mining and Applications, Xi'an, China, 14–16 August 2006; Li, X., Zaïane, O.R., Li, Z., Eds.; Springer: Berlin/Heidelberg, Germany, 2006; pp. 223–238. [CrossRef]
153. Webb, G.I.; Hyde, R.; Cao, H.; Nguyen, H.L.; Petitjean, F. Characterizing Concept Drift. *Data Min. Knowl. Discov.* **2016**, *30*, 964–994. [CrossRef]
154. Faria, E.R.; Goncalves, I.J.; de Carvalho, A.C.; Gama, J. Novelty Detection in Data Streams. *Artif. Intell. Rev.* **2016**, *45*, 235–269. [CrossRef]

155. Bifet, A. Classifier Concept Drift Detection and the Illusion of Progress. In Proceedings of the International Conference on Artificial Intelligence and Soft Computing, Zakopane, Poland, 11–15 June 2017; Springer: Cham, Switzerland, 2017; pp. 715–725.
156. Gemaque, R.N.; Costa, A.F.J.; Giusti, R.; Santos, E.M. An overview of unsupervised drift detection methods. *WIREs Data Min. Knowl. Discov.* **2020**, *10*, e1381. [CrossRef]
157. Hu, H.; Kantardzic, M.; Sethi, T.S. No Free Lunch Theorem for concept drift detection in streaming data classification: A review. *WIREs Data Min. Knowl. Discov.* **2020**, *10*, e1327. [CrossRef]
158. Baena-Garcıa, M.; del Campo-Ávila, J.; Fidalgo, R.; Bifet, A.; Gavalda, R.; Morales-Bueno, R. Early Drift Detection Method. In Proceedings of the Fourth International Workshop on Knowledge Discovery from Data Streams, Philadelphia, PA, USA, 20 August 2006; ACM: New York, NY, USA, 2006; Volume 6, pp. 77–86.
159. Frias-Blanco, I.; del Campo-Ávila, J.; Ramos-Jimenez, G.; Morales-Bueno, R.; Ortiz-Diaz, A.; Caballero-Mota, Y. Online and Non-Parametric Drift Detection Methods Based on Hoeffding's Bounds. *IEEE Trans. Knowl. Data Eng.* **2014**, *27*, 810–823. [CrossRef]
160. Liu, A.; Zhang, G.; Lu, J. Fuzzy Time Windowing for Gradual Concept Drift Adaptation. In Proceedings of the IEEE International Conference on Fuzzy Systems, Naples, Italy, 9–12 July 2017; pp. 1–6.
161. Dasu, T.; Krishnan, S.; Venkatasubramanian, S.; Yi, K. An Information-Theoretic Approach to Detecting Changes in Multi-Dimensional Data Streams. In Proceedings of the Symposium on the Interface of Statistics, Computing Science, and Applications, Pasadena, CA, USA, 24–27 May 2006; American Statistical Association: New York, NY, USA, 2006.
162. Page, E.S. Continuous inspection schemes. *Biometrika* **1954**, *41*, 100–115. [CrossRef]
163. Wang, H.; Abraham, Z. Concept Drift Detection for Streaming Data. In Proceedings of the International Joint Conference on Neural Networks, Killarney, Ireland, 12–17 July 2015; pp. 1–9. [CrossRef]
164. Spinosa, E.J.; de Carvalho, A.P.d.L.F.; Gama, J. Novelty Detection with Application to Data Streams. *Intell. Data Anal.* **2009**, *13*, 405–422. [CrossRef]
165. Faria, E.R.; Gama, J.; Carvalho, A.C. Novelty Detection Algorithm for Data Streams Multi-Class Problems. In Proceedings of the 28th Annual ACM Symposium on Applied Computing, Coimbra, Portugal, 18–22 March 2013; ACM: New York, NY, USA, 2013; pp. 795–800.
166. Din, S.U.; Shao, J. Exploiting Evolving Micro-Clusters for Data Stream Classification with Emerging Class Detection. *Inf. Sci.* **2020**, *507*, 404–420. [CrossRef]
167. Anderson, R.; Koh, Y.S.; Dobbie, G. CPF: Concept Profiling Framework for Recurring Drifts in Data Streams. In Proceedings of the Australasian Joint Conference on Artificial Intelligence, Hobart, TAS, Australia, 5–8 December 2016; Springer: Berlin/Heidelberg, Germany, 2016; pp. 203–214.
168. Anderson, R.; Koh, Y.S.; Dobbie, G.; Bifet, A. Recurring Concept Meta-Learning for Evolving Data Streams. *Expert Syst. Appl.* **2019**, *138*, 112832. [CrossRef]
169. Bifet, A.; de Francisci Morales, G.; Read, J.; Holmes, G.; Pfahringer, B. Efficient Online Evaluation of Big Data Stream Classifiers. In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, 10–13 August 2015; ACM: New York, NY, USA, 2015; pp. 59–68. [CrossRef]
170. Grzenda, M.; Gomes, H.M.; Bifet, A. Delayed labelling evaluation for data streams. *Data Min. Knowl. Discov.* **2020**, *34*, 1237–1266. [CrossRef]
171. Brzezinski, D.; Stefanowski, J. Prequential AUC for Classifier Evaluation and Drift Detection in Evolving Data Streams. In Proceedings of the 3rd International Conference on New Frontiers in Mining Complex Patterns, Nancy, France, 19 September 2014; Springer: Berlin/Heidelberg, Germany, 2014; pp. 87–101. [CrossRef]
172. Bifet, A.; Holmes, G.; Pfahringer, B.; Frank, E. Fast Perceptron Decision Tree Learning from Evolving Data Streams. In Proceedings of the 14th Pacific-Asia Conference on Knowledge Discovery and Data Mining, Hyderabad, India, 21–24 June 2010; Springer: Berlin/Heidelberg, Germany, 2010; pp. 299–310. [CrossRef]
173. McNemar, Q. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika* **1947**, *12*, 153–157. [CrossRef]
174. Wilcoxon, F. Individual Comparisons by Ranking Methods. *Biom. Bull.* **1945**, *1*, 80–83. [CrossRef]
175. Nemenyi, P. Distribution-Free Multiple Comparisons. Ph.D. Thesis, Princeton University, Princeton, NJ, USA, 1663.
176. Bonab, H.; Can, F. Less Is More: A Comprehensive Framework for the Number of Components of Ensemble Classifiers. *IEEE Trans. Neural Netw. Learn. Syst.* **2019**, *30*, 2735–2745. [CrossRef]
177. Sidhu, P.; Bhatia, M.P.S. A Novel Online Ensemble Approach to Handle Concept Drifting Data Streams: Diversified Dynamic Weighted Majority. *Int. J. Mach. Learn. Cybern.* **2018**, *9*, 37–61. [CrossRef]
178. Büyükcakir, A.; Bonab, H.; Can, F. A Novel Online Stacked Ensemble for Multi-Label Stream Classification. In Proceedings of the 27th ACM International Conference on Information and Knowledge Management, Torino, Italy, 22–26 October 2018; ACM: New York, NY, USA, 2018; pp. 1063–1072.
179. Jackowski, K. New Diversity Measure for Data Stream Classification Ensembles. *Eng. Appl. Artif. Intell.* **2018**, *74*, 23–34. [CrossRef]
180. Goncalves, P.M.; de Carvalho Santos, S.G.; Barros, R.S.; Vieira, D.C. A Comparative Study on Concept Drift Detectors. *Expert Syst. Appl.* **2014**, *41*, 8144–8156. [CrossRef]
181. Mahalanobis, P.C. On the generalised distance in statistics. *Proc. Natl. Inst. Sci. India* **1936**, *2*, 49–55.

182. Chamikara, M.A.P.; Bertók, P.; Liu, D.; Camtepe, S.; Khalil, I. Efficient Data Perturbation for Privacy Preserving and Accurate Data Stream Mining. *Pervasive Mob. Comput.* **2018**, *48*, 1–19. [CrossRef]
183. Meurisch, C.; Bayrak, B.; Mühlhäuser, M. Privacy-Preserving AI Services through Data Decentralization. In Proceedings of the Web Conference 2020, Taipei, Taiwan, 20–24 April 2020; Association for Computing Machinery: New York, NY, USA, 2020; pp. 190–200. [CrossRef]
184. Soni, M.; Barot, Y.; Gomathi, S. A Review on Privacy-Preserving Data Preprocessing. *J. Cybersecur. Inf. Manag.* **2020**, *4*, 16–30. [CrossRef]
185. Li, X.; Guivant, J. Efficient and accurate object detection with simultaneous classification and tracking. *arXiv* **2020**, arXiv:2007.02065.
186. Zhang, C.; Li, R.; Kim, W.; Yoon, D.; Patras, P. Driver Behavior Recognition via Interwoven Deep Convolutional Neural Nets With Multi-Stream Inputs. *IEEE Access* **2020**, *8*, 191138–191151. [CrossRef]
187. Lin, Y.; Ding, W.; Qiang, S.; Deng, L.; Li, G. ES-ImageNet: A Million Event-Stream Classification Dataset for Spiking Neural Networks. *Front. Neurosci.* **2021**, *15*, 726582. [CrossRef]
188. Sun, Y.; Sun, Y.; Dai, H. Two-Stage Cost-Sensitive Learning for Data Streams With Concept Drift and Class Imbalance. *IEEE Access* **2020**, *8*, 191942–191955. [CrossRef]
189. Kathirgamanathan, B.; Cunningham, P. A Feature Selection Method for Multi-dimension Time-Series Data. In Proceedings of the Advanced Analytics and Learning on Temporal Data, Ghent, Belgium, 18 September 2020; Lemaire, V.; Malinowski, S.; Bagnall, A., Guyet, T., Tavenard, R., Ifrim, G., Eds.; Springer International Publishing: Cham, Switzerland, 2020; pp. 220–231.
190. Unnikrishnan, V.; Beyer, C.; Matuszyk, P.; Niemann, U.; Pryss, R.; Schlee, W.; Ntoutsi, E.; Spiliopoulou, M. Entity-Level Stream Classification: Exploiting Entity Similarity to Label the Future Observations Referring to an Entity. *Int. J. Data Sci. Anal.* **2020**, *9*, 1–15. [CrossRef]
191. Zhao, Z.; Birke, R.; Han, R.; Robu, B.; Bouchenak, S.; Mokhtar, S.; Chen, L.Y. Enhancing Robustness of On-Line Learning Models on Highly Noisy Data. *IEEE Trans. Dependable Secur. Comput.* **2021**, *18*, 2177–2192. [CrossRef]
192. Zyblewski, P.; Sabourin, R.; Woźniak, M. Data Preprocessing and Dynamic Ensemble Selection for Imbalanced Data Stream Classification. In Proceedings of the Machine Learning and Knowledge Discovery in Databases, Ghent, Belgium, 14–18 September 2020; Cellier, P., Driessens, K., Eds.; Springer: Cham, Switzerland, 2020; pp. 367–379.
193. Bahri, M.; Gomes, H.M.; Bifet, A.; Maniu, S. CS-ARF: Compressed Adaptive Random Forests for Evolving Data Stream Classification. In Proceedings of the 2020 International Joint Conference on Neural Networks (IJCNN), Glasgow, UK, 19–24 July 2020; pp. 1–8.
194. Godahewa, R.; Yann, T.; Bergmeir, C.; Petitjean, F. Seasonal Averaged One-Dependence Estimators: A Novel Algorithm to Address Seasonal Concept Drift in High-Dimensional Stream Classification. In Proceedings of the IEEE International Joint Conference on Neural Networks, Glasgow, UK, 19–24 July 2020; pp. 1–8.
195. Ahrens, K.; Abawi, F.; Wermter, S. DRILL: Dynamic Representations for Imbalanced Lifelong Learning. In Proceedings of the Artificial Neural Networks and Machine Learning Conference; Springer International Publishing: Cham, Switzerland, 2021. [CrossRef]
196. Burhanudin, U.F.; Maund, J.R.; Killestein, T.; Ackley, K.; Dyer, M.J.; Lyman, J.; Ulaczyk, K.; Cutter, R.; Mong, Y.L.; Steeghs, D.; et al. Light Curve Classification with Recurrent Neural Networks for GOTO: Dealing with Imbalanced Data. *Mon. Not. R. Astron. Soc.* **2021**, *505*, 4345–4361. [CrossRef]
197. Li, X.; Zhou, Y.; Jin, Z.; Yu, P.; Zhou, S. A Classification and Novel Class Detection Algorithm for Concept Drift Data Stream Based on the Cohesiveness and Separation Index of Mahalanobis Distance. *J. Electr. Comput. Eng.* **2020**, *2020*, 4027423. [CrossRef]
198. Yan, X.; Homaifar, A.; Sarkar, M.; Girma, A.; Tunstel, E. A Clustering-based framework for Classifying Data Streams. In Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, Montreal, QC, Canada, 19–27 August 2021; pp. 3257–3263. [CrossRef]
199. Alevizopoulou, S.; Koloveas, P.; Tryfonopoulos, C.; Raftopoulou, P. Social Media Monitoring for IoT Cyber-Threats. In Proceedings of the 2021 IEEE International Conference on Cyber Security and Resilience (CSR), Rhodes, Greece, 26–28 July 2021; pp. 436–441.
200. Vicuna, M.; Khannouz, M.; Kiar, G.; Chatelain, Y.; Glatard, T. Reducing Numerical Precision Preserves Classification Accuracy in Mondrian Forests. In Proceedings of the 2021 IEEE International Conference on Big Data (Big Data), Orlando, FL, USA, 15–18 December 2021.
201. Grzyb, J.; Klikowski, J.; Wozniak, M. Hellinger Distance Weighted Ensemble for Imbalanced Data Stream Classification. *J. Comput. Sci.* **2021**, *51*, 101314. [CrossRef]
202. Pugliese, V.; Costa, R.; Hirata, C. Comparative Evaluation of the Supervised Machine Learning Classification Methods and the Concept Drift Detection Methods in the Financial Business Problems. *Lect. Notes Bus. Inf. Process.* **2021**, *417*, 268–292. [CrossRef]
203. Zhang, Z.; Han, H.; Cui, X.; Fan, Y. Novel Application of Multi-Model Ensemble Learning for Fault Diagnosis in Refrigeration Systems. *Appl. Therm. Eng.* **2019**, *164*, 114516. [CrossRef]
204. Nguyen, T.T.; Luong, A.V.; Dang, M.T.; Liew, A.W.C.; McCall, J. Ensemble Selection based on Classifier Prediction Confidence. *Pattern Recognit.* **2020**, *100*, 107104. [CrossRef]
205. Li, P.; Wu, M.; He, J.; Hu, X. Recurring Drift Detection and Model Selection-Based Ensemble Classification for Data Streams with Unlabeled Data. *New Gener. Comput.* **2021**, *39*, 341–376. [CrossRef]