mathematics

Article

Incremental Decision Rules Algorithm: A Probabilistic and Dynamic Approach to Decisional Data Stream Problems

Nuria Mollá [1,2,*], Alejandro Rabasa [2], Jesús J. Rodríguez-Sala [2], Joaquín Sánchez-Soriano [2] and Antonio Ferrándiz [1,3]

[1] Teralco Solutions Ltd., 03203 Elche, Spain; antonio.ferrandiz@ua.es
[2] R.I. Centre of Operations Research, Miguel Hernandez University of Elche, 03202 Elche, Spain; a.rabasa@umh.es (A.R.); jesuja.rodriguez@umh.es (J.J.R.-S.); joaquin@umh.es (J.S.-S.)
[3] Computer Technology Department, University of Alicante, 03001 Alicante, Spain
* Correspondence: nuria.molla@goumh.umh.es

Abstract: Data science is currently one of the most promising fields used to support the decision-making process. Particularly, data streams can give these supportive systems an updated base of knowledge that allows experts to make decisions with updated models. Incremental Decision Rules Algorithm (IDRA) proposes a new incremental decision-rule method based on the classical ID3 approach to generating and updating a rule set. This algorithm is a novel approach designed to fit a Decision Support System (DSS) whose motivation is to give accurate responses in an affordable time for a decision situation. This work includes several experiments that compare IDRA with the classical static but optimized ID3 (CREA) and the adaptive method VFDR. A battery of scenarios with different error types and rates are proposed to compare these three algorithms. IDRA improves the accuracies of VFDR and CREA in most common cases for the simulated data streams used in this work. In particular, the proposed technique has proven to perform better in those scenarios with no error, low noise, or high-impact concept drifts.

Keywords: data mining methods for data streams; explainable temporal data analysis; classification methods

check for updates

Citation: Mollá, N.; Rabasa, A.; Rodríguez-Sala, J.J.; Sánchez-Soriano, J.; Ferrándiz, A. Incremental Decision Rules Algorithm: A Probabilistic and Dynamic Approach to Decisional Data Stream Problems. *Mathematics* **2022**, *10*, 16. https://doi.org/10.3390/math10010016

Academic Editor: Liangxiao Jiang

Received: 27 October 2021
Accepted: 17 December 2021
Published: 21 December 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Nowadays, machine learning techniques are increasingly used by a huge variety of sectors to improve their planification and actions based on previous knowledge. Going a step further with this machine learning approach and adding an adaptive skill might lead these models to react to changes early, improving their performance and avoiding greater losses. In this sense, in the management framework, slight changes could affect the inference capacity of a model before the manager even notices this variation. An algorithm that includes this incremental learning approach would provide a self-contained engine of knowledge, while other models would need a tracing process to detect and rebuild models. A wide variety of techniques have been implemented to deal with data streams as decision trees [1,2], decision rules [3,4], clustering methods [5] or association rules techniques [6].

These adaptive or incremental learning methods are being applied to many different fields such as Prognostics and Health Management [7], renewable energy systems [8] or driving cycle prediction [9] among others. Particularly in the field of decision support systems (DSS), the interest in efficiently converting data streams into operational intelligence is increasing. In this sense, some previous works have proposed the application of streaming methods to DSS as in [10,11].

A data stream is a potentially unbounded and ordered sequence of examples that arrive to a system over time. Data stream size is assumed to be infinite since it is not possible to predict when the stream will end, therefore it is not possible to keep all the data in the memory. Thus, models need to keep low computational and memory costs due to the permanent data streams. New instances may arrive to the system both one by one (online processing) or block by block (chunk-based processing). Apart from these considerations, the data stream properties may be influenced by changes over time, known as concept drifts. There are many types of drift that may affect the behavior of a system [12]. These changes could appear at any moment with no warning, so detecting and handling them is a challenge. A concept drift could arise at any time in any attribute of the dataset, as well as in the objective variable. Among the most relevant changes that can be found are abrupt or sudden drift, incremental, gradual, or reoccurring drifts. The experiments carried out in this work are simulated under abrupt-changing conditions.

Usually, when a new instance is received by these systems, the true label associated with a studied variable may be expensive to calculate in time and resources. Therefore, an approximation of this value is estimated using machine learning techniques. Particularly, classification methods estimate a label for this studied variable, best known as an objective variable, underlining possible patterns that may affect it. One of the most important techniques in classification tasks are decision trees. Among the contributions in streaming decision trees, the Hoeffding tree, also known as Very Fast Decision Trees (VFDT) and proposed by Domingos and Hulten [1], is considered one of the most relevant works. This method builds an adaptive tree using the Hoeffding bound to decide the exact number of examples that are necessary at each node. This bound is also used in many other algorithms such as the decision rule method known as Very Fast Decision Rules (VFDR) [3], explained in Section 2 and used in this paper as a benchmark technique.

Nevertheless, tree structures may be complex depending on their depth, making it difficult for experts to read information. This knowledge, derived from a tree structure, can also be represented in easier ways such as rules. A decision rule is a logic predicate structured as a condition “IF antecedent THEN label”. The antecedent is a conjunction of conditions for the attributes included in the dataset and the values in their domains. These decision or classification rules may contain the same information as the tree branches, however in a modular and more interpretable way. Unlike tree branches, each decision rule is independent and can be understood in isolation from the rest of the set. This approach might be more interesting for decisional contexts due to its readability and natural integration in a decision system engine.

This work proposes a novel decision rules method based on the Classification Rules Extraction Algorithm (CREA) [13], ID3 [14] and on RBS values [15]. This algorithm, Incremental Decision Rules Algorithm (IDRA), tries to propose a solution for those contexts where the data input is not fast, and the accuracy might be preferable. In this sense, IDRA is presented as an algorithm that automatically integrates new observations and provides updated probabilistic rules easily integrated in a decision support system. Our approach can both label an instance with a single class based on a majority class criteria, or give the probability distribution of the classes for that instance. Thus, a decision maker has more information about the possible classes, and their associated probabilities, before an action is taken, and therefore external conditions, such as the cost, the risk, or the tradeoff of the action, can be considered. IDRA does not aim to compete with the existing very fast algorithms but to solve different kinds of problems that also need updated and, maybe, more accurate bases of knowledge with not so demanding times. For instance, a high-risk scenario in which an error has a special impact on the system, might prefer a more accurate and time affordable response over a very fast and less precise one. Health or management decision systems could be examples of those that could value more a high accurate response while, a warning system based on sensors or fast decisions taken by automatons (such as, moving robots, self-driving cars, etc.) could be examples of a very fast action need. The presence of a human in the decision-making process affects considerably the time restrictions of a problem and, therefore, the algorithms that could be usable. In the case of expert systems that make decisions with no human involved in the process, very fast algorithms could be preferable, while for decision support systems with human action involved the preference may lie with more accurate and time affordable systems.

The paper is organized as follows. Section 2 is devoted to briefly introducing some related works and describing VFDR and CREA algorithms, the latter being the antecedent of the proposal of this work. In Section 3, the new algorithm, called IDRA, is presented and described. The description of the computational experiment and the obtained results are shown in Section 4. In Section 5, a discussion about the outcome is held, while in Section 6 a more general conclusion is offered together with further research.

2. Related Works

In this section, a non-exhaustive description of related works is presented to describe what has been previously made in the field. Within the data stream analysis field, there are many research areas such as data stream mining or data stream machine learning [2,3], as well as data profiling [16,17] or continuous queries for data streams [18,19].

As described in [20], the data stream solutions could be categorized in data-based and task-based ones. On the one hand, data-based solutions examine only a subset of the whole dataset, this being either a vertical or horizontal transformation of it. On the other hand, task-based solutions modify existing techniques or invent new ones to address data stream problems. The vast amount of works within the data stream field makes it difficult to make an exhaustive review of the literature, so this section will focus only on previous classification algorithms since it fits better with the nature of our proposal.

2.1. Classification Techniques for Data Stream

A previous classification technique proposed by Shaker and Hüllermeier [21] and called IBLStreams develops an instance-based learning classifier for data streams that adds or removes instances from the model based on temporal and spatial relevance and consistency. This method monitors the error rate and manages two registers during training, an error rate, and the standard deviation.

Among the most important classification methods, we can highlight VFDT as proposed by Domingos and Hulten [1]. This method builds a decision tree based on Hoeffding bounds, which guarantee constant time and memory per example. The output model given by this algorithm is asymptotically nearly identical to that given by a batch conventional learner. Other similar works have evolved this first version such as CVFDT extended in [22], the VFDTc proposed by Gama et.al. [23] or the Extremely Fast Decision Tree version [2]. In this last work, Bifet et. al. [2] remark three dimensions of interest to study: accuracy, memory and time, and defend the classification methods as one of the most used techniques in data mining. This paper follows the premises of this study [2] and the one led by Kosina and Gama [24] to evaluate the novel proposal. In this case, two of the three mentioned dimensions (accuracy and time) are explored for the simulated data streams.

2.2. Decision Rules for Data Stream

A commonly used strategy to obtain a decision rule set is the extraction of these rules from a decision tree [25]. This method allows the easy transformation of any decision tree into a collection of rules. Each rule corresponds to the path from the root to a leaf, and there are as many rules as leaves, with these rules being as complex as the decision tree itself. A decision rule is formed by an antecedent that implies a consequent. The antecedent describes the attributes that an instance needs to fit to be classified with that rule, while the consequent gives a prediction label. Each rule has support and confidence metrics, which express the relative use of a rule in a set (support) and the probability of a successful prediction for that rule (confidence).

Among the decision rules techniques, the first rule learner designed for processing data streams is the FACIL algorithm proposed by Ferrer et al. [26]. This method is based on filtering the examples lying near to the decision boundaries (border examples). Consistent rules (those including only one type of example) classify new test examples by covering them, while inconsistent rules (those mixing positive and negative examples, border examples) classify them by distance as the nearest neighbor algorithm.

Of the others, the best-known decision rules algorithm for data streams, and one which has been used to compare the proposal of this work, is the Very Fast Decision Rules (VFDR) proposed by Kosina and Gama [3]. VFDR is a single pass algorithm that learns ordered and/or unordered rules with either numerical or categorical variables. The rules expand considering those literals that minimize the class labels' entropy of the examples covered by each rule. This algorithm uses the Hoeffding bound [1] to determine the number of observations after which a rule can be expanded, or a new rule can be induced. The Hoeffding bound is defined by:

$$\epsilon = \sqrt{\frac{R^2 \cdot \ln\left(\frac{1}{\delta}\right)}{2n}} \tag{1}$$

where R is the range of the random variable, δ is the desired probability of the estimate not being within ϵ of its expected value, and n is the number of instances included in the node. This condition is checked only after a given number of examples (Nmin) to improve efficiency. Thus, the rule set may adapt every Nmin instance and would increase its predictive capacity over time.

Despite reasonable results, Hoeffding bound has been proven to be formally incorrect in solving such problems. Rutkowski et al. [27], discuss this idea and propose a generalization of Hoeffding inequality called McDiarmid inequality as split criteria.

Besides this, the limited size of the rule sets and the low time requirements of VFDR make it a high-quality data stream method. The algorithm introduced in this work is compared with VFDR since it is the main reference within the field of decision rules for data streams. Notably, it is important to highlight that these two algorithms differ slightly in nature. For VFDR the time may prevail over accuracy, while IDRA will try to provide more accurate solutions in an affordable time for the decision problem at hand. Therefore, the algorithm selection would depend on the problem we have to face and its associated restrictions. In Table 1, the advantages and disadvantages of some of the mentioned algorithms are shown.

Table 1. Scenarios, noise and mislabeling levels and threshold limits.

	Advantages	**Disadvantages**
VFDT	-Ability to learn from data streams -Very fast learning model -Constructs an identical or quasi identical tree to the traditional	-Tree structures need conversion to integrate in a DSS -A concept drift affects the structure of the tree -Uses the Hoeffding bound, which has been proven to be formally incorrect to solve such problems [27]
CVFDT	-Ability to learn from data streams -Very fast learning model	-Tree structures need conversion to integrate in a DSS -A concept drift affects the structure of the tree -Uses the Hoeffding bound, which has been proven to be formally incorrect to solve such problems [27]
VFDR	-Ability to learn from data streams -Very fast learning model -Readability -Rule structure is easily integrable in a DSS -Rules are handled independently from each other	-Uses the Hoeffding bound, which has been proven to be formally incorrect to solve such problems [27]

Table 1. *Cont.*

	Advantages	Disadvantages
CREA	-Readability -Rule structure is easily integrable in a DSS	-Cannot learn from data streams -Limited to discrete problems -Not suitable for very fast contexts
IDRA	-Ability to learn from data streams -Readability -Rule structure is easily integrable in a DSS -Rules are handled independently from each other -Soft rules with probability distributions	-Limited to discrete problems -Not suitable for very fast contexts

2.3. *Antecedents for IDRA*

The antecedent for the algorithm introduced in this work is the Classification Rules Extraction Algorithm (CREA) [13], based on the well-known ID3 rule algorithm [14]. CREA introduces a non-recursive approach to the ID3 decision tree generation process, which extracts the decision rules with no need to keep the tree structure in memory. This feature of the algorithm makes it a good antecedent for an incremental data stream adaptation since a new instance seen is assimilated by the system with no need to keep it in the memory.

This algorithm builds an ordered rule set from a discrete dataset, in which every rule has two key metrics, support and confidence. Under this premise, the first version of CREA generates all the possible rules for an antecedent, meaning one rule for each consequent. All these rules have the same support and split up the confidence regarding the data distribution. Nevertheless, the version used in this paper gathers all the same-antecedent rules in a single rule with all the classes and the confidence distribution. Thus, the number of rules decreases considerably, and with it the evaluation time. The predicted class is defined by a majority class criterion, in which the class with higher confidence in the consequent distribution is selected.

This algorithm includes a filtering method based on the Rules Based on Significance (RBS) criteria introduced in [15]. With this method, only the significant rules, those located in significant regions, with relevant levels of support and confidence, are used to predict labels.

The main steps followed by CREA to create and filter the decision rule set are described in this subsection, as well as the premises and its pseudocode. First of all, it is important to define that an instance is formed by a set of discrete descriptive attributes, $\boldsymbol{x}$ and a predictable variable, y, i.e., instance $\mathrm{I} = \{\boldsymbol{x}, y\}$ such that, $\boldsymbol{x} = \{x_1, x_2, \ldots, x_{c-1}\}$ where the values of x_i are defined as $v_i \in x_i$, $i = 1, \ldots, c-1$. Then, a data set D stands for a set of instances such that, $D = \{\mathrm{I}_i, i = 1, \ldots, n\}$.

A rule is formed by an antecedent and a consequent based on the values of the descriptive attributes $\boldsymbol{x}$ and the class prediction $w_j \in y, j = 1, \ldots, k$. CREA works with two different types of rules, those called totally expanded rules and those called not totally expanded rules. A totally expanded rule is one in which the antecedent has an assigned value for each attribute of D. On the contrary, a not totally expanded rule is a rule in which one or more than one descriptive attribute is not present in the antecedent of the rule. The rules are generated following an iterative process in which the descriptive attributes are expanded based on the information gain criteria [14]. This process continues until this rule is considered final, that means that all the attributes are expanded (totally expanded rule) or that all the values of the predictive variable belong to the same class (not totally expanded rule).

The first step of Algorithm 1 is the creation of the needed variables. A rule set that will store the set of built rules, a default rule and a list of no final rules (NFR) are initialized, as well as the RBS values in InitializeRBSvalues function. These values of Rules Based on Significance (RBS) are explained in detail in [15] and will be referenced in this work as RBS values. These values define the significance regions that will be used to filter the final rule set. The first rule of the NFR set will always be an empty rule (a rule that has not expanded any attribute yet) used as a root that will be expanded in the iterator. After these initializations, an iterative process starts until all the extracted rules are final, so that the no final rules set (NFR) is empty. A rule is expanded based on the best information gain attribute defined with BestGainRatioAttribute function. Then, another iterative process starts in which several rules are expanded for each value (v_i) of the selected attribute (x_i) with the ExpandRule function.

Next, the algorithm will keep expanding attributes until the rule is either a final not totally expanded rule or a final totally expanded rule. In the case of a final not totally expanded rule, CREA will create a consequent with the only class it remains, while in the case of a final totally expanded rule, it will create a rule with each one of the remaining classes as a consequent. In any of these cases, after the rule or rules are inserted in the rule set RS, the RBS values are updated with the UpdateRBSvalues function. Once the rule set is completely built, the RBS regions of significance are calculated, and the rules filtered based on the significance region to which they belong.

With this procedure, CREA builds the set of rules RS with no need to keep any seen instance on memory and then, filters the set to keep the most significant rules [15]. This algorithm is adapted in the proposal of this work to endow it with an incremental logic. The static version of CREA is compared with this new incremental version (IDRA) to study how these changes affect the process.

Algorithm 1: CREA: Classification Rule Extraction Algorithm.

```
input:    D: data set
output:   RS: rule set
begin
    Let RS ← { }
    Let defaultrule r ← Ø
    Let NFR ← {r}
    RBS ← InitializeRBSvalues()
    while NFR is not empty do
        r ← Extract any rule from NFR
        $x_i$ ← BestGainRatioAttribute(r,D)
        foreach $v_i \in x_i$ do
            r′ ← ExpandRule(r, $x_i = v_i$)
            if r′ is a final totally expanded rule then
                foreach $w_j \in y$ do
                    RS ← RS ∪ ExpandRule(r′, $y = w_j$)
                RBS ← UpdateRBSvalues()
            else if r′ is final not totally expanded rule then
                Let class value of r′ $y \leftarrow w_j$
                RS ← RS ∪ ExpandRule(r′, $y = w_j$)
                RBS ← UpdateRBSvalues()
            else
                NFR ← NFR ∪ {r′}
    CalculateRBSRegions(RBS)
    foreach r″∉ SignificantRBSRegion do
        RS ← RS – {r″}
```

2.4. Data Stream Tools

There are several frameworks or toolkits designed to simulate and analyze data streams. The best-known ones are Massive Online Analysis [28], Apache SAMOA [29], Spark Streaming [30] or Very Fast Machine Learning (VFML) [31]. There are also other tools developed within other fields that also work with data streams, such as the data profiling field. In this case, tools like metanome [32] or stradyvar [33] are focused on discovering metadata and representing data streams, compiling several data profiling techniques on them.

Specifically, among the mentioned tools for data stream analysis, both Apache SAMOA [29] and Spark Streaming [30] work with distributed environments that, at the moment, do not fit the requirements of our problem. VFML is one of the first toolkits for mining high-speed data streams and it is mostly written in C. Finally, MOA is a software environment for implementing algorithms and running experiments for online learning from evolving data streams. This software is one of the most important environments to analyze data streams and has a vast amount of available documentation. This framework is mostly written in Java, which is the environment used to run the VFDR algorithm, implemented under the name of DecisionRules.

3. Incremental Decision Rules Algorithm

The Incremental Decision Rules Algorithm (IDRA) is a new proposal based on the traditional ID3 method [14] and the optimized rule extraction method that has already been explained in detail in Section 2.3.

IDRA extends the functionality of CREA to data stream changing contexts adding an incremental logic to the decision rules. This new disposal will help IDRA to assimilate new data and adapt the predictive rule set. IDRA is specially designed for decisional data stream contexts. The management environment will favor an accurate and time-affordable response over a very-high-speed and less precise solution. IDRA trains an initial model that provides the rule set that will evolve incrementally over time, adding or ignoring rules. This initial model is built using the CREA algorithm, with a slightly different approach to the rule structure. This new version uses soft rules, with a (empirical) probability distribution of classes, to make a hard classification, so that only one class is given as a prediction. After this initial model is generated, the incremental learning stage begins, in which the new instances are integrated in the model.

To make predictions, IDRA works with a decision rule set (DRS) that contains those rules used to predict or support decisions, and a potential rule set (PRS) that includes those rules that are not yet relevant enough, although they may change in importance at some point. A rule might belong to the decision rule set or the potential rule set based on its support and a variable boundary defined following the minimum support axis proposed by [15]. This threshold is an adaptation of an RBS value used by CREA to filter generated rules. This boundary is recalculated after a given number of examples; therefore, the decision rule set (DRS) is constantly changing its size. Thus, the algorithm ignores the less relevant rules until they become important inside the rule set and uses the most relevant ones to classify until they lose importance. In IDRA, a new rule is added to the potential rule set every time an unseen example comes. This means that when that rule gains enough importance (there are a minimum of occurrences) this will be used to predict similar examples that come to the system. Thus, IDRA provides a consistent system that can predict with a high accuracy rate and with a certain complexity control over the rule set.

Figure 1 shows a high-level scheme of the proposed algorithm when an instance arrives to the system. When a new instance is observed, IDRA finds the best fitting rule in the decision rule set (DRS), which gives a class label. After that, the algorithm searches for the perfectly fitting rule, that is the rule that matches all its antecedents with the attributes of the example. Should that rule exist, the statistics are updated, while if it does not exist, a new rule is induced and added to the potential rule set (PRS). After a given delay, the true label arrives or can be calculated, so the algorithm updates the statistics of the model and sets the accuracy and error rates.

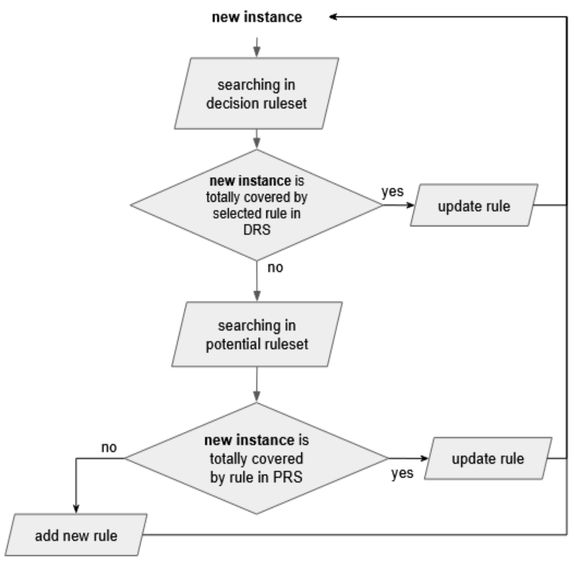

new instance
searching in decision ruleset
new instance is totally covered by selected rule in DRS
yes
update rule
no
searching in potential ruleset
new instance is totally covered by rule in PRS
no
yes
update rule
add new rule

Figure 1. High level flowchart of IDRA.

IDRA introduces rules whose consequents compile a probabilistic distribution of different class labels, what we call soft rules. This means that a single rule might contain several confidence rates relating to different classes, so there is no single metric to measure the inference capacity of that rule. To solve this problem, IDRA uses the entropy of the precision rates as a metric to compare rules. The lower the entropy is in a rule, the greater the difference among their class label confidences and therefore, the better capacity to predict. Thus, a rule with a single-class confidence of 100% would have a Shannon entropy of 0, while a rule with a confidence equally distributed into two classes would have a Shannon entropy of 1. The best inference capacity is given by a Shannon entropy of 0 and the worst inference capacity by the maximum possible Shannon entropy is given by this formula:

$$H_{\max} = \log_2(\#\text{Classes}) \tag{2}$$

The main steps of IDRA and the modifications made to adapt the CREA algorithm to the incremental logic are described in detail in this subsection, as well as the premises followed and the pseudocode of the algorithm. The description of IDRA will be presented using the same nomenclature used in the description of CREA (see Section 2.3).

In this case, the IDRA rules have an antecedent that implies a consequent such that the consequent is defined as $[w_1{:}p_1, w_2{:}p_2, \ldots, w_k{:}p_k]$, where p_j is the probability of the w_j class.

The required input of Algorithm 2 is a training data set, a stream data set, the size of the evaluation window and an entropy threshold. Algorithm 2 starts creating the same structures needed in Algorithm 1 and generates an initial rule set (RS) with a similar process. A loop starts extracting any rule of the no final rule set (NFR) and then selects an attribute x_i based on the information gain criterion [14] and expands all its values v_i with the ExpandRule function. After that, IDRA checks if this expanded rule is final, inserting r to the rule set RS if it is, and to the no final rule set (NFR) if it is not. After all the rules are final and the NFR set is empty, the initial rule set is built, and the data streaming process starts. The RBS values are initialized with the UpdateRBSThreshold function and based on this threshold, the decision rule set (DRS) and the potential rule set (PRS) are delimited with the defineDRSandPRS function. This method divides the RS, based on the RBS threshold, in a set of rules that are used to make predictions (DRS) and a set of rules that are expected to gain importance before being used in the decision-making process. After this, an auxiliar rule set (AuxRS) is created for all the not totally expanded rules (NTER). This auxiliar rule set contains all the totally expanded rules of each not totally expanded rule. That means that if there is a not totally expanded rule in RS, such that, $\boldsymbol{x} = \{x_1, x_2, \ldots, x_{i-1}, x_{i+1}, \ldots, x_{c-1}\}$ in AuxRS will be a set of totally expanded rules for all $v_i \in x_i$.

After the creation of the auxiliar rule set, the incremental learning process starts. For each instance received by the data stream, a prediction is made using the bestRule function, which selects the best rule of the DRS based on a given criteria, in this case the support. Next, IDRA searches for a rule in RS (DRS and PRS) that totally covers the descriptive attributes of the instance, updating the statistics of the rule if it is found or creating a new rule in the PRS if it is not. Thus, the algorithm slowly integrates new instances not observed before, increasing their importance after each occurrence. When a new set of explanatory attributes (or antecedents) is observed enough times, the associated rule climbs positions in the PRS and it might move to the DRS if it meets the significance conditions established in the RBSThreshold. In the case that the best rule in RS is a not totally expanded rule, the statistics of the corresponding auxiliar rules of AuxRS will also be updated to build auxiliar rules, as significant as possible, whose consequents represent the updated class distribution shown on the data stream.

After that, if a window check is required, several statistics within the model are updated. The RBS threshold is recalculated and, therefore, the DRS and the PRS are divided again based on this new limit. At this point, the entropy of the consequents of each not totally expanded rule is calculated based on the Shannon entropy [34]. Those rules whose Shannon entropy is above the entropyTolerance limit, will be replaced by the equivalent rules of AuxRS. The checkEntropy function will also erase the replaced rules from the NTER and the equivalent rules from the AuxRS. Thus, those not totally expanded rules with consequents that possess a not clear classification option (the Shannon entropy is far from the perfect case, H = 0), evolve to more concrete rules that might classify with better Shannon entropy rates (closer to 0). Finally, IDRA sorts the RS based on the support criteria and rules might reorder, producing changes in the system rule preferences when a classification action is taken.

Algorithm 2: IDRA: Incremental Decision Rule Algorithm.

```
input:    D: training data set
          DS: data stream
          Window: window size
          Entropy_tolerance
output:   RS: rule set
begin
    Let RS ← { }
    Let defaultrule r ← Ø
    Let NFR ← {r}
    while NFR is not empty do
        r ← Extract any rule from NFR
        $x_i$ ← BestGainRatioAttribute(r,D)
        foreach $v_i \in x_i$ do
            r′ ← ExpandRule(r, $x_i = v_i$)
            if r′ is a final rule then
                RS ← RS ∪ ExpandRule(r′,[$w_1$:$p_1$, $w_2$:$p_2$, . . . , $w_k$:$p_k$])
            else
                NFR ← NFR ∪ {r′}
    UpdateRBSThreshold()
    defineDRSandPRS(RS,RBSThreshold)
    Let NTER ← {r ∈ RS, r is not totally expanded rule}
    AuxRS ← ExpandAllRules(NTER)
    foreach instance {$\boldsymbol{x}$,$\boldsymbol{y}$} in DS do
        r ← bestRule(x,DRS)
        if exists TotallyCoveringRule($\boldsymbol{x}$, RS) then
            updateTotallyCoveringRuleStats($\boldsymbol{x}$,$\boldsymbol{y}$,RS)
```

Algorithm 2: *Cont.*

```
else
    PRS ← PRS ∪ newTotallyCoveringRule($x$,$y$)
if r is not totally expanded then
    update(r,AuxRS)
if Window check is Required then
    UpdateRBSThreshold()
    defineDRSandPRS(RS,RBSThreshold)
    checkEntropy(NTER,AuxRS,entropyTolerance)
sortBySupport(RS)
```

4. Experimental Evaluation

4.1. Dataset

To test the new proposed algorithm in data stream contexts, several datasets are simulated under the same conditions with different seeds, as well as a real dataset. Each simulated dataset has a total of 60,000 instances, 600 of them are used to train the initial model and the remaining 59,400 to simulate the stream. These proportions are taken from the work of Kosina and Gama [24]. In this sense, the method used to evaluate the models is the interleaved test-then-train or prequential evaluation. In this scheme, every individual example is used to test the model before it is used to update it. Thus, the model is always being tested on instances never seen before. With test-then-train evaluation, all examples are considered to compute accuracy, while prequential only uses those contained in a sliding window. In this case, the test-then-train scheme is used with no sliding window.

To simulate the changing data, the SEA method (Streaming Ensemble Algorithm) proposed by Street and Kim [35] is used. This technique generates abrupt changing datasets formed by three numerical variables (two of which are relevant) with values between 0 and 10, and a binary objective variable labelled following the next expression:

$$\mathrm{x1} + \mathrm{x2} \leq \theta \,, \tag{3}$$

where x1 and x2 are the values of the two relevant variables, and θ is the changing threshold used to generate different concept drifts. Should the addition of the first two attributes (x1 + x2) be less than or equal to θ, the example would be labelled as negative ($-$) and if it is more than θ, it would be labelled as positive (+). Following this logic, four concepts (three drifts) are generated by varying the value of θ, each one of them containing 15,000 examples. The default values of θ are 9, 8, 7 and 9.5. All the datasets simulated with these premises around the same distributions of 64% positive labelling and 36% negative labelling. This method is selected mostly as it fits the simulation requirements of a decisional context. The changes are abrupt, however its magnitude does not have a high impact on the data. That explains the decisional environments quite well, in which the context may change abruptly yet the scope is in most cases controlled and delimited in an expected range. An abrupt change with a high impact is not seen so often, however it is possible. The behaviors of all the algorithms are also studied in a highly demanding scenario with a considerable difference between the values of θ. In this case, the threshold limits are set to 9, 15, 8 and 14, generating concept drifts of a high impact. The simulated data for this high impact scenario also includes a 10% rate of noise.

Besides the simulated datasets that provide the experiment with a deep study of the condition that most affect each algorithm, a real dataset is also used to test the three algorithms. This dataset [36,37] contains data from the marketing campaigns of a bank and classifies the subscription of each client to a product. This dataset has a total of 45,211 instances with 11 discrete columns containing information about different aspects of a client. In this case, to simplify the analysis, the five most relevant attributes of each dataset are selected according to gain ratio criteria [38]. To the best of our knowledge, this dataset contains neither errors nor mislabeling, so in the next sections it will not be considered.

The set of data has no explicit concept drift so a similar performance of the static and the incremental algorithms (CREA and IDRA) is expected.

4.2. Noised Data

The data in real scenarios are exposed to errors that might be caused by miscaptured data in noisy environments, as well as mislabeled classes produced by human mistakes. To measure the algorithms' tolerance to these kinds of error, different levels of noise and mislabeling are included in simulated datasets. Each type of error is generated following a different logic. In the case of noise, this is generated by adding a random error, comprehended between the negative and positive value of the maximum error. That means that the classes were produced following this expression:

$$x1 + x2 + \text{ noise } \leq \theta\,, \tag{4}$$

The maximum error is calculated based on the noise rate and the attributes range. In this case, using SEA generation, the variables take values between 0 and 10, so if the noise level is fixed to 10%, the error oscillates within the $[-1, +1]$ range. In the case of mislabeling, the process consists of changing the correct class to the wrong one in each number of examples, depending on the defined level of error. Thus, if the level of mislabeling is set to 5%, 3000 out of 60,000 instances are incorrectly labelled. In real scenarios, this type of error might be caused mainly by human error and has a considerable impact on the models' performance.

Table 2 shows the different simulated scenarios in which the methods are tested, as they can be observed a total of 5 times each. The proposed scenarios include the perfect case (No error) with neither noise nor mislabeling, noisy scenarios (10% and 20% rates) and mislabeled scenarios (5% and 10% rates). The most used scenario in the literature is the one that includes 10% noise; therefore, the authors consider it to be the most important. However, other scenarios are also studied to delimit which algorithm is best for each situation, and which kind of error most affects each method.

Table 2. Scenarios, noise and mislabeling levels and threshold limits for simulated datasets.

Scenario Type	Experiment	Noise	Mislabel	Threshold θ
No error	5 datasets mean	0	0	[9,8,7,9.5]
Noisy	5 datasets mean	10	0	[9,8,7,9.5]
Noisy	5 datasets mean	20	0	[9,8,7,9.5]
Mislabeled	5 datasets mean	0	5	[9,8,7,9.5]
Mislabeled	5 datasets mean	0	10	[9,8,7,9.5]
High impact	5 datasets mean	10	0	[9,15,8,14]

4.3. Simulation Conditions and Limitations of the Experiment

The main purpose of these experiments is to study the behavior of the proposed algorithm in abrupt changing data stream contexts. IDRA is compared with the benchmark decision rule algorithm VFDR and a static rule oriented optimized ID3 (CREA) in terms of performance and time. In this sense, a wide variety of experiments are designed to study how different types of error affect each algorithm. The computer used for these experiments has a Processor Intel i7−10710U CPU @ 1.10GHz, 16GB RAM, Windows 10 20H2.

The algorithms tested in this section follow different rationales. First, CREA is a static algorithm commonly trained with batch data and tested using cross-validation. This algorithm has never been tested with data streams before, so the results also indicate the effect of abrupt changes on this method. On the other hand, IDRA is the incremental adaptation of the CREA algorithm. This method progressively updates rules with the incoming data streams with no need to keep examples on memory. Finally, VFDR is considered an adaptive algorithm, which is in this case able to deal with nominal and continuous attributes, as well as ordered and unordered rule sets. To compare the methods in similar conditions, ignoring the basic differences between the static, incremental and adaptive logics, all the algorithms are based on majority class criteria to make predictions and generate ordered rule sets with discrete datasets.

One of the limitations of the proposed algorithm, and therefore of this study is the nature of the analyzed data streams since only discrete data can be processed by IDRA and CREA. In this case, a previous discretization process has taken place to make the simulated data meet this requirement. Another of the limitations of this study is related to the time performances of the three tested algorithms. The version of VFDR used in this paper is integrated in the MOA framework [28] and therefore implemented in Java, while CREA and IDRA algorithms are written in Python. Even in this case, when the hardware conditions are the same, the different implementation languages of the algorithms might make the time comparison not so relevant. Nevertheless, the different nature of the algorithms is not a reason why the two algorithms cannot be compared [39,40]. Thus, the time results are shown as a valuable metric of the algorithms to evaluate which problems could be handled assuming certain time requirements. The study compares the mean accuracy of each method and presents the computational times to know whether an algorithm is suitable for a specific problem or not.

4.4. Experimental Design

The simulated datasets have 60,000 examples, of which 600 are used to train and 59,400 to test the static algorithm and test-then-train the incremental or adaptive ones. The same conditions are replicated in the case of the real dataset, 600 instances are used to train and 44,611 instances to test or test-then-train depending on the algorithm. All three methods are fed with the test data in the form of a stream. In order to compare the evolution of the different models, an evaluation window is fixed to take measures of the model accuracy. In all cases the size of this window is 1000 instances. In this sense, the mean accuracy obtained by the model for a given dataset is again used to calculate the mean accuracy of all datasets within the same scenario. In the same way, the mean time spent by an algorithm in a scenario is used as a comparative metric. In the case of accuracy, its evolution is considered relevant to compare the behavior and qualities of the methods. Every simulated scenario is understood as a set of error parameters. A total of five datasets are simulated and the mean metrics of each studied dimension are reported to compare the three algorithms. In the case of the real dataset, five data streams are generated by randomly shuffling the data. The mean accuracies and times obtained by the three algorithms in these five data inputs are shown in the next subsection.

4.5. Results

In these next paragraphs, the different tables summarizing the obtained results in the experimental evaluation will be shown. The discussion and comments derived from these results will be given in Section 4. Table 3 shows the mean accuracies of all models while Table 4 shows the mean times obtained for each algorithm in the described scenarios. With these measures, it can be quickly identified whether an algorithm is suitable for the time requirements of a system.

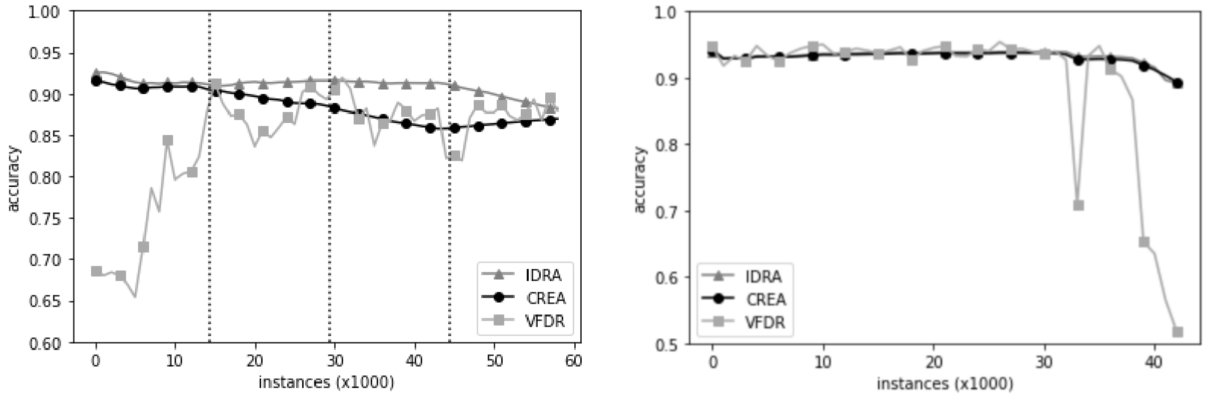
Table 3. Mean accuracy of algorithms.

	Simulated Scenarios						**Real Scenario**
	No Error	**Noise**		**Mislabel**		**High Impact**	**Bank**
Algorithm	0%	10%	20%	5%	10%	10% noise	0%
IDRA	0.930	0.896	0.848	0.814	0.744	0.841	0.9345
CREA	0.899	0.879	0.840	0.836	0.756	0.794	0.9329
VFDR	0.829	0.836	0.815	0.780	0.726	0.824	0.8882

Table 4. Mean time (in seconds) of the three algorithms.

	Simulated Scenarios						Real Scenario
	No Error	**Noise**		**Mislabel**		**High Impact**	**Bank**
Algorithm	0%	10%	20%	5%	10%	10% noise	0%
IDRA	32.5	31.5	31.2	29.8	33.8	30.7	13.3
CREA	16.4	17.5	18.6	18.9	21.7	17.3	11.1
VFDR	4.4	4.2	4.5	4.3	3.9	4.5	2.3

In Figure 2 we can see the evolution of the accuracy rate for all three algorithms in two experiments. We appreciate that both IDRA and CREA have more stable behaviors, while VFDR have more variable measures of accuracy. This last method is affected more by changes, however it is also able to recover in a suitable time.



(**a**) (**b**)

Figure 2. Accuracy evolution of IDRA, CREA and VFDR with two data streams. The concept drifts are represented with dashed lines. (**a**) Accuracy evolution of the three algorithms in the 10% noise simulated scenario. (**b**) Accuracy evolution of the three algorithms in one of the generated bank data streams.

Table 5 shows the different ranges of accuracy variability obtained by each model. These ranges are the mean differences between the maximum accuracy rate and the minimum in each scenario. This table tries to show the methods' volatility and discusses their reliability compared to its mean accuracy.

Table 5. Mean range of accuracy variability.

	Simulated Scenarios						Real Scenario
	No Error	**Noise**		**Mislabel**		**High Impact**	**Bank**
Algorithm	0%	10%	20%	5%	10%	10% noise	0%
IDRA	0.048	0.042	0.032	0.089	0.094	0.185	0.065
CREA	0.063	0.052	0.039	0.052	0.040	0.181	0.065
VFDR	0.345	0.274	0.272	0.229	0.212	0.380	0.428

5. Discussion

Table 3 shows that the mean accuracies obtained by IDRA are, in most cases, higher than the rest of the algorithms for the computational experiment. In the benchmark case, the simulated scenario with 10% of noise, IDRA obtains almost 2 points more of mean accuracy than CREA and 6 points more than the VFDR implementation. The comparison between models has its highest difference in the non-error simulated scenario and in the bank dataset, in which IDRA gets 93% and 93.7% of mean precision while CREA gets 90% and 93.5% and VFDR, 83% and 88.8%. In the case of the bank dataset, the results of CREA and IDRA are quite similar due to the lack of concept drifts that affect the model. Considering that the time differences between CREA and IDRA are affordable, the fact that IDRA could assimilate new data distributions with no need to detect and remodel, could be seen as an advantage over CREA.

Table 4 shows that the running times measured for the VFDR version are considerably lower than the other two algorithms. Nevertheless, as previously explained, in decisional contexts with no such demanding speeds, the accuracy rate may prevail over time. The results of each algorithm in a specific scenario may help to get an idea of their potential effectiveness in a particular case or industry. In the case of IDRA, the time is around 30 s in all cases for the simulated datasets with almost 60,000 instances and around 13 s for the bank dataset with more than 45,000 instances. That means that, in the simulated datasets, the model classifies 2000 instances per second, or put another way, it takes 0.5 milliseconds per instance. In the case of the bank dataset, the model classifies approximately 3400 instances per second or 0.3 milliseconds per instance. If the data stream is below these rates or the decision system can afford to have a small delay in classification, then the proposal is suitable for the scenario or industry studied.

In the high impact simulated scenario, the best mean performance is achieved again by the proposed algorithm, IDRA, with a difference of almost 2 points over VFDR implementation and 5 points over the static version. In the mislabeling scenarios, CREA becomes the best performing method with 5% rate obtaining 0.836 of mean accuracy (2 and almost 6 points of difference with respect to IDRA and VFDR, respectively). This better performance of CREA may be caused by its static nature and the data error distribution. Since IDRA and VFDR implementations include incremental logics, the mislabeled instances are added to the system at some point. CREA, with its static logic, does not include these mistaken examples so the model is not so affected, at least in a low error rate (5% mislabeling). In this sense, it can be concluded that the mislabeling error has more affect than the common noise error on both the static and incremental systems. In the case of 10% of mislabeled examples, the performance of CREA drops to lower mean accuracies than the high impact changes, 0.756 in 10% mislabeling scenario and 0.794 in high impact changes scenario. Also, in the cases of IDRA and the VFDR implementation, the mislabeling error, at both the 5% and 10% rate, have more of an effect on the mean accuracy than high impact changes.

Regarding the accuracy evolution in the simulated scenario (Figure 2a), both IDRA and CREA behave similarly before the first concept drift. From this point onwards, IDRA generally maintains its performance over time, while CREA degrades. This algorithm's drop of accuracy is very dependent on the θ used to build the initial set of rules that means the magnitude of the drift. In Figure 2a, it can be seen that CREA has a downward trend that slightly recovers when it reaches the fourth concept (from instance 44,400 onwards). In the case of VFDR, the precision changes quicker yet remains below the IDRA accuracy level most of the time. The accuracy evolutions of CREA and IDRA show a greater degradation of CREA performance when new observations arrive to the system. Thus, when the algorithm is exposed to data streams that are potentially infinite, the trend of the methods is a relevant element to consider. Therefore, even if in some cases, the trade-off between accuracy and time of CREA might seem preferable, the long-term maintenance of this method may lead to its degradation until the accuracy rates are not acceptable. In the bank data stream scenario the accuracy evolution (Figure 2b) shows a quasi-identical performance of IDRA and CREA. The fact that IDRA performs equally to CREA in data streams with no change, and that it can adapt in those that do contain changes, shows the improvements achieved by this new incremental version.

Table 5 shows that the mean accuracy ranges of IDRA are considerably lower compared to VFDR. That means that the model generated by IDRA is more stable over time than VFDR for the data streams used in this work, as it can also be seen in Figure 2. This behavior derives from a very-fast-changing policy that makes VFDR have lower accuracies at specific points beyond variable changing moments. IDRA has no explicit forgetting method so, in those changing moments the algorithm suffers no high impact in its accuracy. A permanent change in data is assimilated by the reranking of the rules. This process is not as fast as other methods implemented in very fast algorithms, however this might be preferable in cases where data is not changing so fast. This adaptation method of IDRA allows the model to ignore temporal changes in exchange for a slower update. In some practical cases, an algorithm that is able to maintain a stable accuracy rate while incrementally learning from changing data might be desirable over a very-fast-changing model.

6. Conclusions and Further Research

In conclusion, for the simulated data streams and the real bank dataset used in this paper, the new proposed incremental method IDRA performs more accurately than the other two algorithms considered in this work in most common scenarios. The accuracy of the VFDR algorithm is improved; specifically the MOA implementation of it, in almost all scenarios with error or without error. This also improves the static version of the algorithm, CREA, in all the cases except for two (mislabeling cases). In those two cases, the difference with IDRA is lower than the one with the VFDR proposal. This new proposed highly accurate method maintains its running times around 30 s classifying almost 60,000 instances in the simulated scenarios and around 13 s classifying more than 45,000 in the bank dataset. Commonly, decisional contexts have neither high time restrictions, nor deal with high rates of data arrival so the most accurate incremental model might be preferred. However, if the decisional context in which the algorithm will be running has faster response time requirements and precision is not so valuable, other algorithms such as VFDR might be preferable to IDRA.

This first version of the Incremental Decision Rules Algorithm (IDRA) is still subject to improvement, yet it has proven itself with a promising performance over other benchmark methods with the data streams used in this work. A stable behavior when exposed to changes and the resilience to error rate variations can make IDRA a comparable and sometimes a preferable algorithm for some contexts, which tend to be mostly decisional ones. However, for each practical situation, it must be evaluated which classification algorithms may be more suitable or have a better performance according to the objectives pursued.

Further research will try to equip IDRA with mechanisms to forget and adapt faster to the high impact scenarios. IDRA is specially designed to be implemented in a Decision Support System (DSS) that guides the actions and policies of the company using it. For example, in some investment procedures, it is important to reduce uncertainty, and therefore highly accurate algorithms add great value to these contexts. In this sense, the computational experiment carried out in this paper shows that IDRA can balance good accuracy rates with affordable times for decision making.

Future works will test IDRA in real decision contexts with other data streams and, the design of similar algorithms that work with precise rule sets in decision contexts will allow this algorithm to be tested against same-nature algorithms. Additionally, the analysis of other metrics as criteria to select decision rules in IDRA, might lead to the improvement of the performance of the algorithm in future research. By means of these criteria or other techniques, a management of expired instances could be also studied in further works.

Finally, in the studied case, we have established a demanding entropyTolerance level for IDRA, which has led to only high accurate rules being kept unexpanded in the system. However, the analysis of how different values of this metric could affect the algorithm might also be interesting with regard to further research.

Supplementary Materials: The following supporting information can be downloaded at: https://www.mdpi.com/article/10.3390/math10010016/s1.

Author Contributions: Conceptualization, N.M., A.R. and J.J.R.-S.; methodology, N.M., A.R. and J.J.R.-S.; algorithm, N.M. and J.J.R.-S.; software, N.M. and A.F.; validation, N.M., A.R. and J.S.-S.; formal analysis, J.S.-S. and N.M.; investigation, N.M.; writing—original draft preparation, N.M. and

A.R.; writing—review and editing, N.M., J.S.-S. and A.F.; visualization, N.M.; supervision, J.S.-S. and A.F.; project administration, A.R. and A.F.; funding acquisition, N.M., A.R. and A.F. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by grant DIN2018-010101 funded by MCIN/AEI/10.13039/501100011033, Teralco Solutions Ltd, and PROMETEO2021/063 funded by the Generalitat Valenciana. Open Access funding provided by Miguel Hernández University of Elche.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available in [36,37] and Supplementary Materials.

Acknowledgments: We thank three anonymous reviewers for their time to review our paper and for their helpful comments. These comments have been of great help to improve our work. We acknowledge grant DIN2018-010101 funded by MCIN/AEI/10.13039/501100011033, Teralco Solutions Ltd., project grant PROMETEO2021/063 funded by the Generalitat Valenciana, and the OA funding provided by Miguel Hernández University of Elche.

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

References

1. Domingos, P.; Hulten, G. Mining high-speed data streams. In Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, 20–23 August 2000; Association for Computing Machinery: New York, NY, USA, 2000; pp. 71–80.
2. Bifet, A.; Zhang, J.; Fan, W.; He, C.; Zhang, J.; Qian, J.; Holmes, G.; Pfahringer, B. Extremely fast decision tree mining for evolving data streams. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, 13–17 August 2017; Association for Computing Machinery: New York, NY, USA, 2017; pp. 1733–1742.
3. Gama, J.; Kosina, P. Learning Decision Rules from Data Streams. In Proceedings of the IJCAI International Joint Conference on Artificial Intelligence, Barcelona, Spain, 16–22 July 2011; AAAI Press/International Joint Conferences on Artificial Intelligence: Menlo Park, CA, USA, 2011; pp. 1255–1260.
4. Ferrer-Troyano, F.; Aguilar-Ruiz, J.S.; Riquelme, J.C. Data streams classification by incremental rule learning with parameterized generalization. In Proceedings of the 2006 ACM Symposium on Applied Computing, Dijon, France, 23–27 April 2006; Association for Computing Machinery: New York, NY, USA, 2006; pp. 657–661.
5. Aggarwal, C.C.; Han, J.; Wang, J.; Yu, P.S. A framework for clustering evolving data streams. In Proceedings of the 2003 VLDB Conference, Berlin, Germany, 9–12 September 2003; Morgan Kaufmann: Burlington, MA, USA, 2003; pp. 81–92.
6. Jiang, N.; Gruenwald, L. Research issues in data stream association rule mining. *ACM Sigmod Rec.* **2006**, *35*, 14–19. [CrossRef]
7. Zhang, L.; Lin, J.; Liu, B.; Zhang, Z.; Yan, X.; Wei, M. A Review on Deep Learning Applications in Prognostics and Health Management. *IEEE Access* **2019**, *7*, 162415–162438. [CrossRef]
8. Severiano, C.A.; de Silva, P.C.L.E.; Cohen, M.W.; Guimarães, F.G. Evolving fuzzy time series for spatio-temporal forecasting in renewable energy systems. *Renew. Energy* **2021**, *171*, 764–783. [CrossRef]
9. Liu, C.; Chen, Y.; Zhao, L. An adaptive prediction method based on data stream mining for future driving cycle of vehicle. *Proc. Inst. Mech. Eng. Part D J. Automob. Eng.* **2021**, *235*, 1702–1712. [CrossRef]
10. Wang, Y.; Zhang, X.; Wang, Z. A proactive decision support system for online event streams. *Int. J. Inf. Technol. Decis. Mak.* **2018**, *17*, 1891–1913. [CrossRef]
11. Yang, H.; Li, P.; He, Z.; Guo, X.; Fong, S.; Chen, H. A decision support system using combined-classifier for high-speed data stream in smart grid. *Enterp. Inf. Syst.* **2016**, *10*, 947–958. [CrossRef]
12. Gama, J.; Zliobaite, I.; Bifet, A.; Pechenizkiy, M.; Bouchachia, A. A Survey on Concept Drift Adaptation. *ACM Comput. Surv.* **2014**, *46*, 1–37. [CrossRef]
13. Rodríguez-Sala, J.J. Método para generación y ordenación de reglas de clasificación. Diseño y estudio computacional. Aplicación a la Inteligencia de Negocio. Ph.D. Thesis, Miguel Hernández University of Elche, Elche, Spain, 2014.
14. Quinlan, J.R. Induction of Decision Trees. *Mach. Learn.* **1986**, *1*, 81–106. [CrossRef]
15. Almiñana, M.; Escudero, L.; Martín, A.P.; Rabasa, A.; Santamaría, L. A classification rule reduction algorithm based on significance domains. *TOP* **2012**, *22*, 397–418. [CrossRef]
16. Schirmer, P.; Papenbrock, T.; Kruse, S.; Naumann, F.; Hempfing, D.; Mayer, T.; Neuschäfer-Rube, D. DynFD: Functional Dependency Discovery in Dynamic Datasets. In Proceedings of the 22nd International Conference on Extending Database Technology, Advances in Database Technology-EDBT, Lisbon, Portugal, 26–29 March 2019; OpenProceed-ings.org. pp. 253–264, ISBN 978-3-89318-081-3.

17. Caruccio, L.; Cirillo, S.; Deufemia, V.; Polese, G. Efficient Validation of Functional Dependencies during Incremental Discovery. In Proceedings of the 29th Italian Symposium on Advanced Database Systems, Pizzo Calabro, Italy, 5–9 September 2021.
18. Zaniolo, C. Logical Foundations of Continuous Query Languages for Data Streams. In *Datalog in Academia and Industry*; Lecture Notes in Computer Science; Barceló, P., Pichler, R., Eds.; Springer: Berlin/Heidelberg, Germany, 2012; Volume 7494. [CrossRef]
19. Ghanem, T.M.; Hammad, M.A.; Mokbel, M.F.; Aref, W.G.; Elmagarmid, A.K. Incremental Evaluation of Sliding-Window Queries over Data Streams. *IEEE Trans. Knowl. Data Eng.* **2007**, *19*, 57–72. [CrossRef]
20. Gaber, M.M.; Zaslavsky, A.; Krishnaswamy, S. Mining data streams. *ACM Sigmod Rec.* **2005**, *34*, 18. [CrossRef]
21. Shaker, A.; Hüllermeier, E. IBLStreams: A system for instance-based classification and regression on data streams. *Evol. Syst.* **2012**, *3*, 235–249. [CrossRef]
22. 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, 23–29 August 2001; Association for Computing Machinery: New York, NY, USA, 2001. [CrossRef]
23. Gama, J.; Rocha, R.; Medas, P. Accurate decision trees for mining high-speed data streams. In Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, 23–27 August 2003; Association for Computing Machinery: New York, NY, USA, 2003. [CrossRef]
24. Kosina, P.; Gama, J. Very fast decision rules for classification in data streams. *Data Min. Knowl. Discov.* **2013**, *29*, 168–202. [CrossRef]
25. Quinlan, J.R. *C4.5: Programs for Machine Learning*; Learning decision lists; Morgan Kaufmann Publishers: San Mateo, CA, USA, 1987; pp. 229–246.
26. Ferrer, F.; Aguilar, J.; Riquelme, J. Incremental rule learning and border examples selection from numerical data streams. *J. Univers. Comput. Sci.* **2005**, *8*, 1426–1439.
27. Rutkowski, L.; Pietruczuk, L.; Duda, P.; Jaworski, M. Decision Trees for Mining Data Streams Based on the McDiarmid's Bound. *IEEE Trans. Knowl. Data Eng.* **2013**, *25*, 1272–1279. [CrossRef]
28. Bifet, A.; Holmes, G.; Kirkby, R.; Pfahringer, B. MOA: Massive online analysis. *J. Mach. Learn. Res.* **2010**, *11*, 1601–1604.
29. Apache SAMOA—Scalable Advanced Massive Online Analysis. Available online: https://svn.apache.org/repos/asf/incubator/samoa/site/index.html (accessed on 30 November 2021).
30. Spark Streaming. Available online: https://spark.apache.org/streaming/ (accessed on 30 November 2021).
31. Domingos, H. VFML—A Tool Kit for Mining High-Speed Time-Changing Data Streams. 2003. Available online: https://cs.washington.edu/dm/vfml/ (accessed on 30 November 2021).
32. Papenbrock, T.; Bergmann, T.; Finke, M.; Zwiener, J.; Naumann, F. Data profiling with metanome. *Proc. VLDB Endow.* **2015**, *8*, 1860–1863. [CrossRef]
33. Breve, B.; Caruccio, L.; Cirillo, S.; Deufemia, V.; Polese, G. Dependency Visualization in Data Stream Profiling. *Big Data Res.* **2021**, *25*, 100240. [CrossRef]
34. Shannon, C.E. A Mathematical Theory of Communication. *Bell Syst. Tech. J.* **1948**, *27*, 379–423. [CrossRef]
35. Street, W.N.; Kim, Y.S. 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; Association for Computing Machinery: New York, NY, USA, 2001; Volume 4, pp. 377–382.
36. Moro, S.; Cortez, P.; Rita, P. A Data-Driven Approach to Predict the Success of Bank Telemarketing. *Decis. Support Syst.* **2014**, *62*, 22–31. [CrossRef]
37. UCI—Machine Learning Repository. Bank Marketing Data Set. Available online: https://archive.ics.uci.edu/ml/datasets/Bank%2BMarketing (accessed on 9 December 2021).
38. Orenes, Y.; Rabasa, A.; Rodriguez-Sala, J.J.; Sanchez-Soriano, J. Benchmarking Analysis of the Accuracy of Classification Methods Related to Entropy. *Entropy* **2021**, *23*, 850. [CrossRef] [PubMed]
39. Prechelt, L. An empirical comparison of C, C++, Java, Perl, Python, Rexx, and Tcl for a search/string-processing program. *Adv. Comput.* **2000**, *33*, 23–29.
40. Destefanis, G.; Ortu, M.; Porru, S.; Swift, S.; Marchesi, M. A statistical comparison of Java and Python software metric properties. In Proceedings of the 7th International Workshop on Emerging Trends in Software Metrics, Austin, TX, USA, 14–22 May 2016; pp. 22–28. [CrossRef]