Information Sciences 442–443 (2018) 220–234

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Concept drift detection based on Fisher's Exact test

Danilo Rafael de Lima Cabral, Roberto Souto Maior de Barros*

Centro de Informática, Universidade Federal de Pernambuco, Cidade Universitária, Recife, 50740-560, Brazil

ARTICLE INFO

Article history:
Received 5 August 2017
Revised 19 February 2018
Accepted 20 February 2018
Available online 21 February 2018

Keywords:
Concept drift
Data streams
Drift detection
Online learning
Statistical tests

ABSTRACT

Concept drift detectors are software that usually attempt to estimate the positions of concept drifts in large data streams in order to replace the base learner after changes in the data distribution and thus improve accuracy. Statistical Test of Equal Proportions (STEPD) is a simple, efficient, and well-known method which detects concept drifts based on a hypothesis test between two proportions. However, statistically, this test is not recommended when sample sizes are small or data are sparse and/or imbalanced. This article proposes an ingeniously efficient implementation of the statistically preferred but computationally expensive Fisher's Exact test and examines three slightly different applications of this test for concept drift detection, proposing FPDD, FSDD, and FTDD. Experiments run using four artificial dataset generators, with both abrupt and gradual drift versions, as well as three real-world datasets, suggest that the new methods improve the accuracy results and the detections of STEPD and other well-known and/or recent concept drift detectors in many scenarios, with little impact on memory and run-time usage.

1. Introduction

Data streams are environments that frequently contain very large (possibly infinite) amounts of data, flowing rapidly and continuously. Thus, methods that learn from these streams are usually online and under restrictions regarding the usage of memory and run-time. Moreover, reading the same data instance more than once is normally not possible. In addition, this scenario considers that the target distribution of the data may change over time, a situation known as concept drift [23].

A very common categorization of concept drift is based on the speed of change. When the changes between concepts are sudden and/or rapid, they are called *abrupt* and, when the transitions from one concept to another occur over a number of instances, they are called *gradual* [23].

There are many examples of online learning applications which may be affected by concept drift [49], including filtering spam in e-mail messages [30], monitoring data from sensors [33], intrusion detection [32], as well as sentiment analysis [47], among others.

Different directions have been proposed to learn from data streams with concept drift. A common approach is based on concept drift detection methods [25] which are implemented as lightweight software that usually monitor the prediction results of a base classifier and focus on identifying possible changes in the data distribution.

Several researchers have proposed ensembles using a base classifier, sometimes with more sophisticated strategies, and/or adopting different weighting functions to compute the resulting classification, e.g. Dynamic Weighted Majority (DWM) [31], Diversity for Dealing with Drifts (DDD) [38], Adaptable Diversity-based Online Boosting (ADOB) [46], Boosting-

* Corresponding author.
E-mail addresses: drlc@cin.ufpe.br (D.R.d.L. Cabral), roberto@cin.ufpe.br (R.S.M.d. Barros).

https://doi.org/10.1016/j.ins.2018.02.054
0020-0255/

like Online Learning Ensemble (BOLE) [7], and Fast Adaptive Stacking of Ensembles (FASE) [21]. Other methods concentrate on detecting concepts that recur to reuse previously trained classifiers, e.g. Recurring Concept Drifts (RCD) [24]. Moreover, some of these ensemble methods also rely on an auxiliary drift detection method [7,10,21,24,38,46].

Several concept drift detectors have been proposed over the years and the most well-known methods are Drift Detection Method (DDM) [22], Early Drift Detection Method (EDDM) [3], Adaptive Windowing (ADWIN) [8], Statistical Test of Equal Proportions (STEPD) [39], Paired Learners (PL) [2], and EWMA for Concept Drift Detection (ECDD) [43]. Of the above-mentioned methods, DDM and STEPD are among the most simple ones and, despite their simplicity, present good all-round performance [25]. Other more recent drift detection methods have also been proposed, including Sequential Drift (SeqDrift) [40], SEED Drift Detector (SEED) [28], Drift Detection Methods based on Hoeffdings Bounds (HDDM) [20], Fast Hoeffding Drift Detection Method (FHDDM) [41], Reactive Drift Detection Method (RDDM) [4,5], and Wilcoxon Rank Sum Test Drift Detector (WSTD) [4,6].

One problem of STEPD is that Nishida et al. [39] adopted a statistical test of equal proportions to detect concept drifts, even when the number of samples is small. The authors acknowledged the problem and claimed the reason for their decision not to use Fisher's Exact test [19] (when samples were small) was its high computational cost.

Besides being commonly used in the medical literature for statistical analysis [37], Fisher's Exact test was also effectively applied in data mining in order to discover dependencies between attributes [26]. Ross et al. [44], in their sequential monitoring of binomial variables, presented another application of this test for drift detection, implemented at an acceptable computational cost. Thus, an efficient implementation of Fisher's Exact test is a worthy contribution.

This work takes advantage of specific details of the intended application to provide an ingeniously efficient simple implementation of the computationally expensive Fisher's Exact test in order to propose three different applications of this test in the concept drift detection problem. More specifically, we propose Fisher Proportions Drift Detector (FPDD), which is a variation of STEPD using Fisher's Exact test when samples are small, Fisher Square Drift Detector (FSDD), which is similar to FPDD but uses the chi-square test instead of the test of equal proportions, and Fisher Test Drift Detector (FTDD), which always detects drifts using Fisher's Exact test.

In addition, using the Massive Online Analysis (MOA) framework [9], we tested the three proposed detectors against DDM, ECDD, SEED , FHDDM, and STEPD in quite a large number of scenarios, with both artificial and real-world datasets, using two different base classifiers, and also performed statistical evaluations and drift identification analysis of the results.

The rest of this article is organized as follows: Section 2 briefly surveys related work, with special attention given to STEPD ; Section 3 describes Fisher's Exact statistical test and its given implementation; Section 4 presents the three proposed detection methods and their respective abstract pseudo-codes; Section 5 details the configuration of the experiments, also including brief descriptions of the datasets used in the tests; Section 6 shows the results obtained, performs evaluations of accuracy, run-time, and memory consumption, statistically comparing accuracies, and analyses the drift identifications; and, finally, Section 7 summarizes our conclusions.

2. Related work - drift detection methods

In data stream environments, a common organization of the learning process is to use a concept drift detector together with a base learner. In general, the concept drift detection method analyses the prediction results of the base classifier and applies some decision model to attempt to detect changes in the data distribution. Methods that follow this approach include DDM [22], EDDM [3], and STEPD [39].

Different concept drift detection methods surveil the performance of the base learner using distinct strategies and/or statistics to decide when concept drifts have occurred. Also, a lower confidence level is usually set to indicate warning levels, signaling that concept drifts may take place. At these points, the detectors create a new instance of the base learner to be trained in parallel. Whenever a concept drift is confirmed, this new instance replaces the original classifier; and if the warning is found to be a false alarm, the new instance is discarded.

DDM detects concept drifts in a sequence of examples by analysing the error rate (the probability of making an incorrect prediction) and its corresponding standard deviation. On the other hand, EDDM is similar but uses the distance between two classification errors rather than the error rate.

DDM assumes the error rate decreases with more examples when the distribution is stationary. Also, when the error increases, DDM presumes that the data distribution has changed and the base classifier has become inefficient. In EDDM, the distance between two consecutive errors tends to increase and drifts are detected when it decreases.

Both methods use parametrized thresholds for the detection of *warnings* (w) and *drifts* (d). The parameters of DDM and their defaults are $w = 2.0$, $d = 3.0$, and $n = 30$, where n is the minimum number of instances before the detection of drifts is permitted. The parameters of EDDM and their defaults are $w = 0.95$, $d = 0.9$, and the minimum number of errors before drift detection is permitted, $e = 30$.

ADWIN [8] uses a variable sized sliding window, which is reduced when drifts occur and becomes larger with longer concepts. Two dynamic sub-windows store older and recent data. Drifts are detected when the difference in the averages between these sub-windows is higher than a given threshold. The parameters of ADWIN and their default values in its MOA implementation are the confidence level to reduce the window size ($\delta = 0.002$) and the minimum frequency of instances needed to reduce the window size ($f = 32$).

ECDD [43] adapts Exponentially Weighted Moving Average (EWMA) [42] to data streams with concept drifts. Given the mean and standard deviation of the data, EWMA detects significant changes in the mean of a sequence of random variables. In ECDD, the mean and standard deviation are not needed. The MOA implementation of ECDD has two parameters: the weights used to differentiate recent from old instances (λ) and the minimum number of instances before the detection of drifts is permitted (n). Its default configuration in MOA is: λ=0.2 and n=30.

SEED [28] draws on ADWIN and compares two sub-windows within a window W as well. When the two sub-windows have distinct averages (higher than a given threshold δ), the older sub-window (W_L) is dropped. SEED uses the Hoeffding Inequality with Bonferroni correction proposed in ADWIN to calculate the test statistic. It also performs block compression to eliminate unnecessary cut points and merge blocks that are homogeneous in nature. The authors of SEED claim it is faster and more memory efficient than ADWIN. The parameters of SEED and their respective default values in MOA are a block size ($b = 32$), a compress term ($c = 75$), the threshold ($\delta = 0.05$), the growth parameter that controls the magnitude of the increment in a linear fashion ($\alpha = 0.8$), and the base value for the linear function ($\epsilon = 0.01$).

The algorithm of FHDDM [41] uses a sliding window and Hoeffding's inequality [27]. It detects a drift when a significant difference between the *maximum* and the *most recent* probabilities of correct predictions is observed. The authors claim that the FHDDM algorithm results in less detection delay, fewer false positives and false negatives when compared to the state-of-the-art. The parameters of FHDDM and their respective default values are the size of the sliding window ($n = 25$) and the probability of error allowed ($\delta = 0.000001$).

2.1. STEPD

Similarly to DDM and EDDM, STEPD also monitors the predictions of a base classifier to decide when to signal warnings and drifts. Moreover, STEPD also has two parametrized thresholds referring to significance levels for the detection of drifts and warnings: $\alpha_d = 0.003$ and $\alpha_w = 0.05$, respectively.

STEPD considers the accuracy of the base classifier over two windows: a *recent* window, containing the last examples, and an *older* window, covering all the other examples seen by the current base learner, i.e., the instances processed after the last detected concept drift. The size of the *recent* window (w) is also a parameter and its default value is 30 instances.

This method assumes that the accuracies of the base learner over the two aforementioned windows should be the same, provided that no concept drift has occurred. Thus, a significant decrease in the accuracy over the examples of the *recent* window is the criterion for signaling warnings and drifts.

To compare the accuracies over the two windows, STEPD prescribes a hypothesis test of equal proportions with continuity correction, as presented in Eq. (1) [39]. Note that r_o is the number of correct predictions in the n_o examples of the *older* window, r_r is the number of correct predictions in the n_r (w) examples of the *recent* window, and $\hat{p} = (r_o + r_r)/(n_o + n_r)$.

$$T(r_o, r_r, n_o, n_r) = \frac{|r_o/n_o - r_r/n_r| - 0.5 \times (1/n_o + 1/n_r)}{\sqrt{\hat{p} \times (1 - \hat{p}) \times (1/n_o + 1/n_r)}} \tag{1}$$

The result of Eq. (1) is then used to find the p-value in the standard normal distribution table, which will be compared to the significance levels adopted for drifts and warnings. When p-value $< \alpha_d$ the null hypothesis ($r_o/n_o = r_r/n_r$) is rejected and STEPD detects a concept drift. Similarly, warnings are signalled when p-value $< \alpha_w$.

As previously mentioned, the authors of STEPD recognized, in a footnote [39], that they should have used Fisher's Exact test instead of the statistical test of equal proportions "where sample sizes are small" and explicitly admitted they "did not use it due to its high computational costs".

3. Fisher's Exact test and its implementation

Most statistical tests depend on an approximation that becomes more precise with an increase in the sample size and tends to be exact when the sample size heads towards infinity.

In scenarios where sample sizes are small or data are sparse and/or imbalanced, those tests that depend on approximations are likely to lose precision [37].

However, Fisher's Exact test [19,48] is different; it was originally proposed to analyse contingency tables and is recommended for the analysis of samples of *any* size. It is considered *exact* because its statistics directly returns a p-value, without the need to use any tables. Therefore it does *not* depend on approximations and is precise regardless of the sample size.

Even though it can be used to analyse tables of dimensions $m \times n$ [36], Fisher's Exact test is commonly applied to 2×2 contingency tables, which fits the case of STEPD windows as well as those of many other problems.

Let's consider Table 1 as a Fisher's Exact test generalized contingency table referring to hypothetical errors and hits of two windows of data – *a, b, c* and *d*, respectively, which has been adapted for the case of STEPD windows, where n is the total number of examples, i.e. $a + b + c + d$.

Fisher [19] and Yates [48] observed that, in a contingency table similar to Table 1, the probability that there are a (w_r) occurrences in $a + b$ ($w_r + w_o$) independent events can be determined by $((a + b)!/(a! \times b!)) \times p^a \times q^b$, where p is the probability of any event, $q = 1 - p$, and "!" is the *factorial* function. Similarly, the probability of c (r_r) events in a sample of $c + d$

Table 1
Fisher's Exact test generalized contingency table adapted for STEPD windows.

Labels	Recent window	Older window	Total
Errors	$w_r = a$	$w_o = b$	a + b
Hits	$r_r = c$	$r_o = d$	c + d
Total	$w = a + c$	$n_o = b + d$	$n = a + b + c + d$

$(r_r + r_o)$ trials is expressed by $((c+d)!/(c! \times d!)) \times p^c \times q^d$. Accordingly, the probability of all observed frequencies in a 2×2 contingency table can be generalized by Eq. (2). Moreover, Fisher [19] and Yates [48] also proved that Eq. (2) can be reformulated as in Eq. (3), using combinatorial analysis.

$$p = \frac{(a+b) \times (c+d)!}{a! \times b! \times c! \times d!} \times p^{a+c} \times q^{b+d} \tag{2}$$

$$p = \frac{\binom{a+b}{a} \times \binom{c+d}{c}}{\binom{n}{a+c}} = \frac{(a+b)! \times (c+d)! \times (a+c)! \times (b+d)!}{a! \times b! \times c! \times d! \times n!} \tag{3}$$

Assuming the null hypothesis that both windows are equally affected by errors, Eq. (3) returns the probability that the $a+b$ errors are *not* equally distributed on the two windows. Thus the null hypothesis that errors *are* equally distributed on both windows should be rejected *if and only if* the value of p obtained from the contingency table is smaller than the adopted significance level.

Note that, since the idea is to check whether any of the two windows is more likely to be subject to classification errors, this is considered a two-tailed hypothesis test, and the value of p must therefore be doubled [1] before it is compared to the adopted significance level.

For example, if the value of p resulting from Eq. (3) is 0.03, it is 0.06 that must be used in the test. Thus, should the significance level be 0.05, the null hypothesis would not be rejected, being rejected only for significance levels greater than 0.06, e.g. 0.1.

3.1. Implementation

As previously mentioned, Nishida et al. [39] avoided using Fisher's Exact test in the implementation of STEPD based solely on computational costs.

Considering the calculation of the p`-value` presented in Eq. (3), it is obvious that are the numerous calculations of the factorial function that makes the test computationally expensive, because a literal implementation of the equation would mean nine factorial calculations for each instance of the stream.

However, looking more carefully at the problem in hand, it is possible to observe that *all* these calculated factorials refer to numbers in the $[0 \ldots n]$ interval, where n is the number of examples seen. Therefore, given that the size of the *recent* window is fixed, one possible way to lower the cost of these calculations would be to limit the size of the *older* window of STEPD as well. Nevertheless, if implemented strictly, this would mean ignoring some of the data in the *older* window.

The first step in the solution we adopted to simplify the calculations and make them computationally efficient, without discarding any data, was to consider the size of the *older* window *as if it were* the same size of the *recent* window, i.e. the w parameter of STEPD . With this arrangement, $n = 2 \times w$, so the calculated factorials are restricted to numbers in the $[0 \ldots 2w]$ interval, making it very likely that, in a literal implementation of the test, most of the factorial calculations would be repeated several times when we consider the complete data stream.

To make this solution work well, without discarding any instances, we converted the number of errors ($w_o = b$) and hits ($r_o = d$) of the *older* window using a proportion of the change in its size: $w_p = w_o \times w \,/\, n_o$ and $r_p = w - w_p$.

Even though this arrangement could potentially result in some small loss of precision, similar losses could also occur in the calculations of very large factorials using float-point numbers (type Double), which are needed because the largest integral type implemented in Java (Long) is insufficient for the results. Additionally, we have observed experimentally that the proportions have practically no effect on the test results, producing the same results most of the time and very close results otherwise.

Assuming that many factorial calculations would be needed repeatedly, our next step to provide an efficient implementation of Fisher's Exact test was to previously calculate and store in an array, for direct access, all the possible factorial results that might be necessary. Moreover, all these factorials can be calculated for the cost of a single calculation, specifically the factorial of $2 \times w$, by storing its intermediate results.

Given that the default value of the recent window size w in STEPD is 30, this effectively means calculating the factorials of all the numbers in the [0... 60] interval at the cost of calculating the factorial of 60. Notice that, even if one decides to work with a larger value for w, e.g. 200, the difference in both run-time and memory usage is very likely to be negligible in practical terms.

4. Proposed methods

As discussed in Section 2, STEPD maintains statistics of two windows of data and the *recent* window has a fixed and rather limited size – default is 30. Thus, it is very plausible that the number of errors (or the number of correct predictions) in this window is sometimes going to be very small and, in these cases, the statistical approximation used in STEPD is likely to become imprecise.

To provide methods capable of being statistically precise, irrespective of window size, this work proposes three new methods that use Fisher's Exact test in the detection of concept drifts, namely FPDD, FSDD, and FTDD.

All three methods work very similarly to STEPD : they all monitor the prediction results of a base classifier, use the same two windows (*recent* and *older*), rely on statistical tests to signal warnings and drifts and have the same three parameters and default values, i.e. the size of the *recent* window ($w = 30$) and the significance levels for the detection of drifts ($\alpha_d = 0.003$) and warnings ($\alpha_w = 0.05$).

These parameters were set with the same defaults of STEPD , allowing for a fair comparison of the methods. In addition, we have run exploratory experiments and a genetic algorithm [45] to search for better sets of values for all methods, but we could not find any significantly better set of values over a large number of datasets for any of the detectors. In the case of the *recent* window size, 30 is also the smallest value that enables the comparison of the two windows to have statistical significance.

The only differences are related to the adopted statistical test arrangements. Besides using different tests in parts of the samples, the new methods measure the differences in the *errors*, whereas STEPD uses the number of correct predictions of the windows. Nevertheless, this modification was only adopted to make their implementations in the MOA framework run faster; they do *not* change the results.

Also, the three proposed methods use our strategy and implementation of the Fisher Exact test, based on the contingency table of errors and correct predictions in the two windows of data (as described in Section 3) to calculate the p-value.

Algorithm 1 presents an abstract pseudo-code representing the general flow of execution of the three proposed drift detection methods. The inputs to the methods are a data stream and the same three parameters of STEPD .

Lines 1–5 show a simplified high-level summary of the data that needs to be initialized at the beginning of the methods. Notice *methodStats* is an abstraction that refers to the specific variables that each method uses to implement its detection, *fact* is the array used to store the factorial results needed to calculate the p-value using Fisher's Exact test, and *constF* is used to capture the part of the Fisher test's p-value equation that remains constant for every instance.

Lines 6–19 refer to the main part of the algorithm. It is worth emphasizing that the methods implement their necessary adjustments after a drift detection when they receive the first instance of the new concept (lines 7–9). Thus, the contents of attribute *changeDetected* at line 7 is the one set at the previous instance. Note that, in MOA , changes in the base learner after detecting drifts are *not* directly implemented in the code of the methods – they only signal the drift points to other shared classes of the MOA implementation.

Algorithm 1: Fisher-based statistical drift detectors.

Input: Data Stream s, Recent Window Size w, Drift Level α_d, Warning Level α_w

```
1  reset methodStats
2  changeDetected ← false
3  maxim ← 2 × w
4  fact [ ] ← factorials of all numbers in [0...maxim] interval
5  constF ← fact[w]^2 / fact[maxim]
6  foreach instance in s do
7      if changeDetected then
8          reset methodStats
9          changeDetected ← false
10     Updates predictions in older and recent windows
11     isWarningZone ← false
12     if n_o ≥ w then
13         w_p ← round (w_o × w / n_o)
14         r_p ← w - w_p
15         Calculates the p-value, according to the specific method
16         if p-value < α_d then
17             changeDetected ← true
18         else if p-value < α_w then
19             isWarningZone ← true
```

Algorithm 2: FPDD – Calculation of p-value.

```
1  if w_r < 5 or r_r < 5 or w_p < 5 or r_p < 5 then
2      Calculates the p-value, according to FTDD
3  else
4      z ← w_r + w_p
5      z ← (|w_r − w_p| - 1) / sqrt (z × (maxim − z) / maxim)
6      p-value ← normalProbability (|z|)
7      p-value ← 2 × (1 - p-value)
```

Line 10 abstracts the updates needed in both windows every time a new instance of data is processed: this new instance is included in the *recent* window and the oldest instance of this window is moved to the *older* window.

Line 12 guarantees that detections only take place after the *older* window has at least w instances, i.e. after $2 \times w$ processed instances, whereas lines 13–14 mean that the proportional values of the *older* window prediction errors and hits needed for the calculations of the p-value are updated.

Line 15 abstracts the calculation of the p-value, which is different from one method to another. The details of these calculations for the three proposed methods are presented in the next subsections.

Finally, drifts and warnings are detected in lines 16–19.

4.1. FPDD

The first of the proposed methods, FPDD, is aimed at using Fisher's Exact test in the situations where either the number of errors or the number of correct predictions, in either of the two windows, is small. Otherwise, it uses the test of equal proportions, just like STEPD .

To decide whether the numbers of errors and hits are small or not, it was necessary to choose the minimum value that would make the test of equal proportions applicable. The chosen value was *five* [12].

Moreover, to gain efficiency, our implementation of FPDD uses the number of errors of the windows in the hypothesis test calculations and, in the *older* window, it uses the proportional value of errors (w_p) instead of the original value (w_o), as these decisions have virtually no effect on the accuracy results.

Algorithm 2 presents the pseudo-code with the implemented p-value calculation of FPDD. Note that the code in lines 4–5 is the result of simplifications in the original equation, demonstrated in Proposition 1, as the two windows considered in the calculation have size w. In addition, in lines 6–7 the p-value is calculated and, since the test is two-tailed, its value is doubled for comparison with the significance levels [12].

Proposition 1. *Given* $n_p = n_r = w$, *then* $\frac{|w_p/n_p - w_r/n_r| - 0{,}5 \times (1/n_p + 1/n_r)}{\sqrt{(w_p+w_r)/(n_p+n_r) \times (1-(w_p+w_r)/(n_p+n_r)) \times (1/n_p+1/n_r)}} = \frac{|w_p - w_r| - 1}{\sqrt{(w_p+w_r) \times (2w-(w_p+w_r))/2w}}$

Proof.

$$
\begin{aligned}
T(w_p, w_r, n_p, n_r) &= \frac{|w_p/n_p - w_r/n_r| - 0,5 \times (1/n_p + 1/n_r)}{\sqrt{(w_p+w_r)/(n_p+n_r) \times (1-(w_p+w_r)/(n_p+n_r)) \times (1/n_p+1/n_r)}} \\
&= \frac{|w_p/w - w_r/w| - 0,5 \times (1/w + 1/w)}{\sqrt{(w_p+w_r)/(w+w) \times (1-(w_p+w_r)/(w+w)) \times (1/w+1/w)}} \\
&= \frac{|(w_p - w_r)/w| - 0,5 \times 2/w}{\sqrt{(w_p+w_r)/2w \times (1-(w_p+w_r)/2w) \times 2/w}} \\
&= \frac{|w_p - w_r| - 1)/w}{\sqrt{1/w \times (w_p+w_r)/2 \times (1-(w_p+w_r)/2w) \times 2/w}} \\
&= \frac{|w_p - w_r| - 1)/w}{\sqrt{1/w \times (w_p+w_r) \times (1-(w_p+w_r)/2w) \times 1/w}} \\
&= \frac{(|w_p - w_r| - 1)/w}{1/w \times \sqrt{(w_p+w_r) \times (2w-(w_p+w_r))/2w}} \\
&= \frac{|w_p - w_r| - 1}{\sqrt{(w_p+w_r) \times (2w-(w_p+w_r))/2w}} \qquad \square
\end{aligned}
$$

Algorithm 3: FSDD – Calculation of p-value.

1 **if** $w_r < 5$ **or** $r_r < 5$ **or** $w_p < 5$ **or** $r_p < 5$ **then**
2 Calculates the p-value, according to FTDD
3 **else**
4 $ew_r \leftarrow (w_r + w_p) \times (w_r + r_r)\ /\ maxim$
5 $er_r \leftarrow (r_r + r_p) \times (w_r + r_r)\ /\ maxim$
6 $ew_p \leftarrow (w_r + w_p) \times (w_p + r_p)\ /\ maxim$
7 $er_p \leftarrow (r_r + r_p) \times (w_p + r_p)\ /\ maxim$
8 $x \leftarrow (|w_r - ew_r|)^2 / ew_r + (|r_r - er_r|)^2 / er_r$
9 $+ (|w_p - ew_p|)^2 / ew_p + (|r_p - er_p|)^2 / er_p$
10 p-value $\leftarrow$ **chiSquaredProbability** $(x, 1.0)$

Algorithm 4: FTDD – Calculation of p-value.

1 p-value $\leftarrow fact[w_r + w_p]\ /\ fact[w_r]\ /\ fact[w_p]$
2 $\times\ fact[r_r + r_p]\ /\ fact[r_r]\ /\ fact[r_p]$
3 p-value $\leftarrow p$-value $\times\ constF \times 2$

4.2. FSDD

Similarly to FPDD, FSDD also applies Fisher's Exact test when the number of errors or correct predictions in either of the two windows is smaller than *five* [35]. However, in the remaining scenarios, FSDD adopts the chi-square statistical test for homogeneity of proportions [12], rather than the test of equal proportions used in both STEPD and FPDD.

Algorithm 3 captures the calculation of the p-value in FSDD. Note that the equations in lines 4–9 refer to the calculation of the chi-square test for homogeneity of proportions. Also, in line 10, its result is then compared to the critical values of the chi-square distribution, as described by Chernoff and Lehmann [14], with one degree of freedom, following the procedure described by Bluman [12]. Notice that, just like FPDD, FSDD also uses the proportional values of errors (w_p) in the *older* window. It is also worth adding that the chi-square test is always right-tailed [12].

4.3. FTDD

Finally, FTDD detects changes in the data distribution using Fisher's Exact test exclusively. Its calculation of the p-value, shown in Algorithm 4 , is the simplest of the three methods. It is worth pointing out that the order of the operations in the calculation of the p-value of FTDD has been changed, in an attempt to avoid extreme values in the intermediate results, as a precaution measure to make the results as precise as possible.

4.4. Space and time complexity analysis

Analyzing the space complexities of the methods, none of them store large data structures, so their space complexities are all $O(1)$ [15].

Regarding the time complexity, again, none of the methods perform long iterations in their calculations, so the time complexities of all the methods are $O(n)$ [15], where n is the number of processed instances.

5. Experimental setting

This section describes all the relevant information on the experiments that were designed to test the proposed methods. Firstly, all the methods have been tested with both Naive Bayes (NB) and Hoeffding Tree (HT) as base learners, because they are the ones most frequently used in experiments in the area and have implementations in the MOA framework.

The accuracy evaluation was performed using the prequential methodology [16] with a sliding window of size 1000 as its forgetting mechanism (default in MOA). Also, each incoming instance is used initially for testing and subsequently for training. This means that every instance is used both for testing and for training but there is no training prior to testing.

The experiments were executed using a PC with a Core i7 4790K processor, 16GB of DDR3 1866 Mhz RAM, a SSD, and the Ubuntu Desktop 14.04 LTS 64 bits operating system.

We selected four artificial dataset generators of different complexities, described below, and built abrupt and gradual concept drift versions of three different sizes (50K, 100K, and 200K instances, respectively) for a total of 24 artificial datasets. In all these datasets, there are four concept drifts distributed at regular intervals. Thus, the size of the concepts in each dataset version of the same generator is different, covering three different scenarios. Also, the abrupt drifts were simulated

by joining different concepts, whereas the gradual changes were generated using a probability function to increase the chance of selecting instances from the new concept rather than the old one. Finally, in the gradual concept drifts datasets, the changes last for 500 instances.

In the artificial datasets, the experiments were executed 40 times to calculate the accuracies of the methods and the mean results were computed with 95% confidence intervals.

In addition to the artificial datasets, three well-known real-world datasets were selected to complement the evaluation of the proposed methods and these are also described below.

5.1. Datasets

This subsection introduces the datasets chosen for the experiments reported in Section 6. These experiments compare the performances of the proposed methods to those of DDM, ECDD, SEED , FHDDM, and STEPD . All of these have been previously used in the area and are publicly available, either in the MOA framework, from the MOA website, or at https://sites.google.com/site/moaextensions.

It is important to test using artificial datasets because the number, position, and duration of concept drifts can be chosen to cover different scenarios. The four selected generators are:

- LED generator [24,40,46] represents the problem of predicting the digit shown by a seven-segment LED display. It has 24 categorical attributes, 17 of which are irrelevant, and a categorical class, with ten possible values. Also, each attribute has a 10% probability of being inverted (noise). Concept drifts are simulated by changing the position of the relevant attributes.
- Mixed generator [3,22,24] has two boolean (v, w) and two numeric (x, y) attributes. Each instance can be classified as positive or negative. They are positive when at least two of the three following conditions are met: v, w, and $y < 0.5 + 0.3 \times \sin(3\pi x)$. Concept drifts are simulated by reversing the classification.
- RandomRBF generator [34,40] uses n centroids with their centers, labels and weights randomly defined, and a Gaussian distribution to determine the values of m attributes. A concept drift is simulated by changing the number and positions of the centroids. This dataset has six classes, 40 attributes and 50 centroids.
- Sine generator [22,41,43] uses two numeric attributes (x, y) and two contexts: Sine1 and Sine2. In Sine1, each instance is classified as positive if the point (x, y) is below the curve $y = \sin(x)$, whereas Sine2 uses the condition $y = 0.5 + 0.3 \times \sin(3\pi x)$. Concept drifts can be simulated either by alternating between Sine1 and Sine2 or by reversing the aforementioned conditions, i.e. points below the curves become negative.

On the other hand, the unpredictability of the real-world datasets is also useful, as the number and position of the concept drifts (if existent) are usually unknown. For our experiments, we chose three datasets from the most widely used in the area of data streams.

The specific datasets used are:

- Airlines [11] is a binary dataset composed of 539,383 instances. The goal is to predict whether flights are delayed or not, based on a set of flight information: name of the company, departure time, flight number, duration, airports of origin and destination, and day of the week.
- Covertype [20] contains 30 m $\times$ 30 m cells of data from US Forest Service (USFS) Region 2. It has 581,012 instances and 54 attributes, both numeric and categorical, and the goal is to predict the forest cover type. The version we used was *sorted* by the *elevation* attribute [29], which induces a natural gradual concept drift on the class distribution. This occurs because, depending on the elevation, some types of vegetation disappear while others start to appear [29].
- Pokerhand [41,46] represents the problem of identifying the value of a five-card hand in the game of poker. It has five categorical and five numeric attributes and the value of a hand (e.g. a pair, two pairs, etc.) is a categorical class with 10 possible values. In this original and harder-to-classify version, with one million instances, the cards are not ordered.

6. Experimental results and analysis

This section presents the results of the experiments run and includes analyses of accuracy, run-time, and memory usage over the 24 artificial dataset versions and the three real-world datasets using two base learners – NB and HT.

6.1. Accuracy results and analysis

Tables 2 and 3 present the accuracy results of the eight tested methods in all selected datasets, as well as their ranks, using NB and HT, respectively. In each dataset and in the ranks, the best result is written in **bold**.

Notice that, in absolute terms, the three proposed methods (FPDD, FSDD, and FTDD) improved the predictive accuracies of STEPD in most tested configurations, i.e. the results improved in all sizes of concepts, across all four tested dataset generators, with both abrupt and gradual concept drifts, as well as in the real-world datasets and in the two base learners, with only one exception (Airlines using HT), though some of these results are quite similar.

Moreover, in the artificial datasets with abrupt drifts, these methods achieved the very best results in all the versions using NB and in 66% of the datasets using HT as base classifier. In the gradual datasets, FHDDM presented the best results

Table 2
Average accuracies in percentage (%) using NB , with 95% confidence intervals in the artificial datasets.

DATASET	DDM	ECDD	SEED	FHDDM	STEPD	FPDD	FSDD	FTDD
A-50K-LED	71.93 (+-0.55)	68.75 (+-0.29)	56.81 (+-0.47)	72.14 (+-0.25)	68.55 (+-0.70)	72.25 (+-0.18)	**72.26 (+-0.18)**	72.23 (+-0.19)
A-50K-Mixed	90.91 (+-0.73)	89.80 (+-0.12)	91.39 (+-0.10)	91.68 (+-0.08)	91.40 (+-0.12)	**91.71 (+-0.08)**	**91.71 (+-0.08)**	**91.71 (+-0.08)**
A-50K-RBF	47.97 (+-1.19)	39.90 (+-0.32)	49.18 (+-0.31)	49.29 (+-0.30)	48.85 (+-0.37)	**50.60 (+-0.49)**	50.52 (+-0.50)	50.58 (+-0.50)
A-50K-Sine	83.57 (+-1.31)	86.42 (+-0.09)	87.08 (+-0.09)	87.38 (+-0.10)	87.26 (+-0.10)	**87.39 (+-0.10)**	**87.39 (+-0.09)**	**87.39 (+-0.10)**
A-100K-LED	72.58 (+-0.31)	69.00 (+-0.19)	57.46 (+-0.62)	72.52 (+-0.19)	69.32 (+-0.50)	72.87 (+-0.15)	**72.88 (+-0.17)**	**72.88 (+-0.16)**
A-100K-Mixed	90.90 (+-0.88)	89.83 (+-0.08)	91.70 (+-0.06)	91.89 (+-0.05)	91.53 (+-0.07)	**91.90 (+-0.05)**	**91.90 (+-0.05)**	**91.90 (+-0.05)**
A-100K-RBF	48.91 (+-1.24)	39.91 (+-0.20)	49.90 (+-0.22)	49.70 (+-0.29)	49.27 (+-0.34)	51.32 (+-0.41)	51.33 (+-0.43)	**51.39 (+-0.40)**
A-100K-Sine	82.82 (+-1.64)	86.43 (+-0.08)	87.25 (+-0.06)	87.42 (+-0.08)	87.28 (+-0.07)	**87.43 (+-0.08)**	**87.43 (+-0.08)**	**87.43 (+-0.08)**
A-200K-LED	71.98 (+-0.62)	69.06 (+-0.13)	58.47 (+-1.00)	72.82 (+-0.13)	69.89 (+-0.36)	73.18 (+-0.14)	**73.40 (+-0.12)**	73.16 (+-0.14)
A-200K-Mixed	90.35 (+-1.68)	89.89 (+-0.07)	91.85 (+-0.04)	91.97 (+-0.03)	91.57 (+-0.04)	**91.98 (+-0.03)**	**91.98 (+-0.03)**	**91.98 (+-0.03)**
A-200K-RBF	48.92 (+-1.21)	40.02 (+-0.14)	50.40 (+-0.19)	49.67 (+-0.23)	49.36 (+-0.30)	**51.68 (+-0.29)**	51.65 (+-0.29)	51.66 (+-0.29)
A-200K-Sine	81.85 (+-1.74)	86.45 (+-0.05)	87.34 (+-0.04)	87.44 (+-0.05)	87.32 (+-0.04)	**87.45 (+-0.05)**	**87.45 (+-0.05)**	**87.45 (+-0.05)**
G-50K-LED	**72.33 (+-0.22)**	68.38 (+-0.29)	56.99 (+-0.55)	71.68 (+-0.22)	68.01 (+-0.77)	71.64 (+-0.16)	71.66 (+-0.16)	71.63 (+-0.16)
G-50K-Mixed	90.39 (+-0.10)	88.80 (+-0.12)	90.15 (+-0.10)	**90.52 (+-0.08)**	90.08 (+-0.10)	90.41 (+-0.09)	90.42 (+-0.09)	90.41 (+-0.09)
G-50K-RBF	46.60 (+-0.89)	39.90 (+-0.32)	48.19 (+-0.30)	48.06 (+-0.36)	47.71 (+-0.34)	**48.83 (+-0.59)**	48.44 (+-0.78)	48.63 (+-0.62)
G-50K-Sine	86.30 (+-0.20)	85.60 (+-0.11)	86.11 (+-0.09)	**86.75 (+-0.09)**	86.23 (+-0.10)	86.56 (+-0.09)	86.58 (+-0.09)	86.56 (+-0.09)
G-100K-LED	72.47 (+-0.39)	68.81 (+-0.20)	57.30 (+-0.37)	72.29 (+-0.19)	69.05 (+-0.52)	72.41 (+-0.14)	**72.56 (+-0.15)**	72.43 (+-0.13)
G-100K-Mixed	91.23 (+-0.05)	89.32 (+-0.08)	91.07 (+-0.06)	**91.31 (+-0.05)**	90.87 (+-0.07)	91.23 (+-0.06)	91.24 (+-0.05)	91.23 (+-0.06)
G-100K-RBF	46.70 (+-1.23)	39.91 (+-0.20)	49.30 (+-0.40)	48.97 (+-0.25)	48.66 (+-0.33)	49.23 (+-0.50)	**49.39 (+-0.49)**	**49.39 (+-0.47)**
G-100K-Sine	86.38 (+-0.56)	86.00 (+-0.08)	86.71 (+-0.06)	**87.17 (+-0.07)**	86.73 (+-0.06)	87.03 (+-0.07)	87.05 (+-0.08)	87.04 (+-0.07)
G-200K-LED	73.21 (+-0.21)	69.05 (+-0.11)	59.85 (+-0.97)	72.68 (+-0.11)	69.65 (+-0.26)	73.16 (+-0.08)	**73.24 (+-0.08)**	73.17 (+-0.08)
G-200K-Mixed	91.65 (+-0.03)	89.64 (+-0.05)	91.56 (+-0.03)	**91.69 (+-0.03)**	91.26 (+-0.04)	91.65 (+-0.03)	91.66 (+-0.03)	91.66 (+-0.03)
G-200K-RBF	47.47 (+-1.12)	40.02 (+-0.14)	**50.16 (+-0.22)**	49.42 (+-0.24)	49.10 (+-0.30)	50.12 (+-0.51)	50.06 (+-0.55)	50.09 (+-0.49)
G-200K-Sine	84.86 (+-0.28)	84.25 (+-0.04)	85.02 (+-0.03)	**85.25 (+-0.04)**	84.99 (+-0.03)	85.19 (+-0.04)	85.20 (+-0.04)	85.19 (+-0.04)
Airlines	65.35	63.66	66.71	65.82	65.73	66.72	**66.91**	66.76
Covertype	67.14	67.39	67.32	67.94	67.62	68.29	68.17	**68.38**
Pokerhand	**50.11**	**50.11**	50.03	48.24	48.82	50.05	49.99	50.05
Rank	5.74074	7.27778	5.7037	3.74074	6.03704	2.75926	**2.2037**	2.53704

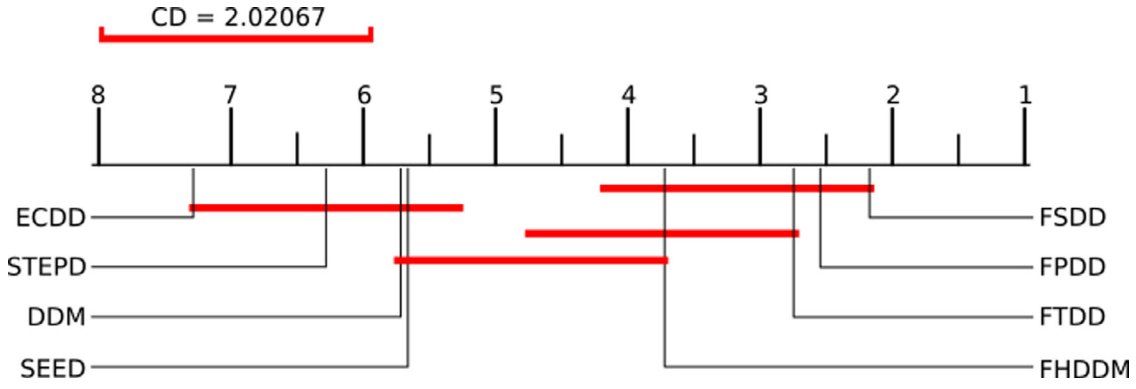

Fig. 1. Comparison results of the methods with Naive Bayes using the Nemenyi post-hoc test, with a 95% confidence on the artificial datasets.

in all the Mixed and Sine dataset versions, whereas DDM and the new methods were usually the best in the LED and RandomRBF datasets.

As a result, FSDD, FPDD, and FTDD, were the three best ranked methods in the tests with both base classifiers, though in different orders using NB and HT. Also, note that the results of the three methods were very similar in most situations. Of the other tested concept drift detectors, FHDDM and DDM were the best performing methods, achieving the fourth best rank in the tests using NB and HT, respectively, as base classifier.

Complementing the analysis of the reported results, we used the F_F statistic, based on the non-parametric Friedman test [17], to compare the results of the experiments. The null hypothesis states that all methods are statistically equal and, because it was rejected, it means that there is a statistical difference in some of the methods, but the test does not specify which. Thus, to discover this, a post-hoc test is necessary. We chose the Nemenyi-test, which compares each method against all the others and uses a critical difference (CD) as reference.

The statistic and the test were applied twice, once for the results from each base learner. We present the results using graphics, where the CD is represented by bars and methods connected by those bars are *not* statistically different. The results of the tests referring to the data in Tables 2 and 3 are summarized in Figs. 1 and 2, respectively.

Table 3
Average accuracies in percentage (%) using HT, with 95% confidence intervals in the artificial datasets.

DATASET	DDM	ECDD	SEED	FHDDM	STEPD	FPDD	FSDD	FTDD
A-50K-LED	72.12 (+-0.39)	68.71 (+-0.29)	56.03 (+-0.45)	72.12 (+-0.25)	67.89 (+-0.93)	**72.23 (+-0.18)**	72.21 (+-0.18)	72.20 (+-0.19)
A-50K-Mixed	91.45 (+-0.29)	89.76 (+-0.12)	90.93 (+-0.12)	**92.16 (+-0.10)**	91.11 (+-0.09)	92.09 (+-0.11)	92.09 (+-0.11)	92.09 (+-0.11)
A-50K-RBF	49.22 (+-0.62)	45.12 (+-0.32)	48.77 (+-0.38)	49.10 (+-0.33)	48.64 (+-0.32)	**50.36 (+-0.45)**	50.20 (+-0.48)	50.33 (+-0.46)
A-50K-Sine	91.05 (+-0.12)	87.09 (+-0.12)	89.26 (+-0.11)	91.50 (+-0.11)	90.40 (+-0.17)	**91.52 (+-0.12)**	**91.52 (+-0.12)**	**91.52 (+-0.12)**
A-100K-LED	72.68 (+-0.24)	68.97 (+-0.19)	56.41 (+-0.70)	72.50 (+-0.19)	68.84 (+-0.64)	72.87 (+-0.15)	72.85 (+-0.16)	**72.88 (+-0.15)**
A-100K-Mixed	92.79 (+-0.13)	89.78 (+-0.08)	91.28 (+-0.10)	93.16 (+-0.06)	91.26 (+-0.08)	**93.17 (+-0.06)**	**93.17 (+-0.06)**	**93.17 (+-0.06)**
A-100K-RBF	51.96 (+-0.57)	**52.78 (+-0.85)**	49.37 (+-0.19)	49.54 (+-0.28)	49.05 (+-0.29)	51.37 (+-0.40)	51.18 (+-0.36)	51.38 (+-0.41)
A-100K-Sine	92.30 (+-0.08)	87.17 (+-0.09)	89.54 (+-0.09)	92.58 (+-0.08)	90.97 (+-0.13)	**92.59 (+-0.08)**	**92.59 (+-0.08)**	**92.59 (+-0.08)**
A-200K-LED	72.60 (+-0.26)	69.04 (+-0.13)	57.02 (+-0.82)	72.81 (+-0.14)	69.53 (+-0.48)	**73.38 (+-0.08)**	73.34 (+-0.08)	73.35 (+-0.08)
A-200K-Mixed	93.63 (+-0.11)	89.83 (+-0.07)	91.61 (+-0.10)	**93.88 (+-0.06)**	91.34 (+-0.08)	93.87 (+-0.06)	93.87 (+-0.06)	93.87 (+-0.06)
A-200K-RBF	**54.60 (+-0.42)**	53.62 (+-0.34)	49.78 (+-0.14)	49.53 (+-0.23)	49.22 (+-0.25)	54.02 (+-0.42)	53.45 (+-0.62)	53.61 (+-0.45)
A-200K-Sine	93.79 (+-0.10)	87.26 (+-0.06)	89.68 (+-0.08)	94.02 (+-0.13)	91.50 (+-0.14)	**94.04 (+-0.13)**	**94.04 (+-0.13)**	**94.04 (+-0.13)**
G-50K-LED	**72.35 (+-0.20)**	68.36 (+-0.30)	55.97 (+-0.37)	71.66 (+-0.22)	67.89 (+-0.80)	71.62 (+-0.16)	71.63 (+-0.15)	71.61 (+-0.15)
G-50K-Mixed	90.88 (+-0.08)	88.69 (+-0.12)	89.70 (+-0.11)	**90.97 (+-0.07)**	89.79 (+-0.09)	90.74 (+-0.07)	90.77 (+-0.07)	90.75 (+-0.07)
G-50K-RBF	47.28 (+-0.64)	45.07 (+-0.33)	47.88 (+-0.37)	47.90 (+-0.38)	47.61 (+-0.34)	**48.58 (+-0.63)**	48.34 (+-0.64)	48.51 (+-0.64)
G-50K-Sine	90.27 (+-0.09)	86.37 (+-0.13)	88.14 (+-0.09)	**90.41 (+-0.08)**	89.17 (+-0.16)	90.26 (+-0.09)	90.27 (+-0.09)	90.26 (+-0.09)
G-100K-LED	**72.64 (+-0.27)**	68.78 (+-0.20)	56.24 (+-0.38)	72.27 (+-0.19)	68.54 (+-0.62)	72.45 (+-0.13)	72.55 (+-0.15)	72.47 (+-0.12)
G-100K-Mixed	92.46 (+-0.07)	89.24 (+-0.08)	90.69 (+-0.11)	**92.56 (+-0.06)**	90.66 (+-0.08)	92.45 (+-0.05)	92.46 (+-0.05)	92.46 (+-0.05)
G-100K-RBF	50.78 (+-0.66)	**52.52 (+-0.81)**	49.03 (+-0.19)	48.83 (+-0.27)	48.46 (+-0.30)	50.39 (+-0.46)	50.09 (+-0.41)	50.22 (+-0.42)
G-100K-Sine	91.97 (+-0.07)	86.82 (+-0.11)	88.86 (+-0.07)	**92.03 (+-0.07)**	90.44 (+-0.13)	91.92 (+-0.07)	91.93 (+-0.07)	91.92 (+-0.07)
G-200K-LED	73.09 (+-0.13)	69.03 (+-0.10)	57.90 (+-0.57)	72.67 (+-0.11)	69.25 (+-0.32)	73.13 (+-0.07)	**73.14 (+-0.08)**	**73.14 (+-0.07)**
G-200K-Mixed	93.97 (+-0.04)	89.58 (+-0.05)	91.42 (+-0.13)	**94.04 (+-0.05)**	91.10 (+-0.06)	93.97 (+-0.03)	93.97 (+-0.04)	93.97 (+-0.04)
G-200K-RBF	**53.92 (+-0.46)**	53.52 (+-0.31)	49.48 (+-0.16)	49.29 (+-0.24)	48.96 (+-0.23)	52.85 (+-0.43)	52.10 (+-0.46)	52.72 (+-0.39)
G-200K-Sine	93.47 (+-0.10)	85.51 (+-0.06)	88.26 (+-0.08)	**93.58 (+-0.10)**	89.42 (+-0.17)	93.48 (+-0.11)	93.48 (+-0.11)	93.48 (+-0.11)
Airlines	65.30	63.82	**65.44**	65.37	65.37	64.76	64.73	64.75
Covertype	**75.64**	70.05	70.93	70.84	70.75	72.04	71.36	72.78
Pokerhand	**51.85**	51.00	50.21	49.49	49.75	51.44	51.79	51.43
Rank	3.35185	6.59259	6.51852	3.77778	6.57407	**2.94444**	3.16667	3.07407

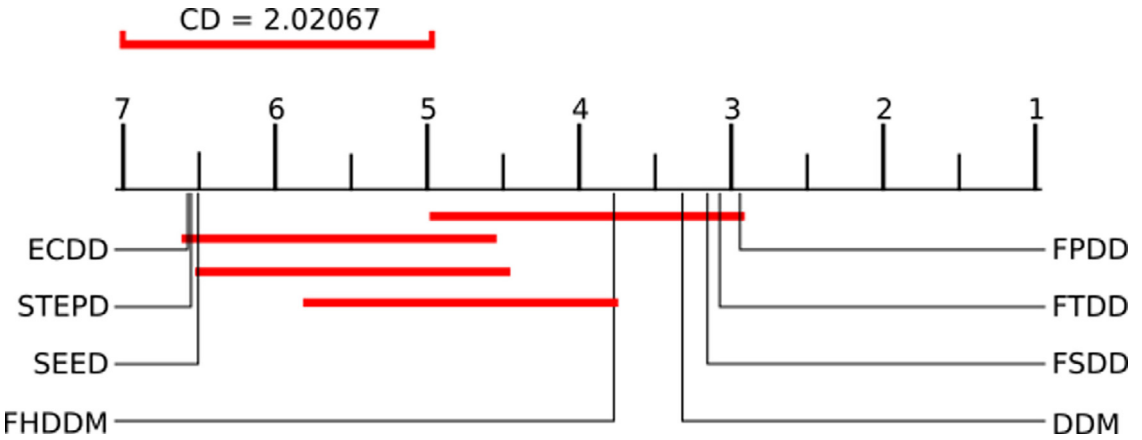

Fig. 2. Comparison results of the methods with Hoeffding Tree using the Nemenyi post-hoc test, with a 95% confidence on the artificial datasets.

Notice that, according to Fig. 1 (using NB), FSDD, FPDD, and FTDD are significantly better than SEED , DDM, STEPD , and ECDD. Also, FHDDM is only better than STEPD and ECDD, even though FHDDM is *not* statistically different to any of the proposed methods.

On the other hand, when the base learner is HT (Fig. 2), there is *no* statistical difference between FPDD, FTDD, FSDD, DDM, and FHDDM. In addition, these five methods are all statistically superior to SEED , STEPD , and ECDD.

6.2. Drift identification analysis

A different perspective regarding the performance of the drift detectors can be obtained by analysing the number and position of the concept drifts identified by each method.

Tables 4 and 5 present, for each *abrupt* dataset configuration, using NB and HT, respectively, the mean distance to the real drift points (μD) in the true positive drift detections as well as the total number of false negatives (FN) and false positives (FP) of each method considering the 40 repetitions. In each dataset, the best results are written in **bold**.

Table 4
Concept drift identifications of the methods in the abrupt datasets using NB base classifiers.

DET.	μD	FN	FP	Precision	Recall	F1	DATASET	DET.	μD	FN	FP	Precision	Recall	F1
DDM	148.04	114	124	0.270588	0.287500	0.278788		STEPD	47.07	37	2373	0.049279	0.768750	0.092620
ECDD	**27.32**	48	1445	0.071933	0.700000	0.130460	LED – 50K	FPDD	56.36	34	73	**0.639535**	0.687500	**0.662651**
SEED	81.69	83	8477	0.009002	0.481250	0.017673		FSDD	47.65	41	127	0.483740	0.743750	0.586207
FHDDM	38.36	**26**	277	0.326034	**0.837500**	0.469352		FTDD	55.69	51	**72**	0.602210	0.681250	0.639296
DDM	70.51	2	26	0.858696	0.987500	0.918605		STEPD	10.44	**0**	418	0.276817	**1.000000**	0.433604
ECDD	**10.00**	**0**	2498	0.060196	**1.000000**	0.113556	Mixed – 50K	FPDD	17.13	**0**	7	0.958084	**1.000000**	0.978593
SEED	30.00	**0**	365	0.304762	**1.000000**	0.467153		FSDD	17.13	**0**	7	0.958084	**1.000000**	0.978593
FHDDM	19.81	**0**	**0**	**1.000000**	**1.000000**	**1.000000**		FTDD	17.13	**0**	7	0.958084	**1.000000**	0.978593
DDM	157.50	104	105	0.132231	0.133333	0.132780		STEPD	56.70	29	716	0.112763	0.758333	0.196332
ECDD	N/A	120	**0**	0.000000	0.000000	0.000000	RBF – 50K	FPDD	76.53	25	36	**0.725191**	0.791667	**0.756972**
SEED	92.95	**8**	558	0.167164	**0.933333**	0.283544		FSDD	66.70	23	51	0.655405	0.808333	0.723881
FHDDM	**48.08**	16	733	0.124253	0.866667	0.217346		FTDD	69.90	23	40	0.708029	0.808333	0.754864
DDM	88.26	45	137	0.456349	0.718750	0.558252		STEPD	14.62	2	589	0.211513	0.987500	0.348401
ECDD	**9.87**	2	3396	0.044457	0.987500	0.085083	Sine – 50K	FPDD	20.13	**0**	3	0.987654	**1.000000**	0.993789
SEED	35.25	**0**	259	0.381862	**1.000000**	0.552677		FSDD	19.44	**0**	2	0.987654	**1.000000**	0.993789
FHDDM	19.81	**0**	**0**	**1.000000**	**1.000000**	**1.000000**		FTDD	19.69	**0**	2	0.987654	**1.000000**	0.993789
DDM	247.78	88	103	0.411429	0.450000	0.429851		STEPD	61.11	**25**	4336	0.030195	**0.843750**	0.058303
ECDD	40.98	48	2999	0.036001	0.700000	0.068481	LED – 100K	FPDD	65.63	50	**62**	**0.633166**	0.787500	**0.701950**
SEED	136.19	26	16,543	0.008035	0.837500	0.015917		FSDD	58.29	31	206	0.385075	0.806250	0.521212
FHDDM	**39.40**	26	634	0.174479	0.837500	0.288793		FTDD	66.35	34	78	0.617647	0.787500	0.692308
DDM	101.76	7	48	0.761194	0.956250	0.847645		STEPD	10.25	**0**	919	0.148285	**1.000000**	0.258273
ECDD	**9.88**	**0**	5141	0.030183	**1.000000**	0.058597	Mixed – 100K	FPDD	18.00	**0**	7	0.958084	**1.000000**	0.978593
SEED	40.00	**0**	416	0.277778	**1.000000**	0.434783		FSDD	18.00	**0**	7	0.958084	**1.000000**	0.978593
FHDDM	19.69	**0**	**0**	**1.000000**	**1.000000**	**1.000000**		FTDD	18.00	**0**	7	0.958084	**1.000000**	0.978593
DDM	255.95	83	110	0.251701	0.308333	0.277154		STEPD	75.77	16	1427	0.067929	0.866667	0.125984
ECDD	N/A	120	**0**	0.000000	0.000000	0.000000	RBF – 100K	FPDD	98.52	5	36	**0.761589**	0.958333	**0.848708**
SEED	91.51	**1**	837	0.124477	**0.991667**	0.221190		FSDD	91.32	6	65	0.636872	0.950000	0.762542
FHDDM	**44.73**	10	1476	0.069357	0.916667	0.128957		FTDD	94.17	5	39	0.746753	0.958333	0.839416
DDM	137.39	41	157	0.431159	0.743750	0.545872		STEPD	13.19	**0**	1148	0.122324	**1.000000**	0.217984
ECDD	**10.20**	9	6955	0.021250	0.943750	0.041563	Sine – 100K	FPDD	18.94	**0**	**2**	**0.981595**	**1.000000**	**0.990712**
SEED	40.00	**0**	303	0.345572	**1.000000**	0.513644		FSDD	18.31	**0**	6	0.963855	**1.000000**	0.981595
FHDDM	20.31	**0**	3	**0.981595**	**1.000000**	**0.990712**		FTDD	18.56	**0**	3	**0.981595**	**1.000000**	**0.990712**
DDM	447.16	72	**79**	0.526946	0.550000	0.538226		STEPD	83.31	12	7964	0.018245	0.925000	0.035783
ECDD	61.08	30	6174	0.020622	0.812500	0.040223	LED – 200K	FPDD	70.70	31	**79**	**0.620192**	0.806250	**0.701087**
SEED	185.07	**8**	31,447	0.004810	**0.950000**	0.009572		FSDD	82.96	18	269	0.345499	0.887500	0.497373
FHDDM	**56.58**	14	1169	0.111027	0.912500	0.197966		FTDD	70.55	32	97	0.568889	0.800000	0.664935
DDM	157.01	3	49	0.762136	0.981250	0.857923		STEPD	10.63	**0**	1844	0.079840	**1.000000**	0.147874
ECDD	**9.74**	4	10,217	0.015039	0.975000	0.029621	Mixed – 200K	FPDD	18.81	**0**	7	0.958084	**1.000000**	0.978593
SEED	40.00	**0**	510	0.238806	**1.000000**	0.385542		FSDD	18.81	**0**	7	0.958084	**1.000000**	0.978593
FHDDM	19.44	**0**	**0**	**1.000000**	**1.000000**	**1.000000**		FTDD	18.81	**0**	7	0.958084	**1.000000**	0.978593
DDM	525.79	44	106	0.417582	0.633333	0.503311		STEPD	525.79	44	106	0.417582	0.633333	0.503311
ECDD	N/A	120	**0**	0.000000	0.000000	0.000000	RBF – 200K	FPDD	96.44	2	51	**0.698225**	0.983333	**0.816609**
SEED	86.58	**0**	1315	0.083624	**1.000000**	0.154341		FSDD	87.29	2	96	0.551402	0.983333	0.706587
FHDDM	**54.17**	17	3073	0.032431	0.858333	0.062500		FTDD	93.90	2	60	0.662921	0.983333	0.791946
DDM	216.88	51	168	0.393502	0.681250	0.498856		STEPD	**12.25**	**0**	2271	0.065817	**1.000000**	0.123504
ECDD	25.13	2	13,992	0.011166	0.987500	0.022082	Sine – 200K	FPDD	19.19	**0**	**8**	**0.952381**	**1.000000**	**0.975610**
SEED	39.75	**0**	354	0.311284	**1.000000**	0.474777		FSDD	18.63	**0**	14	0.919540	**1.000000**	0.958084
FHDDM	19.75	**0**	12	0.930233	**1.000000**	0.963855		FTDD	18.88	**0**	**8**	**0.952381**	**1.000000**	**0.975610**

To compute the true positive identifications, we considered the concept drifts detected within 2% of the concept size after the correct drift points. For instance, in the 50K datasets, the concepts last for 10K instances and thus detections occurred up to 200 instances after the exact points were considered true positives.

It is worth pointing out this analysis used only the abrupt datasets because we know exactly where the concept drifts occur. Since there is no single change point in the gradual concept drift datasets, it is not clear how the identifications should be classified as positive or negative.

The mean distances of the correct detections of ECDD and STEPD were the closest to the exact points in most of the configurations tested. However, these results usually came at the cost of many false positives, impairing their accuracies in most datasets. In addition, in all versions of RandomRBF, ECDD did *not* identify any concept drifts using NB . The detections of FPDD, FSDD, FTDD, and FHDDM were usually fairly close to the best results.

The false negatives are related to existing drifts *not* detected by the methods. In this metric, the methods had fairly similar performances, with the exceptions of DDM and, to some extent, ECDD, which usually performed worse than the other methods. Also, all the methods presented many false negatives in the LED and RandomRBF datasets.

False positives refer to identified drifts where none exist. In this case, FPDD and FTDD were the best methods in most versions of LED, whereas FHDDM was the best in most versions of Mixed and Sine. In the three aforementioned dataset generator versions, these three methods presented comparable false positive results. In RandomRBF, ECDD had no false

Table 5
Concept drift identifications of the methods in the abrupt datasets using HT base classifiers.

DET.	μD	FN	FP	Precision	Recall	F1	DATASET	DET.	μD	FN	FP	Precision	Recall	F1
DDM	148.26	114	124	0.270588	0.287500	0.278788		STEPD	50.70	32	2734	0.044724	0.800000	0.084712
ECDD	**28.48**	48	1437	0.072305	0.700000	0.131071	LED – 50K	FPDD	56.36	50	**66**	**0.625000**	0.687500	**0.654762**
SEED	79.33	70	8891	0.010021	0.562500	0.019692		FSDD	47.65	41	145	0.450758	0.743750	0.561321
FHDDM	38.36	**26**	277	0.326034	**0.837500**	0.469352		FTDD	55.69	51	80	0.576720	0.681250	0.624642
DDM	63.48	5	85	0.645833	0.968750	0.775000		STEPD	10.63	**0**	649	0.197775	**1.000000**	0.330237
ECDD	**10.06**	2	2456	0.060444	0.987500	0.113915	Mixed – 50K	FPDD	18.38	**0**	41	0.796020	**1.000000**	0.886427
SEED	31.06	**0**	833	0.161128	**1.000000**	0.277537		FSDD	18.38	**0**	42	0.792079	**1.000000**	0.883978
FHDDM	19.94	**0**	**4**	**0.975610**	**1.000000**	**0.987654**		FTDD	18.38	**0**	41	0.796020	**1.000000**	0.886427
DDM	130.00	90	108	0.217391	0.250000	0.232558		STEPD	56.24	27	789	0.105442	0.775000	0.185629
ECDD	N/A	120	**0**	0.000000	0.000000	0.000000	RBF – 50K	FPDD	59.18	23	40	**0.708029**	0.808333	**0.754864**
SEED	92.61	**5**	624	0.155616	**0.958333**	0.267753		FSDD	54.32	25	64	0.597484	0.791667	0.681004
FHDDM	**45.20**	20	727	0.120919	0.833333	0.211193		FTDD	55.67	23	46	0.678322	0.808333	0.737643
DDM	70.06	**0**	165	0.492308	**1.000000**	0.659794		STEPD	11.84	2	489	0.244204	0.987500	0.391574
ECDD	**10.27**	10	3293	0.043567	0.937500	0.083264	Sine – 50K	FPDD	17.56	**0**	11	0.935673	**1.000000**	0.966767
SEED	36.75	3	1066	0.128373	0.981250	0.227043		FSDD	17.56	**0**	11	0.935673	**1.000000**	0.966767
FHDDM	20.50	**0**	**0**	**1.000000**	**1.000000**	**1.000000**		FTDD	17.56	**0**	11	0.935673	**1.000000**	0.966767
DDM	244.31	88	**88**	0.450000	0.450000	0.450000		STEPD	64.10	38	4885	0.024366	0.762500	0.047223
ECDD	41.43	48	2987	0.036141	0.700000	0.068733	LED – 100K	FPDD	65.20	35	115	**0.520833**	0.781250	**0.625000**
SEED	136.61	36	17,716	0.006951	0.775000	0.013778		FSDD	58.37	31	264	0.328244	0.806250	0.466546
FHDDM	**39.40**	**26**	634	0.174479	**0.837500**	0.288793		FTDD	66.35	34	131	0.490272	0.787500	0.604317
DDM	83.38	3	110	0.588015	0.981250	0.735363		STEPD	11.88	**0**	1350	0.105960	**1.000000**	0.191617
ECDD	**9.81**	1	5085	0.030320	0.993750	0.058845	Mixed – 100K	FPDD	16.88	**0**	23	0.874317	**1.000000**	0.932945
SEED	40.00	**0**	1594	0.091220	**1.000000**	0.167189		FSDD	16.81	**0**	24	0.869565	**1.000000**	0.930233
FHDDM	19.88	**0**	**4**	**0.975610**	**1.000000**	**0.987654**		FTDD	16.88	**0**	23	0.874317	**1.000000**	0.932945
DDM	229.19	83	140	0.209040	0.308333	0.249158		STEPD	73.96	19	1550	0.061175	0.841667	0.114060
ECDD	**25.22**	74	**39**	**0.541176**	0.383333	0.448780	RBF – 100K	FPDD	89.29	36	75	0.528302	0.700000	**0.602151**
SEED	107.71	11	1112	0.089271	0.908333	0.162565		FSDD	76.92	29	109	0.455000	0.758333	0.568750
FHDDM	44.11	**8**	1480	0.070352	**0.933333**	0.130841		FTDD	72.71	35	87	0.494186	0.708333	0.582192
DDM	91.31	**0**	133	0.546075	**1.000000**	0.706402		STEPD	**11.82**	1	884	0.152445	0.993750	0.264339
ECDD	14.70	11	6658	0.021889	0.931250	0.042773	Sine – 100K	FPDD	17.06	**0**	10	0.941176	**1.000000**	0.969697
SEED	40.94	**0**	2190	0.068085	**1.000000**	0.127490		FSDD	17.06	**0**	10	0.941176	**1.000000**	0.969697
FHDDM	21.00	**0**	**0**	**1.000000**	**1.000000**	**1.000000**		FTDD	17.06	**0**	10	0.941176	**1.000000**	0.969697
DDM	271.50	53	**167**	**0.390511**	0.668750	**0.493088**		STEPD	79.86	14	8819	0.016286	0.912500	0.032000
ECDD	60.83	27	6125	0.021253	0.831250	0.041446	LED – 200K	FPDD	95.21	43	318	0.268966	0.731250	0.393277
SEED	223.31	**9**	34,610	0.004344	**0.943750**	0.008648		FSDD	64.72	33	513	0.198438	0.793750	0.317500
FHDDM	**56.58**	14	1169	0.111027	0.912500	0.197966		FTDD	93.56	42	378	0.237903	0.737500	0.359756
DDM	110.89	3	93	0.628000	0.981250	0.765854		STEPD	10.56	**0**	2725	0.055459	**1.000000**	0.105090
ECDD	**9.74**	4	10,148	0.015140	0.975000	0.029817	Mixed – 200K	FPDD	14.50	**0**	29	0.846561	**1.000000**	0.916905
SEED	40.25	**0**	2915	0.052033	**1.000000**	0.098918		FSDD	14.50	**0**	30	0.842105	**1.000000**	0.914286
FHDDM	19.94	**0**	**4**	**0.975610**	**1.000000**	**0.987654**		FTDD	14.50	**0**	29	0.846561	**1.000000**	0.916905
DDM	561.83	27	109	0.460396	0.775000	0.577640		STEPD	94.21	6	3075	0.035748	0.950000	0.068903
ECDD	**28.44**	56	280	0.186047	0.533333	0.275862	RBF – 200K	FPDD	94.34	7	**101**	**0.528037**	0.941667	**0.676647**
SEED	110.61	6	2079	0.051984	0.950000	0.098573		FSDD	84.64	8	159	0.413284	0.933333	0.572890
FHDDM	39.70	21	3044	0.031499	0.825000	0.060680		FTDD	86.35	**5**	121	0.487288	**0.958333**	0.646067
DDM	115.63	**0**	136	0.540541	**1.000000**	0.701754		STEPD	11.63	**0**	1488	0.097087	**1.000000**	0.176991
ECDD	**10.13**	4	13,281	0.011610	0.975000	0.022946	Sine – 200K	FPDD	16.44	**0**	15	0.914286	**1.000000**	0.955224
SEED	40.69	1	4501	0.034120	0.993750	0.065975		FSDD	16.44	**0**	15	0.914286	**1.000000**	0.955224
FHDDM	20.75	**0**	**0**	**1.000000**	**1.000000**	**1.000000**		FTDD	16.44	**0**	15	0.914286	**1.000000**	0.955224

positives with NB , but this was the result of not identifying any drifts at all. The three proposed methods also presented competitive results and DDM was often the next best method, but well behind the others. SEED , STEPD , and FHDDM were usually much further behind in these RandomRBF datasets. In addition, FSDD usually presented competitive results in most dataset versions, except for the LED datasets.

Tables 4 and 5 also present results regarding the evaluation of the methods using *Precision, Recall*, and F-Measure (F1) [18]. Precision is the proportion of predicted drifts that are existing drifts and is defined as $TP \,/\, (TP + FP)$. Recall, given by $TP \,/\, (TP + FN)$, is the proportion of the existing drifts that were correctly detected by each method. F1 is the harmonic mean of precision and recall, being defined as $2 \times Precision \times Recall / (Precision + Recall)$. In all of these, higher values indicate that the corresponding methods perform better.

On the basis of these three criteria, FPDD, FSDD, and FTDD were the best methods in Precision and F1 with both HT and NB , despite FHDDM delivering the top results in most Mixed and Sine datasets. In the case of Recall, in most tested datasets, several methods presented similar results, with the exception of DDM, which was generally the worst method, especially in the LED and RandomRBF datasets. Furthermore, the proposed detectors were consistently better than DDM, ECDD, SEED, and STEPD with both base classifiers.

Given the results of our experiments regarding the drift identifications and accuracy, reported in Section 6.1, we conclude that FPDD, FSDD, and FTDD were the best performing methods, with little difference between them, and FHDDM followed fairly close behind them.

6.3. Memory and run-time results and analysis

We have also computed the results of memory usage in bytes per second and average run-time in seconds of the eight methods using NB and HT as base learners, but the tables with the results are omitted to avoid excessive information.

Analyzing the aforementioned results, we observed that, in many datasets, the three methods proposed consume slightly more memory and run-time than the other methods tested, especially with HT. However, the absolute numbers are completely negligible for modern computers.

7. Conclusions

This article proposed an efficient implementation of the computationally expensive Fisher's Exact statistical test, which is commonly indicated for many problems based on 2×2 contingency tables. The provided implementation is based on an astute limitation of the factorial calculations needed in Fisher's Exact test and on their storage to reuse in repeated calculations. This implementation is by itself a worthy contribution because Fisher's Exact test can be effectively applied to many other problems as well.

In addition, using the given implementation, this work proposed three new methods to detect concept drifts in data streams, based on slightly different applications of Fisher's Exact test using a *recent* and an *older* window of data, the same arrangement as that adopted in STEPD.

More specifically, FPDD is a variation of STEPD using Fisher's Exact test when samples are small. FSDD adopts the chi-square test, instead of the test of equal proportions used in both STEPD and FPDD, together with Fisher's Exact test. Finally, FTDD uses only Fisher's Exact test to detect drifts.

Experiments were run to compare the proposed methods to five well-known and/or recent concept drift detectors, namely DDM, ECDD, SEED , FHDDM, and STEPD . These experiments used both Naive Bayes (NB) and Hoeffding Tree (HT) as base learners, included artificial dataset versions of four different generators, with both abrupt and gradual concept drifts, and also tested real-world datasets.

The results of these experiments suggested that the predictive accuracies of FPDD, FSDD, and FTDD were better than those of the other methods in most datasets and, were consequently the three top-ranked methods in accuracy with both base classifiers, though in different orders. Moreover, according to the Nemenyi post-hoc test, they were all significantly superior to STEPD , as well as to SEED and ECDD, with both NB and HT, and also better than DDM with NB . Finally, the accuracies of the proposed methods were marginally better in datasets with abrupt concept drifts than in the ones with gradual drifts.

Regarding drift detections, the experiments have shown that, in general, FPDD and FTDD were the methods with the lowest false positive identifications, while maintaining similar results in the number of false negatives and in the distance to the correct position of the identified drifts. FSDD presented a similar performance, with the exception of the LED datasets, whereas FHDDM did *not* perform well in the RandomRBF datasets.

Based on the results of our experiments, and also on the fact that the total consumption of memory and run-time are negligible even for modern entrance-level computers, we claim FPDD, FSDD, and FTDD were tied as the best performing concept drift detection methods tested, with FHDDM coming closely behind.

Future work includes experimenting with other combinations of statistical tests to detect concept drift. Also, it might be interesting to include different scenarios in the artificial datasets, with other frequencies of drifts and longer transition periods in the gradual concept drift datasets.

Finally, it is worth stating explicitly that we adopted an empirical approach. In addition, Fisher's Exact test and the three methods were implemented in the MOA framework. Their source codes will soon be freely available, permitting further experiments by other researchers.

Acknowledgments

This article is based on Danilo Cabral's M.Sc. work [13]. In addition, the authors would like to thank Bruno Maciel for his MOA script generator and results extraction tool. The latter is still under development, but greatly helped speed up the generation of scripts and the analysis of the results of the experiments.

References

[1] P. Armitage, G. Berry, J. Matthews, Statistical Methods in Medical Research, (4th ed.), Blackwell Science, Oxford, UK, 2002.
[2] S.H. Bach, M.A. Maloof, Paired learners for concept drift, in: Proceedings of 8th IEEE International Conference on Data Mining (ICDM'08), 2008, pp. 23–32. Pisa, Italy.
[3] M. Baena-Garcia, J. Del Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavaldà, R. Morales-Bueno, Early drift detection method, in: Proceedings of International Workshop on Knowledge Discovery from Data Streams, 2006, pp. 77–86.

[4] R.S.M. Barros, Advances in data stream mining with concept drift, 2017, Professorship (Full) Thesis. Centro de Informática, Universidade Federal de Pernambuco, Brazil.
[5] R.S.M. Barros, D.R.L. Cabral, P.M. Gonçalves Jr., S.G.T.C. Santos, RDDM: Reactive drift detection method, Expert Syst. Appl. 90 (C) (2017) 344–355.
[6] R.S.M. Barros, J.I.G. Hidalgo, D.R.L. Cabral, Wilcoxon rank sum test drift detector, Neurocomputing 275 (C) (2018) 1954–1963, doi:10.1016/j.neucom.2017.10.051.
[7] R.S.M. Barros, S.G.T.C. Santos, P.M. Gonçalves JR., A boosting-like online learning ensemble, in: Proceedings of IEEE International Joint Conference on Neural Networks (IJCNN), Vancouver, Canada, 2016, pp. 1871–1878.
[8] A. Bifet, R. Gavaldà, Learning from time-changing data with adaptive windowing, in: Proceedings of the 7th SIAM International Conference on Data Mining (SDM'07), Minneapolis, MN, USA, 2007, pp. 443–448.
[9] A. Bifet, G. Holmes, R. Kirkby, B. Pfahringer, MOA: Massive online analysis, J. Mach. Learn. Res. 11 (2010) 1601–1604.
[10] A. Bifet, G. Holmes, B. Pfahringer, Leveraging bagging for evolving data streams, in: Machine Learning and Knowledge Discovery in Databases, LNCS, 6321, Springer, 2010b, pp. 135–150.
[11] A. Bifet, J. Read, I. Žliobaitė, B. Pfahringer, G. Holmes, Pitfalls in benchmarking data stream classification and how to avoid them, in: Machine Learning and Knowledge Discovery in Databases, LNCS, 8188, Springer, 2013, pp. 465–479.
[12] A. Bluman, Elementary Statistics: A Step by Step Approach, 9th, McGraw-Hill, New York, USA, 2014.
[13] D.R.L. Cabral, Statistical tests and detection of concept drifts in data streams, Centro de Informática, Universidade Federal de Pernambuco, Brazil. In Portuguese, 2017 M.Sc. Dissertation.
[14] H. Chernoff, E.L. Lehmann, The use of maximum likelihood estimates in χ^2 tests for goodness of fit, Ann. Math. Stat. 25 (3) (1954) 579–586.
[15] T. Cormen, C. Leiserson, R. Rivest, C. Stein, Introduction to Algorithms, 3rd ed., The MIT Press, 2009.
[16] A.P. Dawid, Present position and potential developments: some personal views: statistical theory: the prequential approach, J. R. Stat. Soc. Ser. A 147 (2) (1984) 278–292.
[17] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.
[18] T. Fawcett, An introduction to roc analysis, Pattern Recognit. Lett. 27 (8) (2006) 861–874.
[19] R. Fisher, Statistical Methods for Research Workers, Biological Monographs and Manuals, Oliver and Boyd, London, England, 1934.
[20] I. Frías-Blanco, J. del Campo-Ávila, G. Ramos-Jiménez, R. Morales-Bueno, A. Ortiz-Díaz, Y. Caballero-Mota, Online and non-parametric drift detection methods based on Hoeffding's bounds, IEEE Trans. Knowl. Data Eng. 27 (3) (2015) 810–823.
[21] I. Frías-Blanco, A. Verdecia-Cabrera, A. Ortiz-Díaz, A. Carvalho, Fast adaptive stacking of ensembles, in: Proceedings of the 31st ACM Symposium on Applied Computing (SAC'16), Pisa, Italy, 2016, pp. 929–934.
[22] J. Gama, P. Medas, G. Castillo, P. Rodrigues, Learning with drift detection, in: Advances in Artificial Intelligence: SBIA 2004, LNCS, 3171, Springer, 2004, pp. 286–295.
[23] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, A. Bouchachia, A survey on concept drift adaptation, ACM Comput. Surv. 46 (4) (2014). 44:1–37.
[24] P.M. Gonçalves Jr., R.S.M. Barros, RCD: A recurring concept drift framework, Pattern Recognit. Lett. 34 (9) (2013) 1018–1025.
[25] P.M. Gonçalves Jr., S.G.T.C. Santos, R.S.M. Barros, D.C.L. Vieira, A comparative study on concept drift detectors, Expert Syst. Appl. 41 (18) (2014) 8144–8156.
[26] W. Hamalainen, Efficient discovery of the top-k optimal dependency rules with Fisher's Exact test of significance, in: Proceedings of 2010 IEEE International Conference on Data Mining (ICDM), Sydney, Australia, 2010, pp. 196–205.
[27] W. Hoeffding, Probability inequalities for sums of bounded random variables, J. Am. Stat. Assoc. 58 (1963) 13–30.
[28] D.T.J. Huang, Y.S. Koh, G. Dobbie, R. Pears, Detecting volatility shift in data streams, in: Proceedings of 2014 IEEE International Conference on Data Mining (ICDM), Shenzhen, China, 2014, pp. 863–868.
[29] D. Ienco, A. Bifet, I. Žliobaitė, B. Pfahringer, Clustering based active learning for evolving data streams, in: J. Fürnkranz, E. Hüllermeier, T. Higuchi (Eds.), Discovery Science (DS'13), LNCS, 8140, Springer, 2013, pp. 79–93.
[30] I. Katakis, G. Tsoumakas, I. Vlahavas, Tracking recurring contexts using ensemble classifiers: an application to email filtering, Knowl. Inf. Syst. 22 (3) (2010) 371–391.
[31] J.Z. Kolter, M.A. Maloof, Dynamic weighted majority: an ensemble method for drifting concepts, J. Mach. Learn. Res. 8 (2007) 2755–2790.
[32] T. Lane, C.E. Brodley, Approaches to online learning and concept drift for user identification in computer security, in: Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining, AAAI Press, Menlo Park, CA, 1998, pp. 259–263.
[33] Y. Lee, L. Wang, K. Ryu, A system architecture for monitoring sensor data stream, in: Proceedings of the 7th IEEE International Conference on Computer and Information Technology, 2007, pp. 1026–1031.
[34] B.I.F. Maciel, S.G.T.C. Santos, R.S.M. Barros, A lightweight concept drift detection ensemble, in: Proceedings of 27th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'15), Vietri sul Mare, Italy, 2015, pp. 1061–1068.
[35] J.H. McDonald, Handbook of Biological Statistics, (3rd Ed.), Sparky House Publishing, Baltimore, Maryland, USA, 2014.
[36] C. Mehta, N. Patel, A network algorithm for performing Fisher's Exact test in r x c contingency tables, J. Am. Stat. Assoc. 78 (382) (1983) 427–434.
[37] C.R. Mehta, The exact analysis of contingency tables in medical research, Stat. Methods Med. Res. 3 (2) (1994) 135–156, doi:10.1177/096228029400300203. PMID: 7952429.
[38] L.L. Minku, X. Yao, DDD: A new ensemble approach for dealing with concept drift, IEEE Trans. Knowl. Data Eng. 24 (4) (2012) 619–633.
[39] K. Nishida, K. Yamauchi, Detecting concept drift using statistical testing, in: Proceedings of 10th International Conference on Discovery Science (DS'07), in: LNCS, 4755, Springer, 2007, pp. 264–269.
[40] R. Pears, S. Sakthithasan, Y. Koh, Detecting concept change in dynamic data streams, Mach. Learn. 97 (3) (2014) 259–293.
[41] A. Pesaranghader, H. Viktor, Fast Hoeffding drift detection method for evolving data streams, in: Machine Learning and Knowledge Discovery in Databases, LNCS, 9852, Springer, 2016, pp. 96–111.
[42] S.W. Roberts, Control chart tests based on geometric moving averages, Technometrics 1 (3) (1959) 239–250.
[43] G.J. Ross, N.M. Adams, D.K. Tasoulis, D.J. Hand, Exponentially weighted moving average charts for detecting concept drift, Pattern Recognit. Lett. 33 (2) (2012) 191–198.
[44] G.J. Ross, D.K. Tasoulis, N.M. Adams, Sequential monitoring of a Bernoulli sequence when the pre-change parameter is unknown, Comput. Stat. 28 (2) (2013) 463–479.
[45] S.G.T.C. Santos, R.S.M. Barros, P.M. Gonçalves Jr., Optimizing the parameters of drift detection methods using a genetic algorithm, in: Proceedings of the 27th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'15), 2015, pp. 1077–1084. Italy.
[46] S.G.T.C. Santos, P.M. Gonçalves Jr., G.D.S. Silva, R.S.M. Barros, Speeding up recovery from concept drifts, in: Machine Learning and Knowledge Discovery in Databases, LNCS, 8726, Springer, 2014, pp. 179–194.
[47] J. Smailovic, M. Grčar, N. Lavrač, M. Žnidaršič, Stream-based active learning for sentiment analysis in the financial domain, Inf. Sci. 285 (2014) 181–203.
[48] F. Yates, Contingency tables involving small numbers and the x^2 test, Suppl. J. R. Stat. Soc. 1 (2) (1934) 217–235. URL: http://www.jstor.org/stable/2983604.
[49] I. Žliobaitė, M. Pechenizkiy, J. Gama, An overview of concept drift applications, in: N. Japkowicz, J. Stefanowski (Eds.), Big Data Analysis: New Algorithms for a New Society, Springer, 2016, pp. 91–114.

Danilo Rafael de Lima Cabral received his B.Sc. degree in Systems Analysis and Development from Universidade Estácio de Sá, Brazil, in 2014, and his M.Sc. degree in Computer Science from Universidade Federal de Pernambuco(UFPE), Brazil, in 2017. He works as IT Software Developer at UFPE since 2014 and he is also a Ph.D. student at UFPE. His main research areas are pattern recognition and machine learning, with special interest in concept drift and statistics.

Roberto Souto Maior de Barros received B.Sc. and M.Sc. degrees in Computer Science from Universidade Federal de Pernambuco (UFPE), Brazil, in 1985 and 1988, respectively, and his Ph.D. degree in Computing Science from The University of Glasgow, Scotland (UK), in 1994. From 1985 to 1995 he worked as systems analyst at UFPE and he is a full time Professor (full) and Researcher, also at UFPE, since 1995. His main research areas are software engineering, programming languages, XML, pattern recognition, and machine learning, with special interest in concept drift.