Continuous Performance Evaluation for Business Process Outcome Monitoring

Suhwan Lee[1(✉)], Marco Comuzzi[2], and Xixi Lu[1]

[1] Utrecht University, Utrecht, The Netherlands
{s.lee,x.lu}@uu.nl
[2] Ulsan National Institute of Science and Technology, Ulsan, Republic of Korea
mcomuzzi@unist.ac.kr

Abstract. While a few approaches to online predictive monitoring have focused on concept drift model adaptation, none have considered in depth the issue of performance evaluation for online process outcome prediction. Without such a continuous evaluation, users may be unaware of the performance of predictive models, resulting in inaccurate and misleading predictions. This paper fills this gap by proposing a framework for evaluating online process outcome predictions, comprising two different evaluation methods. These methods are partly inspired by the literature on streaming classification with delayed labels and complement each other to provide a comprehensive evaluation of process monitoring techniques: one focuses on real-time performance evaluation, i.e., evaluating the performance of the most recent predictions, whereas the other focuses on progress-based evaluation, i.e., evaluating the ability of a model to output correct predictions at different prefix lengths. We present an evaluation involving three publicly available event logs, including a log characterised by concept drift.

Keywords: predictive monitoring · process outcome · event stream

1 Introduction

The process mining research in recent years has started focusing on the *online* realisation of typical use cases, such as process discovery [5] and conformance checking [4]. In the *online* perspective, an event log is a stream of events, which become available for analysis as soon as they are logged. Conversely, the traditional *offline* perspective considers an event log as a batch of events logged in a certain time span.

On the one hand, the online perspective naturally brings some benefits: online models need not waiting for a large number of events to be accumulated in an event log before performing an analysis; they also allow updating the analytic models in real time when a new event is received, and, consequently, they may naturally adapt to concept drift in the process generating the events [13]. On the other hand, this perspective also poses a number of challenges: new techniques

J. Munoz-Gama and X. Lu (Eds.): ICPM 2021 Workshops, LNBIP 433, pp. 237–249, 2022.
https://doi.org/10.1007/978-3-030-98581-3_18

must be developed to adapt to the streaming nature of events; owing to the finite memory assumption of streaming analytics, only a limited number of recent events can be available for the analysis at any given time [8]; finally, run time may become a concern, since models may need to be updated with every new event received and before the next event will be received.

This paper focuses on the predictive monitoring use case in process mining and, more specifically, on the continuous evaluation of the predictions of process outcomes [16], whereby the objective is to predict the (usually binary) outcome label of a running process case and to continuously evaluate these predictions. For instance, the possible outcome of a case would be that the personal loan request is accepted or rejected in a loan application process.

In the offline perspective, the outcome prediction problem is solved by encoding the completed cases into feature-label vectors, which are then used to train and test a predictive classification model. Besides the obvious need to consider online classification techniques for developing the predictive model, in the online perspective the outcome prediction is an instance of the *delayed labels* [10] online classification task: while in the batch perspective all the feature vectors and labels of completed cases are available for training and testing, in the online perspective the label of a case normally becomes available only when the last event of that case is received. This is an issue to be taken into account when updating the predictive model and, consequently, to assess its performance.

The contribution of the paper is to develop two performance evaluation methods specifically-tailored to online outcome predictive monitoring. These methods are developed adapting the notion of *continuous evaluation* [8], which recently has emerged as a novel perspective for evaluating the performance in streaming classification with delayed labels, to the domain of process outcome prediction.

The paper is organised as follows. Related work is discussed in the next section. Section 3 introduces the overall framework, while the performance methods are presented in Sect. 4. The experimental results are reported in Sect. 5, while conclusions are drawn in Sect. 6.

2 Related Work

Several approaches recently have been proposed to deal with online process discovery [2,6] and online conformance checking [4,17].

As far as process predictive monitoring is concerned, Maisenbacher and Weidlich [13] have proposed to use incremental classifiers to deal with an event log as a stream of events, specifically aiming at creating outcome prediction models that can adapt to concept drift. They propose to evaluate the models using average accuracy across all the labels received in the stream and they evaluate their approach on different concept drifts injected in a single artificially-generated event log. Baier et al. [1] have investigated the issue of optimal data selection point for retraining an offline predictive model when a concept drift is observed in an event stream.

In the more general field of streaming classification, Žliobaitė [18] first has identified the issue of delayed labels, suggesting to map dynamically the distribution of the labels to detect the concept drift. Grzenda et al. [8] recently have introduced the continuous evaluation methodology for streaming classification with delayed labels, whereby the performance of a model is evaluated for each observation considering the amount of time left before the arrival of the corresponding label.

3 Continuous Prediction Evaluation Framework

Given the first n positive natural numbers $\mathbb{N}_n^+$ and a target set S, a sequence s is a function $s : \mathbb{N}_n^+ \to S$ mapping integer indexes to the elements of S. Given a set of activity labels A, the domain $\mathbb{N}^+$ of timestamps, and a set of I attribute domains D_i, we define the set of event attributes as $E = A \times \mathbb{N}^+ \times [D_1 \times \ldots \times D_i \times \ldots \times D_I]$. A trace σ is a sequence of n events $\sigma : \mathbb{N}_n^+ \to E$. We denote with $\mathcal{T}$ the universe of sequences of events and with $\mathcal{E}$ the event universe, with $\mathcal{E} = E \times J$, where J is a set of possible case ids. An event stream is an infinite sequence $\Psi : \mathbb{N}^+ \to \mathcal{E}$.

For simplicity, we write events as $e_{k,j}$, where k indicates their position in a trace and traces as $\sigma_j = \langle e_{1,j}, \ldots, e_{i,j}, \ldots, e_{N_j,j} \rangle$, where N_j is the number of events in the trace σ_j. The function $t : \mathcal{E} \to \mathbb{N}^+$ returns the timestamp of an event. The prefix function $pref : \mathcal{T} \times \mathbb{N}^+ \to \mathcal{T}$ returns the first p events of a trace, i.e., $pref(\sigma_j, p) = \langle e_{1,j}, \ldots, e_{p,j} \rangle$, with $p \leq N_j$. Note that, for the evaluation, event streams are generated from event logs in which multiple events may have the same timestamp. For the events that have the same timestamp, we assume that the ordering of the events in an event log reflects their true ordering and use this order in the stream to calculate prefixes.

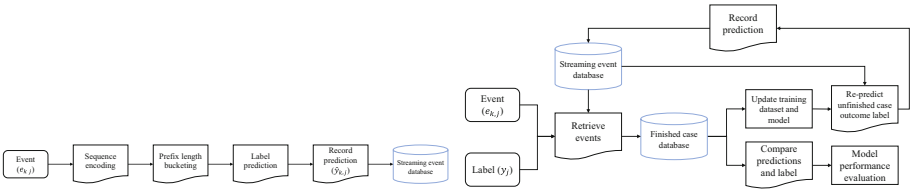
A trace σ is associated with a binary outcome label and, without loss of generality, we assume that the value of this label becomes known with the last event $e_{N_j,j}$ of a trace. Therefore, we define a labelling function as a partial function $y : E \nrightarrow \{0, 1\}$, which returns the label of a trace in correspondence of its last event. For clarity and with an abuse of notation, we denote the label of a trace σ_j as y_j.

A sequence encoder is a function f, with $f : \mathcal{T} \to \mathcal{X}_1 \times \ldots \times \mathcal{X}_w \times \ldots \times \mathcal{X}_W$ mapping a prefix into a set of features defined in the domains $\mathcal{X}_w$. A process outcome prediction model pom is a function $\hat{y} : \mathcal{X}_1 \times \ldots \times \mathcal{X}_w \times \ldots \times \mathcal{X}_W \to \{0, 1\}$ mapping an encoded prefix into its predicted label.

In offline settings, prefixes may be divided into separate buckets and a different prediction model may be maintained (trained/tested) for each bucket of prefixes. We adopt the same design in this work considering prefix-length bucketing [11] of traces: a different predictive model pom_k is trained and tested using a set of prefixes of length $k = 1, \ldots, K$, where the maximum prefix K may vary for each event log. Thus, we define an outcome prediction framework pof as a collection of outcome prediction models pom_k, that is, $pof = \{pom_k\}_{k=1,\ldots,K}$. We use index-based encoding of prefixes [11], in which features in a prefix are generated for each event in it. We use one-hot encoding for the categorical attributes, such

as the activity or the resource label, whereas continuous values are encoded as is. As classifiers, we consider incremental streaming classifiers that can be updated when a new label is received [9].

The processing of one event $e_{k,j}$ belonging to trace σ_j is schematised in Fig. 1. Note that this way of processing events applies after a given grace period, which is defined by a specific number L of labels received. That is, during the grace period, the labels received are only used to train the models in the framework. The event $e_{k,j}$ may either be the last of σ_j, i.e., $k = N_j$, in which case the label y_j becomes known, or not. When an event is not the last one of its trace (see Fig. 1a), it is used to generate a new prefix $pref(\sigma_j, k)$. Then, a prediction $\hat{y}_{k,j}$ for the new prefix $pref(\sigma_j, k)$ can be computed using the model pom_k. Receiving the last event $e_{k,j}$, with $k = N_j$ of a trace σ_j and its label (see Fig. 1b) enables (i) to evaluate all the predictions $\hat{y}_{n,j}$ that have been generated for the prefixes $pref(\sigma_j, n)$, with $n = 1, \ldots, \max\{K, N_j\}$ using the model pom_n (evaluation before training) and (ii) to update the models pom_n, with $n = 1, \ldots, \max\{K, N_j\}$ in the framework, owing to the availability of new labelled prefixes. Finally, it is possible (iii) to compute a new set of predicted labels $\hat{y}_{l,k}$, with $l \neq j$ and $k = 1, \ldots, \max\{N_l, N_j\}$ for all the prefixes for which a label has not been yet received (train and retest).



(a) Processing of an event without label (b) Processing of an event with label

Fig. 1. An overview of the continuous evaluation framework

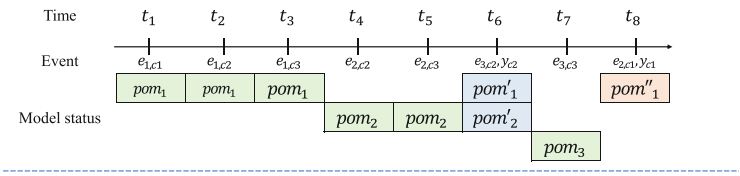
Next, we propose the novel methods to evaluate the performance of an online outcome prediction framework pof.

4 Performance Evaluation Methods

One of the major challenges in streaming classification is the performance evaluation, particularly in cases, such as the one of online process outcome prediction, in which the labels are *delayed*. The challenge arises because of the dynamic nature of the classification models considered in the framework: the models available to generate predictions are updated with each new label received; therefore, the same observation may be associated with different predictions generated by different versions of the model that applies to it.

Figure 2 exemplifies what stated above in the context of the proposed framework, considering 3 process cases and prefix length up to 3. First, note that

different versions of the same model pom_k are generated along the considered timeline. In particular, a new version of pom_k is generated when a new label y_j for a case σ_j, with $N_j < k$, is received. Second, new predictions for prefixes of length k are generated each time a new version of pom_k is available. Finally, note that a prediction can only be evaluated when the corresponding label becomes available. In the example, the predictions generated for all the prefixes of case 3 cannot be evaluated because the label of case 3 has yet to be received at t_8.



Label prediction		Timeline								
Case ID	Prefix length	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	$y(\sigma_j)$
1	1	$\hat{y}_{1,c1}$=0	>>	>>	>>	>>	$\hat{y}'_{1,c1}$=0	>>	-	1
2	1		$\hat{y}_{1,c2}$=0	>>	>>	>>	-	-	-	0
	2				$\hat{y}_{2,c2}$=1	>>	-	-	-	
3	1			$\hat{y}_{1,c3}$=0	>>	>>	$\hat{y}'_{1,c3}$=1	>>	$\hat{y}''_{1,c3}$=1	Unknown
	2					$\hat{y}_{2,c3}$=1	$\hat{y}'_{2,c3}$=1	>>	>>	
	3							$\hat{y}_{3,c3}$=0	>>	

Fig. 2. Evaluation methods: supporting example

We propose two ways to approach the issue of performance evaluation of the proposed framework: using a local observation timeline within a process case or a real-time global perspective on recent process cases. The former is inspired by the literature on streaming classification with delayed labels [8], whereas the latter is a novel perspective that we argue is specifically tailored to the context of process outcome predictive monitoring.

4.1 Evaluating Performance Using a Local Timeline

The local timeline perspective on performance evaluation in streaming classification is also referred to as the *continuous evaluation* of a model [8]. In the context of process outcome predictive monitoring, it translates naturally into evaluating the performance along a timeline that establishes the *progress* of the execution of a case. The traditional view of case progress in predictive monitoring is the *prefix length*, i.e., measuring the progress of a case using the number of events that have occurred in it. Therefore, we define a continuous evaluation method by prefix length.

Continuous Evaluation by Prefix Length. The objective of the continuous evaluation by prefix length is to evaluate the performance of an online outcome

classification framework at each prefix, i.e., to answer the question "*How likely is the framework to output a correct prediction for a running trace at prefix length k?*".

The design of a suitable performance measure starts from aggregating the predictions available for a case at a given prefix length, in order to obtain one reference value for each trace for which a label has been received at each prefix length. Inspired by the literature on streaming classification with delayed labels [8], we aggregate multiple predictions using a majority rule. That is, given the set $\hat{Y}_{k,j} = \{\hat{y}^l_{k,j}\}_{l=1,\ldots,L}$ of L predictions available for trace σ_j at prefix length k, and given $\hat{Y}^o_{k,j} = \{y \in \hat{Y}_{k,j} : y = o\}$ as the set of predictions evaluating to o, with $o \in \{0, 1\}$, the aggregated prediction for σ_j at prefix k is:

$$\hat{y}^{agg}_{k,j} = \begin{cases} 1 & \text{if } |\hat{Y}^1_{k,j}| \geq |\hat{Y}^0_{k,j}| \\ 0 & \text{otherwise} \end{cases}$$

Once the multiple predictions for a case at a given prefix length have been aggregated, the performance can be evaluated using any of the standard confusion matrix-based performance measure for classification. For instance, given the accuracy $acc(\hat{y}_j)$ of an individual prediction for trace σ_j at any prefix length:

$$acc(\hat{y}_j) = \begin{cases} 1 & \text{if } \hat{y}_j = y_j \\ 0 & \text{otherwise} \end{cases}$$

the accuracy $acc_k(pof)$ of an outcome prediction framework pof at prefix length k is defined as:

$$acc_k(pof) = \frac{1}{J} \cdot \sum_{j=1}^{J} acc(\hat{y}^{agg}_{k,j})$$

where J is the number of traces in the stream (or labels received).

For example, in Fig. 2, let us consider only the traces $c1$ and $c2$, for which the label has been received. The most frequent prediction at prefix length $k = 1$ for both trace $c1$ and $c2$ is 0 (no predictions equal to 1 are available). Given that the label of $c1$ and $c2$ are 1 and 0, respectively, the accuracy of the framework at prefix length $k = 1$ is 0.5.

4.2 Real-Time Model Performance

This method for performance evaluation considers a global perspective on recent predictions obtained by the framework, answering the question "*How likely are the most recent prediction(s) obtained from a model to be eventually correct?*" Instead of aggregating the performance at given progress rates or prefix lengths for cases, in the real-time method we first define w as the size of a test window containing the traces $\{\sigma_w\}_{w=1,\ldots,W}$ associated with the latest W labels y_w that have been received. We then consider the average of the performance across all the predictions available, at any prefix length, for each trace σ_w in this window.

Table 1. Descriptive statistics of event logs used in the evaluation

	# cases	# events	# activity	# variants	Avg events/case	Median events/case	# true labels	# false labels
BPIC 2015_1	1199	52217	289	1100	43.55	44	506	693
BPIC 2017	1878	23941	22	376	12.75	12	576	1302
IRO5K	1000	10756	20	111	10.76	11	237	763

When a new label is received, then, to accommodate this new trace, the trace in the window associated with the oldest label received is removed from the window.

Given $\hat{Y}_w$ as the set of predictions $\hat{y}_{k,w}$ available for a trace σ_w at any prefix length k, the real-time accuracy $acc_{rt}(pof)$ of an outcome predictive framework is then defined as follows:

$$acc_{rt}(pof) = \frac{1}{W} \cdot \sum_{w=1}^{W} \left[\frac{1}{|\hat{Y}_w|} acc(\hat{y}_{k,w}) \right].$$

Let us consider $W = 2$ in the example of Fig. 2. At t_8, the traces $c1$ and $c2$ are included in the window, because they are associated to the last 2 labels received. For $c1$, there are 2 predictions available (at $t1$ and $t6$), all incorrect. For $c2$, there are 2 predictions available (one correct $t2$ and one incorrect at $t4$). Therefore, the real-time accuracy at t_8 for $W = 2$ of the framework is 0.25.

5 Experimental Analysis and Results

We consider 3 publicly accessible event logs. The BPIC 2015_1[1] is a log from a Dutch municipality of a process for granting building permissions. The outcome label in this log is 1 (true) when a trace contains the activity 'create procedure confirmation', and 0 otherwise. The BPIC 2017[2] event log refers to a personal loan request process at a Dutch financial institute. The outcome label evaluates to 1 (true) if a request is accepted, and 0 otherwise. The IRO5K[3] event log is a synthetic log regarding the assessment of loan applications [12]. The outcome label evaluates to 1 (true) if a request is accepted, and 0 otherwise. The two BPIC logs have been chosen because they are real world event logs that have been used in the previous research on outcome predictive monitoring [16] and they differ greatly in terms of variability. Specifically (see Table 1), the BPIC 2015 event log shows a higher number of activity labels and trace variants in respect of BPIC 2017. The IRO5K event log has been chosen because it is characterised by process drift.

[1] at: https://data.4tu.nl/articles/dataset/BPI_Challenge_2015_Municipality_1/12709154/1.
[2] at: https://data.4tu.nl/articles/dataset/BPI_Challenge_2017_-_Offer_log/12705737.
[3] at: https://data.4tu.nl/articles/dataset/Business_Process_Drift/12712436.

The process outcome prediction is an instance of early time series prediction and the research community focuses on building an accurate model for early predictions [16]. We consider a different maximum prefix length for each event log: 44 for BPIC 2015_1, 14 for BPIC 2017 and 11 for IRO5K. The minimum prefix length is set to 2 for all event logs.

As streaming classifiers, we consider 3 different tree-based incremental classifiers typically adopted in streaming classification: the Hoeffding Tree Classifier (HTC) [9], the Hoeffding Adaptive Tree Classifier (HATC) [3], and the Extremely Fast Decision Tree (EFDT) [14]. These algorithms are tree-based classifiers which incrementally construct split points depending on the confidence for information gain.

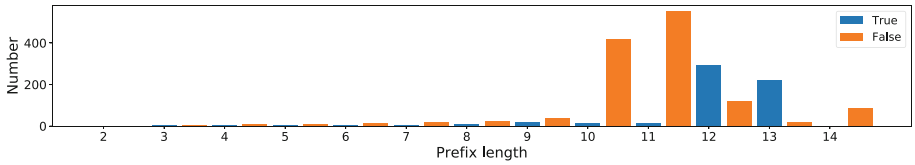
Any streaming classification framework normally requires a grace period to allow a proper initialisation of the classifier [7]. As grace period in all experiments, we consider 200 labels received. That is, until the 200-th label is received, the events without label in the stream are not processed and the labels are used only to update the classification models. We consider the implementation of the classifiers provided by the Python package River [15], setting 100 observations for the classification tree leaf between split attempts and maximising information gain as split criterion in all streaming classifiers.

For the real-time performance evaluation, we consider $W = 50$ cases as window size and we also include as a baseline the results obtained using an offline outcome predictive model developed using the Random Forest (RF) classifier, implemented using the Python package 'scikit-learn' with 100 estimators, using a 70/30 train/test split and 10-fold cross-validation. Finally, to support the discussion we also plot for each log the number of true and false labels received at each prefix length.

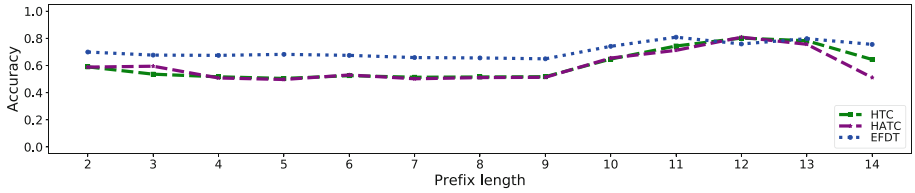
The code and data to reproduce the experiments presented in this section, as well as additional results that have been omitted in this section due to lack of space, are available at https://github.com/ghksdl6025/streaming_prediction4pm.

Figure 3b shows the continuous evaluation results for BPIC2017 using the prefix length method. Regarding the prefix length method, the EFDT shows a better performance than the other classifiers. Generally, the accuracy of the classification increases after prefix length 9.

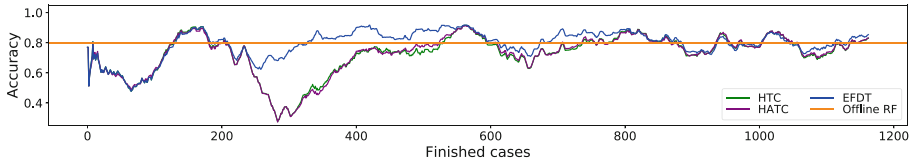
From the results, we can observe that the continuous evaluation method provides diagnostic information to help deciding which model to deploy. The prefix length method reveals that EFDT performs better than other classifiers for ongoing cases until prefix length 11. Therefore, the EFDT classifier should be preferred if the predictions obtained from the outcome decision framework are used to take the decision after an event in a case has occurred (e.g., "What's the best thing to do after the client has replied?"), and that event normally happens within the first 11 executed in a trace.



(a) Number of labels received by prefix length



(b) Accuracy by prefix length acc_k



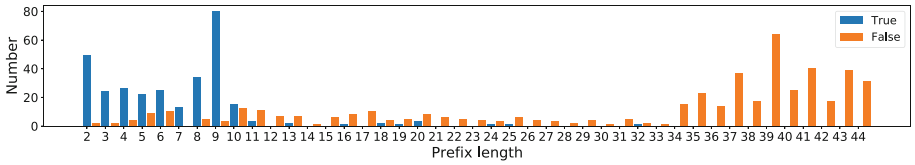
(c) Real-time accuracy acc_{rt} ($W = 50$)

Fig. 3. BPIC 2017 log experiment results.

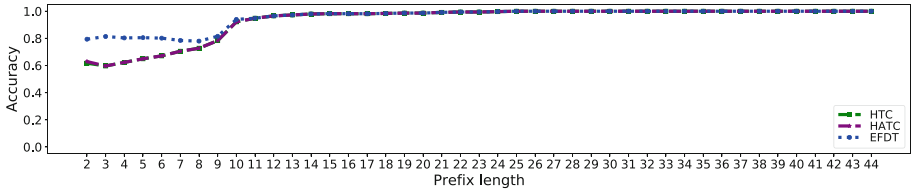
Figure 3c shows the real-time perspective results for BPIC2017. Except for a sharp drop in the accuracy of HATC and HTC after the 200-th label received, the accuracy remains above 0.6 and comparable with the one of the offline baseline. From a diagnostic standpoint, the real-time performance perspective generally reveals whether the predictive framework outputs correct predictions for all cases *now*, on cases recently finished. It also can provide diagnostic information when there is a sudden change in the process, showing how each model performs after such a change.

For the BPIC2015_1 log (see Fig. 4), let us first consider the real-time evaluation method (Fig. 4c). We observe that after approximately 350 labels received, the performance of all models drops significantly. This may be caused by an (unknown) concept drift in the event log. After 600 cases, we see that EFDT recovers whereas the performance of the other two models does not improve until the end. Therefore, also in this case EFDT emerges as more likely to recover after a drop in the performance than HATC and HTC.

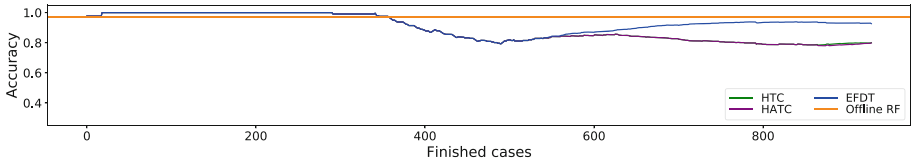
For the IRO5K event log, Fig. 5b shows the results of the continuous evaluation by prefix length. The results for this log must be interpreted considering the distribution of labels received across prefix lengths. Until prefix length 6 no new labels are received, which justifies the constant high accuracy by prefix length until then (given that the models trained during the grace period are



(a) Number of labels received by prefix length



(b) Accuracy by prefix length acc_k

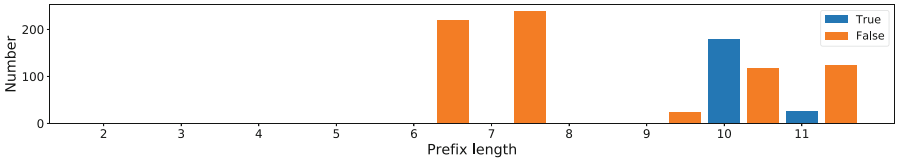


(c) Real-time accuracy acc_{rt} ($W = 50$)

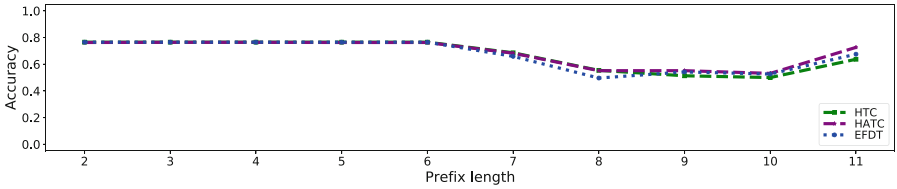
Fig. 4. BPIC 2015_1 log experiment results.

fairly accurate). Then most false labels are received at prefix length 6 and 7, whereas no true labels are received before prefix 10. Therefore, the models used to generate predictions change substantially after prefix 6, which may justify the drop in performance.

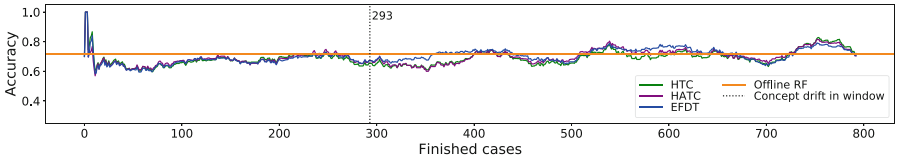
More insightful for this event log is the analysis of the real-time performance, which is shown in Fig. 5c. In particular, there is no specific drop of the accuracy when the process drift occurs. The EFDT classifier, in particular, actually increases its accuracy after the concept drift. Generally, after the concept drift occurs the performance of all classifiers recovers relatively quickly (in less than 100 labels received) and, until the end of the stream, remains higher for most time in respect of the offline baseline. This can be interpreted as encouraging evidence that the proposed framework with the EFDT model, at least in this case, naturally adapts to concept drift in the process generating the event stream.



(a) Number of labels received by prefix length



(b) Accuracy by prefix length acc_k



(c) Real-time accuracy acc_{rt} ($W = 50$); the concept drift occurs after the 293-th label is received.

Fig. 5. IRO5K log experiment results.

6 Conclusions

In this paper, we propose a continuous performance evaluation framework for online process outcome prediction techniques. Moreover, we propose two concrete evaluation methods, which assess the performance of the prediction techniques from both a local perspective and a global real-time perspective. The experimental analysis of our framework on the three real-life logs has shown that our framework can reveal very interesting results from different perspectives and provide novel insights into how predictive models perform.

As far as the experimental analysis is concerned, the streaming classifier EFDT has emerged as the best performing and robust classifier for outcome prediction with event streams. This confirms the claim of the proposers of the EFDT classifier that it should be preferred to other incremental tree classifiers based on the Hoeffding bound in most application scenarios [14]. Unexpectedly, although HATC is specifically designed to adapt to concept drift, our evaluation framework shows that EFDT appears as the best classifier at dealing with concept drifts in the event logs. This may be due to the window selection of EFDT, which simply adapts quickly to the new data, whereas in the other two models (HATC and HTC) there is a threshold controlling the model adaptation, which may limit the ability to adapt to small changes in the feature vectors.

As a future work, the proposed framework can be extended with various performance evaluation methods. For example, instead of the prefix-length perspective, we may also complement the framework with a method that evaluates the accuracy from the last-state perspective. Providing performance evaluation from multiple perspectives may help ease the issue of explainability of online predictive monitoring models, which should also be further investigated.

References

1. Baier, L., Reimold, J., Kühl, N.: Handling concept drift for predictions in business process mining. In: 2020 IEEE 22nd Conference on Business Informatics (CBI), vol. 1, pp. 76–83. IEEE (2020)
2. Batyuk, A., Voityshyn, V.: Streaming process discovery for lambda architecture-based process monitoring platform. In: 2018 IEEE 13th International Scientific and Technical Conference on Computer Sciences and Information Technologies (CSIT), vol. 1, pp. 298–301. IEEE (2018)
3. Bifet, A., Gavaldà, R.: Adaptive learning from evolving data streams. In: Adams, N.M., Robardet, C., Siebes, A., Boulicaut, J.-F. (eds.) IDA 2009. LNCS, vol. 5772, pp. 249–260. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03915-7_22
4. Burattin, A., Carmona, J.: A framework for online conformance checking. In: Teniente, E., Weidlich, M. (eds.) BPM 2017. LNBIP, vol. 308, pp. 165–177. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-74030-0_12
5. Burattin, A., Sperduti, A., van der Aalst, W.M.: Heuristics miners for streaming event data. arXiv preprint: arXiv:1212.6383 (2012)
6. Burattin, A., Sperduti, A., van der Aalst, W.M.: Control-flow discovery from event streams. In: 2014 IEEE Congress on Evolutionary Computation (CEC), pp. 2420–2427. IEEE (2014)
7. Domingos, P., Hulten, G.: Mining high-speed data streams. In: Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 71–80 (2000)
8. Grzenda, M., Gomes, H.M., Bifet, A.: Delayed labelling evaluation for data streams. Data Mining Knowl. Disc. **34**(5), 1237–1266 (2019). https://doi.org/10.1007/s10618-019-00654-y
9. 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, pp. 97–106 (2001)
10. Krempl, G., et al.: Open challenges for data stream mining research. ACM SIGKDD Explor. Newsl. **16**(1), 1–10 (2014)
11. Leontjeva, A., Conforti, R., Di Francescomarino, C., Dumas, M., Maggi, F.M.: Complex symbolic sequence encodings for predictive monitoring of business processes. In: Motahari-Nezhad, H.R., Recker, J., Weidlich, M. (eds.) BPM 2015. LNCS, vol. 9253, pp. 297–313. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-23063-4_21
12. Maaradji, A., Dumas, M., La Rosa, M., Ostovar, A.: Fast and accurate business process drift detection. In: Motahari-Nezhad, H.R., Recker, J., Weidlich, M. (eds.) BPM 2015. LNCS, vol. 9253, pp. 406–422. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-23063-4_27

13. Maisenbacher, M., Weidlich, M.: Handling concept drift in predictive process monitoring. SCC **17**, 1–8 (2017)
14. Manapragada, C., Webb, G.I., Salehi, M.: Extremely fast decision tree. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1953–1962 (2018)
15. Montiel, J., et al.: River: machine learning for streaming data in python (2020)
16. Teinemaa, I., Dumas, M., Rosa, M.L., Maggi, F.M.: Outcome-oriented predictive process monitoring: review and benchmark. ACM Trans. Knowl. Disc. Data (TKDD) **13**(2), 1–57 (2019)
17. van Zelst, S.J., Bolt, A., Hassani, M., van Dongen, B.F., van der Aalst, W.M.P.: Online conformance checking: relating event streams to process models using prefix-alignments. Int. J. Data Sci. Anal. **8**(3), 269–284 (2017). https://doi.org/10.1007/s41060-017-0078-6
18. Žliobaite, I.: Change with delayed labeling: when is it detectable? In: 2010 IEEE International Conference on Data Mining Workshops, pp. 843–850. IEEE (2010)