Knowledge and Information Systems (2022) 64:1385–1416
https://doi.org/10.1007/s10115-022-01666-9

REGULAR PAPER

How do I update my model? On the resilience of Predictive Process Monitoring models to change

Williams Rizzi[1,2] · Chiara Di Francescomarino[1] · Chiara Ghidini[1] · Fabrizio Maria Maggi[2]

Received: 21 May 2021 / Revised: 16 February 2022 / Accepted: 17 February 2022 / Published online: 21 March 2022

Abstract
Existing well-investigated Predictive Process Monitoring techniques typically construct a predictive model based on past process executions and then use this model to predict the future of new ongoing cases, without the possibility of updating it with new cases when they complete their execution. This can make Predictive Process Monitoring too rigid to deal with the variability of processes working in real environments that continuously evolve and/or exhibit new variant behaviours over time. As a solution to this problem, we evaluate the use of three different strategies that allow the periodic rediscovery or incremental construction of the predictive model so as to exploit new available data. The evaluation focuses on the performance of the new learned predictive models, in terms of accuracy and time, against the original one, and uses a number of real and synthetic datasets with and without explicit Concept Drift. The results provide an evidence of the potential of incremental learning algorithms for predicting process monitoring in real environments.

Keywords Predictive Process Monitoring · Incremental Learning · Concept Drift · Process Mining

✉ Williams Rizzi
wrizzi@fbk.eu

Chiara Di Francescomarino
dfmchiara@fbk.eu

Chiara Ghidini
ghidini@fbk.eu

Fabrizio Maria Maggi
maggi@inf.unibz.it

1 Fondazione Bruno Kessler (FBK), Trento, Italy

2 Free University of Bozen-Bolzano, Bolzano, Italy

1 Introduction

Predictive Process Monitoring Maggi et al. [25] is a research topic aiming at developing techniques that use the abundant availability of event logs extracted from information systems in order to predict how ongoing (uncompleted) process executions (a.k.a. cases) will unfold up to their completion. In turn, these techniques can be embedded within information systems to enhance their ability to manage business processes. For example, an information system can exploit a predictive monitoring technique to predict the remaining execution time of each ongoing case of a process Rogge-Solti and Weske [35], the next activity that will be executed in each case Evermann et al. [15], or the final outcome of a case w.r.t. a set of possible outcomes Maggi et al. [25] Metzger et al. [27], Metzger et al. [28].

Existing Predictive Process Monitoring techniques first construct a predictive model based on data coming from past process executions. Then, they use this model to predict the future of an ongoing case (e.g. outcome, remaining time, or next activity). However, when the predictive model has been constructed, it won't automatically take into account new cases when they complete their execution. This is a limitation in the usage of predictive techniques in the area of Business Process Monitoring: well-known characteristics of real processes are, in fact, their complexity, variability, and lack of steady state. Due to changing circumstances, processes (and thus their executions) evolve and increase their variability, and systems need to adapt in a timely manner.

While a rough answer to this problem would be the one of re-building new predictive models from the wider available set of data, one could observe that building predictive models has a cost and this option should therefore be well understood before embracing it; moreover, preliminary studies such as the one of Maisenbacher and Weidlich [26] investigate the usage of incremental techniques Gepperth and Hammer [19] in the presence of Concept Drift phenomena, thus suggesting a diverse strategy of updating a Predictive Process Monitoring model.

In this paper, we tackle the problem of updating Predictive Process Monitoring models in the presence of new process execution data in a principled manner by investigating, in a comparative and empirically driven manner, how different strategies to keep predictive models up-to-date work. In particular, given an event log $\mathcal{TR}_0$, and a set of new traces $\mathcal{TR}_1$, we focus on four diverse strategies to update a predictive model $\mathcal{M}_0$ built using the traces of $\mathcal{TR}_0$, to also take into account the set of new traces $\mathcal{TR}_1$:

- **Do nothing**. In this case, $\mathcal{M}_0$ is never updated and does not take into account $\mathcal{TR}_1$ in any way. This strategy acts also as a baseline against which to compare all the other strategies.
- **Re-train with no hyperopt**. In this case, a new predictive model $\mathcal{M}_1$ is built using $\mathcal{TR}_0 \cup \mathcal{TR}_1$ as train set but no optimisation of the hyperparameters is performed and the ones of $\mathcal{M}_0$ are used;
- **Full re-train**. In this case, a new predictive model $\mathcal{M}_2$ is built using $\mathcal{TR}_0 \cup \mathcal{TR}_1$ as the train set and a new optimisation of the hyperparameters is performed;
- **Incremental update**. In this case, a new predictive model $\mathcal{M}_3$ is built starting from $\mathcal{M}_0$ using the cases contained in $\mathcal{TR}_1$ in an incremental manner (that is, using incremental learning algorithms).

The evaluation aims at investigating two main aspects of these update strategies: first, their impact on the quality[1] of the prediction in event logs that exhibit/do not exhibit Concept

[1] We measure the prediction quality using different metrics precisely defined later in the paper.

Drift; second, their impact on the time spent for building the predictive model $\mathcal{M}_0$ and their updates.

The reason why we provide two different strategies for re-training the model, i.e. with no optimisation and with an optimisation of the hyperparameters, is because the two costly activities when building a predictive model are the actual training of the model w.r.t. a train set and the optimisation of the hyperparameters for the constructed model. Therefore, when evaluating the impact of re-training on an extended set of data, we aim at investigating the impact of building a new predictive model and the impact of optimising the hyperparameter in a separate manner.

The problem upon which we investigate these four strategies is the one of outcome predictions, where the outcomes are expressed by using either Linear Temporal Logic (LTL) formulae Pnueli [32], in line with several works such as Di Francescomarino et al. [12], Maggi et al. [25], or case duration properties. The four different strategies are evaluated in a broad experimental setting that considers different real and synthetic datasets. Since we focus on outcome predictions, we have decided to centre our evaluation on Random Forest. This algorithm was chosen as it was experimentally proven to be one of the best performing techniques on the outcome prediction problem—see Teinemaa et al. [40] for a rigorous review—and is therefore widely used on event log data usually used in Predictive Process Monitoring.

Perhaps not surprisingly, the results show that the do-nothing strategy is not a viable strategy (and therefore the issue of updating a Predictive Process Monitoring model is a real issue) and that full re-training and incremental updates are the best strategy in terms of quality of the updated predictive model. Nonetheless, the incremental update is able to keep up with the re-training strategy and deliver a properly fitted model almost in real time, whereas the full re-training might take hours and in some cases even days, suggesting that the potential of incremental models is under-appreciated, and clever solutions could be applied to deliver more stable performance while retaining the positive side of the *update* functions.

The rest of the paper is structured as follows: Section 2 provides the necessary background on Predictive Process Monitoring and incremental learning; Sect. 3 presents two exemplifying scenarios of process variability and explicit Concept Drift; Sect. 5 illustrates the data and procedure we use to evaluate the proposed update strategies, while Sect. 6 presents and discusses the results. We finally provide some related work (Sect. 7) and concluding remarks (Sect. 8).

2 Background

In this section, we provide an overview of the four main building blocks that compose our research effort: Predictive Process Monitoring, Random Forest, hyperparameter optimisation, and Concept Drift.

2.1 Predictive Process Monitoring

Predictive Process Monitoring Maggi et al. [25] is a branch of Process Mining that aims at predicting at runtime and as early as possible the future development of ongoing cases of a process given their uncompleted traces. In the last few years, a wide literature about Predictive Process Monitoring techniques has become available—see Di Francescomarino, Ghidini , Maggi and Milani [13] for a survey—mostly based on Machine Learning techniques. The main dimension that is typically used to classify Predictive Process Monitoring techniques is the type of prediction, which can belong to one of the three macro-categories: numeric predictions (e.g. time or cost predictions); categorical predictions (e.g. risk predictions or specific categorical outcome predictions such as the fulfilment of a certain property); next activities predictions (e.g. the sequence of the future activities, possibly with their attributes).

Frameworks such as `Nirdizati` Rizzi et al. [34], Jorbina et al. [21] collect a set of Machine Learning techniques that can be instantiated and used for providing different types of predictions to the user. In detail, these frameworks take as input a set of past executions and use them to train predictive models, which can then be stored to be used at runtime to continuously supply predictions to the user. Moreover, the computed predictions can be used to compute accuracy scores for specific configurations. Within these frameworks, we can identify two main modules: one for the *case encoding*, and one for the *supervised learning*. Each of them can be instantiated with different techniques. Examples of case encodings are *index-based encodings* presented in Leontjeva et al. [22]. Supervised learning techniques instead vary and can also depend on the type of prediction a user is interested in, ranging from Decision Tree and Random Forest, to regression methods and Recurrent Neural Networks.

2.2 Random forest

Random Forest [20] is an ensemble learning method used for classification and regression. The goal is to create a model composed of a multitude of Decision Trees Quinlan [33]. In a Decision Tree, each tree interior node corresponds to one of the input variables and each leaf node to a possible classification or decision. Each different path from the root to the leaf represents a different configuration of input variables. A tree can be "learned" by bootstrapping the source set into subsets based on an attribute value test. This process is repeated on each derived subset in a recursive manner called recursive partitioning. The result is a tree in which each selected variable will contribute to the labelling of the relative example. When an example needs to be labelled, it is run through all the Decision Trees. The output of each Decision Tree is counted. The most occurring label will be the output of the model.

In this work, we use non-incremental and incremental versions of Random Forest. In a nutshell, the non-incremental version builds a predictive model once and for all using a specific set of training data in a single training phase. Instead, in addition to the step of building a predictive model during the training phase, the incremental learning versions are able to update such a model whenever needed through an update function. The specific implementation used in this work incrementally updates the starting model by adding new decision trees as soon as new data are available.

As already mentioned in Introduction, Random Forest was chosen as it was experimentally proven to be one of the best performing techniques on the outcome prediction problem in Predictive Process Monitoring. The interested reader is referred to Teinemaa et al. [40] for a rigorous review.

2.3 Hyperparameter optimisation

Machine Learning techniques are known to use model parameters and hyperparameters. Model parameters are automatically learned during the training phase so as to fit the data. Instead, hyperparameters are set outside the training procedure and used for controlling how flexible the model is in fitting the data. While the values of hyperparameters can influence the performance of the predictive models in a relevant manner, their optimal values highly depend on the specific dataset under examination, thus making their setting rather burdensome. To support and automatise this onerous but important task, several hyperparameter optimisation techniques have been developed in the literature Bergstra and Bengio [4], Bergstra et al. [3] also for Predictive Process Monitoring models—see, for example, Teinemaa et al. [40], Di Francescomarino, Dumas, Federici, Ghidini, Maggi, Rizzi and Simonetto [11], Teinemaa et al. [41]. While in Teinemaa et al. [40], Di Francescomarino, Dumas, Federici, Ghidini, Maggi, Rizzi and Simonetto [11], the Tree Parser Estimator (TPE) has been used for outcome-oriented Predictive Process Monitoring solutions, in Teinemaa et al. [41], Random Search has been used for hyperparameter optimisation in a comparative analysis focusing on how stable outcome-oriented predictions are along time. Although hyperparameter optimisation techniques for Predictive Process Monitoring have shown their ability to identify accurate and reliable framework configurations, they are also an expensive task and we have hence decided to evaluate the role of hyperparameter tuning in our update strategies.

2.4 Concept drift

In Machine Learning, *Concept drift* refers to a change over time, in unforeseen ways, of the statistical properties of a target variable a learned model is trying to predict. This drift is often due to changes in the target data w.r.t. the one that was used in the training phase. These changes are problematic as they cause the predictions to become less accurate as time passes. Depending on the type of change (e.g. gradual, recurring, or abrupt), different types of techniques have been proposed in the literature to detect and handle them Gama et al. [18], Widmer and Kubat [17], Schlimmer and Granger [37,49].

Business processes are subject to change due to, for example, changes in their normative, or organisational context, and so are their executions. Processes and executions can hence be subject to Concept Drifts, which may involve several process dimensions such as its control-flow dependencies and data handling. For instance, an organisational change might affect how a certain procedure is managed by employees in a Public Administration scenario (e.g. a further approval by a new manager is required for closing the procedure), or a normative change might affect either the way in which patients are managed in an emergency department (e.g. patients have to be tested for COVID-19 before they can be visited) or the age of the customers who are allowed to submit a loan request procedure in a bank process. The Concept Drift phenomenon has originated few works that focus on drift detection and localisation in procedural and in declarative business processes—see Bose et al. [6], Carmona and Gavaldà [10] and Maggi et al. [24], respectively—as well as on attempts to deal with it in the context of Predictive Process Monitoring Maisenbacher and Weidlich [26], Pauwels and Calders [31].

3 Two descriptive scenarios

We aim at assessing the benefits of incremental learning techniques in scenarios characterised by process variability and/or explicit Concept Drift phenomena. In this section, we introduce two typical scenarios, which refer to some of the datasets used in the evaluation described in Sect. 5.

Scenario 1 (Dealing with Process Variability) Information systems are widely used in health-care and several scenarios of predictive analytics can be provided in this domain. Indeed, the exploitation of predictive techniques in healthcare is described as one of the promising big data trends in this domain Bughin et al. [8], Munoz-Gama et al. [30].

Despite some successful evaluation of Predictive Process Monitoring techniques using healthcare data Maggi et al. [25], predictive monitoring needs to consider a well-known feature of healthcare processes, that is, their variability Rojas et al. [36], i.e. the variety of different alternative paths characterising the executions of a process. Whether they refer to non-elective care (e.g. medical emergencies), or elective care (e.g. scheduled standard, routine and non-routine procedures), healthcare processes often exhibit characteristics of high variability and instability. For instance, the treatment processes related to different patients can be quite different due to allergies or comorbidities or other specific characteristics of a patient. In fact, when attempting to discover process model from data related to these processes, they are often spaghetti-like, i.e. cumbersome models in which it is difficult to distil a stable procedure. Moreover, small changes in the organisational structure (e.g. new personnel in charge of a task, unforeseen seasonal variations due to holidays or diseases) may originate subtle variability not detectable in terms of stable Concept Drifts, but nonetheless relevant in terms of predictive data analytics.

In such a complex environment, an important challenge concerns the emergence of new behaviours: regardless of how much data we consider, an environment highly dependent on the human factor is likely to exhibit new variants that may not be captured when stopping the training at a specific time. Similarly, some variants may become obsolete, thus making the forgetting of data equally important.

Thus, a way for adapting the predictions to these changes and an investigation of which update strategies are especially suited to highly variable and realistic process executions would be of great impact.

Scenario 2 (Dealing with explicit Concept Drift) The presence of Concept Drift in business processes, due to, for example, changes in the organisational structures, legal regulations, and technological infrastructures, has been acknowledged in the Process Mining manifesto van der Aalst and et al. [44] and literature Maggi et al. [24], together with some preliminary studies on its relation with Predictive Process Monitoring Gepperth and Hammer [19].

Such a sudden and abrupt variation in the data provides a clear challenge to the process owners: they must be ready to cope with a Predictive Process Monitoring model with degraded performance on the drifted data, or to perform an update that allows the Predictive Process Monitoring technique to support both the non-drifted and the drifted trends of the data (as ongoing executions may still concern the non-drifted cases).

Similarly to the above, an investigation of which update strategies are especially suited to realistic process executions that exhibit an explicit Concept Drift would provide a concrete support for the maintenance of Predictive Process Monitoring models.

4 Update strategies

Predictive Process Monitoring provides a set of techniques that use the availability of execution traces (or cases) extracted from information systems in order to predict how ongoing (uncompleted) executions will unfold up to their completion.

Thus, assume that an organisation was able to exploit a set of process executions $\mathcal{TR}_0$, collected within a period of time that we will call "Period A" to obtain a predictive model $\mathcal{M}_0$, and that it starts to exploit $\mathcal{M}_0$ to perform predictions upon new incomplete traces (see Fig. 1 for an illustration of this scenario). As soon as the new incomplete traces terminate,

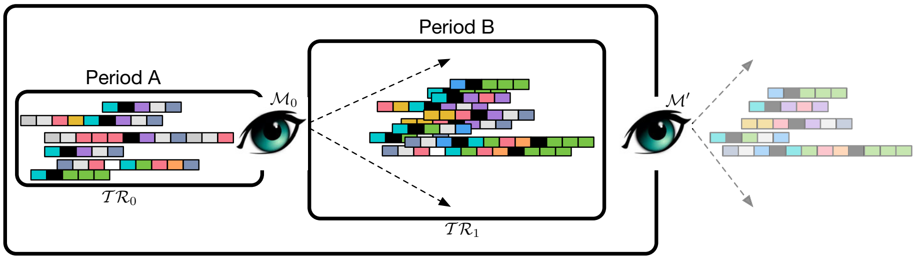


Fig. 1 The general idea

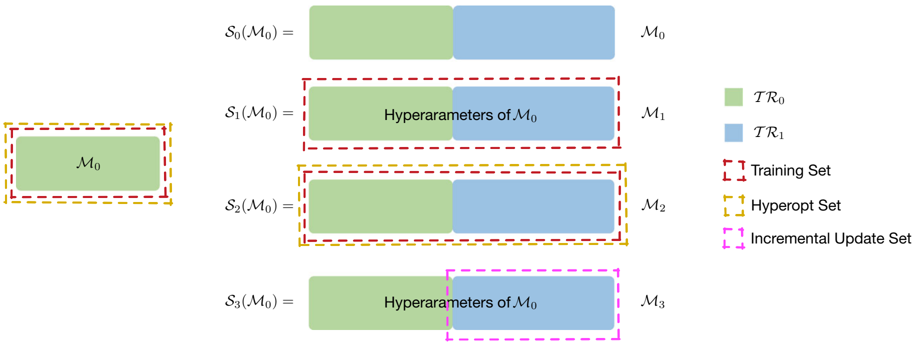

$\mathcal{S}_0(\mathcal{M}_0) =$
$\mathcal{M}_0$
$\mathcal{S}_1(\mathcal{M}_0) =$
Hyperarameters of $\mathcal{M}_0$
$\mathcal{M}_1$
$\mathcal{S}_2(\mathcal{M}_0) =$
$\mathcal{M}_2$
$\mathcal{S}_3(\mathcal{M}_0) =$
Hyperarameters of $\mathcal{M}_0$
$\mathcal{M}_3$
$\mathcal{M}_0$
$\mathcal{TR}_0$
$\mathcal{TR}_1$
Training Set
Hyperopt Set
Incremental Update Set

Fig. 2 Four strategies to produce $\mathcal{M}'$

they become new data, potentially available to be exploited for building a new model $\mathcal{M}'$, that in turn can be used to provide predictions on new incomplete traces. The need for exploiting new execution traces and building such an updated $\mathcal{M}'$ could be due to several reasons, among which the evolution of the process at hand (and thus of its executions) to which the system needs to adapt in a timely manner.

In this paper, we provide 4 different strategies for computing $\mathcal{M}'$, by exploiting a new set of process executions $\mathcal{TR}_1$, collected in a period of time "Period B" subsequent to "Period A", along with the original set $\mathcal{TR}_0$. The strategies are summarised in Fig. 2. The figure represents, on the left hand side, model $\mathcal{M}_0$ and, on the right hand side, the operations performed starting from $\mathcal{M}_0$ to obtain model $\mathcal{M}'$ according to the four update strategies.

The first strategy, $\mathcal{S}_0$, is a **do nothing** strategy. This strategy simply disregards that new traces are produced in "Period B" and continues to use $\mathcal{M}_0$ as a predictive model. This strategy may prove to be useful when the processes remain stable and it acts also as a baseline against which to compare all the other strategies.

The second strategy, $\mathcal{S}_1$, exploits the new traces in $\mathcal{TR}_1$ produced in "Period B" for training but not for the optimisation of the hyperparameters. In this **re-train with no hyperopt** strategy, $\mathcal{M}_0$ is replaced by a new predictive model $\mathcal{M}_1$ built from scratch by using $\mathcal{TR}_0 \cup \mathcal{TR}_1$ as train set. No optimisation of the hyperparameters is made in the construction of $\mathcal{M}_1$ and the values of the ones computed for $\mathcal{M}_0$ are instead used. This strategy aims at exploiting the new data in $\mathcal{TR}_1$ still avoiding the costly steps needed for hyperparameter optimisation.

The third strategy, $\mathcal{S}_2$, completely replaces the old model $\mathcal{M}_0$ with a new predictive model $\mathcal{M}_2$ built from scratch using both $\mathcal{TR}_0$ and $\mathcal{TR}_1$. This strategy aims at performing a **full re-train**, thus exploiting to the outmost all the available data. Also, the comparison between

$\mathcal{S}_1$ and $\mathcal{S}_2$ enables us to investigate the specific role of the hyperparameter tuning in the predictions.

The final strategy, $\mathcal{S}_3$, exploits the new traces in $\mathcal{TR}_1$ produced in “Period B” for training the predictive model in an incremental manner. Differently from $\mathcal{S}_1$, the data of $\mathcal{TR}_1$ are added as training data in a continuous manner by means of incremental Machine Learning algorithms, to extend the existing knowledge of model $\mathcal{M}_0$. The **incremental update** strategy is chosen as an example of dynamic technique, which can be applied when training data become available gradually over time or the size of the training data is too large to store or process it all at once. Similarly to $\mathcal{S}_1$, the value of the hyperparameters does not change when adding new training data.

5 Empirical evaluation

The evaluation reported in this paper aims at understanding the characteristics of the four different update strategies introduced in the previous section in terms of accuracy and time. We aim at evaluating these strategies with two types of real-life event log data: event logs without an explicit Concept Drift and event logs with an explicit Concept Drift. As such, we have selected four real-life datasets,[2] two for the first scenario and two for the second one. To consolidate the evaluation on the Concept Drift scenario, we also expanded the evaluation to include a synthetic event log with explicit Concept Drifts introduced in Maaradji et al. [23].

In this section, we introduce the research questions, the datasets, the metrics used to evaluate the effectiveness of the four update strategies described in Sect. 4, the procedure, and the tool settings. The results are instead reported in Sect. 6.

5.1 Research questions

Our evaluation is guided by the following research questions:

RQ1 How do the four update strategies **do nothing**, **re-train with no hyperopt**, **full re-train**, and **incremental update** compare to one another in terms of accuracy?

RQ2 How do the four update strategies **do nothing**, **re-train with no hyperopt**, **full re-train**, and **incremental update** compare to one another in terms of time performance?

RQ1 aims at evaluating the quality of the predictions returned by the four update strategies, while **RQ2** investigates the time required to build the predictive models in the four scenarios and, in particular, aims at assessing the difference between the complete periodic rediscovery (**full re-train**) and the other two update strategies **re-train with no hyperopt** and **incremental update**.

5.2 Datasets

The four update strategies are evaluated using five datasets. Three of them are real-life event logs provided for a Business Process Intelligence (BPI) Challenge, in different years, without an explicit Concept Drift: the BPI Challenges 2011 3TU Data Center [1], 2012 van Dongen [45], and 2015 van Dongen; [46]. They are examples of event logs exhibiting Process Variability as described in the first scenario in Sect. 3. The remaining two datasets are instead

[2] From the repository available at https://data.4tu.nl/.

examples of logs with explicit Concept Drift as described in the second scenario in Sect. 3. Our aim was to evaluate the four strategies on real-life event logs, but, to the best of our knowledge, the only publicly available event log which contains an explicit concept drift is the BPI Challenge 2018 van Dongen and Borchert [47]. Therefore, we decided to augment the evaluation considering also one of the synthetic event logs introduced in Maaradji et al. [23]. Here, we report the main characteristics of each dataset, while the outcomes to be predicted for each dataset are contained in Table 1.[3]

The first dataset, originally provided for the BPI Challenge 2011, contains the treatment history of patients diagnosed with cancer in a Dutch academic hospital. The log contains 1,140 cases and 149,730 events referring to 623 different activities. Each case in this log records the events related to a particular patient. For instance, the first labelling (ϕ_{11}), for this dataset, is such that the positive traces are all the ones for which if activity CEA - tumour marker using meia occurs, then it is followed by an occurrence of activity squamous cell carcinoma using eia.

The second dataset, originally provided for the BPI Challenge 2012, contains the execution history of a loan application process in a Dutch financial institution. It is composed of 4,685 cases and 186,693 events referring to 36 different activities. Each case in this log records the events related to a particular loan application. For instance, the first labelling (ϕ_{21}), for this dataset, is such that the positive traces are all the ones in which event Accept Loan Application occurs.

The third dataset, originally provided for the BPI Challenge 2015, concerns the application process for construction permits in five Dutch municipalities. We consider the log pertaining to the first municipality, which is composed of 1,199 cases and 52,217 events referring to 398 different activities.

The fourth dataset, originally provided for the BPI Challenge 2018, concerns an event log from the European Agricultural Guarantee Fund pertaining to an application process for EU direct payments for German farmers from the European Agricultural Guarantee Fund. Depending on the document types, different branches of the workflow are performed. The event log used in this evaluation is composed of 29,302 cases and 1,661,656 events referring to 40 different activities.

The fifth and the sixth datasets, hereafter called DriftRIO1 and DriftRIO2, are synthetic event logs that use a "textbook" example of a business process for assessing loan applications Weber et al. [48]. The DriftRIO event logs introduced in Maaradji et al. [23] have been built by alternating traces executing the original "base" model and traces modified so as to exhibit complex Concept Drifts obtained by composing simple log changes, namely, re-sequentialisation of process model activities (R), insertion of a new activity (I) and optionalisation of one activity (O)[4] DriftRIO1 is composed of 3,994 cases and 47,776 events related to 19 activities. DriftRIO2 is composed, instead, of 2,000 cases and 21,279 events referring to 19 different activities. The outcomes to be predicted for each dataset are expressed by using LTL formulae for the four BPI Challenges Di Francescomarino et al. [12], Maggi et

[3] We assume that events occurring during the process execution fall in the set of atomic propositions. LTL rules are constructed from these atoms by applying the temporal operators in addition to the usual Boolean connectives. Given a formula φ, $\mathbf{F}\varphi$ indicates that φ is true sometimes in the future. $\mathbf{G}\varphi$ means that φ is true always in the future. $\varphi\mathbf{U}\psi$ indicates that φ has to hold at least until ψ holds and ψ must hold in the current or in a future time instant.

[4] The logs used in Maaradji et al. [23] have been slightly modified (i) by further strengthening the introduced Concept Drift with the increase in the time duration of two activities, as well as (ii) by considering only the first parts of the event logs, so as to have a single occurrence of the Concept Drift rather than multiple occurrences as in the original event logs.

Table 1 The outcome formulae

Dataset	Outcome
BPIC11	$\phi_{11} = \mathbf{G}(\text{CEA - tumour marker using meia} \rightarrow \mathbf{F}(\text{squamous cell carcinoma using eia}))$
	$\phi_{12} = \neg(\text{histological examination - biopsies nno})\mathbf{U}(\text{squamous cell carcinoma using eia})$
	$\phi_{13} = \mathbf{F}(\text{histological examination} - \text{big resectiep})$
BPIC12	$\phi_{21} = \mathbf{F}(\text{Accept Loan Application})$
	$\phi_{22} = \mathbf{F}(\text{Reject Loan Application})$
	$\phi_{23} = \mathbf{F}(\text{Cancel Loan Application})$
BPIC15	$\phi_{31} = \mathbf{F}(\text{start WABO procedure}) \wedge \mathbf{F}(\text{extend procedure term})$
	$\phi_{32} = \mathbf{F}(\text{receive additional information}) \vee \mathbf{F}(\text{enrich decision})$
	$\phi_{33} = \mathbf{G}(\text{send confirmation receipt} \rightarrow \mathbf{F}(\text{retrieve missing data}))$
BPIC18	$\phi_{41} = \mathbf{F}(\text{Calculate} \rightarrow (\text{Parcel document} \vee \text{Geo parcel document})) \vee \mathbf{F}(\text{Finish editing} \rightarrow \text{Begin editing})$
DriftRIO1	$\phi_{51} = \text{fast case}$
DriftRIO2	$\phi_{61} = \text{fast case}$

al. [25] and by using the case duration property of being a fast case for DriftRIO[5] (see Table 1).

5.3 Metrics

In order to answer the research questions, we use two metrics, one for accuracy and one for time. The one for accuracy is used to evaluate **RQ1**, whereas the time measure is used to evaluate **RQ2**.

The accuracy metric. In this work, we exploit a typical evaluation metric for calculating the performance of a classification model, that is AUC-ROC (hereafter only AUC). The ROC curve is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied. The curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) using various threshold settings. In formulae, $TPR = \frac{T_P}{T_P + F_N}$, and $FPR = \frac{F_P}{FP + T_N}$, where T_P, T_N, F_P, and F_N are the true-positives (positive outcomes correctly predicted), the true-negatives (negative outcomes correctly predicted), the false-positives (negative outcomes predicted as positive), and the false-negatives (positive outcomes predicted as negative), respectively. In our case, the AUC is the area under the ROC curve and, when using normalised units, it can be intuitively interpreted as the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one. As usual, T_P, T_N, F_P, and F_N are obtained by comparing the predictions produced by the predictive models against a *gold standard* that indicates the correct labelling of each case. In our experiments, we have built the gold standard by evaluating the outcome of each completed case in the test set[6]

The time metric. We measure the time spent to build the predictive model in terms of execution time. The execution time indicates the time required to create and update (in the case of incremental algorithms) the predictive models. We remark here that the execution time does not include the time spent to load and pre-process the data, but only the bare processing time.

5.4 Experimental procedure

We adopt the classical Machine Learning Train/Validate/Test experimental procedure and configure it for the four different strategies we want to compare. The procedure consists of the following main steps: (1) dataset preparation; (2) classifier training and validation; (3) classifier testing and metrics collection.

[5] Specifically, the property defines fast a case with a cycle time lower than the average cycle time of the event log.

[6] To avoid biases related to the chosen metric, in our experiments we have also measured accuracy in terms of average F-measure and accuracy defined as $\frac{T_P + T_N}{T_P + T_N + F_P + F_N}$, and the results remain consistent. We have, therefore, decided to focus only on AUC for the sake of simplicity and readability of the results.

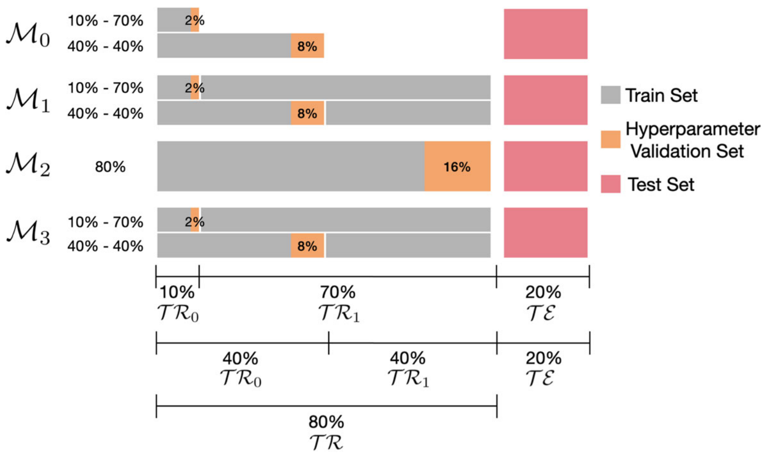


Fig. 3 The experimental settings used to build the predictive models

Table 2 Dataset entropy

Dataset	Trace Entropy			Global Block Entropy		
	$\mathcal{TR} \cup \mathcal{TE}$ 0–100%	$\mathcal{TR}_\mathbf{0} \cup \mathcal{TE}$ 0–40%–80–100%	$\mathcal{TR}_\mathbf{1} \cup \mathcal{TE}$ 40–80%–80–100%	$\mathcal{TR} \cup \mathcal{TE}$ 0–100%	$\mathcal{TR}_\mathbf{0} \cup \mathcal{TE}$ 0–40%–80–100%	$\mathcal{TR}_\mathbf{1} \cup \mathcal{TE}$ 40–80%–80–100%
BPIC11	9.62	9.08	8.97	24.71	23.73	24.13
BPIC12	11.65	11.05	11.08	16.18	15.9	16.08
BPIC15	10.16	9.4	9.48	18.96	18.55	18.4
BPIC18	12.88	12.7	11.56	20.72	20.54	19.74
DriftRIO1	4.66	4.71	4.29	8.52	8.54	8.38
DriftRIO2	4.27	4.29	4.1	8.41	8.41	8.37

In the dataset preparation phase, the execution traces are first ordered according to their starting date-time so as to be able to meaningfully identify those referring to “Period A” (that is, $\mathcal{TR}_0$), those referring to the subsequent “Period B” (that is, $\mathcal{TR}_1$), and those referring to the test set $\mathcal{TE}$, which here represents the most recent set of traces where the actual predictions are made. The predictions are tested using the four different models $\mathcal{M}_0$-$\mathcal{M}_3$ corresponding to the different strategies $\mathcal{S}_0$-$\mathcal{S}_3$ described in Sect. 4.

The actual splits between train, hyperparameter validation, and test sets used for evaluating the four strategies are illustrated in Fig. 3. Following a common practice in Machine Learning, we have decided to use 80% of the data for training/validation and 20% for testing.

To test whether the performance of the different strategies was connected to different sizes of $\mathcal{TR}_0$ and $\mathcal{TR}_1$, we devised two experimental settings to supply the train data to the learning algorithms. The first experimental setting divides the train set unequally in $\mathcal{TR}_0$ (10%) and $\mathcal{TR}_1$ (70%); the second experimental setting divides the train set equally in $\mathcal{TR}_0$ (40%) and $\mathcal{TR}_1$ (40%). These two settings do not concern the construction of $\mathcal{M}_2$, which is always built using the entire set of available train data. The hyperparameter validation set is extracted from the train set through a randomised sampling procedure. It amounts to 20%

Table 3 Dataset label distribution

Dataset	Formula	$\mathcal{TR}_0$				$\mathcal{TR}$		$\mathcal{TE}$		Total	
		0–10%		0–40%		0–80%		80–100%		0–100%	
		True	False	True	False	True	False	True	False	True	False
BPIC11	ϕ_{11}	50	64	156	300	359	663	109	119	458	682
	ϕ_{12}	96	18	391	65	742	170	151	77	893	247
	ϕ_{13}	20	94	79	377	161	639	180	732	259	881
BPIC12	ϕ_{21}	227	241	904	970	1761	1987	482	455	2243	2442
	ϕ_{22}	163	305	663	1211	1366	2382	274	663	1640	3045
	ϕ_{23}	78	390	307	1567	621	3127	181	756	802	3883
BPIC15	ϕ_{31}	6	90	7	472	59	900	63	178	122	1078
	ϕ_{32}	33	63	110	369	223	736	110	131	333	867
	ϕ_{33}	33	63	124	355	215	744	83	158	298	902
BPIC18	ϕ_{41}	1994	936	7787	3933	16351	7090	4384	1477	20735	8567
DriftRIO1	ϕ_{51}	25	272	132	1081	989	1987	582	436	1571	2423
DriftRIO2	ϕ_{61}	83	117	398	402	834	766	229	171	1063	937

of the train set, which corresponds to 2% of the dataset for the train set at 10%, 8% for the train set at 40%, and 16% for the train set at 80% (see Fig. 3).

The variability of the log behaviour of each dataset can be measured through the *trace entropy* and the *Global block entropy* metrics Back et al. [2] (Table 2). While the first metric mainly focuses on the number of variants in an event log, the latter also takes into account the internal structure of the traces. Note that the BPI Challenge 2018 and DriftRIO datasets contain a Concept Drift in $\mathcal{TR}_1$ in both settings (it affects the last 30% of data in $\mathcal{TR}$, when ordered according to their starting date-time) and, for these datasets, the difference between the entropy value of the split 0%-40%–80%–100% is higher than the entropy value of the split 40%–80%–80%–100% for both entropy metrics. Comparing the behaviour variability of the splits 0%–10%–80%–100% and 10%–80%–80%–100% is instead less useful since the two sets of traces have different sizes and the smaller set has obviously a lower entropy w.r.t. the larger one for all datasets. Finally, the entropy on the complete datasets (column 0%–100%) is useful to understand what type of input is provided when evaluating strategy $\mathcal{S}_2$ that uses the full dataset to train the predictive model.

Additional information about the input logs used in our experiments is reported in Table 3 that shows, for each dataset and for each setting, the distribution of the labels on both the train and the test sets, thus providing an idea of how much balanced the datasets are.

Once the data are prepared, training and validation start by extracting execution prefixes and by encoding them using the complex index encoding introduced by Leontjeva et al. [22].[7] In the complex index encoding, the data related to a process execution are divided into static and dynamic information. Static information is the same for all the events in the sequence (e.g. the age of a patient), while dynamic information changes for different events occurring in the process execution (e.g. the name of the activity, the pressure value of a patient associated

[7] We have decided to use the complex index encoding as it is the one providing better performance in Predictive Process Monitoring – see Leontjeva et al. [22].

Table 4 Model hyperparameters

Dataset	Formula	0–10% $\mathcal{M}_0, \mathcal{M}_1, \mathcal{M}_3$			0–40% $\mathcal{M}_0, \mathcal{M}_1, \mathcal{M}_3$			0–80% $\mathcal{M}_2$		
		# est	max_d	max_f	# est	max_d	max_f	# est	max_d	max_f
BPIC11	ϕ_{11}	163	4	$\log_2$(n)	537	8	sqrt(n)	397	15	auto
	ϕ_{12}	273	9	$\log_2$(n)	155	7	auto	991	7	n
	ϕ_{13}	395	28	n	865	12	n	203	7	n
BPIC12	ϕ_{21}	157	16	sqrt(n)	427	14	sqrt(n)	206	9	auto
	ϕ_{22}	700	6	auto	558	9	sqrt(n)	536	6	auto
	ϕ_{23}	180	7	$\log_2$(n)	163	20	auto	587	27	$\log_2$(n)
BPIC15	ϕ_{31}	725	15	$\log_2$(n)	377	19	n	563	16	sqrt(n)
	ϕ_{32}	977	13	auto	937	8	sqrt(n)	417	14	sqrt(n)
	ϕ_{33}	374	29	auto	374	29	auto	161	13	sqrt(n)
BPIC18	ϕ_{41}	990	8	n	650	26	sqrt(n)	165	24	n
DriftRIO1	ϕ_{51}	913	4	$\log_2$(n)	971	9	n	293	4	$\log_2$(n)
DriftRIO2	ϕ_{61}	998	23	n	151	4	sqrt(n)	178	5	auto

with an event aimed at measuring the patient's pressure). We encode static information as attribute-value pairs, while, for dynamic information, the order in which events (and the related attributes) occur in the sequence is also taken into account.

The hyperparameter optimisation function uses the Tree of Parzen Estimators (TPE)[8] to retrieve the hyperparameter configuration (see Sect. 2.3), with a maximum of 1000 iterations, and the AUC as objective evaluation measure to maximise. Models $\mathcal{M}_0$-$\mathcal{M}_3$ are produced at the end of this phase. The specific hyperparameters used by the models are reported in Table 4. The table shows that indeed differences exist in the optimised hyperparameters computed starting from different train sets, i.e. 10% of the datasets, 40% of the datasets (both used for $\mathcal{M}_0$, $\mathcal{M}_1$, and $\mathcal{M}_3$) and 80% of the datasets (used for $\mathcal{M}_2$).

After the training and validation procedures are completed, we test the resulting model with the test set, and we collect the scored metrics (see Sect. 5.3) and store them in a database. Concerning time, we have decided to measure here the time spent to build the initial model $\mathcal{M}_0$, plus the time needed to update it according to the four different strategies. Thus, for $\mathcal{S}_0$, this will coincide with the time spent for building $\mathcal{M}_0$ (as, in this case, no further action is taken for updating the model); for $\mathcal{S}_1$, we compute the time spent for building $\mathcal{M}_0$ plus the time spent for the re-training over $\mathcal{TR}$ (no hyperopt); for $\mathcal{S}_2$, we compute the time spent for building $\mathcal{M}_0$ plus the time needed for the re-training over $\mathcal{TR}$ (with hyperopt); and, finally, for $\mathcal{S}_3$, we compute the time spent for building $\mathcal{M}_0$ plus the time needed for updating it with the data in $\mathcal{TR}_1$.

[8] TPE has been shown to be a good solution for complex hyperparameter optimisation problems Bergstra et al. [3] and is traditionally used for outcome-oriented Predictive Process Monitoring solutions Teinemaa et al. [40].

5.5 Experimental settings

The tool used for the experimentation is `Nirdizati` Rizzi et al. [34]. The experimental evaluation was performed on a workstation Dell Precision 7820 with the following configuration: (i) 314GB DDR4 2666MHz RDIMM ECC of RAM; (ii) double Intel Xeon Gold 6136 3.0GHz, 3.7GHz Turbo, 12C, 10.4GT/s 3UPI, 24.75MB Cache, HT (150W) CPU; and (iii) one 2.5" 256GB SATA Class 20 Solid State Drive. We assumed to have only 1 CPU for training the predictive models, i.e. we did not parallelise the training of the base learners of the Random Forests. We also ensured that there was no racing condition over the disk and no starvation over the RAM usage by actively monitoring the resources through Netdata Tsaousis [42]. Each experiment was allowed to run for at most 100 hours. No other experiment or interaction with the workstation was performed other than the monitoring of the used resources.

6 Results

In this section, we present the results of our experiments, reported in Tables 5, 6, 7, and discuss how they allow us to answer the two research questions introduced before. We also provide a discussion about the four update strategies with a cost-effectiveness analysis and an analysis of the validity threats. To ensure reproducibility, the datasets used, the configurations, and the detailed results are available at http://bit.ly/how_do_I_update_my_model.

6.1 Discussion

Answering RQ1. The AUC of all models, for the two experimental settings 10%–70% and 40%–40%, for all datasets, is reported in Tables 5a and 5b. The best result for each dataset and labelling is emphasised in italic[9]. Since, in many cases, different models have very close accuracy, we have emphasised in bold the results that differ from the best ones for less than 0.01. Tables 6a and 6b report, for the two experimental settings 10%–70% and 40%–40%, the percentage of gain/loss of $\mathcal{M}_1$, $\mathcal{M}_2$, and $\mathcal{M}_3$ w.r.t. $\mathcal{M}_0$. The percentage of gain or loss[10] is reported together with an histogram of gains and losses. In order to ease the comparison, $\mathcal{M}_2$ is reported in both tables.

By looking at the tables, we can immediately see that $\mathcal{M}_2$ and $\mathcal{M}_3$ are the clear best performers, especially in the first setting 10%–70%, and that $\mathcal{M}_0$ is almost consistently the worst performer, often with a significant difference in terms of accuracy. This highlights that the need to update the predictive models is a real issue in typical Predictive Process Monitoring settings. The only exception to this finding is provided by the results obtained for the BPIC15 dataset, where the performance of $\mathcal{M}_0$ is comparable to that of the other models for almost all the outcome formulae. This is due to the fact that, for this dataset, there is a high homogeneity of the process behaviour over time. This is confirmed by the entropy values, provided in Table 2, that remain quite stable across the entire log. Moreover, we can observe that BPIC12 with labelling ϕ_{23} has overall the lowest accuracy for all the four update strategies. This is possibly due to the high label unbalance characterising this dataset (see Table 3).

[9] We indicate in italic the best result, which may be due to digits smaller than the third decimal one reported in the paper.

[10] Computed as $\frac{\mathcal{M}_i - \mathcal{M}_0}{\mathcal{M}_0}$, $i \in \{1, 2, 3\}$.

Table 5 The accuracy results

Dataset	ϕ	$\mathcal{M}_0$	$\mathcal{M}_1$	$\mathcal{M}_2$	$\mathcal{M}_3$
(a) Setting 10–70%					
BPIC11	ϕ_{11}	0.673	0.885	***0.919***	0.833
	ϕ_{12}	0.745	0.887	***0.964***	0.883
	ϕ_{13}	0.843	**0.916**	**0.921**	***0.926***
BPIC12	ϕ_{21}	0.576	0.648	***0.702***	0.671
	ϕ_{22}	0.672	**0.733**	***0.740***	0.676
	ϕ_{23}	0.517	0.504	0.514	***0.561***
BPIC15	ϕ_{31}	0.908	0.916	0.935	***0.993***
	ϕ_{32}	0.928	0.911	0.923	***0.972***
	ϕ_{33}	0.961	0.976	**0.988**	***0.995***
BPIC18	ϕ_{41}	0.532	**0.999**	**0.991**	***0.999***
DriftRIO1	ϕ_{51}	0.603	**0.964**	***0.965***	**0.964**
DriftRIO2	ϕ_{61}	0.761	**0.856**	**0.854**	***0.857***

Dataset	ϕ	$\mathcal{M}_0$	$\mathcal{M}_1$	$\mathcal{M}_2$	$\mathcal{M}_3$
(b) Setting 40–40%					
BPIC11	ϕ_{11}	0.781	***0.935***	0.919	0.902
	ϕ_{12}	0.811	0.909	***0.964***	0.930
	ϕ_{13}	0.894	**0.918**	***0.921***	**0.920**
BPIC12	ϕ_{21}	0.631	0.671	***0.702***	0.682
	ϕ_{22}	0.674	0.672	***0.740***	0.702
	ϕ_{23}	0.509	0.511	0.514	***0.560***
BPIC15	ϕ_{31}	0.779	***0.991***	0.935	0.944
	ϕ_{32}	0.895	0.907	0.923	***0.953***
	ϕ_{33}	0.972	***0.994***	**0.988**	**0.987**
BPIC18	ϕ_{41}	0.543	**1.000**	**0.991**	***1.000***
DriftRIO1	ϕ_{51}	0.559	**0.964**	**0.965**	***0.969***
DriftRIO2	ϕ_{61}	0.805	0.698	***0.854***	**0.841**

The performance of all the evaluated strategies is overall higher in the 40%–40% setting, and this is likely related to the higher amount of data used for hyperparameter optimisation. Nonetheless, the lower performance of $\mathcal{M}_0$ also in the 40%–40% setting indicates the need to update the models with new data at regular intervals. Concerning the possible differences between the results obtained using datasets with and without an explicit Concept Drift, our experiments did not find any striking variation in the different strategies, thus consolidating the finding that devising update strategies is important in general, also in scenarios where the process changes over time are not so definite. Nonetheless, if we look at Tables 6a and 6b, it is easy to see that the experiments with an explicit Concept Drift are the ones with the greatest difference between $\mathcal{M}_0$ and all the other models, thus confirming that an explicit Concept Drift can have a significant negative influence on the performance of a Predictive Process Monitoring model, if it is not updated. This is especially true for BPIC18, in which

Table 6 Accuracy improvement against $\mathcal{M}_0$

Dataset	ϕ	$\mathcal{M}_1$	$\mathcal{M}_2$	$\mathcal{M}_3$
(a) Setting 10–70%				
BPIC11	ϕ_{11}	0.31	0.36	0.23
	ϕ_{12}	0.19	0.29	0.18
	ϕ_{13}	0.08	0.09	0.09
BPIC12	ϕ_{21}	0.12	0.21	0.16
	ϕ_{22}	0.09	0.10	0.00
	ϕ_{23}	−0.02	0.00	0.08
BPIC15	ϕ_{31}	0.01	0.03	0.09
	ϕ_{32}	−0.01	0.00	0.04
	ϕ_{33}	0.01	0.02	0.03
BPIC18	ϕ_{41}	0.87	0.86	0.87
DriftRIO1	ϕ_{51}	0.59	0.59	0.59
DriftRIO2	ϕ_{61}	0.12	0.12	0.12

Dataset	ϕ	$\mathcal{M}_1$	$\mathcal{M}_2$	$\mathcal{M}_3$
(b) Setting 40–40%				
BPIC11	ϕ_{11}	0.19	0.17	0.15
	ϕ_{12}	0.12	0.18	0.14
	ϕ_{13}	0.02	0.03	0.03
BPIC12	ϕ_{21}	0.06	0.11	0.08
	ϕ_{22}	−0.00	0.09	0.04
	ϕ_{23}	0.00	0.01	0.10
BPIC15	ϕ_{31}	0.27	0.20	0.21
	ϕ_{32}	0.01	0.03	0.06
	ϕ_{33}	0.02	0.01	0.01
BPIC18	ϕ_{41}	0.84	0.82	0.84
DriftRIO1	ϕ_{51}	0.72	0.72	0.73
DriftRIO2	ϕ_{61}	−0.13	0.06	0.04

the Concept Drift highly affects the entropy measure (Table 2) in a way that the difference between the entropy value of the split 0%–40%–80%–100% is higher than the entropy value of the split 40%–80%–80%–100% for both entropy metrics.

Tables 6a and 6b show also another interesting aspect of our evaluation: while $\mathcal{M}_2$ and $\mathcal{M}_3$ tend to always gain against $\mathcal{M}_0$ (or to be stable in very few cases), the same cannot be said for $\mathcal{M}_1$. In fact, if we look at BPIC12 with labelling ϕ_{23} and BPIC15 with labelling ϕ_{32} in Table 6a, and, particularly, at DriftRIO with labelling ϕ_{52} in Table 6b, we can see a decrease in the accuracy. By carrying out a deeper analysis of the chosen hyperparameters, we found that the lower accuracy of $\mathcal{M}_1$ is due to the inappropriateness of the hyperparameters derived from $\mathcal{TR}_0$ to the new data used to build $\mathcal{M}_1$. While this aspect may need to be better investigated, we can conclude that while **re-train with no hyperopt** is usually a viable solution, it is nonetheless riskier than **full re-train** or **incremental update**.

The general findings and trends derived from the results obtained using Random Forest as classifier are further confirmed, with few exceptions, by the results obtained by using Perceptron Minsky and Papert [29] as predictive model in the analysis of the four update strategies. The perceptron results are reported in Appendix A.

To sum up, concerning **RQ1**, our evaluation shows that **full re-train** and **incremental update** are the best performing update strategies in terms of accuracy, followed by **re-train with no hyperopt**. With the exception of BPIC15 in the 10%–70% setting, **do nothing** is, often by far, the worst strategy, indicating the importance of updating the predictive models with new data, when it becomes available.

Answering RQ2

The time spent for creating the four models, for the two experimental settings 10%–70% and 40%–40%, for all datasets, is reported in Tables 7a and 7b using the "hh:mm:ss" format. The best results for each dataset and labelling (that is the lower execution times) are emphasised in italic, while execution times that differ for less than 60 seconds from the best ones are indicated in bold. The last two rows of each table report the average time (and standard deviation) necessary to train a model for a given strategy. In order to ease the comparison, $\mathcal{M}_2$ is reported in both tables. $\mathcal{M}_0$ and $\mathcal{M}_2$ are self-contained models that are "built from scratch" and, therefore, the time reported in the tables for these models is the time spent to train them. Differently, $\mathcal{M}_1$ and $\mathcal{M}_3$ are built in a two-steps fashion that includes a training phase but also the usage of the hyperparameters used to build $\mathcal{M}_0$. Therefore, their construction time is measured by summing up the time spent for the training phase and the time spent for building $\mathcal{M}_0$.

By looking at Tables 7a and 7b, we can immediately see that, among all the evaluated strategies, $\mathcal{M}_0$ is the clear best performer, especially for the 10%–70% setting, and that $\mathcal{M}_2$ is almost consistently the worst performer, often with a significant difference in terms of time spent to build the model (with few exceptions in the 40%–40% case that we will discuss below). As a second general observation, we note that $\mathcal{M}_1$ and $\mathcal{M}_3$ share almost the same construction time with $\mathcal{M}_0$. This fact is not particularly surprising, as the hyperparameter optimisation routine is often the most expensive step in the construction of this type of predictive models. Therefore, the two strategies $\mathcal{S}_1$ and $\mathcal{S}_3$ that underline the construction of these models are highly inexpensive, when we consider the time dimension, especially when $\mathcal{M}_0$ is already available.

If we compare the two experimental settings 10%–70% and 40%–40%, we can observe that while $\mathcal{M}_0$ is almost always the best performer in both settings, the difference between the construction time of $\mathcal{M}_0$ (and thus of $\mathcal{M}_1$ and $\mathcal{M}_3$) and the construction time of $\mathcal{M}_2$ is significantly higher for the 10%–70% setting. While the investigation of *when* it is convenient to perform a **full re-train** is out of the scope of the paper and is left to further investigations, this finding emphasises the fact that the cost of a **full re-train** may increase in a significant manner if the update of the predictive model over-delays and the amount of new data greatly increases. Interestingly enough, the 40%–40% setting presents two cases in which $\mathcal{M}_2$ is the fastest model to be built. The case of BPIC11 with labelling ϕ_{12} is likely due to an "unfortunate" guess-estimate in the hyperparameter optimisation step for $\mathcal{M}_0$, which makes the training time explode[11] the case of BPIC15 with labelling ϕ_{31}, instead, represents a situation in which $\mathcal{M}_0$ and $\mathcal{M}_2$ take almost the same time to be built. Concerning possible differences between datasets with and without an explicit Concept Drift, our experiments did not find any striking difference among the evaluated strategies.

[11] Note that an explosion of the training time for the same reason affects also model $\mathcal{M}_2$ for BPIC11 with labelling ϕ_{13}.

Table 7 The time results

Dataset	ϕ	$\mathcal{M}_0$	$\mathcal{M}_1$	$\mathcal{M}_2$	$\mathcal{M}_3$
(a) Setting 10–70%					
BPIC11	ϕ_{11}	***05:31:49***	05:37:18	10:12:42	05:35:53
	ϕ_{12}	***06:03:05***	06:05:01	09:23:21	06:04:32
	ϕ_{13}	***01:00:21***	**01:00:25**	30:43:11	**01:00:25**
BPIC12	ϕ_{21}	***00:46:59***	00:51:00	08:02:32	**00:47:02**
	ϕ_{22}	***00:46:42***	***00:46:42***	09:03:06	**00:46:43**
	ϕ_{23}	***03:38:22***	***03:38:22***	11:37:39	**03:38:37**
BPIC15	ϕ_{31}	***00:13:10***	**00:13:11**	00:27:31	**00:13:11**
	ϕ_{32}	***00:14:49***	**00:14:52**	00:51:01	**00:14:51**
	ϕ_{33}	***00:13:03***	**00:13:04**	00:52:12	***00:13:03***
BPIC18	ϕ_{41}	***26:44:03***	27:26:00	74:44:03	26:51:24
DriftRIO1	ϕ_{51}	***00:12:38***	**00:12:39**	00:16:17	**00:12:39**
DriftRIO2	ϕ_{61}	***00:13:19***	**00:13:21**	00:14:22	**00:13:21**
Average Time		***01:43:07***	01:44:10	12:52:20	**01:43:40**
Std deviation		02:14:45	02:15:53	21:06:04	02:15:44

Dataset	ϕ	$\mathcal{M}_0$	$\mathcal{M}_1$	$\mathcal{M}_2$	$\mathcal{M}_3$
(b) Setting 40%–40%					
BPIC11	ϕ_{11}	***05:00:33***	**05:00:38**	10:12:42	**05:00:36**
	ϕ_{12}	18:53:46	18:54:40	***09:23:21***	18:54:14
	ϕ_{13}	***05:13:21***	**05:13:38**	30:43:11	**05:13:29**
BPIC12	ϕ_{21}	***03:23:40***	03:27:06	08:02:32	**03:23:42**
	ϕ_{22}	***03:40:02***	***03:40:02***	09:03:06	**03:40:03**
	ϕ_{23}	***05:00:07***	**05:00:08**	11:37:39	**05:00:25**
BPIC15	ϕ_{31}	00:29:11	00:29:16	***00:27:31***	00:29:15
	ϕ_{32}	***00:31:51***	**00:31:54**	00:51:01	**00:31:53**
	ϕ_{33}	***00:35:26***	**00:35:27**	00:52:12	***00:35:26***
BPIC18	ϕ_{41}	***55:06:43***	55:08:42	74:44:03	**55:06:57**
DriftRIO1	ϕ_{51}	***00:14:39***	**00:14:41**	00:16:17	**00:14:41**
DriftRIO2	ϕ_{61}	***00:13:53***	***00:13:53***	**00:14:22**	***00:13:53***
Average Time		***03:56:03***	**03:56:29**	12:52:20	**03:56:09**
Std deviation		05:23:12	05:23:24	21:06:04	05:23:20

Finally, our evaluation did not find any fixed correlation between the training times for all strategies and (i) the size of the dataset and the alphabet of the dataset within the same settings or (ii) the quality of the predictive model in terms of accuracy (and thus the difficulty of the prediction problem). As an example of the first, we can observe that BPIC12 contains four times the cases of BPIC11; nonetheless, most of the prediction models built for BPIC11 take more time to be constructed than the ones built for BPIC12. Similarly, BPIC15 has an alphabet with a number of activities that is almost 10 times the one of BPIC2018, but the prediction models built for BPIC18 take much more time to be constructed than the ones built for BPIC15. As an example of the second, we can observe, from Table 7a, that the time needed for building $\mathcal{M}_0$ for BPIC11 with labelling ϕ_{23} is much greater than the one needed for building $\mathcal{M}_0$ for BPIC11 with labelling ϕ_{13}, even if the accuracy for the same cases, in Table 5a, follows the inverse trend.[12]

To sum up, concerning **RQ2**, our evaluation shows that once $\mathcal{M}_0$ is available, **incremental update** and **re-train with no hyperopt** are the two most convenient update strategies—as they can be built in almost no time. This may suggest the possibility to implement an almost continuous update strategy whenever new data become available. While the investigation of *when* it is convenient to perform a **full re-train** is out of the scope of the paper, our experiments show that the cost of a **full re-train** may increase in a significant manner if the update of the predictive model over-delays and the amount of new data increases significantly.
Overall Conclusions. The plots in Figs. 4 and 5 show inaccuracy and time related to the 10%–70% and 40%–40% settings, respectively, for each of the considered datasets. The closer the item is to the origin, the best is the balance between the time required for training, re-training, or updating the model and the accuracy of the results. By looking at the plots, it is clear that the worst choice in terms of balance is given by $\mathcal{M}_0$, while, for the other three models, the choice somehow depends on the dataset and on the labelling. With the only exception of ϕ_{12}, ϕ_{21}, ϕ_{22}, and ϕ_{33} for both settings, as well as of ϕ_{11} for the 10%–70% setting and of ϕ_{61} for the 40%–40% setting, for all other datasets and labellings, $\mathcal{M}_1$ and/or $\mathcal{M}_3$ are the only non-dominated update strategies, i.e. those strategies for which another strategy improving both the inaccuracy and the time dimension does not exist.[13]

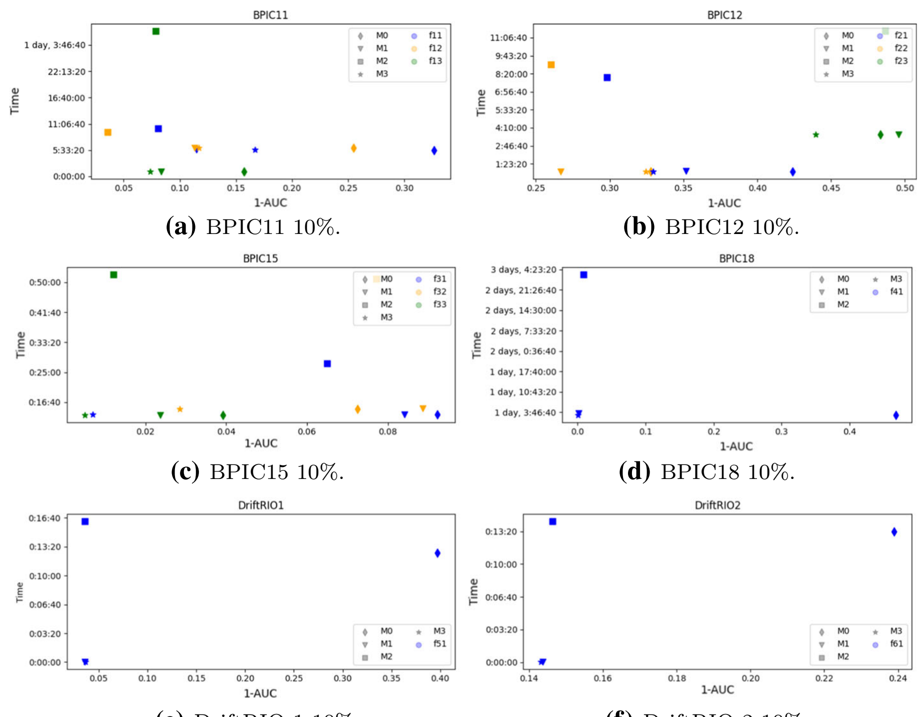
To conclude, our evaluation shows that the **do-nothing** strategy is not a viable strategy as the accuracy performance of a non-updated model tends to significantly decrease for typical real-life datasets (with and without explicit Concept Drift), whereas lightweight update strategies, such as the **incremental update** and **re-train with no hyperopt**, are, instead, often extremely effective in updating the models. **Full re-train** offers a strategy that almost always achieves the best accuracy (or an accuracy in line with the best one). Nonetheless, its training time may increase significantly, especially in the presence of an abundance of new data. According to our experiments, the **incremental update** is able to keep up with the **full re-train** strategy and deliver a properly fitted model almost in real time, suggesting that the potential of incremental models is under-appreciated in Predictive Process Monitoring, and smart Predictive Process Monitoring solutions could be developed leveraging this update strategy.

6.2 Cost-effectiveness analysis

In order to have a better grasp of the cost-effectiveness of the different update strategies, we also investigated the costs required by the update strategies, when new batches of data become available along the time. In particular, given a set of batches of train sets $\mathcal{TR}_0$, $\mathcal{TR}_1$, $\ldots\mathcal{TR}_n$, and the corresponding batches of test sets $\mathcal{TE}_0$, $\mathcal{TE}_1$, $\ldots\mathcal{TE}_n$ (where $\mathcal{TE}_0$ is the test set immediately following $\mathcal{TR}_0$, $\mathcal{TE}_1$ the test set immediately following $\mathcal{TR}_1$ in the temporal timeline), we can define $CE_M(\mathcal{TR}_0, \mathcal{TR}_1, \ldots\mathcal{TR}_n, \mathcal{TE}_0, \mathcal{TE}_1, \ldots\mathcal{TE}_n)$—from

[12] This is just one of many samples of an inverse relation between time and accuracy that can be found in the tables.

[13] We did not take into account small differences that cannot be observed with the naked eye in the plots.



(a) BPIC11 10%. **(b)** BPIC12 10%.

(c) BPIC15 10%. **(d)** BPIC18 10%.

(e) DriftRIO 1 10%. **(f)** DriftRIO 2 10%.

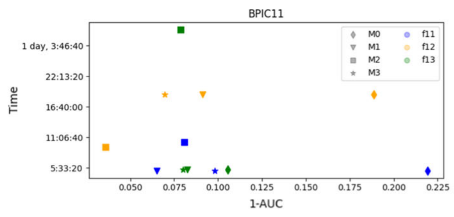
Fig. 4 Inaccuracy versus time plots (10%)

here on shortened as $CE_M(\mathcal{TR}_{0...n}, \mathcal{TE}_{0...n})$—as the cost of a model trained with n batches of arriving data $\mathcal{TR}_0, \mathcal{TR}_1, \ldots \mathcal{TR}_n$, and tested with the corresponding batches of test data $\mathcal{TE}_1, \ldots \mathcal{TE}_n$.

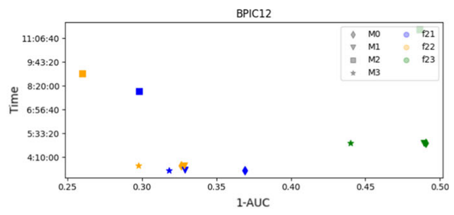
In our scenario, the cost-effectiveness of the update strategies is characterised by two main aspects: on the one hand, the cost of the time required for building the model and, on the other, the cost of returning wrong predictions (prediction inaccuracy). We can hence define $CE_{\mathcal{M}}(\mathcal{TR}_{0...n}, \mathcal{TE}_{0...n})$ as the sum of (i) $CT_{\mathcal{M}}(\mathcal{TR}_{0...n}, \mathcal{TE}_{0...n})$, i.e. the cost of the time required for building, training and, when necessary, re-training the model M, whenever a new batch of traces arrives;[14] and of (ii) $CI_{\mathcal{M}}(\mathcal{TR}_{0...n}, \mathcal{TE}_{0...n})$, i.e. the cost of the inaccuracy due to wrong predictions returned by the trained models on the traces of the test sets. Low values for a given model indicate a good cost-effectiveness model.

Defining $CT_T(TR)$, $CT_T^H(TR)$, and $CT_U(TR)$ as the time required for training, training and optimising the hyperparameters, and incrementally updating a model with the train set TR, respectively, the time costs related to the four models can be computed as reported in Eq. 1. The time cost for $\mathcal{M}_0$ is given by the only cost required for training and optimising the hyperparameters on $\mathcal{TR}_0$. For $\mathcal{M}_1$ ($\mathcal{M}_3$), besides the cost for training and optimising the hyperparameters on $\mathcal{TR}_0$, when the i-th train set batch is available, the costs for training (updating) the model with the union of the train set batches up to the i-th one (with the i-th train batch), have also to be considered. Finally, the time cost for $\mathcal{M}_2$ is given by the cost required for re-training and optimising the hyperparameters from scratch each time a new

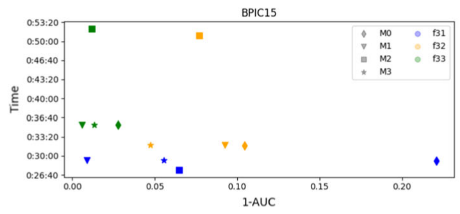
[14] We assume that the time cost for providing predictions at runtime, i.e. for testing our model is negligible.



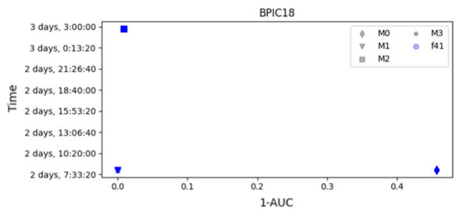
(a) BPIC11 40%.



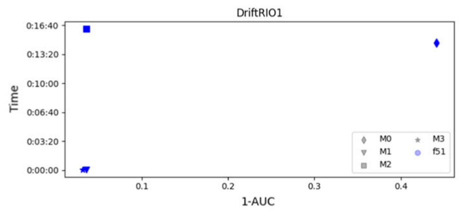
(b) BPIC12 40%.



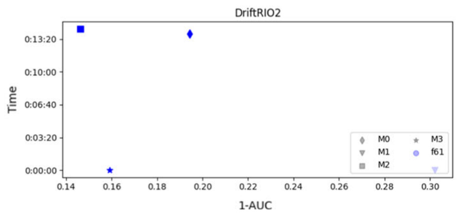
(c) BPIC15 40%.



(d) BPIC18 40%.



(e) DriftRIO 1 40%.



(f) DriftRIO 2 40%.

Fig. 5 Inaccuracy versus time plots (40%)

batch arrives.

$$
\begin{aligned}
CT_{\mathcal{M}_0}(\mathcal{TR}_{0\ldots n}, \mathcal{TE}_{0\ldots n}) &= CT_T^H(\mathcal{TR}_0)\\
CT_{\mathcal{M}_1}(\mathcal{TR}_{0\ldots n}, \mathcal{TE}_{0\ldots n}) &= CT_T^H(\mathcal{TR}_0) + \sum_{i=1}^{n-1} CT_T(\bigcup_{j=1}^{i} \mathcal{TR}_i)\\
CT_{\mathcal{M}_2}(\mathcal{TR}_{0\ldots n}, \mathcal{TE}_{0\ldots n}) &= \sum_{i=0}^{n-1} CT_T^H(\bigcup_{j=0}^{i} \mathcal{TR}_i)\\
CT_{\mathcal{M}_3}(\mathcal{TR}_{0\ldots n}, \mathcal{TE}_{0\ldots n}) &= CT_T^H(\mathcal{TR}_0) + \sum_{i=1}^{n-1} CT_U(\mathcal{TR}_i)
\end{aligned}
\tag{1}
$$

The inaccuracy costs are instead reported in Eq. 2. For $\mathcal{M}_0$, the inaccuracy cost is given by the sum of the costs obtained by providing predictions using the model trained on the batch $\mathcal{M}_0$ and tested on each new test set batch $\mathcal{TE}_i$. For the other three models, instead, the cost of inaccuracy, at the arrival of the train set batch $\mathcal{TR}_i$, is given by the inaccuracy cost obtained from models trained, updated and/or optimised on a train set that takes into account the information on the data of all train set batches up to $\mathcal{TR}_i$, and evaluated on the test set batch $\mathcal{TE}_i$. The inaccuracy cost is given by the sum of the costs for each new training and test batch $\mathcal{TR}_i$ and $\mathcal{TE}_i$.

$$\begin{aligned} CI_{\mathcal{M}_0}(\mathcal{TR}_{0\ldots n}, \mathcal{TE}_{0\ldots n}) &= \sum_{i=0}^{n-1} CI_{\mathcal{M}_0}(\mathcal{TR}_0, \mathcal{TE}_i) \\ CI_{\mathcal{M}_1}(\mathcal{TR}_{0\ldots n}, \mathcal{TE}_{0\ldots n}) &= \sum_{i=0}^{n-1} CI_{\mathcal{M}_1}(\bigcup_{j=0}^{i} \mathcal{TR}_j, \mathcal{TE}_i) \\ CI_{\mathcal{M}_2}(\mathcal{TR}_{0\ldots n}, \mathcal{TE}_{0\ldots n}) &= \sum_{i=0}^{n-1} CI_{\mathcal{M}_2}(\bigcup_{j=0}^{i} \mathcal{TR}_j, \mathcal{TE}_i) \\ CI_{\mathcal{M}_3}(\mathcal{TR}_{0\ldots n}, \mathcal{TE}_{0\ldots n}) &= \sum_{i=0}^{n-1} CI_{\mathcal{M}_3}(\bigcup_{j=0}^{i} \mathcal{TR}_j, \mathcal{TE}_i) \end{aligned} \tag{2}$$

We assume we can approximate the inaccuracy cost of the model $\mathcal{M}_0$ tested on the i-th test set batch $\mathcal{TE}_i$—$CI_{\mathcal{M}_0}(\mathcal{TR}_0, \mathcal{TE}_i)$—with the inaccuracy cost of $\mathcal{M}_0$ tested on $\mathcal{TE}_1$ plus an extra inaccuracy cost $\delta_i^{\mathcal{M}_0}$, i.e. $CI_{\mathcal{M}_0}(\mathcal{TR}_0, \mathcal{TE}_i) = CI_{\mathcal{M}_0}(\mathcal{TR}_0, \mathcal{TE}_1) + \delta_i^{\mathcal{M}_0}$. Similarly, the inaccuracy costs of the other three models—$CI_{\mathcal{M}}(\bigcup_{j=0}^{i} \mathcal{TR}_j, \mathcal{TE}_i)$—can be approximated with the inaccuracy cost computed on the first test set plus an extra inaccuracy cost $\delta_i^{\mathcal{M}}$, i.e. $CI_{\mathcal{M}}(\bigcup_{j=0}^{i} \mathcal{TR}_j, \mathcal{TE}_i)$=$CI_{\mathcal{M}}(\bigcup_{j=0}^{i} \mathcal{TR}_j$,$\mathcal{TE}_1)$+$\delta_i^{\mathcal{M}}$.

Defining c_t as the time hourly cost and c_e as the unary prediction error cost, we can compute the time cost and the inaccuracy cost of a model $\mathcal{M}$ starting from the time required for training and updating the model $T_{\mathcal{M}}$ and from the number of prediction errors (i.e. false positive + false negatives) $E_{\mathcal{M}}$ as:

$$\begin{aligned} CT_{\mathcal{M}}(\mathcal{TR}, \mathcal{TE}) &= c_t * T_{\mathcal{M}} \\ CI_{\mathcal{M}}(\mathcal{TR}, \mathcal{TE}) &= c_e * E_{\mathcal{M}} \end{aligned} \tag{3}$$

In order to have an estimate of the costs in our scenario, we instantiated such a cost-effectiveness framework. In detail, we set the extra-inaccuracy costs to 0 ($\delta_i^{\mathcal{M}} = 0$) and chose a couple of sample configurations for the prediction error unitary costs and for the hourly unitary cost ($c_t = 0.1$, $c_e = 100$ and $c_t = 100$, $c_e = 0.1$), as well as for the number of batches n ($n = 1$ and $n = 5$). Table 8 reports the obtained results using as reference values the ones of the experimental setting 40%-40%. The best result among the four strategies for each outcome is emphasised in italic, while the results that differ from the best ones for less than 1 are emphasised in bold.

The results in the table show that, depending on the unitary costs of time and prediction errors, differences can exist in the choice of the cheapest model. In both settings, $\mathcal{M}_3$ seems to be the cheapest model for most of the labellings and for different numbers of batches. In the setting in which $c_t = 0.1$ and $c_e = 100$, due to the low hourly unitary cost, $\mathcal{M}_2$ is the cheapest model for some of the labellings. Moreover, no significant differences exist in terms of costs when more and more data batches arrive (at least for the specific assumptions made for this cost-effectiveness framework). In the other setting, i.e. when the time cost is much higher than the error cost ($c_t = 100$ and $c_e = 0.1$), $\mathcal{S}_2$ is always the most expensive update strategy, due to its substantial training time. Moreover, in this setting, the cheapest update strategy can change when the number of arriving data batches increases. Indeed, while with only one batch of data, for some of the labellings (e.g. ϕ_{12}), $\mathcal{M}_1$ is slightly cheaper than or has the same cost as $\mathcal{M}_3$, in the long run, $\mathcal{M}_3$ is cheaper than $\mathcal{M}_1$.

To sum up, this instantiation of the cost-effectiveness framework confirms the results of the plots reported in Fig. 5, i.e. that overall $\mathcal{S}_1$ and $\mathcal{S}_3$ are the strategies providing the best balance between time and accuracy and hence the cheapest update strategies. Moreover, the analysis suggests that, based on the unitary costs of time and errors, as well as on the number of available data batches, differences can exist related to the best update strategy, although $\mathcal{M}_3$ seems to be consistently cheaper for most of the tested settings.

6.3 Threats to validity

The main threats affecting the validity of the evaluation carried out are external validity threats, limiting the generalisability of the results. Indeed, although we investigated the usage of different update strategies on different types of labellings, we limited the investigation to outcome predictions and to classification techniques typically used with this type of predictions. We plan to inspect other types of predictions, i.e. numeric and sequence predictions, together with typical techniques used with them, i.e. regression and deep learning techniques, for future work.

Finally, the lack of an exhaustive investigation of the hyperparameter values affects the construction validity of our experimentation. We limited this threat by using standard techniques for hyperparameter optimisation Bergstra et al. [5].

7 Related work

To the best of our knowledge, no other work exists on the comparison of update strategies for Predictive Process Monitoring models with the exception of the two by Pauwels and Calders [31] and Maisenbacher and Weidlich Pauwels and Calders [26]. We hence first position our work within the Predictive Process Monitoring field and then address a specific comparison with [31] and Maisenbacher and Weidlich [26].

We can classify Predictive Process Monitoring works based on the types of predictions they provide. A first group of approaches deals with numeric predictions, and, in particular, predictions related to time van der Aalst et al. [43], Folino et al. [16], Rogge-Solti and Weske [35]. A second group of approaches focuses on the prediction of next activities. These approaches mainly use deep learning techniques—specifically techniques based on LSTM neural networks Tax et al. [38], Di Francescomarino et al. [14], Camargo et al. [9], Brunk et al. [7], Taymouri et al. [39]. These studies have shown that when the datasets are large, deep learning techniques can outperform techniques based on classical Machine Learning techniques. A third group of approaches deals with outcome predictions Teinemaa et al. [40], Maggi et al. [25], Di Francescomarino et al. [12], Leontjeva et al. [22], which are the ones we focus on. A key difference between these works and the work presented in this paper is that we do not aim at proposing/supporting a specific outcome prediction method, rather we aim at evaluating different update strategies.

The work by Pauwels and Calders [31] leverages deep learning models to address the challenge of next activity prediction in the context of incremental Predictive Process Monitoring. The goal of their paper is twofold: they explore different strategies to update a model over time for next-activity prediction, and they investigate the potential of neural networks for the incremental Predictive Process Monitoring scenario. The goal is reached by (i) identifying different settings related to the data to use for training, updating, and testing the models, both in a static and a dynamic scenario; and (ii) showing the positive impact of catastrophic forgetting of deep learning models for the Predictive Process Monitoring use-case. In our work, we focus on another type of techniques/predictions, i.e. we aim at investigating the potential

Table 8 Cost-effectiveness framework instantiation with $\delta_i^{\mathcal{M}} = 0$ and experimental setting 40%–40%

Dataset	ϕ	$\mathcal{M}_0$	$\mathcal{M}_1$	$\mathcal{M}_2$	$\mathcal{M}_3$
(a) n=1, $c_e = 100$, and $c_t = 0.1$					
BPIC11	ϕ_{11}	7800.5	***4600.5***	4601.52	4800.5
	ϕ_{12}	5001.89	3401.89	***2502.83***	4301.89
	ϕ_{13}	4300.52	3900.52	***3501.19***	4100.52
BPIC12	ϕ_{21}	46600.34	36500.35	31001.14	***10900.34***
	ϕ_{22}	14800.37	13400.37	6601.27	***5400.37***
	ϕ_{23}	20900.5	274000.5	***19701.66***	20900.5
BPIC15	ϕ_{31}	5400.05	6000.5	6200.1	***2900.05***
	ϕ_{32}	3800.05	4100.05	3600.14	***3400.05***
	ϕ_{33}	2900.06	3300.06	3100.15	***300.06***
BPIC18	ϕ_{41}	236200.8	2600.81	20901.08	***300.81***
DriftRIO1	ϕ_{51}	58200.02	***11000.02***	11400.05	***11000.02***
DriftRIO2	ϕ_{61}	***7900.02***	***7900.02***	**7900.05**	***7900.02***
Dataset	ϕ	$\mathcal{M}_0$	$\mathcal{M}_1$	$\mathcal{M}_2$	$\mathcal{M}_3$
(b) n=5, $c_e = 100$, and $c_t = 0.1$					
BPIC11	ϕ_{11}	39500.5	***23000.5***	23015.82	24000.5
	ϕ_{12}	25001.89	17001.91	***12516***	21501.9
	ϕ_{13}	21500.52	19500.53	***17510.6***	20500.52
BPIC12	ϕ_{21}	233000.3	182500.4	155012.4	***54500.34***
	ϕ_{22}	74000.37	67000.37	33024.94	***27000.37***
	ϕ_{23}	104500.5	137000.5	***98517.94***	104500.5
BPIC15	ϕ_{31}	27000.05	30000.05	30500.74	***14500.05***
	ϕ_{32}	19000.05	20500.05	18001.33	***17000.05***
	ϕ_{33}	14500.06	16500.06	15501.36	***1500.06***
BPIC18	ϕ_{41}	1181001	13000.86	57000.43	***1500.813***
DriftRIO1	ϕ_{51}	291000	55000.03	57000.43	***55000.02***
DriftRIO2	ϕ_{61}	***39500.02***	***39500.02***	**39500.38**	***39500.02***
Dataset	ϕ	$\mathcal{M}_0$	$\mathcal{M}_1$	$\mathcal{M}_2$	$\mathcal{M}_3$
(c) n=1, $c_e = 0.1$, and $c_t = 100$					
BPIC11	ϕ_{11}	508.72	***505.66***	1526.68	**505.8**
	ϕ_{12}	1894.61	***1894.51***	2831.03	**1894.69**
	ϕ_{13}	***526.55***	**526.65**	1197.72	**526.6**
BPIC12	ϕ_{21}	386.04	381.67	1174.67	***350.4***
	ϕ_{22}	381.52	380.12	1278.49	***372.15***
	ϕ_{23}	***521.09***	527.59	1682.64	**521.59**

Table 8 continued

Dataset	ϕ	$\mathcal{M}_0$	$\mathcal{M}_1$	$\mathcal{M}_2$	$\mathcal{M}_3$
BPIC15	ϕ_{31}	54.04	54.78	100.6	***51.62***
	ϕ_{32}	**56.88**	**57.27**	141.71	***56.54***
	ϕ_{33}	61.96	62.38	149.16	***59.38***
BPIC18	ϕ_{41}	1047.42	817.13	1105.54	***811.91***
DriftRIO1	ϕ_{51}	82.62	***35.47***	62.96	***35.47***
DriftRIO2	ϕ_{61}	***31.04***	***31.04***	54.98	***31.04***
Dataset	ϕ	$\mathcal{M}_0$	$\mathcal{M}_1$	$\mathcal{M}_2$	$\mathcal{M}_3$
(d) n=5, $c_e = 0.1$, and $c_t = 100$					
BPIC11	ϕ_{11}	539.92	**526**	15841.42	***525.33***
	ϕ_{12}	***1914.61***	1929.11	15985.86	**1915**
	ϕ_{13}	***543.75***	549.25	10619.33	544
BPIC12	ϕ_{21}	572.44	607.78	12557.78	***394.22***
	ϕ_{22}	440.72	433.72	13977.22	***393.86***
	ϕ_{23}	***604.69***	637.19	18039.94	607.19
BPIC15	ϕ_{31}	75.64	80.72	767.06	***63.56***
	ϕ_{32}	72.08	74.83	1346.5	***70.36***
	ϕ_{33}	73.56	75.97	1379.56	***60.69***
BPIC18	ϕ_{41}	1992.22	873.81	5016.97	***814.67***
DriftRIO1	ϕ_{51}	315.42	**80.25**	488.5	***79.69***
DriftRIO2	ϕ_{61}	***62.64***	***62.64***	421.81	***62.64***

of classical Machine Learning models in the Predictive Process Monitoring scenario for the prediction of an outcome.

The work by Maisenbacher and Weidlich [26] is the only one we are aware of that exploits classical incremental Machine Learning in the context of Predictive Process Monitoring. The goal of that paper is to show the usefulness of incremental techniques in the presence of Concept Drift. The goal is proved by performing an evaluation over synthetic logs which exhibit different types of Concept Drifts. In our work, we aim at comparatively investigating four different model update strategies (which include the case of the incremental update) both in terms of accuracy of the results and in terms of time required to update the models. We carry on our evaluation on real-life and synthetic logs with and without an explicit Concept Drift.

8 Conclusion

In this paper, we have provided a first investigation of different update strategies for Predictive Process Monitoring models in the context of the outcome prediction problem. In particular, we have evaluated the performance of four update strategies, namely **do nothing**, **re-train with no hyperopt**, **full re-train**, and **incremental update**, applied to Random Forest, the reference

Table 9 The accuracy results related to the Perceptron model

Dataset	ϕ	$\mathcal{M}_0$	$\mathcal{M}_1$	$\mathcal{M}_2$	$\mathcal{M}_3$
(a) Setting 10–70%					
BPIC11	ϕ_{11}	0.504	0.536	***0.625***	0.566
	ϕ_{12}	0.682	0.413	***0.867***	0.749
	ϕ_{13}	0.676	**0.877**	***0.88***	0.863
BPIC12	ϕ_{21}	***0.69***	0.633	0.655	0.68
	ϕ_{22}	***0.72***	**0.711**	0.686	**0.719**
	ϕ_{23}	**0.515**	***0.518***	0.483	0.5
BPIC15	ϕ_{31}	0.675	0.725	0.732	***0.777***
	ϕ_{32}	0.827	0.809	0.832	***0.873***
	ϕ_{33}	0.745	0.841	***0.886***	0.788
BPIC18	ϕ_{41}	0.386	0.392	0.397	***0.509***
DriftRIO1	ϕ_{51}	0.369	**0.738**	**0.742**	***0.747***
DriftRIO2	ϕ_{61}	0.701	0.945	***0.961***	0.908
Dataset	ϕ	$\mathcal{M}_0$	$\mathcal{M}_1$	$\mathcal{M}_2$	$\mathcal{M}_3$
(b) Setting 40%–40%					
BPIC11	ϕ_{11}	0.596	0.643	0.625	***0.666***
	ϕ_{12}	0.802	0.788	***0.867***	0.807
	ϕ_{13}	0.826	**0.873**	***0.88***	**0.879**
BPIC12	ϕ_{21}	***0.658***	0.646	**0.655**	0.478
	ϕ_{22}	**0.682**	***0.69***	**0.686**	0.574
	ϕ_{23}	***0.524***	0.501	0.483	**0.521**
BPIC15	ϕ_{31}	0.651	0.698	**0.732**	***0.736***
	ϕ_{32}	***0.854***	0.801	0.832	0.836
	ϕ_{33}	**0.88**	0.864	***0.886***	0.876
BPIC18	ϕ_{41}	0.426	0.446	0.397	***0.574***
DriftRIO1	ϕ_{51}	0.727	0.747	0.742	***0.908***
DriftRIO2	ϕ_{61}	0.594	0.939	***0.961***	0.839

technique for outcome-oriented predictions, on a number of real and synthetic datasets with and without explicit Concept Drift. The cost-effectiveness of the different update strategies has been evaluated in the simple case where only one train and one test set are available, and in the more complex scenario where new batches of data become continuously available. The results show that the need to update a Predictive Process Monitoring model is real for typical real-life event logs (regardless of the presence of an explicit Concept Drift). They also show the potential of incremental learning strategies for Predictive Process Monitoring in real environments. An avenue for future work is the extension of our evaluation to different prediction problems such as remaining time and sequence predictions, which would, in turn, extend the evaluation to different reference Machine Learning techniques such as regression and LSTM, respectively. Also, a deeper investigation of the proposed cost-effectiveness framework in the context of the proposed update strategies will allow us to come up with more detailed best practices to guide the user in understanding which strategy is the most appropriate one under specific contextual conditions.

To conclude, we believe that the potential of incremental models is under-appreciated in the Predictive Process Monitoring field. To allow researchers to better understand the usefulness of the update strategies proposed in this paper, we made them readily available in the latest release of `Nirdizati` Rizzi et al. [34].

Acknowledgements We thank anonymous reviewers for their very useful comments and suggestions.

Funding Open access funding provided by Free University of Bozen-Bolzano within the CRUI-CARE Agreement.

A. Further Results

We report, in this appendix, the results obtained using Perceptron [29] (rather than Random Forest) as classifier. Tables 9a and 9b show the results obtained with such a classifier for the different strategies and for the different considered settings. Also for these experiments, the best result for each dataset and labelling is emphasised in italic, while the results that differ from the best ones for less than 0.01 are emphasised in bold.

As for the case of Random Forest, we can observe that the most accurate strategies are $\mathcal{M}_2$ and $\mathcal{M}_3$, with the exception of datasets BPIC12 and BPIC15 with labelling ϕ_{32} in the setting 40%–40%, in which the best results are obtained with $\mathcal{M}_0$. Moreover, we can observe a significant difference, in terms of performance, with respect to the Random Forest, for BPIC18: the accuracy obtained with Perceptron for this dataset is very low for all the four update strategies. Also with Perceptron, the accuracy is overall better for the 40%–40% setting w.r.t. 10%–70% and no relevant differences in terms of winning strategies for datasets with and without explicit Concept Drift can be devised.

Overall, the evaluation with Perceptron confirms that $\mathcal{M}_2$ and $\mathcal{M}_3$ (i.e. **full re-train** and **incremental update**, respectively) are the best performing update strategies in terms of accuracy.

References

1. 3TU Data Center, (2011) BPI Challenge 2011 Event Log. https://doi.org/10.4121/uuid:d9769f3d-0ab0-4fb8-803b-0d1120ffcf54
2. Back CO, Debois S, Slaats T (2019) Entropy as a measure of log variability. J. Data Semant. 8(2):129–156. https://doi.org/10.1007/s13740-019-00105-3
3. Bergstra J, Bardenet R, Bengio Y, Kegl B (2011) Algorithms for hyper-parameter optimization. In: Shawe Taylor J, Zemel RS, Bartlett PL, Pereira FCN, Weinberger KQ, eds, Advances in Neural Information

Processing Systems 24. In: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain., pp 2546–2554

4. Bergstra J, Bengio Y (2012) Random search for hyper-parameter optimization. J Mach Learn Res 13:281–305
5. Bergstra J, Yamins D, & Cox DD (2013) Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures, In: Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013, Vol. 28 of JMLR Workshop and Conference Proceedings, JMLR.org, pp 115–123
6. Bose RPJC, van der Aalst WMP, Zliobaite I, Pechenizkiy M (2014) Dealing with concept drifts in process mining. IEEE Trans. Neural Netw. Learning Syst. 25(1):154–171
7. Brunk J, Stottmeister J, Weinzierl S, Matzner M, Becker J (2020) Exploring the effect of context information on deep learning business process predictions. Journal of Decision Systems 1–16
8. Bughin J, Hazan E, Ramaswamy S, Chui M, Allas T, Dahlstrom P, Henke N, Trench M (2017) Artificial intelligence, the next digital frontier? McKinsey
9. Camargo M, Dumas M, & Rojas OG (2019) , Learning accurate LSTM models of business processes, In: T. T. Hildebrandt, B. F. van Dongen, M. oglinger, & J. Mendling, eds, Business Process Management - 17th International Conference, BPM 2019, Vienna, Austria, September 1-6, 2019, Proceedings, Vol. 11675 of Lecture Notes in Computer Science, Springer, pp 286–302
10. Carmona J, & Gavalda R (2012) Online techniques for dealing with concept drift in process mining, In: J. Hollmen, F. Klawonn, & A. Tucker, eds, Advances in Intelligent Data Analysis XI - 11th International Symposium, IDA 2012, Helsinki, Finland, October 25-27, 2012. Proceedings, Vol. 7619 of Lecture Notes in Computer Science, Springer, pp 90–102
11. Di Francescomarino C, Dumas M, Federici M, Ghidini C, Maggi FM, Rizzi W, Simonetto L (2018) Genetic algorithms for hyperparameter optimization in predictive business process monitoring. Inf. Syst. 74(Part):67–83
12. Di Francescomarino C, Dumas M, Maggi FM, Teinemaa I (2019) Clustering-based predictive process monitoring. IEEE Trans Serv Comput 12(6):896–909
13. Di Francescomarino C, Ghidini C, Maggi FM, & Milani F (2018) Predictive process monitoring methods: Which one suits me best?, In: M. Weske, M. Montali, I. Weber, & J. vom Brocke, eds, Business Process Management - 16th International Conference, BPM 2018, Sydney, NSW, Australia, September 9-14, 2018, Proceedings, Vol. 11080 of Lecture Notes in Computer Science, Springer, pp 462–479
14. Di Francescomarino C, Ghidini C, Maggi FM, Petrucci G, & Yeshchenko A (2017) An eye into the future: Leveraging a-priori knowledge in predictive business process monitoring, In: BPM, pp 252–268
15. Evermann J, Rehse J, Fettke P (2017) Predicting process behaviour using deep learning. Decision Support Syst 100:129–140
16. Folino F, Guarascio M, & Pontieri L (2012) Discovering context-aware models for predicting business process performances. In: R. Meersman, H. Panetto, T. S. Dillon, S. Rinderle-Ma, P. Dadam, X. Zhou, S. Pearson, A. Ferscha, S. Bergamaschi, & I. F. Cruz, eds, On the Move to Meaningful Internet Systems: OTM 2012, Confederated International Conferences: CoopIS, DOA-SVI, and ODBASE 2012, Rome, Italy, September 10-14, 2012. Proceedings, Part I, Vol. 7565 of Lecture Notes in Computer Science, Springer, pp 287–304
17. Gama J, Medas P, Castillo G Rodrigues PP (2004) Learning with drift detection. In: A. L. C. Bazzan, & S. Labidi, eds, 'Advances in Artificial Intelligence - SBIA 2004, 17th Brazilian Symposium on Artificial Intelligence, Sao Luis, Maranhao, Brazil, September 29 - October 1, 2004, Proceedings, Vol. 3171 of Lecture Notes in Computer Science, Springer, pp 286–295
18. Gama J, Zliobaite I, Bifet A, Pechenizkiy M, Bouchachia A (2014) A survey on concept drift adaptation. ACM Comput. Surv. 46(4):44:1-44:37
19. Gepperth A, & Hammer B (2016) Incremental learning algorithms and applications. In: 24th European Symposium on Artificial Neural Networks, ESANN 2016, Bruges, Belgium, April 27-29, 2016
20. Ho, TK (1995) , Random decision forests. In: Third International Conference on Document Analysis and Recognition, ICDAR 1995, August 14 - 15, 1995, Montreal, Canada. Volume I, IEEE Computer Society, pp 278–282
21. Jorbina K, Rozumnyi A, Verenich I, Di Francescomarino C, Dumas M, Ghidini C, Maggi FM., Rosa ML, & Raboczi S (2017) Nirdizati: A web-based tool for predictive process monitoring. In: R. Clariso, H. Leopold, J. Mendling, W. M. P. van der Aalst, A. Kumar, B. T. Pentland, & M. Weske, eds, Proceedings of the BPM Demo Track and BPM Dissertation Award co-located with 15th International Conference on Business Process Modeling (BPM 2017), Barcelona, Spain, September 13, 2017, Vol. 1920 of CEUR Workshop Proceedings, CEUR-WS.org
22. Leontjeva A, Conforti R, Di Francescomarino C, Dumas M, & Maggi, FM (2015) Complex symbolic sequence encodings for predictive monitoring of business processes. In: H. R. Motahari-Nezhad, J. Recker,

& M. Weidlich, eds, Business Process Management - 13th International Conference, BPM 2015, Innsbruck, Austria, August 31 - September 3, 2015, Proceedings, Vol. 9253 of Lecture Notes in Computer Science, Springer, pp 297–313

23. Maaradji A, Dumas M, Rosa ML, & Ostovar A (2015) Fast and accurate business process drift detection, In: H. R. Motahari-Nezhad, J. Recker & M. Weidlich, eds, Business Process Management - 13th International Conference, BPM 2015, Innsbruck, Austria, August 31 - September 3, 2015, Proceedings, Vol. 9253 of Lecture Notes in Computer Science, Springer, pp 406–422
24. Maggi FM, Burattin A, Cimitile M, & Sperduti A (2013) Online process discovery to detect concept drifts in ltl-based declarative process models, In: R. Meersman, H. Panetto, T. S. Dillon, J. Eder, Z. Bellahsene, N. Ritter, P. D. Leenheer, & D. Dou, eds, On the Move to Meaningful Internet Systems: OTM 2013 Conferences - Confederated International Conferences: CoopIS, DOA-Trusted Cloud, and ODBASE 2013, Graz, Austria, September 9-13, 2013. Proceedings, Vol. 8185 of Lecture Notes in Computer Science, Springer, pp 94–111
25. Maggi FM, Di Francescomarino C, Dumas M, & Ghidini C (2014) Predictive monitoring of business processes. In: M. Jarke, J. Mylopoulos, C. Quix, C. Rolland, Y. Manolopoulos, H. Mouratidis, & J. Horkoff, eds, Advanced Information Systems Engineering - 26th International Conference, CAiSE 2014, Thessaloniki, Greece, June 16-20, 2014. Proceedings, Vol. 8484 of Lecture Notes in Computer Science, Springer, pp 457–472
26. Maisenbacher M, Weidlich M (2017). In: Liu XF, Bellur U (eds) Handling concept drift in predictive process monitoring. IEEE Computer Society, pp 1–8
27. Metzger A, Franklin R, Engel Y (2012) Predictive monitoring of heterogeneous service-oriented business networks: The transport and logistics case, In: 2012 Annual SRII Global Conference, San Jose, CA, USA, July 24–27, 2012. IEEE Computer Society 313–322
28. Metzger A, Leitner P, Ivanovic D, Schmieders E, Franklin R, Carro M, Dustdar S, Pohl K (2015) Comparing and combining predictive business process monitoring techniques. IEEE Trans Syst, Man, Cybern: Syst 45(2):276–290
29. Minsky M, Papert SA (2017) Perceptrons: An introduction to computational geometry. MIT press
30. Munoz-Gama J, Martin N, Fernandez-Llatas C, Johnson OA, Sepulveda M, Helm E, Galvez-Yanjari V, Rojas E, Martinez-Millana A, Aloini D, Amantea IA, Andrews R, Arias M, Beerepoot I, Benevento E, Burattin A, Capurro D, Carmona J, Comuzzi M, Dalmas B, de la Fuente R, Di Francescomarino C, Ciccio CD, Gatta R, Ghidini C, Gonzalez-Lopez F, Ibanez-Sanchez G, Klasky HB, Prima Kurniati A, Lu X, Mannhardt F, Mans R, Marcos M, Medeiros de Carvalho R, Pegoraro M, Poon SK, Pufahl L, Reijers HA, Remy S, Rinderle-Ma S, Sacchi L, Seoane F, Song M, Stefanini A, Sulis E, ter Hofstede AH, Toussaint PJ, Traver V, Valero-Ramon Z, van de Weerd I, van der Aalst WM, Vanwersch R, Weske M, Wynn MT, Zerbato F (2022) Process mining for healthcare: characteristics and challenges. J Biomed Inform 22:36
31. Pauwels S, & Calders T (2021) Incremental predictive process monitoring: The next activity case, In: A. Polyvyanyy, M. T. Wynn, A. V. Looy, & M. Reichert, eds, Business Process Management - 19th International Conference, BPM 2021, Rome, Italy, September 06-10, 2021, Proceedings, Vol. 12875 of Lecture Notes in Computer Science, Springer, pp 123–140
32. Pnueli A (1977) The temporal logic of programs, in Foundations of Computer Science, 1977, 18th Annual Symposium on'. IEEE 46–57
33. Quinlan JR (1986) Induction of decision trees. Machine Learning 1(1):81–106
34. Rizzi W, Simonetto L, Di Francescomarino C, Ghidini C, Kasekamp T, & Maggi FM (2019) Nirdizati 2.0: New features and redesigned backend, In: B. Depaire, J. D. Smedt, M. Dumas, D. Fahland, A. Kumar, H. Leopold, M. Reichert, S. Rinderle-Ma, S. Schulte, S. Seidel, & W. M. P. van der Aalst, eds, Proceedings of the Dissertation Award, Doctoral Consortium, and Demonstration Track at BPM 2019 co-located with 17th International Conference on Business Process Management, BPM 2019, Vienna, Austria, September 1-6, 2019, Vol. 2420 of CEUR Workshop Proceedings, CEUR-WS.org, pp 154–158
35. Rogge-Solti A, & Weske M (2013) Prediction of remaining service execution time using stochastic petri nets with arbitrary firing delays. In: S. Basu, C. Pautasso, L. Zhang, & X. Fu, eds, Service-Oriented Computing - 11th International Conference, ICSOC 2013, Berlin, Germany, December 2-5, 2013, Proceedings, Vol. 8274 of Lecture Notes in Computer Science, Springer, pp 389–403
36. Rojas E, Munoz-Gama J, Sepulveda M, Capurro D (2016) Process mining in healthcare: a literature review. J Biomed Inform 61:224–236
37. Schlimmer JC, Granger RH (1986) Incremental learning from noisy data. Mach. Learn. 1(3):317–354
38. Tax N, Verenich I, La Rosa M, Dumas M (2017) Predictive business process monitoring with LSTM neural networks. pp 477–492
39. Taymouri F, Rosa ML, Erfani SM, Bozorgi ZD, Verenich I (2020) , Predictive business process monitoring via generative adversarial nets: The case of next event prediction. In: D. Fahland, C. Ghidini, J. Becker,

& M. Dumas, eds, Business Process Management - 18th International Conference, BPM 2020, Seville, Spain, September 13-18, 2020, Proceedings, Vol. 12168 of Lecture Notes in Computer Science, Springer, pp 237–256
40. Teinemaa I, Dumas M, La Rosa M, Maggi FM (2019) Outcome-oriented predictive process monitoring. Rev benchmark TKDD 13(2):17–57
41. Teinemaa I, Dumas M, Leontjeva A, Maggi FM (2018) Temporal stability in predictive process monitoring. Data Min. Knowl. Discov. 32(5):1306–1338
42. Tsaousis C (2013) Netdata, https://github.com/netdata/netdata/
43. van der Aalst W, Schonenberg MH, Song M (2011) Time prediction based on process mining. Inf. Syst. 36(2):450–475
44. van der Aalst W, amp, et al (2012) Process mining manifesto. In: Daniel F, Barkaoui K, Dustdar S (eds) Business Process Management Workshops (BPM 2011 International Workshops, Clermont-Ferrand, France, August 29, 2011, Revised Selected Papers, Part I). Lecture Notes in Business Information Processing, Springer, Germany, pp 169–194
45. van Dongen B (2012) Bpi challenge 2012. https://doi.org/10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f
46. van Dongen B (2015) BPIC 2015. https://doi.org/10.4121/uuid:31a308ef-c844-48da-948c-305d167a0ec1
47. van Dongen B & Borchert FF (2018) Bpi challenge 2018. https://data.4tu.nl/articles/dataset/BPI_Challenge_2018/12688355/1
48. Weber B, Reichert M, Rinderle-Ma S (2008) Change patterns and change support features - enhancing flexibility in process-aware information systems. Data Knowl. Eng. 66(3):438–466
49. Widmer G, Kubat M (1996) Learning in the presence of concept drift and hidden contexts. Mach. Learn. 23(1):69–101

Williams Rizzi received his bachelor's and master's degree in Computer Science from the University of Trento, Italy. He is currently a PhD student in Computer Science enrolled in a joint PhD Programme between Fondazione Bruno Kessler and Free University of Bozen-Bolzano, Italy. His research interests focus on the application of Machine Learning techniques to the Predictive Process Monitoring domain and he is currently actively developing Nirdizati, one of the state-of-the-art Predictive Process Monitoring tools. He took part in both industrial and research projects at the University of Trento and Fondazione Bruno Kessler. He has been publishing several papers on Predictive Process Monitoring (including one recognised with a distinguished paper award) in the top conferences in the field of Business Process Management and Information Systems since 2015.

Chiara Di Francescomarino is a researcher at Fondazione Bruno Kessler (FBK) in the Process and Data Intelligence (PDI) Unit. She received her PhD in Information and Communication Technologies from the University of Trento, working on business process modelling and reverse engineering from execution logs. She is currently working in the field of process mining, investigating problems related to process monitoring, process discovery, as well as Predictive Process Monitoring based on historical execution traces. She has published papers in the top business process conferences and journals (e.g. BPM, TKDE, and IS), and she has worked in local and international research projects. She serves as PC member in top conferences in the business process management field and as peer reviewer in international journals.

Chiara Ghidini is a senior Research Scientist at Fondazione Bruno Kessler (FBK), Trento, Italy, where she heads the Process & Data Intelligence (PDI) research unit. Her scientific work in the areas of Semantic Web, Knowledge Engineering and Representation, Multi-Agent Systems and Process Mining is internationally well known and recognised, and she has published more than 100 papers in those areas. Dr. Ghidini has acted as PC or track chair in the organisation of workshops and conferences on multiagent systems (EUMAS'04), Contexts-based representations (Context-03), Knowledge Engineering and Capturing (EKAW 2018), Semantic Web (ESWC 2012 and ISWC 2019) and Business Process Management (BPM2020). She has been involved in a number of international research projects, among which the FP7 Organic.Lingua and SO-PC-Pro European projects, as well as industrial projects with companies in the Trentino area.

Fabrizio Maria Maggi is currently Associate Professor at the Research Centre for Knowledge and Data (KRDB)—Faculty of Computer Science—Free University of Bozen-Bolzano. His research interest has focused in the last years on the application of Artificial Intelligence to Business Process Management. He authored more than 150 articles on process mining, declarative and hybrid business process notations, business constraint verification and monitoring, predictive business process monitoring. He serves as program committee member of the top conferences in the field of Business Process Management and Information Systems. He is leading the Rule Mining (RuM) initiative in collaboration with the University of Tartu aimed at providing tool support for declarative process mining.