





Updating Prediction Models for Predictive Process Monitoring

Alfonso E. Márquez-Chamorro^{1,2}(✉) , Isabel A. Nepomuceno-Chamorro¹ ,
Manuel Resinas^{1,2} , and Antonio Ruiz-Cortés^{1,2} 

¹ I3US Institute, Universidad de Sevilla, Seville, Spain
{`amarquez6`, `inepomuceno`, `resinas`, `aruiz`}@us.es

² SCORE Lab, Universidad de Sevilla, Seville, Spain

Abstract. Predictive monitoring is a key activity in some Process-Aware Information Systems (PAIS) such as information systems for operational management support. Unforeseen circumstances like COVID can introduce changes in human behaviour, processes, or computing resources, which lead the owner of the process or information system to consider whether the quality of the predictions made by the system (e.g., mean time to solution) is still good enough, and if not, which amount of data and how often the system should be trained to maintain the quality of the predictions. To answer these questions, we propose, compare, and evaluate different strategies for selecting the amount of information required to update the predictive model in a context of offline learning. We performed an empirical evaluation using three real-world datasets that span between 2 and 13 years to validate the different strategies which show a significant enhancement in the prediction accuracy with respect to a non-update strategy.

Keywords: Predictive process monitoring · Process mining · Process-aware information systems · Prediction models · Model updating

1 Introduction

Predictive process monitoring (PPM) provides proactive and corrective actions to improve the process performance and mitigate potential risks in real time. PPM retrieves information from Process-Aware Information Systems (PAIS) stored in event logs to make predictions of evaluation metrics, also known as process performance indicators (PPIs) [1]. A path extensively followed in the literature for predictive monitoring is adapting existing machine learning techniques [2] such as decision trees, clustering methods or neural networks to obtain predictive models with higher accuracy. When these approaches are used, the

Work funded by grants RTI2018-101204-B-C21 and RTI2018-101204-B-C22 funded by MCIN/ AEI/ 10.13039/501100011033/ and ERDF A way of making Europe; grant P18-FR-2895 and US-1381595 funded by Junta de Andalucía/ERDF, UE.

typical procedure for predictive monitoring comprises two steps. First, a training stage in which the predictive models are trained using data collected in the event logs. Second, once the model is built, it is deployed and it is used to predict PPIs for current and/or future process executions.

In the absence of significant changes, *ceteris paribus* (all else being equal), this approach works fine, but processes are subject to continuous changes. For instance, the response to COVID may introduce new ways of performing activities, users can behave in a different way, or human or computing resources can change over time. These changes may negatively affect the performance of the predictive model since the data used to train them does not reflect reality any more. Therefore, the only way to keep this performance over a desired threshold is by adapting the model to the changes.

In the machine learning community, there are two main adaptation approaches for that, namely, online and offline learning. In online learning, the predictive model is being updated continuously from the data it receives. Conversely, in offline learning, the predictive model is rebuilt again from the ground. In this paper, we decide to focus on offline learning mainly for two reasons. Firstly, the pace of change and the pace of new events in the processes we are interested in, gives enough time to completely rebuild new models. Secondly, its use allows one to reuse a huge amount of machine learning techniques that are available for offline learning, which is much more comprehensive than that of online learning. Furthermore, these techniques do not need to make compromises to keep a reasonable learning time.

In this context, the goal of this paper is to provide details on how to face two of the questions that arise in the update of predictive models: “Which data should be considered in the new model that is being built?” and “How the selection of data does impact on the performance of the predictive models?”.

By answering these questions, we contribute to the state of the art on PPM by proposing six different strategies for updating predictive models (baseline, cumulative, non-cumulative, ensemble, sampling, and concept drift) and comparing their performance. Our experimentation was validated using three real-life event logs that span between 2 and 13 years. We have also performed a comparison of different well-known classifiers used in related literature.

The reminder of this paper is organized as follows. Section 2 summarises basic concepts in predictive monitoring. Section 3 presents the strategies for updating predictive models. The experiment and the discussion of the obtained results are presented in Sect. 4. Section 5 summarises the related work. Finally, Sect. 6 concludes the work and presents possible future directions.

2 Predictive Process Monitoring

In the following, we introduce some basic concepts of predictive process monitoring. As defined in [3], an *event log* (L) is composed of a set of *traces* (T). Each trace (T_i) reflects an execution of a process instance. Formally, we can express a trace as an ordered list of *events* $T_i = [E_{i_1}, \dots, E_{i_m}]$ where E_{i_1} represents the

first event and E_{i_m} the final event of trace T_i . Similarly, a log can be expressed as the set of traces for the instances that have finished in an interval of time $L = [T_1, \dots, T_n]$ where T_1 is the first and T_n is the last executed trace in the time interval. Finally, an event represents the execution of an activity of the process. Each event contains a set of *attributes* (a), which represents information related to such event, *e.g.* timestamp, the resource that executes the activity, or the value of some data used throughout the instance, $E_j = [a_{j_1}, \dots, a_{j_o}]$ where o determines the total number of attributes of the event.

A process *indicator* (I) is a quantifiable metric focused on measuring the progress toward a goal or strategic objective. Indicators can be classified into two types: single-instance indicators or aggregated indicators. The former is computed for each trace in the log using the values of the attributes of the events that compose this trace. Therefore, it can be defined as a function of a trace T_i , *i.e.* $I(T_i)$. This function can return a binary value, *e.g.* a determined condition fulfilled by the trace, or a real value, *e.g.* the duration of an activity. Instead, an aggregated indicator is computed for a set traces by aggregating a single-instance indicator using some aggregation function, *e.g.* sum or average. An example of this type of indicator could be the percentage of incidents solved in a certain period of time.

A predictive model for an indicator I is a function $P_I([E_{i_k}, \dots, E_{i_l}])$, with $k \leq l$, that computes a prediction for I from the partial trace $[E_{i_k}, \dots, E_{i_l}]$, where E_{i_l} is the last event that have occurred in trace T_i at a given moment. If $k = 1$, then all events that have occurred in the process instance at hand are considered. Instead, if $k = l$, then only the last event of the process instance is considered.

In order to train a predictive model \hat{I} for a key performance indicator (KPI) I , an encoded fixed-size representation \mathcal{C} of all the cases C , where $C \subseteq \mathcal{C}$, included in the training set is required. This encoding, generally represented as a feature matrix (X), should store enough information of the process, and will be used as input for the machine learning technique employed to build the model together with the value of the KPI I for each case in C , which represents the target variable (y). The feature matrix X is obtained after applying a sequence encoding function \mathcal{F} , which receives a set of cases C and returns a matrix X . Each row of the matrix represents an event E , *i.e.* the execution of an activity of the process, of a case $c \in C$ and each column represents the different (encoded) attributes a of the event. Various sequence encoding techniques have been proposed in the literature for this task such as last state encoding [4], aggregation encoding [5], or index-based encoding [6]. The other decision is if only one classifier is trained for the whole dataset or, on the contrary, if cases are grouped into several buckets and a different classifier is trained for each one. Several case bucketing techniques have been proposed in the literature [7]: Zero bucketing [5], prefix length bucketing [6], or cluster bucketing [8]. After these two decisions are made, a predictive model is built using some machine learning algorithm using the pair (X, y) as the input.

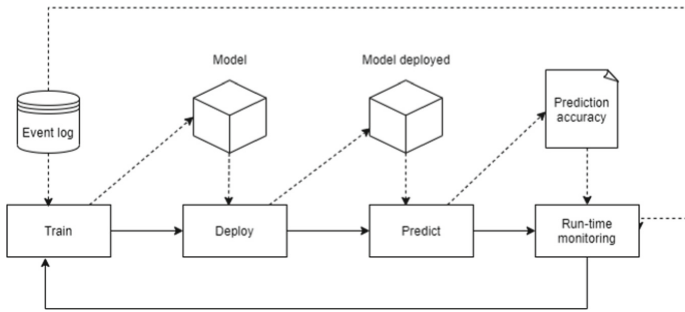


Fig. 1. Updating models system in a predictive monitoring process.

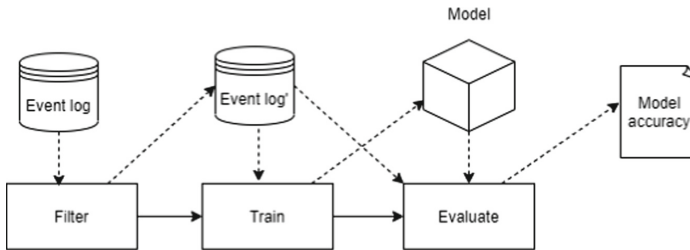


Fig. 2. Training stage.

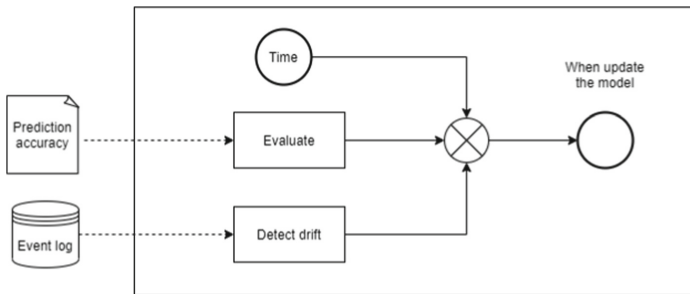


Fig. 3. Run-time monitoring stage.

An indicator I can represent different issues, such as a certain outcome, the next activity of the process or remaining cycle time of a given process case. In this work we have focused on the outcome-based prediction. Therefore, we are predicting an outcome value per case instead of a value per each event.

3 Updating Predictive Models

Once the predictive models are generated following the mechanisms described in the previous section are deployed into production, they start making predictions

that can be used to timely react to operational issues. However, after a while, the way the process was performed might have changes; resources participating in the process may come and go; even the structure of the process may suffer changes. All these changes are ignored by the predictive model that was deployed some time ago, so these changes may negatively affect the performance of the predictive model. Therefore, it is necessary to provide mechanisms to update the predictive model. Figure 1 shows a system for the updating of predictive models described below. It consists of a training stage depicted in Fig. 2 (which involves the filtering of the event log, the generation of the predictive model and the evaluation of this model), the deployment of this model, the prediction using this model, and finally, a mechanism for the update of the deprecated models, named Run-time monitoring process as shown in Fig. 3. This mechanism can evaluate the model in terms of the performance of the predictions and decide when the predictive model should be updated according to three possible parameters: the time elapsed from the last deployed model, the accuracy of predictions and the possible occurrence of a concept drift [9].

As mentioned in the introduction, there are two approaches for updating predictive models, namely online and offline learning. In this paper, we focus on offline learning because it allows one to use the huge amount of machine learning techniques for offline learning, which is much more comprehensive than that of online learning and, furthermore, these techniques do not need to make compromises in order to keep a reasonable learning time. To the best of our knowledge, any other work related to offline strategies for updating predictive models appears in the literature, so that comparison with other papers is not possible. In the context of offline learning, two basic questions need to be answered to update the predictive model:

1. When should the predictive model be updated?
2. Which data should be considered in the new model that is being built?

For the first question, several strategies can be considered (Run-time monitoring stage in Fig. 3). The most straightforward is a periodical update of the model [10]. A reasonable deadline for the change of the model can be fixed, e.g. six-monthly periodicity, and then, a new generation of the model will be carried out. A different strategy might involve monitoring the accuracy of our predictions. When it begins to decrease over an uninterrupted period of time, it may be recommended to change the predictive model. A threshold of error can be set, and if the prediction exceeds this threshold, the prediction model will be updated. Finally, a third strategy could involve using the detection of drifts in processes [11] to trigger updates in the predictive model [10]. Several strategies could be also combined to design a more robust system.

Our focus in this paper is, however, on the second question. This question is relevant because of two reasons that involve the quality and cost of predictive models. Concerning the former, if the reason for updating the predictive model is because the process has changed, it is reasonable to think that learning past behaviours of the process may not be beneficial for the performance of the predictive model, so it might make sense not to include the whole data set, but only

the most recent behaviour. As for the latter, the computational cost of building a predictive model increases with the size of the input data set. Therefore, a goal should be to achieve the best predictive performance by using the smallest possible input data set. In [10], authors propose two possible solutions to the second question: retraining and incremental update of a predictive model, however, all predictive algorithms cannot learn incrementally (e.g. random forests). Therefore, we have collected a set of strategies for choosing the data set used for building a new predictive model regardless of the predictive algorithm used.

A strategy for choosing the data set used for building a new predictive model can be seen as a function \mathcal{S} that receives a training and test set pair (X, y) and returns another training and test set pair (X', y') , such that $X' \subseteq X$ and $y' \subseteq y$. Next we detail several possible data selection strategies. We use the notation $X_{[i,j]}$ to select the subset of X that is between i and j , where i and j could be either an instant in time such as the 7th of March of 2019, or an instance number since the first one received. Furthermore, if they take the value 0, it refers to the first event in the data set and if they take the value c , it refers to the last event received in the data set. Therefore, $X_{[0,c]} = X$.

In the following, we present the different strategies for the selection of data. Figure 5 shows a graphical representation of the different strategies described in this section. Figure 4 depicts an event log that will be used to explain the different strategies in Fig. 5. This event log is split into several intervals from t_1 to t_n . Each interval represents all the process instances executed during a certain period of time, e.g. six months.

1. **Baseline strategy (\mathcal{S}_B):** In this strategy the model is not updated throughout the life of process:

$$\mathcal{S}_B(X, y) = (X_{[0,c]}, y_{[0,c]})$$

Figure 5a shows a graphical representation of the baseline strategy. With this strategy, a first interval is selected as the training set, and it is not updated throughout the life of the process. We use the rest of the intervals as test sets.

2. **Cumulative strategy (\mathcal{S}_C):** This strategy involves including all the cases that are available for training since the beginning:

$$\mathcal{S}_C(X, y) = (X_{[0,c]}, y_{[0,c]})$$

Cumulative strategy is represented in Fig. 5b. This strategy involves adding all instances of a process as training set. We split the event log into training and test sets, and we incrementally add each interval from t_1 to t_n in the training set and use the rest for the test set.

3. **Non-cumulative strategy (\mathcal{S}_N):** This strategy involves choosing only the most recent cases for training. It includes a parameter tn that determines how many recent cases should be included in the training:

$$\mathcal{S}_N(X, y) = (X_{[c-tn,c]}, y_{[c-tn,c]})$$

The advantage of this strategy in comparison to the cumulative strategy is that it is more efficient computationally and it might solve problems derived from using cases that do not follow the current behaviour of the process. The drawback is that having less training instances might hurt the performance of the predictive model.

For the non-cumulative strategy (Fig. 5c), we select an unique interval in the training phase. In this manner, we only include the most recent cases for training.

4. **Ensemble strategy (\mathcal{S}_E):** This strategy is similar to the non-cumulative strategy because it only includes the most recent ts cases in training a new model. The difference is that, unlike the non-cumulative strategy, this strategy do not throw away older models, but keep them and combine them using some ensemble technique [12]. For instance, one can use a weighted voting technique in which the prediction of each model is considered a vote and combined using different weights for each model to make the final prediction. These weights can be updated each time a new model is added to the ensemble so that older models have a lower weight. To this end, weights can be modeled using an exponential decay function like $e^{-\lambda t}$. Furthermore, besides these weights, if the last model had very bad performance, we might be interested in removing it from the ensemble so that it does not hurt the overall performance. To this end, we set a threshold parameter so that if the quality metric of choice, e.g. f-score, of the previous model did not meet the threshold in the last interval, it is removed from the ensemble.

The advantage of this strategy is that it has almost the same computational cost as the non-cumulative strategy, but it helps to avoid discarding all of the old cases. However, the combination of the different models might not be as powerful as a model built using the cumulative strategy that includes all previous cases.

In this strategy, depicted in Fig. 5d, we choose the same training and test sets and keep older models to combine them using some ensemble technique to achieve better predictions.

5. **Sampling strategy (\mathcal{S}_S):** This strategy involves a weighted sampling of all the cases that are available for training since the beginning. It includes a parameter ts that determines the number of samples that must be obtained from the data:

$$\mathcal{S}_S(X, y) = (\textit{sampling}(X_{[0,c]}, ts), \textit{sampling}(y_{[0,c]}, ts))$$

Where *sampling* is a function that takes ts samples from X or y , respectively. Sampling can also be weighted so that it is more likely to obtain more recent samples than older samples. A similar approach as the one used in the ensemble strategy can be used here to define these weights. The advantage of this approach in comparison to the cumulative strategy is that it limits the computational cost of the new model. Furthermore, unlike the ensemble strategy it relies on the machine learning algorithm instead of in the voting mechanism to combine both information from old and recent cases.



Fig. 4. Representation of a split event log.

Figure 5e shows the Sampling strategy. We build the training set in an incremental way using a weighted strategy where recent samples are more likely to be selected than older samples.

- Drift strategy (\mathcal{S}_D):** This strategy is similar to the non-cumulative one. It includes the most recent cases for training and, when a concept drift is detected, we include as training set, all the cases after a certain time has passed since the drift has occurred.

Figure 5f shows the drift strategy. When the concept drift is detected, training set is built only with those cases executed after the drift detection.

4 Experimental Evaluation

As we stated in Sect. 1, the goal of this paper is to define the different strategies for the selection of data (described in Sect. 3) and compare the predictive performance of the different strategies proposed. Based on this goal, we define a research question for our experimentation: What is the impact of the different updating strategies on the accuracy of predictions?

The rest of the section is organized as follows: the experiment setup is detailed in Sect. 4.1. The different datasets used in the experimentation are described in Sect. 4.2. Finally, a discussion of the obtained results is provided in Sect. 4.3.

4.1 Experiment Setup

As predictive algorithm we have used random forest [13] as seen in previous works in the literature [14]. This technique combines predictor trees such that each tree depends on the values of a random vector tested independently and with the same distribution for each of them. In [14], authors highlight extreme gradient boosting (XGBoost) and random forest as two of the best techniques in predictive monitoring. Thus, we have selected random forest because the quality of results with respect to XGBoost is similar and it consumes less computational time.

We have selected a typical aggregation encoding described in [14] as one of the most used in the literature to encode the process cases and also one of the best performers [14]. Thus, all events since the beginning of the case are considered. An aggregation function is applied to the values taken by a specific attribute throughout the case lifetime. In our case, this function is the number of times that each specific attribute appears in the case (frequency encoding). We have not divided the cases in the event log into different buckets. This technique is named Zero bucketing as defined in [14]. We have also incorporated the order of the events as a new attribute in all the logs, as well as the elapsed time between

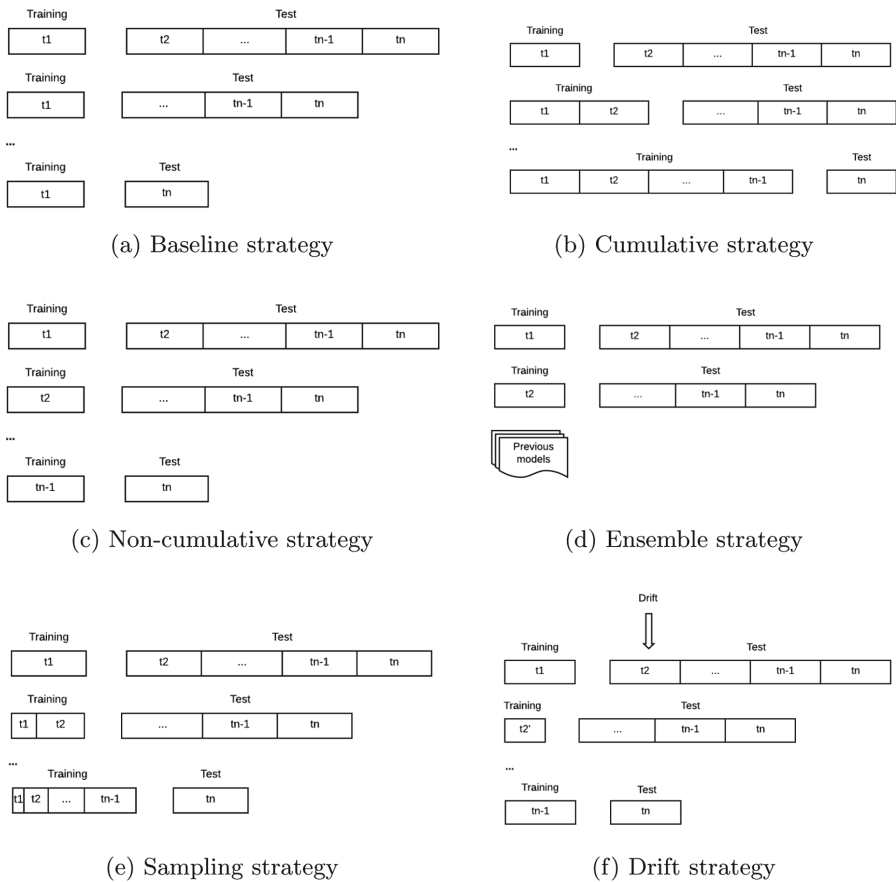


Fig. 5. Graphical representation of different proposed strategies for choosing the data set used for building a new predictive model.

the event and the beginning of the case and the time between the previous event and the current one. The details and the code of the experimentation are available online¹.

4.2 Event Logs

Three different real-life event logs were considered in our experiments: IT Department of an Andalusian organisation (ITA), BPI 2015 (BPI15) [15] and Traffic fines (TRAFFIC) [16]. These logs were chosen because they span several years: 2, 5 and 12, respectively, so they are useful to evaluate the effect of possible changes on the process over time.

¹ <https://github.com/isa-group/predictive-monitoring-evolution>.

ITA was extracted from the IT Department of an organisation in Andalusia. This dataset represents the incident management log of the IT Department in two years. In this scenario, a service level agreement (SLA) is established considering certain key performance indicators (KPIs). This SLA determines the penalties derived from the under-fulfilment of a threshold for each of the KPIs. Thus, predictive monitoring is necessary to warn the possibility of violation of the SLA. For the experiment, we used as target for the prediction if the incident is going to be put in a waiting state by the employee.

BPI15 is provided by five Dutch municipalities. The data contains all building permit applications over a period of approximately five years. There are many different activities present, denoted by both codes (attribute concept:name) and labels. The cases in the log contain information on the main application as well as objection procedures in various stages. Furthermore, information is available about the resource that carried out the task and on the cost of the application. This log covers the period October 2010 - March 2015. This log consists of five different datasets, one for municipality, where Log 1 includes 1,199 cases, Log 2 832 cases, Log 3 1,409 cases, Log 4 1,053 cases and Log 5 1,156 cases. All the logs contain between 40,000 and 60,000 events. The LTL rule used in the labelling function is described as $\varphi = G(\textit{"send confirmation receipt"}) \rightarrow F(\textit{"retrieve missing data"})$.

TRAFFIC represent a road traffic fine management process from Italian police. The log contains information about notifications sent about the fines and information about repayments. The targeted label to be predicted is based on whether the fine is repaid in full or is sent for credit collection. The resulting event log contains 129,615 cases, which were recorded between January 2000 and June 2012. Most of the cases consist of four events only.

4.3 Results

To perform the experimentation, we encode each dataset and split X into a set of intervals $X_{[I_1]}, \dots, X_{[I_n]}$. We also compute the target value y and split it into a set of possibly different intervals $y_{[I_1]}, \dots, y_{[I_m]}$. Like before, the size of intervals depends on the size of the dataset. With this setup, we train a predictive

Table 1. Average results of F-score for the different updating strategies.

Dataset	Strategy					
	Baseline	Non-Cum.	Cumulat.	Ensemble	Sampling	Drift
ITA	0.206883	0.265163	0.216884	0.250157	0.218832	NA
BPI city 1	0.858889	0.907668	0.900652	0.921329	0.908680	0.895364
BPI city 2	0.804032	0.849116	0.907067	0.881576	0.849492	0.790812
BPI city 3	0.710348	0.790268	0.815853	0.840622	0.756128	0.725569
BPI city 4	0.883692	0.925582	0.905460	0.925150	0.890478	0.832110
BPI city 5	0.896918	0.915899	0.913034	0.921801	0.915416	0.904564
TRAFFIC	0.652358	0.694372	0.703657	0.714230	0.705357	NA

model for each interval $X_{[I_i]}$ and evaluate it against all test intervals $y[I_k]$ such that I_k is after I_i . We have used F-score as accuracy measure [17] as seen in other works of the literature [14]. The F-score is the harmonic average of the precision and recall. F-score reaches its best value at 1. We have also detected the concept drifts for the different datasets using the algorithm included in the process mining framework PM4PY described in [18]. Specifically, we have found one concept drift for BPI city 1, two for BPI city 2, one for BPI city 3 and BPI city 4 and two for BPI city 5. We have not detected any drift for TRAFFIC and ITA datasets. These concept drifts have been used for the Drift strategy.

Table 1 summarises the results of the executions for the three datasets. The table is built assuming that the new model generated for each training interval is the one that is used to predict the values until a new model is created. In other words, we do not throw away models that are not performing better than a previous one. Furthermore, to ensure a fair comparison between approaches, we define training and test intervals using the number of instances instead of time. This avoids those cases in time intervals with fewer cases weight more than cases in others.

The values that we used for the intervals depend again on the dataset. For ITA, the training size was 39,000 cases, the interval between new models was 52,000 cases and the test interval was 13,000 cases. For all BPI datasets, the train size was 300 instances, and both the interval between new models and the test interval were 150 cases. Finally, for the TRAFFIC dataset, the train size was 8,641 cases, and the interval between new models and the test interval were 4,320 cases. In addition, for the Ensemble and Sampling strategies we assign weights to each interval using a decaying parameter of $e^{-x/3}$ and we set a threshold for the Ensemble of 0.5 except for ITA for which we set a threshold of 0.25.

From the table, we can conclude that there are no big differences between strategies in terms of F-Score. Nonetheless, as it was expected, the worst results are obtained with the Baseline strategy for all datasets (5–6% points on average less than the other strategies). On the other hand, the Ensemble strategy seems to perform better than the others in most datasets (4 out of 7), the second one is the Non-cumulative strategy (2 out of 7), and the third one is the Cumulative strategy (1 out of 7). The Sampling strategy is the one that seems to perform a bit worse. However, for the datasets in which we use large training

Table 2. Average results of F-score for the executions of different classifiers using BPI15 dataset.

Dataset	RF	BOOST	BN
BPI city 1	0.895364	0.7646200	0.533489
BPI city 2	0.790812	0.8395537	0.468736
BPI city 3	0.725569	0.7122018	0.440710
BPI city 4	0.832110	0.8472612	0.493650
BPI city 5	0.915416	0.9155702	0.450492
Average	0.831854	0.8158410	0.477415

sets like ITA and TRAFFIC, its performance is almost the same as the Cumulative strategy even when it uses much less instances as in input and hence, its computational cost is much more reduced. We can also conclude that Drift strategy does not perform as well as could be expected. This may be due to the fact that a low representative number of instances has been collected as training set after the concept drift has happened. On average, the three winner strategies (Non-cumulative, Cumulative and Ensemble) overcome the Baseline strategy in 4.71, 5 and 6.5% points. Considering the computational cost, it is quite evident that Cumulative strategy consumes more resources than the other two. Ensemble strategy has almost the same computational cost as the Non-cumulative strategy, however the use of older models, which are discarded in the Non-cumulative strategy, provides extra information that help to increase the accuracy prediction in almost 2 points more.

An extra experiment was performed to justify the use of random forest as machine learning algorithm in our experimentation. To this aim, we have compared the results obtained by random forest (RF) with those obtained by other three well-known classifiers also used in related literature [14]: Gradient Boosting (BOOST), and Bayesian Network (BN). For this experimentation we have used BPI15 dataset and drift strategy. We have used the Scikit-learn implementation of all the cited algorithms and we set the parameters of the algorithms by default. Regarding the F-score results in Table 2, we can appreciate a slight improvement of the results using random forest with respect to boosting algorithm. BN presents worse accuracy than RF and BOOST.

In summary, the main conclusion that can be drawn is that using all of the available data to rebuild the models is not a good choice because there is no win in performance and the computational cost is much higher. Furthermore, a significant improvement of accuracy prediction can be obtained using updated models versus non-updated models (Baseline strategy). This percentage can be increased from 5 to 15 points at best (Ensemble strategy). The Ensemble strategy provides good results and it reuses all models created previously, so it is more efficient computationally.

5 Related Work

Model updating has received increasing interest in the data mining community using diverse terminology such as, e.g., concept drift, incremental learning, stream data mining, or dynamic data mining [19]. However, a few works considers the updating of prediction models in the context of predictive process monitoring. Some of them [20, 21] describe the notion of concept drift which is a term applied in machine learning and data mining refers to situations when the relation between the input data and the target variable, which the model is trying to predict, changes over time in unforeseen ways [20]. In [21] authors introduce a paradigm to handle concept drift in predictive process monitoring and also present a systematic experimental study to define different incremental learning strategies and encodings of process traces suited for the predictive

monitoring of continuously evolving processes. Apart from that, a framework and specific techniques to detect when a process changes and to localize the parts of the process that have changed, are proposed in [20].

In [22], a remaining time prediction method along with a concept drift adaptation method is presented. As a predictor, they used an annotated transition system with probabilities obtained from Fuzzy Support Vector Machines based on process instance data. The predicted remaining time is obtained by summing up the durations of future activities estimated using Support Vector Machines. The concept drift adaptation method is based on a multimodel which is trained over different intervals of previous data, and assigns weights to the predictions of each model based on the difference in time between the model and the test data, factoring in an exponential decay and a periodic function.

Authors in [9] analyze which data should be selected for the retraining of the machine learning model after the detection of a concept drift. Therefore, they use different data selection strategies and consider the effects of the different retraining options in a real-life use case in process mining.

CONDA-PM is presented in [23], a four-phased framework that may guide process mining practitioners in assessing the maturity level of a concept drift analysis method. It covers a complete lifecycle of a concept drift analysis method with four phases. This method describes the phases and requirements of a concept drift analysis.

In [10], authors present a switching algorithm which combines the advantages of retraining and incremental updates. They test several drift detectors regarding performance on a real-world data set with incremental drift. They also provide a comparison among drift handling strategies and static models, showing that static models wear out over time and their performance decreases. Furthermore, a comparison among different drift handling strategies is provided.

Authors develop an incremental predictive process monitoring technique (Incremental Clustering-based and Incremental Index-based) applied to logs containing concept drifts in [24]. The updating of models is carried out at runtime; once a new case ends, it is added as a training example to generate a new model, through two algorithms (Hoeffding tree (HT) and Adaptive Hoeffding tree (AT)) and compare them with a standard classification algorithm (random forests). They also examine the impact of incremental learning techniques on real event logs with respect to traditional offline learning in terms of prediction accuracy and conclude that the incremental techniques allow for getting predictions are as accurate as the ones obtained with the periodic rediscovery of the predictive models.

The main highlights of our work are summarized in the following. Our proposal follows an offline strategy for prediction (some of the cited papers follow an online strategy and conclude that the results do not improve [24]), and propose six different strategies to decide how much data should be included in the model rebuild are presented and experimentations with three real-life datasets are performed to validate them. Although some works in the literature propose several strategies to retrain or update a predictive model, they are only focused

on the occurrence of concept drifts [9, 10]. Strategies defined in this work are not necessarily applied after the occurrence of a concept drift, they can be applied since the beginning of the running process instance. Furthermore, they are not limited by the ability to accurately detect concept drifts in the event log.

6 Conclusions

The goal of this paper is to propose and compare several strategies to choose the amount of data that should be used while rebuilding a predictive monitoring model in an offline learning setting. Based on this goal, we performed an empirical evaluation using three real-world datasets that span between 2 and 13 years and obtained the following conclusions.

We analysed the performance of the six data selection strategies described in Sect. 3 (Baseline, Cumulative, Non-cumulative, Ensemble, Sampling and Drift). To this aim, we summarised the results (F-score) of the executions for each of the datasets and we concluded that the Ensemble strategy provides better results and it reuses all models created previously, so it is more efficient computationally. A prediction improvement of up to 15% is achieved with this strategy.

Further research will include the development of techniques that help to decide when the predictive model should be updated. To this end, we plan to include some of the ideas included in the introduction to detect changes in the process. The generalizability of our results are subject to certain limitations. Despite these first promising results, our findings are based on three data sets only. In future work, we want to broaden the field of application by analyzing additional real-world data sets. Furthermore, only one predictive technique has been used to evaluate the strategies. Novel deep learning approaches can be considered for further works.

References

1. del Río-Ortega, A., Resinas, M., Durán, A., Bernárdez, B., Ruiz-Cortés, A., Toro, M.: Visual PPINOT: a graphical notation for process performance indicators. *Bus. Inf. Syst. Eng.* **61**, 137–161 (2019)
2. Márquez-Chamorro, A., Resinas, M., Ruiz-Cortés, A.: Predictive monitoring of business processes: a survey. *IEEE Trans. Serv. Comput.* **11**(6), 962–977 (2017)
3. Márquez-Chamorro, A.E., Revoredo, K., Resinas, M., Del-Río-Ortega, A., Santoro, F.M., Ruiz-Cortés, A.: Context-aware process performance indicator prediction. *IEEE Access* **8**, 222050–222063 (2020)
4. Polato, M., Sperduti, A., Burattin, A., Leoni, M.: Time and activity sequence prediction of business process instances. *Computing* **100**(9), 1005–1031 (2018). <https://doi.org/10.1007/s00607-018-0593-x>
5. de Leoni, M., van der Aalst, W.M., Dees, M.: A general process mining framework for correlating, predicting and clustering dynamic behavior based on event logs. *Inform. Syst.* **56**, 235–257 (2016)
6. Verenich, I., Nguyen, H., La Rosa, M., Dumas, M.: White-box prediction of process performance indicators via flow analysis. In: *ICSSP 2017*, pp. 85–94 (2017)

7. Teinemaa, I., Dumas, M., La Rosa, M., Maggi, F.: Outcome-oriented predictive process monitoring: Review and benchmark. CoRR abs/1707.06766 (2017)
8. Folino, F., Guarascio, M., Pontieri, L.: Discovering context-aware models for predicting business process performances. In: Meersman, R., et al. (eds.) OTM 2012. LNCS, vol. 7565, pp. 287–304. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33606-5_18
9. Baier, L., Reimold, J., Kühl, N.: Handling concept drift for predictions in business process mining. In: CBI 2020, vol. 1, pp. 76–83 (2020)
10. Baier, L., Kellner, V., Kühl, N., Satzger, G.: Switching scheme: a novel approach for handling incremental concept drift in real-world data sets. In: HICSS-54 (2021)
11. Maaradji, A., Dumas, M., La Rosa, M., Ostovar, A.: Detecting sudden and gradual drifts in business processes from execution traces. IEEE Trans. Knowl. Data Eng. **29**(10), 2140–2154 (2017)
12. Rokach, L.: Ensemble-based classifiers. Artif. Intell. Rev. **33**(1–2), 1–39 (2010)
13. Breiman, L.: Random forests. Mach. Learn. **45**(1), 5–32 (2001)
14. Verenich, I., Dumas, M., La Rosa, M., Maggi, F., Teinemaa, I.: Survey and cross-benchmark comparison of remaining time prediction methods in business process monitoring. [arXiv:1805.02896](https://arxiv.org/abs/1805.02896) (2018)
15. Van Dongen, B.: Bpi challenge 2015. tu delft. dataset. 10.4121/uuid:31a308ef-c844-48da-948c-305d167a0ec1 (2015)
16. De Leoni, M., Mannhardt, F.: Road traffic fine management process. Eindhoven university of technology. Dataset. <https://doi.org/10.4121/uuid:270fd440-1057-4fb9-89a9-b699b47990f5> (2015)
17. Powers, D.: Evaluation: From precision, recall and F-Factor to ROC, informedness, markedness & correlation. Mach. Learn. Technol. **2** (2008)
18. Bose, R., Aalst, W., Žliobaitė, I., Pechenizkiy, M.: Handling concept drift in process mining. In: CAISE 2011, pp. 391–405 (2011)
19. Guajardo, J., Weber, R., Miranda, J.: A model updating strategy for predicting time series with seasonal patterns. Appl. Soft Comput. **10**(1), 276–283 (2010)
20. Bose, R., Aalst, W., Žliobaitė, I., Pechenizkiy, M.: Dealing with concept drifts in process mining. IEEE Trans. Neural Netw. Learn. Syst. **25**(1), 154–171 (2014)
21. Maisenbacher, M., Weidlich, M.: Handling concept drift in predictive process monitoring. In: SCC 2017, pp. 1–8 (2017)
22. Firouzian, I., Zahedi, M., Hassanpour, H.: Investigation of the effect of concept drift on data-aware remaining time prediction of business processes. Int. J. Nonlinear Anal. Appl. **10**(2), 153–166 (2019)
23. Elkhawaga, G., Abuelkheir, M., Barakat, S., Riad, A., Reichert, M.: CONDA-PM-a systematic review and framework for concept drift analysis in process mining. Algorithms **13**(7), 161 (2020)
24. Francescomarino, C.D., Ghidini, C., Maggi, F.M., Rizzi, W., Persia, C.D.: Incremental predictive process monitoring: how to deal with the variability of real environments. [arXiv:1804.03967v1](https://arxiv.org/abs/1804.03967v1), p. 1 (2018)