Performance-preserving event log sampling for predictive monitoring

Mohammadreza Fani Sani[1] · Mozhgan Vazifehdoostirani[2] · Gyunam Park[1] · Marco Pegoraro[1] · Sebastiaan J. van Zelst[1,3] · Wil M. P. van der Aalst[1,3]

Abstract
Predictive process monitoring is a subfield of process mining that aims to estimate case or event features for running process instances. Such predictions are of significant interest to the process stakeholders. However, most of the state-of-the-art methods for predictive monitoring require the training of complex machine learning models, which is often inefficient. Moreover, most of these methods require a hyper-parameter optimization that requires several repetitions of the training process which is not feasible in many real-life applications. In this paper, we propose an instance selection procedure that allows sampling training process instances for prediction models. We show that our instance selection procedure allows for a significant increase of training speed for next activity and remaining time prediction methods while maintaining reliable levels of prediction accuracy.

Keywords Process mining · Predictive monitoring · Sampling · Machine learning · Deep learning · Instance selection

✉ Mohammadreza Fani Sani
fanisani@pads.rwth-aachen.de

Mozhgan Vazifehdoostirani
m.vazifehdoostirani@tue.nl

Gyunam Park
gnpark@pads.rwth-aachen.de

Marco Pegoraro
pegoraro@pads.rwth-aachen.de

Sebastiaan J. van Zelst
s.j.v.zelst@pads.rwth-aachen.de

Wil M. P. van der Aalst
wvdaalst@pads.rwth-aachen.de

[1] Chair of Process and Data Science, RWTH Aachen University, Aachen, Germany

[2] Industrial Engineering and Innovation Sciences, Eindhoven University of Technology, Eindhoven, The Netherlands

[3] Fraunhofer FIT, Birlinghoven Castle, Sankt Augustin, Germany

1 Introduction

The main goal of predictive process monitoring is to provide timely information by predicting the behavior of business processes (van der Aalst et al., 2011) and enabling proactive actions to improve the performance of the process (Park & van der Aalst, 2022). It provides various predictive information such as the next performing activity of a process instance (Breuker et al., 2016), e.g., patient and product, its waiting time for an activity, its remaining time to complete the process, etc Marquez-Chamorro et al. (2018). For instance, by predicting the long waiting time of a patient for registration, one can bypass the activity or add more resources to perform it.

A plethora of approaches have been proposed to support predictive process monitoring. In particular, with the recent breakthroughs in machine learning, various machine learning-based approaches have been developed (Marquez-Chamorro et al., 2018). The emergence of ensemble learning methods leads to improvement in accuracy in different areas (Breiman, 1996). eXtreme Gradient Boosting (XGBoost) (Chen & Guestrin, 2016) has shown promising results, often outperforming other ensemble methods such as Random Forest or using a single regression tree (Senderovich et al., 2017; Teinemaa et al., 2019b). Furthermore, techniques based on deep neural networks, e.g., Long-Short Term Memory (LSTM) networks, have shown high performance in different predictive tasks (Evermann et al., 2017).

However, machine learning-based techniques are computationally expensive due to their training process (Zhou et al., 2017). Moreover, they often require exhaustive hyperparameter-tuning to provide acceptable accuracy. Such limitations hinder the application of machine learning-based techniques into real-life business processes where new prediction models are required in short intervals to adapt to changing business situations. Business analysts need to test the efficiency and reliability of their conclusions via repeated training of different prediction models with different parameters (Marquez-Chamorro et al., 2018). Such long training time limits the application of the techniques when considering the limitations in time and hardware (Pourghassemi et al., 2020).

In this regard, *instance selection* has been studied as a promising direction of research to reduce original datasets to a manageable volume to perform machine learning tasks, while the quality of the results (e.g., accuracy) is maintained as if the original dataset was used (Garca et al., 2014). Instance selection techniques are categorized into two classes based on the way they select instances. First, some techniques select the instances at the boundaries of classes. For instance, Decremental Reduction Optimization Procedure (DROP) (Wilson & Martinez, 2000) selects instances using k-Nearest Neighbors by incrementally discarding an instance if its neighbors are correctly classified without the instance. The other techniques preserve the instances residing inside classes, e.g., Edited Nearest Neighbor (ENN) (Wilson, 1972) preserves instances by repeatedly discarding an instance if it does not belong to the class of the majority of its neighbors.

However, it is restricted to directly apply existing techniques for instance selection to predictive process monitoring training, since such techniques assume independence among instances (Wilson & Martinez, 2000). Instead, in predictive process monitoring, instances are computed from event data that are recorded by the information system supporting business processes (de Leoni et al., 2016). Thus, they are highly correlated (van der Aalst, 2016) with the notion of *case*, e.g., a manufacturing product in a factory and a customer in a store.

In this work, we suggest an instance selection approach for predicting the next activity, the remaining time and the outcome of the process that are main applications of predictive business process monitoring. By considering the characteristics of the event data, the proposed approach samples event data such that the training speed is improved while the accuracy of the resulting prediction model is maintained. We have evaluated the proposed methods using three real-life datasets and state-of-the-art techniques for predictive business process monitoring, including LSTM (Huang et al., 2015) and XGBoost (Chen & Guestrin, 2016).

This paper extends our earlier work presented in Sani et al. (2021) in the following dimensions: 1) Evaluating the applicability of the proposed approach, 2) Enhancing the accessibility of the work, and 3) Extending the discussion of strengths and limitations. First, we have evaluated the applicability of the proposed approach both task-wise and domain-wise. For the task-wise evaluation, we have selected the three most well-known predictive monitoring tasks (i.e., *next activity*, *remaining time*, and *outcome* predictions) and evaluated the performance of the proposed approach in the different tasks. For the domain-wise evaluation, we have evaluated the performance of our proposed approach in real-life event logs from different domains, including *finance*, *government*, and *healthcare* domains. Second, we have extended the accessibility of the proposed approach by implementing the proposed sampling methods in the Python platform as well as the Java platform. Finally, we have extensively discussed the strengths and limitations of the proposed approach, providing foundations for further research.

The remainder is organized as follows. We discuss the related work in Section 2. Next, we present the preliminaries in Section 3 and proposed methods in Section 4. Afterward, Section 5 evaluates the proposed methods using real-life event data and Section 6 provides discussions. Finally, Section 7 concludes the paper.

2 Related work

This section presents the related work on predictive process monitoring, time optimization, and instance sampling.

2.1 Predictive process monitoring

Predictive process monitoring is an exceedingly active field, both currently and historically, thanks to the compatibility of process sciences and other branches of data science that include inference techniques, such as statistics and machine learning. At its core, the fundamental component of many predictive monitoring approaches is the abstraction technique it uses to obtain a fixed-length representation of the process component subject to the prediction (often, but not always, process traces). In the earlier approaches, the need for such abstraction was overcome through model-aware techniques, employing process models and replay techniques on partial traces to abstract a flat representation of event sequences. Such process models are mostly automatically discovered from a set of available complete traces, and require perfect fitness on training instances (and, seldomly, also on unseen test instances). For instance, (van der Aalst et al., 2011) proposed a time prediction framework based on replaying partial traces on a transition system, effectively clustering training instances by control-flow information. This framework has later been the basis for a prediction method by Polato et al. (2018), where the transition system is annotated with an ensemble of SVR and Naïve Bayes classifiers, to perform a more accurate time estimation. Some more recent approaches split the predictive contribution of process models and machine learning models in a perspective-wise manner: for instance, (Park & Song, 2020) obtain a representation of the performance perspective using an annotated transition system, and design an ensemble with a deep neural network to obtain the final predictive model. A related approach, albeit more linked to the simulation domain and based on a Monte Carlo method, is the one proposed by Rogge-Solti and Weske (2013), which maps partial process instances in an enriched Petri net.

Recently, predictive process monitoring started to use a plethora of machine learning approaches, achieving varying degrees of success. For instance, (Teinemaa et al., 2016) provided a framework to combine text mining methods with Random Forest and Logistic Regression. Senderovich et al. (2017) studied the effect of using intra-case and inter-case features in predictive process monitoring and showed a promising result for XGBoost compared to other ensemble and linear methods. A comprehensive benchmark on using classical machine learning approaches for outcome-oriented predictive process monitoring tasks (Teinemaa et al., 2019b) has shown that the XGBoost is the best-performing classifier among different machine learning approaches such as SVM, Decision Tree, Random Forest, and logistic regression.

More recent methods are model-unaware and perform based on a single and more complex machine learning model instead of an ensemble. In fact, such an evolution of predictive monitoring mimics the advancement in the accuracy of newer machine learning approaches, specifically the numerous and sophisticated models based on deep neural networks that have been developed in the last decade. The LSTM network model has proven to be particularly effective for predictive monitoring (Evermann et al., 2017; Tax et al., 2017), since the recurrent architecture can natively support sequences of data of arbitrary length. It allows performing trace prediction while employing a fixed-length event abstraction, which can be based on control-flow alone (Evermann et al., 2017; Tax et al., 2017), data-aware (Navarin et al., 2017), time-aware (Nguyen et al., 2020), text-aware (Pegoraro et al., 2021), or model-aware (Park & Song, 2020). Additionally, rather than leveraging control-flow information for prediction, some recent research aims to use predictive monitoring to reconstruct missing control-flow attributes such as labels (van der Aa et al., 2021) or case identifiers (Pegoraro et al., 2022a; 2022b). However, the body of work currently standing in predictive process monitoring, regardless of architecture of the predictive model and/or target of the prediction, does not include strategies for performance-preserving sampling.

2.2 Time optimization and instance sampling

The latest research developments in the field of predictive monitoring, much like both this paper and the application of machine learning techniques in other domains, shifts its focus away from increasing the quality of prediction (with respect to a given error metric or a given benchmark), and specializes in enriching the results of the prediction on additional aspects or perspectives. Unlike the methods mentioned in the previous paragraph, this is usually obtained through dedicated machine learning models—or modifications thereof—rather than designing specific event- or trace-level abstractions. For instance, many scholars have attempted to make predictions more transparent through the use of explainable machine learning techniques (Verenich, 2019; Stierle et al., 2021; Sindhgatta et al., 2020; Galanti et al., 2020; Park et al., 2022). More related to our present work, (Pauwels & Calders, 2021) propose a technique to avoid the time expenditure caused by the retrain of machine learning models; this is necessary when they are not representative anymore—for instance, when changes occur in the underlying process (caused e.g. by concept drift). While in this paper we focus on the data, and we propose a solution based on sampling, Pauwels and Calders intervene on the model side, devising an incremental training schema which accounts for new information in an efficient way.

Another concept similar to the idea proposed in this paper, and of current interest in the field of machine learning, is *dataset distillation*: utilizing a dataset to obtain a smaller set of training instances that contain the same information (with respect to training a machine learning model) (Wang et al., 2020). While this is not considered sampling, since some instances of the distilled dataset are created ex-novo, it is an approach very similar to the one we illustrate in our paper.

The concept of instance sampling, or instance subset selection, is present in the context of process mining at large, albeit the development of such techniques is very recent. Some instance selection algorithms have been proposed to help classical process mining tasks. For example, (Fani Sani et al., 2021) proposes to use instance selection techniques to improve the performance of process discovery algorithms; in this context, the goal is to obtain automatically a descriptive model of the process, on the basis of the data recorded about the historical executions of process instances—the same starting point of the present work. Then, the work in Fani Sani et al. (2020) applies the same concept and integrates the edit distance to obtain a fast technique to approximate the conformance checking score of process traces: this consists of measuring the deviation between a model, which often represents the normative or prescribed behavior of a process, and the data, which represents the actual behavior of a process. This paper integrates the two aforementioned works, extending the effects of strategic sampling: while in Fani Sani et al. (2021) the sampling optimizes descriptive modeling and in Fani Sani et al. (2020) it optimizes process diagnostics, in this work it aids predictive modeling.

To the best of our knowledge, no work present in literature inspects the effects of building a training set for predictive process monitoring through a strategic process instance selection, with the exception of our previous work, which we extend in this paper (Sani et al., 2021). In this paper, we examine the underexplored topic of event data sampling and selection for predictive process monitoring, with the objective of assessing if and to which extent prediction quality can be retained when we utilize subsets of the training data.

3 Preliminaries

In this section, some process mining concepts such as event log and sampling are discussed. In process mining, we use events to provide insights into the execution of business processes. Event logs, i.e., collections of events representing the execution of several instances of a process, are the starting point of process mining algorithms. An example event log is shown in Table 1. Each *event* that relates to a row in the table is related to specific activities of the underlying process. Furthermore, we refer to a collection of events related to a specific process instance of the process as a *case* (represented by the *Case-id* column). Both cases and events may have different attributes. An event log that is a collection of events and cases is defined as follows.

Definition 1 (Event Log) Let $\mathcal{E}$ be the universe of events, $\mathcal{C}$ be the universe of cases, $\mathcal{AT}$ be the universe of attributes, and $\mathcal{U}$ be the universe of attribute values. Moreover, let $C \subseteq \mathcal{C}$ be a non-empty set of cases, let $E \subseteq \mathcal{E}$ be a non-empty set of events. We define (C, E, π_C, π_E) as an event log, where $\pi_C : C \times \mathcal{AT} \nrightarrow \mathcal{U}$ and $\pi_E : E \times \mathcal{AT} \nrightarrow \mathcal{U}$. Any event in the event log has a case, and thus, $\nexists_{e \in E}(\pi_E(e, case) \notin C)$ and $\bigcup\limits_{e \in E}(\pi_E(e, case)) = C$. Let $\mathcal{A} \subseteq \mathcal{U}$ be the universe of activities and $\mathcal{V} \subseteq \mathcal{A}^*$ be the universe of sequences of activities. For any $e \in E$, function $\pi_E(e, activity) \in \mathcal{A}$, which means that any event in the event log has an activity.

Table 1 Simple example of an event log

Case-id	Event-id	Activity name	Starting time	Finishing time ...	
⋮	⋮	⋮	⋮	⋮	…
7	35	Register(a)	2021-01-02 12:23	2021-01-02 12:25	…
7	36	Analyze defect(b)	2021-01-02 12:30	2021-01-02 12:40	…
7	37	Inform user(g)	2021-01-02 12:45	2021-01-02 12:47	…
8	39	Register(a)	2021-01-02 12:23	2021-01-02 13:15	…
7	40	Test repair(e)	2021-01-02 13:05	2021-01-02 13:20	…
7	41	Archive repair(h)	2021-01-02 13:21	2021-01-02 13:22	…
8	42	Analyze defect(b)	2021-01-02 12:30	2021-01-02 13:30	…
⋮	⋮	⋮	⋮	⋮	⋱

[1]Rows capture *events* recorded in the context of the execution of the process

[2]An event describes at what point in time an activity was performed

[3]Other data attributes may be available as well

Moreover, for any $c \in C$ function $\pi_C(c, variant) \in \mathcal{A}^* \setminus \{\langle\rangle\}$ that means any case in the event log has a variant.

Therefore, there are some mandatory attributes that are *case* and *activity* for events and *variants* for cases. For example, for event with Event-id equals to 35 in Table 1, $\pi_E(e, case))=7$ and $\pi_E(e, activity)=$Register(a).

Variants are the sequence of activities that are presented in each case. For example, for case 7 in Table 1, the variant is $\langle a, b, g, e, h\rangle$ (for the simplicity we show each activity by a letter). Variant information plays an important role an in some process mining applications, e.g., process discovery and conformance checking, just this information is considered. In this regard, event logs are considered as a multiset of sequences of activities. In the following, a simple event log is defined.

Definition 2 (Simple event log) Let $\mathcal{A}$ be the universe of activities and let the universe of multisets over a set X be denoted by $\mathcal{B}(X)$. A simple event log is $L \in \mathcal{B}(\mathcal{A}^*)$. Moreover, let $\mathcal{EL}$ be the universe of event logs and $EL=(C, E, \pi_C, \pi_E) \in \mathcal{EL}$ be an event log. We define function $sl{:}\mathcal{EL} \rightarrow \mathcal{B}(\mathcal{A}^*)$ returns the simple event log of an event log where $sl(EL)=[\sigma^k \mid \sigma \in \{\pi_C(c, variant) \mid c \in C\} \wedge k=\Sigma_{c\in C}\big(\pi_C(c, variant)=\sigma\big)]$. The set of unique variants in the event log is denoted by $\overline{sl(EL)}=\{\pi_C(c, variant) \mid c \in C\}$.

Therefore, sl returns the multiset of variants in the event logs. Note that the size of a simple event log equals the number of cases in the event logs, i.e., $|sl(EL)|=|C|$

In this paper, we use sampling techniques to reduce the size of event logs. An event log sampling method is defined as follows.

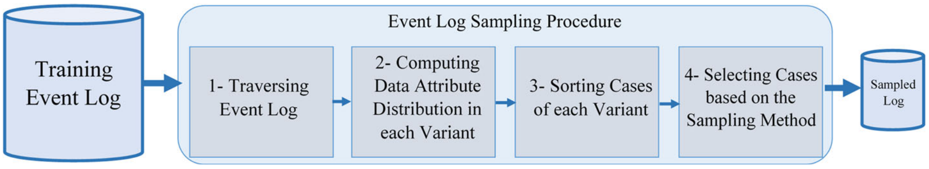

Training Event Log
Event Log Sampling Procedure
1- Traversing Event Log
2- Computing Data Attribute Distribution in each Variant
3- Sorting Cases of each Variant
4- Selecting Cases based on the Sampling Method
Sampled Log

Fig. 1 A schematic view of the proposed sampling procedure

Definition 3 (Event log sampling) Let $\mathcal{EL}$ be the universe of event logs and be the universe of activities. Moreover, let $EL{=}(C, E, \pi_C, \pi_E){\in}\mathcal{EL}$ be an event log, we define function $\delta{:}\mathcal{EL}{\to}\mathcal{EL}$ that returns the sampled event log where if $(C', E', \pi'_C, \pi'_E){=}\delta(EL)$, then $C'{\subseteq}C$, $E'{\subseteq}E$, $\pi'_E{\subseteq}\pi_E$, $\pi'_C{\subseteq}\pi_C$, and consequently, $\overline{sl(\delta(EL))}{\subseteq}\overline{sl(EL)}$. We define that δ is a variant-preserving sampling if $\overline{sl(\delta(EL))}{=}\overline{sl(EL)}$.

In other words, a sampling method is variant-preserving if and only if all the variants of the original event log are presented in the sampled event log.

To use machine learning methods for prediction, we usually need to transfer each case to one or more features. The feature is defined as follows.

Definition 4 (Feature) Let $\mathcal{AT}$ be the universe of attributes, $\mathcal{U}$ be the universe of attribute values, and $\mathcal{C}$ be the universe of cases. Moreover, let $AT{\subseteq}\mathcal{AT}$ be a set of attributes. A feature is a relation between a sequence of attributes' values for AT and the target attribute value, i.e., $f{\in}(\mathcal{U}^{|AT|}{\times}\mathcal{U})$. We define $fe{:}\mathcal{C}{\times}\mathcal{EL}{\to}\mathcal{B}(\mathcal{U}^{|AT|}{\times}\mathcal{U})$ is a function that receives a case and an event log, and returns a multiset of features.

For the next and final activity prediction, the target attribute value should be an activity. However, for the remaining time prediction, the target attribute value is a numerical value. Moreover, a case in the event log may have different features. For example, suppose that we only consider the activities. For the case $\langle a, b, c, d\rangle$, we may have $(\langle a\rangle, b)$, $(\langle a, b\rangle, c)$, and $(\langle a, b, c\rangle, d)$ as features. Furthermore, $\sum\limits_{c\in C} fe(c, EL)$ are the corresponding features of event log $EL{=}(C, E, \pi_C, \pi_E)$ that could be given to different machine learning algorithms. For more details on how to extract features from event logs please refer to Qafari and van der Aalst (2020).

4 Proposed sampling methods

In this section, we propose an event log preprocessing procedure that helps prediction algorithms to perform faster while maintaining reasonable accuracy. The schematic view of the proposed instance selection approach is presented in Fig. 1. First, we need to traverse the event log and find the variants and corresponding traces of each variant in the event log. Moreover, different distributions of data attributes in each variant will be computed. Afterward, using different sorting and instance selection strategies, we are able to select some of the cases and return the sample event log. In the following, each of these steps is explained in more detail. To illustrate the following steps, we provide an example event log with 10 cases, visible in Table 2.

1. *Traversing the event log*: In this step, the unique variants of the event log and the corresponding traces of each variant are determined. In other words, consider event log EL that $\overline{sl(EL)}=\{\sigma_1, \dots, \sigma_n\}$ where $n=|\overline{sl(EL)}|$, we aim to split EL to $EL_1, \dots, EL_n$ where EL_i only contains all the cases that $C_i=\{c \in C \mid \pi_C(c, variant)=\sigma_i\}$ and $E_i=\{e \in E \mid \pi_E(e, case) \in C_i\}$. Obviously, $\bigcup\limits_{1 \le i \le n} (C_i)=C$ and $\bigcap\limits_{1 \le i \le n} (C_i)=\varnothing$. For the event log that is presented in Table 2, we have $n = 4$ variants and $E_1 = \{c_1, c_3, c_4, c_9, c_{10}\}$, $E_2 = \{c_2, c_5, c_8\}$, $E_3 = \{c6\}$, and $E_1 = \{c_7\}$.
2. *Computing Distribution*: In this step, for each variant of the event log, we compute the distribution of different data attributes $a \in AT$. It would be more practical if the interesting attributes are chosen by an expert. Both event and case attributes can be considered. A simple approach is to compute the frequency of categorical data values. For numerical data attributes, it is possible to consider the average or the median of values for all cases of each variant. In the running example for E_3 and E_4, we only have one case for each variant. However, for E_1, and E_2, the average of *Amount* is 500 and 460, respectively.
3. *Sorting the cases of each variant*: In this step, we aim to sort the traces of each variant. We need to sort the traces to give a higher priority to those traces that can represent the variant better. One way is to sort the traces based on the frequency of the existence of the most occurred data values of the variant. For example, we can give a higher priority to the traces that have more frequent resources of each variant. For the event log that is presented in Table 2, we do not need to prioritize the cases in E_3 and E_4. However, if we sort the traces according to their distance of amount value and the average value of each variant, for E_1, we have c_3, c_9, c_4, c_1, and c_{10}. The order for E_2 is c_5, c_2, and c_8. It is also possible to sort the traces based on their arrival time or randomly.
4. *Returning sample event logs*: Finally, depending on the setting of the sampling function, we return some of the traces with the highest priority for all variants. The most

Table 2 An example event log with 10 traces and 4 variants

CaseID	Variant	Amount
c_1	$\langle a, b, c, d \rangle$	100
c_2	$\langle a, c, b, d \rangle$	720
c_3	$\langle a, b, c, d \rangle$	400
c_4	$\langle a, b, c, d \rangle$	800
c_5	$\langle a, c, b, d \rangle$	600
c_6	$\langle a, c, c, d \rangle$	750
c_7	$\langle a, c, d \rangle$	170
c_8	$\langle a, c, b, d \rangle$	60
c_9	$\langle a, b, c, d \rangle$	260
c_{10}	$\langle a, b, c, d \rangle$	940

Each trace has two attributes that are *Variant* and *Amount* important point about this step is to know how many traces of each variant should be selected.

In the following, some possibilities will be introduced.

- *Unique selection*: In this approach, we select only one trace with the highest priority. In other words, suppose that $L'{=}sl(\delta(EL))$, $\forall_{\sigma \in L'} L'(\sigma){=}1$. Therefore, using this approach we will have $|sl(\delta(EL))|{=}|\overline{sl(EL)}|$. It is expected that by using this approach, the distribution of frequency of variants will be changed and consequently the resulted prediction model will be less accurate. By applying this sampling method on the event log that is presented in Table 2, the sampled event log will have 4 traces, i.e., one trace for each variant. The corresponding cases are $C' = \{c_3, c_5, c_6, c_7\}$. For the variants that have more than one trace, the traces are chosen that have the highest priority (their amount value is closer to the average amount value for each variant).
- *Logarithmic distribution*: In this approach, we reduce the number of traces in each variant in a logarithmic way. If $L{=}sl(EL)$ and $L'{=}sl(\delta(EL))$, $\forall_{\sigma \in L'} L'(\sigma){=}[Log_k(L(\sigma))]$. Using this approach, the infrequent variants will not have any trace in the sampled event log and consequently it is not variant-preserving. According the above formula, by using a higher base for the logarithm (i.e., k), the size of the sampled event log is reduced more. By using this sampling strategy with k equals to 3 on the event log that is presented in Table 2, the cases that selected in the sampled event log is $C' = \{c_3, c_9, c_5\}$. Note that for the infrequent variants no trace is selected in the sampled event log.
- *Division*: This approach performs similar to the previous one, however, instead of using logarithmic scale, we apply the division operator. In this approach, $\forall_{\sigma \in L'} L'(\sigma){=}\lceil \frac{L(\sigma)}{k} \rceil$. A higher k results in fewer cases in the sample event log. Note that as $\lceil\rceil$ considered in the above formula, using this approach all the variants have at least one trace in the sampled event log and it is variant-preserving. By using this sampling strategy with $k = 4$ on the event log that is presented in Table 2, the sampled event log will have 5 traces that are $C' = \{c_3, c_9, c_5, c_6, c_7\}$.

There is also a possibility to consider other selection methods. For example, we can select the traces completely randomly from the original event log.

By choosing different data attributes in Step 2 and different sorting algorithms in Step 3, we are able to lead the sampling of the method on *which* cases should be chosen. Moreover, by choosing the type of distribution in Step 4, we determine *how many* cases should be chosen. To compute how sampling method δ reduces the size of the given event log EL, we use the following equation:

$$R_S{=}\frac{|sl(EL)|}{|sl(\delta(EL))|} \tag{1}$$

The higher R_S value means, the sampling method reduces more the size of the training log. By choosing different distribution methods and different *k-values*, we are able to control the size of the sampled event log. It should be noted that the proposed method will apply just to the training event log. In other words, we do not sample event logs for development and test datasets.

5 Evaluation

In this section, we aim at designing some experiments to answer the research question, i.e., "Is it possible to have computational performance improvement of prediction methods by using the sampled event logs, while maintaining a similar accuracy?". It should be noted that the focus of the experiments is not on prediction model tuning to have higher accuracy. Conversely, we aim to analyze the effect of using sampled event logs (instead of the whole datasets) on the required time and the accuracy of prediction models.

In the following, we first explain the evaluation settings and event logs that are used. Afterward, we provide some information about the implementation of sampling methods, and finally, we show the experimental results.

5.1 Evaluation setting

In this section, we first explain the prediction methods and parameters that are used in the evaluation. Afterward, we discuss the evaluation metrics.

5.1.1 Evaluation parameters

We have developed the sampling methods as a plug-in in the ProM framework (Verbeek et al., 2010), accessible via https://svn.win.tue.nl/repos/prom/Packages/LogFiltering. This plug-in takes an event log and returns k different train and test event logs in the CSV format. Moreover, we have also implemented the sampling methods in Python to have all the evaluations in one workflow.

We have used two machine learning methods to train the prediction models, i.e., *LSTM* and *XGBoost*. For predicting the next activity, our LSTM network consisted of an input layer, two LSTM layers with *dropout* rates of 10%, and a dense output layer with the *SoftMax* activation function. We used "categorical cross-entropy" to calculate the loss and adopted *ADAM* as an optimizer. We built the same architecture of *LSTM* for the remaining time predicting with some differences. We employed "mean absolute error" as a loss function and "Root Mean Squared Propagation" *(RMSprop)* as an optimizer. We used *gbtree* with a max depth of 6 as a booster in our XGBoost model for both of the next activity and remaining time prediction tasks. Uniform distribution is used as the sampling method inside our XGBoost model. To avoid overfitting in both models, the training set is further divided into 90% training set and 10% validation set to stop training once the model performance on the validation set stops improving. We used the same parameter setting of both models for original event logs and sampled event logs. The implementations of these methods are available at https://github.com/gyunamister/pm-prediction/.

To train the prediction models using machine learning methods, we extract features from event data. To this end, we use the most commonly-used features for each prediction task in order to reduce the degree of freedom in selecting relevant features. In other words, we focus on comparing the performance of predictions between sampled and non-sampled event data with a fixed feature space. For instance, for the next activity prediction, we use the partial trace (i.e., the sequence of historical activities) of cases and the temporal measures of each activity (e.g., sojourn time) with one-hot encoding (Park & Song, 2019). For the remaining time prediction, we use the partial trace of cases along with case attributes (e.g., cost), resources, and temporal measures (Verenich et al., 2019).

To sample the event logs, we use three distributions that are *log distribution*, *division*, and *unique variants*. For the *log* distribution method, we have used 2,3,5, and 10 (i.e.,

log_2, log_3, log_5, and log_{10}). For the division method, we have used 2,3,5, and 10 (i.e., $d2$, $d3$, $d5$, and $d10$). For each event log and each sampling method, we have used a 5-$fold$ cross-validation. It means we split the data into 5 groups. One of the groups is used as the test event log, and the rest are merged as the training event log. It should be noted that for each event log, the splitting groups were the same for all the prediction and sampling methods. Moreover, as the results of the experiments are non-deterministic, all the experiments have been repeated 5 times, and the average values are represented. Moreover, to have a fair evaluation, in all the steps, one CPU thread has been used.

5.1.2 Metrics

To evaluate the correctness of prediction methods for predicting the next activities, we have considered two metrics, i.e., *Accuracy* and *F1-score*. The F1-score is used for imbalanced data (Luque et al., 2019). For remaining time prediction, we consider Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) measures as they were used in Verenich et al. (2019) that are computed as follows.

$$MAE=\frac{1}{n}\sum_{t=1}^{n}|e_t| \tag{2}$$

$$RMSE=\sqrt{\frac{1}{n}\sum_{t=1}^{n}e_t^2} \tag{3}$$

In the above equations, e_t indicates the prediction error for the t^{th} instance of validation data. In other words, we aim to compute the absolute difference between the predicted and real values (in days) for the remained time. For both measures, the lower value means higher accuracy. Note that, similar to Verenich et al. (2019), we considered seconds as the time unit to compute these two metrics.

To evaluate how accurate the prediction methods using the sampled event logs are, we have used relative metrics that compared them with the case that whole event logs are used according to the following equations.

$$R_{Acc} = \frac{\text{Accuracy using the sampled training log}}{\text{Accuracy using the whole training log}} \tag{4}$$

$$R_{F1}=\frac{\text{F1-score using the sampled training log}}{\text{F1-score using the whole training log}} \tag{5}$$

In both above equations, a value close to 1 means that using the sampling event logs, the prediction methods behave almost similar to the case that the whole data is used for the training. Moreover, values higher than 1 indicate the accuracy/F1-score of prediction methods has improved.

Unlike previous metrics, for MAE and $RMSE$, a higher value means the prediction model is less accurate in predicting the remaining time. Therefore, we use the following measures.

$$R_{MAE}=\frac{MAE \text{ using the whole training log}}{MAE \text{ using the sampled training log}} \tag{6}$$

$$R_{RMSE}=\frac{RMSE \text{ using the whole training log}}{RMSE \text{ using the sampled training log}} \tag{7}$$

In both of the above measures, a higher value means that the applied instance selection method preserves higher accuracy. In case the values of the above measures are higher than

1, the instance selection methods improve the accuracy of prediction models compared to the case that the whole training data have been used.

To compute the improvement in the performance of feature extraction and training time, we will use the following equations.

$$R_{FE} = \frac{\text{Feature extraction time using whole data}}{\text{Feature extraction time using the sampled data}} \tag{8}$$

$$R_t = \frac{\text{Training time using whole data}}{\text{Training time using the sampled data}} \tag{9}$$

For both equations, the resulting values indicate how many times applying the sampled log is faster than using all data.

5.1.3 Event logs

We have used three event logs widely used in the literature. In the *BPIC-2012-W* event log, relating to a process of an insurance company, the average of variant frequencies is low. In the *RTFM* event log, which corresponds to a road traffic management system, we have some highly frequent variants and several infrequent variants. Moreover, the number of activities in this event log is high. In the *Sepsis* event log, relating to a health care process, there are several variants, that most of them are unique. Some of the activities in the last two event logs are infrequent, which makes these event logs imbalanced. Some information about these event logs and the result of using prediction methods on them is presented in Table 3. Note that the time-related features in this table are in seconds.

According to Table 3, using the whole event data we usually have high accuracy for the next activity prediction. However, the F1-score is not that high, which is mainly because the event logs are imbalanced (specifically *RTFM* and *Sepsis*). Moreover, the MAE and RMSE values are very high. Specifically for the *RTFM* event log. It is mainly because process instances' durations are very long in this event log, and consequently, the e_t values are higher. Finally, there is a direct relation between the size of event logs and the required time for extracting features and training the models.

5.2 Evaluation results

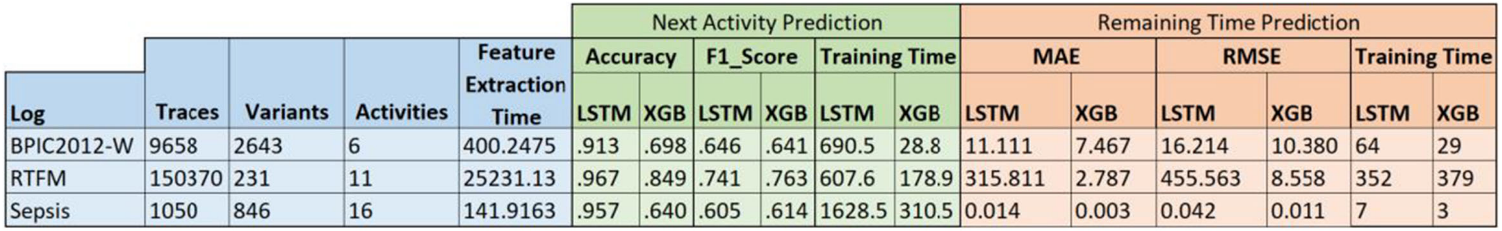
Here, we provide the results of using sampled training event logs instead of whole training event logs. First, we show how by using sampling, the size of training data is reduced in Table 4. As it is expected, the highest reduction occurs when log_{10} is used. Using this sampling, the size of the *RTFM* event log is more than 1000 times reduced. However, for the $Sepsis$ event log, as most variants are unique, sampling the training event logs using divide distribution could not result in high R_S. Moreover, this table shows how using the sampling event logs can reduce the required time to extract features of the event data, i.e., R_{FE}. As it is expected, there is a correlation between the size reduction of the sampled event logs and the improvement in the R_{FE}.

In the following, we show how using the sampling event logs affects the next activity, remaining time, and outcome prediction.

5.2.1 Next Activity Prediction

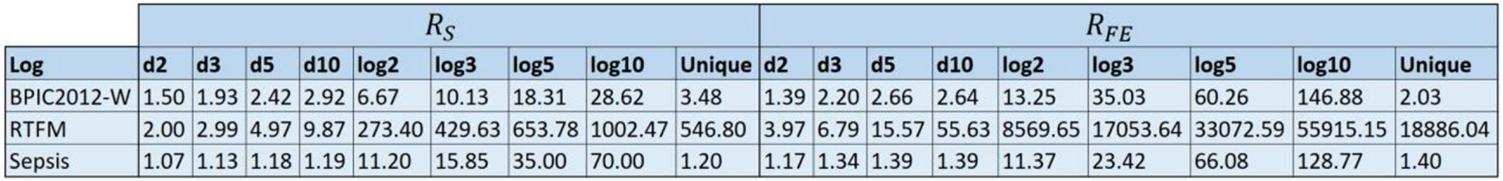
The accuracy of both LSTM and XGboost methods that are trained using sampled training data is presented in Table 5. Results indicate that in most cases, when the division sampling

Table 3 Event logs that are used in the evaluation and results of using them for the next activity and remaining time prediction



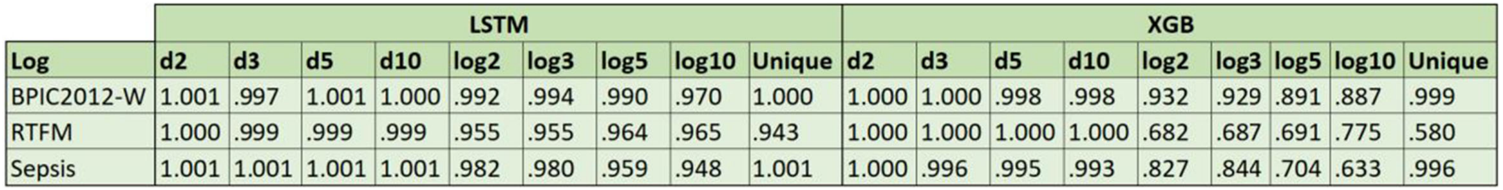
					Next Activity Prediction						Remaining Time Prediction					
					Accuracy		F1_Score		Training Time		MAE		RMSE		Training Time	
Log	Traces	Variants	Activities	Feature Extraction Time	LSTM	XGB	LSTM	XGB	LSTM	XGB	LSTM	XGB	LSTM	XGB	LSTM	XGB
BPIC2012-W	9658	2643	6	400.2475	.913	.698	.646	.641	690.5	28.8	11.111	7.467	16.214	10.380	64	29
RTFM	150370	231	11	25231.13	.967	.849	.741	.763	607.6	178.9	315.811	2.787	455.563	8.558	352	379
Sepsis	1050	846	16	141.9163	.957	.640	.605	.614	1628.5	310.5	0.014	0.003	0.042	0.011	7	3

Table 4 Reduction in size of training event logs (i.e., R_S) and the improvement in the feature extraction process (i.e., R_{FE}) using different sampling methods



	R_S									R_{FE}								
Log	**d2**	**d3**	**d5**	**d10**	**log2**	**log3**	**log5**	**log10**	**Unique**	**d2**	**d3**	**d5**	**d10**	**log2**	**log3**	**log5**	**log10**	**Unique**
BPIC2012-W	1.50	1.93	2.42	2.92	6.67	10.13	18.31	28.62	3.48	1.39	2.20	2.66	2.64	13.25	35.03	60.26	146.88	2.03
RTFM	2.00	2.99	4.97	9.87	273.40	429.63	653.78	1002.47	546.80	3.97	6.79	15.57	55.63	8569.65	17053.64	33072.59	55915.15	18886.04
Sepsis	1.07	1.13	1.18	1.19	11.20	15.85	35.00	70.00	1.20	1.17	1.34	1.39	1.39	11.37	23.42	66.08	128.77	1.40

Table 5 R_{Acc} of different event logs when different sampling methods are used



	LSTM								XGB								
Log	**d2**	**d3**	**d5**	**d10**	**log2**	**log3**	**log5**	**log10**	**Unique**	**d2**	**d3**	**d5**	**d10**	**log2**	**log3**	**log5**	**log10**	**Unique**
BPIC2012-W	1.001	.997	1.001	1.000	.992	.994	.990	.970	1.000	1.000	1.000	.998	.998	.932	.929	.891	.887	.999
RTFM	1.000	.999	.999	.999	.955	.955	.964	.965	.943	1.000	1.000	1.000	1.000	.682	.687	.691	.775	.580
Sepsis	1.001	1.001	1.001	1.001	.982	.980	.959	.948	1.001	1.000	.996	.995	.993	.827	.844	.704	.633	.996

methods are used, we can achieve similar accuracy, i.e., R_{Acc} close to 1, compared to the case where the whole training data is used. In some cases, like using $d2$ for the *Sepsis* event log and LSTM method, the accuracy of the training method is even (slightly) improved. However, for the *RTFM*, using sampling with logarithmic or unique distribution highly changes the frequency distribution of variants and consequently causes a higher reduction in the accuracy of predicting models. Moreover, the accuracy reduction was higher when the XGboost method was used. The result indicated that by increasing the R_S value, we lose more information in the training event logs, and consequently, the quality of the prediction models will be decreased.

In Table 6, R_{F1} of trained models are depicted. The results again indicate that using the sampling method with divide distribution in most cases leads to having a similar (and sometimes higher) F1-score.

By considering the results of these two tables, we found that specifically, when the LSTM method is used, we will have similar accuracy and F1-score. Moreover, for the *Sepsis* and *BPIC2012-W* that variants have similar frequency having all the variants (i.e., using divide and unique distributions) can help the prediction method to have results similar to the case that we use the whole training data. However, for the *RTFM* event log that has some high frequent variants, using the unique distribution results in lower accuracy and F1-score.

Table 7 shows how much training time is faster using the sampled training data instead of using the whole event data. There is a direct relationship between the size reduction and R_t(refer to the results in Table 4). However, in most cases, the performance improvement is bigger for the XGboost method. Considering the results in this table and Table 6, we found that using the sampling method, we are able to improve the performance of the next activity prediction methods on the used event logs while they provide similar results. However, oversampling (e.g., applying $log10$ for *RTFM*) will result in lower accuracy.

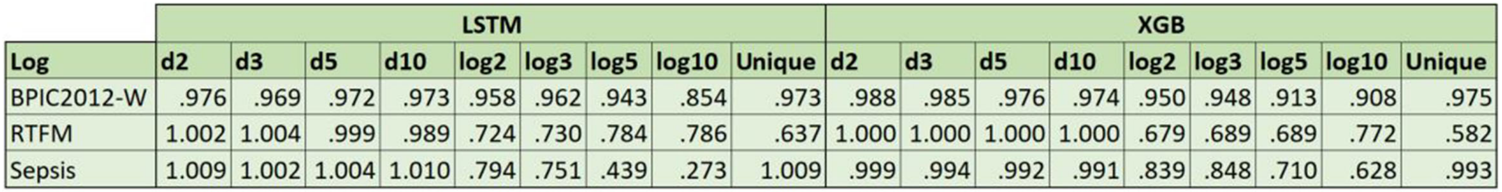
5.2.2 Remaining time prediction

In Tables 8 and 9, we show how by using the sampled event logs, MAE and RMSE of different remaining time prediction methods are changed. The results indicate that for the LSTM method, independent of the sampling method for all event logs, we are able to provide a prediction similar to the case where whole event logs are used. It seems that the settings that are used for training the prediction method are not good. In other words, the trained model is not accurate enough. We repeat this experiment for LSTM with several different parameters, but we have almost the same results. It is mainly caused by the challenging nature of the remaining time prediction task compared to classification-based problems (such as next activity and outcome prediction). However, by sampling the training event logs, we keep the quality of prediction models.

For the XGboost method, the results indicate that if we do not sample a small amount of traces (for example, using logarithmic sampling), we can have high R_{MAE} and R_{RMSE}. In general, as the main attribute for the next activity prediction is the sequence of activities, it is less sensitive to sampling. However, to predict the remaining time, the other data attributes can be essential too. In other words, for the remaining prediction, we need larger sampled event logs.

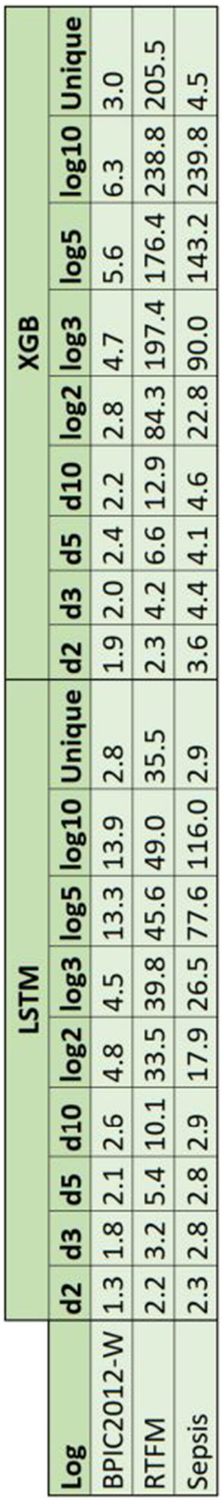
In Table 10, it is shown how by sampling event logs, we are able to reduce the required training time and improve the performance of the remaining time prediction process. By considering the results in Tables 7 and 10, as we have expected, by having higher R_S the R_t value is higher.

Table 6 R_{F1} of different event logs when different sampling methods are used



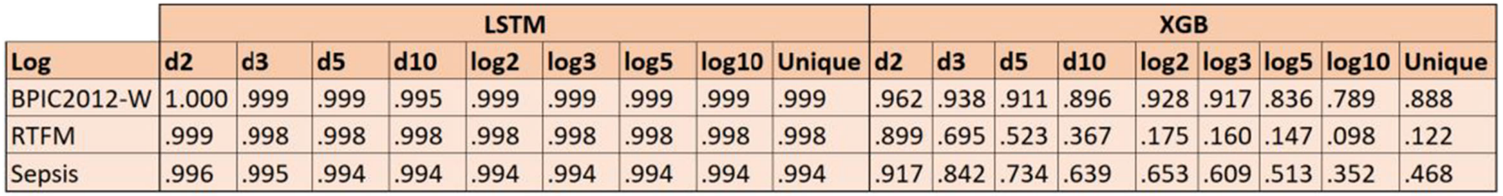
	LSTM									XGB								
Log	**d2**	**d3**	**d5**	**d10**	**log2**	**log3**	**log5**	**log10**	**Unique**	**d2**	**d3**	**d5**	**d10**	**log2**	**log3**	**log5**	**log10**	**Unique**
BPIC2012-W	.976	.969	.972	.973	.958	.962	.943	.854	.973	.988	.985	.976	.974	.950	.948	.913	.908	.975
RTFM	1.002	1.004	.999	.989	.724	.730	.784	.786	.637	1.000	1.000	1.000	1.000	.679	.689	.689	.772	.582
Sepsis	1.009	1.002	1.004	1.010	.794	.751	.439	.273	1.009	.999	.994	.992	.991	.839	.848	.710	.628	.993

Table 7 R_t of different event logs when different sampling methods are used



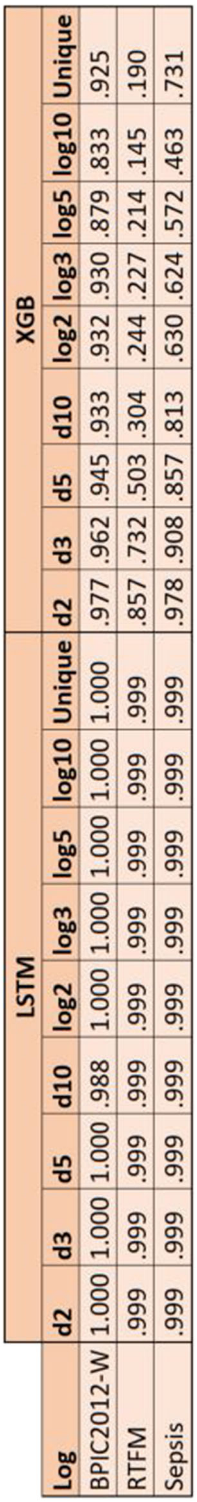
	LSTM									XGB								
Log	**d2**	**d3**	**d5**	**d10**	**log2**	**log3**	**log5**	**log10**	**Unique**	**d2**	**d3**	**d5**	**d10**	**log2**	**log3**	**log5**	**log10**	**Unique**
BPIC2012-W	1.3	1.8	2.1	2.6	4.8	4.5	13.3	13.9	2.8	1.9	2.0	2.4	2.2	2.8	4.7	5.6	6.3	3.0
RTFM	2.2	3.2	5.4	10.1	33.5	39.8	45.6	49.0	35.5	2.3	4.2	6.6	12.9	84.3	197.4	176.4	238.8	205.5
Sepsis	2.3	2.8	2.8	2.9	17.9	26.5	77.6	116.0	2.9	3.6	4.4	4.1	4.6	22.8	90.0	143.2	239.8	4.5

Table 8 R_{MAE} of different event logs when different sampling methods are used



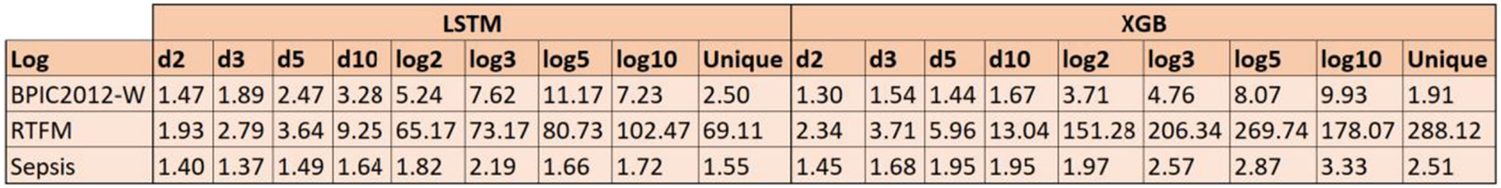
	LSTM									XGB								
Log	**d2**	**d3**	**d5**	**d10**	**log2**	**log3**	**log5**	**log10**	**Unique**	**d2**	**d3**	**d5**	**d10**	**log2**	**log3**	**log5**	**log10**	**Unique**
BPIC2012-W	1.000	.999	.999	.995	.999	.999	.999	.999	.999	.962	.938	.911	.896	.928	.917	.836	.789	.888
RTFM	.999	.998	.998	.998	.998	.998	.998	.998	.998	.899	.695	.523	.367	.175	.160	.147	.098	.122
Sepsis	.996	.995	.994	.994	.994	.994	.994	.994	.994	.917	.842	.734	.639	.653	.609	.513	.352	.468

Table 9 R_{RMSE} of different event logs when different sampling methods are used



	LSTM									XGB								
Log	d2	d3	d5	d10	log2	log3	log5	log10	Unique	d2	d3	d5	d10	log2	log3	log5	log10	Unique
BPIC2012-W	1.000	1.000	1.000	.988	1.000	1.000	1.000	1.000	1.000	.977	.962	.945	.933	.932	.930	.879	.833	.925
RTFM	.999	.999	.999	.999	.999	.999	.999	.999	.999	.857	.732	.503	.304	.244	.227	.214	.145	.190
Sepsis	.999	.999	.999	.999	.999	.999	.999	.999	.999	.978	.908	.857	.813	.630	.624	.572	.463	.731

Table 10 R_t of different event logs when different sampling methods are used



	LSTM									XGB								
Log	**d2**	**d3**	**d5**	**d10**	**log2**	**log3**	**log5**	**log10**	**Unique**	**d2**	**d3**	**d5**	**d10**	**log2**	**log3**	**log5**	**log10**	**Unique**
BPIC2012-W	1.47	1.89	2.47	3.28	5.24	7.62	11.17	7.23	2.50	1.30	1.54	1.44	1.67	3.71	4.76	8.07	9.93	1.91
RTFM	1.93	2.79	3.64	9.25	65.17	73.17	80.73	102.47	69.11	2.34	3.71	5.96	13.04	151.28	206.34	269.74	178.07	288.12
Sepsis	1.40	1.37	1.49	1.64	1.82	2.19	1.66	1.72	1.55	1.45	1.68	1.95	1.95	1.97	2.57	2.87	3.33	2.51

5.3 Outcome prediction

For the outcome prediction, in order to facilitate comparison and remain consistent with previous work on outcome prediction, we transform each event log into different event logs (Teinemaa et al., 2019a). For example, we transform $BPIC\_2012$ event log to $BPIC\_2012\_Accepted$, $BPIC\_2012\_Cancelled$, and $BPIC\_2012\_Declined$.

The R_{Acc} and R_{F1} of both LSTM and XGboost methods that are trained for prediction of outcome using sampled training data is presented in Tables 11 and 12, respectively. The results indicate that, in many cases, we are able to improve the accuracy of the outcome prediction algorithms. Specifically, using *Unique* strategy for sampling the traces for $BPIC\_2012\_Accepted$ event log leads to considerable improvement for both $LSTM$ and XGB methods. Unlike the other two applications, i.e., the next activity and the remaining time prediction, even by oversampling some event logs, e.g., $log10$, we can obtain results similar to the cases in which the whole training event logs are used.

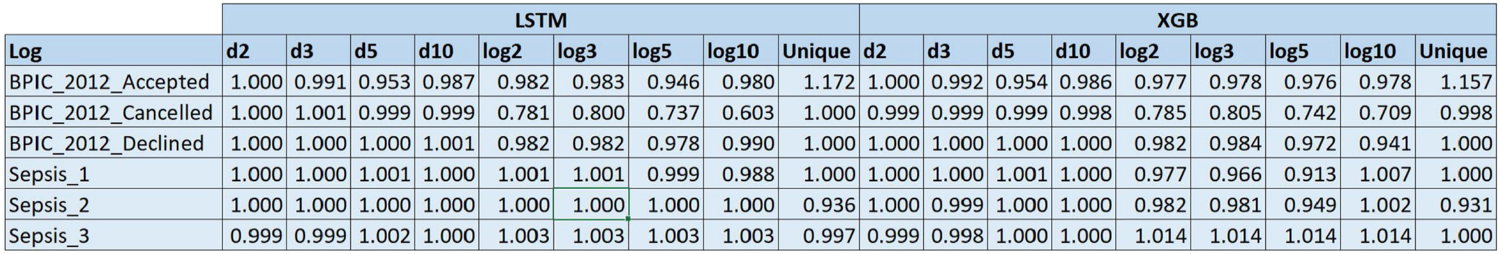
In Table 13, R_t of different sampling methods are shown. The performance improvement is usually bigger for the *LSTM* method. There are several cases in which we are not able to improve the performance of the prediction method (R_t values less than 1). It happens mainly for the *Unique* sampling method. One reason could be by removing the frequencies, the convergence time for the learning method is increased. In case the logarithmic method is used, the performance improvement is around 50 times. It means the training process using the sampled event log is 50 times faster than the case where the whole training log is used.

6 Discussion

In this section, we discuss the results that are illustrated in the previous section. The results indicate that we do not always have a typical trade-off between the accuracy of the trained model and the performance of the prediction procedure. For example, for the next activity prediction, there are some cases where the training process is much faster than the normal procedure, even though the trained model provides an almost similar or higher accuracy and F1-score. Thus, the proposed instance selection procedure can be applied when we aim to apply hyper-parameter optimization (Bergstra et al., 2011). In this way, more settings can be analyzed in a limited time. Moreover, it is reasonable to use the proposed method when we aim to train an online prediction method or on naive hardware such as cell phones.

To achieve the highest performance improvement while the trained model is accurate enough, different sampling methods should be used for different event logs. For example, for the *RTFM* event log—as there are some highly frequent variants—the division distribution may be more useful. In other words, independently of the used prediction method, if we change the distribution of variants (e.g., using $unique$ distribution), it is expected that the accuracy will sharply decrease. However, for event logs with a more uniform distribution, we can use $unique$ distributions to sample event logs. Furthermore, the results indicate that the effect of the chosen distribution (i.e., $unique$, $division$, and $logarithmic$) is more important than the used *k-value*. It is mainly because the $logarithmic$ distribution may remove some of the variants, and the $unique$ distribution change the frequency distribution of variants. Therefore, it would be interesting to investigate more on the characteristics of the given event log and suitable sampling parameters for such distribution. For example, if most variants of a given event log are unique (e.g., $Sepsis$), using the $logarithmic$ distribution leads to having remarkable R_S and consequently, R_{FE} and R_t will be very high. However, we will lose most of the variants, and the trained model might have poor predictions.

Table 11 R_{Acc} of different event logs when different sampling methods are used for outcome prediction



	LSTM									XGB								
Log	**d2**	**d3**	**d5**	**d10**	**log2**	**log3**	**log5**	**log10**	**Unique**	**d2**	**d3**	**d5**	**d10**	**log2**	**log3**	**log5**	**log10**	**Unique**
BPIC_2012_Accepted	1.000	0.991	0.953	0.987	0.982	0.983	0.946	0.980	1.172	1.000	0.992	0.954	0.986	0.977	0.978	0.976	0.978	1.157
BPIC_2012_Cancelled	1.000	1.001	0.999	0.999	0.781	0.800	0.737	0.603	1.000	0.999	0.999	0.999	0.998	0.785	0.805	0.742	0.709	0.998
BPIC_2012_Declined	1.000	1.000	1.000	1.001	0.982	0.982	0.978	0.990	1.000	1.000	1.000	1.000	1.000	0.982	0.984	0.972	0.941	1.000
Sepsis_1	1.000	1.000	1.001	1.000	1.001	1.001	0.999	0.988	1.000	1.000	1.000	1.001	1.000	0.977	0.966	0.913	1.007	1.000
Sepsis_2	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.936	1.000	0.999	1.000	1.000	0.982	0.981	0.949	1.002	0.931
Sepsis_3	0.999	0.999	1.002	1.000	1.003	1.003	1.003	1.003	0.997	0.999	0.998	1.000	1.000	1.014	1.014	1.014	1.014	1.000

Table 12 R_{F1} of different event logs when different sampling methods are used for outcome prediction

	LSTM									XGB								
Log	**d2**	**d3**	**d5**	**d10**	**log2**	**log3**	**log5**	**log10**	**Unique**	**d2**	**d3**	**d5**	**d10**	**log2**	**log3**	**log5**	**log10**	**Unique**
BPIC_2012_Accepted	0.970	1.163	1.071	1.161	1.008	1.078	1.047	1.128	1.459	0.995	1.183	1.101	1.195	1.184	1.186	1.171	1.184	1.463
BPIC_2012_Cancelled	0.983	0.989	0.981	0.980	0.845	0.875	0.772	0.549	0.977	0.990	0.984	0.982	0.981	0.860	0.891	0.791	0.747	0.981
BPIC_2012_Declined	1.000	1.000	1.000	0.999	1.002	1.000	0.987	0.979	1.000	1.000	1.000	1.000	1.000	1.003	1.003	1.002	0.985	1.000
Sepsis_1	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.998	1.000	1.000	0.999	0.999	1.000	0.996	0.995	0.953	0.996	1.000
Sepsis_2	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.903	1.000	0.999	1.000	1.000	0.995	0.995	0.980	1.000	0.907
Sepsis_3	1.004	1.006	1.005	1.005	0.998	0.998	0.998	0.998	1.003	1.000	0.998	0.999	1.000	0.994	0.994	0.994	0.994	1.000

Table 13 R_t of different event logs when different sampling methods are used for outcome prediction

	LSTM									XGB								
Log	**d2**	**d3**	**d5**	**d10**	**log2**	**log3**	**log5**	**log10**	**Unique**	**d2**	**d3**	**d5**	**d10**	**log2**	**log3**	**log5**	**log10**	**Unique**
BPIC_2012_Accepted	2.46	3.43	3.74	5.56	26.67	41.28	44.16	62.74	0.66	1.02	2.48	1.12	6.50	18.11	11.44	20.68	19.22	0.89
BPIC_2012_Cancelled	1.56	1.48	1.43	1.38	6.92	8.97	31.03	49.92	1.37	0.85	1.17	0.67	0.83	2.69	5.38	7.34	25.49	0.94
BPIC_2012_Declined	1.40	1.14	1.12	1.33	4.61	6.18	19.07	46.03	0.93	1.23	1.20	0.59	1.24	1.89	4.96	6.95	10.22	0.95
Sepsis_1	1.06	1.11	1.18	1.00	5.40	6.20	11.35	11.03	1.05	1.55	1.27	1.46	1.49	7.25	15.40	16.88	14.90	1.35
Sepsis_2	1.53	1.62	1.56	1.63	3.70	4.34	8.46	9.86	0.79	3.89	3.91	1.46	4.00	17.58	9.73	16.84	35.49	1.67
Sepsis_3	1.04	1.07	1.19	1.15	6.40	7.54	7.95	11.31	1.25	2.05	2.18	2.58	1.55	16.78	18.36	28.24	26.09	0.82

By analyzing the results, we found that the infrequent activities can be ignored using some hyper-parameter settings. The significant difference between F1-score and Accuracy values in Table 3 indicates this problem too. Using the sampling methods that modify the distribution of the event logs such as the *unique* method can help the prediction methods to also consider these activities. However, as these activities are infrequent, improving in the prediction of them would not impact highly on the presented aggregated F1-score value.

Finally, in real-life business scenarios, the process can change because of different reasons (Carmona & Gavaldà, 2012). This phenomenon is usually called *concept drift*. By considering the whole event log for training the prediction model, it is most probable that these changes are not considered in the prediction. Using the proposed sampling procedure, and giving higher priorities to newer traces, it is expected that we are able to adapt to the changes faster, which may be critical for specific applications.

6.1 Limitations

Comparing the results for the next activity and remaining time prediction, we found that predicting the remaining time of the process is more sensitive to sampling. In other words, this application requires more data to predict accurately. Therefore, for the cases that the target attribute depends more on other data attributes compared to variant information, we need to sample more data. Otherwise, the trained model might be inaccurate.

We also found a critical problem in predictive monitoring. In some cases, specifically using LSTM for predicting the remaining time, the accuracy of the predictions is low. For the next activity prediction, it is possible that the prediction models almost ignore infrequent activities. In these cases, even if we use the training data for the evaluation, we do not have acceptable results. This problem in machine learning is called a high bias error (van der Putten & van Someren, 2004). In other words, the training is not efficient even when using whole data and we need to change the prediction method (or its parameters).

7 Conclusion

In this paper, we proposed an instance selection approach to improve the performance of predictive business process monitoring methods. We suggested that it is possible to use a sample of training event data instead of the whole training event data. To evaluate the proposed approach, we consider two main applications of predictive business monitoring, i.e., the next activity and the remaining time prediction. Results of applying the proposed approaches on three real-life event logs and two widely used machine learning methods, i.e., LSTM and XGboost, indicate that in most cases, we are able to improve the performance of predictive monitoring algorithms while providing similar accuracy compared to the case that the whole training event logs are used. However, by oversampling, the accuracy of the trained model might be reduced. Moreover, we have found that the remaining time prediction application is more sensitive to sampling.

To continue this research, we aim to extend the experiments to study the relationship between the event log characteristics and the sampling parameters. In other words, we aim to help the end-user to adjust the sampling parameters based on the characteristics of the given event log. Moreover, it would be great to investigate how we can apply the proposed sampling procedure for streaming event data which is potentially one of the major advantages of the proposed method in a real-life setting. Finally, it is interesting to investigate more on feature selection methods for improving the performance of the predictive monitoring procedure. It is expected that, similar to process instance sampling, feature selection methods are able to reduce the required training time. In other words, the training is not efficient even using whole data and we need to change the prediction method (or its parameters).

Another important outcome of the results is that for different event logs, we should use different sampling methods to achieve the highest performance. Considering lots of different machine learning methods and their parameters can lead to an increase in the search space and complexity for users. In other words, finding the right setting for sampling may be challenging in real scenarios. Therefore, it would be valuable to research the relationship between the event log characteristics and suitable sampling parameters that can be used for preprocessing the training event log.

Acknowledgment The authors would like to thank the Alexander von Humboldt (AvH) Stiftung for funding this research.

Funding Open Access funding enabled and organized by Projekt DEAL.

Code availability Our proposed are available in https://svn.win.tue.nl/repos/prom/Packages/LogFiltering and https://github.com/gyunamister/pm-prediction/. For a part of the experiments, we have used the implementation that is available at https://github.com/verenich/time-prediction-benchmark.

Data availability We have applied our proposed approach to the following three publicly available datasets (event logs).

- *BPIC-2012*, that is accessible via https://data.4tu.nl/articles/dataset/BPI_Challenge_2012/12689204.
- *RTFM*, that is accessible via https://data.4tu.nl/articles/dataset/Road_Traffic_Fine_Management_Process/12683249.
- *Sepsis* that is accessible via https://data.4tu.nl/articles/dataset/Sepsis_Cases_-_Event_Log/12707639.

Declarations

Conflict of interest In this part, we provide some declarations about the conflict of interest, the code availability, and the availability of data that is used in this paper.

References

Bergstra, J., Bardenet, R., Bengio, Y., & Kégl, B. (2011). Algorithms for hyper-parameter optimization. In *advances in neural information processing systems 24 25th annual conference on neural information processing systems 2011. Proceedings of a meeting held 12–14 December 2011, Granada, Spain,* pp. 2546–2554.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24*(2), 123–140.

Breuker, D., Matzner, M., Delfmann, P., & Becker, J (2016). Comprehensible predictive models for business processes. *Mis Quarterly*, *40*(4), 1009–1034. JSTOR.

Carmona, J., & Gavaldà, R. (2012). Online techniques for dealing with concept drift in process mining. In *advances in intelligent data analysis XI - 11th international symposium, IDA 2012, Helsinki, Finland, October 25–27, 2012. Proceedings, vol. 7619,* (pp. 90–102). https://doi.org/10.1007/978-3-642-34156-4_10.

Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, San Francisco, CA, USA, August 13-17, 2016,* (pp. 785–794). https://doi.org/10.1145/2939672.2939785.

de Leoni, M., van der Aalst, W. M. P., & Dees, M. (2016). A general process mining framework for correlating, predicting and clustering dynamic behavior based on event logs. vol. 56, pp. 235–257. https://doi.org/10.1016/j.is.2015.07.003.

Evermann, J., Rehse, J., & Fettke, P. (2017). Predicting process behaviour using deep learning. *Decision Support Systems*, *100*, 129–140. https://doi.org/10.1016/j.dss.2017.04.003.

Fani Sani, M., van Zelst, S. J., & van der Aalst, W. M. P. (2020). Conformance checking approximation using subset selection and edit distance. In *advanced information systems engineering - 32nd international conference, CAiSE 2020, Grenoble, France, June 8–12, 2020, proceedings, vol. 12127,* (pp. 234–251). https://doi.org/10.1007/978-3-030-49435-3_15.

Fani Sani, M., van Zelst, S. J., & van der Aalst, W. M. P. (2021). The impact of biased sampling of event logs on the performance of process discovery. *Computing*, *103*(6), 1085–1104. https://doi.org/10.1007/s00607-021-00910-4.

Galanti, R., Coma-Puig, B., de Leoni, M., Carmona, J., & Navarin, N. (2020). Explainable predictive process monitoring. In *2nd international conference on process mining, ICPM 2020, Padua, Italy, October 4–9, 2020,* (pp. 1–8). https://doi.org/10.1109/ICPM49681.2020.00012.

Garca, S., Luengo, J., & Herrera, F. (2014). *Data preprocessing in data mining*. Incorporated: Springer Publishing Company.

Huang, Z., Xu, W., & Yu, K. (2015). Bidirectional LSTM-CRF models for sequence tagging. arXiv: 1508.01991.

Luque, A., Carrasco, A., Martín, A., & de las Heras, A. (2019). The impact of class imbalance in classification performance metrics based on the binary confusion matrix. *Pattern Recognition*, *91*, 216–231. https://doi.org/10.1016/j.patcog.2019.02.023.

Marquez-Chamorro, A. E., Resinas, M., & Ruiz-Cortes, A. (2018). Predictive monitoring of business processes. *A Survey*, *11*(6), 962–977. https://doi.org/10.1109/TSC.2017.2772256. Accessed 2021-02-14.

Navarin, N., Vincenzi, B., Polato, M., Sperduti, A., & LSTM networks for data-aware remaining time prediction of business process instances (2017). In *2017 IEEE symposium series on computational intelligence, SSCI 2017, Honolulu, HI, USA, November 27 – Dec. 1, 2017,* (pp. 1–7). https://doi.org/10.1109/SSCI.2017.8285184.

Nguyen, A., Chatterjee, S., Weinzierl, S., Schwinn, L., Matzner, M., & Eskofier, B. M. (2020). Time matters: Time-aware lstms for predictive business process monitoring. In S. J. J. Leemans, & H. Leopold (Eds.) *Process Mining Workshops - ICPM 2020 international workshops, Padua, Italy, October 5-8, 2020, revised selected papers, vol. 406,* (pp. 112–123). https://doi.org/10.1007/978-3-030-72693-5_9.

Park, G., Küsters, A., Tews, M., Pitsch, C., Schneider, J., & van der Aalst, W. M. P. (2022). Explainable predictive decision mining for operational support. arXiv: 2210.16786, https://doi.org/10.48550/arXiv.2210.16786.

Park, G., & Song, M. (2019). Prediction-based resource allocation using LSTM and minimum cost and maximum flow algorithm. In *international conference on process mining, ICPM 2019, Aachen, Germany, June 24–26, 2019,* (pp. 121–128). https://doi.org/10.1109/ICPM.2019.00027.

Park, G., & Song, M. (2020). Predicting performances in business processes using deep neural networks, Decision Support Systems. 129. https://doi.org/10.1016/j.dss.2019.113191.

Park, G., & van der Aalst, W. M. P. (2022). Action-oriented process mining: bridging the gap between insights and actions. Progress in Artificial Intelligence. https://doi.org/10.1007/s13748-022-00281-7.

Pauwels, S., & Calders, T. (2021). Incremental predictive process monitoring: The next activity case. In *business process management - 19th international conference, BPM 2021, Rome, Italy, September 06–10, 2021, proceedings. Lecture notes in computer science, vol. 12875,* (pp. 123–140). https://doi.org/10.1007/978-3-030-85469-0_10.

Pegoraro, M., Uysal, M. S., Georgi, D. B., & van der Aalst, W.M.P. (2021). Text-aware predictive monitoring of business processes. In *24th international conference on business information systems, BIS 2021, Hannover, Germany, June 15–17, 2021,* (pp. 221–232). https://doi.org/10.52825/bis.v1i.62.

Pegoraro, M., Uysal, M. S., Hülsmann, T., & van der Aalst, W. M. P. (2022). Uncertain case identifiers in process mining: A user study of the event-case correlation problem on click data. In *Enterprise,* *business-process and information systems modeling - 23rd international conference, BPMDS 2022 and 27th international conference, EMMSAD 2022, Held at CAiSE 2022, Leuven, Belgium, June 6–7, 2022, proceedings. Lecture notes in business information processing, vol. 450,* (pp. 173–187). https://doi.org/10.1007/978-3-031-07475-2_12.

Pegoraro, M., Uysal, M. S., Hülsmann, T., & van der Aalst, W. M. P. (2022). Resolving uncertain case identifiers in interaction logs: A user study. arXiv: 2212.00009, https://doi.org/10.48550/arXiv.2212.00009.

Polato, M., Sperduti, A., Burattin, A., & de Leoni, M. (2018). Time and activity sequence prediction of business process instances. *Computing*, *100*(9), 1005–1031. https://doi.org/10.1007/s00607-018-0593-x.

Pourghassemi, B., Zhang, C., Lee, J. H., & Chandramowlishwaran, A. (2020). On the limits of parallelizing convolutional neural networks on gpus. In *SPAA '20: 32nd ACM symposium on parallelism in algorithms and architectures, virtual event, USA, July 15-17, 2020,* (pp. 567–569). https://doi.org/10.1145/3350755.3400266.

Qafari, M. S., & van der Aalst, W. M. P. (2020). Root cause analysis in process mining using structural equation models. In *business process management workshops - BPM 2020 International workshops, Seville, Spain, September 13-18, 2020, revised selected papers, vol. 397,* (pp. 155–167). https://doi.org/10.1007/978-3-030-66498-5_12.

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,* (pp. 389–403). https://doi.org/10.1007/978-3-642-45005-1_27.

Sani, M. F., Vazifehdoostirani, M., Park, G., Pegoraro, M., van Zelst, S. J., & van der Aalst, W. M. P. (2021). Event log sampling for predictive monitoring. In *Process Mining Workshops - ICPM 2021 international workshops, Eindhoven, the Netherlands, October 31 - November 4, 2021, revised selected papers. Lecture notes in business information processing, vol. 433,* (pp. 154–166). https://doi.org/10.1007/978-3-030-98581-3_12.

Senderovich, A., Di Francescomarino, C., Ghidini, C., Jorbina, K., & Maggi, F. M. (2017). Intra and inter-case features in predictive process monitoring: A tale of two dimensions. In *international conference on business process management,* (pp. 306–323). Springer.

Sindhgatta, R., Moreira, C., Ouyang, C., & Barros, A. (2020). Exploring interpretable predictive models for business processes. In *Business process management - 18th international conference, BPM 2020, Seville, Spain, September 13-18, 2020, proceedings. Lecture notes in computer science, vol. 12168,* (pp. 257–272). https://doi.org/10.1007/978-3-030-58666-9_15.

Stierle, M., Brunk, J., Weinzierl, S., Zilker, S., Matzner, M., & Becker, J. (2021). Bringing light into the darkness - A systematic literature review on explainable predictive business process monitoring techniques. In *28th European conference on information systems - liberty, equality, and fraternity in a digitizing world, ECIS 2020, Marrakech, Morocco, June 15–17, 2020.* https://aisel.aisnet.org/ecis2021_rip/8.

Tax, N., Verenich, I., Rosa, M. L., & Dumas, M. (2017). Predictive business process monitoring with LSTM neural networks. In *Dubois, E., Pohl, K. (eds.) Advanced information systems engineering - 29th international conference, CAiSE 2017, Essen, Germany, June 12-16, 2017, Proceedings, vol. 10253,* (pp. 477–492). https://doi.org/10.1007/978-3-319-59536-8_30.

Teinemaa, I., Dumas, M., Maggi, F. M., & Di Francescomarino, C. (2016). Predictive business process monitoring with structured and unstructured data. In *international conference on business process management,* (pp. 401–417). Springer.

Teinemaa, I., Dumas, M., Rosa, M. L., & Maggi, F. M. (2019). Outcome-oriented predictive process monitoring, Review and benchmark. *ACM Transactions on Knowledge Discovery from Data*, *13*(2), 17–11757. https://doi.org/101145/3301300.

Teinemaa, I., Dumas, M., Rosa, M. L., & Maggi, F. M. (2019). Outcome-oriented predictive process monitoring: Review and benchmark. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, *13*(2), 1–57.

van der Aa, H., Rebmann, A., & Leopold, H. (2021). Natural language-based detection of semantic execution anomalies in event logs. *Information Systems*, *102*, 101824. https://doi.org/10.1016/j.is.2021.101824.

van der Aalst, W. M. P. (2016). Process Mining - Data Science in Action, Second Edition. Springer https://doi.org/10.1007/978-3-662-49851-4.

van der Aalst, W. M. P., Schonenberg, M. H., & Song, M. (2011). Time prediction based on process mining, vol. 36,(2), pp. 450–475. https://doi.org/10.1016/j.is.2010.09.001. Accessed 2021-01-06.

van der Putten, P., & van Someren, M. (2004). A bias-variance analysis of a real world learning problem: The coil challenge 2000. *Machine Learning*, *57*(1–2), 177–195. https://doi.org/10.1023/B:MACH.0000035476.95130.99.

Verbeek, E., Buijs, J. C. A. M., van Dongen, B. F., van der Aalst, W. M. P., & Prom 6: The process mining toolkit (2010). In *Proceedings of the business process management 2010 demonstration track, Hoboken, NJ, USA, September 14-16, 2010, vol.615*. http://ceur-ws.org/Vol-615/paper13.pdf.

Verenich, I. (2019). Explainable predictive monitoring of temporal measures of business processes. In *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. EUR workshop proceedings, vol. 2420,* (pp. 26–30). http://ceur-ws.org/Vol-2420/paperDA6.pdf.

Verenich, I., Dumas, M., Rosa, M. L., Maggi, F. M., & Teinemaa, I. (2019). Survey and cross-benchmark comparison of remaining time prediction methods in business process monitoring. *ACM Transactions on Intelligent Systems and Technology (TIST)*, *10*(4), 1–34.

Wang, T., Zhu, J.-Y., Torralba, A., & Efros, A. A. (2020). Dataset distillation. arXiv: 1811.10959 [cs.LG].

Wilson, D. L. (1972). Asymptotic properties of nearest neighbor rules using edited data. *Systems, Man and Cybernetics, IEEE Transactions on*, *2*(3), 408–421. https://doi.org/10.1109/TSMC.1972.4309137.

Wilson, D. R., & Martinez, T. R. (2000). Reduction techniques for instance-basedlearning algorithms. *Machine Learning*, *38*(3), 257–286. https://doi.org/10.1023/A:1007626913721.

Zhou, L., Pan, S., Wang, J., & Vasilakos, A. V. (2017). Machine learning on big data: Opportunities and challenges. *Neurocomputing*, *237*, 350–361. https://doi.org/10.1016/j.neucom.2017.01.026.