

# Event Log Sampling for Predictive Monitoring

Mohammadreza Fani Sani<sup>1</sup>, Mozhgan Vazifehdoostirani<sup>2</sup>, Gyunam Park<sup>1</sup>,  
Marco Pegoraro<sup>1</sup>, Sebastiaan J. van Zelst<sup>3,1</sup>, and Wil M.P. van der Aalst<sup>1,3</sup>

<sup>1</sup>Process and Data Science Chair, RWTH Aachen University, Aachen, Germany

<sup>2</sup>Industrial Engineering and Innovation Science, Eindhoven University of Technology,  
Eindhoven, the Netherlands

<sup>3</sup>Fraunhofer FIT, Birlinghoven Castle, Sankt Augustin, Germany

{fanisani, gnpark, pegoraro, s.j.v.zelst,  
wvdaalst}@pads.rwth-aachen.de  
m.vazifehdoostirani@tue.nl

**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, state-of-the-art methods for predictive monitoring require the training of complex machine learning models, which is often inefficient. This paper proposes an instance selection procedure that allows sampling training process instances for prediction models. We show that our sampling method allows for a significant increase of training speed for next activity prediction methods while maintaining reliable levels of prediction accuracy.

**Keywords:** Process Mining · Predictive Monitoring · Sampling · Machine Learning · Deep Learning · Instance Selection

## 1 Introduction

As the environment surrounding business processes becomes more dynamic and competitive, it becomes imperative to predict process behaviors and take proactive actions [1]. Predictive business process monitoring aims at predicting the behavior of business processes, to mitigate the risk resulting from undesired behaviors in the process. For instance, by predicting the next activities in the process, one can foresee the undesired execution of activities, thus preventing possible risks resulting from it [12]. Moreover, by predicting an expected high service time for an activity, one may bypass or add more resources for the activity [15]. Recent breakthroughs in machine learning have enabled the development of effective techniques for predictive business process monitoring. Specifically, techniques based on deep neural networks, e.g., Long-Short Term Memory (LSTM) networks, have shown high performance in different tasks [8]. Additionally, the emergence of ensemble learning methods leads to improvement in accuracy in different areas [4]. Particularly, for predictive process monitoring, eXtreme Gradient Boosting (XGBoost) [6] has shown promising results, often outperforming other ensemble methods such as Random Forest or using a single regression tree [25, 28].

Indeed, machine learning algorithms suffer from the expensive computational costs in their training process [34]. In particular, machine learning algorithms based on neural

networks and ensemble learning might require tuning their hyperparameters to be able to provide acceptable accuracy. Such long training time limits the application of the techniques considering the limitations in time and hardware [21]. This is particularly relevant for predictive business process monitoring techniques. Business analysts need to test the efficiency and reliability of their conclusions via repeated training of different prediction models with different parameters [15]. Moreover, the dynamic nature of business processes requires new models adapting to new situations in short intervals.

Instance selection aims at reducing 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 [11]. 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, Incremental Reduction Optimization Procedure (DROP) [32] 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) [33] preserves instances by repeatedly discarding an instance if it does not belong to the class of the majority of its neighbors.

Such techniques assume independence among instances [32]. However, in predictive business process monitoring training, instances may be highly correlated [2], impeding the application of techniques for instance selection. Such instances are computed from event data that are recorded by the information system supporting business processes [14]. The event data are correlated by the notion of *case*, e.g., patients in a hospital or products in a factory. In this regard, we need new techniques for instance selection applicable to event data.

In this work, we suggest an instance selection approach for predicting the next activity, one of the 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 two real-life datasets and state-of-the-art techniques for predictive business process monitoring, including LSTM [13] and XGBoost [6].

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

Predictive process monitoring is an exceedingly active field of research. At its core, the fundamental component of predictive monitoring 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. [1] 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. [20], where the transition system is annotated with an ensemble of SVR and Naïve Bayes classifiers, to perform a more accurate time estimation. 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 [24], 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, Teinmaa et al. [27] provided a framework to combine text mining methods with Random Forest and Logistic Regression. Senderovich et al. [25] 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 [28] 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. The LSTM network model has proven to be particularly effective for predictive monitoring [8, 26], 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 [8, 26], data-aware [16], time-aware [17], text-aware [19], or model-aware [18].

A 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) [31]. 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. Moreover, recently some instance selection algorithms have been proposed to help process mining algorithms. For example, [9, 10] proposed to use instance selection techniques to improve the performance of process discovery and conformance checking procedures.

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. Each *event* is related to specific activities of the underlying process. Furthermore, we refer to a collection of events related to a specific process instance

as a *case*. 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, and let  $AT \subseteq \mathcal{AT}$  be a set of attributes. We define  $(C, E, \pi_C, \pi_E)$  as an event log, where  $\pi_C: C \times \mathcal{AT} \rightarrow \mathcal{U}$  and  $\pi_E: E \times \mathcal{AT} \rightarrow \mathcal{U}$ . Any event in the event log has a case, therefore,  $\exists_{e \in E} (\pi_E(e, case) \notin C)$  and  $\bigcup_{e \in E} (\pi_E(e, case)) = C$ .

Furthermore, let  $\mathcal{A} \subseteq \mathcal{U}$  be the universe of activities and let  $\mathcal{V} \subseteq \mathcal{A}^*$  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. 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. In some process mining applications, e.g., process discovery and conformance checking, just variant information is considered. Therefore, 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  $sl: \mathcal{EL} \rightarrow \mathcal{B}(\{\pi_E(e, activity) | e \in E\}^*)$  returns the simple event log of an event log. The set of unique variants in the event log is denoted by  $sl(EL)$ .

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.

**Definition 3 (Event log sampling).** Let  $\mathcal{EL}$  be the universe of event logs and  $\mathcal{A}$  be the universe of activities. Moreover, let  $EL = (C, E, \pi_C, \pi_E) \in \mathcal{EL}$  be an event log, we define function  $\delta: \mathcal{EL} \rightarrow \mathcal{EL}$  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,  $sl(\delta(EL)) \subseteq sl(EL)$ . We define that  $\delta$  is a variant-preserving sampling if  $sl(\delta(EL)) = sl(EL)$ .

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  $fe: \mathcal{C} \times \mathcal{EL} \rightarrow \mathcal{B}(\mathcal{U}^{|AT|} \times \mathcal{U})$  is a function that receives a case and an event log, and returns a multiset of features.

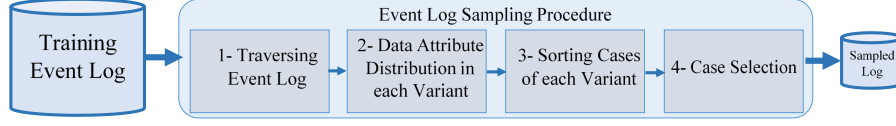


Figure 1: A schematic view of the proposed sampling procedure

For the next activity prediction, i.e., our prediction goal, the target attribute value should be an activity. 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  $\langle \langle a \rangle, b \rangle$ ,  $\langle \langle a, b \rangle, c \rangle$ , and  $\langle \langle a, b, c \rangle, d \rangle$  as features. Furthermore,  $\sum_{c \in C} fe(c, EL)$  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 [23].

## 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 sampling approach is presented in Fig. 1. We first 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.

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  $sl(EL) = \{\sigma_1, \dots, \sigma_n\}$  where  $n = |sl(EL)|$ , 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,  $\bigcup_{1 \leq i \leq n} (C_i) = C$  and  $\bigcap_{1 \leq i \leq n} (C_i) = \emptyset$ .
2. *Distribution Computation*: 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.
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. 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 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  $|sl(\delta(EL))| = |sl(EL)|$ . It is expected that using this approach, the distribution of frequency of variants will be changed and consequently the resulted prediction model will be less accurate.
- *Logarithmic distribution*: In this approach, we reduce the number of traces in each variant in a logarithmic way. If  $L = sl(EL)$  and  $L' = sl(\delta(EL))$ ,  $\forall \sigma \in L' L'(\sigma) = \lceil \text{Log}_k(L(\sigma)) \rceil$ . Using this approach, the infrequent variants will not have any trace in the sampled event log. By using a higher  $k$ , the size of the sampled event log is reduced more.
- *Division*: This approach performs similar to the previous one, however, instead of using logarithmic scale, we apply the division operator. In this approach,  $\forall \sigma \in L' L'(\sigma) = \lceil \frac{(\sigma)}{k} \rceil$ . A higher  $k$  results in fewer cases in the sample event log. Note that using this approach all the variants have at least one trace in the sampled event log.

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  $\delta$  reduces the size of the given event log  $EL$ , we use the following equation:

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

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 our research question, i.e., "Can we improve the computational performance 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

Table 1: Overview of the event logs that are used in the experiments. The accuracy and the required times (in seconds) of different prediction methods for these event logs are also presented.

Event Log	Cases	Activities	Variants	Attributes	FE Time	LSTM Train Time	LSTM Acc	XG Train Time	XG Acc
<i>RTFM</i>	150370	11	231	1	73649	3021	0.791	11372	0.814
<i>BPIC-2012-W</i>	9658	6	2643	2	1212	3344	0.68	2011	0.685

whole datasets) on the required time and the accuracy of prediction models. In the following, we first explain the event logs that are used in the experiments. Afterward, we provide some information about the implementation of sampling methods. Moreover, the experimental setting is discussed and, finally, we show the experimental results.

### 5.1 Event logs

To evaluate the proposed sampling procedure for prediction, we have used two event logs widely used in the literature. Some information about these event logs is presented in Table 1. In the *RTFM* event log, which corresponds to a road traffic management system, we have some high frequent variants and several infrequent variants. Moreover, the number of activities in this event log is high. Some of these activities are infrequent, which makes this event log imbalanced. In the *BPIC-2012-W* event log, relating to a process of an insurance company, the average of variant frequencies is lower.

### 5.2 Implementation

We have developed the sampling methods as a plug-in in the ProM framework [30], 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, to train the prediction method, we have used XGBoost [6] and LSTM [13] methods as they are widely used in the literature and outperformed their counterparts. 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 used *gbtree* with a max depth of 6 as a booster in our XGBoost model. 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 setting of both models for original event logs and sampled event logs. To access our implementations of these methods and the feature generation please refer to <https://github.com/gyunamister/pm-prediction/>. For details of the feature generation and feature encoding steps, please refer to [18].

### 5.3 Evaluation setting

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, and 10 (i.e.,  $\log_2$ ,  $\log_3$ , and  $\log_{10}$ ). For the division method, we have used 2, 5, and 10 (i.e.,  $d_2$ ,  $d_5$ ,

and  $d10$ ). For each event log and for each sampling method, we have used a 5-fold cross-validation. Moreover, as the results of the experiments are non-deterministic, all the experiments have been repeated 5 times and the average values are represented.

Note that, for both training and evaluation phases, we have used the same settings for extracting features and training prediction models. We used one-hot encoding to encode the sequence of activities for both LSTM and XGBoost models. We ran the experiment on a server with Intel Xeon CPU E7-4850 2.30GHz, and 512 GB of RAM. In all the steps, one CPU thread has been used. We employed the *Weighted Accuracy* metric [22] to compute how a prediction method performs for test data. To compare the accuracy of the prediction methods, we use the *relative accuracy* that is defined as follows.

$$R_{Acc} = \frac{\text{Accuracy using the sampled training log}}{\text{Accuracy using the whole training log}} \quad (2)$$

If  $R_{Acc}$  is close to 1, it 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 of prediction methods has improved.

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

$$R_t = \frac{\text{Training time using whole data}}{\text{Training time using the sampled data}} \quad (3)$$

$$R_{FE} = \frac{\text{Feature extraction time using whole data}}{\text{Feature extraction time using the sampled data}} \quad (4)$$

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

#### 5.4 Experimental results

Table 2 presents the reduction rate and the improvement in the feature extraction phase using different sampling methods. As it is expected, the highest reduction rate is for  $log_{10}$  (as it removes infrequent variants and keeps few traces of frequent variants), and respectively it has the biggest improvement in  $R_{FE}$ . Moreover, the lowest reduction is for  $d2$ , especially if there are lots of unique variants in the event log (i.e., for the *RTFM* event log). We expected smaller event logs to require less feature extraction time. However, results indicate that the relationship is not linear, and by having more reduction in the size of the sampled event log there will be a much higher reduction in the feature extraction time.

In Table 3 and Table 4, the results of improvement in  $R_t$  and  $R_{Acc}$  are shown for LSTM and XG prediction methods. As expected, by using fewer cases in the training, the performance of training time improvement will be higher. Comparing the results in these two tables and the results in Table 2, it is interesting to see that in some cases, even by having a high reduction rate, the accuracy of the trained prediction model is close to the case in which whole training log is used. For example, using  $d10$  for the *RTFM* event log, we will have high accuracy for both prediction methods. In other words, we are able to improve the performance of the prediction procedure while the accuracy is still reasonable.



Table 2: The reduction in the size of training logs (i.e.,  $R_S$ ) and the improvement in the performance of feature extraction part (i.e.,  $R_{FE}$ ) using different sampling methods.

Sampling Methods	d2		d3		d10		$log_2$		$log_3$		$log_{10}$		unique	
	$R_S$	$R_{FE}$	$R_S$	$R_{FE}$	$R_S$	$R_{FE}$	$R_S$	$R_{FE}$	$R_S$	$R_{FE}$	$R_S$	$R_{FE}$	$R_S$	$R_{FE}$
Event Log														
RTFM [7]	1.99	4.8	3.0	11.1	9.8	106.9	153.5	12527.6	236.3	23699.2	<b>572.3</b>	<b>74912.8</b>	285.1	24841.8
BPIC-2012-W [29]	1.22	1.37	1.41	1.80	1.66	2.51	6.06	22.41	9.05	37.67	28.50	208.32	1.73	2.36

Table 3: The accuracy and the improvement in the performance of prediction using different sampling methods for LSTM.

Sampling Methods	d2		d3		d10		$log_2$		$log_3$		$log_{10}$		unique	
	$R_{Acc}$	$R_t$	$R_{Acc}$	$R_t$	$R_{Acc}$	$R_t$	$R_{Acc}$	$R_t$	$R_{Acc}$	$R_t$	$R_{Acc}$	$R_t$	$R_{Acc}$	$R_t$
Event Log														
RTFM	1.001	2.0	<b>1.004</b>	2.9	0.990	9.0	0.716	26.7	0.724	33.0	0.767	41.8	0.631	29.1
BPIC-2012-W	1.000	1.4	0.985	1.3	0.938	1.3	0.977	4.7	0.970	5.8	0.876	11.9	0.996	1.6

Table 4: The accuracy and the improvement in the performance of prediction using different sampling methods for XGBoost.

Sampling Methods	d2		d3		d10		$log_2$		$log_3$		$log_{10}$		unique	
	$R_{Acc}$	$R_t$	$R_{Acc}$	$R_t$	$R_{Acc}$	$R_t$	$R_{Acc}$	$R_t$	$R_{Acc}$	$R_t$	$R_{Acc}$	$R_t$	$R_{Acc}$	$R_t$
Event Log														
RTFM	<b>1.000</b>	2.4	<b>1.000</b>	1.4	<b>1.000</b>	<b>84.1</b>	0.686	126.4	0.706	191.8	0.772	355.0	0.582	297.7
BPIC-2012-W	0.999	2.3	0.998	2.4	0.997	3.4	0.923	10.7	0.970	16.7	0.883	64.8	0.997	2.8

When using the LSTM prediction method for the RTFM event log, there are some cases where we have accuracy improvement. For example, using  $d3$ , there is a 0.4% improvement in the accuracy of the trained model. It is mainly because of the existence of high frequent variants. These variants lead to having unbiased training logs and consequently, the accuracy of the trained model will be lower for infrequent behaviors.

## 6 Discussion

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. In other words, there are some cases where the training process is much faster than the normal procedure, even though the trained model provides an almost similar accuracy. We did not provide the results for other metrics; however, there are similar patterns for weighted recall, precision, and f1-score. Thus, the proposed sampling methods can be used when we aim to apply hyperparameter optimization [3]. 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.

Another important outcome of the results is that for different event logs, we should use different sampling methods to achieve the highest performance. 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 logarithmic and unique distributions to sample event logs. The results indicate that the effect of the chosen distribution (i.e., *unique*, *division*, and *logarithmic*) is more important than the used  $k$ -value. Therefore, it would be valuable 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, the *division* and *unique* methods are not able to have remarkable  $R_S$  and consequently,  $R_{FE}$  and  $R_t$  will be close to 1.

Moreover, results have shown that by oversampling the event logs, although we will have a very big improvement in the performance of the prediction procedure, the accuracy of the trained model is significantly lower than the accuracy of the model that is trained by the whole event log. Therefore, we suggest gradually increasing (or decreasing) the size of the sampled event log in the hyper-parameter optimization scenarios.

By analysis of the results using common prediction methods, we have found that the infrequent activities can be ignored using some hyper-parameter settings. This is mainly because the event logs are unbalanced for these infrequent activities. 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.

Finally, in real scenarios, the process can change because of different reasons [5]. 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, we are able to adapt to the changes faster, which may be critical for specific applications.

## 7 Conclusion

In this paper, we proposed to use the subset of event logs to train prediction models. We proposed different sampling methods for next activity prediction. These methods are implemented in the ProM framework. To evaluate the proposed methods, we have applied them on two real event logs and have used two state-of-the-art prediction methods: LSTM and XGBoost. The experimental results have shown that, using the proposed method, we are able to improve the performance of the next activity prediction procedure while retaining an acceptable accuracy (in some experiments, the accuracy increased). However, there is a relation between event logs characteristics and suitable parameters that can be used to sample these event logs. The proposed methods can be helpful in situations where we aim to train the model fastly or in hyper-parameter optimization scenarios. Moreover, in cases where the process can change over time, we are able to adapt to the modified process more quickly using sampling methods.

To continue this research, we aim to extend the experiments to study the relationship between the event log characteristics and the sampling parameters. Additionally, we plan to provide some sampling methods that help prediction methods to predict infrequent activities, which could be more critical in the process. Finally, it is interesting to investigate more on using sampling methods for other prediction method applications such as last activity and remaining time prediction.

## Acknowledgment

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

## References

1. van der Aalst, W.M.P., Schonenberg, M., Song, M.: Time prediction based on process mining **36**(2), 450–475. <https://doi.org/10.1016/j.is.2010.09.001>
2. van der Aalst, W.M.P.: *Process Mining - Data Science in Action, Second Edition*. Springer (2016)
3. Bergstra, J., Bardenet, R., Bengio, Y., Kégl, B.: 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 (2011)
4. Breiman, L.: Bagging predictors. *Machine learning* **24**(2), 123–140 (1996)
5. Carmona, J., Gavaldà, R.: 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. Springer (2012)
6. Chen, T., Guestrin, C.: 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. ACM (2016)
7. De Leoni, M., Mannhardt, F.: Road traffic fine management process. Eindhoven University of Technology. Dataset (2015)
8. Evermann, J., Rehse, J., Fettke, P.: Predicting process behaviour using deep learning. *Decis. Support Syst.* **100**, 129–140 (2017)
9. Fani Sani, M., van Zelst, S.J., van der Aalst, W.M.P.: 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. Springer (2020)
10. Fani Sani, M., van Zelst, S.J., van der Aalst, W.M.P.: The impact of biased sampling of event logs on the performance of process discovery. *Computing* **103**(6), 1085–1104 (2021)
11. Garca, S., Luengo, J., Herrera, F.: *Data Preprocessing in Data Mining*. Springer Publishing Company, Incorporated (2014)
12. Hitfox Group, Breuker, D., Matzner, M., University of Muenster, Delfmann, P., University of Koblenz-Landau, Becker, J., University of Muenster: Comprehensible predictive models for business processes **40**(4), 1009–1034. <https://doi.org/10.25300/MISQ/2016/40.4.10>
13. Huang, Z., Xu, W., Yu, K.: Bidirectional LSTM-CRF models for sequence tagging. *CoRR abs/1508.01991* (2015)
14. de Leoni, M., van der Aalst, W.M.P., Dees, M.: A general process mining framework for correlating, predicting and clustering dynamic behavior based on event logs **56**, 235–257. <https://doi.org/10.1016/j.is.2015.07.003>
15. Marquez-Chamorro, A.E., Resinas, M., Ruiz-Cortes, A.: Predictive monitoring of business processes: A survey **11**(6), 962–977. <https://doi.org/10.1109/TSC.2017.2772256>
16. Navarin, N., Vincenzi, B., Polato, M., Sperduti, A.: LSTM networks for data-aware remaining time prediction of business process instances. In: *2017 IEEE Symposium Series on Computational Intelligence, SSCI 2017, Honolulu, HI, USA, November 27 - Dec. 1, 2017*. pp. 1–7. IEEE (2017)
17. Nguyen, A., Chatterjee, S., Weinzierl, S., Schwinn, L., Matzner, M., Eskofier, B.M.: Time matters: Time-aware lstms for predictive business process monitoring. In: Leemans, S.J.J., Leopold, H. (eds.) *Process Mining Workshops - ICPM 2020 International Workshops, Padua, Italy, October 5-8, 2020, Revised Selected Papers*. vol. 406, pp. 112–123. Springer (2020)
18. Park, G., Song, M.: Predicting performances in business processes using deep neural networks. *Decis. Support Syst.* **129** (2020)

19. Pegoraro, M., Uysal, M.S., Georgi, D.B., van der Aalst, W.M.P.: Text-aware predictive monitoring of business processes. In: Abramowicz, W., Auer, S., Lewanska, E. (eds.) 24th International Conference on Business Information Systems, BIS 2021, Hannover, Germany, June 15-17, 2021. pp. 221–232 (2021)
20. Polato, M., Sperduti, A., Burattin, A., de Leoni, M.: Time and activity sequence prediction of business process instances. *Computing* **100**(9), 1005–1031 (2018)
21. Pourghassemi, B., Zhang, C., Lee, J.H., Chandramowlishwaran, A.: 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. ACM (2020)
22. Powers, D.M.W.: Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. *CoRR* **abs/2010.16061** (2020)
23. Qafari, M.S., van der Aalst, W.M.P.: 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. Springer (2020)
24. Rogge-Solti, A., Weske, M.: Prediction of remaining service execution time using stochastic petri nets with arbitrary firing delays. In: Basu, S., Pautasso, C., Zhang, L., Fu, X. (eds.) Service-Oriented Computing - 11th International Conference, ICSSOC 2013, Berlin, Germany, December 2-5, 2013, Proceedings. vol. 8274, pp. 389–403. Springer (2013)
25. Senderovich, A., Di Francescomarino, C., Ghidini, C., Jorbina, K., Maggi, F.M.: 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 (2017)
26. Tax, N., Verenich, I., Rosa, M.L., Dumas, M.: 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. Springer (2017)
27. Teinmaa, I., Dumas, M., Maggi, F.M., Di Francescomarino, C.: Predictive business process monitoring with structured and unstructured data. In: International Conference on Business Process Management. pp. 401–417. Springer (2016)
28. Teinmaa, I., Dumas, M., Rosa, M.L., Maggi, F.M.: Outcome-oriented predictive process monitoring: Review and benchmark. *ACM Transactions on Knowledge Discovery from Data (TKDD)* **13**(2), 1–57 (2019)
29. Van Dongen, B.F. (Boudewijn): BPI Challenge 2012 (2012). <https://doi.org/10.4121/UUID:3926DB30-F712-4394-AEBC-75976070E91F>
30. Verbeek, E., Buijs, J.C.A.M., van Dongen, B.F., van der Aalst, W.M.P.: Prom 6: The process mining toolkit. In: Proceedings of the Business Process Management 2010 Demonstration Track, Hoboken, NJ, USA, September 14-16, 2010. vol. 615. CEUR-WS.org (2010)
31. Wang, T., Zhu, J.Y., Torralba, A., Efros, A.A.: Dataset distillation. *arXiv preprint arXiv:1811.10959* (2020)
32. Wilson, D.R., Martinez, T.R.: Reduction techniques for instance-based learning algorithms. *Mach. Learn.* **38**(3), 257–286 (Mar 2000). <https://doi.org/10.1023/A:1007626913721>
33. Wilson, D.L.: Asymptotic properties of nearest neighbor rules using edited data. *Systems, Man and Cybernetics, IEEE Transactions on* **2**(3), 408–421 (July 1972). <https://doi.org/10.1109/TSMC.1972.4309137>
34. Zhou, L., Pan, S., Wang, J., Vasilakos, A.V.: Machine learning on big data: Opportunities and challenges. *Neurocomputing* **237**, 350–361 (2017). <https://doi.org/https://doi.org/10.1016/j.neucom.2017.01.026>