Predictive Business Process Monitoring Framework with Hyperparameter Optimization

Chiara Di Francescomarino[2(✉)], Marlon Dumas[1], Marco Federici[3], Chiara Ghidini[2], Fabrizio Maria Maggi[1], and Williams Rizzi[3]

[1] University of Tartu, Liivi 2, 50409 Tartu, Estonia
{marlon.dumas,fmmaggi}@ut.ee
[2] FBK-IRST, Via Sommarive 18, 38050 Trento, Italy
{dfmchiara,ghidini}@fbk.eu
[3] University of Trento, Via Sommarive 9, 38123 Trento, Italy
{marco.federici,williams.rizzi}@studenti.unitn.it

Abstract. Predictive business process monitoring exploits event logs to predict how ongoing (uncompleted) traces will unfold up to their completion. A predictive process monitoring framework collects a range of techniques that allow users to get accurate predictions about the achievement of a goal for a given ongoing trace. These techniques can be combined and their parameters configured in different framework instances. Unfortunately, a unique framework instance that is general enough to outperform others for every dataset, goal or type of prediction is elusive. Thus, the selection and configuration of a framework instance needs to be done for a given dataset. This paper presents a predictive process monitoring framework armed with a hyperparameter optimization method to select a suitable framework instance for a given dataset.

Keywords: Predictive process monitoring · Hyperparameter optimization · Linear temporal logic

1 Introduction

Predictive Business Process Monitoring. [10] is a family of techniques that exploits event logs extracted from information systems in order to predict how current (uncompleted) traces of a process will unfold up to their completion. Based on the analysis of event logs, a runtime component continuously provides the user with estimations of the likelihood that a goal will be achieved upon completion of any given running trace of the process.

In previous work [4,10], we presented a customizable predictive process monitoring framework comprising a set of techniques to construct models to predict whether or not an ongoing trace will ultimately satisfy a given classification function based both on: (i) the sequence of events executed in the given trace; and (ii) the values of data attributes associated to the events. The latter is important as, for example, in a patient treatment process, doctors may decide whether to

S. Nurcan et al. (Eds.): CAiSE 2016, LNCS 9694, pp. 361–376, 2016.
DOI: 10.1007/978-3-319-39696-5_22

perform a surgery or not based on the age of the patient, while in a sales process, a discount may be applied only for premium customers.

Incorporating in a single framework a range of techniques that can be combined and configured in different framework instances is a necessary step in building a tool that supports predictive business process monitoring. The construction and selection of the appropriate framework instance, indeed, can greatly impact the performance of the resulting predictions [9]. Constructing an effective instance of a predictive monitoring framework, able to maximize the performance of the underlying techniques for a given dataset, is, however, non-trivial. For example, this construction may imply a choice among different classification techniques (e.g., decision trees or random forests) and clustering algorithms (e.g., k-means, agglomerative clustering or dbscan), as well as the hyperparameters that these techniques require have to be tuned according to the specific dataset and prediction problem. While these choices may be challenging even for experts, for non-experts they often result in arbitrary (or default-case) choices [17].

The conventional way to face this problem is combining manual and exhaustive search [19]. We also adopt this approach and, in particular, we perform two specific steps: first, we run different configurations of the techniques on an appropriate dataset used for training and validating and, second, we compare the outcomes of the different configurations to select the one that outperforms the others for the given domain.

While this overall strategy has the potential to ease the construction of an effective instance of the predictive monitoring framework, its concrete realization poses two challenges that may hamper its practical adoption. A first challenge is given by the computational burden of running different configurations for different combinations of techniques. A second challenge is related to the complexity of comparing different configurations and then select the best one for a business analyst/process owner.

The framework presented in this paper provides a predictive process monitoring environment armed with a hyperparameter optimization method able to address the two challenges emphasized above. First, it enables to run an exhaustive combination of different technique settings on a given dataset in an efficient and scalable manner. This is realized through a meta-layer built on top of the predictive process monitoring framework. Such a layer is responsible of invoking the framework on different framework instances and to provide, for each of them, a number of aggregated metrics (on a set of validation traces). The meta-layer is optimized to schedule and parallelize the processing of the configurations across different threads and reuse as much as possible the pre-processed data structures. Second, it provides user support for the comparison of the results, thus enabling to easily select a suitable framework instance for a given dataset. This is done by providing the user with a set of aggregated metrics (measuring different dimensions) for each configuration. These metrics can be used for opportunely ranking the configurations according to the user's needs and hence for supporting the user in the parameter tuning.

After an introductory background section (Sect. 2), Sects. 3 and 4 introduce two motivating scenarios and the overall approach, respectively. The overall architecture is then detailed in Sect. 5, and an evaluation presented in Sect. 6. Sections 7 and 8 conclude with related and future works.

2 Background

In this section, we provide background notions useful in the rest of the paper.

Predictive Process Monitoring. The execution of business processes is generally subject to internal policies, norms, best practices, regulations, and laws. For example, a doctor may only perform a certain type of surgery, if a preoperational screening is carried out beforehand. Meanwhile, in a sales process, an order can be archived only after the customer has confirmed the receipt of all ordered items. Based on an analysis of past execution traces, the idea of *Predictive Process Monitoring* [10] is to continuously provide the user with estimations of the likelihood of achieving a user-specified *business goal* in an ongoing trace.[1] Such predictions generally depend both on: (i) the sequence of events executed in the ongoing trace; and (ii) the values of data attributes associated to the events.

Linear Temporal Logic. In our approach, a business goal can be formulated in terms of Linear Temporal Logic (LTL) rules. LTL [14] is a modal logic with modalities devoted to describe time aspects. Classically, LTL is defined for infinite traces. However, when focusing on the compliance of business processes, we use a variant of LTL defined for finite traces (since business process are supposed to complete eventually). We assume that events occurring during the process execution fall in the set of atomic propositions. LTL rules are constructed from these atoms by applying the temporal operators **X** (next), **F** (future), **G** (globally), and **U** (until) in addition to the usual boolean connectives. Given a formula φ, $\mathbf{X}\varphi$ means that the next time instant exists and φ is true in the next time instant (strong next). $\mathbf{F}\varphi$ indicates that φ is true sometimes in the future. $\mathbf{G}\varphi$ means that φ is true always in the future. $\varphi\mathbf{U}\psi$ indicates that φ has to hold at least until ψ holds and ψ must hold in the current or in a future time instant.

Hyperparameter Optimization. Traditionally, machine learning techniques are characterized by model parameters and by *hyperparameters*. While model parameters are learned during the training phase so as to fit the data, hyperparameters are set outside the training procedure and used for controlling how flexible the model is in fitting the data. For example, the number of clusters in the k-means clustering procedure is a hyperparameter of the clustering technique. The impact of hyperparameter values on the accuracy of the predictions can be huge. Optimizing their value is hence important but it can differ based on the dataset. The simplest approaches for hyperparameter optimization are grid

[1] In line with the forward-looking nature of predictive monitoring, we use the term *business goal* rather than *business constraint* to refer to the monitored properties.

search and random search. The former builds a grid of hyperparameter values, evaluates each of them by exploring the whole search space, and returns the one that provides the best result. The latter, instead of exhaustively exploring the search space, selects a sample of values to be evaluated. Several smarter techniques have been recently developed for the hyperparameter optimization. For example, Sequential Model based Optimization (SMBO) [7] is an iterative approach that constructs explicit regression models to describe the dependence of target algorithm performance on hyperparameter settings.

3 Two Motivating Scenarios

We aim at addressing the problem of easing the task of predictive process monitoring, by enabling users to easily select and configure a specific predictive process monitoring scenario to the needs of a specific dataset. In this section, we introduce two motivating scenarios that will be used also as a basis for the evaluation of the *Predictive Process Monitoring Framework* in Sect. 6.

Scenario 1. Predicting Patient History. Let *Bob* be a medical director of an oncology department of an important Hospital who is interested in predicting the type of exams a patient, *Alice*, will perform. In particular, he is interested in knowing, given the clinical record of *Alice* whether: (a) she will need two specific exams named *tumor marker* $CA-19.9$ and $ca-125$ *using meia*, and when; and (b) the occurrence of a particular exam ($CEA-$*tumor marker using meia*) will be followed by another exam for the diagnosis of *squamous cell carcinoma*. Since his department has started an innovative project aiming at using predictive process monitoring techniques based on the analysis of event logs related to the patient history, his Hospital owns a number of relevant datasets to enable the usage of a predictive process monitoring framework. However, when ready to use the framework, he finds out that: (i) he needs to select a number of techniques to create an instance of the framework; (ii) for each of these techniques, he has to set a number of hyperparameters needed for their configuration. However, being a medical doctor he does not have the necessary knowledge to understand which technique is better to use and the parameters to set. His knowledge only enables him to select the predicate he wants to predict and the dataset of similar traces relevant for the prediction. Thus, a way for helping him in understanding which configuration works best for his dataset and specific prediction is needed.

Scenario 2. Predicting Problems in Building Permit Applications. Let *John* be a clerk handling building permit applications of a Municipality. The majority of regular building permit applications required for building, modifying or demolishing houses must be accompanied with the necessary fees and documentation, including design plans, photos and pertinent reports. They are, therefore, often unsuccessfully checked for completeness, and the applicant has to be contacted again for sending the missing documents. This implies extra work for *John* and for the building permit applications office. In addition, many of the permit applications also require an environmental license (WABO) and getting the WABO

license can either be fast or demand for a long extension of the building permit procedure. This would require a rescheduling of the work of the building permit applications office. *John* is, therefore, interested in knowing whether: (a) the 4 applications he has just received and for which he has acknowledged receipt will undergo a series of actions required to retrieve missing data; and (b) these applications will require the environmental license. As in *Scenario 1*, the Municipality where *John* works stores all the necessary datasets to enable the usage of a predictive process monitoring framework, but the difficulty in choosing the right technique and the need of configuring parameters may seriously hamper his ability to use the framework. Thus, a way for helping him to set up the correct configuration which works best for his dataset and specific prediction is needed also in this scenario.

4 Approach

In this section, we describe the approach to provide users with a predictive process monitoring framework equipped with methods to support them in the selection of the framework instance that is most suitable for the dataset and the prediction they are interested in.

The approach is based on two main components: the *Predictive Process Monitoring Framework*, in charge of making predictions on an ongoing trace, and the *Technique and Hyperparameter Tuner*, responsible of the invocation of the *Predictive Process Monitoring Framework* with different configurations (framework instances). Figure 1 shows the conceptual architecture of the framework. The *Predictive Process Monitoring Framework* takes as input a training set, a prediction problem and an ongoing trace, and returns as output a prediction related to the input prediction problem for the ongoing trace. The *Technique and Hyperparameter Tuner* acts as a meta-layer on top of the *Predictive Process Monitoring Framework*. As well as the training set and the prediction problem, the *Technique and Hyperparameter Tuner* takes as input a set of traces (*validation set*) and uses them to feed the *Predictive Process Monitoring Framework* on a set of potentially interesting framework instances. Specifically, for each considered framework instance, the traces of the validation set are replayed and

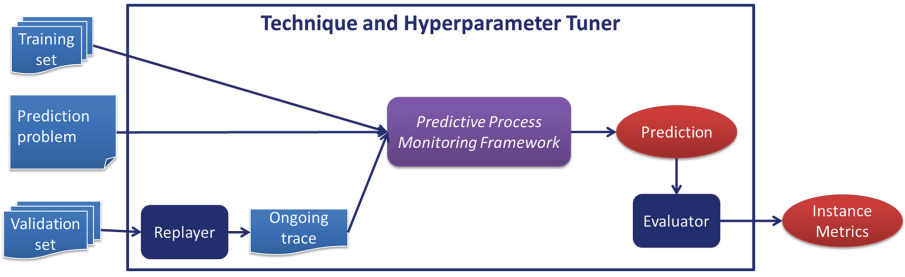

Fig. 1. *Tuning-enhanced Predictive Process Monitoring Framework* architecture

passed as a stream of events to the *Predictive Process Monitoring Framework*. Once a new trace is processed by the *Predictive Process Monitoring Framework* and a predicted value returned, it is compared with the actual value of the trace in the validation set. Based on this comparison and other characteristics of the prediction (e.g., how early along the current trace the prediction has reached a sufficient confidence level), a set of aggregated performance metrics (e.g., the accuracy or the failure rate) is computed. Once the set of all the interesting framework instances has been processed, the user can compare them along the performance dimensions.

5 Architecture

In this section, we describe in detail the two layers of the *Tuning-enhanced Predictive Process Monitoring Framework*. We first introduce the *Predictive Process Monitoring Framework*, by providing an overview of its modules and of the techniques that are currently plugged in each of them, and we then present the tuner layer that supports users in the selection of the framework instance that best suites with their dataset and prediction problem.

5.1 *Predictive Process Monitoring Framework*

As shown in Fig. 1, the *Predictive Process Monitoring Framework* requires as input a set of past executions of the process. Based on the information extracted from such execution traces, it tries to predict how current ongoing executions will develop in the future. To this aim, before the process execution, a pre-processing phase is carried out. In such a phase, state-of-the-art approaches for clustering and classification are applied to the historical data in order to (i) identify and group historical trace prefixes with a similar control flow, i.e., to delimit the search space on the control flow base (clustering from a control flow perspective); and (ii) get a precise classification in terms of data of traces with similar control flow (data-based classification). The data-structures (e.g., clusters and classifiers) computed at the different stages of the pre-processing phase are stored. At runtime, the classification of the historical trace prefixes is used to classify new traces during their execution and predict how they will behave in the future. In particular, the new trace is matched to a cluster, and the corresponding classifier is used to estimate the (class) probability for the trace to achieve a certain outcome and the corresponding (class) support (that also gives a measure of the reliability of classification algorithm outcomes). The overall picture of the framework is illustrated in Fig. 2. Within such a framework, we can identify three main modules: the *encoding*, the *clustering* and the *supervised classification learning* module. Each of them can be instantiated with different techniques. Figure 3 shows an overview of possible framework instances.

For example, for the trace encoding a *frequency based* and a *sequence based* approach have been plugged in the framework. The former is realized encoding each execution trace as a vector of event occurrences (on the alphabet of

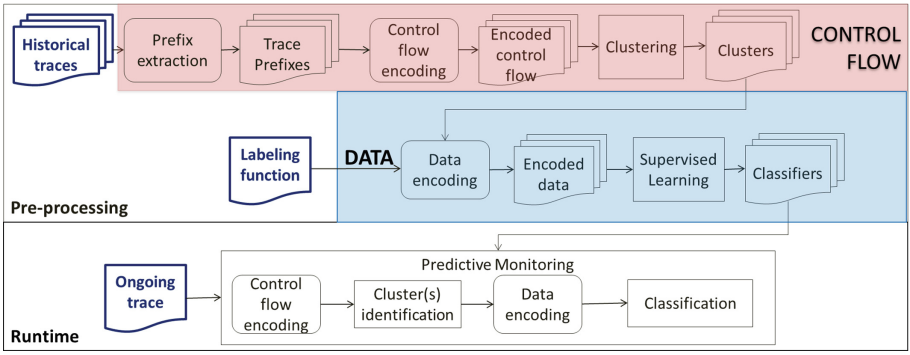

Fig. 2. *Predictive Process Monitoring Framework*

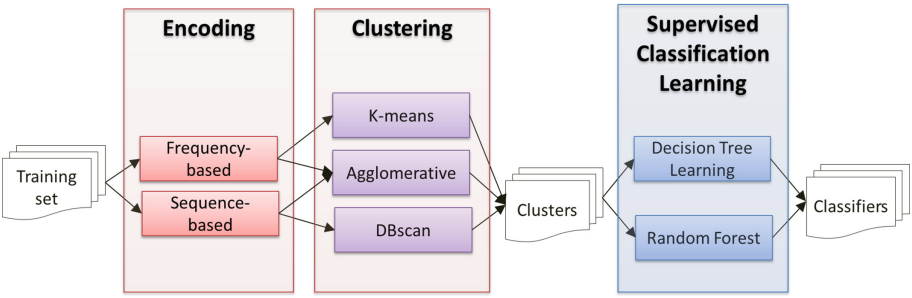

Fig. 3. Framework instances overview

the events), while, in the latter, the trace is encoded as a sequence of events. These encodings can then be passed to the clustering techniques available in the framework: the *dbscan clustering*, the *k-means clustering* and the *agglomerative clustering* algorithms. For example, the *euclidean* distance, used by the *k-means clustering*, is computed starting from the *frequency based* encoding, while the *edit* distance, used by the *dbscan clustering*, is computed starting from the *sequence based* encoding of the traces. Within the supervised learning module, *decision tree* and *random forest* learning techniques have been implemented.

Each of these techniques requires, in turn, a number of hyperparameters (specific for the technique) to be configured. Specifically, k-means and agglomerative clustering take as input the number of clusters, while the dbscan technique requires two parameters: the minimum number of points in a cluster and the minimum cluster ray.

Moreover, the framework also allows for configuring other parameters, such as:

- size of prefixes of historical traces to be grouped in clusters and used for training the classifiers;

- *voting mechanism*, so that the p clusters closest to the current trace are selected, the prediction according to the corresponding classifiers estimated, and the prediction with the highest number of votes (from the classifiers) returned;
- when the prediction is related to a time interval, a mechanism for the definition of the time interval (e.g., q intervals of the same duration, based on q-quantiles, based on a normal distribution of the time).

The framework has been implemented as an Operational Support (OS) provider of the OS Service 2.0 [11,18] of the ProM toolset. Specifically, an OS service is able to interact with external workflow engines by receiving at runtime streams of events and passing them to the OS providers.

5.2 Technique and Hyperparameter Tuning

The *Tuning-enhanced Predictive Process Monitoring Framework* has been designed as a client-server architecture, where the *Predictive Process Monitoring Framework* is the server, and the client is a toolset that can either be used (i) for "replaying" a stream of events coming from a workflow engine and invoke the server to get predictions on a specific problem; or (ii) for evaluation purposes and, in particular, for supporting users in tuning the framework techniques and hyperparameters according to the dataset and the input prediction problem.

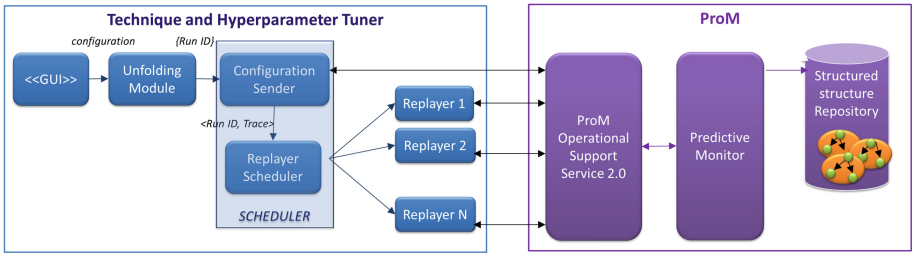

Fig. 4. Logical architecture

When used for evaluation purposes, the client (the *Technique and Hyperparameter Tuner*) evaluates the *Predictive Process Monitoring Framework* for each of the techniques and hyperparameter configurations. Specifically, for each of them, the client replays each trace of the validation set and invokes the *Predictive Process Monitoring Framework* for getting a prediction (and the associated class probability) at different points of the trace (*evaluation points*). As soon as the class probability and support of the returned prediction are above a certain threshold, the prediction is considered reliable enough and kept as the *Predictive Process Monitoring Framework* prediction at the specific evaluation point. The final predicted value is then compared with the actual one. With this information

for each trace, the client is finally able to provide the users with few aggregated metrics about the performance of the framework instance. In detail, the following three *evaluation dimensions* (and corresponding metrics) are computed:

- *Accuracy*, which intuitively represents the proportion of correctly classified results (both positive and negative); it is defined as:

$$accuracy = \frac{T_P + T_N}{T_P + F_P + T_N + F_N}. \quad (1)$$

  Accuracy ranges between 0 and 1. High values of accuracy are preferred to low ones.
- *Failure rate*, which is the percentage of traces for which a reliable prediction cannot be given. Failure rate ranges between 0 and 1. In this case low values are preferred to high ones.
- *Earliness*, which is the ratio between the index indicating the position of the last evaluation point (the one corresponding to the reliable prediction) and the size of the trace under examination. Earliness ranges as well between 0 and 1 and a low value of earliness indicates early predictions along the traces.

In order to speed-up the above time-consuming procedure, the client application implements a scheduling mechanism that distributes the prediction computations across 2 or more parallel replayer threads. In addition, in the pre-processing phase, the data structures are stored for reuse purposes. However, only some of them can be reused. Each choice of technique (and hyperparameter) in the configuration can indeed affect the *Predictive Process Monitoring Framework* flow at different stages. For example, the choice of the encoding type affects the clusters built from the historical traces; the choice of the classification learning technique does not affect the clusters but it does affect the classifiers built on top of them. Only the data structures of previous choices that are not affected by the current choice can be reused.

Figure 4 shows the logical architecture of the client application (left part) and its interactions with the OS Service. It is composed of three main parts: the *Unfolding Module*, the *Scheduler Module* and the *Replayers*.

The *Unfolding Module* combines the sets of techniques (and their hyperparameters) provided by the user through an intuitive GUI into a set of different configuration runs. Each configuration run is associated with an ID (*Run ID*), which is used to refer such a configuration. Once the list of the interesting configurations has been created, the *Configuration Sender* sequentially sends each configuration to the server that uses it to encode the traces, as well as to compute clusters and classifiers for that specific configuration. Once the server has done with the pre-processing, the *Configuration Sender* starts sending the traces to the *Replayer Scheduler* in charge of optimizing the distribution of the traces among different replayers on different threads. Each replayer sends the trace (and the reference to the specific configuration Run ID) to the server and waits for the results. As soon as the results are provided by the OS Service, they are progressively visualized in the result interface (Fig. 5). Each tab of the result

interface refers to a specific configuration run, while the summary tab reports a summary of all the configuration runs with the corresponding evaluation metrics. From this interface, the user can easily sort the configurations based on one or more evaluation metrics.

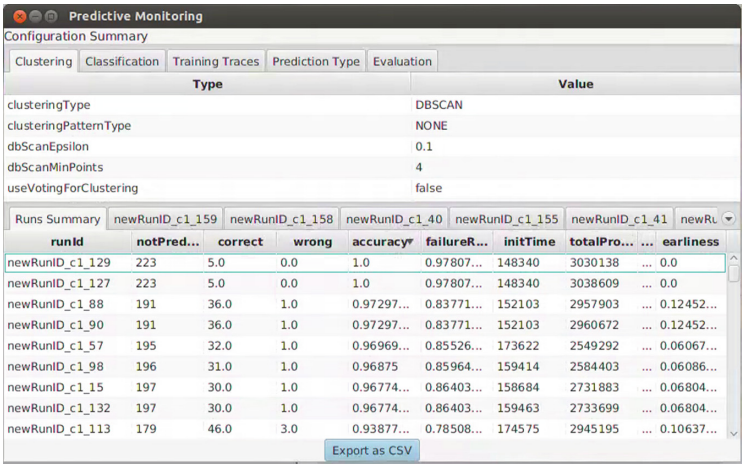

Fig. 5. Result interface

6 Evaluation

In this section, we provide an evaluation of the *Tuning-enhanced Predictive Process Monitoring Framework*. In detail, we would like to investigate if it can be used in practice to support users in selecting a suitable configuration for their prediction problem. Specifically, we want to see whether: (i) the *Tuning-enhanced Predictive Process Monitoring Framework* is effective in returning a set of configurations suitable for the specific dataset and prediction problem; (ii) the configuration suggested by the *Tuning-enhanced Predictive Process Monitoring Framework* actually provides accurate results for the specific prediction problem; (iii) the framework does it in a reasonable amount of time.

6.1 Datasets

For the tool evaluation, we used two datasets provided for the BPI Challenges 2011 [1] and 2015 [5], respectively.

The first event log pertains to the treatment of patients diagnosed with cancer in a large Dutch Academic Hospital. It contains 1,140 traces, 149,730 events referring to 623 different activities. In this case, we used our framework to predict the information that, for example, *Bob* is interested to know about the *Alice*'s case (see *Scenario 1* in Sect. 3). More formally, we used our framework to predict the compliance of a trace with respect to the following two LTL rules:

- $\varphi_{11} = \mathbf{F}(\text{“}tumor\ marker\ CA - 19.9\text{”}) \vee \mathbf{F}(\text{“}ca - 125\ using\ meia\text{”})$,
- $\varphi_{12} = \mathbf{G}(\text{“}CEA - tumor\ marker\ using\ meia\text{”} \rightarrow \mathbf{F}(\text{“}squamous\ cell\ carcinoma\ using\ eia\text{”}))$.

The second log was provided by a Dutch Municipality. The log is composed of 1,199 traces, 52,217 events and 398 event classes. The data contains all building permit applications over a period of approximately four years. It contains several activities, denoted by both codes and labels, both in Dutch and in English. In this case, we used the *Tuning-enhanced Predictive Process Monitoring Framework* to investigate the configurations that are more suitable with respect to the *John*'s problem (see *Scenario 2* in Sect. 3). Formally, we investigate the following two LTL rules:

- $\varphi_{21} = (\mathbf{F}(\text{“}start\ WABO\ procedure\text{”}) \wedge \mathbf{F}(\text{“}extend\ procedure\ term\text{”}))$,
- $\varphi_{22} = (\mathbf{G}(\text{“}send\ confirmation\ receipt\text{”}) \rightarrow \mathbf{F}(\text{“}retrieve\ missing\ data\text{”}))$.

6.2 Experimental Procedure

In order to evaluate the technique and hyperparameter tuning of the *Tuning-enhanced Predictive Process Monitoring Framework*, we adopted the following procedure.

1. We divided both our datasets in three parts: (i) training set: 70 % of the whole dataset; (ii) validation set: 20 % of the whole dataset; (iii) testing set: 10 % of the whole dataset.
2. For both the analyzed scenarios, we used the training and the validation sets for identifying the most suitable (according to one or more evaluation dimensions) *Predictive Process Monitoring Framework* configurations for the specific dataset and prediction problem. Moreover, we computed the time required for tuning the parameters with and without reuse of data structures and with and without replayers working in parallel.
3. We evaluated the identified configurations on the testing set.

6.3 Experimental Results

As described in Sect. 5, the *Tuning-enhanced Predictive Process Monitoring Framework* explores all the configurations of a finite set and computes for each of them, three evaluation metrics: accuracy, failure rate and earliness. Table 1 reports, for each formula of each scenario, the descriptive statistics of these metrics on a set of 160 different configurations, obtained by combining two algorithms for the clustering step (dbscan and k-means), two algorithms for the classifier learning step (decision tree and random forest) and varying a number of hyperparameters (e.g., the number of clusters for k-means or the number of trees in the random forest).

By looking at the table, we can get an idea of the distribution of the configuration settings in the space of the evaluation metrics. We can observe that such a distribution is not the same for all the rules. For example, for the rules in the first scenario, the configurations produce values for all the three evaluation metrics that are widely distributed (e.g., the failure rate for φ_{11} ranges from 0 to 0.98).

Table 1. Descriptive Statistics related to the tuning phase

Rule	Accuracy				Failure rate				Earliness				Computation
	Min	Max	Avg	Std. dev	Min	Max	Avg	Std. dev	Min	Max	Avg	Std. dev	Time (h)
φ_{11}	0.43	1	0.73	0.15	0	0.98	0.42	0.31	0	0.48	0.13	0.13	42.68
φ_{12}	0.55	0.91	0.73	0.08	0	0.93	0.27	0.3	0	0.43	0.07	0.09	32.05
φ_{21}	0.87	0.91	0.87	0.006	0	0.29	0.02	0.05	0	0.09	0.008	0.02	1.87
φ_{22}	0.77	1	0.95	0.06	0	0.76	0.09	0.17	0	0.35	0.06	0.08	2.93

When, like in this case, the results obtained by running different configurations are distributed, the configuration that best fits with the user needs can be identified in the tuning phase (for example, the user could prefer earliness more than accuracy). On the other hand, for the other two rules, and, in particular for φ_{21}, the performance of the different tested configurations do not vary significantly. In this case, the different configuration settings are mostly restricted within a limited area of the space of the three evaluation metrics, thus making the results of the prediction less dependent on the choice of the configuration.

Table 2. Results related to the tuning evaluation

Rule	Conf. ID	Choice criterion	Tuning			Evaluation		
			Accuracy	Failure rate	Earliness	Accuracy	Failure rate	Earliness
φ_{11}	109	accuracy	0.92	0.46	0.074	0.86	0.57	0.056
	4	fail. rate	0.6	0	0.02	0.86	0	0.009
	50	earliness	0.73	0.06	0.004	0.62	0.05	0.003
	108	balance	0.85	0.18	0.096	0.84	0.26	0.107
φ_{12}	108	accuracy	0.89	0.43	0.016	0.95	0.46	0.129
	76	fail. rate	0.75	0	0.03	0.73	0.02	0.026
	149	earliness	0.64	0	0.001	0.69	0	0
	154	balance	0.77	0.1	0.016	0.87	0.05	0.028
φ_{21}	17	accuracy	0.91	0.29	0.033	0.92	0.12	0.013
	86	fail. rate	0.87	0	0.004	0.91	0	0.002
	65	earliness	0.87	0	0	0.91	0	0
	65	balance	0.87	0	0	0.91	0	0
φ_{22}	22	accuracy	1	0.12	0.246	1	0.26	0.335
	136	fail. rate	0.98	0	0.021	1	0	0
	127	earliness	0.98	0.04	0.001	1	0	0
	25	balance	0.99	0.03	0.12	0.96	0.06	0.18

Among the configurations in the set, we picked the ones that a user could be interested in a typical scenario like the ones considered in this evaluation. We selected as choice *criteria* the performance of the configuration with respect to each of the evaluation dimensions and the performance of the configuration with respect to all the evaluation dimensions. Specifically, we selected, for each evaluation dimension, the configuration that scores best (with respect to that dimension), provided that the other two dimensions do not significantly underperform. Furthermore, we manually selected a fourth configuration that balances the performance of the three evaluation dimensions. Table 2 (*Tuning* column)

shows, for each rule, the best (in terms of accuracy, failure rate, earliness and a mix of the three) configurations and the corresponding performance. The identified configurations differ one from another not only for the hyperparameter values but also for the selected algorithms. For example, in configuration 109 for rule φ_{11}, identified as the one with the best accuracy, the clustering algorithm is dbscan, while in configuration 22, i.e., the one with the best accuracy for rule φ_{22}, the clustering algorithm is k-means.

In order to evaluate whether the identified configurations could actually answer the prediction problem in the specific domain, we evaluated them on the testing set. Table 2 (*Evaluation* column) shows the results obtained in terms of accuracy, failure rate and earliness. By comparing the results obtained with the ones obtained in the tuning phase, we can observe that, according to our expectations, they are aligned. Moreover, by further inspecting the table, we have a confirmation of the trend that we observed by looking at the descriptive statistics of the data related to the tuning phase (Table 1). The values of the three metrics along the four selected configurations are quite similar for the rules in *Scenario 2*, whereas they differ for the rules in *Scenario 1*. In the latter, hence, the user (e.g., *Bob*) has the possibility to choose the configuration based on his needs. If, for example, he is more interested in getting accurate predictions, he would choose configuration 109 for φ_{11} and 108 for φ_{12}. If, he is more interested in obtaining a prediction, taking the risk that it could also be inaccurate, then he would choose configurations 4 and 76 for the two rules, respectively. Similarly for early predictions and predictions balancing all the three dimensions.

Finally, we looked at the time required by the *Tuning-enhanced Predictive Process Monitoring Framework* for processing the configurations for each of the four rules. The last column of Table 1 reports the overall time spent to this purpose. Here, we can notice a difference in the computation time required by the two datasets. This difference can be due to the difference in the length of the traces in the two datasets. Indeed, the traces of the dataset related to the Dutch Academic Hospital are on average longer than the ones in the Dutch Municipality dataset. Moreover, in order to investigate the time saved with the reuse of data structures, we performed a run in which all the data structures had already been computed and stored in the server and we observed a time reduction of about 20 %. Finally, we performed a further run with 8 replayers rather than with a single replayer and we observed a further time reduction of about 13.1 %.

Threats to Validity. Three main external threats to validity affect the results of the evaluation: (i) the subjectivity introduced by the user; (ii) the potential overfitting introduced during the tuning phase; and (iii) the limited number of analyzed scenarios. Concerning the first threat, the user is involved in the process (and hence in the evaluation) both in the initial definition of the configurations and in the selection of the configuration. The results of the experiment could hence be influenced by the human subjectivity in these choices. We tried to mitigate the impact of this threat by analyzing what a user would do in "typical" scenarios. As for the second threat, the construction of the configuration

parameters would have benefit of a cross-validation procedure, which would have increased the stability of the results. Finally, although we only limited our evaluation to two datasets and to four specific rules, we defined realistic scenarios and used real-life logs.

7 Related Work

In the literature, there are two main branches of works related to this paper: those concerning predictive monitoring and those related to hyperparameter optimization.

As for the first branch, there are works dealing with approaches for the generation of predictions, during process execution, focused on the time perspective. In [2], the authors present a set of approaches in which annotated transition systems, containing time information extracted from event logs, are used to: (i) check time conformance; (ii) predict the remaining processing time; and (iii) recommend appropriate activities to end users to improve the process performance. In [6], a predictive clustering approach is presented, in which context-related execution scenarios are discovered and modeled through state-aware performance predictors. In [15], the authors use stochastic Petri nets to predict the remaining execution time of a process. Another group of works, in the literature, focuses on approaches that generate predictions and recommendations to reduce risks. For example, in [3], the authors present a technique to support process participants in making risk-informed decisions with the aim of reducing the process risks. In [13], the authors make predictions about time-related process risks by identifying and exploiting statistical indicators that highlight the possibility of transgressing deadlines. In [16], an approach for Root Cause Analysis through classification algorithms is presented.

A key difference between these approaches and the *Tuning-enhanced Predictive Process Monitoring Framework* approach is that they either rely on the control-flow or on the data perspective for making predictions at runtime, whereas the predictive process monitoring framework [4,10] takes both perspectives into consideration. In addition, we provide a general, customizable framework for predictive process monitoring, which is flexible and can be implemented in different variants with different sets of techniques, and which supports users in the tuning phase.

As for the second branch of works, several approaches in machine learning have been proposed for the selection of learning techniques [12], for the tuning of hyperparameters [7], and for the combined optimization of both techniques and hyperparameters [8,17].

The problem that we address is to tune both the machine learning technique and hyperparameter values. One of the first works that falls in this group of approaches is Auto-WEKA [17]. The idea of this work is to map the problem of algorithm selection to that of hyperparameter optimization and to approach this problem based on sequential model-based optimization and a random forest regression model. MLbase [8] also addresses the same problem as Auto-WEKA

and approaches it using distributed data mining algorithms. Differently from these approaches, the problem that we face in this work is more complex. In our case, we have more than one machine learning (sub-)problem (e.g., clustering and classification) and these sub-problems depend on each other. Hence, the algorithm (and hyperparameter) optimization for a (sub-)problem cannot be defined independently of the other sub-problems. This is why the solution we propose combines manual and exhaustive search.

8 Conclusion

The contribution of this paper is a predictive process monitoring framework incorporating a hyperparameter optimization method that supports users in the selection of the techniques and in the tuning of the hyperparameters according to the specific dataset and prediction problem under analysis. We evaluated the approach on two datasets and we found that the *Tuning-enhanced Predictive Process Monitoring Framework* provides users with interesting sets of tunable configurations in a reasonable time. This allows users to adopt configurations that generate accurate predictions for the specific dataset and prediction problem.

In the future, we plan to further investigate: (i) how to increase the user support; (ii) how to optimize the exhaustive search. Concerning the former, we would like to provide users with an automatic heuristic-based approach for the exploration of the search space. This would allow us to go beyond the exhaustive analysis of a limited search space of the configurations by exploiting an objective function to explore a larger search space. For example, we could use as objective function each of the evaluation metrics considered in this work or we could use a multi-objective function for the optimization of all three of them. As for the latter, we would like to borrow state-of-the-art techniques for algorithm selection and hyperparameter tuning and, if possible, to customize them for our problem.

Finally, a further interesting direction is to extend our framework to support prescriptive process monitoring. The idea is to provide recommendations to the user to achieve a certain goal in a given ongoing trace. Recommendations would allow users not only to know whether a goal will be achieved but also what to do for increasing the chances of achieving it.

Acknowledgments. This research is funded by the EU FP7 Programme under grant agreement 609190 (Subject-Orientation for People-Centred Production) and by the Estonian Research Council (grant IUT20-55).

References

1. 3TU Data Center: BPI Challenge 2011 Event Log (2011)
2. van der Aalst, W.M.P., Schonenberg, M.H., Song, M.: Time prediction based on process mining. Inf. Syst. **36**(2), 450–475 (2011)

3. Conforti, R., de Leoni, M., La Rosa, M., van der Aalst, W.M.P.: Supporting risk-informed decisions during business process execution. In: Salinesi, C., Norrie, M.C., Pastor, Ó. (eds.) CAiSE 2013. LNCS, vol. 7908, pp. 116–132. Springer, Heidelberg (2013)
4. Di Francescomarino, C., Dumas, M., Maggi, F.M., Teinemaa, I.: Clustering-Based Predictive Process Monitoring. arXiv preprint (2015)
5. van Dongen; B.: Bpi challenge (2015). doi:10.4121/uuid:a0addfda-2044-4541-a450-fdcc9fe16d17
6. Folino, F., Guarascio, M., Pontieri, L.: Discovering context-aware models for predicting business process performances. In: Meersman, R., Panetto, H., Dillon, T., Rinderle-Ma, S., Dadam, P., Zhou, X., Pearson, S., Ferscha, A., Bergamaschi, S., Cruz, I.F. (eds.) OTM 2012, Part I. LNCS, vol. 7565, pp. 287–304. Springer, Heidelberg (2012)
7. Hutter, F., Hoos, H.H., Leyton-Brown, K.: Sequential model-based optimization for general algorithm configuration. In: Coello, C.A.C. (ed.) LION 2011. LNCS, vol. 6683, pp. 507–523. Springer, Heidelberg (2011)
8. Kraska, T., Talwalkar, A., Duchi, J.C., Griffith, R., Franklin, M.J., Jordan, M.I.: Mlbase: A distributed machine-learning system. In: CIDR (2013). www.cidrdb.org
9. Luo, G.: Mlbcd: a machine learning tool for big clinical data. Health Inf. Sci. Syst. **3**(1), 1–19 (2015)
10. Maggi, F.M., Di Francescomarino, C., Dumas, M., Ghidini, C.: Predictive monitoring of business processes. In: Jarke, M., Mylopoulos, J., Quix, C., Rolland, C., Manolopoulos, Y., Mouratidis, H., Horkoff, J. (eds.) CAiSE 2014. LNCS, vol. 8484, pp. 457–472. Springer, Heidelberg (2014)
11. Maggi, F.M., Westergaard, M.: Designing software for operational decision support through coloured Petri nets. Enterprise Information Systems, 1–21 (2015)
12. Pfahringer, B., Bensusan, H., Giraud-Carrier, C.: Meta-learning by landmarking various learning algorithms. In: ICML, pp. 743–750 (2000)
13. Pika, A., Aalst, W., Fidge, C., Hofstede, A., Wynn, M.: Predicting deadline transgressions using event logs. In: La Rosa, M., Soffer, P. (eds.) BPM Workshops, pp. 211–216. Springer, Heidelberg (2013)
14. Pnueli, A.: The temporal logic of programs. In: 18th Annual Symposium on Foundations of Computer Science, pp. 46–57 (1977)
15. 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.) ICSOC 2013. LNCS, vol. 8274, pp. 389–403. Springer, Heidelberg (2013)
16. Suriadi, S., Ouyang, C., Aalst, W., Hofstede, A.: Root cause analysis with enriched process logs. In: La Rosa, M., Soffer, P. (eds.) BPM Workshops, pp. 174–186. Springer, Heidelberg (2013)
17. Thornton, C., Hutter, F., Hoos, H.H., Leyton-Brown, K.: Auto-WEKA: combined selection and hyperparameter optimization of classification algorithms. In: Proceedings of KDD-2013, pp. 847–855 (2013)
18. Westergaard, M., Maggi, F.M.: Modeling and verification of a protocol for operational support using coloured petri nets. In: Kristensen, L.M., Petrucci, L. (eds.) PETRI NETS 2011. LNCS, vol. 6709, pp. 169–188. Springer, Heidelberg (2011)
19. Wistuba, M., Schilling, N., Schmidt-Thieme, L.: Hyperparameter search space pruning – a new component for sequential model-based hyperparameter optimization. In: Appice, A., Rodrigues, P.P., Santos Costa, V., Gama, J., Jorge, A., Soares, C. (eds.) ECML PKDD 2015. LNCS, vol. 9285, pp. 104–119. Springer, Heidelberg (2015)