Towards Reliable Predictive Process Monitoring

Christopher Klinkmüller[1(✉)], Nick R. T. P. van Beest[2], and Ingo Weber[1]

[1] Data61, CSIRO, Sydney, Australia
{christopher.klinkmuller,ingo.weber}@data61.csiro.au
[2] Data61, CSIRO, Brisbane, Australia
nick.vanbeest@data61.csiro.au

Abstract. Predictive process monitoring is concerned with anticipating the future behavior of running process instances. Prior work primarily focused on the performance of monitoring approaches and spent little effort on understanding other aspects such as reliability. This limits the potential to reuse the approaches across scenarios. From this starting point, we discuss how synthetic data can facilitate a better understanding of approaches and then use synthetic data in two experiments. We focus on prediction as classification of process instances during execution, solely considering the discrete event behavior. First, we compare different feature representations and reveal that sub-trace occurrence can cover a broader variety of relationships in the data than other representations. Second, we present evidence that the popular strategy of cutting traces to certain prefix lengths to learn prediction models for ongoing instances is prone to yield unreliable models and that the underlying problem can be avoided by using approaches that learn from complete traces. Our experiments provide a basis for future research and highlight that an evaluation solely targeting performance incurs the risk of incorrectly assessing benefits and limitations.

Keywords: Behavioral classification · Predictive process monitoring
Process mining · Machine learning

1 Introduction

One goal of *process mining* [1] is to support organizations in identifying deviations from the expected behavior during process execution. Here, *conformance checking* techniques, e.g., [2,3], analyze, if the current state of a process instance in terms of its *event trace* conforms to the normative behavior. Complementary, *predictive process monitoring* is concerned with anticipating the future development of an instance [4] and focuses on (i) estimating the remaining execution time [5,6]; (ii) determining the next events [6–8]; (iii) measuring the risk associated with possible paths to completion [9,10]; and (iv) predicting the outcome of an instance [11–14].

In this work, we primarily focus on the latter category where *prediction models* are derived from historical data which contains event traces of past instances,

J. Mendling and H. Mouratidis (Eds.): CAiSE Forum 2018, LNBIP 317, pp. 163–181, 2018.
https://doi.org/10.1007/978-3-319-92901-9_15

labeled with the classes. Those classes e.g., indicate whether a business constraint was met [11,15], or describe the quality level achieved by the instance [12]. In prior work, a general approach to implementing the prediction is to (i) transform each trace from the historical data into a feature vector; (ii) use standard machine learning algorithms to train a model on the feature vectors and the associated outcome; (iii) represent ongoing traces as features vectors; and (iv) apply the model to obtain a prediction at runtime.

In this context, our work is motivated by the observation that prior work is largely limited to reporting performance metrics obtained on some real-world datasets (see Sect. 2) and to interpreting the results as evidence for the quality of the proposed approach. In the machine learning community, this methodology has long been criticized [17–19], because without further investigation it is unclear whether the performance is due to the discovery of higher level concepts or chance. A recent example in this regard is the study by Jo and Bengio [20]. While convolutional neural networks achieved a high performance for image recognition on various datasets, the study presented evidence that this high performance is partly due to superficial hints in the images which were present in the (training and test) data, but not because the networks derive higher level concepts to describe and recognize objects. Based on this criticism and related empirical evidence, we argue that more emphasis must be put on understanding the circumstances under which certain strategies perform well or poorly, following e.g., [16].

While data from real-world scenarios is certainly important to evaluate and compare the techniques' performance under realistic circumstances, it is often hard to understand to which degree the data enables the learning of a reliable prediction model and to assess whether the techniques learn the correct relationships from the data. In contrast, the generation of synthetic data allows researchers to exert full control over the variables that influence the outcome of a process instance. We therefore discuss and demonstrate how the use of synthetic data as a supplement for real-world data can yield additional insights. In this context and without the loss of generalizability, we focus on the sub-problem of *classifying* traces and only consider categorical outcome variables. Furthermore, we limit our work to *discrete event behavior*, i.e., the ordering of events in terms of sequences of event identifiers. While events can be associated with more information, the ordering of the events is the main dimension that separates predictive process monitoring from other prediction problems. In summary, our contributions are:

1. We present and discuss an approach to generate synthetic data that enables the establishment of an objective baseline, which in turn can be used to gain a more comprehensive understanding of how predictive process mining techniques work.
2. We use this approach to examine two aspects of the classification of traces.
   (a) We analyze different feature encodings for the representation of discrete event behavior and show that our *sub-trace encoding* is better suited for classification of completed traces than encodings from prior work. This

experiment also demonstrates that synthetic data can facilitate the understanding of how approaches pick up relationships between trace properties and outcomes.

(b) We additionally compare two approaches for the classification of ongoing traces. The *local prediction* approach, which works by training a set of classifiers for pre-defined prefix lengths, has often been used in prior work [11,13–15]. We demonstrate that it bears the risk of increasing the importance of irrelevant features. In contrast, a *global prediction* model that relies on a single classifier trained on the completed traces yields a higher reliability. This comparison emphasizes that solely relying on performance measures can be misleading.

The remainder of the paper is organized as follows. Section 2 summarizes related work and Sect. 3 introduces basic definitions. In Sect. 4, we subsequently discuss the process of generating synthetic data and the use of the data for analyzing predictive process mining techniques. Next, Sect. 5 is concerned with comparing different feature encodings. After that, we analyze the local prediction approach in Sect. 6 and the global prediction approach in Sect. 7. Finally, Sect. 8 concludes the paper.

2 Related Work

In the field of *process mining*, comparing event logs with normative process models (i.e. conformance checking [2,3]) or comparing event logs with other event logs (e.g. [21,22]) are common themes. Most of these techniques, however, are offline and focus on full behavior (i.e., partial executions are not taken into account) for the purpose of assessing the differences between normative behavior (represented by models or event logs) and observed behavior (event logs), but not for immediate classification. Partial traces have been analyzed for the purpose of conformance checking (e.g. [23,24]), but these approaches are based on trace alignment, where each event mismatch is taken into account individually and is, therefore, less precise and not complete [3]. In earlier work, we discussed real-time binary conformance checking [25].

The area of *predictive business process monitoring* [4,11] is concerned with forecasting how business process instances will unfold until completion. A first set of approaches aims to predict the next events for a prefix. For example, [6–8] evaluate different neural network architectures for this task. Other approaches focus on estimating the remaining time of a business process instance [5,6,26,27] or until the next event [6,28]. The evaluation of risks arising during execution and threatening the successful completion are studied in [9,10,29].

In the context of this paper, the most important category is the prediction of the outcome of a business process instance. The approaches in [11,13–15,30,31] are what we refer to as *local prediction* approaches, which train prediction models at predefined positions, e.g., at prefixes of lengths 3, 4, 5, etc. As such, they rely on training sets where the outcome is given for the completed traces.

When learning a prediction model at a certain position, the completed traces are reduced to the events that occurred up until this position. We present a *global prediction* approach in [12] where one prediction model is trained on the original training set and applied to all possible partial traces.

Complementary to these approaches, we here provide an analysis which reveals that the local prediction approach is prone to yield unreliable classifiers. Additionally, we demonstrate that global prediction approaches can overcome this limitation. To this end, we discuss an approach that in contrast to [12] also achieves good results for unstructured processes. Moreover, while all the mentioned works limit their evaluation to the presentation of accuracy measures or similar effectiveness metrics, we here additionally study the reliability based on synthetic data that is associated with an objective ground truth. In this context, Metzger and Flöcker [15] studied the filtering of prediction results based on confidence levels. Yet, they relied on a local prediction approach and measured reliability based on the outcome of the prediction model. In contrast, we here assess the reliability with regard to an objective ground truth.

3 Preliminaries

Events and Traces. The execution of a process instance results in *events* that provide information on the current state of the instance. For example, during the processing of a sales order, events might indicate that the order is confirmed, or that the goods were shipped.[1] To capture this information, the observed events are recorded as *traces* in which the events are ordered with respect to the time at which they were observed and described in terms of labels. Events can also be annotated with further information, e.g., with timestamps, resource identifiers, or data that is processed – but we focus solely on the information in a trace that concerns the ordering of events. As mentioned in the introduction, our work can be supplemented with additional features for additional dimensions. A set of traces is grouped in an *event log*.

Definition 1 (Event log, Trace). *Let L be an event log over the set of labels $\mathscr{L}$. Let E be a set of event occurrences and $\lambda : E \to \mathscr{L}$ a labeling function. An event trace $\theta \in L$ is defined in terms of an order $i \in [1, n]$ and a set of events $E_\theta \subseteq E$ with $|E_\theta| = n$ such that $\theta = \langle \lambda(e_1), \lambda(e_2), \ldots, \lambda(e_n) \rangle$. Moreover, given two events $e_i, e_j \in E_\theta$ with $i < j$, we write $e_i \triangleright e_j$ or say that event e_j* follows *e_i.*

During execution, the current state of an instance is given by the events that already occurred. While a trace captures the information of a completed instance, we refer to the sequence of the first l events of an instance as a *prefix*.

[1] The notion of an event used throughout this paper assumes that it corresponds to some activity in an underlying process. System-level logs, for instance, may contain events with some variation in the labels. We assume that, if present, any such variation has been removed in a preprocessing step, and that events belonging to the same activity have the same label.

Definition 2 (Prefix). *Given an event trace* $\theta = \langle \lambda(e_1), \lambda(e_2), \ldots, \lambda(e_n) \rangle$*, a prefix of this trace with length* $l \in \mathbb{N}$ *is defined as*

$$\theta_l := \begin{cases} \langle\rangle & if\ l = 0 \\ \langle \lambda(e_1), \ldots, \lambda(e_l) \rangle & if\ 0 < l \leq n \\ \langle \lambda(e_1), \ldots, \lambda(e_n) \rangle & otherwise \end{cases}$$

For a training log L, we also define the set of all prefixes in this log as $L_{pre} = \{\theta_l | \theta \in L \wedge 1 \leq l \leq |\theta|\}$.

Classification and Features. For each completed trace, the outcome might be given in terms of one of m behavioral classes, where the function $cl : L \rightarrow C$ with $C = \{c_1, \ldots, c_m\}$ represents the classification of the traces. The goal is to learn this classification and to derive a *prediction function* $pr : L_{pre} \rightarrow C$ that can be used to classify traces as well as prefixes. It is important to note that the prediction function is trained on a mapping from the traces to the classes, but allows for the classification of traces and prefixes.

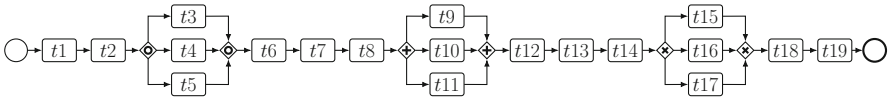
To support the learning of such prediction functions, traces and prefixes are typically encoded using features. More specifically, we are interested in *feature encoding* functions $\phi : L_{pre} \rightarrow \mathbf{F}$ where $\mathbf{F} = (F_1, \ldots, F_n)$ is a vector of n *feature variables*. Thus, prefixes and traces are no longer represented as a sequence of event labels, but as a *feature vector* $\mathbf{f} = (f_1, \ldots, f_n)$ where each element f_i takes a value from the *domain* D_i of the respective feature variable F_i. By relying on a certain feature encoding ϕ, the argument of the prediction function $pr : L_{pre} \rightarrow C$ changes and the function $pr : \phi(L_{pre}) \rightarrow C$ now classifies traces based on feature vectors. Moreover, the learning of such a prediction function is carried out in two steps. The first (optional) step is *feature learning:* given a set of training traces L, the feature encoding function ϕ is learned and each training trace $\theta \in L$ is encoded as a feature vector based on the learned encoding function $\mathbf{f} = \phi(\theta)$. After that, a *prediction model* is learned based on the feature vectors. To classify an observed prefix θ_l^r at runtime, it is first encoded as a feature vector $\mathbf{f}^r = \phi(\theta_l^r)$, which is subsequently passed to the prediction model to obtain the classification $c^r = pr(\mathbf{f}^r)$.

4 Generating Synthetic Data

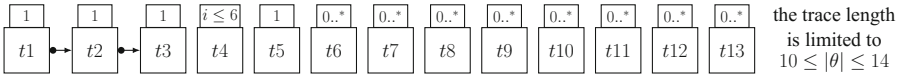
Subsequently, we outline how we generated and used data in our experiments, and point to the aspects that allowed us to gain a deeper understanding of how approaches work. We also made the synthetic data from our experiments publicly available[2].

Generation. To represent a broad variety of scenarios, we generated two datasets. Each dataset is based on one process model. These models are oriented towards the distinction between "Lasagna" and "Spaghetti" processes [1].

[2] https://doi.org/10.4225/08/5acff82350c74.



(a) Low variability model (modeled in BPMN)



(b) High variability model (modeled in DECLARE[2] [32])

Fig. 1. Models with normative behavior (We introduced the notation $i \leq x$ with $x \in \mathbb{N}$ to indicate that an event has to be among the first x events.)

The *low variance* ("Lasagna") model is on one end of the structuredness spectrum. It imposes strict ordering rules, is mostly sequential, and describes a small set of distinct traces. On the contrary, the *high variance* ("Spaghetti") model allows for a more flexible process execution and enforces only a few constraints resulting in a large amount of possible distinct traces. These two models define the respective normative behavior and are presented in Fig. 1.

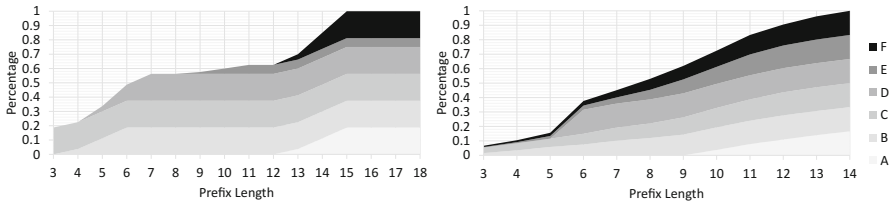
Based on these models, we generate traces and assign them to different behavioral classes by applying a procedure that is inspired by *mutation testing* [33] – a well-known technique for measuring the quality of software test suites. That is, we defined five different mutation operators, which we applied independently to the basic model in order to obtain one model per operator. Table 1 presents the operator descriptions, their class labels (class label A is used for the normative model), and how they were applied to the two models. While these operators represent basic differences in the discrete event behavior, it is conceivable to define more complex operators that also modify other event attributes e.g., the distribution of processing time.

Table 1. Mutation operators and their application to the models

Description	Specific application to models		Class label
	Low variability	High variability	
Change the order of activities	Swap $t6$ and $t8$	Swap $t1$ and $t3$	B
Insert a new activity	Add $t20$ after $t2$	Add $t14$ somewhere	C
Move an activity	Move $t7$ after $t12$	Move $t4$ to $i > 6$	D
Remove an activity	Delete $t9$	Delete $t2$	E
Execute an activity twice	Insert $t14$ after $t14$	Let $t5$ occur twice	F

Subsequently, we simulated the models to obtain a set of distinct traces. For the low variance models, we considered all of the distinct traces that each model defines. This resulted in 90 traces for class E and 270 for each of the other classes. For the high variance model, where the amount of possible traces is very large, we generated 250 distinct traces per class. In the traces, the execution of an activity is represented by a single event whose label is identical to the activity label. Note that, in addition to the structuredness, both datasets also differ with regard to the coverage of the training data relative to the trace space.

Prefix Classifiability. The advantage of the synthetic data is that we do not only know the outcome (i.e., the class) for the entire trace, but we can use the definition of the mutation operators to setup a *classifiability oracle* which constitutes the objective ground truth. This oracle can tell us for each prefix, if the outcome can already be predicted at this point, and if that is the case, what the outcome is. For example, consider the following prefixes of length 3 on the low variability dataset: $\theta_3 = \langle t1, t2, t20 \rangle$ and $\theta'_3 = \langle t1, t2, t3 \rangle$. For θ_3 it can clearly be determined that it belongs to class C, as it contains the event $t20$ which only occurs in traces of this class. Similarly, we can rule out that θ'_3 belongs to C, as it does not contain $t20$ which needed to appear before $t3$. Yet, θ'_3 is still *unclassifiable*, as all remaining classes are still possible for θ'_3. Figure 2 depicts the percentage of classifiable prefixes per class, prefix length, and dataset.



(a) Low variability (b) High variability

Fig. 2. Percentage of classifiable traces per prefix length and class

Observation 1: In our datasets, the percentage of classifiable prefixes overall as well as per class increases monotonically with a growing prefix length.

In addition to this observation, Fig. 2 also shows that on the low variability dataset there are windows of prefix lengths in which traces belonging to a certain class become classifiable, e.g., class D becomes classifiable for prefix lengths between 4 and 7. Yet, the high variability dataset demonstrates that this does not necessarily need to happen. Due to a greater flexibility in the ordering of events, these windows cover the entire range of prefix lengths.

The ground truth of the classification oracle enables us to go beyond an accuracy assessment and to analyze the reliability of correct predictions. In particular,

we can check, if a correctly classified prefix could actually already be classified, or if the classifier was lucky.

Evaluation Metrics. Each set is used once as a test set for the functions that were trained on the union of the remaining trace sets. When evaluating different combinations of feature encodings and prediction functions, we perform a 5-fold cross validation where the event log is split into 5 equally sized sets of traces; training is performed on 4 of the 5 sets, testing against the remaining set. The overall effectiveness reported is the average over all trace sets. As the datasets only contain distinct traces, our evaluation is based on out-of-sample forecasts. That is, a trace is either used for training or testing and it is hence not sufficient for an approach to memorize the data. Instead, it needs to infer higher level relationships and concepts from the data to achieve a high performance.

In the evaluation, we measure the effectiveness of a prediction function for a certain position l (number of events after which we want to predict the outcome) with regard to different confidence levels. In general, the higher the confidence the more certain is the prediction function. At this point, we abstract from the specific confidence measurement and provide more details in the context of the experiments. Given a test trace set T, we obtain the prediction along with the confidence for all prefixes of length l. Next, we determine the set T_{cf} which contains all prefixes where the confidence is equal to or higher than the fixed confidence level. Based on the set of correctly predicted traces $T_{co} \subseteq T_{cf}$, the *precision* $pc = \frac{|T_{co}|}{|T_{cf}|}$ is the ratio of correctly classified and confidently classified prefixes, while the recall $rec = \frac{|T_{co}|}{|T|}$ is the overall share of correctly classified prefixes. In addition to these measures, we also take advantage of the classification oracle and investigate if it supports the correct predictions – i.e., if the predictions are justified given the observed prefix. In particular, we determine the set T_{cl}, which comprises all prefixes in T_{co} that were actually classifiable. This is expressed as the *justification score* $js = \frac{|T_{cl}|}{|T_{co}|}$, i.e., the share of the correct predictions that are justified according to the classification oracle.

5 Feature Encodings

In this section, we compare and analyze five feature encodings, which include four encodings from prior work:

Event occurrence (EvOcc) [13]**:** Each distinct event label is represented as a feature variable. In the feature vector for a trace, an element is set to 1 if the respective label occurs in the trace, and to 0 otherwise.

Event frequency (EvFreq) [13]**:** Again, there is one feature variable per distinct event label. For a given trace, the vector elements contain the number of occurrences of the respective label in the trace.

Simple index (Index) [13]**:** Given the maximal trace length l_m, there is one feature variable for each position $1 \leq p \leq l_m$. For a trace, each vector element

contains the event label from the respective position. If a trace is shorter than the maximal trace length $|\theta| < l_m$, the elements for positions $p > |\theta|$ are set to a default label to indicate the absence of a label.

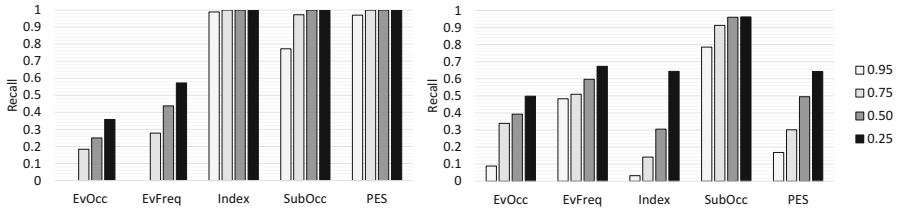
Prime event structures (PES) [12]**:** For each behavioral class, a real-valued feature variable is introduced. Here, a variable represents a matching score that indicates the degree to which a trace resembles the behavior of the respective class. To this end, for each class the according traces from the training log are transformed into a PES that captures the behavioral relations between the event labels, along with the execution frequencies for multiple exclusive branches. Subsequently, a weighted matching score is obtained for each PES, based on the identified behavioral differences with the trace and the branching frequencies of the matched trace in that PES.

Sub-trace occurrence (SubOcc): Here, we consider binary feature variables that indicate whether a sub-trace occurred in a trace. A sub-trace ς is a sequence of labels $\varsigma = \langle \lambda_1, \ldots, \lambda_o \rangle$ and we say that a sub-trace occurs in a trace $\varsigma \in \theta$, if all labels in ς occur in the same order as in θ, i.e., if $\forall 1 \leq i < j \leq |\varsigma| : \exists e_k, e_l \in E_\theta : \varsigma_i = \lambda(e_k) \wedge \varsigma_j = \lambda(e_l) \wedge e_k \triangleright e_l$. To establish the feature variables, we derive the set of sub-traces from the training traces by applying sequence mining and in particular the SPADE algorithm [34]. We control the identification of relevant sub-traces by two parameters. First, we only consider sub-traces with a support of at least .05, i.e., the sub-traces have to occur in at least 5% of the training prefixes. Second, we restrict the length of the sub-traces to 3 which we found to be a sufficient length for our datasets. For each sub-trace in the mined set, there is one feature variable. For a trace, a vector element is set to 1 if the corresponding sub-trace occurs in the trace, and to 0 otherwise. This encoding shares similarities with declarative process discovery algorithms like DECLARE [32] that aim to extract general execution constraints. Such constraints specify, for instance, that two activities are sequential, or activities are executed in a loop, etc. However, we here abstain from this generalization and focus on the concrete behavior. As such we can, for example, capture how often a loop is executed, and similar differentiations which may be important to distinguish classes.

We evaluated these encodings by adopting the experimental setup from [13, 14] and chose Random Forest[3] [35] to learn the prediction functions. At heart, a Random Forest trains a set of decision trees of size *ntree*. Each of these trees is trained on a random sample drawn from the training dataset with replacement. Moreover, not all features, but only a randomly drawn sub-set of features of size *mtry* is considered when a decision node is induced. To discriminate an object, each tree classifies it and the overall result is the class that was yielded by the most trees. We interpret the percentage of the trees that voted for this class as the confidence in the result. Additionally, we followed guidelines from the literature [35,36] and focused on optimizing *mtry*, while we used a default value of

[3] We use the default implementation provided by the `randomForest` package for `R` (https://cran.r-project.org/package=randomForest, accessed: 15/11/2017).

500 for *ntree*. That is, in each iteration of the 5-fold cross-validation that we use to evaluate the effectiveness, we performed another 5-fold cross validation with 3 repetitions on the training set to evaluate different values for *mtry* and chose the one with the highest accuracy to obtain the final Random Forest[4]. Finally, we determined the recall for all encodings and for four different confidence levels (95%, 75%, 50%, 25%). We do not consider the precision and justification score here, as we are evaluating the feature encodings over the completed traces and can thus be sure that there is evidence that supports the classification of a trace. Figure 3 summarizes the recall for each encoding.



(a) Low variability (b) High variability

Fig. 3. Accuracy of the different encodings per confidence level

Observation 2: The sub-trace occurrence yields the highest accuracy.

On both datasets, the EvOcc and EvFreq encodings yield low accuracy values, as they discard information regarding the ordering of events and hence struggle to provide meaningful features in cases where the event ordering is relevant for classification. The Index encoding achieves a perfect accuracy on the low variability dataset, but performs poorly on the high variability dataset because it does not enable a classifier to infer that events need to be positioned relative to each other. For example, consider that event $t2$ has to eventually follow event $t1$, in order for a trace to belong to a certain class. Here, the trees in the Random Forest need to introduce one rule per possible combination of absolute positions that satisfy this characteristic, e.g., $\lambda_1 = t1 \wedge \lambda_2 = t2$, $\lambda_1 = t1 \wedge \lambda_3 = t2 \ldots$, $\lambda_2 = t1 \wedge \lambda_3 = t2, \ldots$. Thus, the higher the variance in the events' positions and the trace length, the more data is needed to infer all these rules. The PES encoding suffers from the same problem, because the approach as presented in [12] focuses on differences in specific positions in the trace and is, as such, more suited for structured processes as it fails to provide a more flexible representation of the behavior in the presence of variability. Finally, the SubOcc provides the richest features, because it also includes relative relationships between events and thus allows the classifier to consider these relationships independent of the

[4] In particular, we applied the random search strategy from the `caret` package for `R` (https://cran.r-project.org/package=caret, accessed: 15/11/2017).

absolute positioning. For low confidence intervals, it achieves a recall of 1 on the low and of .96 on the high variability dataset. Yet, the recall drops with an increase in the confidence level on both datasets. This is due to the high number of features and the random sampling of traces as well as features in the Random Forest, which can result in some irrelevant features distorting the classification.

Conclusion 1: Encoding relative ordering relationships between events as features can improve the effectiveness of prediction models.

6 Local Prediction

Next, we examine the local prediction approach, where one prediction function pr_l is trained for each prefix length l that was (manually) determined beforehand. To induce such a function from the training data, all traces in the dataset are reduced to prefixes of length l. At runtime, the prediction function that matches the number of observed events is used for classification.

Table 2. Illustrative example training log

Class	Traces			
A	$\langle t1, t2, t3, t4, t5\rangle$	$\langle t1, t3, t2, t4, t5\rangle$	$\langle t1, t3, t2, t4, t5\rangle$	$\langle t1, t3, t2, t4, t5\rangle$
B	$\langle t1, t2, t3, t4, t5, t6\rangle$	$\langle t1, t3, t2, t4, t5, t6\rangle$	$\langle t1, t2, t3, t4, t6, t5\rangle$	$\langle t1, t6, t2, t3, t4, t5\rangle$

For illustration, Table 2 depicts an example training log that contains eight traces which are assigned to one of two classes. In class A we observe that $t1$ always occurs first, followed by $t2$ and $t3$ in any order, and ending with event $t4$ followed by $t5$. Class B shows the same behavior, but there is an additional event $t6$ which seems to occur at an arbitrary position. Clearly, the existence of $t6$ is a perfectly discriminating characteristic. Now, consider that we learn a prediction model at position 4 and only consider the prefixes of length 4. Then, $t6$ only occurs in one out of four prefixes in class B, but the sub-trace $\langle t2, t3\rangle$ occurs three times in class B and only once in class A. Thus, a classifier would deem this sub-trace more important for the classification than the existence of $t6$. This shows that by removing the suffixes, we change the importance of features and in the worst case favor irrelevant over relevant features.

To further examine this risk, we evaluate the local prediction approach. That means, for each prefix length greater than 2, we repeat the procedure from Sect. 5 and evaluate the Random Forest in combination with the SubOcc encoding. In addition to the *classifiability oracle* (Oracle Cl.), we also use the *classifiability oracle with a default option* (Oracle Def.) as a baseline: for unclassifiable prefixes this oracle predicts normative class A in the absence of evidence to the contrary. That may be desirable, for instance, in settings where the normative case is much more frequent than any deviation. We then compare the Oracles' recall values

to the recall values of the classifiers, to see how close the classifiers come to the objective ground truth. Figure 4 shows the effectiveness yielded at the different prefix lengths for four confidence levels.

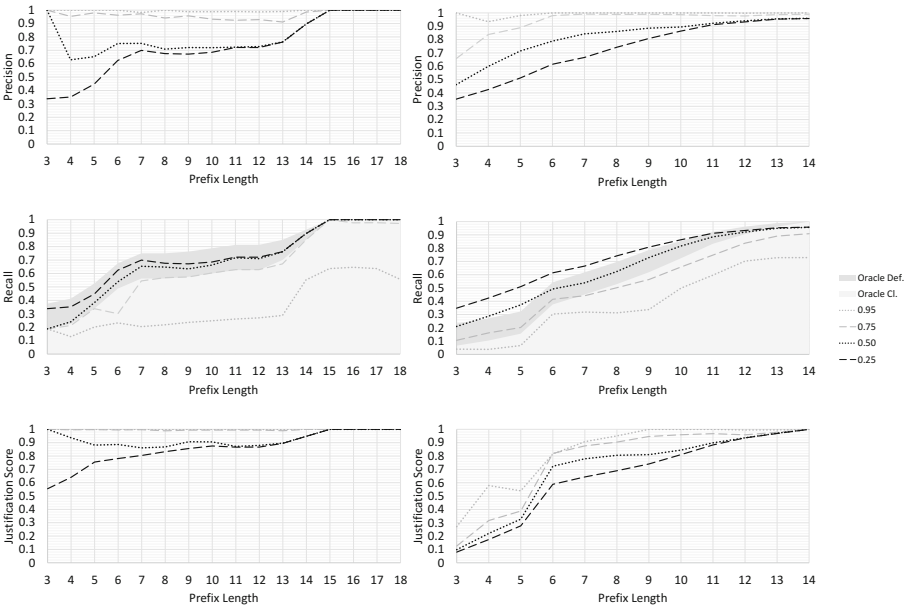


Fig. 4. Prefix length vs. precision, recall and justifiability score on the low (left) and high variability datasets (right)

Observation 3: Raising the confidence (i) increases the precision and justification score; (ii) decreases the recall; but (iii) does not ensure that all correct predictions are justified.

The positive relation between the confidence and the precision as well as the justification score and the negative relation between the confidence and the recall are expected, as a higher confidence level requires more agreement on the result. Interestingly, increasing the confidence does not guarantee a high justification score. In fact, independent of the confidence level the justification score is quite low at the beginning and only approaches 1 at high prefix lengths on the high variability dataset. A similar effect can be observed on the low variability dataset. This implies that for a large portion of correct predictions the classifiers rely on irrelevant features and exploit spurious correlations.

Observation 4: The classifiers' recall can exceed that of the oracles.

On the high variability dataset the classifiers outperform the oracles for low confidence levels and prefix lengths. A performance better than Oracle Cl. can be explained by relying on a default option. However, the improved accuracy in comparison to Oracle Def. substantiates that the classifiers are exploiting spurious correlations.

Observation 5: The recall might decrease with a growing prefix length, even if the justification score is not decreasing.

For example, on the low variability dataset the accuracy drops after position 7 for confidence levels of 25% and 50% or at position 18 for a confidence of 95%. In both cases, the justification score remains stable or increases. A decreasing effectiveness for higher prefix lengths can also be observed in prior work, e.g., in the evaluation in [13]. This indicates that the classification rules at a certain prefix length are in part invalidated at a higher prefix length and that the importance of some irrelevant features decreases with a growing prefix length.

Conclusion 2: Our experiments provide clear evidence that the local prediction approach can provide predictions that are not justified by the data observed, which may be due to an overly high reliance of irrelevant features. This negatively impacts the reliability of the prediction models and thus the practical applicability.

7 Global Prediction

In this section, we test our hypothesis that a global prediction approach can overcome the problems of the local prediction approach. To this end, we first introduce our specific implementation of the global prediction approach and afterwards present the evaluation results.

In contrast to the local prediction approach, we only learn one classifier in the global prediction approach, which is then used to classify any prefix. Moreover, this classifier is trained solely on the completed traces and their associated outcomes. We utilize rule-based classification and in particular a variant of the RIPPER algorithm [37], since a direct application of classification approaches like the Random Forest does not yield high-quality results in this setting (as explained below). That is, the classifier consists of a set of rules of the form `IF condition THEN` c where `condition` defines properties that a prefix needs to possess in order to belong to class c. Given a training log, a set of rules is established based on the following two step procedure.

First, we define the set of elementary conditions that are considered during rule induction. To this end, we start with mining all sub-traces as explained in Sect. 5. For each sub-trace ς, there is one *sub-trace condition* that evaluates to true, if the sub-trace is part of the prefix. These conditions enable us to check whether an event occurs at all or relative to other events (i.e., before or after them). However, sometimes it is the absence of events that is representative for

a certain behavioral class. Therefore, for each ς we introduce *absence conditions*. Such a condition evaluates to true if the prefix reached a length and the sub-trace did not occur beforehand. To this end, we iterate over all event labels and for each event label λ we determine its preset, i.e., the set of labels that solely occur before λ. Next, we group all events that have the same preset. In the preset comparison, we remove those labels from the presets that we are currently comparing. We then yield the set of absence conditions by combining each of the obtained groups with each sub-trace. In the evaluation of an absence condition, we check if any of the position labels is part of the prefix. If that is the case, we check if the sub-trace does not occur before the first label in the prefix. Lastly, we cluster perfectly correlated conditions into an elementary or-condition that evaluates to true if any of the sub-conditions are satisfied. Two conditions are perfectly correlated, if for all training traces the conditions evaluate to the same value. The reason to cluster such conditions is that we want to ensure that we decide on the class as early as possible. For example, consider the case where an event $t1$ needs to occur before two events $t2$ and $t3$ which always occur together but in any order. For the classification of completed traces it would not matter, if we consider $\langle t1, t2\rangle$ or $\langle t1, t3\rangle$ for representing this correlation. Yet, when we classify running instances and only consider $\langle t1, t2\rangle$, we will lately classify prefixes where $t3$ occurs before $t2$. This is where the Random Forest fails: it just picks one of the conditions at random, which can severely delay the classification.

Second, in the rule induction we process each behavioral class independently. For a given class c, we first extract the set of traces T_c from the training log that belong to c and the set of traces $T_{\not c}$ that do not belong to c. We then iteratively learn one rule and after that remove all traces from T_c for which the rule is satisfied. We repeat this step until all traces in T_c are covered by at least one rule.

In each iteration, we learn a rule of the form $\texttt{condition}_1$ `AND` ... `AND` $\texttt{condition}_n$ based on a beam search. That is, we first select the elementary condition that best separates the traces in T_c from those in $T_{\not c}$. Next, we select another elementary condition and combine it with the first elementary condition, if it improves the discriminative power of the rule. We keep adding elementary conditions until the discriminative power of the rule is not improved any further. To measure the discriminative power of a condition we employ the *FOIL gain* [38], which considers not only the overall amount of traces in T_c and $T_{\not c}$ for which the condition is satisfied, but also the ratio of such rules in both T_c and $T_{\not c}$.

At runtime, we classify a prefix or trace by selecting all rules that are satisfied for the prefix. If no rule is satisfied, we do not predict an outcome. Otherwise, we select the most confident rule and return the respective class. The confidence is the percentage of training traces that belong to the rules' class among the set of training traces for which the rule condition is satisfied. Note that in the evaluation we do not obtain a class if the confidence of the selected rule is too low. Figure 5 shows the evaluation results.

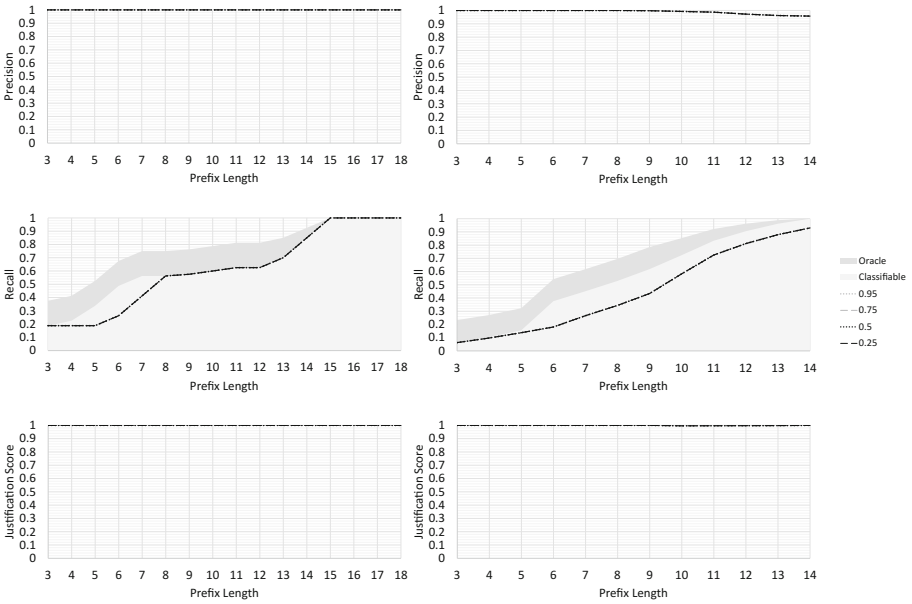


Fig. 5. Prefix length vs. precision, recall and justifiability score on the low (left) and high variability datasets (right). Note that the same results were yielded for all confidence levels.

Observation 6: The confidence level does not impact the effectiveness and reliability of the global prediction approach.

Contrary to the local prediction approach, the confidence level has no impact on the effectiveness. This indicates that global prediction models are reliable indicators which exploit relationships between the relevant features and the outcome.

Observation 7: The precision and the justification score are virtually perfect for all prefix lengths.

In stark contrast to the local prediction approach, the precision and the justification score are 1 for all prefixes, with one exception on the high variability dataset where the precision is slightly lower for prefix lengths greater than 10. This confirms that the results of the global approach are justified.

Observation 8: The recall development is similar to, but at points lower than, that of the classifiability oracle.

The lower recall is due to the simplistic nature of the rule induction algorithm, the relative nature of the sub-trace encoding, which does not consider the absolute positioning of events and (on the high variability dataset)

the high variance in the events positions' due to which there is an extensive amount of possible conditions. Yet, over the completed traces the global approach yields a performance that is virtually equal to the local prediction approach. For smaller prefix lengths, the recall falls behind that of the local prediction approach, which is due to the high percentage of unjustified predictions that the local prediction approach yields. In summary, we thus conclude:

Conclusion 3: Global prediction approaches can avoid the risk of increasing the importance of irrelevant features and hence yield reliable results.

8 Discussion

In this paper, we examined the prediction of the outcome of process instances during execution, especially classification of discrete event behavior. With regard to the design of such prediction approaches, the findings of our systematic evaluation are threefold. First, encodings for representing running process instances should consider the relative positioning of events, as shown by the effectiveness of the sub-trace encoding over completed traces. Second, training prediction models at fixed positions entails the risk of establishing spurious correlations between irrelevant features and the outcome. Third, in order to avoid that risk, the prediction models should be learned based on the *completed traces rather than* reducing the traces to *prefixes*.

In addition, we demonstrated that the evaluation of predictive process monitoring approaches can benefit from synthetic data. Admittedly, the most critical threat to the validity of the experiments based on synthetic datasets pertains the external and ecological validity. However, we argue that synthetic data enables a thorough analysis of the underlying principles, which goes beyond the common effectiveness-driven evaluation in prior work. To repeat such an analysis with a real-world case study would require an enormous effort to control all influential factors and fully understand the reasons for differences. This is demonstrated especially by the negative results regarding local classification. Here, our results can be seen as *counter-examples*, which without doubt pinpoint a real limitation of the local prediction approach.

For future research, we propose two main directions. First, although our results support the usage of the sub-trace encoding along with a global prediction approach, they also suggest that the presented approach can be improved further. Second, research and practice would benefit from a more elaborate set of tools and methods that support the performance and reliability assessment, especially on real-world data.

References

1. van der Aalst, W.M.P.: Process Mining: Discovery, Conformance and Enhancement of Business Processes. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19345-3
2. Rozinat, A., van der Aalst, W.M.: Conformance checking of processes based on monitoring real behavior. Inf. Syst. **33**(1), 64–95 (2008)
3. García-Bañuelos, L., van Beest, N.R.T.P., Dumas, M., La Rosa, M., Mertens, W.: Complete and interpretable conformance checking of business processes. IEEE Trans. Softw. Eng. **44**(3), 262–290 (2018)
4. Dumas, M., Maggi, F.M.: Enabling process innovation via deviance mining and predictive monitoring. In: vom Brocke, J., Schmiedel, T. (eds.) BPM - Driving Innovation in a Digital World. MP, pp. 145–154. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-14430-6_10
5. 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). https://doi.org/10.1007/978-3-642-45005-1_27
6. Tax, N., Verenich, I., La Rosa, M., Dumas, M.: Predictive business process monitoring with LSTM neural networks. In: Dubois, E., Pohl, K. (eds.) CAiSE 2017. LNCS, vol. 10253, pp. 477–492. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-59536-8_30
7. Evermann, J., Rehse, J.R., Fettke, P.: Predicting process behaviour using deep learning. Decis. Support Syst. **100**(August), 129–140 (2017)
8. Di Francescomarino, C., Ghidini, C., Maggi, F.M., Petrucci, G., Yeshchenko, A.: An eye into the future: leveraging a-priori knowledge in predictive business process monitoring. In: Carmona, J., Engels, G., Kumar, A. (eds.) BPM 2017. LNCS, vol. 10445, pp. 252–268. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-65000-5_15
9. 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). https://doi.org/10.1007/978-3-642-38709-8_8
10. Conforti, R., Fink, S., Manderscheid, J., Röglinger, M.: PRISM – a predictive risk monitoring approach for business processes. In: La Rosa, M., Loos, P., Pastor, O. (eds.) BPM 2016. LNCS, vol. 9850, pp. 383–400. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45348-4_22
11. 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, Cham (2014). https://doi.org/10.1007/978-3-319-07881-6_31
12. van Beest, N.R.T.P., Weber, I.: Behavioral classification of business process executions at runtime. In: Dumas, M., Fantinato, M. (eds.) BPM 2016. LNBIP, vol. 281, pp. 339–353. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-58457-7_25
13. Leontjeva, A., Conforti, R., Di Francescomarino, C., Dumas, M., Maggi, F.M.: Complex symbolic sequence encodings for predictive monitoring of business processes. In: Motahari-Nezhad, H.R., Recker, J., Weidlich, M. (eds.) BPM 2015. LNCS, vol. 9253, pp. 297–313. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-23063-4_21

14. Teinemaa, I., Dumas, M., Maggi, F.M., Di Francescomarino, C.: Predictive business process monitoring with structured and unstructured data. In: La Rosa, M., Loos, P., Pastor, O. (eds.) BPM 2016. LNCS, vol. 9850, pp. 401–417. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45348-4_23
15. Metzger, A., Föcker, F.: Predictive business process monitoring considering reliability estimates. In: Dubois, E., Pohl, K. (eds.) CAiSE 2017. LNCS, vol. 10253, pp. 445–460. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-59536-8_28
16. Aha, D.W.: Generalizing from case studies: a case study. In: ICML (1992)
17. Cohen, P.R., Jensen, D.: Overfitting explained. In: AISTATS (1997)
18. Salzberg, S.L.: On comparing classifiers: Pitfalls to avoid and a recommended approach. Data Min. Knowl. Discov. **1**(3), 317–328 (1997)
19. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. **7**, 1–30 (2006)
20. Jo, J., Bengio, Y.: Measuring the tendency of CNNs to learn surface statistical regularities. CoRR abs/1711.11561 (2017)
21. van Beest, N.R.T.P., Dumas, M., García-Bañuelos, L., La Rosa, M.: Log delta analysis: interpretable differencing of business process event logs. In: Motahari-Nezhad, H.R., Recker, J., Weidlich, M. (eds.) BPM 2015. LNCS, vol. 9253, pp. 386–405. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-23063-4_26
22. Maaradji, A., Dumas, M., La Rosa, M., Ostovar, A.: Fast and accurate business process drift detection. In: Motahari-Nezhad, H.R., Recker, J., Weidlich, M. (eds.) BPM 2015. LNCS, vol. 9253, pp. 406–422. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-23063-4_27
23. van Zelst, S.J., Bolt, A., Hassani, M., van Dongen, B.F., van der Aalst, W.M.: Online conformance checking: relating event streams to process models using prefix-alignments. Int. J. Data Sci. Anal. 1–16 (2017). https://doi.org/10.1007/s41060-017-0078-6
24. Burattin, A.: Online conformance checking for petri nets and event streams. In: BPM 2017 Demo Track (2017)
25. Weber, I., Rogge-Solti, A., Li, C., Mendling, J.: CCaaS: online conformance checking as a service. In: BPM, Demo Track (2015)
26. van der Aalst, W., Schonenberg, M., Song, M.: Time prediction based on process mining. Inf. Syst. **36**(2), 450–475 (2011)
27. Metzger, A., Leitner, P., Ivanovic, D., Schmieders, E., Franklin, R., Carro, M., Dustdar, S., Pohl, K.: Comparing and combining predictive business process monitoring techniques. IEEE Trans. SCM Syst. **45**(2), 276–290 (2015)
28. 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: Carmona, J., Engels, G., Kumar, A. (eds.) BPM 2017. LNCS, vol. 10445, pp. 306–323. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-65000-5_18
29. Pika, A., van der Aalst, W.M.P., Fidge, C.J., ter Hofstede, A.H.M., Wynn, M.T.: Predicting deadline transgressions using event logs. In: La Rosa, M., Soffer, P. (eds.) BPM 2012. LNBIP, vol. 132, pp. 211–216. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36285-9_22
30. Di Francescomarino, C., Dumas, M., Federici, M., Ghidini, C., Maggi, F.M., Rizzi, W.: Predictive business process monitoring framework with hyperparameter optimization. In: Nurcan, S., Soffer, P., Bajec, M., Eder, J. (eds.) CAiSE 2016. LNCS, vol. 9694, pp. 361–376. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-39696-5_22

31. Di Francescomarino, C., Dumas, M., Maggi, F.M., Teinemaa, I.: Clustering-based predictive process monitoring. IEEE Trans. Serv. Comput. (2016, in press)
32. Maggi, F.M., Mooij, A.J., van der Aalst, W.M.P.: User-guided discovery of declarative process models. In: CIDM (2011)
33. Jia, Y., Harman, M.: An analysis and survey of the development of mutation testing. IEEE TSE **37**(5), 649–678 (2011)
34. Zaki, M.J.: SPADE: an efficient algorithm for mining frequent sequences. Mach. Learn. **42**(1), 31–60 (2001)
35. Breiman, L.: Random forests. Mach. Learn. **45**(1), 5–32 (2001)
36. Hastie, T., Tibshirani, R., Friedman, J.: Random forests. The Elements of Statistical Learning. SSS, pp. 587–604. Springer, New York (2009). https://doi.org/10.1007/978-0-387-84858-7_15
37. Cohen, W.W.: Fast effective rule induction. In: ICML (1995)
38. Quinlan, J.: Learning logical definitions from relations. Mach. Learn. **5**(3), 239–266 (1990)