Remaining Time Prediction for Processes with Inter-Case Dynamics*

Mahsa Pourbafrani[1], Shreya Kar[1], Sebastian Kaiser[2], and Wil M. P. van der Aalst[1]

[1] Chair of Process and Data Science, RWTH Aachen University, Germany
{mahsa.bafrani,shreya.kar,wvdaalst}@pads.rwth-aachen.de
[2] LMU Munich, Germany, {sebastian.kaiser@stat.uni-muenchen.de}

Abstract. Process mining techniques use event data to describe business processes, where the provided insights are used for predicting processes' future states (*Predictive Process Monitoring*). *Remaining Time Prediction* of process instances is an important task in the field of Predictive Process Monitoring (PPM). Existing approaches have two key limitations in developing *Remaining Time Prediction Models* (RTM): (1) The features used for predictions lack process context, and the created models are black-boxes. (2) The process instances are considered to be in isolation, despite the fact that process states, e.g., the number of running instances, influence the remaining time of a single process instance. Recent approaches improve the quality of RTMs by utilizing process context related to *batching-at-end* inter-case dynamics in the process, e.g., using the time to batching as a feature. We propose an approach that decreases the previous approaches' reliance on user knowledge for discovering fine-grained process behavior. Furthermore, we enrich our RTMs with the extracted features for multiple performance patterns (caused by inter-case dynamics), which increases the interpretability of models. We assess our proposed remaining time prediction method using two real-world event logs. Incorporating the created inter-case features into RTMs results in more accurate and interpretable predictions.

Keywords: process mining, predictive process monitoring, remaining time prediction, inter-case dynamics behavior.

1 Introduction

Remaining time prediction approaches learn from historical process executions and build prediction models for running process instances, i.e., cases, based on the extracted features from the event data. Many approaches have been suggested to solve the remaining time prediction problem [17]. However, most proposed approaches have considerably high prediction errors. Based on [17], the best performing model using an LSTM neural network [10] showed a prediction

* Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC 2023 Internet of Production- Project ID: 390621612. We also thank the Alexander von Humboldt (AvH) Stiftung for supporting our research.

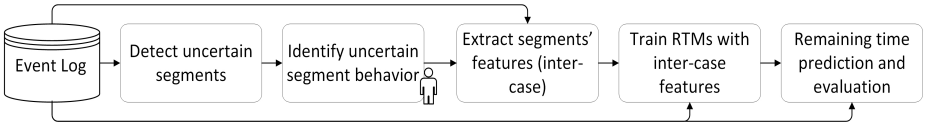

Fig. 1: Our proposed framework for inter-case-aware RTMs. Patterns are discovered after detecting uncertain segments, i.e., segments causing high prediction errors due to inter-case dynamics. RTMs are trained using the extracted features from the patterns within uncertain segments.

error of 178.4 days on average for the Road Traffic Management (RF) event log [9]. These approaches also only consider control-flow-related aspects of processes and individual case properties, i.e., intra-case properties, while making predictions [12]. A process also has other dimensions associated with it [13]. For instance, specific rules determining scheduling and assignment of limited resources, queuing mechanism, and decision logic in the process create inter-case dependencies within the performance of process instances. Moreover, most of the effort put into this research area has focused on applying new predictive modeling techniques, which create black-box prediction models. Considering inter-case along with intra-case process features in RTMs increases the explainability, interpretability, and accuracy of the prediction [8]. Therefore, we aim to improve the quality of RTMs and introduce more interpretability in the predictions. The accuracy of a RTM which is unaware of inter-case behavior is substantially impacted if cases in a process segment, i.e., a pair of related activities, are processed in a batch, First-In-First-Out (FIFO), or other patterns. The prediction accuracy decreases as a case passes through such segments indicating that RTM is uncertain about the underlying process behavior in such segments. We call these process segments uncertain segments. Therefore, recognizing all uncertain segments and translating their various inter-case patterns of process execution into features for training RTMs increases prediction quality.

In this paper, we present a three-step approach for developing inter-case dynamics aware RTMs: (1) Identifying process segments that cause high prediction errors due to inter-case dynamics, i.e., uncertain segments. (2) Discovering insights about the underlying patterns, e.g., *batching*, that leads to inter-case dependencies within the detected segments. (3) Transforming derived insights into features and incorporating them in RTMs to improve the quality of predictions. For instance, the waiting time for the batching in a segment is transformed into a feature and introduced into the RTM. We evaluate the prediction errors of RTMs without incorporating inter-case dependencies, such as batching behavior in a process segment, as shown in Figure 1, and identify uncertain segments that involve inter-case dynamics. We continue by extracting the features associated with the observed patterns in the uncertain segments.

We introduce preliminaries and the related work in Section 2. In Section 3, we present our main approach. We evaluate the approach in Section 4 using real event logs, and Section 5 concludes this work.

2 Preliminaries and Related Work

In this section, we introduce the necessary concepts and related work required to understand the approach presented in this paper.

2.1 Related Work

RTM approaches can be classified into three broad categories [17]. *Process aware approaches* make predictions using explicit process model representations such as transition system [1]. *Process agnostic approaches* typically use machine learning (ML) methods [14] to make predictions. Recent process agnostic approaches predominantly make use of sophisticated neural network architectures like LSTM [16] and explainable AI methods [5] to develop RTMs. *Hybrid approaches* like [11] combine capabilities of both categories by exploiting transition systems that are annotated using a machine learning algorithm. However, most approaches across all three categories only consider the intra-case perspective for predictions.

RTM approaches based on queuing models [15] and supervised learning [14] utilized the inter-case dimension in predictions. They create features on the basis of queuing theory like case priority and open cases of similar type. However, these approaches assume FIFO queuing behavior throughout the entire process. Two recent PPM approaches [3, 8] use performance spectra [2] to learn inter-case dynamics present in the process without any prior assumption. Denisov et al. [3] presented a novel approach to predict the aggregated performance of non-isolated cases that utilize performance-related features. Klijn et al. [8] presented a novel RTM approach that is aware of *batching-at-end* dynamics. In this paper, we extend the process agnostic RTM approach presented in [8] by considering inter-case dynamics caused by *non-batching*, *batching-at-start* patterns too. We use and improve the fine-grained error analysis technique proposed in [8] to identify inter-case dynamics by limiting manual intervention.

2.2 RTM Background

RTM approaches predict the remaining time to completion of an ongoing process instance, i.e., *case*, based on process execution data of completed cases. Process execution of a completed case is recorded as a non-empty sequence of events (e), i.e., $\sigma{=}\langle e_1, .., e_n\rangle$ or *trace*. An event log L is a set of completed traces. Let $\mathcal{A}, \mathcal{T}, \mathcal{E}$ be the universe of activities (event classifiers), timestamps and events. Each event $e{\in}\mathcal{E}$ consists of mandatory and additional attributes. Let AN be the set of attribute names. For $an{\in}AN$, we define $\#_{an}(e)$ as the value of attribute an for event e. An event e has mandatory attributes timestamp $\#_t(e){\in}\mathcal{T}$ at which e occurs and activity $\#_{act}(e){\in}\mathcal{A}$ that occurs during e.

We first need to understand the general steps to develop a RTM described in [17]. In the *offline* or training phase, the first step is to prepare the input data, i.e., event log. Since a RTM makes prediction for incomplete traces, it trains on *prefixes* extracted from traces in L. A prefix is extracted by taking the first $k{\in}\mathbb{N}$ events from a completed trace ($\sigma{=}\langle e_1, .., e_n\rangle$) using function $hd^k(\sigma){=}\langle e_1, .., e_k\rangle, k \leq n$. The resulting prefixes are collectively known as a *prefix log* L^* of L. Therefore, data preparation includes cleaning the data, creating a prefix log and feature engineering. Features like *weekday* or *sojourn time* are extracted from event data and categorical features are encoded.

A RTM can be instantiated based on three main parameters, methods for grouping similar prefixes into buckets, prefix encoding methods, and used predic-

tion techniques. For instance, $RTM{=}(p, a, x)$ represents that the model's prefix bucketing method is based on similar prefix lengths (p), the encoding method is aggregating data of all prefix events (a), and ML algorithm is XGBoost (x). After training, the models are tuned using techniques like hyperparameter optimization. Finally, the optimal model's prediction accuracy is evaluated using aggregated metrics, e.g., Mean Absolute Error (MAE).

2.3 Performance Spectrum with Error Progression

To identify process segments subject to high prediction errors due to inter-case dynamics, Klijn et al. [8] introduced a visual analysis technique, *Performance Spectrum with Error Progression (PSwEP)*. It uses the performance spectrum (PS) [2], which maps the performance of each case passing through a segment over time. A process segment $(a, b){\in}\mathcal{A} \times \mathcal{A}$ can be defined as any two successive steps in the process, e.g., a step from activity a to activity b. For traces of form $\langle ..., e_i, e_{i+1}, ... \rangle$, where $\#_{act}(e_i){=}a, \#_t(e_i){=}t_a, \#_{act}(e_{i+1}){=}b$, and$\#_t(e_{i+1}){=}t_b$, we observe an occurrence of a segment (a, b) from time t_a to t_b. Each occurrence of segment (a, b) representing a case is plotted in a PS as a line from (t_a, a) to (t_b, b). In PSwEP, segment occurrences within a PS are classified based on the error progression of the case while passing through the segment. Let $\mathcal{P}$ be the set of predictions made on test data using RTM. Each prediction $pr_k{\in}\mathcal{P}$ corresponds to a prediction made for prefix $hd^k(\sigma){=}\langle e_1, .., e_k \rangle$ at point of prediction $\#_{act}(e_k){=}a_k$ and $t_{pr_k}{=}\#_t(e_k)$, i.e., the time moment of prediction. y_{pr_k} and $\overline{y_{pr_k}}$ denote the actual and predicted outcomes of pr_k. To measure the error progression of segment occurrence (a_k, a_{k+1}) linked to σ, the prediction errors at a_k and a_{k+1} are compared. The difference in relative absolute errors $DRAE(rae_k, rae_{k+1}) = rae_k - rae_{k+1}$ with $rae_k{=}|\overline{y_{pr_k}} - y_{pr_k}|/y_{pr_k}$ is measured. If the prediction error decreases for a segment occurrence, i.e., $DRAE > 0$ this plotted line is colored red in the PSwEP. If the prediction error increases, i.e., $DRAE < 0$ the line is colored blue. Figure 2 shows PSwEP of segment *(Apply Penalty (AP), Send for Credit Collection (SC))* in the RF event log.

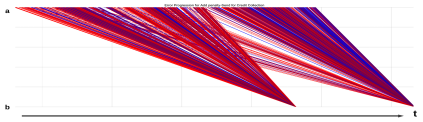

Fig. 2: PswEP for *(Add Penalty (AP), Send for Credit Collection (SC))* in RF: error decrease (red), error increase (blue).

3 Approach

In this section, we will discuss the main approach proposed to develop an inter-case-dynamics-aware RTM. In Section 3.1, we discuss the proposed techniques to automatically identify uncertain segments. In Section 3.2, we discuss the process of identifying and deriving insights about inter-case dynamics. Finally, in Section 3.3, we propose ways to create inter-case features by utilizing derived insights.

3.1 Detecting Uncertain Segments

Measuring Uncertainty of a Process Segment To identify uncertain segments, we need to measure the uncertainty of each process segment. To do so, we

Table 1: Error progression for the occurrence of segments linked to predictions.

Case ID	Prefix	t_{pr_k}	y_{pr_k}	$\overline{y}_{pr_k}$	rae	Segment	$DRAE$	Error Progression
c1	$\langle a \rangle$	1	6	10	0.667			
c1	$\langle a, b \rangle$	2	5	2	0.600	(a, b)	0.007	decrease
c1	$\langle a, b, c \rangle$	4	3	2	0.333	(b, c)	0.267	decrease
c1	$\langle a, b, c, d \rangle$	4	3	2	0.333	(c, d)	0	same
c1	$\langle a, b, c, d, e \rangle$	7	0	0	∞	(d, e)	$-\infty$	increase
c2	$\langle a \rangle$	3	11	14	0.272			
c2	$\langle a, b \rangle$	5	9	14	0.555	(a, b)	-0.283	increase
c2	$\langle a, b, c \rangle$	14	2	3	0.500	(b, c)	0.055	decrease

Table 2: Measuring uncertainty of each segment by aggregating its occurrences to calculate *observations*, *decrease cases*, and *increase cases*.

Segment	Observations	Decrease Cases	Increase Cases
(a, b)	2	1	1
(b, c)	2	2	0
(c, d)	1	1	0
(d, e)	1	0	1

first measure the $DRAE$ (Section 2.3) of individual segment occurrences linked to predictions made using RTM on test data. Table 1 shows an example of how individual predictions are aligned with segment occurrences and the error progression of each occurrence is classified. A *decrease* in error, i.e., $DRAE > 0$ for a case passing through segment (a, b) implies that after the occurrence of activity b the remaining time prediction improves. This decrease could indicate some uncertainty between activity a and b, which gets resolved after activity b completes. An *increase* in error implies that after the occurrence of activity b, the prediction model becomes more unsure about how the partial trace will proceed. If prediction error remains the same, i.e., $DRAE$=0, there is no clear indication of uncertainty within the process segment. We can either ignore such rare cases or include them as error *decrease*, where we consider the latter.

Based on above insights, we use three aggregated metrics to quantify uncertainty of segments. For each segment (S) linked to $\mathcal{P}$, we measure (1) *observations* or total occurrences linked to S in $\mathcal{P}$, (2) *decrease cases* or total occurrences linked to S with $DRAE \geq 0$, and (3) *increase cases* or total occurrences linked to S with $DRAE < 0$. Table 2 is the result of applying the above aggregations to occurrences of segments found in Table 1.

Selecting the Most Uncertain Segments We define a mapping function $u_S : \mathbb{N} \times \mathbb{R} \longrightarrow [0, 1]$ to select a subset of process segments for which inter-case features could be created (Equation 1). The inputs are the number of observations (o) and the ratio $r = d/max(1, i)$ of decrease cases (d) to increase cases (i) for segment S (as shown in Table 2). Output 1 indicates the segment is highly uncertain. Note that ideal candidates for uncertain segments are those where decrease cases are almost the same or more than increase cases, i.e., their ratio should be greater than some threshold t_r. The threshold for the number of observations (t_{obs}) indicates the occurrences of the segments. These thresholds can be set for each process individually.

$$u_S(o, r) = \begin{cases} 1 & \text{if } o \geq t_{obs} \text{ and } round(r) \geq t_r \\ 0 & otherwise \end{cases} \tag{1}$$

Let SG be the set of all segments in a process and SG_{start} be the set of starting segments. Therefore, we apply u_S to $S \in SG \setminus SG_{start}$ based on some t_r and t_{obs} and select set of segments U for which $u_S(o, r)$=1. Removing starting activities in traces is due to the fact that the RTM has too little information,

and the prediction error is likely to decrease when the second activity occurs. We use the RF event log [9] as the running example. First, predictions are made on the last 20% (temporally split) of the event log using a RTM, here $RTM{=}(p, a, x)$. Then, these predictions are used to measure the uncertainty of each process segment and u_S is applied to all non-starting segments. We set $t_r{=}1$ and $t_{obs} > \mu$, e.g., $t_{obs}{=}2 * std$ where μ, std are the mean and standard deviation of segment occurrences. The selected uncertain segments are *(Send Fine (SF), Insert Fine Notification (IF))*, *(Insert Fine Notification (IF), Add Penalty (AP))* and *(Add Penalty (AP), Send for Credit Collection (SC))*. The details of selecting the most uncertain segments presented here[3].

3.2 Identifying Inter-Case Dynamics in Uncertain Segments

In order to diagnose causes for uncertainty within segments, first, we visualize the performance of cases within the process segment using PSwEP (Section 2.3). After that, the observed patterns in the performance spectrum are compared to a taxonomy [2] to identify underlying process behavior that causes inter-case patterns within the process segment. We explain the process of deriving insights for the uncertain segments identified in the running example.

In the shown PSwEP of $(SF,\ IF)$ in Fig. 3 (left), two patterns, *batching-at-start* and *non-batching* FIFO behavior are identified. These are elementary patterns related to the order of case arrival. We notice uncertainty (as shown by the red lines) for non-batched cases. Therefore, RTM is currently not aware that non-batched cases are processed much faster than batch ones. Batched cases within the segment (Fig. 3) are also classified using red. The uncertainty concerning these cases is caused by the prediction model's lack of awareness about *batching-at-start* dynamics. The order of lines in PSwEP of $(AP,\ SC)$ presented before in Fig. 2 clearly shows that the inter-case pattern is caused by *batching-at-end*. The prediction model is currently unaware of this inter-case dynamic within the process segment. In PSwEP of $(IF,\ AP)$ in Fig. 3 (right), we observe a FIFO with a constant time pattern in the order of case arrival. The performance of a case is strongly correlated to the previous case that passed through the segment. We also know that there are two possible activities, *Add Penalty (AP)* or *Insert Date Appeal to Prefecture (ID)*, that can occur after *Insert Fine Notification (IF)* and the time that cases wait within the segments is significantly different. Therefore, incorrectly assuming the path of a case arrives at IF impacts the remaining time prediction. We are able to predict the path by observing the recent performance of cases in $(IF,\ AP)$ and $(IF,\ ID)$ w.r.t. inter-case dependencies. Lastly, across three segments, we observe changing the density of lines indicating varying workloads.

Based on the above derived insights, we define the abbreviated inter-case pattern(s) identified for segments $(SF,\ IF), (IF,\ AP)$ and $(AP,\ SC)$ as $R_1{=}non-batching,\ batch(s)$, $R_2{=}non-batching$ and $R_3{=}batch(e)$ respectively.

[3] https://www.pads.rwth-aachen.de/go/id/qcekn/lidx/1

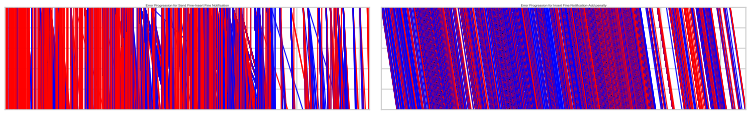

Fig. 3: PSwEP for segments *(Send Fine (SF), Insert Fine Notification (IF))* (left), and *(Insert Fine Notification (IF), Add Penalty (AP))* (right) in the RF event log.

Table 3: The created inter-case features for segment predictions ($\mathcal{C}$=$\{C_S, C_{S_1}, C_{S_2}, C_{S_3}\}$) and waiting time ($w$) within uncertain segments for the RF event log.

Case ID	Activity	Timestamp	...	C_S	C_{S_1}	C_{S_2}	C_{S_3}	w	y
N71924	SF	09-17 08:00	...	1	1	0	0	1154258.7	39229200.0
S120874	AP	05-09 08:00	...	1	0	1	0	2808000.3	28080000.0
S86803	SF	11-03 09:00	...	1	1	0	0	1212661.0	36115200.0
S57422	SC	01-10 09:00	...	0	0	0	0	0.0	0.0
S70222	CF	09-29 08:00	...	0	0	0	0	0.0	40438800.0

3.3 Inter-Case Feature Creation

As the running example shows, ignoring inter-case dynamics results in high prediction errors for prefixes expected to pass through segment $S{\in}U$. Therefore, we need to provide the RTM information about a prefix being subject to inter-case pattern R detected in uncertain segment S prior to the occurrence of the segment. We use these insights to develop inter-case features.

Consider the running example with three uncertain segments S_1, S_2, and S_3 with inter-case pattern(s) R_1, R_2 and R_3, respectively, we define the following inter-case features: (1) $C_S{\in}\{0,1\}$, to indicate if a prefix passes through an uncertain segment $S{\in}U$, (2) $C_{S_1}{\in}\{0,1\}$, to indicate that the prefix passes through S_1 with inter-case pattern(s) R_1, (3) $C_{S_2}{\in}\{0,1\}$, to indicate that prefix passes through S_2 with inter-case pattern(s) R_2, (4) $C_{S_3}{\in}\{0,1\}$, to indicate that prefix passes through S_3 with inter-case pattern(s) R_3, and (5) w, to indicate the waiting time of the prefix in $S{\in}U$, as a result of inter-case pattern(s) R. As a result of the feature creation step for the running example, Table 3 is generated showing inter-case features. These features are used to train an inter-case-dynamics-aware RTM. Feature y is the target feature, i.e., remaining time to completion.

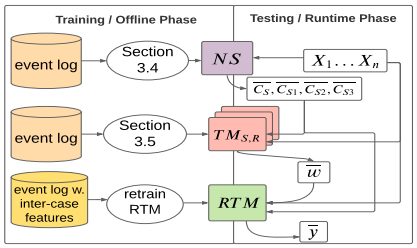

Fig. 4: The overview of feature creation process for RF event log with uncertain segments S_1, S_2 and S_3.

Creating inter-case features for an ongoing case at run-time requires its own prediction models. We need a model (NS) to predict inter-case features related to segment prediction and waiting time prediction model ($TM_{S,R}$) for each uncertain segment $S{\in}U$ with inter-case pattern(s) R. Figure 4 gives an overview of the steps involved in creating the models (offline) and utilization of these models to create inter-case features (at run-time). This process is the extended version of the presented feature-creation in [8].

3.4 Predicting the Next Segment

Classifier NS should determine if a prefix passes through segment $S{\in}U$ at the point of prediction. To build NS, we build a classifier for the next activity

prediction using [18] and modify the outcome to predict the value of segment prediction inter-case features. Let $hd^k(\sigma)$ be the input prefix with last activity a for NS. If the next activity predicted is b, we say that the prefix passes through segment (a, b) at the point of prediction. If $(a, b) \in U$, then $\overline{C_S}$=1, else we set it to 0. If $\overline{C_S}$=1, we set the value of the boolean variable representing the prefix passing through segment (a, b) as 1. Therefore, if predicted (a, b)=S_1, then $\overline{C_{S_1}}$=1, $\overline{C_{S_2}}$=0, and $\overline{C_{S_3}}$=0. The collective set of predicted features using NS is called $\overline{\mathcal{C}}$.

3.5 Predicting Waiting Time

In this section, we present general steps to create a waiting time prediction model ($TM_{S,R}$) that predicts how long a case stays in a segment S with inter-case patterns R. Consider a case c_1 arriving at segment S=(a, b) at time t_a (Fig 5). Because of inter-case dynamics, the waiting time w of c_1 will depend on the performance of other cases in relevant segments in some recent time interval, i.e., historic spectrum (S_h) [3] and relevant individual properties (intra-case features). The intra-case feature of c_1 and performance seen within S_h can be encoded as feature vector $X_1..X_n$ using insights gained about R within S. This allows us to formulate the waiting time prediction problem as a supervised learning problem: w=$f(X_1..X_n) + \varepsilon$, where function f predicts w from $X_1..X_n$. To learn f, we create training samples using the sliding window method and apply a ML method like LightGBM [6] that tries to minimize prediction error ε. Table 4 shows sample data used to train a $TM_{S,R}$ for *(IF, AP)*.

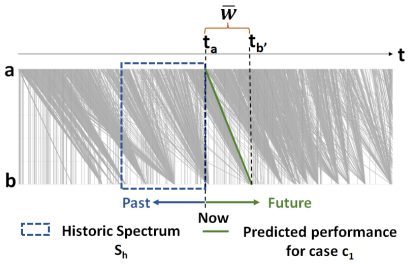

Fig. 5: Illustration of a single instance for $TM_{S,R}$ to learn waiting time for case c_1 using performance-related features extracted from S_h and relevant individual properties of c_1.

Waiting Time Prediction for Non-batching Dynamics In Section 3.1, we learned that $\overline{w}$ of a case in (IF, AP) is influenced by R=$non-batching$ and varying workload in segments (IF, AP) and (IF, AD). To derive workload related context, we define h in S_h as the period between arrival of c_1 and the last case before it and derive: (1) *starting cases* or the number of cases that started (arrived at the segment) in period h, (2) *ending cases* or the number of cases that completed (exited the segment) in period h and (3) *pending cases* or the number of cases that have started within period h and will complete in the future. Since, performance of a case in (IF, AP) strongly depends on the previous case, we also extract last waiting time (w_l), e.g., Table 4.

Table 4: Sample data for training waiting time prediction model ($TM_{S,R}$) for uncertain segment (IF, AP) with pattern R=$non-batching$.

starting cases	ending cases	pending cases	w_l	w
60	37	60	5183000.0	5184000.0
14	10	14	5184000.0	5184000.0
19	17	18	5187000.0	5187000.0

Waiting Time Prediction for Batching-at-Start Dynamics $\overline{w}$ of a case $c1$ arriving at *(SF, IF)* will depend on R=$batch(s), non-batching$ and varying workload within the segment. Therefore, S_h contains only segment *(SF, IF)*. To learn performance related to R=$non-batching$ and the workload, we include features presented in Section 3.5. To include features related to R=$batch(s)$, we extract features related to the previous batch [7] with batching moment BM_l: (1) least (w_{min}) and longest waiting time (w_{max}) in previous batch, (2) previous *batch size* and *batch size percentile*, (3) mean and standard deviation of $IBCT$ or *inter batch case completion time*, which is the time difference between the completion times of two successive cases in the batch and (4) *batch type*, which distinguishes batches with less than 2 observations that behave like simultaneous batches, and (5) CIA or *case inter-arrival time* which the time between arrival of c_1 and the case before it. We also include relevant intra-case features *resource, expense, points* and *weekday, month, hour* of previous batch. *Duration* or the waiting time of the case in the previous segment is also included to distinguish batched and non-batched cases. However, learning case-specific w is difficult because *batching-at-start* cases proceed randomly, i.e., not in the order they arrived at the batch. To avoid learning this random behavior, we propose building a $TM_{S,R}$ that predicts the average of expected waiting times for all cases that arrive along with c_1. Hence, the training data will be prepared by extracting the above-mentioned features and then aggregating (calculating mean) feature values for instances that correspond to cases arriving simultaneously in the segment.

Waiting Time Prediction for Batching-at-End Dynamics *(AP,SC)* contains inter-case dynamics caused by R=$batch(e)$ and varying workload. To consider the varying workload across the segment, we include the features presented in Section 3.5. To learn batching related performance, we extract features w_{min}, w_{max} and CIA described in previous section. Additionally, we include: (1) t_{lb}: or the time elapsed since the occurrence of the last batch, (2): the mean and standard deviation of $IBIA$ *(inter-case arrival rate)* which is the difference between the arrival times of two successive cases in the batch. We also include intra-case features *month* and *weekday*.

4 Evaluation

4.1 Experimental Setup

We evaluate the proposed approach on two real-life event logs: the RF event log [9] and BPIC'20 event log [4]. We implemented inter-case feature creation and PSwEP in Python, which is publicly available[4]. To train and test RTMs, we use the benchmark implementation for RTM approaches[5] [17]. First, we make predictions with RTM=(p, a, x) for both event logs to identify uncertain segments and their patterns. The uncertain segments identified from RF event log are *(SF, IF)*, *(IF, AP)* and *(AP, SC)* with inter-case pattern(s) R_1=$non-batching$, $batch(s)$,

[4] https://github.com/karshreya98/Inter_case_aware_RTM

[5] https://github.com/verenich/time-prediction-benchmark

Table 5: Weighted average MAE (in days) of different RTM models with different bucketing, encoding and ML methods, e.g., (p, a, x), while using no inter-case features $I(\emptyset)$ and with the created inter-case features using segment predictions $I(\overline{\mathcal{C}}, \overline{w})$.

		(p,a,x)	(p,l,x)	(c,a,x)	(c,l,x)	(p,l,r)	(c,a,r)	(c,l,r)	(s,l,x)
RF	$I(\emptyset)$	212.60	209.69	210.32	208.59	221.39	221.05	221.53	203.29
	$I(\overline{\mathcal{C}}, \overline{w})$	**187.65**	**179.78**	**201.17**	**179.34**	**191.06**	**205.87**	**190.63**	**179.78**
BPIC'20	$I(\emptyset)$	3.68	3.66	3.87	3.62	3.85	3.90	3.72	3.66
	$I(\overline{\mathcal{C}}, \overline{w})$	**3.58**	**3.57**	**3.81**	**3.53**	**3.69**	**3.70**	**3.65**	**3.48**

R_2=$non-batching$ and R_3=$batch(e)$, respectively. The two identified uncertain segments from BPIC'20 event log are *(Declaration Final Approved by Administration (DF), Request Payment (RP))* and *(Request Payment (RP), Payment Handled (PH))*. The inter-case pattern(s) identified for segments *(DF, RP)* and *(RP, PH)* are R_1=$non-batching$, $batch(s)$, and R_2=$batch(e)$ respectively. To create inter-case features, we implement NS using [18] and follow steps described in Section 3.5 to create $TM_{S,R}$ models using LightGBM [6]. Predictions are made with different bucketing prefixes, encoding prefix events, and ML methods. We consider *prefix bucketing methods* to be grouping by prefix lengths (p), using a clustering algorithm (c) or grouping all prefixes in a single bucket (s). Common *prefix encoding methods* include data of only last prefix event (l) or aggregating data of all prefix events (a), and apply ML models, XGBoost (x) or random forest (r) to the input encoded feature vectors. The following input configurations are used: (1) $I(\emptyset)$: event log with no inter-case features, (2) $I(\mathcal{C}, \overline{w})$: event log with inter-case features created using actual segment prediction $\mathcal{C}$, and (3) $I(\overline{\mathcal{C}}, \overline{w})$: event log with inter-case features created using segment prediction made using NS. We use 80% and 20% (by temporally splitting) of the event logs for training and testing the RTMs. To measure overall prediction accuracy, we measure the weighted average MAE [17] of all predictions $\mathcal{P}$ made on test data.

4.2 Results

Table 5 shows that using inter-case features leads to an increase in performance for all 8 combinations of bucketing prefixes, encoding prefix events, and ML methods in RTMs against baseline $I(\emptyset)$. For the RF event log, we see that prediction error decreases by a maximum of 14.26% and a minimum of 4.27% for methods (p, l, x) and (c, a, x), respectively, with $I(\overline{\mathcal{C}}, \overline{w})$. For the BPIC'20 event log, we observe a maximum decrease of 5.12% and a minimum decrease of 1.55% in weighted average MAE for methods (c, a, r) and (c, a, x), respectively. Since BPIC'20 is a smaller event log with fewer cases subject to the identified inter-case patterns, the overall reduction in prediction error is smaller. The most accurate predictions for the RF event log obtained using $I(\overline{\mathcal{C}}, \overline{w})$ with (c, l, x), has a MAE 0.6 days less than the benchmark result [17]. However, our approach's privilege is that these predictions can be interpreted more easily because of the inter-case features.

Table 6: MAE (in days) for different configurations (I) with the similar lengths bucketing (p), aggregating events data for encoding prefix events (a), and XGBoost (x) as the ML method, $RTM = (p, a, x)$. $\mathcal{P}_k$ is the set of all predictions for prefixes of length k.

RF	$I(\emptyset)$	$I(\mathcal{C}, \overline{w})$	$I(\overline{\mathcal{C}}, \overline{w})$	BPIC'20	$I(\emptyset)$	$I(\mathcal{C}, \overline{w})$	$I(\overline{\mathcal{C}}, \overline{w})$
$\mathcal{P}_{k=2}$	176.37	107.85	**106.74**	$\mathcal{P}_{k=3}$	4.03	**3.84**	4.06
$\mathcal{P}_{k=3}$	227.38	**189.22**	200.02	$\mathcal{P}_{k=4}$	2.64	**2.22**	2.23
$\mathcal{P}_{k=4}$	202.92	**123.19**	171.11	$\mathcal{P}_{k=5}$	1.07	0.98	**0.97**

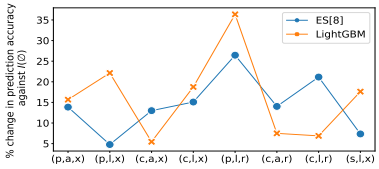

Fig. 6: Comparing prediction results for RF

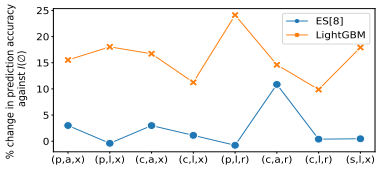

Fig. 7: Comparing prediction results for BPIC'20

In our approach, inter-case features are primarily included for prefixes passing through uncertain segments which occur at some step k of the process. Therefore, we look at MAE of predictions made for all prefixes of relevant length k, i.e., $\mathcal{P}_k \subseteq \mathcal{P}$. Segments $(SF,\ IF)$, $(IF,\ AP)$ and $(AP,\ SC)$ of the RF event log occur predominantly at step $k = 2$, $k = 3$ and $k = 4$ of the process respectively. Segments $(DF,\ RP)$ and $(RP,\ PH)$ of the BPIC'20 log occur predominantly at steps $k = 3$ and $k = 4,\ 5$ respectively. Table 6 shows us the results for predictions made using $RTM = (p, a, x)$. For the RF event log, the prediction error decreases by 39%, 12% and 15% for $\mathcal{P}_2$, $\mathcal{P}_3$ and $\mathcal{P}_4$, respectively using $I(\overline{\mathcal{C}}, \overline{w})$ over baseline. For BPIC'20, error decreases up to 15% and 9% for $\mathcal{P}_4$ and $\mathcal{P}_5$, respectively, when using $I(\overline{\mathcal{C}}, \overline{w})$. However, the MAE of $\mathcal{P}_3$ is slightly higher for configuration $I(\overline{\mathcal{C}}, \overline{w})$ compared to $I(\emptyset)$. This is because of incorrect segment predictions for *(DF, RP)* made by NS which is proven by the results of $I(\mathcal{C}, \overline{w})$. Figures 6 and 7 compare the *batching-at-end* aware predictions made using inter-case features created in our approach that uses LightGBM [6]) and previous approach [8] that uses exponential smoothing (ES). We measure the increase/decrease in performance of $\mathcal{P}_4$ made using different combination of RTMs over their respective baselines. We compare only predictions at k=4 for both logs where uncertain segments with *batching-at-end* dynamics occur. Figure 6 shows that, our approach performs better than previous approach in 5 of the 8 input configuration (I) for batched cases in RF event log. Figure 7 shows that for the batched cases in BPIC'20 log, our method performs better for all the configurations.

5 Conclusion

We presented an approach to systematically discover a subset of uncertain process segments with inter-case dynamics that cause high prediction errors. Contrary to previous approaches, our designed function for detecting the subset of uncertain segments, limited the manual intervention to the identification of inter-case patterns within these segments. Using visual analysis, we identified and gained insights about inter-case pattern(s) within uncertain segments. In particular, we gained insights into non-batching (FIFO and unordered), *batching-at-start*, and *batching-at-end* inter-case patterns. Subsequently, we included these insights in remaining time predictions by transforming them into the inter-case features. For instance, there is a maximum increase in overall prediction performance by 14.2% for RF event-log. Since there is no standardized process to create a ML model for inter-case feature creation, our proposed approach is also sensitive to user interpretation. Yet, it provides more interpretability to RTMs. Note that despite an overall decrease in prediction error, some prefixes were heav-

ily over-predicted or under-predicted. Therefore, the next step is to improve the prediction models and leverage routing probability derived from stochastic process models. It improves the inter-case feature creation for segment prediction. Another possible path is to make RTM aware of non-case-related aspects, e.g., resources dependencies.

References

1. van der Aalst, W.M.P., Schonenberg, M., Song, M.: Time prediction based on process mining. Information Systems **36**(2), 450–475 (2011)
2. Denisov, V., Fahland, D., van der Aalst, W.M.P.: Unbiased, fine-grained description of processes performance from event data. In: Business Process Management (2018)
3. Denisov, V., Fahland, D., van der Aalst, W.M.P.: Predictive performance monitoring of material handling systems using the performance spectrum. In: International Conference on Process Mining (ICPM). pp. 137–144. IEEE (2019)
4. van Dongen, B.F.: "BPI Challenge 2020: Domestic declarations dataset (2020)
5. Galanti, R., Coma-Puig, B., de Leoni, M., Carmona, J., Navarin, N.: Explainable predictive process monitoring. In: (ICPM). pp. 1–8. IEEE (2020)
6. Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., Liu, T.Y.: LightGBM: A highly efficient gradient boosting decision tree. p. 3149–3157. NIPS'17
7. Klijn, E., Fahland, D.: Performance mining for batch processing using the performance spectrum. In: Business Process Management Workshops (2019)
8. Klijn, E.L., Fahland, D.: Identifying and reducing errors in remaining time prediction due to inter-case dynamics. In: (ICPM). pp. 25–32. IEEE (2020)
9. de Leoni, M.M., Mannhardt, F.: Road traffic fine management process (2015)
10. Navarin, N., Vincenzi, B., Polato, M., Sperduti, A.: LSTM networks for data-aware remaining time prediction of business process instances. 2017 IEEE Symposium Series on Computational Intelligence (SSCI) pp. 1–7 (2017)
11. Polato, M., Sperduti, A., Burattin, A., de Leoni, M.: Time and activity sequence prediction of business process instances. Computing **100** (2018)
12. Pourbafrani, M., van der Aalst, W.M.P.: Extracting process features from event logs to learn coarse-grained simulation models. In: CAiSE 2021. pp. 125–140 (2021). https://doi.org/10.1007/978-3-030-79382-1_8
13. Pourbafrani, M., Jiao, S., van der Aalst, W.M.P.: SIMPT: process improvement using interactive simulation of time-aware process trees. In: RCIS 2021. Lecture Notes in Business Information Processing, vol. 415, pp. 588–594. Springer (2021). https://doi.org/10.1007/978-3-030-75018-3_40
14. 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: Business Process Management, vol. 10445, pp. 306–323. Cham (2017)
15. Senderovich, A., Weidlich, M., Gal, A., Mandelbaum, A.: Queue mining for delay prediction in multi-class service processes. Information Systems **53**, 278–295 (2015)
16. Tax, N., Verenich, I., La Rosa, M., Dumas, M.: Predictive Business Process Monitoring with LSTM Neural Networks **10253**, 477–492 (2017)
17. Verenich, I., Dumas, M., Rosa, M.L., Maggi, F.M., Teinemaa, I.: Survey and cross-benchmark comparison of remaining time prediction methods in business process monitoring. ACM **10**(4), 1–34 (2019)
18. Wang, J., Yu, D., Liu, C., Sun, X.: Outcome-oriented predictive process monitoring with attention-based bidirectional LSTM neural networks. In: ICWS. pp. 360–367 (2019)