Predictions in Predictive Process Monitoring with Previously Unseen Categorical Values

Johannes Roider[1], Weixin Wang[2], Dario Zanca[1], Martin Matzner[2], and Bjoern M. Eskofier[1]

[1] Machine Learning and Data Analytics (MaD) Lab, Friedrich-Alexander Universität Erlangen-Nürnberg, Erlangen, Germany
johannes.roider@fau.de
[2] Chair of Digital Industrial Service Systems, Friedrich-Alexander Universität Erlangen-Nürnberg, Nürnberg, Germany

Abstract. Predictive process monitoring (PPM) methods provide users with real-time predictions about ongoing process instances. Machine learning models used for such tasks do not account for data variability, such as the occurrence of previously unseen categorical feature values. Concept drift adaptation solutions are suggested in such scenarios. However, adapting to new feature values requires time and a sample size large enough to train a well-generalizing model. Still, users expect seamless communication during the timeframe between the first occurrence of a new value and the availability of an updated model. Dedicated solutions are needed since encoding techniques like one hot encoding cannot handle previously unseen values by default. In this work, we first introduce and discuss possible solutions from a business perspective, ranging from temporary shutdowns to dedicated manual and technical solutions for an uninterrupted continuation of predictive services. Next, we present five variants for one hot encoding to handle previously unseen categorical values. This is followed by a case study using six real-world event logs and two machine learning models, XGBoost and LSTM, to identify the variants that produce the most reliable remaining time predictions. The study also includes the evaluation of two baseline models as an alternative to the machine learning models. The results show that previously unseen categorical values can be handled on a technical level without severely affecting the remaining time prediction quality. However, future research is required to provide more practical recommendations.

Keywords: Predictive Process Monitoring · Remaining Time · Machine Learning.

1 Introduction

Predicting the remaining time of business processes is a major task in predictive process monitoring (PPM), next to outcome and next activity prediction. To provide remaining time prediction services, machine learning models are first trained on historical data (offline phase) and then deployed and applied to real-time streaming data (online phase) [2].

Users of predictive services expect an uninterrupted, consistent prediction quality. However, business process data is subject to change over time. Examples include the occurrence of new activities, customers, or products over time. These are examples of data variability on the level of categorical values [7]. One solution to such concept drifts discussed in previous work is to (re)train machine learning models with the new data [6], [10]. However, there are two practical challenges:

1. When a previously unseen categorical value is encountered, there are no ground truth labels available to (re)train a model. Still, predictions are expected by end users.
2. Machine learning models need a sample size large enough in order not to overfit to the training data. Even if a small sample size with ground truth labels is available for new categorical values, there is a risk that a (re)trained model overfits and produces low-quality predictions. Collecting a certain amount of data before (re)training a model is required.

In both scenarios there is a gap in time where new categorical values occur but a machine learning model that incorporates them is not available. Predictive service providers are challenged to bridge this time interval without negatively impacting user satisfaction. Therefore, we list and discuss possible solution strategies from a business perspective in Sec. 4.

Challenges with previously unseen categorical values also arise on a technical level. Many machine learning models require specifying the number of distinct values for categorical features in advance. However, it is not foreseeable whether and how many new values will occur in the future. If a new value occurs during the online phase that has not been considered in the offline phase, it can lead to service disruptions.

For these reasons, our evaluations focus on previously unseen categorical values occurring over time and its handling with one hot encoding. This encoding technique is commonly used to encode categorical features. In Sec. 5, we discuss one-hot encoding and the reasons of technical failures with new categorical values in detail. Furthermore, we present five specific ways to encode previously unseen categorical values with one-hot encoding. This is followed by a benchmark study in the context of remaining time prediction. We evaluate the proposed solutions with two variants of XGBoost and a Long Short-Term Memory (LSTM) model on six real-world datasets and compare them to the performance of two simple baselines presented in prior literature.

The remainder of this work is structured as follows: In Sec. 2 we discuss related work, followed by general concepts relevant to this work in Sec. 3. New concepts are introduced in Sec. 4 by discussing possible solution strategies from a business perspective, and in Sec. 5 by presenting one-hot encoding schemes for previously unseen categorical values. In Sec. 6 we outline our experiments which are discussed in Sec. 7. In Sec. 8 we conclude our work.

2 Related Work

Predicting the remaining time has been studied in several works. It has been analyzed with structured reviews and benchmarking methods. Verenich et al. [13] conducted a review and benchmark study where they compared process-aware methods, variants of XGBoost, and a neural network based method. They found that DA-LSTM, introduced by Navarin et al. [8], outperforms process-aware and classical machine learning methods. Rama-Maneiro et al. [11] focused on deep learning in their benchmark study. They compare four methods for remaining time prediction and also determined DA-LSTM as the best method. Recently, classical methods were investigated again due to their lower computational needs. Aalikhani et al. [1] compare four different methods in their regression-based and classification-based counterparts. The latter methods require a discretization of labels, but they often outperform regression-based models. Oyamada et al. [9] compare three models with a focus on time-related feature engineering. They find that LSTM outperforms the classical methods, but classical methods can be improved with a careful choice of time-related features.

Few works in PPM have motivated the problem of new categorical values during the online phase. Mangat and Rinderle-Ma [7] propose two solution strategies for one-hot feature encoding (void encoding and reserving additional capacities). However, they only consider void encoding and not different variants during their evaluation for predicting the next activity. Pasquadibisceglie et al. [10] use word2vec from the Gensim package[3] since it provides automatic handling of new categorical values. However, the original implementation of Word2vec does not support new values, and the Gensim package provides one specific implementation. There might be a variety of possible solutions available. The same applies to one-hot encoding, for which we will discuss several variations.

Other domains also encounter the challenge of handling previously unseen categorical values. Dedicated work exists in the field of recommendation systems. An overview of encoding strategies is given by Shiao et al. [12]. Some of the proposed solutions overlap with the work by Mangat et al. [7], for example both papers propose zero encoding. Shiao et al. [12] propose further solutions like mean embedding and random embeddings, some of which we consider in Sec. 5.

3 Prerequisites

Data The data used in predictive process monitoring are *event logs*. Event logs are sets of sequences, also called *cases* or *traces*. Each trace is a sequence of *events*. Events are standardized data structures which have specific attributes assigned. Common attributes are *case identifiers*, *activities*, and *timestamps*, which identify to which trace an event belongs, the task that is being performed, and when it has been performed, respectively. Further optional attributes, called *context attributes*, can be assigned. Examples include resources or costs.

[3] https://radimrehurek.com/gensim/

Training samples called *prefixes* are generated from each trace to train a machine-learning model. For each event in the trace one prefix is created by including the whole history of the trace up to this specific event. The corresponding remaining time label is represented by the difference in time from the last event of the trace to the specific event at hand.

Prefixes are preprocessed to reflect a data format which can be utilized by machine learning algorithms. The specific representation depends on the algorithm used. We differentiate between numerical and categorical features. Numerical features take on continuous values. Categorical features are discrete. While numerical features can be directly utilized, optionally with a predefined normalization beforehand, categorical features often require specific encoding.

Encoding of Categorical Features The encoding of categorical values depends on the machine learning model employed. Tree-based models like random forests or XGBoost process categorical features by determining splits in individual trees directly on the discrete feature values. Another approach for XGBoost is to transform categorical features first into a numerical representation like one-hot encoding. In one-hot encoding, a feature vector is created whose length corresponds to the number of categories observed in the dataset at hand (see Sec. 5). The type of encoding for tree-based methods depends heavily on the implementation. For example, XGBoost[4] up to version 1.6 did not support direct categorical encodings but required a numerical representation like one-hot encoding. Scikit-learn[5], a commonly used library for tree-based methods like random forests, requires numerical representation and direct categorical splits are not supported as of the latest version 1.5.

Neural networks require a numerical encoding and one-hot encoding is often chosen. Since one-hot encoding is applicable with both, XGBoost and neural networks, and since it was also used in prior literature (see Verenich et al. [13]), we conducted our experiments with one-hot encoding.

4 Handling Previously Unseen Categorical Values from a Business Perspective

We motivated in Sec. 1 the problem of previously unseen categorical values. From a business perspective this should be handled without negative impact on the satisfaction levels of end users. Service providers need to be aware of possible solutions and cope with the challenge based on their individual end users' needs. We have identified four different solutions strategies:

1. Disable the service (partially): To avoid inaccurate predictions, the service is disabled for samples where previously unseen values are encountered. A more rigid approach is to deactivate the service completely. The service is resumed after a large enough sample size for the new value is collected, a new model is trained, evaluated and deployed.

[4] https://xgboost.ai
[5] https://scikit-learn.org

2. Escalate predictions: Traces containing new values are escalated to human experts. These take over the task and provide a prediction based on their domain knowledge. While the service can be continued, extra resources are required to uphold it. Furthermore, human interventions take more time to finish, and dedicated user interfaces are needed to support the manual work.
3. Exclude features with previously unseen values: A backup model is used which does not use the affected feature. This can be a model using all other available features or a simple baseline solution predicting a standard value. This ensures that the service is continued fully automated, but there is a high risk that the quality of predictions drops. Furthermore, backup models need to be maintained to be immediately available.
4. Encode feature value as "new value": Use the deployed model, but provide an input signal that there is a previously unseen categorical value encountered. There are different possibilities for such an encoding which we discuss in Sec. 5. Advantages are that the service is not interrupted and there is only one model to be maintained. However, such a strategy can lead to a reduced prediction accuracy.

Table 1. Solution strategies to handle previously unseen categorical values

Mitigation Strategy	Advantages	Disadvantages
1. Disable service (partially)	No risk of low quality predictions;	Service interruptions can affect customer satisfaction;
2. Escalate predictions	High quality predictions;	Time consuming due to manual work; Risk of inconsistent predictions (different experts make different predictions); Escalation procedure and additional user interfaces needed;
3. Exclude categorical feature	Fast response times;	Risk of reduced prediction quality; Additional model(s) to be maintained;
4. Encode "new value"	Fast response times; Only one model to be maintained;	Risk of reduced prediction quality;

The derived advantages and disadvantages are summarized in Tab. 1. The right strategy to be chosen depends on individual business requirements. If high-quality predictions are paramount, strategies 1. and 2. might be preferred. If fast response times and no interruptions are required while reduced accuracy can be tolerated, strategies 3. and 4. may offer the better choice.

In our case study, we will compare strategies 3. and 4. We investigate whether models receiving the input signal of a "new value" can outperform simple baseline

models or if they fail to produce reliable predictions. This will give practitioners insights on whether technical solutions are feasible.

5 Encoding of Previously Unseen Categorical Values

Let's assume a categorical feature for which n distinct values occur in the training set, i.e. the set of distinct values is defined as $C_{\text{train}} = \{c_1, c_2, ..., c_n\}$. Next, we define a bijective function f that maps a value c_j to a unique integer i_j, i.e. $f : C_{\text{train}} \rightarrow I$ where $\text{I} = \{0, 1, ..., n-1\}$. In other words, each categorical value can be mapped uniquely to exactly one integer value, and vice versa.

For one-hot encoding, we create a feature vector V with length n and set all values to 0 (zero), i. e. $V = [v_0, v_1, ..., v_{n-1}]$ where $\forall v_m \in V : v_m = 0$. Given attribute value c_j, we obtain i_j by applying $f : c_j \rightarrow i_j$ and set $\forall v_m \in V : v_m = 1, \text{if } m = i_j$. As an example, let's assume $C_{\text{train}} = \{A, B, C, D\}$. Function f gives the following mappings: $A \rightarrow 0, B \rightarrow 1, C \rightarrow 2, D \rightarrow 3$. The one-hot encoded feature vector for event C is given by $V = [0, 0, 1, 0]$.

Let's now assume a test set c_{test} such that $C_{\text{test}} \setminus C_{\text{train}} \neq \emptyset$. That means a new feature value c_k occurs in the test set which is not present in the training set. A mapping by f for c_k is not defined. In technical implementations such an approach can lead to execution errors. Therefore, a contingency method needs to be established on how to encode previously unseen values to avoid such errors and ensure the continuation of a service.

We present solutions to situations where $c_k \notin c_{\text{train}} \wedge c_k \in c_{\text{test}}$ and c_k needs to be one-hot encoded. The corresponding example assumes that $C_{\text{train}} = \{A, B, C, D\}$ and $C_{\text{test}} = \{A, B, C, D, E\}$ where value E represents c_k. *Zero encoding* and *Dummy encoding* were previously mentioned by Mangat and Rinderle-Ma [7] and denoted as *Void encoding* and *Additional Capacity*, respectively.

Zero Encoding If there is no mapping defined by f for c_k, the one-hot encoding for c_k is defined as a zero vector, i.e. $V = [v_0, v_1, ..., v_{n-1}]$ where $\forall v_j \in V : v_j = 0$. Encoding value E corresponds to the following vector: $V = [0, 0, 0, 0]$.

Dummy Encoding With this encoding we reserve an additional position in the one-hot encoded vector, such that $V = [v_0, v_1, ..., v_n]$. During training, v_n will always be 0. On an algorithmic level, we then set $v_n = 1$ in case f is not defined for c_k. In our example, value E is therefore defined as $V = [0, 0, 0, 0, 1]$. Please note that this encoding is used for any new categorical value. If another value F occurs, the encoding is also $V = [0, 0, 0, 0, 1]$.

One-over-n Encoding The idea for one-over-n encoding is to provide an uninformative input to a model, similar to zero encoding, but to keep the sum of all elements of the input vector to 1, i.e. $\sum_{j=0}^{n-1} v_j = 1$, where $v_j \in V$. Instead of 0, we assign to each $v_j \in V$ a value of $\frac{1}{n}$, i.e. $V = [v_0, v_1, ..., v_{n-1}]$ where $\forall v_j \in V : v_j = \frac{1}{n}$. Categorical value E in our example would therefore be encoded as $V = [0.25, 0.25, 0.25, 0.25]$ since $n = 4$.

Distribution-based Encoding In distribution-based encoding we provide for each element in the one-hot encoded vector the relative frequency of the corresponding feature value. In order to do so, we define a function $\text{count}(c_j)$ that counts for each value $c_j \in C_{\text{train}}$ the number of occurrences in the training set, i. e. $\text{count} : C \to \mathbb{N}_0$. The sum of all events is defined as $M = \sum_{j=1}^{n} \text{count}(c_j)$ where $c_j \in C_{\text{train}}$. For any $v_j \in V$ we retrieve the distribution-based encoding with $v_j = \frac{\text{count}(f^{-1}:i_j \to c_j)}{M}$ where $i_j \in I$ and f^{-1} is the inverse of f such that we retrieve the original categorical value given an index of the one-hot encoding. In other words, we count for each categorical value how often it occurs and then divide it by the total number of events in the training set.

Let's assume that $\text{count}(C_{train})$ defines the following mappings: $A \to 100, B \to 200, C \to 400, D \to 300$. Encoding previously unseen value E gives the following one-hot encoded vector: $V = [0.1, 0.2, 0.4, 0.3]$. This encoding has the same property as one-over-n encoding such that $\sum_{j=0}^{n-1} v_j = 1$.

Random Encoding To benchmark the other encoding techniques, we define random encoding which we assume to produce less accurate results. The one-hot encoding is defined as $V = [v_0, v_1, ..., v_{n-1}]$ where we sample uniformly and independently any $v_j \in V$ in the value range of $[0, 1)$ where $v_j \in \mathbb{R}$.

6 Experimental Setting

In our experiments we use XGBoost and LSTM's, representing state-of-the-art classical and deep learning methods, as outlined in Sec. 2. XGBoost has the advantage of fast computations with competitive performance. LSTM neural networks are computationally demanding, but show high prediction quality.

We trained the models with two features: Activities and a time-related feature. Activities represent the categorical feature in our study due to its consistent presence in all event logs. The time related feature is numerical and indicates for an event the time that has passed since the process has started.

We use six publicly available event logs[6] which show an occurrence of new activities over time. We preprocessed them such that they only contain completed cases and split them temporally into training, validation and test sets. 64% of the oldest cases, determined by the timestamp of the first event, are used for training, the next 16% represent the validation set and the last 20% of cases are used for testing. This is close to a real-life situation in which new activities occur over time. The datasets as well as statistics on the test sets relevant to our study are listed in Tab. 2.

To encode the activity feature we use one-hot encoding. We determine the number of activities for one-hot encoding on the training set solely. If a new activity occurs in the validation set we encode it as "unknown". We have not considered additional values from the validation set for one-hot encoding during training since this would effectively lead to dummy encoding for these activities.

[6] Available at https://data.4tu.nl/

To avoid this danger of mixing different encoding techniques, we defined "known" activities strictly on the training set. Since we apply this approach consistently to all models, it allows for a fair comparison.

Table 2. Test Set Statistics. First row: Total number of prefixes in the test set. Second row: Number of prefixes for which at least one new activity is present. These are the samples which are considered in our experimental evaluation. Third row: Average proportion of events which are previously unseen in the affected prefixes.

Dataset	2015_1	2015_2	2015_3	2015_4	2015_5	helpdesk
Num. Prefixes	5,827	8,526	11,526	4,699	10,896	3,415
Num. Affected Prefixes	5,104	3,640	4,777	1,516	5,730	207
Avg. Percentage of New Activities per Affected Prefix	16.41	4.75	4.01	5.10	5.67	29.04

We trained two variants of XGBoost models. One with with aggregation encoding and one with last-state encoding. Aggregation encoding embeds the relative frequency of how often an activity is present in the prefix and the mean of the time-related feature over all events in the prefix. For last state encoding we applied one-hot encoding to the last activity and appended its associated time-related feature. We used the parameters of the standard implementation of the XGBoost package. The only exception is the loss function, for which we used the mean absolute loss instead of a square error loss.

For the LSTM model, we defined a fixed setting with three LSTM layers with a hidden size of 100 neurons each. As optimizer we use NAdam with a learning rate of 0.001. The hyperparameters are determined based on the insights from prior studies [8], [13]. The input is provided as tensor-encoded vector [13] and we use the mean absolute error (MAE) as optimization metric. For the implementation we built on a generic codebase by Liessmann et al. [5][7].

We evaluated all results with the mean absolute error (MAE). Furthermore, we trained each model variant five times and calculated the average MAE achieved over the five runs. We repeated training because random factors like the order of the data can influence a model's convergence during training. To average out random effects we train and evaluate each model five times.

We evaluated two baseline models to cover Solution 3. as presented in Sec. 4. The first baseline by van der Aalst et al. [3] simply predicts half of the average runtime of traces in the training set. The second baseline by Ceci et al. [4] uses the time-related feature and predicts the difference between the average runtime of traces in the training set and the time that has passed for the last event of a prefix. We denote the approaches as *baseline1* and *baseline2*, respectively.

Since previous work has shown that machine learning models outperform simple baselines, we focused our analysis only on prefixes of the test set where a previously unseen activity is encountered for any event. In this way we avoid

[7] https://github.com/fau-is/ECIS24-TL4PM

that good predictions for prefixes without new activities average out potentially bad predictions when calculating the MAE.

7 Results and Discussion

The results of our case study are displayed in Tab. 3. The values show the mean MAE across the five runs. Bold numbers indicate the overall best model for a dataset. For the baselines, no encoding techniques are applied, therefore only one number is reported.

Table 3. Results of benchmark study. Abbreviations: Dst: Distribution; Dm: Dummy; N: One-over-n; Rnd: Randomg; Zro: Zero; b1: baseline1; b2: baseline2.

bpic2015_1					
	LSTM	XGB Agg	XGB Last	b1	b2
Dst	36.75	41.15	35.03	38.99	41.03
Dm	35.30	41.48	34.72		
N	**33.03**	39.80	34.59		
Rnd	40.81	134.33	35.13		
Zro	34.82	41.53	35.17		

bpic2015_2					
	LSTM	XGB Agg	XGB Last	b1	b2
Dst	65.35	65.02	**31.02**	48.89	68.40
Dm	72.55	68.82	31.05		
N	64.10	64.37	31.14		
Rnd	44.90	441.05	31.05		
Zro	68.3	65.18	31.23		

bpic2015_3					
	LSTM	XGB Agg	XGB Last	b1	b2
Dst	6.85	7.72	7.42	29.11	22.91
Dm	**5.45**	8.61	7.41		
N	5.61	7.35	7.41		
Rnd	5.88	92.68	7.39		
Zro	5.64	10.69	7.46		

bpic2015_4					
	LSTM	XGB Agg	XGB Last	b1	b2
Dst	49.18	46.08	44.12	44.85	58.41
Dm	44.00	46.25	44.08		
N	47.07	46.29	43.95		
Rnd	**37.58**	60.46	44.40		
Zro	48.53	46.20	44.79		

bpic2015_5					
	LSTM	XGB Agg	XGB Last	b1	b2
Dst	41.55	36.50	34.68	40.80	42.11
Dm	35.58	36.43	34.69		
N	**34.51**	36.94	34.78		
Rnd	37.82	244.45	34.81		
Zro	34.86	36.81	34.72		

helpdesk					
	LSTM	XGB Agg	XGB Last	b1	b2
Dst	**3.70**	5.06	3.98	8.72	7.30
Dm	3.97	4.76	3.98		
N	3.85	4.76	3.98		
Rnd	5.01	7.42	3.98		
Zro	3.88	4.76	3.98		

Comparing Machine Learning Methods to Baseline Methods The overall best model for a given dataset outperforms the two baseline models each time. A variant of the LSTM model is the overall best model for the datasets helpdesk, bpic2015_1, bpic2015_3, bpic2015_4, and bpic2015_5, however, with changing one-hot encodings. For bpic2015_2, XGBoost (Last) performs best, whereas most variants of LSTM and XGBoost (Agg) are outperformed by baseline1. This shows that machine learning models are generally capable of handling previously unseen categorical values efficiently, compared to simple baseline models. However, further research is required to determine the best suited model type for a given dataset. We have not found any insights based on specific dataset characteristics which explain the best methods for a given dataset.

Robustness of Machine Learning Models to Encoding Techniques The least deviation in MAE across different encoding techniques is shown by XGBoost (Last). The numbers in Tab. 3 are close to each other for each dataset. XGBoost (Agg) produces the worst results with random one-hot encoding. One-over-n encoding is the best approach for XGBoost (Agg) on 4 datasets and close to the best-performing approach for bpic2015_3 and bpic2015_4. For LSTM the results are less clear. Random encoding is the worst approach for the datasets helpdesk and bpic2015_1. In Tab. 2 we show the average percentage of new activities present in prefixes of the test set for which at least one new activity is occurs. It shows the highest numbers for the datasets helpdesk and bpic2015_1 and it is a result of previously unseen activities being mainly present at the end of traces. Therefore, we conclude that random encoding has a negative effect in prefixes with many new activities. However, random encoding also produces the best results with considerable distance for the datasets bpic2015_2 and bpic2015_4. We could not not identify any unique dataset characteristic for these datasets which differentiates them from others. Leaving random encoding aside we found one-over-n encoding delivering stable results for LSTM. A future challenge is to investigate the results obtained with random encoding further to allow more specific recommendations to practitioners.

Comparison of Encodings over all Machine Learning Methods There is no single approach that is clearly the best. Across all datasets, regardless of the machine learning method, each encoding produces once the best result, except for one-over-n encoding, which is best on two datasets (bpic2015_1, bpic2015_5). However, we observed that one-over-n encoding produces for each model consistently results which are competitive to the best performing one. The only exception is given for LSTM when random encoding is best.

Discussion To minimize the overall risk of low performance with new categorical values, we recommend using one-over-n encoding. However, the type of selected machine learning model is difficult to determine in advance. There are scenarios in which one-over-n encoding can lead to low performance if the right machine learning model is not chosen. The relatively good performance of last state encoding indicates that the time-related features as well knowledge of the last event are the most important features for accurate predictions, whereas knowledge about all events of a prefix adds additional minor information for predicting the remaining time. Providing complete traces could even lead to overfitting, which is a possible reason for the good performance of the random encoding for LSTM on datasets bpic2015_2 and bpic2015_4. For bpic2015_4 overfitting might even apply to the last activity where the performance of all methods is close to the baseline models. This suggests that mainly the time-related feature is informative in this case. For future studies we recommend ablation studies to understand the importance of individual features such that the effect of new categorical values can be isolated in more detail.

Besides these challenges, we have found that machine learning models can handle newly occurring categorical feature values and outperform simple baseline models. Therefore, it is a viable alternative for practitioners to avoid additional maintenance and escalation overhead. One-over-n encoding showed stable results. Still, open questions remain to provide fully informed recommendations.

8 Conclusion and Outlook

In this work, we motivated the need for solutions when concept drifts, represented by previously unseen categorical values, occur over time. However, (re)trained models are not immediately available for predictions. We have discussed four solution strategies from a business perspective that can help practitioners decide on a strategy based on the needs of their end users. Next, we focused on technical solutions to maintain predictive services. We have presented five different one-hot encoding strategies and compared them in a benchmark study in combination with LSTM models and two variants of XGBoost with two simple baseline solutions. The results show that one-over-n encoding provides the most promising results.

However, this study only represents a first investigation of the topic; further research is required in the future to provide more practical insights. This includes tackling limitations of our study. For example, only a small number of event logs is available where new activities occur over time and most of them, namely the bpic 2015 datasets, relate to the same underlying process execution. Also, we focused our study only on two features, activity and a time-related feature, but the complex interplay of several categorical features where new values occur over time, but also numerical features, needs to be evaluated, especially in light of recent suggestions of extending contextual data sources even further, for example by using data from complete enterprise process networks instead of isolated processes, as motivated by Weinzierl et al. [14]. Furthermore, our statistical approach of training five versions of each model and calculating the average MAE on the test set of these five models does not allow for statistically significant tests. We highlight that our approach is more reliable than just training one model variant which is mostly done in related literature. However, more runs will give statistically more significant insights. Besides these limitations, further aspects can be investigated in future research. Different encoding techniques such as word2vec might be evaluated. Also, other model types and additional tasks for PPM, like next activity prediction or outcome prediction, can be assessed.

Overall, we have motivated the need for handling previously unseen categorical values during runtime. Practitioners need to decide if services are continued and in which form. Technical solutions exist, but have not been compared with each other previously. Our study suggests that these technical solutions can provide reliable performance which are better than simple baseline solutions. However, clear recommendations are hard to give at this point. To close this gap, more research is required in the future.

Acknowledgments. This study was supported by Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) (grant number 456415646).

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Aalikhani, R., Fathian, M., Rasouli, M.R.: Comparative analysis of classification-based and regression-based predictive process monitoring models for accurate and time-efficient remaining time prediction. IEEE Access (2024)
2. van der Aalst, W.M., Carmona, J.: Process mining handbook. Springer Nature (2022)
3. van der Aalst, W.M., Schonenberg, M.H., Song, M.: Time prediction based on process mining. Information systems **36**(2), 450–475 (2011)
4. Ceci, M., Lanotte, P.F., Fumarola, F., Cavallo, D.P., Malerba, D.: Completion time and next activity prediction of processes using sequential pattern mining. In: Discovery Science: 17th International Conference, DS 2014, Bled, Slovenia, October 8-10, 2014. Proceedings 17. pp. 49–61. Springer (2014)
5. Liessmann, A., Wang, W., Weinzierl, S., Zilker, S., Matzner, M.: Transfer learning for predictive process monitoring. ECIS 2024 Proceedings **4** (2024)
6. Maisenbacher, M., Weidlich, M.: Handling concept drift in predictive process monitoring. In: 2017 IEEE International Conference on Services Computing (SCC). pp. 1–8. IEEE (2017)
7. Mangat, A.S., Rinderle-Ma, S.: Next-activity prediction for non-stationary processes with unseen data variability. In: International Conference on Enterprise Design, Operations, and Computing. pp. 145–161. Springer (2022)
8. Navarin, N., Vincenzi, B., Polato, M., Sperduti, A.: Lstm networks for data-aware remaining time prediction of business process instances. In: 2017 IEEE Symposium Series on Computational Intelligence (SSCI). pp. 1–7. IEEE (2017)
9. Oyamada, R.S., Tavares, G.M., Junior, S.B., Ceravolo, P.: Enhancing predictive process monitoring with time-related feature engineering. In: International Conference on Advanced Information Systems Engineering. pp. 71–86. Springer (2024)
10. Pasquadibisceglie, V., Appice, A., Castellano, G., Malerba, D.: Darwin: An online deep learning approach to handle concept drifts in predictive process monitoring. Engineering Applications of Artificial Intelligence **123**, 106461 (2023)
11. Rama-Maneiro, E., Vidal, J.C., Lama, M.: Deep learning for predictive business process monitoring: Review and benchmark. IEEE Transactions on Services Computing **16**(1), 739–756 (2021)
12. Shiao, W., Ju, M., Guo, Z., Chen, X., Papalexakis, E., Zhao, T., Shah, N., Liu, Y.: Improving out-of-vocabulary handling in recommendation systems. arXiv preprint arXiv:2403.18280 (2024)
13. 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 Transactions on Intelligent Systems and Technology (TIST) **10**(4), 1–34 (2019)
14. Weinzierl, S., Zilker, S., Dunzer, S., Matzner, M.: Machine learning in business process management: A systematic literature review. Expert Systems with Applications **253** (2024)