IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

*Digital Object Identifier 10.1109/ACCESS.2017.DOI*

# Assessing the Performance of Remaining Time Prediction Methods for Business Processes

**JOHANNES ROIDER[1], AN NGUYEN[1], DARIO ZANCA[1] AND BJOERN M.ESKOFIER[1, 2], (Senior Member, IEEE)**

[1]Machine Learning and Data Analytics (MaD) Lab, Friedrich-Alexander Universität Erlangen-Nürnberg, Erlangen, Germany
[2]Translational Digital Health Group, Institute of AI for Health, Helmholtz Zentrum München - German Research Center for Environmental Health, Neuherberg, Germany

Corresponding author: Johannes Roider (e-mail: johannes.roider@fau.de).

**ABSTRACT** The prediction of the remaining time for business processes is a major task in predictive process monitoring (PPM). In the last years, various machine learning methods were introduced which reduced error levels steadily. However, the commonly applied metric for optimization and evaluation, the Mean Absolute Error (MAE), has limitations regarding its interpretability. In this work we introduce and evaluate the normalized Mean Absolute Error (nMAE) as an interpretable metric for model evaluation. It accounts for different kinds of label shifts, which are a special type of concept drift that can distort remaining time results. We investigate these concepts in a thorough benchmark study and use them to assess the current state of remaining time prediction for business processes. This includes the evaluation of four different baseline models, identifying the most accurate one. Furthermore, our study compares three different state-of-the-art methods, namely XGBoost, DA-LSTM, and PGT-Net. In contrary to prior studies we find that there is no significant difference in the performance between these models. Additionally, using the nMAE as evaluation metric we find that these models do not perform reasonably well on a range of event logs. Initial ideas for this behaviour are discussed and consolidated along with other findings from the case study into a comprehensive list motivating future research directions.

**INDEX TERMS** Business Process, Graph Neural Network, LSTM, Machine Learning, Predictive Process Monitoring, Process Mining, Remaining Time, XGBoost

## I. INTRODUCTION

PREDICTIVE process monitoring (PPM) methods utilize historical business process data in the form of event logs to train machine learning models (offline phase). The obtained models are applied in an online phase to make predictions about new incoming cases during daily operations. Typical tasks in PPM include next activity predictions, outcome predictions, and remaining time predictions [1].

Predicting the remaining time of business process instances is a service offering which has practical advantages for company internal as well as external user groups. It allows internal user groups to intervene in cases that are expected to run too long and it supports the optimization of resources in order to reduce costs [1]–[3]. Furthermore, an efficient communication of expected remaining times can help to increase satisfaction levels of external stakeholders, for example customers. To support these goals reliably, machine learning models providing predictions need to be thoroughly evaluated and constantly monitored to ensure consistent, high-quality predictions. For such evaluations it is crucial to choose suitable evaluation metrics.

In the literature, mostly the Mean Absolute Error (MAE) is used to optimize models as well as to evaluate their performance. The MAE has a minimum value of zero for a perfect model, but its maximum is not bounded and can be any arbitrary positive value. Therefore, evaluating a model based on the achieved MAE is only possible with supporting knowledge about the underlying process like the average case duration. This is in contrast to classification problems like next activity or outcome predictions, where metrics like Accuracy and F1-Score allow a precise first assessment of a model since these metrics are normalized in a range between 0 and 1. A main contribution of this work is the proposal of a normalized version of the Mean Absolute Error, called

normalized Mean Absolute Error (nMAE). It is interpretable with regard to its fit to a given test set and accounts for label shifts.

In order to derive the nMAE we introduce and discuss different types of label shifts which can occur between a training and a test set. We show that such shifts can have positive, negative, as well as neutral effects with respect to the MAE. We will perform this analysis with baseline models. Since there is no unique definition of a baseline model for remaining time prediction of business processes in the literature, we derive a simplistic baseline model based on theoretical properties in machine learning.

All of the aforementioned concepts are evaluated in an empirical case study comprising 16 publicly available event logs, four different baseline models and three different machine learning methods.

The case study evaluates state-of-the-art methods for remaining time prediction of business processes and summarizes the current state of research in that domain. We additionally address weaknesses of prior studies. This results in a summary of unaddressed research questions that can serve as a research agenda for future research.

Our work is summarized in the visual abstract in Fig. 1.

The remainder of this work is structured as follows: First, we discuss related literature, followed by in an introduction of important concepts needed throughout the rest of this work. Next, we introduce our novel concepts by defining a suitable baseline model, discussing different kinds of label shifts, and introducing the nMAE metric. This is followed by an empirical case study. The case study is guided and discussed based on five research questions. Next, we summarize our work and discuss open research questions. This is followed by an overview about the limitations of the conducted study and a final conclusion.

## II. BACKGROUND AND RELATED WORK

### A. REMAINING TIME PREDICTION METHODS

A large body of work is dedicated to improve methods for remaining time prediction of business processes. To structure the range of proposed methods, benchmark studies have compared previously introduced methods on a variety of datasets [4], [5]. Verenich et al. [4] compared several process-aware methods with classical machine learning methods, represented by XGBoost, and neural networks, represented by the DA-LSTM architecture which was originally introduced by Navarin et al. [3]. Rama-Maneiro et al. [5] focused on the comparison of neural network-based methods. Both studies found that DA-LSTM outperforms all other methods. More recently, graph-based neural networks have been proposed and the corresponding case studies suggest that they can outperform DA-LSTM's [6], [7]. Most recently, Elyasi et al. [6] introduced PGT-Net, a graph-based neural network, and showed that it outperforms not only DA-LSTM but also other graph-based neural networks. Historically, the development of remaining time prediction methods started with process-aware methods, which were outperformed by classical machine learning methods. These, in turn, were replaced by neural network architectures, specifically by the DA-LSTM model [8], and most recently by graph-based neural networks [6].

Despite the strong evidence of superior performance by neural networks, the relevance of classical machine learning methods was recently highlighted again due to advantages in interpretability and computational demand [9]. Recent work has focused on improving the predictive quality of such methods. One direction is focused on engineering dedicated input features, for example time-related features by Oyamada et al. [9] or inter-case features by Pourbafrani et al. [10]. Another direction is the evaluation of alternative optimization approaches, for example by suggesting a discretization of target labels followed by applying classification-based optimizations as proposed by Aalikhani et al. [11].

### B. EVALUATION METRICS

The MAE has emerged as a prevalent evaluation metric used to derive a final judgment about models' performance. The choice for the MAE has been justified in a few works. Verenich et al. [4] discuss it in closer detail and note that the range of values for the remaining time is "highly varying across cases of the same process, sometimes with values on different orders of magnitude." The commonly used metrics Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) are highly sensitive to large errors resulting from such outliers. For these metrics individual prediction errors are squared before they are averaged, which has the effect that large errors due to outliers are weighted heavily and therefore distort the result. In order to be more robust, Verenich et al. [4] favour the MAE instead. This metric takes the absolute value of prediction errors before averaging them. The effect of large errors due to outliers is reduced with this metric, compared to the MSE and RMSE. Furthermore, Verenich et al. [4] consider the Mean Absolute Percentage Error (MAPE) in their discussion, which they note to be skewed in situations where the ground truth label is close to zero. For the MAPE, individual absolute errors are divided by the underlying ground truth labels to retrieve an error as a percentage of the ground truth labels, and these percentages are then averaged. If a ground truth label is close to zero the percentage error for a sample might become very large due to the division by a very small number, which produces large outlier values that skew the MAPE metric. Most recent work [2]–[7] mainly follows this reasoning and regards the MAE as the most suitable evaluation metric to compare different models. Recent results and discussions in the domain are therefore largely based on the MAE.

While the MAE gives a good indication whether business targets can be met on average, it has no further explanatory power. For example, it does not indicate directly whether the result achieved is a good fit to the data, close to random guessing, or even overfitting. An evaluation metric that aims at such interpretability is the coefficient of determination [12]. However, the coefficient of determination explains re-

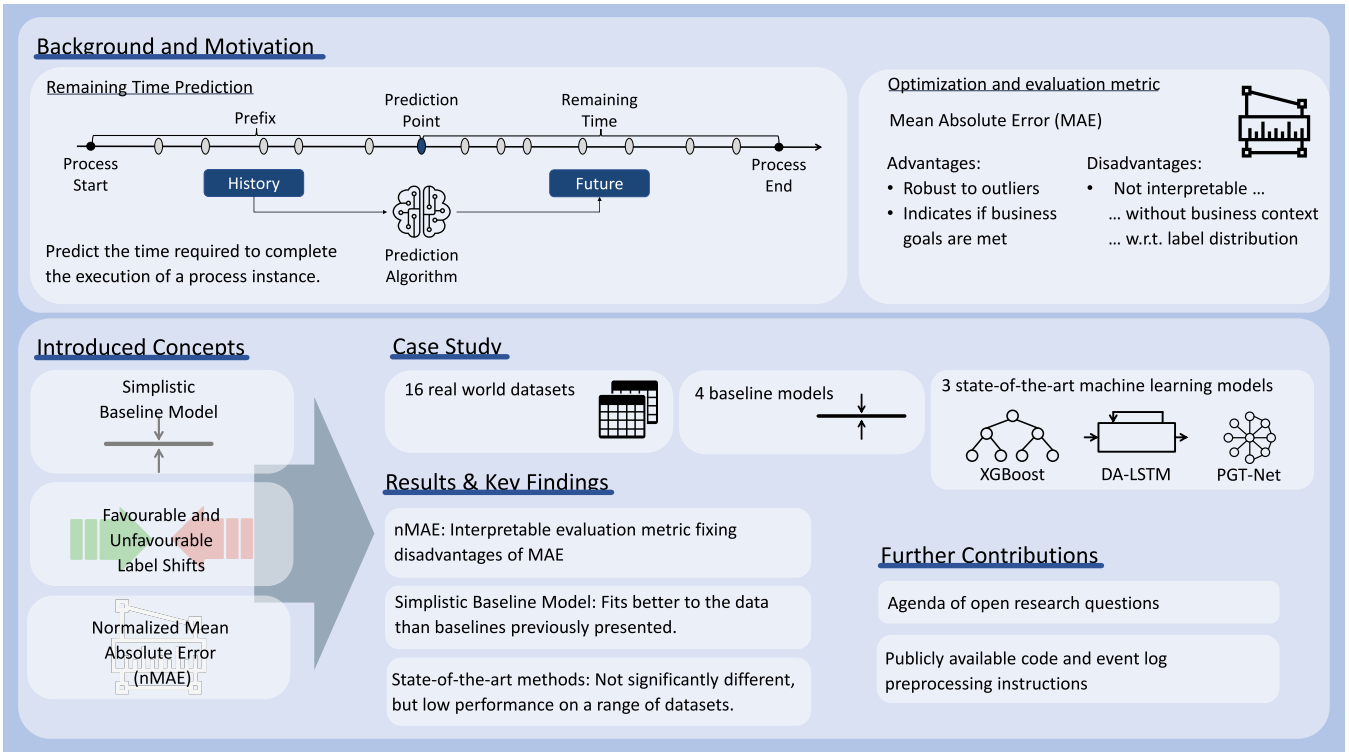


**FIGURE 1.** Visual Abstract. We introduce the normalized-MAE, define different types of label shifts and propose a baseline model. These are investigated in a case study including 16 datasets, 4 baseline models, and 3 machine learning models. Our contributions allow valuable insights for model evaluation and to assess the state of remaining time predicition in PPM. All insights are consolidated to motivate future research.

sults with regards to squared errors, whereas in remaining time prediction for business processes absolute errors are computed. While the coefficient of determination has been widely adopted, there is no established metric that allows an interpretation of results for datasets with outliers where the error is computed as an absolute deviation. Still, there are several attempts that present specific solutions. Chicco et al. [12] provide an overview about related work that proposes adapted versions of the coefficient of determination for absolute errors. Further work focuses specifically to variations representing robust solutions. McKean et al. [13], for example, introduce two versions to normalize the MAE. Renaud and Victoria-Feser [14] discuss different generic solutions and point out that these might be biased. They introduce an own metric which, however, requires the definition of additional weights. The drawback of such metrics is that they either incorporate specific assumptions regarding data distributions, they assume ordinary least squares or linear absolute deviation regression moels, or they introduce additional weights for their computation which are not trivial to determine. This makes the choice of the right metric challenging. In this work we revert to a rather simple formulation presented initially by McKean et al. [13]. We will show that is has desirable characteristics specifically for remaining time prediction of business processes which brings new insights.

### C. FEATURE REPRESENTATION

In machine learning, the type of machine learning method used determines the possible formats for data representation. Classical machine learning methods require the input to be of a fixed size vector. This is achieved by a tabular representation of the data. The best performing type of models for tabular data are tree-based methods like XGBoost [15]. Verenich et al. [4] summarize the different possibilities to represent event logs in a tabular form. These include aggregation encoding, last state encoding, and index-based encoding. In aggregation encoding, all events of a trace alongside with their features are encoded with statistical measures. For categorical features, which includes the activities, the occurrence of each value is counted. For numerical features different measures like the mean or standard deviation are calculated. The temporal order of information is therefore lost in aggregation encoding. For last state encoding, the last $m$ features of a prefix are considered and directly encoded, where $m$ is typically set to a value of one. In this case, only data obtained from the last event is used. For index-based encoding, each feature of a prefix with length $n$ is directly encoded, therefore the representation is lossless. While Verenich et al. [4] explicitly differentiate between last state and index-based encoding, we can regard them as the same method if $m = n$. The representations are not exclusive, that means the different encodings can be combined and used for a machine learning model.

For deep learning the data is encoded sequential-based. In the literature this is also denoted as tensor encoding [4]. Architectures like LSTM's [2], [3] or Convolutional Neural Networks (CNN's) [16] are typically used in combination with such a representation. While a tabular representation is two-dimensional, the tensor representation adds an additional temporal dimension. That means, each timestep is encoded explicitly, where the representation for one timestep is tabular. In other words, sequential-based encoding is a sequence of tabular data, where one table represents one timestep of a prefix. The information content for index-based encoding and tensor encoding is the same and both representations are lossless.

A recently applied type of encoding are graph representations. These allow to explicitly encode the control-flow perspective of an event log as a directed and attributed graph [6]. In this work, we will focus on PGT-Net, introduced by Elyasi et al. [6]. It is a neural network-based graph transformer. Each activity is represented as a node, and each directly follows relation of a prefix is represented as a directed edge. The number of occurrences for these transitions is added as edge weight. Additionally, categorical and numerical features of the target node of an edge are encoded as edge features for the last transition of a directly follows relation in the prefix. This representation is therefore not lossless.

In all of these approaches categorical features are represented as one hot encoded feature vectors. The only exception is the representation of activities for graph-based methods since they are encoded directly in a graph format. For neural networks, numerical input features are typically normalized. This is optional for tree-based methods.

### D. CONCEPT DRIFTS

Concept drifts are a well-known issue in the process mining literature [17]. In the context of predictive process monitoring most approaches are dedicated to next activity [18] and outcome prediction problems [19], [20]. Firouzian et al. [21] proposed an approach for concept drift adaptation in remaining time prediction with support vector regressors. All of these approaches actively tackle concept drifts by retraining or refining models in order to improve predictions. However, different types of concept drifts are not regarded in these studies.

Quionero-Candela et al. [22] provide an overview about possible types of concepts drifts from a generic machine learning perspective. Simple covariate shifts occur if there is a change in the probability distribution for the input features $P(X)$ between a training and a test set. This might occur for example when the relative frequency of process variants changes between a training and a test set. A prior probability shift changes the distribution in the target labels $P(Y)$. This might be a direct consequence of a change in $P(X)$ since the shift from a simpler to a more complex process variant might also lead to longer remaining times and therefore a shift in $P(Y)$. However, it could also arise from a change in the joint distribution $P(X;Y)$, for example by introducing a more efficient information system that allows to perform the same tasks in shorter times. Many possible reasons for changes in the label distribution $P(Y)$ are therefore possible. In this work we will focus on changes in $P(Y)$ per se, leaving possible reasons for that change aside. We will show that considering changes just in $P(Y)$ brings already valuable insights to the analysis of remaining time prediction methods.

## III. PRELIMINARIES

*Process Data*. The data used in PPM applications is represented by event logs. An event log is a collection of sequences. Each sequence, also called case or trace, consists of several events over time. Associated to these is always an activity (what has been done) and a completion timestamp (when has it been done). Additional context features are often provided. Examples include resources (who performed a task), costs or any other complementary information that might be useful to build a predictive model. The features (excluding the case identifier) are used as input for a predictive model to generate a prediction. Formally, we define these concepts as follows:

*Definition 1 (Event, Trace, Event Log):* An **event** $e$ is a tuple $(a, c, t, l, (d_1, v_1), ...(d_m, v_m))$, where $a$ defines an activity name, $c$ is a case identifier, $t$ the completion timestamp of event $e$ and $l$ denotes the lifecycle status of an event. $d_i$ denote the names of additional attributes. $v_i$ is the corresponding value for attribute $d_i$. $m$ is the number of additional features given.

A **trace** is a sequence of events $\sigma = \langle e_1, ..., e_n \rangle$ such that $\forall e_i, e_j \in \sigma; i, j \in [1, n] : j > i \wedge c(e_i) = c(e_j) \wedge t(e_j) \geq t(e_i)$.

An **event log** is a set of traces $L = \{\sigma_i : \sigma_i \in S, 1 \leq i \leq K\}$, where $S$ is the set of all possible traces and $K$ is the number of traces in the event log.

Table 1 shows an example of a trace from the publicly available credit requirements process dataset [23]. First, a credit application is registered. Next, different checks are performed in order to verify that the debitor is worthy of a credit. Lastly, the credit committee decides and the process is concluded with a final requirements review. Each of the activities is performed by different organizational resource groups. For most logs, as in this case, the *Lifecycle* column only contains the value *complete*. Some logs also differentiate the lifecycle status, for example by indicating when an activity is scheduled, when it has started, and when it was completed. In remaining time prediction, mostly the *complete* lifecycle is used and all events containing different values than *complete* are discarded. We follow this approach in our work.

The overall goal in remaining time prediction is to predict after every event the time it takes until the corresponding case is completed. The time of completion is denoted by the timestamp of the last event of a case. It is typically not included for evaluation since predicting that a case is already over is out of scope for such models [2]. To each label a prefix

**TABLE 1.** Example trace from credit dataset.

| Activity $a$ | Case ID $c$ | Completion Timestamp $t$ | Lifecycle $l$ | Resource |
|---|---|---|---|---|
| Register | 1 | 2014-04-02 08:00:48 | complete | System |
| Acceptance of requests | 1 | 2014-04-02 08:18:43 | complete | group1 |
| Collection of documents | 1 | 2014-04-02 09:47:48 | complete | group1 |
| Completeness check | 1 | 2014-04-02 11:05:04 | complete | group2 |
| Credit worth check | 1 | 2014-04-02 12:40:06 | complete | group3 |
| Collateral check | 1 | 2014-04-02 13:35:35 | complete | group4 |
| Credit committee | 1 | 2014-04-02 14:17:41 | complete | group5 |
| Requirements review | 1 | 2014-04-02 14:29:46 | complete | group6 |

is associated which contains information of all events from the start of the case up to the current event:

*Definition 2 (Prefix, Remaining Time):* The **k-prefix** $hd^k(\sigma)$ of trace $\sigma = \langle e_1, ..., e_n \rangle$ is a partial sequence from the first to the k-th event of a trace, e. g. $hd^k(\sigma) = \langle e_1, ..., e_n \rangle$, where $k \leq n$ and $n$ denotes the total number of events in a trace.

The **Remaining Time** $RT$ attributed to an event $e_j$ of trace $\sigma$ is the difference between the completion time of the last event of a case $t_{e_n,\sigma}$ and the completion time of the last event of a prefix $t_{e_j,\sigma}$, i.e. $RT(\sigma, e_j) = t_{e_n,\sigma} - t_{e_j,\sigma}$, where $j \leq n$ and $n$ denotes the last event of a trace $\sigma$.

*Evaluation Split.* An event log is represented in a way such that it can be used by a machine learning algorithm to predict the remaining time. Examples include aggregation encoding, tensor encoding, and graph encodings as outlined in Sec. II. Formally, a dataset $\mathcal{D}$ consists of a set of samples $X = \{x_1, x_2, ..., x_N\}$ and target labels $Y = \{y_1, y_2, ..., y_N\}$, where $N$ denotes the number of samples, $y_i$ is a scalar value and $x_i$ is the representation of a prefix in an arbitrary data format to be fed to a machine learning model.

The dataset is split into a training, validation, and test set for evaluation purposes, which results into the sets $\mathcal{D}_{\text{train}} = \{X_{\text{train}}, Y_{\text{train}}\}$, $\mathcal{D}_{\text{validation}} = \{X_{\text{validation}}, Y_{\text{validation}}\}$, and $\mathcal{D}_{\text{test}} = \{X_{\text{test}}, Y_{\text{test}}\}$, where $\mathcal{D}_{\text{train}} \cap \mathcal{D}_{\text{validation}} = \emptyset$, $\mathcal{D}_{\text{train}} \cap \mathcal{D}_{\text{test}} = \emptyset$, and $\mathcal{D}_{\text{validation}} \cap \mathcal{D}_{\text{test}} = \emptyset$. Furthermore, the split is performed such that prefixes of the same case always occur in the same set.

*Evaluation Metric.* To optimize and evaluate models, the Mean Absolute Error (MAE) is used as a metric. Formally, the MAE is defined as

$$\text{MAE} = \frac{1}{N} \sum_{i=1}^{N} |\hat{y}_i - y_i| \tag{1}$$

where $N$ is the number of samples, $\hat{y}_i$ is a model prediction, $y_i$ is the corresponding ground truth label, and the $|.|$ operator returns the absolute value of a scalar value.

*Probability Distributions.* The operator $P(.)$ defines the probability distribution of a one- or multi-dimensional input. $P(X_{\text{set}})$ defines the joint distribution of all input features for a specific set, where set $\in \{\text{train}, \text{validation}, \text{test}\}$. The univariate label distribution of a set is given as $P(Y_{\text{set}})$. The joint distribution of a set between the input features $X$ and the corresponding target labels $Y$ is given as $P(X_{\text{set}}; Y_{\text{set}})$. In predictive process monitoring, event logs are assumed to be created from an underlying unknown distribution function $P(\mathcal{D})$.

*Distribution Shift.* A distribution shift denotes a change in $P(.)$ between two probability distributions. In this work we focus on shifts in the label domain between a training and a test set. We use the terms *prior probability shift* and *label shift* interchangeably to denote this type of shift. Formally, it is denoted as: $P(Y_{\text{train}}) \neq P(Y_{\text{test}})$.

## IV. CONCEPTS TO ASSESS MODEL PERFORMANCE

### A. BASELINE MODEL

A baseline model is a simplistic approach that a machine learning method can be compared to. If the machine learning approach is better than the baseline, it shows that it's able to make meaningful predictions based on the input data and that it should be preferred over a simplistic approach. In the simplest case, the baseline method predicts a constant value.

#### 1) Existing Approaches

There is currently no consensus in the literature of remaining time prediction for business processes on the definition of a baseline. Early work defined baselines based on the average case duration in an event log. For example, Van der Aalst et al. [24] define the half of the average case duration of the cases in the training set as a baseline prediction. Ceci et al. [25] predict the difference of already passed time to the average case duration. In later work, approaches are compared to earlier methods without comparing the results to a simplistic baseline model [3], [8]. Elyasi et al. [6] group all training samples by prefix length and predict as a baseline the mean of the labels of a group.

#### 2) Theoretical Considerations

The training of machine learning models follows under the assumption that each training sample is sampled independently [26]. In predictive process monitoring one prefix along with its associated label represents one sample. Therefore, all samples of a training set should be utilized to determine the baseline model. The approaches by Van der Aalst et al. [24] and Ceci et al. [25] restrict their baselines to use only the maximum label of each trace, which corresponds to the individual case durations. However, they do not incorporate information how remaining times are distributed within cases. For example, events might occur in one event log predominantly directly after a case starts, which results in overall larger remaining time labels. For another event log, events might occur frequently directly before the case

ends, which results in many small remaining time labels. A machine learning model adjusts to such distributions to minimize the optimization metric. A baseline model should also follow such an approach.

Furthermore, a baseline should be chosen that is optimal with regard to the optimization metric used. It has been shown analytically that the optimal constant to predict is the mean of all labels in the training set when the optimization metric is the mean squared error (MSE). For the MAE, the median is the optimal constant to predict [27].

#### 3) Baseline Model

We define the median of all labels in the training set as a constant to be predicted by the baseline method since it takes all available labels into consideration, not only case durations, and since it is optimal with regard to the MAE metric.

Formally, it is defined as:

$$\hat{y}_{\text{baseline}} = \text{median}(Y_{\text{train}}) \tag{2}$$

where median$(.)$ is an algorithm to determine the median on a univariate input. In case the number of observations is even, we take the mean between the two values which are candidates for the median.

Next to following the theoretical guidelines, this baseline has further advantages over previously used baselines. The baselines used by Ceci at al. [25] and Elyasi et al. [6] require the usage of additional input features. The method by Van der Aalst et al. [24] requires knowledge about the mapping of prefixes and labels to the case id they belong to in order to filter out overall case durations. In that regard, our metric is simplistic since it requires only the labels of the training set and no further data. Furthermore, the method by Elyasi et al. [6] has practical limitations if prefix lengths occur in the test set which are longer than any prefix in the training set. In that case, the value to be predicted cannot be defined in a straightforward way. Our method does not encounter such a limitation.

Given the aforementioned reasons, we showed that using the median of the training labels as a baseline predictor has a strong theoretical foundation as well as practical advantages. Therefore, it should be regarded as a reasonable baseline model.

### B. EXPECTED WORST CASE PERFORMANCE

We define the MAE achieved by the baseline method as introduced in Sec. IV-A on the training set as the expected worst case performance. This concept allows a first assessment of practical problems at hand and we will use it later in Sec. IV-C for analyzing label shifts.

In practice, engineers are faced with the challenge that specific expectations on the performance of a model are given from business stakeholders. For example, statements in the form of "a model should on average not be off by X time units" might be given. Practitioners are additionally challenged with providing an estimation how feasible it is to achieve this goal. If requirements from a business perspective are close to the expected worst case performance, achieving the business goal is realistic. If the business requirement is further away from the baseline performance, more requirements regarding good data quality of training features as well as more sophisticated models to find a good mapping between the training features and the labels are required.

In case there are no concept drifts, the expected worst case performance will be the same as the achieved MAE of the baseline model on the test set. In practice, this is unrealistic to occur. The MAE on the test set might be higher or lower, depending on the distribution of the labels in the test set. Therefore, the expected worst case performance is a rough estimation under the assumption that no concept drifts occur.

While distribution shifts are a limiting factor about the expressivity of the expected worst case performance, we still think it can be beneficial to practitioners as a starting point to analyze the baseline performance and to compare it to the business requirements. This gives first insights about the complexity of the problem at hand.

To address the limiting factor of label shifts between a training and a test set, we introduce and discuss different types of label shifts in the next section.

### C. LABEL SHIFTS

As outlined in Sec. IV-B, differences in the label distributions between the training and test set might lead to deviations of the baseline from the expected worst case performance. This can lead to larger or lower errors on the test set. We present in this section first a definition of different label shifts, followed by a detailed explanation using artificial examples.

#### 1) Different Types of Label Shifts

If a label shift leads to a larger error on the test set than on the training set, we denote it as an *unfavourable label shift*. If a label shift leads to a lower error on the test set than on the training set, we call it a *favourable label shift*. If a shift is present which does not lead to differences between the MAE on the training and the test set, we denote it as a *neutral shift*.

Favourable label shifts lead to overly optimistic assessments about the performance of models. Good performance might be attributed to an efficient mapping of input features to target labels, whereas in reality an improved performance is caused by a shift in the label distribution.

Knowing about unfavourable label shifts could be advantageous when investigating bad model behaviour. If a model does not uphold its performance, unfavourable label shifts can be identified as one possible reason. Knowing the reasons for bad model performance can then help to define mitigation strategies.

#### 2) An Intuitive Example

In order to explain the concept of label shifts more intuitively we discuss them based on a fictional example, visualized in Fig. 2. The concept of favourable label shifts is emphasised in

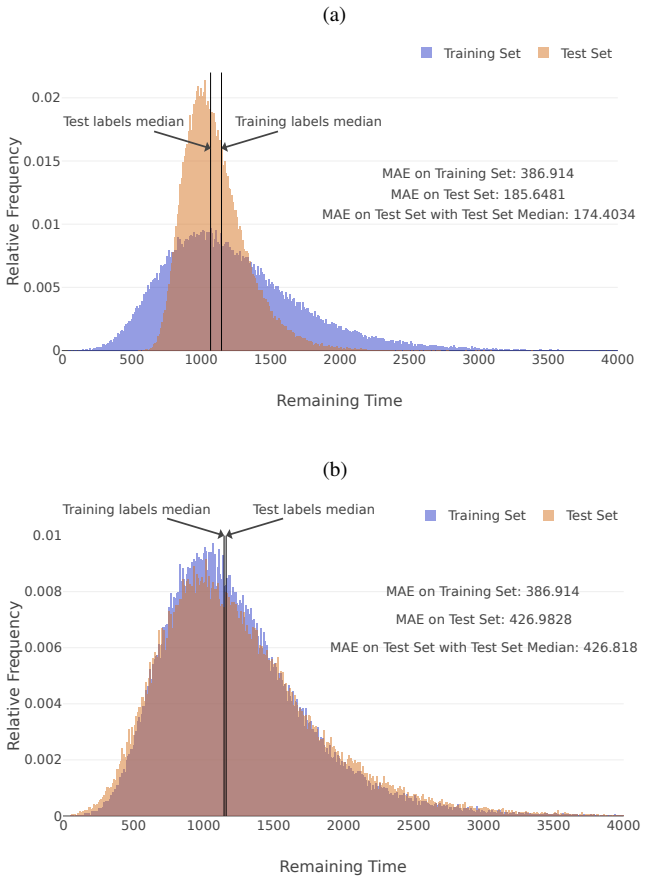


**FIGURE 2.** Histograms of label distributions (synthetic example). (a) Favourable label shift. (b) Unfavourable label shift.

Fig. 2 (a). The figure shows a histogram with the distribution of all labels in a training (blue) and a test (red) set. Using the median of the training set to make a prediction gives an error of $386.914$ time units on the training set and an error of $185.6481$ time units on the test set. The lower error on the test set indicates the favourable label shift. This is because the distribution of the test set centers closer around the median of the training set than the distribution of the training set. The MAE on the test set is therefore smaller than the expected performance, even though the value used for the prediction is not optimal with regard to the test set. The optimal constant determined on the test set would result in an MAE of $174.4034$ time units.

Fig. 2 (b) shows a scenario of an unfavourable label shift. The MAE is higher on the test set than on the training set since the test distribution is more spread out around the median than the training distribution. Please note that the distributions of the training set in Fig. 2 (a) and Fig. 2 (b) are the same, but the scale for the relative frequency changes for visualization purposes.

In the next section we will introduce a metric that accounts for label shifts to give a more realistic picture about the performance of a machine learning method for predicting the remaining time of business processes.

### D. NORMALIZED MEAN ABSOLUTE ERROR

A weakness of the MAE is that it's not normalized. Interpreting the MAE depends highly on knowledge about the business context and underlying process behavior like average case durations. Additionally, comparisons of several models or at least to a baseline model are necessary to detect exceptionally good or bad performance. Building on the previous concepts, we discuss in this section a scale-independent metric that allows a first assessment of a model's performance without needing specific domain knowledge or a baseline model.

#### 1) Definition

Formally, we define the metric, denoted as normalized MAE (nMAE), as:

$$\text{nMAE} = \frac{\frac{1}{N}\sum_{i=1}^{N}|\hat{y}_i - y_i|}{\frac{1}{N}\sum_{i=1}^{N}|\bar{y} - y_i|} \tag{3}$$

where $N$ is the number of samples in a dataset to be evaluated, typically a given test set, $\hat{y}_i$ is the prediction of a machine learning model for the $i$-th sample of the dataset, $y_i$ is the corresponding ground truth label, $\bar{y}$ is the median of the dataset, and the $|.|$ operator returns the absolute value of a scalar value.

Normalizing the MAE is not a new concept in the literature as outlined in Sec. II. A version of the nMAE was first presented and discussed by McKean and Sievers [13]. In their evaluation the nMAE has shown to be susceptible to outliers when small samples sizes are used (less than $10$). This is typically not a concern in predictive process monitoring, where at least hundreds of samples are available for most datasets.

#### 2) An Intuitive Example

In order to explain the choice for normalization, consider the following example. In the example we discuss two reasonable alternatives for normalizing the MAE and we will show that they fail in adjusting to label shifts.

The first alternative is to normalize the MAE of a model on the test set by the MAE that the baseline model achieves on the training set. The second alternative is to normalize the MAE of a model on the test set by the MAE that the baseline model achieves on the test set. These are straightforward choices since they set the performance of a machine learning model in relation to a simplistic baseline model.

Consider Fig. 3. A baseline model is fitted to the training set and then applied in two scenarios with unfavourable label shifts. In both scenarios the MAE on the test set is around $427$ time units. This is larger than the expected MAE of $386.914$ time units. However, in the first scenario the distribution is squashed, but the medians of the training and test sets are similar. In the second scenario, the test set distribution is shifted to the right and the median between the training and

the test set varies more considerably. If the true median of the test set would be known already at training time, the error achieved on the test set would be 386.9918 time units in scenario 2. However, due to the shift and the usage of the median of the training set the error on the test set is 426.9662 time units. This is different to the first scenario. If the median of the test set would be known at training time, the MAE on the test set would still be 426.818 time units in scenario 1. Knowledge about the test distribution does in scenario 1 not lead to considerably better results than knowing the training median.

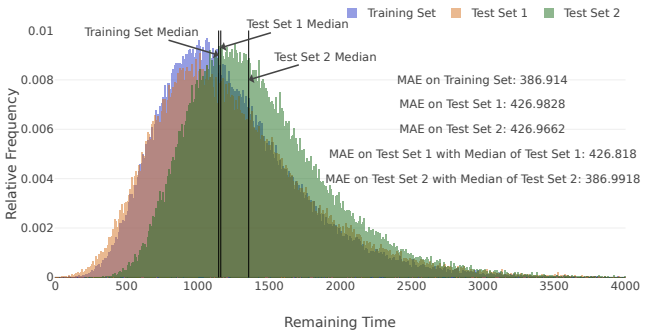


**FIGURE 3.** Different label shift scenarios (synthetic example).

Revisiting normalization alternative one, normalizing by the expected baseline performance of 386.914 time units would in both scenarios lead to a value of 1.19. For alternative two the performance of the baseline model would be indicated in both scenarios by a value of 1.0. The information of different distribution shifts in both scenarios is therefore lost. The nMAE as defined in Eq. 3 accounts for different types of shifts. In scenario 1 the nMAE gives a value of 1.00 and therefore indicates that the baseline model is still a good fit to the data. In scenario 2 we get an nMAE of 1.19. It shows that the baseline model is a bad fit to the test data since knowing the true median of the test set would lead to a considerably lower error. The bad fit is reflected by a higher value than 1.0 for the nMAE.

Table 2 summarizes these insights and shows that the nMAE as proposed in Eq. 3 is able to differentiate between label shifts while the two discussed alternatives for normalizing the MAE are not.

**TABLE 2.** Comparison of alternatives for nMAE

| | Normalization Alternative 1 | Normalization Alternative 2 | nMAE |
|---|---|---|---|
| Shift Scenario 1 | 1.19 | 1.00 | 1.00 |
| Shift Scenario 2 | 1.19 | 1.00 | 1.19 |

Another advantage of the nMAE over the two discussed alternatives is that it can be directly calculated just by using the test set labels. There is no need to utilize any information from the training set as in the two alternatives. This can be an advantage in practical settings where access is only given to a trained model and a specific dataset at hand without any access to the original training data.

### 3) Discussion

By design, the best possible nMAE that a baseline model as defined in Eq. 2 can achieve is a value of 1.0. If the labels in the test set and the labels in the training set follow the same distribution, i.e. $P(Y_{\text{training}}) = P(Y_{\text{test}})$, the median in both sets has the same value. Since the baseline predicts a constant, we can state that $\hat{y}_1 = \hat{y}_2 = ... = \hat{y}_N = \bar{y}$ (see Eq. 3). In that case the numerator and denominator have the same value and the nMAE becomes 1.0.

For any model that uses informative input features to make a prediction, the nMAE should be lower than 1.0. This is a desirable characteristic of our metric since it directly provides information about the information content of the training features and how well a machine learning model utilizes these.

Furthermore, the nMAE is directly proportional to the MAE. That means, if model A has a two times larger MAE than model B, the nMAE will also be two times larger. Therefore, the information provided by the MAE is also reflected in the nMAE.

In summary, the nMAE is constructed such that the following considerations are possible:

- If the value is 0.0, we have a perfect model
- If the value is 1.0, we have a random model. That means the model can approximate the median value of the test set for its predictions.
- If the value is between 0.0 and 1.0 the model is able to meaningfully utilize the input features.
- If the value is larger than 1.0, the model is not a good fit to the test data.

Test data in this context can either be an extra test split from available historical data, but also new incoming data collected in a productive system.

Please note that the nMAE allows a quick judgment whether a model performs good or bad, but it does not identify the specific reasons for bad model performance. Possible reasons for bad model performance are noisy or irrelevant input features, but also concept drifts where the model was trained on outdated data. The specific reasons need to be investigated with dedicated techniques. Still, the nMAE is a suitable indicator to assess the performance of a model in the first place.

## V. CASE STUDY

In Sec. IV we have introduced and discussed several concepts based on theoretically motivated considerations. In this section we aim to answer the question whether these contributions can also be confirmed empirically with real-world event logs. To do so, we perform a case study applying different baselines and state-of-the-art machine learning methods for remaining time prediction of business processes to publicly available event logs.

To guide the case study, we first formulate specific research questions to be answered. Next, we describe the datasets used and how we preprocessed them. This is followed by a description of the data split, and how input features are preprocessed for different machine learning methods. These are then described in the next steps. Lastly, we discuss our evaluation approach.

### A. RESEARCH QUESTIONS

The research questions to guide the case study are as follows:
*RQ 1:* Can we confirm empirically the theoretically motivated improvements of our baseline over other baselines?
*RQ 2:* Can we confirm the presence of favourable and unfavourable label shifts in real-world event logs?
*RQ 3:* Which additional insights can be generated on real-world event logs with the nMAE compared to the MAE?
*RQ 4:* How do commonly applied machine learning methods perform based on the insights generated by the nMAE?
*RQ 5:* How do commonly applied machine learning methods perform relative to each other?

Research questions 1 to 3 directly aim at the contributions presented in Sec. IV. Since the case study also represents a general investigation of machine learning methods and their application to a wide range of different event logs, we have defined research questions 4 and 5 in order to derive general insights about the current state of remaining time prediction for business processes.

### B. DATASETS

We use a collection of event logs which have been used previously in benchmarking studies. These are available in a public repository[1]. In this section, we give a short introduction to the datasets used in our case study.

*bpic2012* [28]: It is a collection of data about the approval processes for personal loans and overdrafts from a financial institution in the Netherlands. It contains three different subprocesses which are handled separately in our study. The corresponding logs are denoted as bpic2012a, bpic2012o, and bpic2012w. This is in agreement with prior studies [4].

*bpic2015* [29]: This set of event logs contains data about a building permit approval process from five different Dutch municipalities. For each municipality one event log is provided, denoted by bpic2015_1, bpic2015_2, bpic2015_3, bpic2015_4, and bpic2015_5.

*bpic2020* [30]: This dataset contains logs from a reimbursement process at a Dutch university. We consider four different event logs for domestic travel declarations, international travel declarations, the reimbursement process for prepaid travel costs and requests for payment. We denote them respectively as bpic2020_dd, bpic2020_id, bpic2020_ptc, and bpic2020_rp.

*credit* [23]: This dataset contains information about a process to check credit requirements at a bank in Bosnia-Herzegovina. It is a sequential control-flow representing the simplest event log in our case study.

*helpdesk* [31]: This event log contains data about the helpdesk ticketing process in an Italian software company. It covers the the whole lifetime of a ticket from opening to closure.

*hospital* [32]: This dataset covers the billing process for medical services of a hospital.

*sepsis* [33]: This event log represents the journey of patients through a hospital for which sepsis was diagnosed. It starts with initial registering of a patient and ends with a final discharge.

The described datasets result in a final set of 16 event logs which we evaluate in our case study.

### C. EVENT LOG PREPROCESSING

We include only traces which are completed since the target label is represented by the time difference from the last event in a prefix to the last event of the whole trace. This requires that the last event of a trace is part of the log. We highlight that we could not find suitable information for the datasets of the BPI 2015 challenge and bpic2012w how completed cases can be defined. We have checked submissions to the corresponding BPI challenges (see for example [34]–[36]) which agree that defining a definite start and/or end of cases is difficult to determine with the dataset descriptions provided. However, since these datasets were used previously to report remaining time prediction results, we included them based on datasets provided by Verenich et al. [4] who indicate in their work that they preprocessed the logs such that they only contain completed cases. In other words, for the BPI 2015 datasets and bpic2012w we used the same cases as used by Verenich et al. [5]. However, a documentation of what constitutes completed cases for these logs is not provided by them. In order to enable reproducibility of our research we provide the definition of what constitutes completed cases where we preprocessed the logs on our own at a publicly available repository[2].

For each trace, we only keep prefixes which have an associated remaining time larger than zero. This approach is also applied by previous work [2], [4]. If several events at the end of a case have the same timestamp and therefore a remaining time of zero, we remove all of them. Furthermore, the PGT-Net architecture used in our experiments cannot make predictions for the first event of a trace since it requires at least two events to build a graph [6]. Therefore, we only consider prefixes with two or more events for training and evaluation. This is in line with the study conducted by Elyasi et al. [6].

### D. DATA SPLIT

We apply a temporal Training-Validation-Test split with a ratio of 60%-20%-20%. The temporal ordering of cases is defined by the timestamp of the first event of a case. Weytjens and De Weerdt [37] discuss more elaborate unbiased splits in such a setting. However, this requires to remove cases where

[1]https://data.4tu.nl/

[2]https://github.com/RoiderJ/assessing_remaining_time_methods

events overlap between the training and test set. This results in a large number of cases to be removed for some datasets which is impractical in our setting. To be consistent across all datasets, we allow such overlaps for all datasets. To avoid any data leakage, we do not use any inter-case features.

We do not follow a k-fold cross validation approach since we regard the evolvement of process executions over time as a natural characteristic which needs to be tackled in practical settings. Models are always trained on historical data and then applied to future data. A temporal split into training, validation, and test sets resembles such a scenario best, whereas a cross validation evaluation would cancel out effects due to concept drifts and other relevant challenges faced in practice. This view is also supported and discussed in more detail by Weytjens and De Weerdt [37].

### E. FEATURE PREPROCESSING

As training features we use the activity (control-flow perspective) as well as different timestamp related features like day of the week (categorical), time since process start, time since midnight, and time since Sunday (numerical). Using timestamp related features is common practice in the literature and is applied by Tax et al. [2] and Navarin et al. [3] amongst others. If additional categorical or numerical features are available for a log we use them. Furthermore, we set categorical values which occur in less than 2% of the cases in the training set to a combined category denoted as "others".

We do not use any inter case features. On one hand it would lead to a bias in the test set as discussed by Weytjens and De Weerdt [37]. On the other hand, we do not know whether the logs really contain all relevant cases or just a subset from a database. In the latter case, such features do not reflect a realistic picture for example of the current workload for that process since the dataset is biased by a sample selection bias from a database.

We consider only activities which are completed according to the *lifecycle* attribute in the logs. This is in line with prior studies [4], [5]. The original implementation for the PGT-Net considers all transitions. However, many of the datasets contain only the *complete* transition anyway, including the BPIC 2015 datasets on which PGT-Net reported the largest improvements [6]. Since the graph representation does not change for these logs by allowing only completed transitions, we do not expect a disadvantage to this method.

We devote special attention to categorical data, represented by the activity (control-flow) and categorical features. A practical challenge is the handling of new activities or other categorical features in the validation or test set, or later in a productive environment. Categorical features might be encountered which were not present in the training set. This problem is underrepresented in the literature. We are aware of only one study that investigates this issue for next activity prediction. Mangat and Rinderle-Ma [38] define two possible solutions of which they evaluate an approach they call void encoding. This requires a one hot encoding of categorical features. If a previously unseen categorical value occurs, all values in the one-hot encoded vector are simply set to zero in void encoding. Prior studies for remaining time prediction do not explicitly denote how such situations are handled. There is the danger of data leakage issues when determining the number of categorical values on the complete dataset before applying a split into training, validation, and test sets.

For the methods applied in our study, categorical features are encoded as one hot encoded vectors. There is only one exception for PGT-Net, which represents activities as nodes in a graph structure. All other categorical features are also one hot encoded for PGT-Net. Therefore, we apply void encoding for all categorical features except the activity. Handling previously unseen activities for PGT-Net represents a challenge since there is no clear way on how to represent previously unknown events for graph structures. From a practical perspective, we regard it as reasonable to exclude new activities from making a prediction. Since these activities were not used for training, making predictions using them has a high risk of unexpected behavior. Therefore, we remove all events from the validation and test set which contain an activity that is not part of the training set.

All numerical features and the labels are normalized by dividing them by the absolute maximum value present in the training set.

### F. MACHINE LEARNING MODELS

Due to the large variety of presented methods in the past, a reasonable subset of models needs to be chosen for our case study to keep the computational effort manageable. We have chosen three methods as representative state-of-the-art methods. The choice is based on insights of prior work, where possible based on benchmark studies containing a larger number of methods and datasets. As outlined in Sec. II, deep learning methods and graph neural networks represent the latest developments providing most accurate predictions and should therefore be considered. Since these are computational demanding and limited in interpretability, we decided to include additionally a classical machine learning method.

The DA-LSTM model, introduced by Navarin et al. [3], has been evaluated in many works. The benchmark studies by Verenich et al. [8] and Rama-Maneiro et al. [5] both found DA-LSTM to outperform classical machine learning models as well as other deep learning methods, respectively. While other deep learning methods were introduced since then, for example by Bukhsh et al. [39], the DA-LSTM model has been evaluated in several works independently [5], [8]. Additionally, it is still commonly used and a generally accepted method in benchmark studies also in most recent work, for example by Elyasi et al. [6]. Therefore, due to the strong evidence of high performance of the DA-LSTM model in prior benchmark studies, as well as the widespread adoption including most recent work, we chose DA-LSTM as a representative for a state-of-the-art deep learning method for our case study.

Recently, graph-neural networks have emerged for remaining time prediction of business processes. Elyasi et al. [6] represents the most recent peer-reviewed work. They compare their proposed method, PGT-Net, not only to deep learning methods but also to another graph-based neural network, which they outperform. Therefore, we regard PGT-Net as the most relevant graph-neural network to be included in our case study.

Recent work by Oyamada et al. [9] motivates the advantages of classical machine learning methods due to their computational and interpretational advantages compared to neural networks. In real world scenarios, having an interpretable model might be valued more than having the most accurate one. Also, computational resources might be limited which makes deep learning and graph neural networks not feasible to use. In such scenarios, classical machine learning is a viable alternative. Verenich et al. [8] present a review about classical machine learning-based predictive process monitoring methods. They found that decision tree-based methods have an advantage in terms of interpretability and computational performance compared to other classical methods. The computational advantages are supported in the study by Oyamada et al. [9]. XGBoost, a decision tree-based method, has additionally shown to consistently outperform other classical machine learning methods according to the review by Verenich et al. [8]. This is also supported by recent research in machine learning outside of the predictive process monitoring domain. Data is represented in a tabular form in classical machine learning. It has been shown that tree-based methods can outperform even neural networks on tabular data [15]. Due to these reasons we chose XGBoost instead of other classical machine learning approaches as an interpretable and computationally efficient alternative to neural networks.

Additionally, to investigate research question 1, we apply different baseline estimators. These are the prediction of the training label mean, the training label median, which is our approach recommended as presented in Eq. 2, the average case runtime of cases in the training set as introduced by Van der Aalst et al. [24], as well as the difference between already passed process time to the average case runtime as used by Ceci et al. [25].

Applying XGBoost, we build on work by Verenich et al. [4] and made our implementation based on their publicly available repository[3]. Upon inspection we realized that their models were optimized using the standard predefined optimization metric by the XGBoost package [40] which is the squared error. This is not the optimal optimization metric for the MAE [27]. Therefore, we trained two versions for XGBoost, one in the original implementation by Verenich at al. [4] using the squared error as loss function, and one version using the absolute error as objective since this should, theoretically, lead to a better minimization of the MAE. To encode the data, we used *combined* encoding as introduced by Verenich et al. [4]. It encodes case features directly, applies aggregation encoding to represent the trace history of dynamic features, and it additionally provides the feature values from the most recent event (basically a combination of aggregation and last state encoding). We only train with one bucket. That means all samples are fitted with the same model. For hyperparameter tuning we use the same parameters as tuned by Verenich et al. [4], but a smaller range of values. An overview about the exact setting is shown it Table 3.

For the DA-LSTM model [3] we built on the Pytorch [41] implementation by Elyasi et al. [6]. However, we adjusted the implementation to reflect the original implementation in Tensorflow [42] by Navarin et al. [3] more closely. This includes the initialization of weights as well as the usage of recurrent dropout which is by default different between Pytorch and Tensorflow. We tune the number of layers as well as the number of neurons per LSTM layer (see Table 3). The hyperparameters are based on values reported by Navarin et al. [3]. For the learning rate we chose the standard setting used by Navarin et al. [3] as well as Elyasi et al. [6] with Nadam optimizer and a learning rate of 0.001. We apply early stopping if the performance on the validation set does not improve for 20 consecutive epochs.

For PGT-Net we used the implementation provided by Elyasi et al. [6]. It is based on the generic graph-based neural network architecture GraphGPS introduced by Rampášek et al. [43] which was extended such that event log data is represented in the correct format. GraphGPS offers various choices for different positional and structural encodings. Elyasi et al. [6] used different encodings, but the exact hyperparameter settings and their influence on expected model performance are not provided by them. We observed that using the GPS Graph Transformer in combination with Laplacian positional encoding (LapPE) and random walk structural encoding (RWSE) was the predominant architecture choice across the event logs investigated in their study. Therefore, we restrict our setting to this architecture and tune the corresponding hyperparameters. The tuned hyperparameters are listed in Table 3. All other settings for optimization like optimizer and learning rate were taken from the given implementation. Since we observed upon first tests that the best performance on the validation set was reached within the first few epochs, we also apply early stopping if the performance on the validation set does not improve for 20 consecutive epochs.

The maximum number of epochs for the neural network-based methods is set to 200 and we trained them with a batch size of 128.

### G. EVALUATION APPROACH

We first apply hyperparameter tuning and search the best parameters in a grid search approach. We tune on the training set and choose the best configuration based on the minimum MAE achieved on the validation set. All of the methods evaluated in our study do not have a deterministic training behavior. Instead, the convergence of the models depends

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

**TABLE 3.** Hyperparameters for Grid Search Procedure

| Method | Hyperparameters | Values |
|---|---|---|
| XGBoost | n_estimators | 100, 500 |
| | learning_rate | 0.1, 0.01 |
| | subsample | 0.3, 0.7 |
| | colsample_bytree | 0.3, 0.7 |
| | max_depth | 2, 6 |
| DA-LSTM | number_neurons | 50, 100, 150 |
| | number_layers | 1, 2, 4 |
| PGT-Net | Positional Encoding Dim | 2, 4, 8 |
| | Positional Encoding Times | 4, 8, 16 |

on random influences. For XGBoost, features that determine a split are chosen randomly, as well as the values for the specific splits. Weights for neural networks, which includes DA-LSTM and PGT-Net, are initialized by sampling values randomly from a predefined probability distribution. Furthermore, samples are shuffled randomly for the composition of mini-batches. All of these factors determine the convergence of a model which results therefore in slightly different model performance with a different random seed. To investigate the stability of methods with respect to random influences we train 10 models based on the best hyperparameter configuration found. For a given best configuration, we train 10 models with different seed, each time on the training set and using the validation set for early stopping of the neural network-based methods. With each of the 10 models we predict then the test set once. Finally, we report the average of the 10 MAE and nMAE metrics obtained from the 10 models. We are aware that 10 runs are a small sample size for statistical evaluations. However, it allows more insights compared to a commonly applied approach of only one run. Due to the computational requirements on the whole case study, we regard 10 runs as a reasonable tradeoff between accounting for random training factors and computational effort.

## VI. RESULTS

In this section we discuss the results of the case study. The discussion is guided by the research questions introduced in Sec. V-A.

*RQ1: Can we confirm empirically the theoretically motivated improvements of our baseline over other baselines?*

In Sec. IV-A we motivate that a baseline model should consider the independency assumption in machine learning as well as the underlying optimization metric. This is reflected in our baseline method as defined in Eq. 2. Theoretically, it should therefore outperform other baselines in terms of MAE.

Table 4 shows the MAE of different baselines on the training and the test sets used. To compare the baselines we perform a Nemenyi test [44]. The corresponding critical differences plots are shown in Fig. 4. Our proposed baseline significantly outperforms all other baselines on the training set (Fig. 4(a)). On the test set we significantly outperform the method by Ceci at al. [25] and the baseline using the label mean as a prediction (Fig. 4(b)). The difference to the baseline by Van der Aalst [24] is not significant on the test set. Still, our method has the highest average rank.

We regard our method therefore better suited as a baseline predictor than the other baselines due to the following reasons: 1) It significantly outperforms all other baselines on the training set and has the highest average rank on the test set, 2) It is simplistic in the sense that it just predicts a constant instead of calculating a dedicated prediction value for each sample, and 3) It does not require any additional information from the input features but can be calculated just from the training labels.

Based on these findings we conclude that the theoretically motivated improvements of our baseline hold true in an empirical setting.

**TABLE 4.** MAE in days of different baseline methods. The first number is the MAE on the training set, the second number is the MAE on the test set.

| Dataset | Baseline by [24] | Baseline by [25] | Label Median | Label Mean |
|---|---|---|---|---|
| bpic2012a | 11.613<br>7.913 | **10.371**<br>**7.319** | 10.379<br>7.771 | 10.619<br>8.641 |
| bpic2012o | 11.613<br>**6.464** | 10.371<br>7.080 | **10.379**<br>6.750 | 10.619<br>7.927 |
| bpic2012w | 9.767<br>6.596 | 9.783<br>**6.476** | **9.763**<br>6.650 | 10.181<br>7.922 |
| bpic2015_1 | 51.300<br>**40.702** | 57.675<br>43.771 | **47.671**<br>44.768 | 54.781<br>41.236 |
| bpic2015_2 | 80.525<br>63.083 | 94.776<br>82.099 | **79.585**<br>**59.494** | 87.495<br>76.439 |
| bpic2015_3 | 36.090<br>27.631 | 39.705<br>28.466 | **26.455**<br>**15.716** | 34.214<br>25.340 |
| bpic2015_4 | 54.493<br>45.245 | 58.724<br>51.903 | **53.727**<br>**44.723** | 56.848<br>46.576 |
| bpic2015_5 | 37.911<br>45.145 | 48.676<br>56.178 | **37.217**<br>**43.840** | 43.300<br>50.685 |
| bpic2020_dd | 4.023<br>4.431 | 5.611<br>5.3851 | **4.007**<br>**4.428** | 4.371<br>4.670 |
| bpic2020_id | 34.448<br>26.549 | 31.440<br>33.969 | **25.450**<br>**16.952** | 30.177<br>21.638 |
| bpic2020_ptc | 18.859<br>13.222 | 22.073<br>19.345 | **15.168**<br>**9.009** | 18.849<br>13.211 |
| bpic2020_rp | 4.654<br>5.546 | 5.961<br>6.228 | **4.653**<br>**5.536** | 5.156<br>5.809 |
| credit | 0.470<br>0.481 | **0.351**<br>**0.349** | 0.355<br>0.374 | 0.447<br>0.459 |
| helpdesk | 15.382<br>9.847 | **7.201**<br>**7.0225** | 9.119<br>12.090 | 9.157<br>12.803 |
| hospital | 72.532<br>60.902 | 66.925<br>75.586 | **53.906**<br>**43.450** | 60.515<br>48.960 |
| sepsis | 43.612<br>22.425 | 51.545<br>35.291 | **42.151**<br>**17.663** | 52.787<br>37.420 |

*RQ2: Can we confirm the presence of favourable and unfavourable label shifts in real-world event logs?*

In Sec. IV-C we have introduced the concepts of favourable and unfavourable label shifts. Favourable shifts are present if the baseline shows a lower MAE on the test set than on the training set. Unfavourable label shifts are

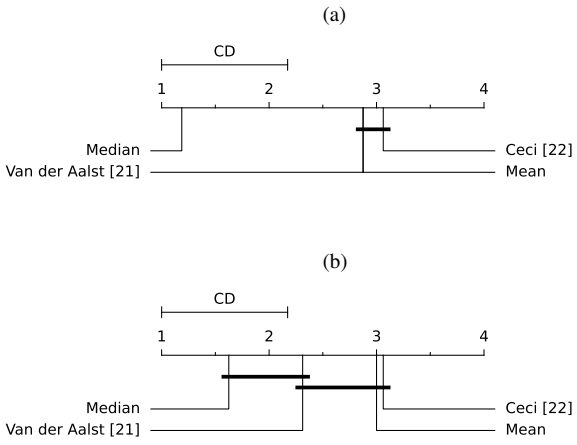


**FIGURE 4.** Critical differences plots of baselines. (a) Comparing average rank of baselines on training set. (b) Comparing average rank of baselines on test set.

characterized by a higher error of the baseline on the test set than on the training set. Since these are new concepts in the related literature, we aim to show that they can be identified in real-world datasets.

For better comprehensibility, we have plotted the results of our baseline based on the median from Table 4 in Fig. 5. It plots the MAE on the training set against the MAE on the test. Datasets which lie on the diagonal in that graph represent a neutral label shift. Points above the diagonal are examples of unfavourable label shifts. Points below the diagonal show a favourable label shift between the training and test set. We observe unfavourable label shifts for the datasets bpic2020_dd, bpic2020_rp, credit, and helpdesk. For all other datasets we observe favourable label shifts.

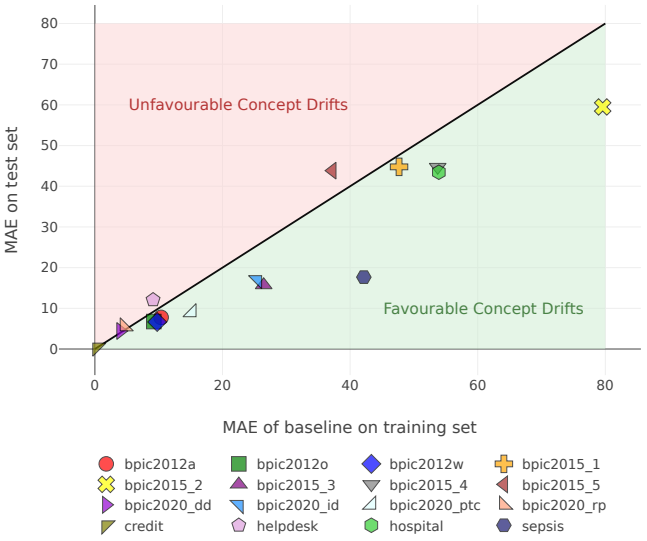


**FIGURE 5.** MAE of baseline on training and test set for different event logs.

To further illustrate this with the same visualizations as in Sec. IV-C, we plot the specific label distributions in the training and test sets for three event logs in Fig. 6. The helpdesk dataset is an example for an unfavourable label shift. The median of the labels in the training set is different to the median of the labels in the test set such that the MAE of our baseline is also higher on the test set than on the training set. For the credit dataset we observe an almost perfect overlap between the training and test distribution and the MAE of the baseline between the training and test set is close together. This represents a situation close to a neutral label shift. In the sepsis dataset many outliers are present in the training set at the right side of the chart. In the test set less outliers are present such that the distribution centers more around the label median. The median between the training and the test set shifts slightly to the left. Still the median of the training set produces a lower MAE on the test set than on the training set. This represents a favourable label shift.

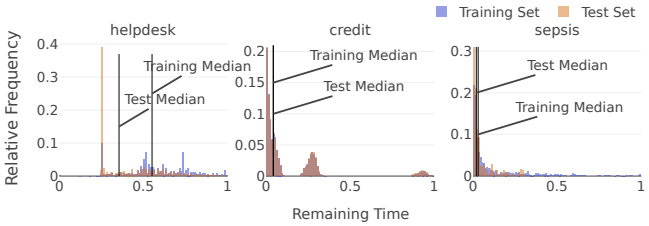


**FIGURE 6.** Examples of label drifts on real event logs.

These examples show that favourable, unfavourable, and neutral label shifts are present in real world datasets. Knowing about these shifts has the following advantages: First, the nMAE as introduced in Eq. 3 accounts for label shifts. Understanding label shifts helps understanding the nMAE. Second, knowing about label shifts can help in practical settings for model monitoring. They can be a factor to explain the performance of models. If a model performs very good, practitioners could tend to attribute the good performance to the generalization capabilities of a model, whereas a simpler and more straightforward explanation might be favourable label shifts. If a model performs badly, unfavourable label shifts can help to explain the performance drop.

*RQ3: Which additional insights can be generated on real-world event logs with the nMAE compared to the MAE?*

Table 5 shows the MAE and nMAE achieved by different methods on 16 event logs. We report the average and standard deviation over 10 runs as described in Sec. V-G. To show that the nMAE provides important information, we discuss specific datasets in more detail. The examples are chosen such that we discuss one dataset without label shift, one dataset with an unfavourable label shift, and one dataset with a favourable label shift.

The credit dataset has shown no label shift. All methods besides PGT-Net show an nMAE considerably smaller than 1.0. The event log provides no other data than the control-flow and timestamp-based features. These are less features than in other datasets. Based on the results of XGBoost

**TABLE 5.** MAE (in days, first number) and nMAE (second number) on the test set. Average of 10 runs.

| model | XGBoost squared loss | XGBoost absolute loss | DA-LSTM | PGT-Net |
|---|---|---|---|---|
| bpic2012a | 7.971 ± 0.004<br>1.076 ± 0.001 | 6.348 ± 0.005<br>0.857 ± 0.001 | **6.338** ± 0.090<br>**0.856** ± 0.012 | 14.710 ± 6.912<br>1.986 ± 0.933 |
| bpic2012o | 6.782 ± 0.009<br>1.055 ± 0.001 | 5.482 ± 0.004<br>0.853 ± 0.001 | 5.602 ± 0.141<br>0.872 ± 0.022 | **5.452** ± 0.069<br>**0.848** ± 0.011 |
| bpic2012w | 6.825 ± 0.029<br>1.058 ± 0.004 | **5.991** ± 0.015<br>**0.929** ± 0.002 | 6.388 ± 0.299<br>0.991± 0.046 | 6.305 ± 0.293<br>0.978 ± 0.045 |
| bpic2015_1 | 37.757 ± 0.827<br>0.929 ± 0.020 | 32.669 ± 1.602<br>0.804 ± 0.039 | **30.610** ± 1.631<br>**0.753** ± 0.040 | 32.408 ± 1.303<br>0.797 ± 0.032 |
| bpic2015_2 | 93.373 ± 4.573<br>1.668 ± 0.082 | 65.900 ± 3.906<br>1.177 ± 0.070 | 55.453 ± 2.272<br>0.991± 0.041 | **54.812** ± 2.203<br>**0.979** ± 0.039 |
| bpic2015_3 | 30.620 ± 3.524<br>2.029 ± 0.234 | 13.985 ± 1.256<br>0.927 ± 0.083 | 14.410 ± 1.087<br>0.955 ± 0.072 | **12.255** ± 1.025<br>**0.812** ± 0.068 |
| bpic2015_4 | 47.898 ± 0.575<br>1.071 ± 0.013 | **43.582** ± 5.605<br>**0.975** ± 0.125 | 47.059 ± 1.078<br>1.052 ± 0.024 | 51.370 ± 2.732<br>1.149 ± 0.061 |
| bpic2015_5 | 61.074 ± 14.024<br>1.409 ± 0.324 | 42.635 ± 0.547<br>0.984 ± 0.013 | 43.283 ± 2.423<br>0.999 ± 0.056 | **42.375** ± 1.453<br>**0.979** ± 0.034 |
| bpic2020_dd | 3.762 ± 0.004<br>0.850 ± 0.001 | **3.453** ± 0.006<br>**0.780** ± 0.001 | 3.607 ± 0.220<br>0.815 ± 0.050 | 3.464 ± 0.030<br>0.782 ± 0.007 |
| bpic2020_id | 13.486 ± 0.013<br>0.797 ± 0.001 | **10.950** ± 0.013<br>**0.647** ± 0.001 | 12.635 ± 0.728<br>0.746 ± 0.043 | 11.934 ± 0.665<br>0.7050 ± 0.039 |
| bpic2020_ptc | 8.886 ± 0.023<br>0.987 ± 0.003 | **7.020** ± 0.099<br>**0.780** ± 0.011 | 7.565 ± 0.414<br>0.840 ± 0.046 | 7.407 ± 0.259<br>0.823 ± 0.029 |
| bpic2020_rp | 4.595 ± 0.012<br>0.830 ± 0.002 | 4.341 ± 0.009<br>0.785 ± 0.002 | 4.389 ± 0.298<br>0.793 ± 0.054 | **4.323** ± 0.095<br>**0.781** ± 0.017 |
| credit | 0.069 ± 0.001<br>0.184 ± 0.002 | 0.063 ± 0.001<br>0.167 ± 0.002 | **0.062** ± 0.003<br>**0.164** ± 0.009 | 0.623 ± 0.225<br>1.664 ± 0.602 |
| helpdesk | 6.013 ± 0.022<br>0.612 ± 0.002 | 5.136 ± 0.067<br>0.522 ± 0.007 | **5.041** ± 0.347<br>**0.513** ± 0.035 | 5.271 ± 0.451<br>0.536 ± 0.046 |
| hospital | 43.769 ± 0.130<br>1.008 ± 0.003 | 35.930 ± 0.057<br>0.827 ± 0.001 | 36.450 ± 0.214<br>0.839 ± 0.005 | **35.871** ± 0.664<br>**0.826** ± 0.015 |
| sepsis | 37.545 ± 0.224<br>2.217 ± 0.013 | 18.135 ± 0.076<br>1.071 ± 0.005 | **17.349** ± 0.197<br>**1.024** ± 0.012 | 18.569 ± 0.325<br>1.096 ± 0.019 |

and DA-LSTM we conclude that the information present in the training features is still highly informative in order to predict the remaining time on the test set. PGT-Net has a very high MAE compared to the other methods. Since XGBoost and DA-LSTM would be also affected by concept drifts or uninformative features, we assume the low performance of PGT-Net is due to overfitting. Overfitting describes the problem that a model is fitted exactly on the training data and therefore loses its generalization capabilities.

The sepsis dataset shows a favourable concept drift. The nMAE achieved is larger than 1.0 for all models. This indicates that the input features for the sepsis dataset are not informative, regardless whether they are represented in a tabular format, an event sequence, or a graph-based format. This is further supported by comparing the performance of the models with the baseline model. Our baseline model does not utilize any input features for making a prediction, still it achieves an MAE of 17.663 days on the test set. The best performance of the models, achieved by DA-LSTM, is 17.349 days. This is very close to the baseline. The other models are worse than the baseline. The nMAE directly indicates this problem without having to compare the performance of a model additionally with other models or the baseline.

For the helpdesk dataset we observe an unfavourable label shift. However, the nMAE for all models shows values substantially smaller than 1.0. This shows that the training features are actually informative and using them allows to mitigate the negative effects of the label shift, which is represented by the nMAE smaller than 1.0.

These examples show that the nMAE has desirable characteristics. It allows an assessment whether the features present in a dataset are useful to predict the remaining time and whether machine learning models train efficiently. The MAE is still an important metric to look at, especially in practical settings, since it provides knowledge whether business goals can be met. However, for model assessment the nMAE is a useful tool. For the sepsis dataset, we have shown that the input features used are not informative to predict the remaining time, whereas for credit and helpdesk we saw that informed predictions are made. We believe that this is a promising starting point for future research to assess models and to develop and investigate strategies for improving models more target oriented.

*RQ4: How do commonly applied machine learning methods perform based on the insights generated by the nMAE?*

Table 5 shows that the nMAE is on some datasets still close to a value of 1.0. We observe this for the datasets sepsis, bpic2015_2, bpic2015_4, and bpic2015_5. This is unexpected since the introduction and comparison of various methods in the past suggested that the data was used more and more efficiently. One possible explanation for this discrepancy could be concept drifts between the training and test set, such that the process behavior from the training set is outdated and the learned mappings are not useful for prediciting the test set. Other explanations are that the event logs are either not used efficiently, despite different approaches like tabular encoding, sequential encoding, and graph representations, or the data, including the control-flow, might not be informative at all. The low performance could also arise due to a combination of all these factors. Therefore, an explanation is not trivial and we restrict our conclusion to the finding that commonly applied state-of-the-art methods perform poorly on a range of datasets. We generated this insight using the nMAE. Finding the exact reasons for poor performance goes beyond the scope of the nMAE and requires dedicated methods which we motivate as open research question.

For all other datasets the methods perform reasonably well. However, a value close to zero is only achieved for the credit dataset. The nMAE on the helpdesk dataset is already 0.522 for the best performing model, and even higher on all other datasets. Therefore, we assume that the data commonly used for remaining time prediction could be improved.

To answer RQ4, we state that the performance of machine learning models based on the nMAE should be judged less optimistic than indicated in previous work. Despite continuous improvements, there are challenges in handling specific datasets. Identifying useful features and handling the causes of bad performance is an open research question.

*RQ5: How do commonly applied machine learning methods perform relative to each other?*

Table 5 shows that XGBoost using a squared loss is outperformed by all other approaches. XGBoost with an absolute loss criterion, DA-LSTM, and PGT-Net show similar results without any method being clearly the best one. These findings contradict the conclusions of prior studies in two regards. In order to draw reliable conclusions from the results of our study, we therefore first analyze and explain the contradictions to prior studies. Then we revert back to discussing the research question.

### 1) Explaining Differences to Prior Studies

As a first contradiction, Verenich et al. [4] found that DA-LSTM significantly outperforms classical machine learning methods, represented by XGBoost. This discrepancy can be easily explained since Verenich et al. [4] used the squared error criterion for their experiments, but compared the models based on the MAE. Changing the loss function for XGBoost to an absolute loss leads to XGBoost outperforming DA-LSTM on 10 out of 16 datasets in our study.

The second contradiction is the performance of PGT-Net. Our study couldn't replicate the results of the study by Elyasi et al. [6] where PGT-Net achieves remarkable low errors on a wide range of datasets, including the bpic 2015 datasets. For example, we retrieve an MAE for bpic2015_1 of $32.408$ days and an MAE of $54.812$ days for bpic2015_2. Elyasi et al. [6] retrieve with PGT-Net MAE values of $1.76$ and $3.02$ days, respectively. These differences can't be explained based on concepts discussed so far. Rather, we assume that the discrepancies arise due to a different setup between our case study and the study by Elyasi et al. [6].

The study by Elyasi et al. [6] mainly differs in three points: First, they do not preprocess the event logs such that they contain only completed cases. As a second point, Elyasi et al. [6] consider also the column *lifecycle* as previously discussed. As a third point, they perform a 5-fold cross validation procedure and report the average MAE across the five folds.

In order to investigate these points, we performed follow-up experiments in order to identify the reasons behind the discrepancies in the studies. For these experiments we used the original implementation by Elyasi et al.[4]. In the first experiment, we used two of our preprocessed event logs, namely bpic2015_1 and bpic2015_2, and trained them with the implementation by Elyasi et al. [6]. As a second experiment, we trained and evaluated with the original logs used by Elyasi et al. [6], but we changed the original implementation such that the *lifecycle* column was not used anymore. Both these experiments achieved similar low error rates that were reported in the study by Elyasi et al. [6]. Based on this finding, we investigated the implementation by Elyasi et al. [6] and found that during their cross-validation procedure prefixes from the same case are present in the training, validation and test set. We regarded this as a potential source of data leakage. Therefore, we changed the original code of PGT-Net such that prefixes of one case are either in the training, validation, or the test set during the cross-validation procedure. For training, we took the unpreprocesed bpic2015_1 and bpic2015_2 event logs originally used in their study and there were no other changes applied to the code than the sampling for the folds. In this experiment, the MAE across the five folds for bpic2015_1 rose to $32.8542$ days and to $59.7325$ days for bpic2015_2. We can therefore attribute the results obtained by Elyasi et al. [6] to a data leakage issue in which PGT-Net overfitted to prefixes of one case during training and inferred the right prediction for other prefixes of the same case in the test set of a fold. This explains the good performance on the bpic2015 datasets. There are many unique traces in these datasets which makes it easy to recognize the same cases between a training and a test set.

We also observe that PGT-Net seems to overfit easier than XGBoost and DA-LSTM, for example at the credit and bpic2012a datasets. This overfitting property led to believe that PGT-Net generalizes well on event logs with many unique traces, while in reality the good performance during the study of Elyasi et al. [6] can be attributed to a data leakage issue during model validation combined with the the strong overfitting capabilities of PGT-Net.

Based on these results we conclude that we removed drawbacks present in prior studies and we revert back to answering research question 4.

### 2) Discussing Research Question 5

To compare the models we performed a Nemenyi test. The corresponding critical differences plot is shown in Fig. 7. XGBoost with an absolute error criterion, DA-LSTM, and PGT-Net do not perform significantly different, but they all outperform the XGBoost model with a squared loss significantly. XGBoost with an absolute error criterion has the overall best performance on $5$ datasets, PGT-Net is the best model on $6$ datasets, and DA-LSTM performs best on $5$ datasets. In order to investigate if the difference in performance can be attributed to specific dataset characteristics, we calculated a range of event log metrics based on an implementation by Zandkarimi et al. [45]. There was no metric that clearly explains the difference in performance based on dataset characteristics.

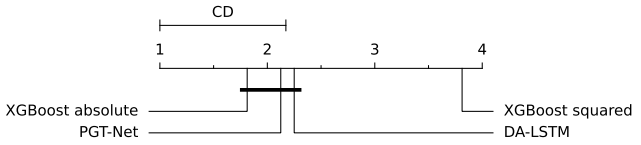


**FIGURE 7.** Critical differences plot for machine learning methods.

To further investigate the results, we compared the models based on the standard deviation of the nMAE over the 10 training runs with different random seeds. Fig. 8 shows a

[4]https://github.com/keyvan-amiri/PGT-Net/tree/main

graphical representation for the models XGBoost (absolute error), DA-LSTM, and PGT-Net with boxplots depicting the performance on the test set over the 10 runs. We first checked whether the reported mean value reported in Table 5 could be distorted such that a specific model produces actually very good models, but the good performance is averaged out by worse variants. For this analysis, we check how often each of the approaches produced the overall lowest nMAE for a dataset over all 10 runs, i. e. we compare the minimum value of the boxplots in Fig. 8 with each other. We found that PGT-Net has the overall lowest nMAE on 7 datasets, DA-LSTM on 6 datasets, and XGBoost is best on 3 event logs. This shows that the neural networks have a slight advantage over XGBoost to reach an overall minimum nMAE.

However, PGT-Net and DA-LSTM have a larger variation in the nMAE than XGBoost on most datasets. This mitigates the good performance of the neural networks. In other words, the training procedure of the neural networks does not guarantee to reach the best possible performance. We therefore see advantages for XGBoost since it shows a more stable training behavior when training with different random seeds. Neural networks, on the other hand, have slight advantages in reaching the overall best performance.

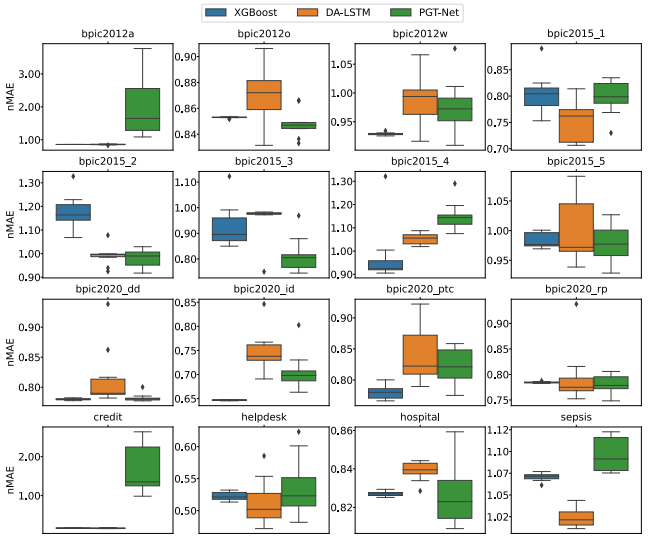


**FIGURE 8.** Variation of nMAE on test set across 10 runs.

Based on the results we conclude that XGBoost using an absolute error criterion, PGT-Net, and DA-LSTM perform equally well. We could not confirm findings of prior studies that one method is clearly the best one. Each of the methods utilizes the data in the event logs equally well, despite the different data representations and algorithmic learning approaches.

## VII. SUMMARY AND OPEN RESEARCH QUESTIONS

In this section we shortly summarize the results obtained from answering the research questions in Sec. VI and discuss research questions that directly result from our findings.

Discussing *RQ1* we have shown that previously used baseline methods are not optimal with regard to the MAE. Comparisons of methods for remaining time prediction with these baselines might provide an overly optimistic picture about the capability of these methods. We have introduced a simplistic baseline which is optimal with regard to the MAE and showed that it is more accurate than other baseline methods. Using our baseline method for comparisons with machine learning methods allows a more realistic judgment about the capability of these methods.

With research questions *RQ2* and *RQ3* we have shown that favourable and unfavourable label shifts are present in real world datasets and that the newly introduced nMAE provides useful insights into the performance of methods by considering label shifts and hinting at the effectiveness of input feature utilization of a method. This can help to motivate future research questions and to communicate scientific insights.

Building on these insights we have shown by discussing *RQ4* that state-of-the-art methods for remaining time predictions of business processes lack in predictive power for a range of commonly used datasets, namely sepsis, bpic2015_2, bpic2015_4, and bpic2015_5. For other datasets the input features are used efficiently such that an nMAE considerably smaller than 1.0 is achieved.

Discussing *RQ6* we have shown that the three considered state-of-the-art methods, namely XGBoost, DA-LSTM and PGT-Net, perform similarly. There is not one single best method which significantly outperforms others. Another major finding is the high variance of the nMAE on the test set for neural network-based methods in comparison with XGBoost. The influence of random factors for model training has not been discussed so far in the literature about remaining time prediction for business processes, according to our best knowledge. However, it raises relevant questions for practitioners. Training a model once might not necessarily lead to the best possible performance due to random factors during training. As a consequence, training a model several times might be considered. This, however, induces additional computational overhead. Methods which are robust to initial starting conditions and other random influences might therefore be preferred. How this can be achieved is an open research question.

Based on these insights we have identified several starting points for future research opportunities in order to generate scientific insights that benefit the practical application of machine learning methods for remaining time prediction of business process. A summary of these points is given below:

- There is no consensus for specific datasets how they can be filtered for complete cases. This requires knowledge about the definite start and end of cases. This is a threat to the validity of all prior studies for remaining time prediction. We have made our definition of complete cases publicly available, but future work might improve and complete this collection.

- The range of values for activities and other categorical features might differ between the training and test sets. Especially in practical settings this has to be tackled since practitioners face the problem that new categorical values might occur in a productive environment. To our best knowledge, this has not been explicitly handled in the literature so far for the prediction of the remaining time. Previous work [3], [6] determines the range of values for encoding based on the whole dataset and then conducts a split into training and validation sets. We regard this as a form of data leakage. We have found one study that investigates this problem for next activity prediction [38]. In our case study we handled this problem by simply ignoring events with previously unseen activities and using void encoding for other categorical features. Future work should investigate whether there are more suitable ways to handle this.
- We have shown that optimizing XGBoost with an absolute error criterion leads to improved results. Future work might re-evaluate other classical machine learning methods which were fitted using the squared error criterion but not the absolute error criterion.
- Neural network-based methods show a high variance in MAE and nMAE for a fixed hyperparameter setting. Future work might investigate mitigation strategies that lead to lower variance, for example by investigating different kernel initialization strategies or finding architectures that lead to more stable results.
- For some datasets the nMAE is close to 1.0 or larger, which indicates strong concept drifts and/or uninformative input features. We were able to identify these issues based on our introduced concepts, however, differentiating between the possible reasons for bad model performance is not possible. Future work might follow this direction in order to enable a more differentiated analysis of bad model performance.
- PGT-Net is a relatively new architecture which shows promising results on a range of datasets, even though we could not confirm the exceptionally good results from a prior study. For datasets with a small number of control-flow variants we found strong signs of overfitting. Future work should investigate the search for optimal architectures for PGT-Net based on different dataset characteristics.

## VIII. LIMITATIONS

Regarding our experiments, we have identified a few threats to validity. For our evaluation, we first retrieved the best hyperparameter setting by comparing different configurations based on the achieved MAE on the validation set. Next, we trained 10 models with the best configuration to account for random factors. A more thorough approach would be to evaluate each possible configuration several times already to account for random factors also during hyperparameter tuning. However, this would result in a high computational effort.

Training 10 times gives already first insights on the variation of different results, but for statistical tests it is still a low number of observations. We therefore refrained from performing statistical tests on the difference in variations and limited the discussions to general observations.

Another simplification in our study was the removal of events in the validation and test sets for which new activities occur which are not part of the training set. This was required to enable a fair comparison of the different methods because there is no straightforward solution for PGT-Net on how to handle new activities without having a high risk of skewing results, except than excluding events with previously unseen activities.

With the nMAE we have used a specific metric to judge the performance of models. There is a large variety of metrics in the literature and the choice of the right metric is not straightforward. Typically, such metrics have several advantages and disadvantages [14]. While we have discussed the desired properties of the nMAE thoroughly and showed that valuable conclusions can be derived specifically for remaining time prediction of business processes, future work might discover weaknesses which need to be accounted for.

## IX. CONCLUSION

In this work we have assessed the current state of remaining time prediction for business processes. We started by evaluating what constitutes a good baseline model from a theoretical perspective and proposed to choose the label median of the test set as a constant predictor. As a next step, we introduced the notion of different types of label shifts. Then, we proposed the nMAE to overcome weaknesses of the MAE. It accounts for the aforementioned label shifts and enables considerations regarding the utilization of input features and therefore the usefulness of machine learning methods with respect to a given test set. This supports model evaluation and monitoring to ensure reliable predictive services to internal and external stakeholders. All of our introduced concepts were evaluated with theoretical considerations as well as an empirical study, considering four baseline methods and three state-of-the-art machine learning models.

Furthermore, we have tackled weaknesses of prior studies. Specifically, we have trained XGBoost with an absolute loss criterion and removed data leakage issues which were present in a previous study for PGT-Net. The results showed that there is no significant difference in the performance between these models. Additionally, we have demonstrated that these state-of-the-art methods do not perform well on a range of datasets.

Our implementation for the case study as well as the definition of complete cases are available in a public repository for reproducibility and future research. As a last contribution, we have motivated several weaknesses and open questions in current research for remaining time prediction for business processes.

We have, therefore, reassessed the current state of remaining time prediction research, motivated open challenges and

weaknesses, and provided initial concepts and solutions to tackle these and to motivate further research in the future in that direction.

## ACKNOWLEDGMENT

The authors gratefully acknowledge the scientific support and HPC resources provided by the Erlangen National High Performance Computing Center (NHR@FAU) of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU). The hardware is funded by the German Research Foundation (DFG).

## REFERENCES

[1] C. Di Francescomarino and C. Ghidini, "Predictive process monitoring," in Process Mining Handbook, pp. 320–346, Springer International Publishing Cham, 2022.
[2] N. Tax, I. Verenich, M. La Rosa, and M. Dumas, "Predictive business process monitoring with lstm neural networks," in Advanced Information Systems Engineering: 29th International Conference, CAiSE 2017, Essen, Germany, June 12-16, 2017, Proceedings 29, pp. 477–492, Springer, 2017.
[3] N. Navarin, B. Vincenzi, M. Polato, and A. Sperduti, "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.
[4] I. Verenich, M. Dumas, M. L. Rosa, F. M. Maggi, and I. Teinemaa, "Survey and cross-benchmark comparison of remaining time prediction methods in business process monitoring," ACM Transactions on Intelligent Systems and Technology (TIST), vol. 10, no. 4, pp. 1–34, 2019.
[5] E. Rama-Maneiro, J. C. Vidal, and M. Lama, "Deep learning for predictive business process monitoring: Review and benchmark," IEEE Transactions on Services Computing, vol. 16, no. 1, pp. 739–756, 2021.
[6] K. Amiri Elyasi, H. van der Aa, and H. Stuckenschmidt, "Pgtnet: A process graph transformer network for remaining time prediction of business process instances," in International Conference on Advanced Information Systems Engineering, pp. 124–140, Springer, 2024.
[7] L. T. Duong, L. Travé-Massuyès, A. Subias, and C. Merle, "Remaining cycle time prediction with graph neural networks for predictive process monitoring," in Proceedings of the 2023 8th International Conference on Machine Learning Technologies, pp. 95–101, 2023.
[8] I. Verenich, M. Dumas, M. La Rosa, and H. Nguyen, "Predicting process performance: A white-box approach based on process models," Journal of Software: Evolution and Process, vol. 31, no. 6, p. e2170, 2019.
[9] R. S. Oyamada, G. M. Tavares, S. B. Junior, and P. Ceravolo, "Enhancing predictive process monitoring with time-related feature engineering," in International Conference on Advanced Information Systems Engineering, pp. 71–86, Springer, 2024.
[10] M. Pourbafrani, S. Kar, S. Kaiser, and W. M. van der Aalst, "Remaining time prediction for processes with inter-case dynamics," in International Conference on Process Mining, pp. 140–153, Springer International Publishing Cham, 2021.
[11] R. Aalikhani, M. Fathian, and M. R. Rasouli, "Comparative analysis of classification-based and regression-based predictive process monitoring models for accurate and time-efficient remaining time prediction," IEEE Access, 2024.
[12] D. Chicco, M. J. Warrens, and G. Jurman, "The coefficient of determination r-squared is more informative than smape, mae, mape, mse and rmse in regression analysis evaluation," Peerj computer science, vol. 7, p. e623, 2021.
[13] J. W. McKean and G. L. Sievers, "Coefficients of determination for least absolute deviation analysis," Statistics & Probability Letters, vol. 5, no. 1, pp. 49–54, 1987.
[14] O. Renaud and M.-P. Victoria-Feser, "A robust coefficient of determination for regression," Journal of Statistical Planning and Inference, vol. 140, no. 7, pp. 1852–1862, 2010.
[15] D. McElfresh, S. Khandagale, J. Valverde, V. Prasad C, G. Ramakrishnan, M. Goldblum, and C. White, "When do neural nets outperform boosted trees on tabular data?," Advances in Neural Information Processing Systems, vol. 36, 2024.
[16] H. Weytjens and J. De Weerdt, "Learning uncertainty with artificial neural networks for predictive process monitoring," Applied Soft Computing, vol. 125, p. 109134, 2022.
[17] D. M. V. Sato, S. C. De Freitas, J. P. Barddal, and E. E. Scalabrin, "A survey on concept drift in process mining," ACM Computing Surveys (CSUR), vol. 54, no. 9, pp. 1–38, 2021.
[18] V. Pasquadibisceglie, A. Appice, G. Castellano, and D. Malerba, "Darwin: An online deep learning approach to handle concept drifts in predictive process monitoring," Engineering Applications of Artificial Intelligence, vol. 123, p. 106461, 2023.
[19] M. Maisenbacher and M. Weidlich, "Handling concept drift in predictive process monitoring," in 2017 IEEE International Conference on Services Computing (SCC), pp. 1–8, IEEE, 2017.
[20] W. Rizzi, C. Di Francescomarino, C. Ghidini, and F. M. Maggi, "How do i update my model? on the resilience of predictive process monitoring models to change," Knowledge and Information Systems, vol. 64, no. 5, pp. 1385–1416, 2022.
[21] I. Firouzian, M. Zahedi, and H. Hassanpour, "Investigation of the effect of concept drift on data-aware remaining time prediction of business processes," International Journal of Nonlinear Analysis and Applications, vol. 10, no. 2, pp. 153–166, 2019.
[22] J. Quionero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence, Dataset Shift in Machine Learning. The MIT Press, 2009.
[23] A. Djedović, "Credit requirement event logs," 2017.
[24] W. M. Van der Aalst, M. H. Schonenberg, and M. Song, "Time prediction based on process mining," Information systems, vol. 36, no. 2, pp. 450–475, 2011.
[25] M. Ceci, P. F. Lanotte, F. Fumarola, D. P. Cavallo, and D. Malerba, "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.
[26] C. M. Bishop and N. M. Nasrabadi, Pattern recognition and machine learning, vol. 4. Springer, 2006.
[27] H. Hewamalage, K. Ackermann, and C. Bergmeir, "Forecast evaluation for data scientists: common pitfalls and best practices," Data Mining and Knowledge Discovery, vol. 37, no. 2, pp. 788–832, 2023.
[28] B. van Dongen, "Bpi challenge 2012," 2012.
[29] B. B. van Dongen, "Bpi challenge 2015," 2015.
[30] B. van Dongen, "Bpi challenge 2020," 2020.
[31] M. Polato, "Dataset belonging to the help desk log of an italian company," 2017.
[32] F. Mannhardt, "Hospital billing - event log," 2017.
[33] F. Mannhardt, "Sepsis cases - event log," 2016.
[34] A. D. Bautista, L. Wangikar, and S. M. K. Akbar, "Process mining-driven optimization of a consumer loan approvals process," BPI Challenge, 2012.
[35] C. J. Kang, C. K. Shin, E. S. Lee, J. H. Kim, and M. A. An, "Analyzing application process for a personal loan or overdraft of dutch financial institute with process mining techniques," in Proceedings of the 8th International Workshop on Business Process Intelligence, 2012.
[36] I. Teinemaa, A. Leontjeva, and K.-O. Masing, "Bpic 2015: Diagnostics of building permit application process in dutch municipalities," BPI Challenge Report, vol. 72, 2015.
[37] H. Weytjens and J. De Weerdt, "Creating unbiased public benchmark datasets with data leakage prevention for predictive process monitoring," in International Conference on Business Process Management, pp. 18–29, Springer, 2021.
[38] A. S. Mangat and S. Rinderle-Ma, "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.
[39] Z. A. Bukhsh, A. Saeed, and R. M. Dijkman, "Processtransformer: Predictive business process monitoring with transformer network," arXiv preprint arXiv:2104.00721, 2021.
[40] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16, (New York, NY, USA), pp. 785–794, ACM, 2016.
[41] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in Advances in Neural Information Processing Systems 32, pp. 8024–8035, Curran Associates, Inc., 2019.
[42] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp,

G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

[43] L. Rampášek, M. Galkin, V. P. Dwivedi, A. T. Luu, G. Wolf, and D. Beaini, "Recipe for a general, powerful, scalable graph transformer," Advances in Neural Information Processing Systems, vol. 35, pp. 14501–14515, 2022.

[44] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," The Journal of Machine learning research, vol. 7, pp. 1–30, 2006.

[45] F. Zandkarimi, P. Decker, and J.-R. Rehse, "Fig4pm: a library for calculating event log measures," ICPM Doctoral Consortium and Demo Track, pp. 27–28, 2021.

JOHANNES ROIDER received a B.Sc. degree in european economic studies from Otto-Friedrich-University Bamberg, Germany, in 2015. As part of his studies he spent one year as an exchange student at the University of Padova, Italy, focusing on economics, followed by a one year internship at Jebsen Group in Hong Kong. He then received a second B.Sc. in business information systems from the University of Regensburg, Germany, in 2018, and a M.Sc. in computer science from Friedrich-Alexander-University Erlangen-Nürnberg, Germany, in 2021.

He is currently pursuing a Ph.D. degree with the Machine Learning and Data Analytics Lab (MaD), Friedrich-Alexander-University Erlangen-Nürnberg (FAU), Erlangen, Germany. His research interests include data analysis and machine learning for time-series regression problems in business applications.

AN NGUYEN received the B.Sc. and M.Sc. degrees in electrical engineering from the Technical University of Berlin, in 2016 and 2018, respectively, and the M.S.E. degree in electrical and computer engineering from the University of Michigan, in 2017. He obtained his Ph.D. in Computer Science/Machine Learning from Friedrich Alexander University Erlangen Nuremberg in 2023.

He is currently a Senior Data Scientist at Siemens Healthineers, Erlangen, Germany, and a PostDoc at Friedrich-Alexander-University Erlangen-Nürnberg (FAU), Germany. His research interests include data science, machine learning, and process mining to analyze time series in industrial and healthcare applications.

DARIO ZANCA received the B.Sc. and M.Sc. degrees in mathematics from the University of Palermo, Italy, in 2013 and 2015, respectively, and the Ph.D. degree in smart computing from the University of Florence, Italy, in 2019. During his Ph.D. studies, he was partly working as a Visiting Researcher at the California Institute of Technology, California City. In addition, he served as a Postdoctoral Fellow at the Department of Medicine, Surgery, and Neuroscience, University of Siena, Italy.

He is currently a Postdoctoral Fellow at the Machine Learning and Data Analytics Lab (MaD), Friedrich-Alexander-University Erlangen-Nürnberg (FAU), Germany. His research interests include the areas of deep learning, computer vision, and neuroscience, with an emphasis on building artificial models that approach human capabilities of robust perception and reasoning.

BJOERN M. ESKOFIER (Senior Member, IEEE) heads the Machine Learning and Data Analytics (MaD) Lab with the Friedrich-Alexander-University Erlangen-Nuernberg (FAU), Erlangen, Germany. He is also the founding Spokesperson of FAU's Department Artificial Intelligence in Biomedical Engineering, the German Ministry of Economic Affairs and Climate Action GAIA-X usecase project TEAM-X and Co-spokesperson of the German Research Foundation collaborative research center EmpkinS(www.empkins.de). Since April 2023, he has been an Associate Principal investigator and leader of the research group Translational Digital Healthwith the Helmholtz Zentrum Munich. From April 2023 to August 2023, he was a Visiting Professor with Prof. Scott Delp's NMBL lab that is part of Stanford University's Schools of Engineering and Medicine. Dr. Eskofier studied Electrical Engineering with the FAU and graduated in 2006. He received the Ph.D. degree in biomechanics under the supervision of Prof. Dr. Benno Nigg from the University of Calgary Calgary, AB, Canada.

He authored more than 400 peer reviewed articles, holds 5 patents, started three spinoff startup companies, and is in a supporting role for further startups. He was the recipient of the several medical-technical research awards, including the Curious Mindsaward 2021 in Life Sciences by Manager Magazin and Merck. In 2016, he was a Visiting Professor with Prof. Paolo Bonato's Motion Analysis Lab, Harvard Medical School (February-March), and in 2018, he was a visiting Professor with Prof. Alex Sandy Pentland's Human Dynamics group, MIT Media Lab (March-August).

He was the Area Editor for the IEEE Open Access Journal of Engineering in Medicine and Biology and Associate Editor for the IEEE Journal of Biomedical and Health Informatics. He is also active in the organization of several IEEE and ACM meetings (e.g., BSN, BHI, EMBC, IJCAI, ISWC, UbiComp), most recently as General Chair of BHI 2023.