Understanding the impact of design choices on the performance of predictive process monitoring*

Sungkyu Kim[1][0000−0002−9311−6373], Marco Comuzzi[1][0000−0002−6944−4705], and Chiara Di Francescomarino[2][0000−0002−0264−9394]

[1] Ulsan National Institute of Science and Technology, Ulsan, Korea
{kimkangf3,mcomuzzi}@unist.ac.kr
[2] University of Trento, Trento, Italy
c.difrancescomarino@unitn.it

Abstract. Predictive process monitoring (PPM) aims at creating models that predict aspects of interest of process execution using historical data available in event logs, mostly using machine learning (ML) techniques. When developing a PPM model, one has several design choices, encompassing both ML-related concerns, such as which classification or regression model to choose, and PPM-specific concerns, such as how to encode the trace prefixes or whether to drop infrequent activities when training a model. While the literature has seen a few attempts to study how these choices impact the performance of a PPM model, no systematic studies on this matter exist. This paper moves towards closing this gap. We propose a framework to interpret the impact of design choices on the performance of a PPM model. The proposed framework uses as building blocks a search space exploration algorithm, which is able to generate different model configurations, and explainable AI techniques, e.g., SHAP, to analyze the impact of design choices on the model performance based on the generated configurations. We show an instantiation of the framework in the use case of outcome-oriented PPM, discussing also the experimental results obtained using publicly available event logs.

Keywords: Business Process · Prediction · Guidelines.

1 Introduction

Predictive process monitoring (PPM) aims at creating predictive models of business process execution using the historic data available in event logs, often exploiting machine learning (ML) techniques [8]. PPM historically has considered three aspects to be predicted: the activities that will be executed next in a running case, the timestamps of such activities (including the remaining case duration), and the outcome of running cases, as usually captured by a categorical label.

The development of PPM models requires setting a value for various hyperparameters [5, 1]. These include both typical ML hyperparameters of the classification or regression techniques chosen to develop a model, as well as others

* Sponsored by the NRF Korea, Grant Number 2022R1F1A1072843.

that are specific to the PPM task at hand. For instance, one needs to choose how to encode the trace event data to obtain a feature vector and whether to divide the trace prefixes into buckets when training a model, determining if needed the number and types of buckets.

Getting insights into the impact of these design choices on the performance of a PPM model can be crucial for model developers. While a few papers have tried to create benchmarks for different PPM tasks (e.g. [13, 16, 11]) and few works have focused on searching for the best hyperparameters for a PPM task [5, 1], there is no empirical work in the literature specifically aiming at understanding the impact of the hyperparameter values on the performance of a PPM technique. Moreover, since some of these hyperparameters are PPM-specific, we cannot simply adapt insights obtained for other ML scenarios. In this context, this work focuses on the outcome-oriented PPM task, aiming to answer the following research question: "How does the value of PPM-specific hyperparameters impact the performance of outcome-oriented PPM models?"

Answering this question can be crucial for model developers. By knowing in advance which parameter values are more likely to yield a well-performing model, they may save huge amounts of time and computational resources when developing a PPM model. Knowledge about the optimal hyperparameter values can also inform the development of AutoML solutions for PPM, reducing the effort of exploring the space determined by the hyperparameter value combinations.

From a methodological standpoint, this question could be tackled by creating an empirical benchmark testing several configurations of hyperparameter values on different event log datasets. As mentioned earlier, this has been tried in [13, 16, 11], even though not systematically, and it requires a massive effort in terms of computational cost and analysis. In this work, we take a novel and more lightweight approach, exploiting the capabilities of explainable AI techniques.

More in detail, we propose a general framework for interpreting the impact of design choices on the PPM model performance that includes three phases. In the first phase, a search space exploration algorithm is used to generate an extensive number of hyperparameter value configurations to configure an outcome-oriented PPM model. In the second phase, these configurations are used as feature vectors associated with a numerical value of the performance of the corresponding PPM model, e.g., model accuracy or AUC, to fit a regression model. Finally, in the third phase, XAI techniques are used to "interpret" the contribution made by each feature, i.e., hyperparameter value, on the model performance, i.e., the predicted numerical label.

We discuss an instantiation of the proposed framework based on the genetic algorithm-based search space exploration techniques proposed in [5], as well as SHAP and Explainable Boosting Machines as XAI techniques, discussing the results obtained on publicly available event logs.

The paper is organised as follows. After a discussion of the related work in Section 2, Section 3 introduces the general framework. The specific instantiation of the framework that we implemented is discussed in Section 4, while the experimental results are reported and discussed in Section 5. Conclusions are finally drawn in Section 6.

2 Related Work

We can roughly classify the literature related to this paper into two groups: (i) the works related to outcome-oriented Predictive Process Monitoring; (ii) the state-of-the-art concerning AutoML.

Outcome-oriented Predictive Process Monitoring focuses on predicting the outcome (e.g., the satisfaction of a business objective) of a process [13]. In [8, 2] the sequence of activities already carried out and the data payload of the last activity are leveraged to make predictions on the fulfilment (or the violation) of a boolean predicate in a running case. In [6], traces are considered as complex symbolic sequences, i.e., sequences of activities each carrying its data payload, and different approaches for feature encoding are considered. In [15], the approach in [6] has been extended by clustering the historical traces before classification. In [13], a comparison of the existing outcome-based predictive monitoring approaches is presented.

AutoML automates the process of developing the best model, e.g., the most accurate one, to address a given machine learning task, or speeding up the model development phase. Different AutoML frameworks, such as Auto-sklearn, Tree-Based Pipeline Optimization Tool (TPOT), or H2O, provide different automated solutions for each different step of the typical machine learning pipeline [17, 4], such as data preparation or hyperparameter optimisation. Several approaches in machine learning have been proposed for the selection of a learning algorithm [10], for the tuning of hyperparameters [3], and for the combined optimization of both the algorithm and the hyperparameters [14]. AutoML has been generally neglected by the PPM literature, with the exception of [5, 1]. In [1], an approach based on a genetic algorithm has been proposed for the identification of the best configuration, in terms of predictive models, encodings and bucketing methods, for PPM tasks. In [5], besides encoding and bucketing methods, additional parameters, such as the dropping of infrequent activities, as well as a broader set of models have been used.

3 A Framework for analyzing design choices

Figure 1 depicts the proposed framework for the analysis of the design choices in outcome-based PPM. We assume that the design choices are captured by the values of PPM model hyperparameters. As mentioned in the Introduction, these may range from the classification and trace encoding technique adopted, up to the number of buckets in which trace prefixes are divided. The *search space* is constituted by the combination of all the values of such hyperparameters. A *configuration* is a point in the search space, in which one value is assigned to each hyperparameter. For example, a configuration may be determined by choosing to

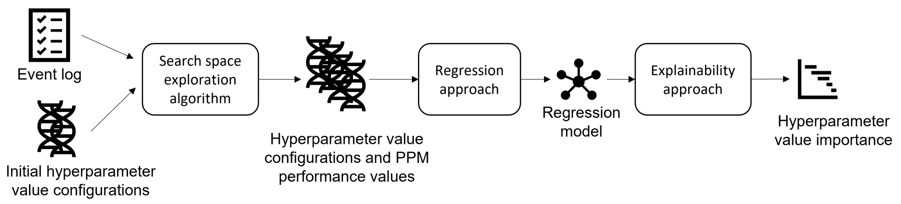
Event log
Initial hyperparameter value configurations
Search space exploration algorithm
Hyperparameter value configurations and PPM performance values
Regression approach
Regression model
Explainability approach
Hyperparameter value importance

Fig. 1. Framework overview

use a decision tree, encoding traces using index-based encoding, and considering only one bucket containing all the encoded trace prefixes when training/testing.

The input of the framework is an event log and, if necessary, an initial hyperparameter configuration. In the first step of the framework, a search-space exploration algorithm is used to generate a set of hyperparameter configurations and to compute the corresponding PPM performance values, that is, the PPM performance values obtained by using that configuration. To this aim, we start from an event log and an initial set of hyperparameter configurations and we explore the search space of the PPM model hyperparameter configuration values by generating new configurations. For each configuration, the performance of the PPM model obtained by leveraging it on the event log is computed. Note that different types of algorithms may be chosen, e.g., grid search, optimization-based, or evolutionary-based.

Once the hyperparameter configurations and the performance metrics of the corresponding outcome-oriented PPM model have been generated, in the second step of the framework, they are transformed into numeric feature vectors and used to train a regression model that aims at predicting, given a configuration of hyperparameter values, the performance of the corresponding PPM model, that is, the model built using that particular configuration.

Finally, in the third and last step, a post-hoc explainer for XAI is applied on top of the regression model in order to understand the impact of each feature (i.e., hyperparameter configuration value) on the performance of the model to give users explainability in setting hyperparameter configuration of PPM.

4 Instantiating the framework in outcome-oriented PPM

We instantiate the framework described in Section 3 by leveraging for the exploration of the search space an existing GA-based exploration approach (described in [5]). The algorithm aims at optimizing the performance of the outcome-oriented PPM models built leveraging the hyperparameter value configurations. More specifically, for each event log, each individual of the population of the genetic algorithm corresponds to a hyperparameter value configuration as well as to the outcome-oriented PPM model obtained by using that specific configuration. The fitness function, which is defined as:

$$f(i) = [sc(i) + re(i)]/2 \tag{1}$$

aims at optimizing the performance of such a PPM model by maximizing a performance score $sc(i)$, computed as the average AUC and accuracy of the PPM model i, and minimizing the error rate. The latter is captured by the term $re(i)$, which is defined as $1 - failurerate(i)$, where the failure rate is defined as the percentage of cases in which the outcome predicted by the model i has a class probability lower than 0.7.

The GA-based exploration approach introduced in [5] considers the following design space:

`Model`: Even though any classification model can be used, the literature highlights that tree-based classifiers show better performance in outcome-based PPM [13]. We consider four tree-based models, including both individual and ensemble classifiers: Decision Tree (DT), Random Forest (RF), XGBoost (XGB), and LightGBM (LGBM).

`Drop_act`: this configuration parameter captures the process of removing low-frequency activities from an event log, which may reduce the computational cost and improve the model performance [12]. We consider a discrete gap-based scale for this parameter, i.e., dropping the 2, 4, 6, or 8 less frequent activities in an event log.

`Bucketing`: When pre-processing an event log for outcome-based prediction, the prefixes of each trace are extracted to construct a prefix log. The prefixes then can be grouped into so-called *buckets*. A different classification model is trained for each bucket. In this paper, we consider prefix-length bucketing, in which prefixes are grouped by length, i.e., number of events. A base strategy (zero-bucketing) groups all prefixes in a single bucket, thus training a single classifier. Bucketing allows the grouping of homogeneous prefixes, which may improve the accuracy of the trained models. Given an input event log, this parameter assumes values comprised between 1 (corresponding to zero-bucketing) up to two times the mean length of the traces in an event log. The value n of this parameter signifies that n buckets are created. If for instance, $n = 3$ and the maximum length of trace in a log is six events, then three buckets are created containing the prefixes of length 1 and 2, 3 and 4, 5 and 6, respectively.

`Encoding`: The prefixes must be numerically encoded to be fed into the model. The problem of encoding prefixes is one of complex symbolic sequence encoding [6] and can be approached in multiple ways. In this paper, we consider the *aggregation* and *index-based* encodings. Aggregation is a lossy encoding, which represents entire event sequence attributes into a single entity, for example, based on frequency. Index-based is a lossless encoding that maintains the order of events in a prefix. In index-based encoding, each event in a prefix is encoded into a fixed number of numerical features.

Table 1 shows an example of a possible set of generated hyperparameter value configurations and the corresponding PPM model performance values.

As a regression model, we consider the Explainable Boosting Machine (EBM), a tree-based cyclic gradient boosting generalized additive model with automatic interaction detection [7]. Despite the simplicity of the prediction task for the regression model in this research, we selected EBM over the simple linear regres-

Model	Drop_act	Bucketing	Encoding	**Fitness function**
DT	2	1	index-based	0.93
XGB	3	3	aggregation	0.97
DT	4	6	aggregation	0.85
...				

Table 1. Example of hyperparameter values configurations and corresponding fitness function values

sion model because of its intelligibility, accuracy, and ability to detect pairwise interaction among features. A pairwise interaction refers to how two features in a statistical or machine learning model interact with each other to determine the outcome of the model.

Thus, EBM is interpretable and it offers global explanations of the model in terms of both feature contribution and interaction effects. Yet, we also consider the Shapley Additive exPlanations (SHAP) to answer our research question precisely. While EBM provides global explanations with both feature contribution and interaction terms, SHAP is considered to offer more precise feature contribution values by considering marginal contributions. To summarize, the instantiated framework for outcome-oriented PPM provides explanations in the form of feature contribution and feature interaction analysis of the regression model: SHAP is used to calculate the feature contribution, whereas the EBM model is used to analyze the interaction among features towards determining the output of the model.

5 Experimental Results and Discussion

In this section, we first introduce the event log datasets that we considered in the experiment in Section 5.1. Then, we aim to answer our research question: "How does the value of PPM-specific hyperparameters impact the performance of outcome-oriented PPM models?". This is done in two steps with XAI techniques. First, we present the results of the design feature contribution analysis using SHAP in Section 5.2; then we analyze the results obtained for the design feature contribution interaction analysis using EBM in Section 5.3.

5.1 Datasets

We consider four event logs made available by the Business Process Intelligence Challenge (BPIC) that are commonly used in the literature. We followed the same outcome labelling strategy of [13] for a total of 15 datasets. Table 2 shows the characteristics of each dataset.

The BPIC2011 log refers to a diagnosis and treatment process in the gynaecology department of a Dutch academic hospital. Since the treatments do not follow a strict process, it shows relatively high trace length compared to other datasets. The BPIC2012 log refers to a personal loan application process in a Dutch financial institute. There are three outcome labels defined for this

Datasets	# Traces	Min length	Med length	Max length	# Events	# Activities	# Variants	Class ratio
BPIC2011_1	1058	1	24	1814	57850	193	734	0.77
BPIC2011_2	1058	1	50	1814	138542	251	900	0.36
BPIC2011_3	1045	1	21	1368	69078	190	783	0.91
BPIC2011_4	1058	1	42	1432	84873	231	900	0.76
BPIC2012_1	4685	15	35	175	186693	36	3790	0.46
BPIC2012_2	4685	15	35	175	186693	36	3790	0.70
BPIC2012_3	4685	15	35	175	186693	36	3790	0.84
BPIC2015_1	696	2	42	101	28775	380	677	0.72
BPIC2015_2	753	1	55	132	41202	396	752	0.77
BPIC2015_3	1328	3	42	124	57488	380	1285	0.76
BPIC2015_4	577	1	42	82	24234	319	576	0.82
BPIC2015_5	1051	5	50	134	54562	376	1049	0.64
BPIC2017_20	4982	10	22	148	139232	25	1460	0.71
BPIC2017_30	7473	10	26	148	240537	25	3104	0.63
BPIC2017_40	9964	10	29	148	341953	25	4625	0.60

Table 2. Descriptive statistics of the datasets used in the experiments

log, i.e., whether an application is approved, cancelled, or rejected which yields three datasets in Table 2 (BPIC2012_1, BPIC2012_2, and BPIC2012_3, respectively). The BPIC2015 log refers to a building permit application process in 5 different Dutch municipalities. The process at each municipality is captured by a different log, hence 5 datasets are considered (BPIC2015_1 to BPIC2015_5). Due to its long recording period (about four years for each municipality), the process has changed over the years, resulting in a significant increase in the number of traces relative to the number of variants compared to other datasets. Finally, the BPIC2017 log is an updated version of the BPIC2012 referring to a more recent period in which a new information system has been used at the Dutch financial institute. It follows the same labelling strategy of BPIC2012. However, since it is a very large event log, we created three versions of it considering only the "accepted" label and sampling 20%, 30%, and 40% of the original datasets, respectively, by removing traces belonging to infrequent variants. The datasets and the code to reproduce the experiments discussed next are available at `https://github.com/brucks1217/Understanding-the-impact-of-design-choices`.

5.2 Analyzing the design choices using SHAP

Initially, we present the results of the mean absolute SHAP values to analyze the contribution of each configuration feature (that is, design choice) on the performance of the model. Note that the SHAP mean absolute value does not give any insights into the direction of the impact of the feature on the model performance, i.e., whether positive or negative.

Fig. 2 shows the results obtained. An initial insight that can be drawn is that the feature `bucketing` is the most impactful design choice across all datasets. By looking at the differences among the different event logs, it seems that this feature is less impacting for event logs with a high number of activities, such as the BPIC2012 and BPIC 2017 event logs. Apart from `bucketing`, the choice of the `model` (in particular choosing RF) and of the index-based `encoding` seems to

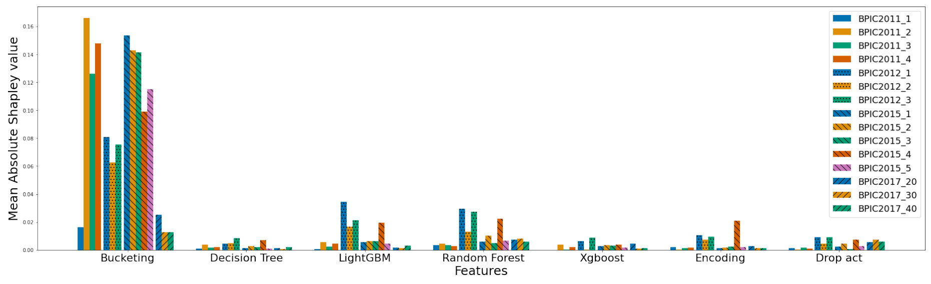

Fig. 2. Mean Absolute SHAP value of design choices

have an impact on the performance, while the impact of `drop-act` looks limited. Another interesting remark is that the SHAP values of `model` (especially LGBM and RF) for BPIC2012 are relatively higher in respect of the one of `bucketing` when compared to other datasets. This implies that, for BPIC2012, the choice of the `model` has a more significant impact on the classification performance compared to other event logs. This is not the case, for instance, for the BPIC2011_1 and BPIC2017 datasets. For these logs, the SHAP value of all other features, except for `bucketing`, appears particularly low.

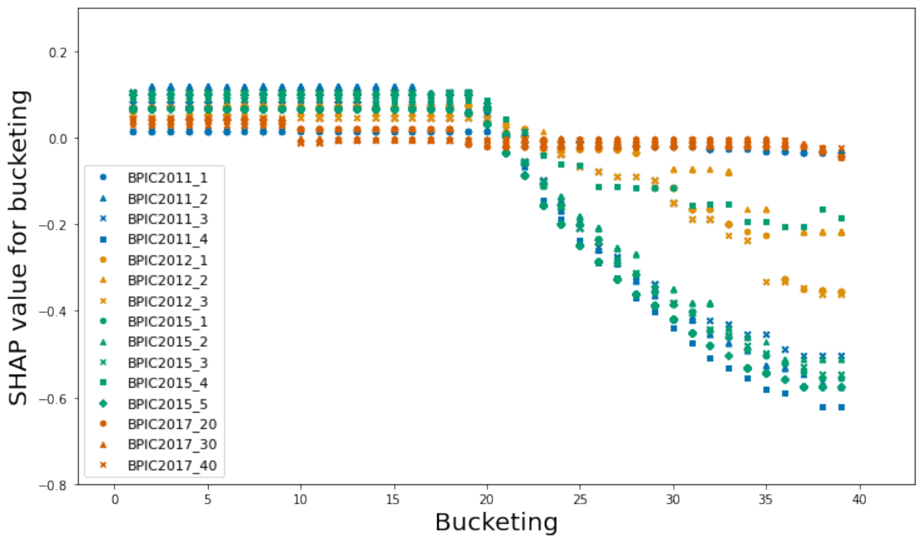

Fig. 3. SHAP values for design dimension `bucketing`

In contrast to the mean absolute SHAP value, which does not discriminate between positive and negative feature contributions to the model output, the SHAP value for each feature reveals how that feature impacts the model pre-

	BPIC2011_1	**BPIC2011_2**	**BPIC2011_3**	**BPIC2011_4**	**BPIC2012_1**
Decision Tree	-0.00669	-0.00969	-0.00879	-0.00664	-0.06741
Random Forest	**0.00431**	-0.00036	**0.003**	0.00261	**0.02539**
LightGBM	0.0001	**0.0046**	-0.00055	**0.00359**	0.01108
Xgboost	0.00265	0.00144	0.0028	-0.00307	0.01603
	BPIC2012_2	**BPIC2012_3**	**BPIC2015_1**	**BPIC2015_2**	**BPIC2015_3**
Decision Tree	-0.04098	-0.06018	-0.01325	-0.02103	-0.02345
Random Forest	**0.01206**	0.01463	0.00259	0.00308	**0.00514**
LightGBM	0.00033	0.01789	0.00524	0.0086	0.00312
Xgboost	0.0064	**0.022228**	**0.00535**	**0.00802**	0.0014
	BPIC2015_4	**BPIC2015_5**	**BPIC2017_20**	**BPIC2017_30**	**BPIC2017_40**
Decision Tree	-0.06446	-0.01537	-0.01542	-0.0169	-0.01391
Random Forest	**0.03258**	0.00446	-0.00507	-0.00201	-0.00224
LightGBM	-0.00768	0.00294	0.00837	0.00235	0.00288
Xgboost	0.0138	**0.0053**	**0.01198**	**0.0171**	**0.01209**

Table 3. SHAP values for the classification model design choice.

diction according to its variations. We discuss here the results obtained for the `bucketing` design feature (see Fig. 3), which appears to be the most impactful according to Fig. 2, the type of classification model chosen (see Table 3), and the encoding method (see Table 4). We leave instead out the `drop_act` feature, which does not seem to have an impact on the performance of the model.

In Fig. 3, we observe that the impact of the design feature `bucketing` is relatively small and positive for low values (below 20) and, at least for some event logs, e.g., the BPIC2011, stronger and negative for higher values. This could be due to the combined effect of the curse of dimensionality and sample size. As the number of buckets considered increases, in fact, while the size of the feature vector remains unchanged, the higher-sized buckets tend to contain a lower number of training observations, from which it becomes harder to learn a high-performing model. To sum up, based on the results shown in Fig. 3, we suggest that choosing a lower number of buckets is generally a good design choice.

From Table 3, we can observe that the decision tree model shows the lowest SHAP value, even negatively contributing to the model performance. The result confirms the trend already observed in existing outcome-oriented predictive process monitoring studies [6]: models based on ensemble principles outperform decision trees. The other models, differently from decision trees, are instead based on ensemble principles and, therefore, are generally more robust against higher dimension input datasets [9]. In the case of bagging (RF), multiple decision trees are built on random subsets of features and data. This randomness helps reducing the dominance of any single feature, thereby reducing the impact of irrelevant or noisy dimensions. On the other hand, the boosting algorithms (LightGBM, XGB) assign higher importance to the most informative features, thus effectively reducing the dimension of the problem. To sum up, our analysis suggests that an ensemble classifier should always be preferred in outcome-oriented PPM.

	BPIC2011_1	BPIC2011_2	BPIC2011_3	BPIC2011_4	BPIC2012_1
Index	**0.00039**	**0.00508**	-0.00212	**0.00406**	**0.03083**
Aggregate	-0.0008	-0.00673	**0.00249**	-0.00496	-0.03924
	BPIC2012_2	**BPIC2012_3**	**BPIC2015_1**	**BPIC2015_2**	**BPIC2015_3**
Index	**0.02038**	**0.02012**	**0.00568**	**0.00579**	**0.00646**
Aggregate	-0.01476	-0.02269	-0.00592	-0.00652	-0.00596
	BPIC2015_4	**BPIC2015_5**	**BPIC2017_20**	**BPIC2017_30**	**BPIC2017_40**
Index	**0.01876**	**0.0056**	**0.00185**	**0.00117**	**0.00297**
Aggregate	-0.02033	-0.00373	-0.00177	-0.00143	-0.00322

Table 4. SHAP values for the encoding method design choice.

	Bucket size									
Model	1	10	20	30	39	1	10	20	30	39
			BPIC2011					**BPIC2012**		
Decision Tree	**0.004257**	**0.003379**	-0.0004314	-0.003761	-0.008527	**0.03858**	**0.03858**	**0.02379**	-0.08415	-0.09621
Random Forest	-0.00005222	-0.0000507	-0.00004967	-0.00148	**-0.0009228**	-0.005275	-0.006052	-0.000339	0.01191	0.008619
LightGBM	0.000428	0.0007286	0.0006177	-0.002685	-0.007428	-0.02017	-0.008089	-0.007931	**0.02883**	**0.03131**
Xgboost	0.0007186	0.000747	**0.001203**	**-0.001244**	-0.004534	-0.009792	-0.007307	-0.003472	0.01785	0.01785
			BPIC2015					**BPIC2017**		
Decision Tree	**0.0124**	**0.01233**	**0.003992**	-0.02707	-0.03202	**0.008328**	**0.000498**	-0.003385	-0.003385	-0.004397
Random Forest	-0.005799	-0.005793	-0.005522	**0.008644**	0.007628	0.004186	0.00001259	-0.001621	-0.001004	-0.0004125
LightGBM	-0.002255	0.002389	0.0002062	-0.006223	-0.006966	-0.001326	-0.0006834	0.0006826	0.0004067	0.0008105
Xgboost	-0.002724	-0.001934	0.001076	0.0009886	**0.0102**	-0.003683	-0.001349	**0.00286**	**0.00286**	**0.002662**

Table 5. Interaction score between design choices `bucketing` and `model`.

Regarding `encoding` methods (see Table 4), it appears that the index-based encoding has in most cases a positive, albeit relatively small, impact on the model performance, whereas the aggregation encoding method has a (small) negative impact. Although aggregation encoding leverages information from the events, it still incurs in information loss by disregarding the sequential order of events. Furthermore, event attributes may lose their characteristics as they are represented in the form of descriptive statistics rather than in their original state like in index-based encoding. Hence, we conclude that, according to our analysis, index-based should be preferred.

5.3 Analyzing the feature interaction contribution with EBM

Detecting pairwise interactions is one of the key functions of EBM, implemented by a score that captures the combined effect of two features. Briefly, given a pair of features x_i and x_j, the EBM creates a new *interaction* feature $x_{i,j} = (x_i, x_j)$. Then, the interaction score of these two features is the contribution towards the output of the model of $x_{i,j}$.

As an illustration, we discuss here the interaction scores that we obtained between `bucketing` (the most impactful design choice based on the analysis of Section 5.2) and `model`, and between `bucketing` and `encoding`. Table 5 reports the mean interaction score across all datasets between the `bucketing` and `model` design choices. We can observe that when the bucketing size is below 20, the DT model shows a higher interaction score than other models. When the bucketing

Encoding method	**Bucket size**									
	1	10	20	30	39	1	10	20	30	39
	BPIC2011					**BPIC2012**				
Index-based	**0.003773**	0.0003773	**0.0008611**	**-0.001043**	-0.003819	**0.001459**	**0.001459**	**-0.0006398**	-0.005839	-0.0104
Aggregate	-0.001813	**0.001903**	-0.001327	-0.001677	**-0.001939**	-0.001304	-0.001304	-0.001105	**0.001635**	**0.01911**
	BPIC2015					**BPIC2017**				
Index-based	-0.005708	**0.001278**	-0.002065	-0.002983	**0.002614**	**0.001155**	**0.001155**	**0.001387**	-0.003359	-0.01516
Aggregate	**0.003296**	0.001943	**0.00259**	**0.002686**	-0.02655	-0.002047	0.0005249	-0.000774	**-0.0002954**	**-0.00003619**

Table 6. Interaction score between design choices `bucketing` and `encoding`.

size exceeds 20, the DT interaction score becomes negative. This shows a typical moderation effect that can be highlighted by the EBM analysis: when choosing to use a DT, a lower bucketing size is likely to improve the model performance.

Table 6 shows the mean interaction score across all datasets between `bucketing` and `encoding`. Except for the BPIC2015 case, the table confirms the findings of the SHAP analysis: index-based encoding and low bucketing size are "good" design choices. Index-based encoding, indeed, creates a more positive interaction effect on the model performance than aggregate encoding when the size of buckets is below 20. As mentioned, the only exception to this trend is BPIC2015. This could be due to the fact that the BPIC2015 datasets refer to different municipalities — possibly with different characteristics — so the average scores for these logs may not be reliable.

6 Conclusions

We have presented a framework to understand the impact of design choices on the performance of a PPM model. The framework has been instantiated in the case of outcome-oriented PPM, using an existing GA-based model configuration generator and SHAP/EBM to explain the impact of the model design features on the model performance. We have applied the framework to several publicly available event logs, obtaining a set of general recommendations for developing high-performing outcome-oriented PPM, such as preferring a lower bucketing size, ensemble classifiers, and using index-based trace encoding.

The work presented here can be extended in many ways. New instantiations can be generated and possibly compared. These may refer to other PPM use cases and/or different search space exploration algorithms and explainability approaches. The results presented in this paper suggest that the best design choices may depend on the context, i.e., the type of process generating a log. Event log complexity meta-features can be used to increase the degree of explainability, suggesting design choices for unseen contexts. We are also working on generating synthetic event logs with predefined characteristics to be able to assess more rigorously the insights obtained from the instantiation of the framework.

References

1. C. Di Francescomarino, M. Dumas, M. Federici, C. Ghidini, F. M. Maggi, W. Rizzi, and L. Simonetto. Genetic algorithms for hyperparameter optimization in predic-

tive business process monitoring. *Information Systems*, 74:67–83, 2018. Information Systems Engineering: selected papers from CAiSE 2016.
2. C. Di Francescomarino, M. Dumas, F. M. Maggi, and I. Teinemaa. Clustering-based predictive process monitoring. *IEEE Transactions on Services Computing*, PP(99):1–1, 2017.
3. F. Hutter, H. H. Hoos, and K. Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *Learning and Intelligent Optimization*, pages 507–523. Springer, 2011.
4. S. K. Karmaker ("Santu"), M. M. Hassan, M. J. Smith, L. Xu, C. Zhai, and K. Veeramachaneni. AutoML to Date and Beyond: Challenges and Opportunities. *ACM Comput. Surv.*, 54(8), Oct. 2021. Place: New York, NY, USA Publisher: Association for Computing Machinery.
5. N. Kwon and M. Comuzzi. Genetic algorithms for automl in process predictive monitoring. In *ICPM Workshops*, pages 242–254. Springer, 2022.
6. A. Leontjeva, R. Conforti, C. Di Francescomarino, M. Dumas, and F. M. Maggi. Complex symbolic sequence encodings for predictive monitoring of business processes. In *Proc. of BPM*, pages 297–313. Springer, 2015.
7. Y. Lou, R. Caruana, J. Gehrke, and G. Hooker. Accurate intelligible models with pairwise interactions. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 623–631, 2013.
8. F. M. Maggi, C. Di Francescomarino, M. Dumas, and C. Ghidini. Predictive monitoring of business processes. In *Proc. of CAiSE 2014*, volume 8484 of *LNCS*, pages 457–472. Springer, 2014.
9. A. Mohammed and R. Kora. A comprehensive review on ensemble deep learning: Opportunities and challenges. *Journal of King Saud University-Computer and Information Sciences*, 2023.
10. B. Pfahringer, H. Bensusan, and C. Giraud-Carrier. Meta-learning by landmarking various learning algorithms. In *In Proceedings of the 17th Int. Conference on Machine Learning*, pages 743–750. Morgan Kaufmann, 2000.
11. B. A. Tama and M. Comuzzi. An empirical comparison of classification techniques for next event prediction using business process event logs. *Expert Systems with Applications*, 129:233–245, 2019.
12. N. Tax, N. Sidorova, and W. M. P. van der Aalst. Discovering more precise process models from event logs by filtering out chaotic activities. *J. Intell. Inf. Syst.*, 52(1):107–139, 2019.
13. I. Teinemaa, M. Dumas, M. L. Rosa, and F. M. Maggi. Outcome-oriented predictive process monitoring: Review and benchmark. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 13(2):1–57, 2019.
14. C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown. Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms. In *Proc. of KDD-2013*, pages 847–855, 2013.
15. I. Verenich, M. Dumas, M. La Rosa, F. M. Maggi, and C. Di Francescomarino. Complex symbolic sequence clustering and multiple classifiers for predictive process monitoring. In *BPM Workshops 2015*, pages 218–229, 2015.
16. I. Verenich, M. Dumas, M. La 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*, 10(4):1–34, 2019.
17. Q. Yao, M. Wang, H. J. Escalante, I. Guyon, Y. Hu, Y. Li, W. Tu, Q. Yang, and Y. Yu. Taking human out of learning applications: A survey on automated machine learning. *CoRR*, abs/1810.13306, 2018.