# Quantifying Explainability in Outcome-Oriented Predictive Process Monitoring

Alexander Stevens[1 [0000−0001−6140−8788]], Johannes De Smedt[1 [0000−0003−0389−0275]], and Jari Peeperkorn[1 [0000−0003−4644−4881]]

Research Centre for Information Systems Engineering, KU Leuven, Belgium

**Abstract.** The growing interest in applying machine and deep learning algorithms in an Outcome-Oriented Predictive Process Monitoring (OOPPM) context has recently fuelled a shift to use models from the explainable artificial intelligence (XAI) paradigm, a field of study focused on creating explainability techniques on top of AI models in order to legitimize the predictions made. Nonetheless, most classification models are evaluated primarily on a performance level, where XAI requires striking a balance between either simple models (e.g. linear regression) or models using complex inference structures (e.g. neural networks) with post-processing to calculate feature importance. In this paper, a comprehensive overview of predictive models with varying intrinsic complexity are measured based on explainability with model-agnostic quantitative evaluation metrics. To this end, explainability is designed as a symbiosis between interpretability and faithfulness and thereby allowing to compare inherently created explanations (e.g. decision tree rules) with post-hoc explainability techniques (e.g. Shapley values) on top of AI models. Moreover, two improved versions of the logistic regression model capable of capturing non-linear interactions and both inherently generating their own explanations are proposed in the OOPPM context. These models are benchmarked with two common state-of-the-art models with post-hoc explanation techniques in the explainability-performance space.

**Keywords:** Predictive Process Monitoring · XAI · Machine Learning · Deep Learning

## 1 Introduction

Sparked by the growing research on machine learning, the analysis of processes through data-driven approaches has seen a surge under the label process mining [1]. Recently, the subtrack of predictive process monitoring [14] has known a strong uptake as it allows identifying trends in processes concerning the obtainment of particular goals (e.g. will customers be awarded credit?), impeding bottlenecks, and whether particular activities will occur in the future. As the concrete goal is to predict the future state as accurately as possible, the anticipated trend is to increase the predictive performance with the use of deep learning instead of the more classical machine learning models [14, 4]. The intrinsic complexity of these predictive models, however, causes a lack of transparency of the model, intertwined with the inability to interpret the predictions.

In predictive process monitoring, several papers have already suggested model-agnostic explainability techniques on top of the machine learning models, e.g. SHAP/LIME [10], with similar developments in a deep learning context [3, 12]. Nevertheless, the accuracy by which these post-hoc explainability techniques reflect the effective model behaviour of the predictive model is often inadequate. Moreover, identifying faithful explanations that are interpretable remains a challenge for black box models. This has already been raised by XAI proponents [11], advising to adopt inherently interpretable models rather than trying to explain black box models when it comes down to high-stake decision-making. Nonetheless, a description of the technical limits of post-hoc XAI techniques is deemed out of scope.

To the best of our knowledge, none of these works have addressed the systematic comparison of interpretable models versus post-hoc explainability in the context of OOPPM. To this end, this paper introduces a definition of explainability that allows to compare different predictive models based on model-agnostic quantitative measures. Furthermore, given the need for more inherently interpretable machine learning models which excel in terms of the predictive performance–explainability trade-off [11, 5], the Logit Leaf Model (LLM) and the Generalized Logistic Rule Model (GLRM) are adapted to the OOPPM context. The former naturally clusters the data with a decision tree and builds linear models which are directly interpretable in the leave nodes. The latter creates binary rules from the input data with a generalized logistic rule model which is less transparent but has been shown to outperform even very intricate inference mechanisms such as neural networks [16]. We benchmark these techniques with two established techniques aimed at introducing post-hoc explainability, i.e., XGBoost with Shapley values, and recurrent neural networks with attention. The rest of the paper is organized as follows. First, Section 2 provides a brief overview of the preliminaries. Next, Section 3 defines how explainability can be quantitatively measured, while introducing two models to the field of OOPPM. This is followed by an experimental evaluation in Section 4, where the experimental setup, the implementation details and the results are reported. Finally, the models are compared alongside the conclusion in Section 5.

## 2 Preliminaries

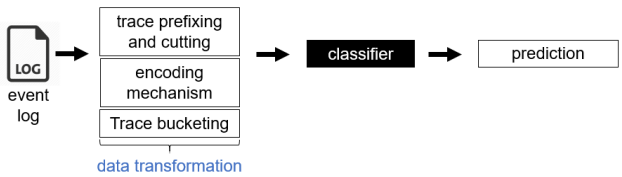


**Fig. 1.** Preliminary steps (simplified)

OOPPM relies on the use of historic process data recorded in event logs. An event log is a list of traces which represent the enactment of a particular case within an information system [1]. Moreover, a trace is considered as a sequence of timestamped events which are tuples of $p$ features $[x_1, \ldots, x_\mathrm{p}]$ such as an activity name, timestamp, and so on. On the other hand, machine and deep learning models such as the ones used for OOPPM are built to work on tabular data, with every row representing a new instance while having a fixed length of feature values. In order to use event logs in combination with AI models, the data transformation steps from Fig. 1 need to be applied.

First, (trace) prefixes are extracted from the completed cases to be able to learn, preferably incrementally, from the development of the traces. To this end, a prefix log is typically derived, which is the extracted event log that contains all the prefixes of each case in the original event log.

The second data transformation step describes the encoding mechanism [15] that enables the user to work with a varying amount of features. An often used encoding is frequency aggregation, which takes the frequencies of the categorical values while calculating the summary statistics of the numeric values (min, max, mean, sum and std). This transformation step results in a trace prefix of indefinite length being displayed as a row with a fixed amount of features, with $x'_{i,j}$ the frequency/statistics of instance $i \in [1 \ldots n]$ on the transformed feature $x'_j$, $j \in [1 \ldots p]$. Nonetheless, this encoding mechanism neglects the order of the timestamped events and therefore results in a loss of information. By contrast, the use of frequency aggregation in step-based models such as recurrent neural networks becomes superfluous given their sequential setup. To exploit this efficiently, a low-dimensional representation of discrete features in the form of embeddings is an often performed encoding [14, 12]. This mapping transforms the categorical feature to a vector of continuous numbers in a meaningful way. The use of embeddings is preferred over one hot-encoding, where high-cardinality features cause the feature space to explode while simultaneously ignoring the similarity between these vectors.

The last step before the data can be fed to a model is trace bucketing, a commonly used data transformation step that supports the discovery of heterogeneous segments in the data while creating separate models for each of them [15]. Techniques such as K-nearest neighbours or K-Means clustering measure the (dis)similarity between traces depending on the parameter k. However, while bucketing can effectively diminish the runtime performance [15], the clustering does not necessarily result in an intuitive or interpretable outcome. E.g., clustering techniques can base their grouping on a high number of dimensions that are not interpretable. Furthermore, there is no guarantee that the use of a bucketing technique will effectively improve performance [15]. The above was tested by benchmarking against the single bucketing technique, in which only one bucket is created [15].

As a final step, the model can be used to make certain predictions $y^* = F(x_j)$ with $F$ the model/function to make a prediction based on the features $x_j$ with $j \in [1 \ldots p]$.

## 3 Explainability in OOPPM

This Section introduces a general definition of explainability based on XAI concepts, which can be used to evaluate OOPPM techniques. Next, two interpretable models with varying model complexity are introduced to OOPPM. The Logit Leaf Model (as presented in this paper) is a transparent and interpretable model. The second algorithm is the Generalized Logistic Rule Model that uses column generation to find the optimal set of rule-based features and able to create its own explanations.

### 3.1 Explainability through Interpretability and Faithfulness

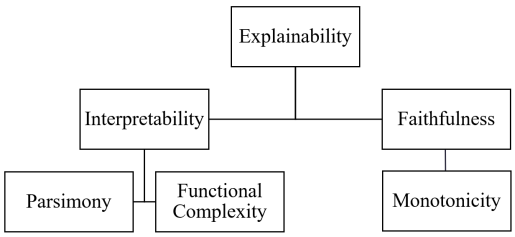


**Fig. 2.** Explainability Through Interpretability and Faithfulness

The use of the predictive models for high-stake decision-making processes is finding its way to ever more applications. While simple models are able to generate their own explanations, XAI tries to approximate the behaviour of the model with post-hoc explainable techniques such as e.g. Shapley values, feature importance, etc. This leads to a widespread urge to evaluate the faithfulness (and interpretability) of these models, by looking at whether the original task model (e.g. black box model) is accurately reflected by the explainability model (e.g. post-hoc explainability technique).
First of all, even though often used interchangeable, there is a subtle difference between interpretability and explainability. This boils down to the fact that understanding the internal working of the model is different from the ability to link the inputs with its predicted output in a faithful way. Moreover, an unambiguous and simple interpretation of a prediction has a substantial loss of trustworthiness if it does not accurately represent the effective behaviour of the model. To emphasize, a simple explanation generated for a rain forecast prediction could be: *'if the grass is green, it will rain'*, which is easy to interpret, but is unfaithfulness to the actual behaviour of rain.
The necessity to distinguish between faithfulness and interpretability was defined by [5] and is adapted in this paper for OOPPM purposes. As a result, the combination of *interpretability* (further decomposed in parsimony and functional

complexity) and *faithfulness* (by means of monotonicity), allows quantifying and thus evaluate explainability in an OOPPM context (see Fig. 2).

**Parsimony** is a property of interpretability that discusses the complexity of the model [5], an often used metric for linear regression models. In this paper, the parsimony of a model $F$ is quantified by the amount of features in the resulting model. This can be seen as the number of features with non-zero weights e.g., linear regression coefficients with a corresponding weight different from 0, in other cases the non-zero weights provided by the post-hoc feature importance. As a result, the parsimony of a model is maximally equal to the total amount of features. Moreover, a parsimonious (i.e. simple) model corresponds to a low value for $C_F$.
Assume features $x_{\mathrm{i}}$ with $a_i$ the weight of the features $i \in [1 \dots p]$ indicated by the feature importance of a model, where the total parsimony $C_F$ is calculated as followed:

$$C_F = \sum_{i=1}^{p} C(i) \text{ with}$$

$$C(i) = \begin{cases} 0, & \text{if } a_i > 0, \\ 1, & \text{otherwise.} \end{cases} \tag{1}$$

**Functional complexity** is, alongside parsimony, a metric of model complexity and measures how strong the model is dependent of the features [8].
Assume the prediction of an instance $i$ by the model $F$ is indicated by $\hat{y}_i = F(x_{i,1}, \dots, x_{i,p})$. Furthermore, the prediction after feature permutation is defined by $\hat{y}_i^* = F(x_{i,1}, ., x_{i,j}^*, ., x_{i,p})$, where $x_{i,j}^*$ is a randomly permuted feature value. The total functional complexity is calculated as the amount of prediction changes *before* and *after* permutation for all the instances and features, divided by the number of instances and the parsimony $C_F$ of the model. Therefore, the functional complexity is quantified by regarding a feature as 'used' when changing the feature changes the prediction. A lower value for $U_F$ means less model complexity and therefore higher model interpretability. This paper introduces a slightly different perspective from the original computation [8], where the functional dependency of a model was originally determined by examining *how often* the model predictions change when changing the value $x_{i,j}$ of an instance $i \in [1 \dots n]$ on a feature $j \in [1 \dots p]$.

$$U_F = \frac{1}{n} \frac{1}{C_F} \sum_{j=1}^{p} \sum_{i=1}^{n} u_{i,j} \text{ with}$$

$$u_{i,j} = \begin{cases} 1, & \text{if } \hat{y}_i \neq y_i^* \ , \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

**Monotonicity** is the notion that describes the extent to which the feature importance ranking of the explainability model is faithful to the ranking of the task model feature importance. By definition, monotonicity is ensured (i.e. M=1) if the model is able to create its own explanations, i.e. inherently generated explanations [9]. The monotonicity of a (post-hoc explainability) model therefore needs to be quantified by the Spearman's correlation coefficient [9].

$$M = \rho(a, e)$$

with $a = (|a_1|, ..., |a_p|)$ containing the absolute values of the feature weights of the task model and $e = (|e_1|, ..., |e_p|)$ the absolute values of the feature weights of the explainability model. This correlation coefficient is a non-parametric measure that takes a value between [-1,1] and describes the association of rank. A perfectly faithful model has a correlation coefficient of +1, where a loss in faithfulness corresponds with a value closer to 0. Consequently, a negative value corresponds to a negative rank association between the two feature importance weights.

### 3.2 Logit leaf Model

Evidently, the use of inherently interpretable models for high-stake decision-making is preferred over black box models [11], which leads to the introduction of the Logit Leaf Model. This model is constructed as a hybrid of logistic regression and decision tree clustering, with the former model applied in the leaves of the latter to create predictions. An initial version was proposed by [2] in the context of churn prediction where in each leaf node, only the variable that contributes to the maximal Akaike Information Criterion decrease is added, until the stopping criteria are met. Here, we adapt the general idea to the context of OOPPM.

The adapted algorithm starts similarly as a simple decision tree, with the tree splitting the data in an iterative manner into smaller, more homogeneous samples. All the possible splits are evaluated based on the information gain obtained by the decrease in entropy, where after the best possible split is performed. The stopping decision consist out of two different approaches: a maximum number of tree levels together with a minimal number of samples in a leaf. Hence, all the possible models that abide the stopping decision rules are evaluated based on their final predictive performance where only the optimal model is returned. Lastly, the assignment decision of the leave nodes is revised, where a logistic regression model is learnt for each segment of the data instead of performing majority voting.

The strength of decision trees lies in the ability to discover XOR-style interactions, as opposed to the model failing when it comes to discovering linear relations between the predictor variables. By contrast, logistic regression models are unable to deal with these interaction effects but manage to handle linear relations well. As a result, the logit leaf model overcomes both disadvantages while exploiting their respective strengths. The nature of this model ensures that the explanations consist out of a combination of decision tree rules and logistic regression coefficients, making it a faithful model by design.

The model does, however, have two major drawbacks. Firstly, while decision trees can capture the interaction effects, the logistic regression model might not deal with them well in case they are still left in impure splits. This can result in reduced predictive performance. Secondly, the model can still become large when many coefficients have higher/lower values, impeding the overall interpretability.

### 3.3 Generalized Logistic Rule Model

The second introduced model is the Generalized Logistic Rule Model [16], which combines the linear elements from logistic regression together with a conjunctive ruleset, where each rule is constructed as a conjunction of binarized features making the GLRM model a rule ensemble [6]. Here, the assumption of a Gaussian distribution of the residuals in a general linear regression model is relaxed and generalized for different distributions. By induction, a logistic regression model is therefore also a generalized linear model [16].
In order to find the optimal set of rule-based features, column generation is performed using integer programming to improve the objective function [16]. First, the GLRM model transforms the original features to rule-based features before making predictions. To this end, numerical features are binarized through bi-directional comparisons to a set of thresholds, while categorical features are one-hot encoded.
In the final model, the probability of y being classified as 'deviant' is predicted as $log(z)$, where z is a linear combination of the discovered rules.

$$log(z) = \frac{1}{(1 + e^{-z}))} \tag{3}$$

Similar to a logistic regression, the GLRM model is able to create coefficients for a single feature, but by extension also for an AND-combination of two features. Furthermore, the rule-based features can handle both linear and non-linear dependencies (analogous to LLM) as an improvement over the competencies of the logistic regression model. The strength of the GLRM model lies in the ability to reduce the amount (and length) of rules with the use of regularization parameters, a means to improve the interpretability of the rules. Moreover, $\lambda_0$ denotes the fixed cost of each rule (penalizes the amount of the rules), while $\lambda_1$ is the additional cost of each literal in rule (penalizes the length of a rule). This ensures that GLRM can compete directly in terms of performance while providing its own, relatively simple explanations.
The intrinsic complexity of this model, i.e. a column generation subproblem solved using integer programming, can be seen as a drawback in the context of high-stake decision making. Next, generalized linear models are known to be sensitive to outliers. The last drawback is the need of relatively large sample sizes, and an exponential increase of binarized features with an increase in the amount of predictor variables.

## 4 Experimental Evaluation

In this Section, the two benchmark models are briefly discussed, while indicating which post-hoc explainability techniques are used on top of these black box models. Next, the different event logs and their corresponding statistics are elaborated on. This is followed by detailed information about the implementation steps performed in this experiment. Finally, an overview of the quantitative metric results is given, which is subsequently summarized in Section 5.

### 4.1 Benchmark Models

XGBoost is one of the most widely-used ensemble methods in machine learning due to its ability to outperform most of the existing models. Several studies in a predictive process monitoring context have already used this gradient boosting machine [15], where weak learners are improved after each iteration to a final strong learner by incorporating the loss function of the previous weak learner(s). As this is a black box model, Shapley values need to be calculated in order to explain the model [13]. The Shap value for each instance-feature combination is obtained, whereby the calculation is based on coalitional game theory. Therefore, XGBoost is not an interpretable model, as the inherit complexity is what bestows the predictive abilities on this black box model. The feature importance is calculated by the average of the amount that each feature split point improves the purity (i.e. Gini index) weighted by the number of instances in the respective nodes, across all the decision trees within the model.
The second model is a recurrent neural network with Long Short-Term Memory (LSTM), with the long-term relations and dependencies encoded in the cell state vectors, therefore solving the vanishing gradient problem. The advantage of LSTM over classical machine learning models lies in the ability to model time-dependent and sequential data tasks, where the categorical values are encoded in embeddings. Similar to XGBoost, the complexity of the internal representation of an LSTM neural network does not allow for inherent explanations of predictions. Consequently, recent work in deep learning to predict the next activity have come with solutions to provide post-hoc explanations [3, 7], whereby the use attention layers to create post-hoc explainability in predictive process monitoring stems from [12]. In [3], an LSTM model in combination with Shapley values allow the user to identify the influence of certain features in the different steps of the process, while [7] focuses on creating local post-hoc explanations with the use of a surrogate decision tree. In addition, [17] introduces a widely-used approach of machine learning to offer explainability in the light of deep neural networks for remaining-time and next-activity predictions respectively. Finally, in the case of long short-term neural networks, the feature importance of the task model is approximated with the use of a perturbation method.

### 4.2 Event logs

The first event log TF1 contains notifications from an Italian local police force, e.g. the reason, the total amount, and the amount of repayments. The original

event log has 1,198,366 events divided over 129,615 cases (see Table 1). The second event log BPIC2017 assembles the execution history of a loan application process in a certain Dutch financial institution. The dataset contains events related to a particular loan application, with the label indicating if the loan application was accepted (regular), or not (deviant). The last event log BPIC2015 assembles events from the second Dutch municipality (see [15]), pertaining to the building permit application process. The different event logs can be found at the website of 4TU Centre for Research Data[1].

**Table 1.** Event Logs

| event log | events | cases | cutoff len. | features (orig.) | features (agg. enc.) | features (select.) |
|---|---|---|---|---|---|---|
| TF1 | 460,556 | 129,615 | 10 | 21 | 254 | 3 |
| BPIC2017 | 1,198,366 | 31,413 | 20 | 26 | 259 | 10 |
| BPIC2015 | 41,202 | 753 | 40 | 21 | 391 | 50 |

### 4.3 Implementation

The event logs are split on an 80/20 ratio, with the cases ordered based on their timestamp and only the first 80 percent used for training purposes (after cutting the events of the training cases that overlap with the test period), an analogue implementation approach to [15]. Next, the data transformation steps as described in Fig. 1 are performed. After trace prefixing and cutting (with a predefined cut-off length), different sequence encoding techniques are implemented for the machine learning algorithms (i.e. aggregation encoding) and the LSTM model (i.e. embedding). Therefore, the total amount of deduced columns *after aggregation encoding* in Table 1 is only applicable for the machine learning models. Lastly, no trace bucketing technique has been applied for any of the respective models. Instead, in order to improve runtime performance and interpretability, feature selection is performed, where only uncorrelated predictor features (with $\geq 10\%$ Pearson correlation with the target feature) are selected, which made trace bucketing unnecessary. Furthermore, the hyper optimization for the machine learning models is performed with the use of hyperopt[2], while the LSTM neural network has an analogue setting to [12], with the predictive function transformed into a binary outcome-oriented prediction by stripping of the final layer and inserting a sigmoid output layer instead. As a final remark, detailed information about design implementations and parameters are provided, to enhance reproducible results[3].

[1] https://data.4tu.nl/
[2] http://hyperopt.github.io/hyperopt/
[3] https://github.com/AlexanderPaulStevens/OOPPM

**Table 2.** Quantitative Metrics (overview)

| | Traffic Fines | | | | BPIC2017 | | | | BPIC2015 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **LLM** | **GLRM** | **XGB** | **LSTM** | **LLM** | **GLRM** | **XGB** | **LSTM** | **LLM** | **GLRM** | **XGB** | **LSTM** |
| **Parsimony** | 2 | 3 | 3 | 2 | 9 | 6 | 10 | 7 | 18.5 | 13 | 40 | 15 |
| **Functional Complexity** | 0.00 | 0.00 | 0.57 | 0 | 0.97 | 0.55 | 0.25 | 0.00 | 0.29 | 0.15 | 1.24 | 0.12 |
| **Monotonicity** | 1 | 1 | 1 | -1 | 1 | 1 | 0.42 | -0.43 | 1 | 1 | 0.31 | -0.12 |

### 4.4 Quantitative Metrics Results

The parsimony of a model is described with the use of an absolute number instead of, e.g., a ratio, allowing for comparability between event logs with different dimensions. Intuitively, the parsimony of a model displays the amount of features used in the explainability model, with on average the highest value denoted for the XGBoost model in contrast to the lowest value for the rule-based GLRM. Furthermore, the parsimony of the LLM tends to increase more compared to the LSTM with an increased amount of predictor variables.

The functional complexity of each model describes the dependency of the model on the features, and only makes sense when analysing the same event log. Again, the XGBoost model reports the highest value on average, while the LSTM neural network has the lowest functional dependency (on average) on its features. This intuitively boils down to the fact that changing a value in an LSTM neural network has a smaller effect on the value of the prediction due to the more complex inference structure that makes for more stable predictions. Furthermore, this metric is by design (as the parsimony metric is in the denominator instead of the total amount of column) able to demonstrate that e.g. the XGBoost model is *also* functionally dependent on feature(s) that were assigned a zero-attribution, as a value $> 1$ for event log BPIC2015 is reported. Further insights are that the functional complexity of the GLRM and LLM do not have a linear relationship with the basic statistics from Table 1, where the functional complexity in the XGB model seems to depend on the number of selected features.

As the faithfulness of both LLM and GLRM are guaranteed by definition (the task model and the explainability model are the same), only monotonicity values for the XGBoost and LSTM model have to be calculated. For the BPIC2017 event log, the Spearman's rank correlation coefficient of the XGBoost vs. Shapley value feature importance is 0.42, where the underlying message is that the Shapley values do not accurately reflect the model behaviour, indicating a loss of faithfulness. For the LSTM neural network, the feature importance calculated with the attention values versus feature importance based on the perturbation importance are the values used to calculate the monotonicity, with a negative value of -0.43 reported in Table 2 for the event log BPIC2017. This interesting value is visualized in Fig. 3, where it is clear that there is uncertainty about the influence of the time component on the model.

Lastly, the performance of the different models over the different event logs show that the introduced models are competitive with the less interpretable models (with an outlier value for the LSTM on the event log BPIC2015).

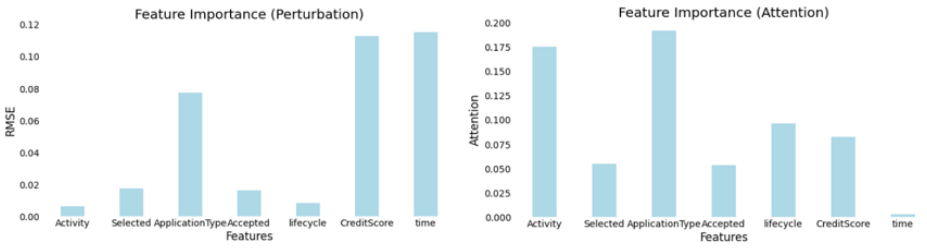


**Fig. 3.** BPIC2017 LSTM (Attention vs. Perturbation)

## 5 Conclusion

While data fuels the advances in machine learning and artificial intelligence, the centre of attention has mostly been on the computational aspects, thereby often neglecting the interpretation, actionability, and implications of the results. Moreover, an easily interpretable explanation must also be faithful to the effective model of behaviour, with the quantitative evaluation of explainability techniques on top of models with varying intrinsic complexity as an increased necessity. To this end, this paper has introduced a definition (explainability as a symbiosis between interpretability and faithfulness) and quantitative metrics (parsimony, functional complexity and monotonicity) to evaluate and rank different algorithms based on their explainability. Moreover, it is desirable that a model uses a small amount of features for its predictions (low parsimony), which are as functionally independent as possible (low functional complexity), without compromising on faithfulness (monotonicity of 1).

Furthermore, this paper also introduced two improved versions of the logistic regression model, which were found to have comparative performance results when compared with the two benchmark models. In addition, both the GLRM and LLM show better (or at least comparable) overall results based on parsimony, functional complexity and monotonicity on the three event logs with varying statistics. Lastly, the faithfulness of the explanations is ensured for these models by definition, while the study shows that the post-hoc explainability models on top of the XGBoost and LSTM models are associated with an imperfect faithfulness. As a result, the use of GLRM is recommended (over LLM) due to the overall better results. This predictive model can handle both linear and non-linear dependencies and the amount (and length) of rules can be reduced with the use of regularization parameters, which is favourable to the parsimony and possibly the functional complexity.

Future research consists of evaluating the impact of the index sequence encoding technique, seeking justification for the obtained negative values of the monotonicity of the LSTM, and identifying the most important components for evaluating model explainability. Additionally, this paper will be extended to a benchmarking study with additional methods (classic logistic regression, CNN, etc.) and metrics.

## References


1. van der Aalst, W.M.P.: Process Mining - Data Science in Action, Second Edition. Springer (2016)
2. Caigny, A.D., Coussement, K., Bock, K.W.D.: A new hybrid classification algorithm for customer churn prediction based on logistic regression and decision trees. Eur. J. Oper. Res. **269**(2), 760–772 (2018)
3. Galanti, R., Coma-Puig, B., de Leoni, M., Carmona, J., Navarin, N.: Explainable predictive process monitoring. In: ICPM. pp. 1–8. IEEE (2020)
4. Kratsch, W., Manderscheid, J., Röglinger, M., Seyfried, J.: Machine learning in business process monitoring: A comparison of deep learning and classical approaches used for outcome prediction. Bus. Inf. Syst. Eng. **63**(3), 261–276 (2021)
5. Markus, A.F., Kors, J.A., Rijnbeek, P.R.: The role of explainability in creating trustworthy artificial intelligence for health care: A comprehensive survey of the terminology, design choices, and evaluation strategies. J. Biomed. Informatics **113**, 103655 (2021)
6. McCullagh, P., Nelder, J.A.: Generalized Linear Models. Springer (1989)
7. Mehdiyev, N., Fettke, P.: Explainable artificial intelligence for process mining: A general overview and application of a novel local explanation approach for predictive process monitoring. CoRR **abs/2009.02098** (2020)
8. Molnar, C., Casalicchio, G., Bischl, B.: Quantifying model complexity via functional decomposition for better post-hoc interpretability. In: PKDD/ECML Workshops (1). Communications in Computer and Information Science, vol. 1167, pp. 193–204. Springer (2019)
9. Nguyen, A., Martínez, M.R.: On quantitative aspects of model interpretability. CoRR **abs/2007.07584** (2020)
10. Rizzi, W., Francescomarino, C.D., Maggi, F.M.: Explainability in predictive process monitoring: When understanding helps improving. In: BPM (Forum). Lecture Notes in Business Information Processing, vol. 392, pp. 141–158. Springer (2020)
11. Rudin, C.: Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. Nature Machine Intelligence **1**(5), 206–215 (2019)
12. Sindhgatta, R., Moreira, C., Ouyang, C., Barros, A.: Exploring interpretable predictive models for business processes. In: BPM. Lecture Notes in Computer Science, vol. 12168, pp. 257–272. Springer (2020)
13. Sindhgatta, R., Ouyang, C., Moreira, C.: Exploring interpretability for predictive process analytics. In: ICSOC. Lecture Notes in Computer Science, vol. 12571, pp. 439–447. Springer (2020)
14. Tax, N., Verenich, I., Rosa, M.L., Dumas, M.: Predictive business process monitoring with LSTM neural networks. In: CAiSE. Lecture Notes in Computer Science, vol. 10253, pp. 477–492. Springer (2017)
15. Teinemaa, I., Dumas, M., Rosa, M.L., Maggi, F.M.: Outcome-oriented predictive process monitoring: Review and benchmark. ACM Trans. Knowl. Discov. Data **13**(2), 17:1–17:57 (2019)
16. Wei, D., Dash, S., Gao, T., Günlük, O.: Generalized linear rule models. In: ICML. Proceedings of Machine Learning Research, vol. 97, pp. 6687–6696. PMLR (2019)
17. Weinzierl, S., Zilker, S., Brunk, J., Revoredo, K., Matzner, M., Becker, J.: XNAP: making lstm-based next activity predictions explainable by using LRP. In: Business Process Management Workshops. Lecture Notes in Business Information Processing, vol. 397, pp. 129–141. Springer (2020)