XAI in the context of Predictive Process Monitoring: Too much to Reveal

Ghada Elkhawaga[1,2][0000−0001−7801−7310], Mervat
Abuelkheir[3][0000−0003−0958−7322], and Manfred Reichert[1][0000−0003−2536−4153]

[1] Institute for Databases and Information Systems, Ulm University, Ulm, Germany
{ghada.el-khawaga,manfred.reichert}@uni-ulm.de
[2] Faculty of Computers and Information, Mansoura University, Dakahlia, Egypt
[3] Faculty of Media Engineering and Technology, German University in Cairo,
New Cairo, Egypt
mervat.abuelkheir@guc.edu.eg

Abstract. Predictive Process Monitoring (PPM) has been integrated
into process mining tools as a value adding task. PPM provides useful
predictions on the further execution of the running business processes.
To this end, machine learning-based techniques are widely employed in
the context of PPM. In order to gain stakeholders trust and advocacy
of PPM predictions, eXplainable Artificial Intelligence (XAI) methods
are employed in order to compensate for lack of transparency of most of
efficient predictive models. Even when employed under the same settings
regarding data, preprocessing techniques, and ML models, explanations
generated by multiple XAI methods differ profoundly. A comparison is
missing to distinguish XAI characteristics or underlying conditions that
are deterministic to an explanation. To address this gap, we provide a
framework to enable studying the effect of different PPM-related settings
and ML model-related choices on characteristics and expressiveness of re-
sulting explanations. In addition, we compare how different explainabil-
ity methods characteristics can shape resulting explanations and enable
reflecting underlying model reasoning process.

Keywords: Predictive Process Monitoring · Machine Learning eXplain-
ability · XAI · Outcome-Prediction · Process Mining · Machine Learning

1 Introduction

1.1 Problem Statement

Predictive process monitoring (PPM) [1,2], as a use case of process mining,
supports stakeholders with predictions about the future of a running business
process instance. A process instance represents one specific execution instance
out of all possible ones enabled by a business process and defined using a business
process model. Process mining [3] and PPM both aim at informing stakeholders
of how a business process is currently operating or expected to operate in the
near future. However, employing black box techniques does not help achieve this

purpose. As stakeholders engagement is at the center of process mining tasks, performance and accuracy are not the only aspects which matter while carrying on a PPM prediction task. While depending on ML models in predicting the future of running business process instances, it becomes necessary to persuade business stakeholders of the validity of reasoning mechanisms followed by a predictive model. Justifying predictions to their recipients enable gaining users' trust, engagement and advocacy of PPM employed mechanisms.

EXplainable Artificial Intelligence (XAI) [4] methods and mechanisms [5,6,7,8,9,10,11,12] are put in place to provide explanations of predictions generated by a ML model. However, the aforementioned explanations are expected to reflect how a predictive model is influenced by different choices made through PPM workflow. Moreover, PPM tasks employ specific mechanisms to ensure aligning process mining artefacts to ML models requirements. Therefore, while having different XAI methods addressing explainability needs in the context of PPM, studying how explanations of XAI methods differ is an important step to understand the settings suitable for employing these methods, and how to interpret explanations in terms of the underlying influencing factors. On the other hand, as the number of XAI methods increase, and while they address different purposes and different users needs using varying techniques, there is a need to compare different XAI methods outcomes given the same PPM workflow settings.

1.2   Contributions

With an attempt to address the need to understand and gain insights into the application of XAI methods into PPM, this paper presents:

- Comparison of explanations globally and locally, separated and against each other, using different PPM workflow settings using criteria predefined based on underlying data, predictive models and XAI methods characteristics.
- A study of explanations generated by three different global XAI methods, and two local XAI methods for predictions of two predictive models over process instances from 27 event logs preprocessed with two different preprocessing combinations.

Section 2 provides background information on basic topics needed for understanding this work. In Section 3, we highlight the basic research questions investigated in this paper. In Sections 4 and 5, we discuss the settings of the conducted experiments, experimental results and observations. Section 6 highlights lessons learned and conclusions answering the basic research questions. Related work is illustrated in Section 7. Finally, we conclude the paper in Section 8.

2   Backgrounds

This section introduces basic concepts and background knowledge necessary to understand our work. Section 2.1 introduces PPM and associated steps to carry

on predictions of information relevant to a running business process. Then, we discuss available explainability methods with an in-depth look into those addressing tabular data as these data type is the focus of this work.

2.1   Predictive Process Monitoring

Predictive Process Monitoring (PPM) addresses a critical process mining goal resembled in the need to provide decision makers with predictions of the future of a running business process execution instance. This goal can be realised by building models to generate predictions about running process-related information. Examples of these predictive tasks; or what we denote as PPM tasks; are the next activity to be carried out, time-related information (e.g. elapsed time, remaining time till the end), outcome of the process instance, execution cost, or executing resource [3]. Event logs are the input to PPM tasks, and it documents the execution history of a process terms of traces, each of them representing execution data belonging to a single business process instance. A trace contains mandatory attributes, such as *Case identifier*, *event class*, and *timestamp* [3]. A trace may contain *dynamic attributes* representing information about a single event. Resources fulfilling tasks and documents associated with each event are examples of dynamic attributes. Besides dynamic attributes there are *Static attributes* which have constant values for all events of a given trace.

PPM Workflow According to the survey results reported in [1,2], a PPM task follows two stages, an offline and another is online. Each stage has steps, to have the two stages with their steps constituting what we call PPM workflow.

1. **PPM offline Stage.** An offline stage starts with *constructing a prefix log.* Contemporary PPM approaches that use ML, take as input a prefix log constructed from the input event log. A prefix log is needed to provide a predictive model with incomplete process instances (i.e., a partial trace) for training. Therefore, prefixes can be generated by truncating process instances in an event log up to a predefined number of events. Truncating a process instance can be done up to the first k events of its trace, or up to k events with a gap (g) step separating each two events, where k and g are user-defined. The latter prefixing approach is called *gap-based prefixing.*

   The following step is to *preprocess prefixes*, in order to be input to a predictive model. Prefix preprocessing sub-steps include bucketing and encoding (cf. Figure 1). Prefix bucketing means grouping the prefixes according to certain criteria (e.g., number of activities or reaching a certain state in the process execution). The former criteria are defined by the bucketing technique [2]. Single, state-based, prefix length-based, clustering, and domain knowledge-based are all examples on prefix bucketing techniques. For example in single bucketing,all prefixes generated from the traces of an event log are treated as a single bucket [1,2]. Meanwhile in state-based bucketing, different process execution states are determined from the relevant process

model,and prefixes are grouped accordingly into buckets. Encoding is the second sub-step of prefixes preprocessing. Prefix encoding means transforming a prefix to a numerical feature vector that serves as input to the predictive model, either for training or for making predictions. Encoding techniques [1,2] include static, aggregation, index-based, and last state techniques. Note that static encoding is always used for static attributes encoding, and should be aggregated with one of the aforementioned techniques for encoding dynamic attributes.
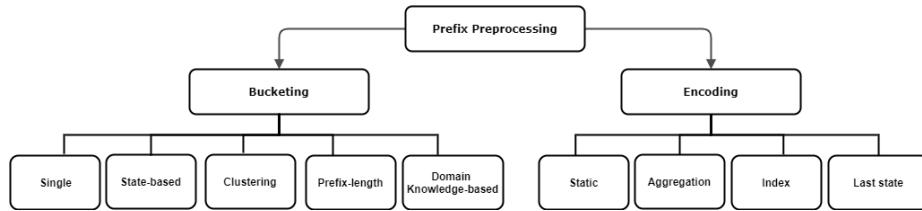

Fig. 1: Bucketing and Encoding techniques

The following step is to *construct a predictive model.* Depending on the PPM task, the respective predictive model is chosen. The prediction task type, i.e., being classification or regression, is not the only factor guiding a predictive model selection process. Other factors include scalability of the model when facing larger amounts of data, simplicity and interpretability of results. An important step after selecting a model is to find out the suitable set of hyperparameters values, using hyperparameter optimisation. Hyperparameter optimisation [13] is the automated process of defining an optimal configuration of hyperparameters that may help increase accuracy. The next sub-step is to train the predictive model on encoded prefixes representing completed process instances. Note that for each bucket a dedicated predictive model needs to be trained, i.e., the number of predictive models depends on the bucketing technique chosen. Finally, after generating predictions for the training dataset, the performance of a predictive model needs to be evaluated.

2. **PPM online Stage.** This stage starts with an incomplete process instance, i.e., a running process instance. Buckets formed in the offline stage are recalled to determine the suitable one for the running process instance. This is accomplished based on the similarity between the running process instance and prefixes in a bucket according to the criteria defined by the bucketing method. Afterwards, the running process instance is encoded according to the encoding method chosen for the PPM task. The encoded form of the running process instance becomes qualified as an input to the prediction method after determining the relevant predictive model from the models created in the offline stage. Finally, the online stage is concluded by the predictive model generating a prediction for the running process instance according to the predefined goal of the PPM task.

2.2   eXplainable Artificial Intelligence

PPM inherits the challenges faced by ML approaches, as a reasonable consequence of employing the latter approaches. One of these challenges concerns the need to gain user trust in the predictions generated. The field of explainable artificial intelligence (XAI) addresses this issue. According to [14], an explanation is *"a human-interpretable description of the process by which a decision maker took a particular set of inputs and reached a particular conclusion"*. In the context of our research, the decision maker is a ML-based predictive model. Transparency is considered an important aspect of explainability. Model transparency can be an inherent characteristic of itself or be achieved through an explanation of a model. Transparent models are understandable on their own and satisfy one or all model transparency levels [15]. Linear models, decision trees, Bayesian models, rule-based learning, and General Additive Models (GAM) are all considered being transparent/interpretable models. The degree of transparency realised through an explanation may be affected by several factors. These factors include (but are not limited to) the incompleteness of the problem formulation or understanding, the time frame allocated to evaluate the satisfaction of a certain dimension, the level of complexity of an explainability solution, and the number and length of cognitive chunks made available to the user through an explanation.

Several approaches are proposed under the umbrella of explainability. Explanations construction approaches can be categorised along several dimensions (cf. Figure 2). In the following, we illustrate these explainability dimensions.

1. **How to explain.** This dimension is concerned with the approach used to explain how a predictive model derives its predictions based on given inputs. Corresponding approaches have been categorised from different perspectives including design goals and evaluation measures [16], transparency of the explained model and explanation scope [15], granularity [17], and relation to the black box model [18]. For example, a group of approaches tend to generate an *explanation by simplification*. These approaches simplify a complex model by using a more interpretable model called *surrogate* or *proxy model*. The simplified model is supposed to generate understandable predictions achieving an accuracy level comparable to the black box one. Another group of approaches study *feature relevance*. They aim at tracing back the importance of a feature for deriving a prediction. Another family of approaches tend to *explain by example*. Approaches from this category tend to select representative samples that allow for insights into the model's internal reasoning [15,17]. The final category in this dimension tend to explain through *visualisation*, i.e., intermediate representations and layers of a predictive model are visualised with the aim to qualitatively determine what a model has learned [17].
   Approaches belonging to this XAI dimension are further categorised as being either *model-agnostic* or *model-specific*. Model-agnostic approaches are able to explain any type of ML predictive model, whereas model-specific approaches can only be used on top of specific models. For example, DeepLift
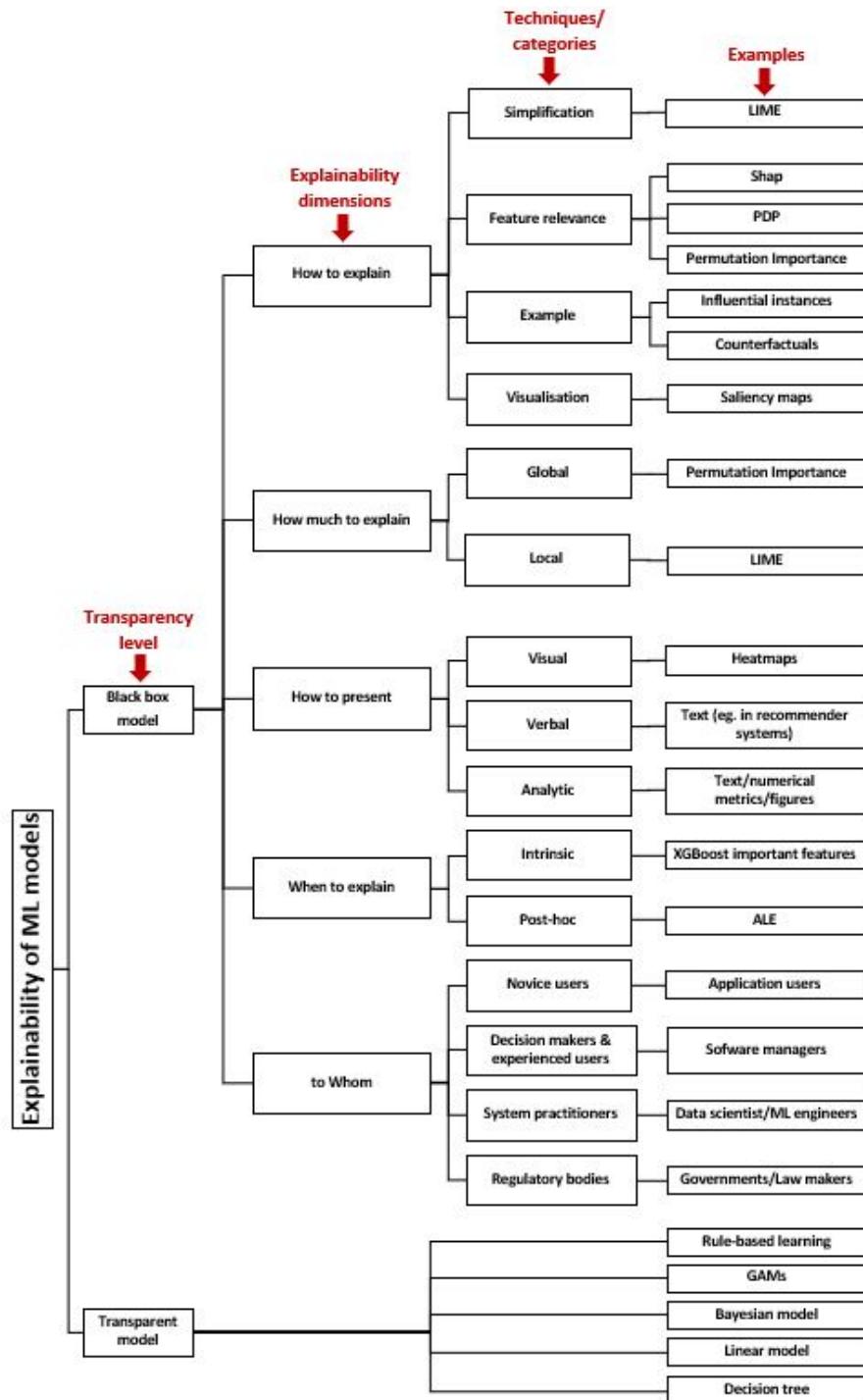
Fig. 2: Explainability taxonomy in ML

[7] and LRP [8] provide explanations for neural network-based predictive models.

2. **How much to explain.** An explanation may be generated of a various levels of granularity. An explanation effort can be localised to a specific instance or its vicinity, and it can provide global insights into which factors contributed to the decision of a predictive model. *Local explanations* are generated to study interactions between patterns and to better understand how specific input led to a certain output in a given instance, and must not be generalised to provide insights over larger number of instances [19]. *Global explanations* in turn, provide insights into patterns used by a predictive model over a large number of instances. The scope of an explanation and, subsequently, the chosen technique depend on several factors. One of these factors is the purpose of the explanation, e.g., whether it shall allow debugging the model or gaining trust into its predictions. Target stakeholders constitute another deterministic factor. For example, a ML engineer prefers gaining a holistic overview of the factors driving the reasoning process of a predictive model, whereas an end user is only interested in why a model made a certain prediction for a given instance.

3. **How to present.** Choosing the form an explanation is presented is determined by the way the explanation is generated, the characteristics of the end user (e.g., level of expertise), and the scope of the explanation, the purpose of generating an explanation, (e.g., to visualise effects of feature interactions on decisions of the respective predictive model). [16] introduces three categories of presentation forms. The first one comprises *visual explanation* that uses visual elements like saliency maps [10] and charts to describe deterministic factors of a decision in accordance to the chosen explained perspective of a model. *Verbal explanation* provides another way of presenting explanations where natural language is used to describe model reasoning (e.g., in recommender systems). The final form of presentation is *analytic explanation* where a combination of numerical metrics and visualisations are used to reveal model structure or parameters, e.g., using heatmaps and hierarchical decision trees.

4. **When to explain.** This dimension of an explainability approach is concerned with the point in time an explanation shall be provided. Agreeing on explainability as being a subjective topic, and depending on the receiver's understanding and needs, we can regard explainability provision from two perspectives. The first one considers explainability as gaining an understanding of decisions of a predictive model and being bounded by model characteristics. Adopting this perspective, explainability is imposed through mechanisms put in place while constructing the model to obtain a white-box predictive model, i.e., *intrinsic explanation*. Using an explanation method to understand the reasoning process of a model in terms of its outcomes is called *post-hoc explanation* of a predictive model. The second explainability perspective provides an understanding in terms of the whole reasons behind the mapping process between inputs to outputs. Moreover, it provides a holistic view of input characteristics which led to predictive model decisions.

Following this perspective implies placing explainability techniques, including pre-modelling, modelling and post-modelling explainability techniques, throughout the whole development pipeline.

5. **Explain to Whom.** Studying the target group of each explainability solution becomes necessary to tailor the explanations and to present them in a way that maximizes interpretability of a predictive model and forms a mental model of it. The receivers of an explanation should be at the center of attention when designing an explainability solution. Those receivers can be categorised into different user groups including novice users, decision makers and experienced users, system practitioners, and regulatory bodies [15,16]. Targeting each user group with suitable explanations contributes to achieving the explanation process purpose. The purpose of an explanation can be about understanding how a predictive model works or how it makes decisions, or which patterns are formed in the learning process, or which features are crucial to reaching a decision, or any other purpose. Therefore, it is important to understand each user group, identify its relevant needs, and define design goals accordingly.

Note that the various dimensions are tightly interrelated, i.e., a particular choice in one dimension, might affect choices made in other dimensions. Making choices on the different dimensions of the presented taxonomy is guided by several factors. These factors include enhancing understandability and simplicity of the explainability solution. In addition, the availability of software implementations of explainability methods and whether these implementations are model-agnostic or specific, might be deterministic factors of the choice. Moreover, the type of output of each explanation method and hence choosing a suitable presentation type which suits the target group, are also additional factors to be considered. Having explanations serving different user groups with different explainability goals mandates putting explainability evaluation techniques in place to ensure obtaining explanations serving their intended purpose.

3   Research Questions

The goal of this research is to study how explanations are affected by underlying PPM workflow-related choices. It is crucial to study different XAI methods characteristics influencing the final outcome of explanation process. Overall, this leads to the following research questions (RQ)s:

RQ1: How can different XAI methods be compared? Conducting a benchmark study that allows comparing all available explainability approaches is not likely to be possible [20]. This is due to the varying characteristics of these approaches in terms of the dimensions, as mentioned in Section 2.2. However, as many techniques have been proposed for evaluating explanations [21,22], a constrained study would be useful to compare the relative performance of explainability methods that have been applied in the context of PPM approaches. In addition, the consistency of explainability methods applied to PPM results

needs to be studied in order to shed light on one of the potential vulnerabilities of XAI methods, namely sensitivity of explanations. To this end, we subdivide **RQ1** into the following sub-research questions:

RQ1.1: To what extent are explanations consistent when executing an explanation method several times using the same underlying settings?

RQ1.2: How are the explanations which are generated by a XAI method affected by predictive model choices?

Explainability methods vary by the extent to which an explanation can be generalised over several data samples that belong to the same vicinity. Explainability methods also vary in the granularity of the explanations they provide and their suitability to explain a number of data samples independent of whether they are small or large, i.e., independent of whether the explainability method is local or global. However, note that the number of explained data samples comes with a computational cost. Therefore, we add another sub-research question to compare explainability methods with respect to their execution time:

RQ1.3: How do explainability methods differ in terms of execution time?, and how needed time differs with respect to different dataset characteristics, preprocessing choices, and chosen predictive models.

4   Experiments

This section describes choices and findings of the experiments we performed to compare basic XAI methods. For the basic infrastructure of a PPM outcome prediction task, we are inspired by the framework and findings demonstrated in [2] and available at [23]. Note that we are not changing any of the steps carried out through the aforementioned framework. Settings preservation is needed to be able to observe the impact of the settings described in [2], given the reported performance of studied predictive models and preprocessing techniques, from an explainability perspective.

Figure 3 shows a taxonomy of implemented experiments organized under dimensions resembling a ML model creation pipeline, being aligned with the PPM offline workflow, and incorporating an explainability-related dimension. These dimensions are a means to categorise our experiments with the aim of answering the research questions introduced in Section 3. These dimensions are further discussed through this section. All experiments were run using Python 3.6 and the scikit-learn library [24] on a 96 core of a Intel(R) Xeon(R) Platinum 8268 @2.90GHz with 768GB of RAM. Note that we apply all available combinations from each dimension, using the taxonomy defined in Figure 3 in a dedicated experiment while fixing other options from other dimensions. The code of executed experiments is available through our Github repository[1] to enable open access for interested practitioners.

---

[1] `https://github.com/GhadaElkhawaga/PPM_XAI_Comparison.git`
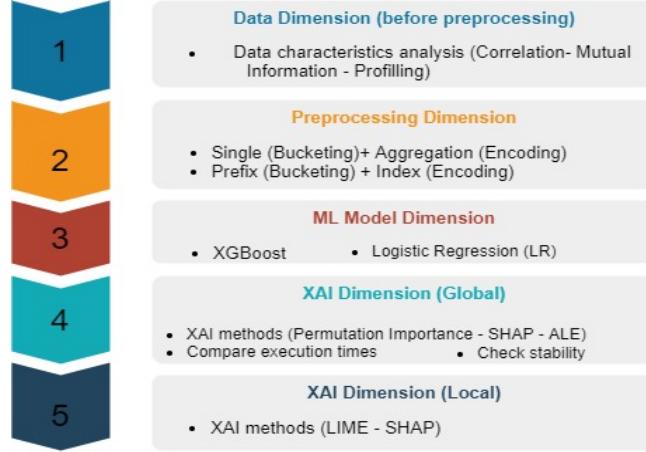
Fig. 3: Experiments Taxonomy

4.1   Experimental setup

In this subsection, we describe the building blocks of our experiments, including data, chosen preprocessing techniques, selected predictive models, and employed XAI algorithms. Categorising our experiments, and the techniques we employed (as shown in Figure 3) is done with the aim of studying the change of explanations. Observing the impact of modifying different parameters at each dimension is planed to the be the basis of this study.

Data dimension The experiments are carried on three real-life event logs which are publicly available from the 4TU Centre for Research Data [25]. The chosen event logs vary in the considered domain (government and banking), the number of traces (representing process instances), and the number of events in each trace. These event logs further vary also in the number of static and dynamic attributes, the number of categorical attributes and, as a result, the number of categorical levels available through each categorical attribute. The three basic event logs [25] used are as follows:

- **Sepsis.** This event log belongs to the healthcare domain and reports cases of Sepsis as a life threatening condition.
- **Traffic fines.** This event log is a governmental one extracted from an Italian information system for managing road traffic fines.
- **BPIC2017.** This event log documents load application process in a Dutch financial institution.

[2] applies several labelling functions to classify each process instance into one of two classes, i.e., a binary classification task. Applying different labelling functions resulted in sub-versions for some of the event logs. These labelling variations result in three extracted logs from the *Sepsis* event log, and three

logs extracted from *BPIC2017* event log. These different labelling functions increased the number of used event logs from three to seven event logs. Table 1 shows basic statistics of the event logs used in our experiments. These event logs are cleaned, transformed, and labelled according to the rules defined by the framework available in [2].

Table 1: Event logs statistics.

Event log	# traces	Short. trace len.	Avg. trace len.	Long. trace	Max prfx len.	# trace variants	%pos class	# event class	# static col	# dynamic cols	# cat cols	# num cols	# cat levels static cols	# cat levels dynamic cols
Sepsis1	776	5	14	185	20	703	0.0026	14	24	13	28	14	76	38
Sepsis2	776	4	13	60	13	650	0.14	14	24	13	28	14	76	39
Sepsis3	776	4	13	185	31	703	0.14	14	24	13	28	14	76	39
Traffic fines	129615	2	4	20	10	185	0.455	10	4	14	13	11	54	173
BPIC2017_Accepted	31413	10	35	180	20	2087	0.41	26	3	20	12	13	6	682
BPIC2017_Cancelled	31413	10	35	180	20	2087	0.47	26	3	20	12	13	6	682
BPIC2017_Refused	31413	10	35	180	20	2087	0.12	26	3	20	12	13	6	682

Preprocessing dimension. Referring back to the preprocessing techniques discussed in Section 2.1 and presented in Figure 1, we make two choices about bucketing and encoding techniques. For bucketing traces of chosen event logs, we apply single bucketing and prefix-length bucketing, with a gap of 5 events. Moreover, we apply aggregation-based and index-based encoding techniques. Normally, both encoding techniques are coupled with static encoding to transform static attributes into a form, in which they can serve as input to a predictive model. As a result, we obtain two combinations of bucketing and encoding techniques namely, *single-aggregation* and *prefix-index*. As index encoding leads to dimensionality explosion due to encoding each categorical level of each feature as a separate column, we decided to apply this encoding on certain event logs, i.e., *Sepsis* (the three derived event logs), *Traffic_fines* and *BPIC2017_Refused.*

ML model dimension In this study, we employ two predictive models, i.e., XGBoost-based and a Logistic regression (LR). LR can be considered as a transparent model for the same reasons as in the context of a linear regression model, (cf. Section 2.2). In LR, to each predictor, i.e., feature, a weight is assigned. These weights can be used to indicate how the predictive model has used relevant features in its reasoning process. XGBoost is an ensemble-based boosting algorithm that has proven to be efficient in the context of several PPM tasks [1,2]. XGboost is supported by a mechanism to query the model and retrieve a ranked list of important features upon which the model has based its reasoning process.

XAI dimension Referring back to explainability taxonomy from Figure 3 and the dimensions of an explanation (cf. Section 2.2), we decided to choose certain explainability methods (*how*), at both explainability levels (*how much*),

presented in a certain form (*presentation*), at a certain stage of the predictive model lifetime (*when*). Note that XAI methods we employed in this study fall all under the *model-agnostic* category. Meanwhile, we conducted a complementary study [26] conducting experiments using *model-specific* methods. To address the current RQs of this study, we considered only model-agnostic XAI methods. Making choices in this dimension are impacted by certain factors:

- The ability of an explainability method to overcome the shortcomings of other methods that explain the same aspects of the reasoning process of a predictive model. For example, Accumulated Local Effects (ALE) [12] is adopting the same approach as Partial Dependence Plots (PDP) method [6]. However, unlike PDP, ALE takes the effects of certain data characteristics (e.g., correlations) into account when studying features effects [4].
- Comprehensiveness regarding explanation coverage by using local and global explainability methods. Through local explanations, the influence of certain features can be observed. In turn, through global explanations, the reasoning process a predictive model has followed can be inspected. This approach allows reaching conclusions that may provide a holistic view of both, the data and the model applied on the data. Note that it is hard to find a single explainability method that provides explanations at both levels. However, one of the applied methods (i.e., SHAP [5]) starts at the local level by calculating contributions of the features on a prediction. It then aggregates these contributions at a global level to give an impression of the impact a feature has on the whole predictions based on a given dataset.
- The availability of a reliable implementation of the explainability method. This implementation should enable the integration of the explainability method with the chosen predictive model as well as in the underlying PPM workflow. For example, as we are not using a deep learning model, model-specific explainability methods specialised with deep learning models are not an option in our experiments.

In Table 2, we categorize each explainability method we applied in the experiments according to the dimensions of an explanation (cf. Section 2.2). Note that the information in the (*whom*) column which corresponds to the explanations' user groups dimension is case-dependent. Information given in this column is initial and is highly flexible according to several factors including users expertise, application domain and the purpose of the explainability experiment.

Table 2: Explainability dimensions applied on inspected XAI methods.

eXML method	How (explain)	Specificity	How much	How (present)	When	Whom
Permutation Feature Importance (PFI)	Feature importance	Model-agnostic	Global	Numerical	Post-hoc	System practitioners
ALE	Feature effects	Model-agnostic	Global	Analytic	Post-hoc	System practitioners, decision makers & experienced users
SHAP	Feature contributions	Model-agnostic	Global & Local	Analytic	Post-hoc	Novice users, System practitioners,
LIME	Simplification	Model-agnostic	Local	Analytic	Post-hoc	Novice users, System practitioners, decision makers & experienced users

4.2   Experiments description

Referring back to the taxonomy from Figure 3, we carry out different experiments at each PPM task dimension. For preprocessing and ML model dimensions, experiments conducted are related to analysing the influence of using different techniques (in case of the preprocessing dimension) and the model employed (in case of the ML model dimension). To study the effects of data characteristics, as well as choices made in preprocessing and ML dimensions, we conduct a complementary study concentrating on these factors [26]. In this study, we concentrate on how explanations differ based on characteristics and deficiencies of XAI methods. We study how XAI methods highlight the same facts about a predictive model differently. Therefore, here we concentrate on reasoning behind our choices in the XAI dimension.

XAI dimension Explainability experiments are divided into two sets, global and local experiments, based on the coverage of the explainability method applied, and the type of XAI method used. After executing each group of experiments, we compare the results of the different XAI methods at the level they are applied on.

Global explainability analysis In this set of experiments, we aim at understanding how a predictive model learns patterns from the training subset of an event log it is fitted to. We use training subsets for two reasons. First, we want to understand how much the model relies on each feature for making predictions. This can be achieved by studying the change in model accuracy after modifying a certain feature value. To this end, training subsets are more qualified to provide insights into this aspect. Second, we include two model-specific methods in our study, which build their outcomes through the training phase of the predictive model, i.e., based on the training subset of the event log. We have to unify our study of XAI methods outcomes to be based on training subsets of event logs. This decision is made in order to be able to compare the outcomes of model-specific methods with the ones of model-agnostic methods used in the same set of experiments.

As mentioned previously in this section, Permutation Feature Importance (PFI), ALE and SHAP (the global form) are the model-agnostic methods we used. To check stability of executions, we run the whole experiments taxonomy with different settings twice. Stability checks following this definition are expensive to run, due to expensive computational costs. These costs are affected by the number of datasets with different sizes (where some of the datasets experience dimensionality explosion after the preprocessing phase, and this complicates subsequent explainability steps), and the number of explainability methods applied. As a result, running the whole experiments taxonomy in the context of stability checks was not possible to be done for more than twice. Maybe depending on smaller event logs in the future enables more systematic stability check over higher number of runs.

To compare the outcomes of all XAI methods we followed the steps illustrated in Algorithm **??**. For PFI and SHAP methods, we study highly correlated features and their importance according to the XAI methods. In addition, we study how the applied model-agnostic XAI methods analyse the importance of features denoted as important to the predictive models through their model-specific explanations, i.e., coefficients in case of Logit and features importance in case of XGBoost. In addition to the aforementioned comparisons, we compare execution times for all applied XAI methods, including time for imitating the explainer and computing features importance as well. We included the training time of the predictive model as the execution time for model-specific methods.

Local explainability analysis In this set of experiments, we apply two model-agnostic XAI methods, namely LIME [11] and SHAP [5], analyse their outcomes separately. We apply variable and coefficient stability analysis [22] on LIME to study the stability of important feature sets and their coefficients across several runs.

5   Results and Observations

It is important to view explanations in the light of all contributing factors, e.g., input characteristics, the effect of preprocessing inputs, and the way how certain predictive model characteristics affect its reasoning process. In [26], we study the effect of different input characteristics, preprocessing choices, and ML models characteristics and sensitivities on resulting explanations. In this section, we illustrate the observations we made during the experiments defined in Section 4. Due to lack of space, we focus on the most remarkable outputs illustrated by figures and tables. Further results can be generated by running the code of the experiments, which can be accessed via our Github repository. Note that this public availability enables experiments replication and code reusability.

5.1   Global methods comparability

In this subsection, an analysis of PFI, ALE, SHAP results is presented. We execute two runs of each XAI method to query LR and XGBoost models trained over the event logs preprocessed with single aggregation and prefix index combination. Results are compared in order to get insights into stability of results.

Permutation Feature Importance (PFI). The basic idea of PFI is to measure the average between the error in prediction after and before permuting the values of a feature [4]. Each of the two PFI execution runs included 10 permutation iterations. The mean importance of each feature is computed. PFI execution led to the following observation:

> In single-aggregated event logs, the results of the two runs are consistent with respect to feature sets and the weights of these features. In prefix-indexed event logs, the two runs are consistent in all event logs.

An exception in prefix-indexed event logs is present in ones derived from *BPIC2017_Refused*. In the latter event logs, the dissimilarity between the feature sets across the two runs increases with increasing length of the prefixes. This observation can be attributed to the effect of the increased dimensionality in the event logs with longer prefixes. For prefix-indexed event log, weights of important features change with increasing prefix length.

Accumulated Local Effects (ALE). ALE [12] calculates the change in prediction as a result of changing the values of a feature, while taking features interactions into account [4]. This implies dividing feature values into quantiles [27] and calculating the difference for feature values with a little shift above and below the feature value within a quantile. The described mechanism complicates calculating ALE effects for categorical features, especially one-hot encoded features. However, recently the common python implementation of ALE was modified to compute ALE effects for one-hot encoded features using small values around 0 and 1 [27]. Despite the workaround, computing ALE effects for categorical attributes can be criticised for being inaccurate, because values of these features don't maintain order [4], [27].

Another issue is that after running ALE over a given event log, effects of each feature are not in a form that yields a rank directly. Therefore, we ranked features based on their entropy.

While comparing two execution runs of ALE over the analysed event logs, we make the following observation:

> The encoding technique applied plays a critical role in (dis)similarity between feature ranks in the two execution runs. As an overall observation, ALE tends to be unstable through two runs.

In single-aggregated event logs, there is no similarity between the most influencing features in two execution runs. Meanwhile, in prefix-indexed event logs, this observation is not valid in all cases. For example, in prefix-indexed versions of all *Sepsis* logs, there is no similarity between features ranks in both execution runs. Meanwhile, in *Traffic_fines* and *BPIC_Refused* encoded using the same combination, the top ranked features are categories derived from the same categorical attributes. Note that due to the inefficiency of ALE to compute effects of categorical attributes, we lean towards a conclusion that a rank where categories derived from categorical attributes dominate, tends to be highly affected by collinearity between categories of such features. This conclusion is valid especially in event logs encoded with a technique which increases the number of categorical attributes exponentially, i.e., index encoding. In aggregation encoding where results from two runs disagree, ranks tend to be more reliable. Aggregation encoding tends to increase the number of ordinal categorical attributes and preserve the existence of numerical attributes. Therefore, feature ranks of a single run tend to be less affected by collinearity. These observations are valid regardless of the underlying predictive model, which enables neutralising the effect of characteristics of the used predictive model.

SHapley Additive exPlanations (SHAP). SHAP is an explanation method belonging to the class of feature additive attribution methods [5]. These methods use a linear explanation model to compute the contribution of each feature to a change in the prediction outcome with respect to a baseline prediction. Afterwards, a summation of the contributions of all features approximates the prediction of the original model. To maintain comparability of the global XAI methods used in our experiments, we constructed a SHAP explainer model on training event logs independently of another SHAP explainer model constructed on relevant testing event logs. The concluded observations made in this section are drawn based on the training SHAP explainer model, whereas the observations based on the testing SHAP explainer model are discussed in Section 5.1 along with other observations on local XAI methods.

> While comparing the two execution runs, results did not depend on the preprocessing technique used, but differed depending on the predictive model being explained.

While explaining predictions of the LR model, performing two executions of the SHAP method did neither result in different feature sets nor different ranks based on SHAP values, regardless the preprocessing combination used. Meanwhile, explaining predictions of XGBoost model reveals having the first most contributing feature as being the same across both execution runs, while the rest of the feature set is the same, but differs in features ranks. An exception is present in the feature set of the three *Sepsis* event logs, where feature ranks are the same across both runs.
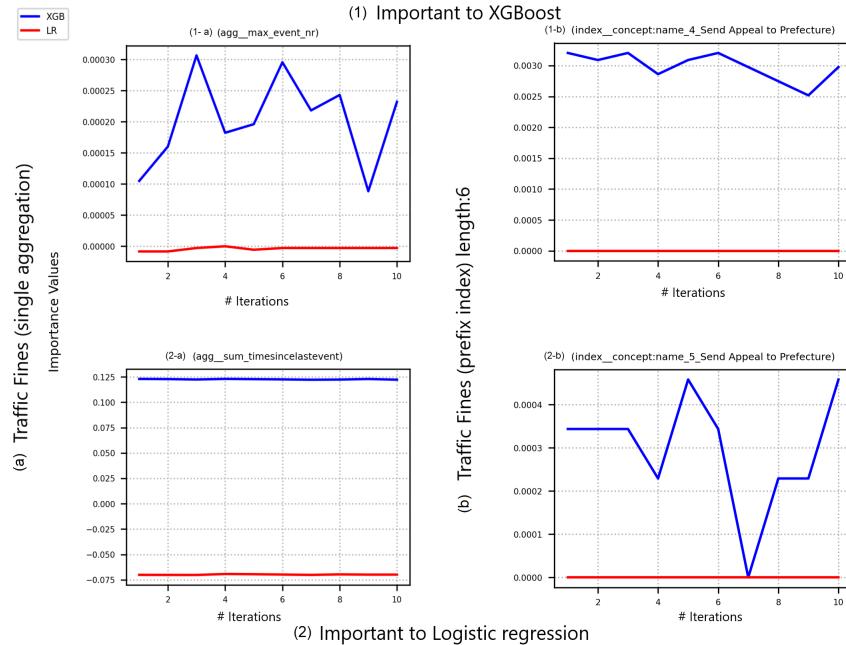

Fig. 4: PFI scores over *Traffic_fines* event log

Comparability Studying how PFI, ALE and SHAP analysed the effects or contributions of the most important feature for each predictive model, results in some interesting observations:

> (1) The cardinality of categorical features has an effect on explanations generated by permutation-based methods which depend on measuring prediction error after changing a feature value. The effect of the used preprocessing combination is dependent on the used predictive model.

For features with lower number of categories, PFI is not able to capture the effect of shuffling feature values on predictions. The effect of shuffling a feature value is static in most event logs predictions using LR, approaching zero in most cases. However, a slight change across PFI iterations can be observed in event logs preprocessed with single aggregation techniques. This observation indicates an effect of the encoding technique along with the interactions between the features. Shuffling values in PFI aims to break dependencies between the features [4]. However, with a low cardinality of a feature, the chances of reducing dependencies decreases over a few number of iterations.

PFI over XGBoost is presenting slightly higher shuffling effects in prefix-indexed event logs than in LR. This effect is slightly changing over shuffling iterations in the same event logs. However, the increase of changes in XGBoost predictions over iterations of shuffled feature values are observed in prefix-indexed event logs to be higher than in single-aggregated ones. Figure 4 shows PFI scores for *Traffic_fines* event log, that was preprocessed with single aggregation (Figures 4(1-a), 4(2-a)) and prefix index (Figures 4(1-b), 4(2-b)). Figures 4(1-a), 4(1-b) represent change in prediction errors for both predictive models while changing the top important features to XGBoost and Figures 4(2-a), 4(2-b) represent the same for the top two important features to LR.

Dependence plots in SHAP represent an illustrative way presenting the effect of low cardinality in some features. These plots offer analyses of the effect of changing a feature's values on SHAP values while taking the interaction effect of another feature into account. As opposed to PFI, this facility enables acquiring more information from categorical features. For example, Figure 5 shows SHAP values of a feature (CreditScore_other) according to both, XGBoost and LR predicting outcomes over *BPIC2017_Refused* event log preprocessed with single aggregation combination. Note that this feature is indicated as the most important one according to LR coefficients and as one of the top five important features based on the XGBoost gain criterion. This feature is a binary feature. However, it has multiple categorical levels due to being encoded based on its frequency of occurrence in a process instance (cf. Section 2.1 for encoding types). SHAP values of the feature form a nonlinear curve in XGBoost, and a linear one in LR. In both sub-figures, a point corresponds to a process instance. A point is colored according to its value of an interacting feature, which in this case is "std_event_nr". According to LR, shap values increase linearly with increasing "CreditScore_other" values. The interacting feature values are increasing, as well. However, as a result of having all points of the same "CreditScore_other" value with identical shap value, it is unclear whether all points of the same

"CreditScore_other" have the same "std_event_nr" value. Meanwhile, according to XGBoost, there is an interaction between both features resulting in having points with the same "CreditScore_other" value to obtain different shap values. These shap values decrease as the values of "std_event_nr" increase for points with the same "CreditScore_other" value.
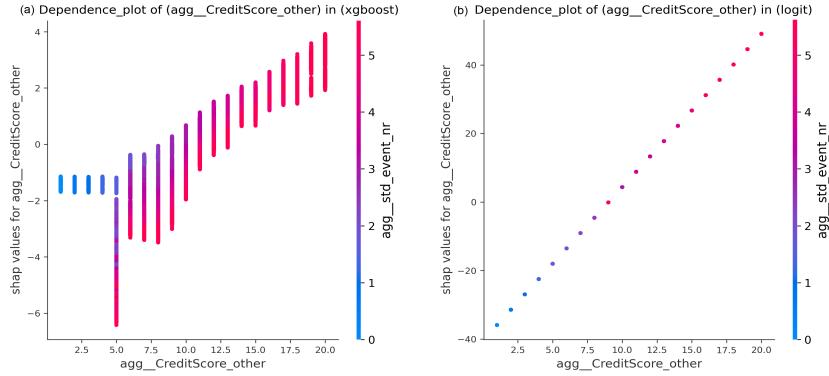

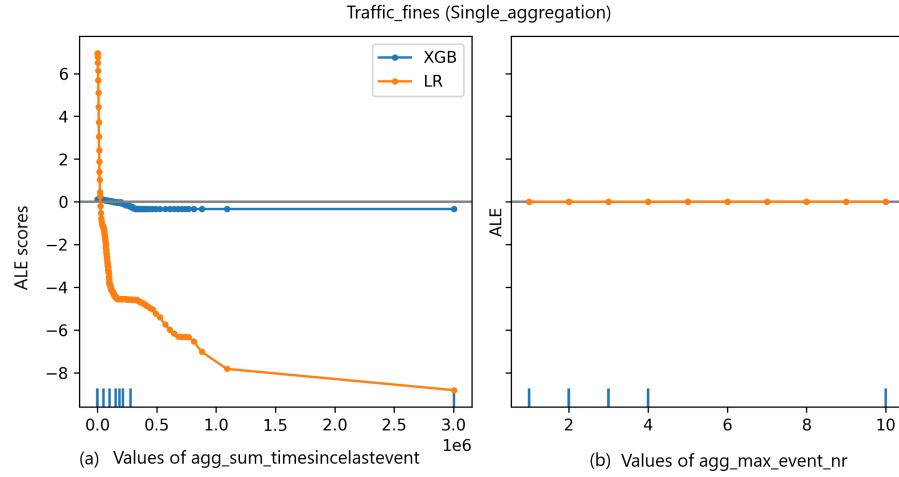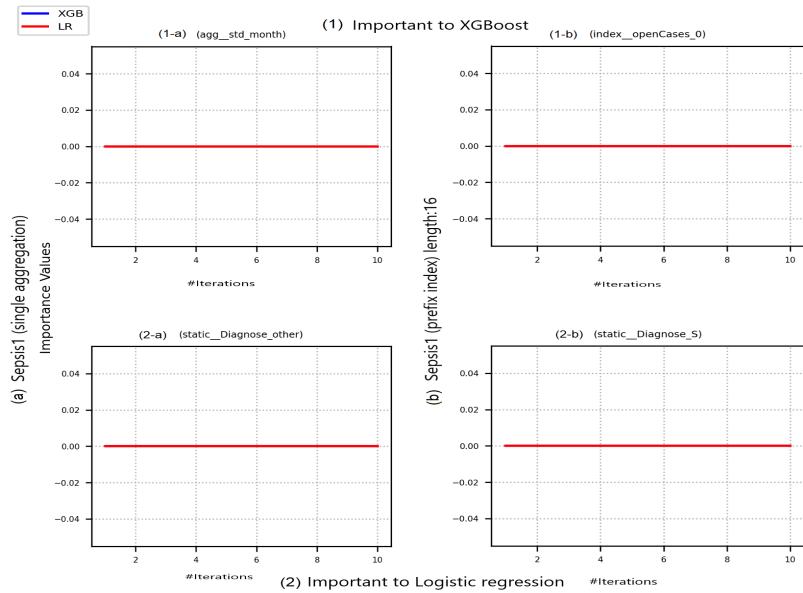Fig. 5: SHAP Dependence plots of *BPIC2017_ Refused* (single aggregation) event log

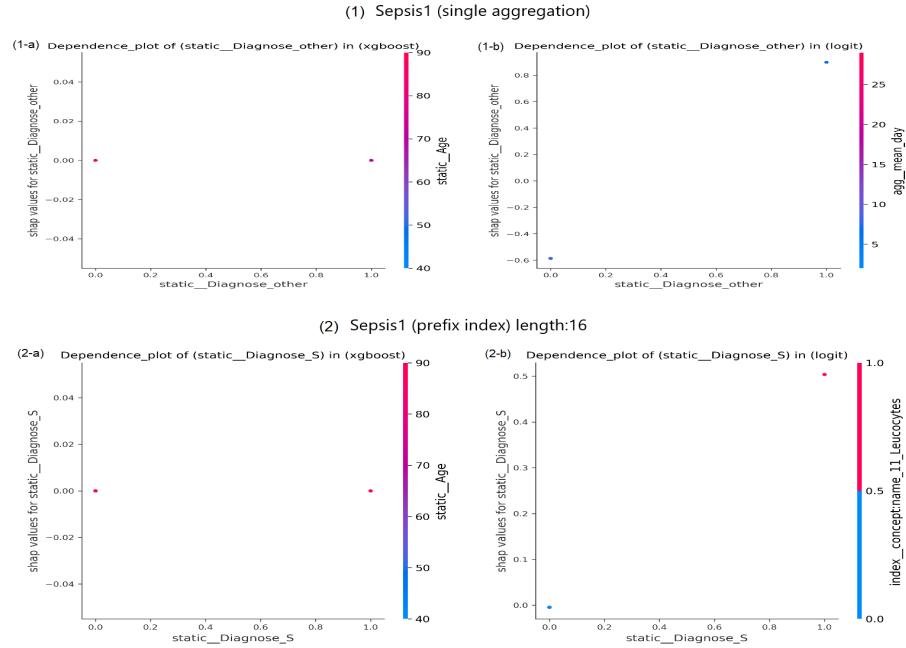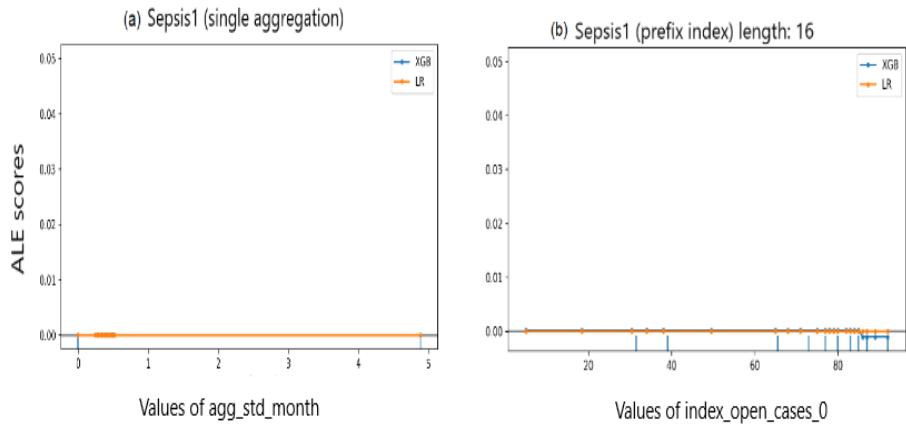(2) In all event logs, ALE plots for LR are linear themselves.

An exception to this observation is present in *Traffic_fines* event logs preprocessed with single aggregation. In figure 6(a), the importance of feature "sum_timesincelastevent", as indicated by LR coefficients, is confirmed by the change in ALE scores with a sudden decrease with increasing feature values. Although Figure 6 also shows a drecrease in XGBoost predictions, the expected decrease is stable and not steep after "sum_timesincelastevent" is reaching a value of $(0.5 * 10^6)$ unlike the case with LR model. However, in both cases ALE plot is interpolating as a result of the big gap in feature distribution. Note that the deciles on the x-axis represent interval edges at which ALE scores are calculated. Meanwhile, in between the deciles are parts in which the ALE plot is interpolating. Absence of data might be an issue in non-linear models, where the ALE plot is interpolating and hence results may be unreliable in such regions.

(3) As expected, explaining how a predictive model uses a feature in predicting outcomes of an event log with class imbalance does not reveal a lot of information.

This observation is confirmed in results of the three XAI methods, especially when explaining LR reliance on important features. Results of event log *Sepsis1* shown in Figures [7-9] represent an example of this observation.

In Figure 7, despite the plotted features are indicated as important to the predictive models, after shuffling feature values over 10 iterations, there is no observed effect on the prediction error. Meanwhile, in Figures 8(1-a) and 8(2-a),

Fig. 6: ALE plot of *Traffic_fines* event log

Fig. 7: PFI scores over *Sepsis1* event log

Fig. 8: SHAP Dependence plots of *Sepsis1* event log

Fig. 9: ALE scores of *Sepsis1* event log

the feature is showing no contribution to change in XGBoost predictions. This observation is represented by the zero SHAP values scored by the categorical features "Diagnose_other" (Figure 8(1-b)) and "Diagnose_S" (Figure 8(2-b)) in single-aggregated and prefix-indexed versions of *Sepsis1*, respectively. While a change is observed in LR predictions as a contribution of "Diagnose_other" in the single aggregation-encoded *Sepsis1*, without effect of the interaction with the feature "mean_day". However, in Figure 8(2-b), there is an effect of the interaction between "Diagnose_S" and "concept:name_11_Leucocytes". It is unclear whether this is the main affecting interaction, as all points with the same value of "Diagnose_S" provide the same contribution in terms of same SHAP values. ALE as PFI and SHAP are unable to reveal more information about *Sepsis1* and its related event logs. In Figure 9, both predictive models are not affected by changes in analysed features values, despite being indicated as the most important features. In both sub-figures, the ALE plot is linearly interpolating in between available data. However, ALE scores in both sub-figures are nearly zero.

As an example, consider the global analysis of *BPIC2017_ Accepted* event log, which is shown in Figures [10-12]. The high cardinality of the analysed features along with the high number of process instances expose the effect of value change on LR predictions, especially using PFI as shown in Figure 10. The same factors are affecting important features to XGBoost using SHAP and ALE as shown in Figures [11-12], especially with ALE where the plot is less interpolating with more dispersion of data. In SHAP dependence plots, SHAP values are following a non-linear form, while the interaction effect is clear even in "std_MonthlyCost" feature which is more important to LR than XGBoost.

Execution Times Execution times for applied XAI methods are computed for SHAP, ALE and PFI over the training event logs. Prediction times of LR and XGBoost are included as the time for computing features important to the predictive models. However, the prediction time is computed as the average of three prediction trials. Execution times are presented in Tables 3 and 4, for execution times of event logs preprocessed with single aggregation and prefix index combination respectively. Note that in the event log column in Table 4, each event log is augmented with the length of its prefixes.
 In terms of the underlying preprocessing technique, the overall numbers across all XAI methods, show that event logs preprocessed with prefix index combination are faster to be explained than those preprocessed with single aggregation.

> (1) As a result, we conclude that the chosen bucketing technique is a determinant factor affecting execution duration of global explanations generation.

As illustrated by [1,2], in index encoding, as the prefix length increases, the number of features increases, while the number of process instances decreases in an event log. This fact justifies the faster execution times on prefix length-preprocessed event logs can be based on the reduced number of process instances in an event log compared to a single-aggregated event log. Despite enabling
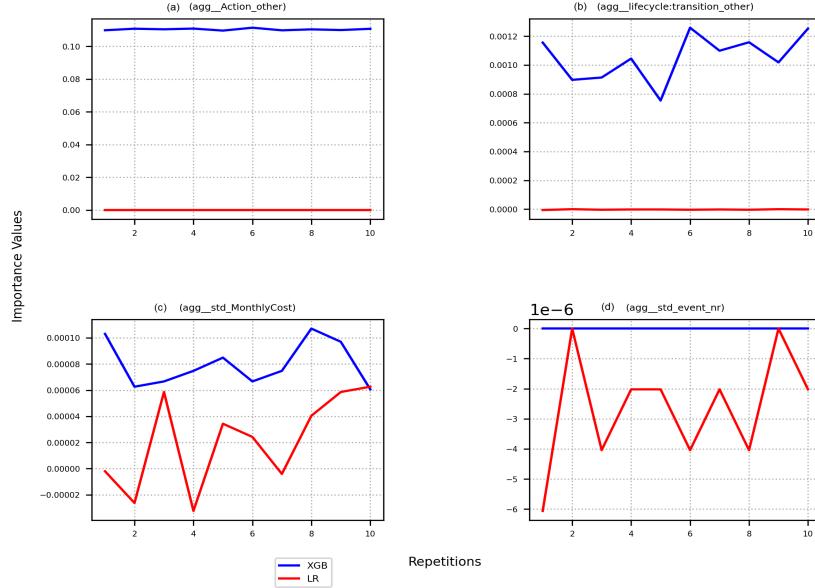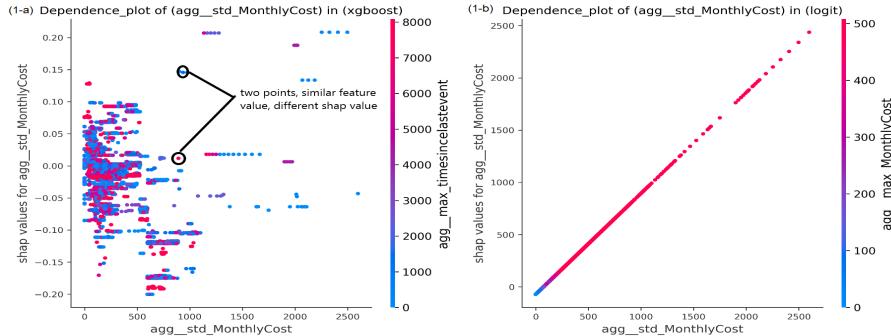
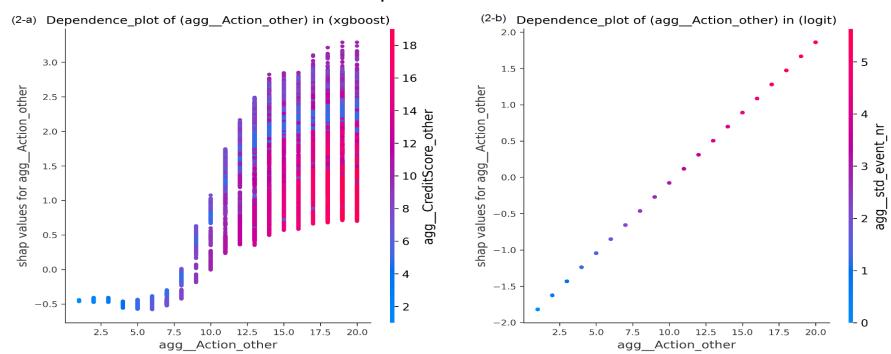Fig. 10: PFI scores of *BPIC2017_Accepted* event log

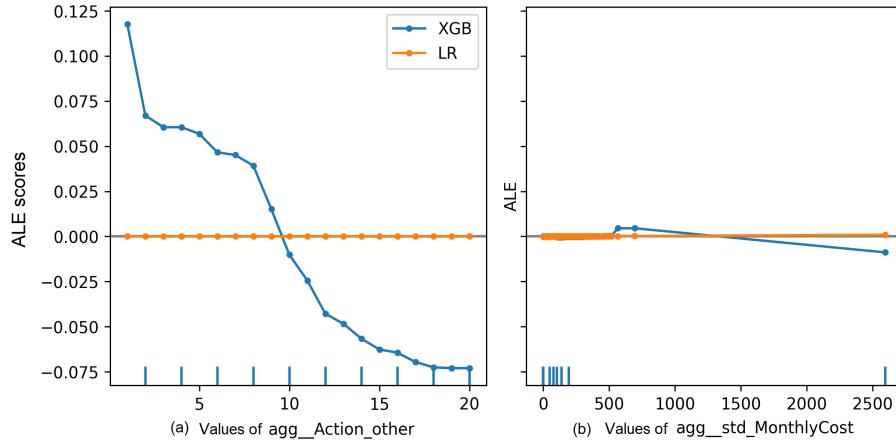Fig. 11: SHAP Dependence plots of *BPIC2017_Accepted* event log

Fig. 12: ALE scores of *BPIC2017_ Accepted* event log

Table 3: Execution times (in seconds) of XAI methods on event logs preprocessed using single aggregation combination.

Event log	XGBoost				Logistic regression (LR)			
	Prediction	SHAP	ALE	PFI	Prediction	SHAP	ALE	PFI
Sepsis1	8.71	0.20	15.5605	10.54	0.1265	0.00645	8.00679	4.773
Sepsis2	16.74	2.87	13.3611	12.32	0.061	0.00687	5.77876	4.8206
Sepsis3	30.53	9.55	19.15117	19.71	0.092	0.00585	8.2588	6.4126
Traffic_fines	4285.7	91145.55	6516.123	4834.37	10.395	0.6789	5769.03	139.29
BPIC2017_Accepted	3794.15	288.73	147845.645	2179.73	29.89	2.638	144387.72	1782.8
BPIC2017_Cancelled	10000.33	34131.79	149374.35	11421.7	36.36	2.666	144490.6677	995.43
BPIC2017_Refused	5294.84	764.84	148574.512	2992.14	25.278	2.655	144083.7028	554.774

faster executions, as expected, prefix-indexed event logs with longer prefixes show longer execution times over all XAI methods than event logs with shorter prefixes.

> (2) the overall numbers show that explaining decisions of a LR model takes less time compared to an XGBoost-based model.

This observation is proved to be true by results across different event log sizes, preprocessing settings and over different XAI methods. A justification of this observation may be that all the studied global XAI methods involve querying the underlying model. The boosting mechanism employed in XGBoost complicates the prediction process, and hence demands more time to produce either predictions, or subsequently, explanations.

> (3) While regarding execution times of different XAI methods, it is observed that SHAP has the fastest executions over almost all event logs, regardless the used preprocessing techniques with few exceptions.

An exception to this observation is the execution time of SHAP over the shortest prefix version of *Traffic_fines* event log being preprocessed with prefix index combination and having an XGBoost model trained over its process instances. SHAP values for this event log are the slowest among all XAI methods used over this event log.

> (4) The slowest XAI method differs based on the preprocessing techniques used, while having ALE performing the worst in most cases.

In prefix-indexed event logs, PFI achieves the worst performance in most event logs while combined with logit, and ALE is the worst combined with XGBoost. Meanwhile, in single aggregation-based event logs, ALE provides the least efficient XAI method to be combined with either predictive models. PFI is performing the worst in smaller event logs with relevant small numbers of features compared to other logs, e.g., in the three *Sepsis* event logs.

Table 4: Execution times (in seconds) of XAI methods on event logs preprocessed using prefix index combination.

Event log	XGBoost				Logistic regression (LR)			
	Prediction	SHAP	ALE	PFI	Prediction	SHAP	ALE	PFI
Sepsis1_1	0.388	0.0098	3.575	6.61	0.013	0.00051	0.346	3.16
Sepsis1_6	0.75	0.0116	9.059	5.509	0.025	0.00045	1.1537	3.7387
Sepsis1_11	1.112	0.012	15.948	7.01	0.021	0.00075	2.137	4.166
Sepsis1_16	0.496	0.0055	19.394	6.59	0.015	0.00031	2.421	3.6
Sepsis2_1	0.77	0.1797	3.804	6.41	0.01	0.00022	0.269	3.26
Sepsis2_6	1.74	0.138	8.757	11.249	0.0075	0.00052	1.08999	5.92
Sepsis2_11	1.203	0.028	15.122	7.52	0.007	0.000558	2.3622	4.38
Sepsis3_1	0.475	0.101	3.4351	8.725	0.0678	0.00013	0.2655	4.366
Sepsis3_6	0.999	0.0704	8.93579	6.566	0.0684	0.00045	1.1067	5.8
Sepsis3_11	1.73	0.0416	16.015989	6.943	0.0947	0.00055	2.2307	4.788
Sepsis3_16	0.6166	0.0083	18.9257	7.216	0.067483	0.0003	1.7351	5.054
Sepsis3_21	0.426	0.00284	20.954	6.6596	0.041555	0.00016	1.4537	5.134
Sepsis3_26	0.245	0.0017	22.42477	7.23	0.029335	0.00008	1.49295	5.749
Sepsis3_31	0.16	0.00111	24.785	8.676	0.026648	0.000072	1.5815	6.022
Traffic_fines_1	1999.87	74009.516	335.4324	2632.54	0.903	0.21602	72.1658	32.99
Traffic_fines_6	157.018	10.881	233.545	47.776	0.422	0.05684	78.58	20.71
BPIC2017_Refused_1	113.23	212.403	21.412	319.73	0.07	0.0125	5.84036	27.45
BPIC2017_Refused_6	847	414.45	860.8566	515.45	1.034	0.21099	345.113	84.71
BPIC2017_Refused_11	2521.397	341.585	9184.41	2012.69	2.488	0.7547	3744.7	461.968
BPIC2017_Refused_16	3775.526	279.275	29963.995	6542.17	3.8	1.99307	12625.59	2331.89

5.2   Local methods comparability

LIME and SHAP, which are the two local XAI methods used in this study, share the underlying mechanism. Both are feature additive attribution methods [5]. LIME is looking for weights of features, to indicate their importance, While SHAP calculates contributions of features to shifting the current process instance prediction towards or apart from a base prediction. To gain insights into how LIME and SHAP results can differ while explaining the same process instances, we compared their explanations of both predictive models reasoning, i.e., LR and XGBoost, over selected set of process instances from each event log preprocessed with both preprocessing combinations. After analysing resulting explanations, the following observations are made:

> (1) It could be rarely observed when both methods match in their attributions to important features or in the strength and direction of the effect features have on driving the current prediction towards or away from the base prediction.

It is supposed that whenever both XAI methods explain the prediction generated by the same predictive model for the same process instance, a similarity between explanations exists. However, the former observation is valid across all event logs regardless preprocessing combinations used. If both methods agree on a subset of features, they do not have the same importance ranks. Note that in LIME, the goal is to identify a set of features which are important in confirming the current prediction or driving the prediction towards the other class in case of binary classification tasks. Meanwhile in SHAP, the purpose is to order features descending based on their SHAP values, i.e., their contributions in driving the current prediction away or towards the base prediction over the whole event log. For example, consider the two explanations in Figure 13. The Figure shows the LIME explanation (Figure 13 (a)), and the SHAP decision plot (Figure 13 (b)) of prediction made by LR of the same process instance of *Traffic_fines* event log preprocessed with prefix index. In Figure 13 (a), important features are listed in descending order along with their values in the event log based on coefficients of these features in the approximating model created by LIME. On the bottom of the figure, these features are listed again with the percentage of their contributions. Note that the color code of these features repesent whether a feature is driving the prediction towards the current predicted class or towards the other class. In this case, the currently predicted class is "deviant".

Figure 13(b) shows the decision plot highlighting the features with the highest shap values in a descending order. The line at the center represents the base prediction or model output. The zigzag line represents the current prediction and it stricks the top at exactly the current prediction value. Predition units are in log odds. The pattern at which the zigzag line is going towards and outwards from the line at the center represents contributions of features at the y-axis to reducing/increasing the difference between the base and current predictions. In addition, feature values are stated in brackets over the zigzag line at the point where the line intersects with the horizontal line. It can be observed that the decision plot is indicating that the current prediction of the LR model is highly

affected by collinear features.

(2) In explanations of both methods, dynamic attributes are dominating feature sets.

(a) LIME explanation

Variable stability: 36.67%
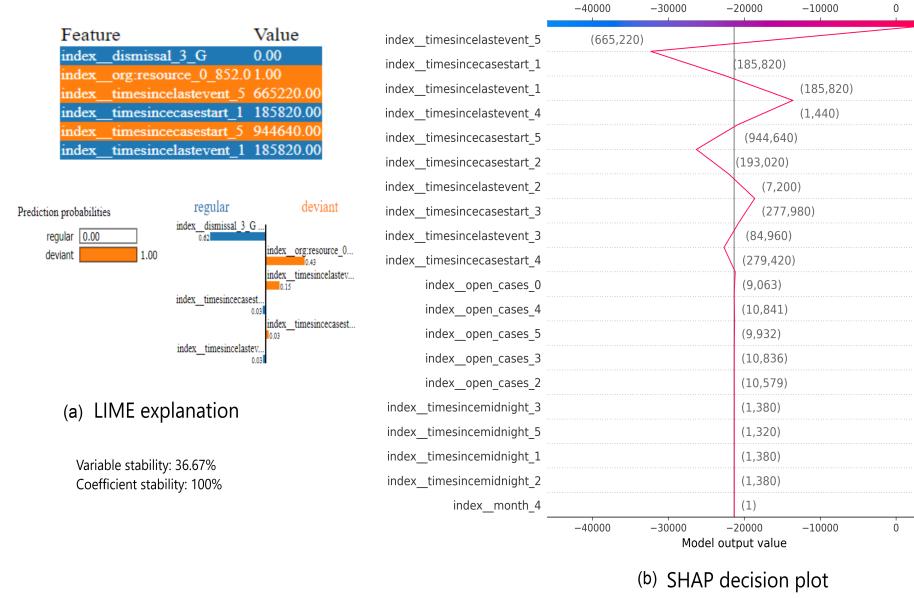Coefficient stability: 100%

(b) SHAP decision plot

Fig. 13: LIME vs. SHAP explanations of LR prediction for one instance in *Traffic_fines* preprocessed with prefix index combination

This observation about explanations can be justified by having all event logs, except for the three derived from *Sepsis* event log, with a larger number of static attributes and a relatively lower number of dynamic ones (cf. Table 1). Therefore, dynamic attributes as expected are dominating feature sets provided by local explanations. This observation may be justified by the fact that both encoding techniques employed are able to increase the number of features derived from dynamic ones, especially in event logs when the dynamic features dominate. Another example explanation is presented in Figure 14. It shows LIME (Figure 14(a)) and SHAP (Figure 14(b)) explanations of a prediction generated by XG-Boost model for a process instance of the *BPIC2017_Cancelled* event log while being preprocessed with single aggregation combination. In both explanations "CreditScore_other" is the first or second important feature. In LIME, this feature is the only one driving the prediction towards the currently predicted class. Meanwhile, in SHAP, the same feature has the highest SHAP value and is highest value in driving the current prediction away from the base prediction. This means that this feature has the highest impact according to both explanations.

As mentioned in Section 4.1, stability measures, namely Variable Stability Index (VSI) and Coefficient Stability Index (CSI) are applied on LIME explanations. VSI [22] measures to what extent the same set of important features will

(a) LIME explanation

Variable stability: 93.7%
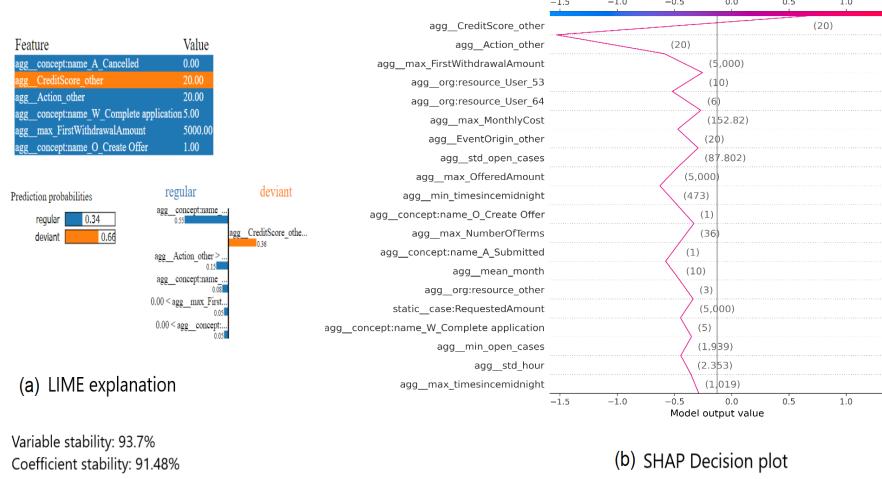Coefficient stability: 91.48%

(b) SHAP Decision plot

Fig. 14: LIME vs. SHAP explanations of XGBoost prediction for one instance in *BPIC2017_Cancelled* preprocessed with single aggregation combination

be generated if LIME is executed several times over the same event log under the same conditions. Meanwhile, CSI [22] measures the stability in coefficients of features within the important feature set over several runs of LIME.

> (3.1) As expected, Variable Stability Index (VSI) for the randomly selected process instances from different event logs show high instability in LIME explanations over several runs.

This observation can be justified by the high dimensionality of analysed event logs. According to [22], the underlying approximating model used by LIME, fits a dataset generated based on multivariate distribution of features in the original dataset. The original model is queried for labels of the newly generated dataset. Afterwards, the distance between the newly generated data points and the sample to be explained is measured. In case of high dimensionality, it is not possible to distinguish between distant and near points. The latter phenomena will result in generating a dataset that will differ greatly from the explained sample. As a result, the approximating model is expected to be locally inaccurate compared to the original model [5,22], besides having a different approximating model at each LIME run. Consequently, different feature sets will result whenever querying the approximating model.

> (3.2) VSI of LIME explanations of XGBoost predictions are higher in balanced event logs while being higher in imbalanced event logs in case of LIME explanations of LR predictions.

For example, VSI for the explanation of the instance in Figure 14 is 93.7%.

(3.3) CSI measures are high for LIME explanations of XGBoost predictions for all event logs, while the same measures are zeros for explanations of predictions by LR on almost all imbalanced event logs.

This observation can be justified by the underlying collinearity of data and sensitivity of linear models; in this case the approximating one; to such phenomena. Consequently, the approximating model is expected to assign unstable coefficients to collinear features at each run of LIME. Therefore, CSI measures are unstable.

6   Discussion

Explaining ML-based predictions is a necessity for gaining user acceptance and trust in a predictive models' predictions. It is necessary to regard explainability as a continuous process which should be integrated throughout the whole ML pipeline. A first step towards such integration would be a study of the effect of different pipeline decisions on resulting explanations. Our main concern in this research is to study the ability of an explanation to reflect how a predictive model is affected by different settings in the ML pipeline. Experiments results described and analysed in Section 5 confirm the following conclusions:

- Both studied encoding techniques load the event log with a large number of derived features. However, the situation is worse in index-based encoding, as the number of resulting features is increasing proportional to the number of dynamic attributes, especially the number of categorical levels of a dynamic categorical attribute. Dimensionality explosion has an effect on explanations generation as well as predictions generation. On one side, explaining high dimensional event logs becomes expensive in terms of computational resources, especially in XAI methods which run multiple iterations to rank features based on their importance, for example in the case of PFI. We can call this "the horizontal effect of dimensionality". Furthermore, some other XAI methods can not work on lower cardinality features, e.g., ALE , or can work but will not yield useful insights. A high dimensional event log may hold non-useful features which may have been used by the predictive model while can not be used to explain the prediction generated. This can be called "the vertical effect of dimensionality". However, SHAP is the only XAI method among those compared to be able to mitigate the effect of lower cardinality and produce meaningful explanations while highlighting the effect of interactions in dependency plots. These effects are observed in explanations of process instances from event logs preprocessed using index encoding more than aggregation-based preprocessed event logs.
- Increased collinearity in the underlying data is another problem resulting from encoding techniques with varying degrees. The effect of collinearity is observed in index-based preprocessed event logs, while not completely absent in aggregation-based event logs. This collinearity is reflected through

explanations of predictions on process instances from prefix-indexed event logs as the length of a prefix increases. Another effect of collinearity is the instability of LIME explanations. This instability is due to the approximating model being affected by collinear features (in terms of unstable feature coefficients) and high dimensionality (in terms of unstable feature sets).

– PFI showed to be more stable and consistent along two execution runs, while SHAP stability is affected by the underlying predictive model while may be (in)sensitive to underlying data characteristics. ALE is mostly unstable and affected by its inability to accurately analyse effects of changes in categorical attributes on predictions generated.

Observations made in the context of this study raise potential opportunities for further research. Setting criteria for evaluating XAI methods is a demanding need to ensure acceptance and trustworthiness of XAI methods themselves. Measurements should be put in place to ensure stability and replicability of results across several runs of a XAI method. Sensitivity of a XAI method to changes in underlying data or underlying predictive model should be measured and an XAI method should be evaluated for. This issue highlights a need to regard an explainability method as an optimisation problem where different underlying choices affect the final outcome, as we prove by results of this study.

7   Related Work

Some work available from related research areas is considered complementary to the work we present in this paper. These research efforts may enable gaining solid understanding of how different aspects of predictions explainability are studied in the context of PPM, besides shedding light on possible research gaps.

7.1   Leveraging PPM with explanations

[28] proposes an approach which integrates Layer-wise Relevance Propagation(LRP) to explain next activity predicted using an LSTM predictive model. This approach tends to propagate relevance scores backwards through the model to indicate which previous activities were crucial to obtaining the resulting prediction. This approach provides explanations for single predictions, i.e, local explanations.

Another approach explaining an LSTM decisions is present in [29]. However, [29] claim that their approach is model-agnostic and not dependent on the chosen predictive model. This approach predicts the execution of certain activities, besides predicting remaining time and cost of a running process instance. The total number of process instances where a certain feature is contributing either positively or negatively to a prediction is identified at each timestamp for the whole dataset. This identification is directed by SHAP values. [29] used the same approach in providing local explanations for running process instances.

Explanations can also be used to leverage a predictive model performance as proposed by [30]. Using LIME as a post-hoc explanation technique to explain

predictions generated using Random Forest, [30] identified feature sets which contributed the producing wrong predictions. After identifying these feature sets, their values are randomised, provided that they don't contribute to generating right predictions for other process instances. The resulting randomised dataset is then used to retrain the model again till its perceived accuracy is improved.

7.2   Using transparent models in PPM tasks

Proposals in [32,31] represent attempts to provide a transparent PPM approach, supported by explanation techniques with the aim of providing a transparent predictive model. [32] conducts experiments on three different predictive models to predict next activity only using control-flow information, next activity supported with dynamic attributes, and next activity along with remaining time. [32] uses LSTM with attention mechanisms to provide attention values indicating attention weights used to direct the attention of the predictive model to certain features. According to claims in [33] and [34], there is a lot of argument in the field of Natural Language processing (NLP) about whether to use attention weights as explanations, as attention weights are not always correlated to feature importance [33].

[31] proposed a process-aware PPM approach to predict the remaining time of a running process instance, while providing a transparent model. However, [31] could not prove that the proposed approach provides comprehensible explanations from the user perspective. Meanwhile, the proposed approach includes discovering a process model, training a set of classifiers to provide probabilities of gates in the discovered process model, and training a set of regressors to perform the main prediction task. These procedures are source of computation overhead which is not confirmed or disproved by the authors of [31].

8   Conclusions

In this research, a framework is implemented for comparing explanations generated by a selected number of currently available XAI methods. The XAI methods under analysis explain decisions of ML models generating predictions in the context of PPM. Our study includes comparing XAI methods at different granularity levels, given different underlying PPM workflow decisions. A study of how an XAI method is able to reflect a predictive model's sensitivity towards underlying data characteristics is provided. Meanwhile, different XAI methods are compared according to different criteria including stability and execution duration, over different granularities, i.e., globally and locally. This study has revealed how explanations can highlight data problems through analysing the model reasoning process. It has also emphasized based on experiments, the importance of feature selection after preprocessing an event log. Our study has highlighted situations where data problems may not affect the accuracy of predictions, but do affect usefulness and meaningfulness of explanations. Explainability should be seamlessly integrated into PPM workflow stages as an inherent task not as a follow up effort.

References

1. Verenich, I., et al.: Survey and Cross-benchmark Comparison of Remaining Time Prediction Methods in Business Process Monitoring. ACM Trans. Intell. Syst. Technol. **10**(4), 34 pages (2019). https://doi.org/10.1145/3331449

2. Teinemaa, I., et al.: Outcome-Oriented Predictive Process Monitoring: Review and Benchmark. ACM Trans. Knowl. Discov. Data **13**(2), 57 pages (2019). https://doi.org/10.1145/3301300

3. Van der Aalst, W.: Process Mining: Data science in Action. 2nd ed. Springer-Verlag Berlin Heidelberg (2016)

4. Molnar, Christoph: "Interpretable machine learning: A Guide for Making Black Box Models Explainable", (2019). Available online at `https://christophm.github.io/interpretable-ml-book/`.

5. Lundberg, S.; Lee, S.: A unified approach to interpreting model predictions. NIPS'17: Proceedings of the 31st International Conference on Neural Information Processing Systems (2017), pp. 4768–4777.

6. Friedman, J. H.: Greedy function approximation: A gradient boosting machine. Ann. Statist. **29**(5), (2001), pp. 1189–1232. https://doi.org/10.1214/aos/1013203451.

7. Shrikumar, A.; et al.: Learning important features through propagating activation differences. ICML'17: Proceedings of the 34th International Conference on Machine Learning, (2017), pp. 145–3153. https://doi.org/arXiv:1704.02685

8. Binder, A.; et al.: Layer-Wise Relevance Propagation for Neural Networks with Local Renormalization Layers. In Villa, A.; Masulli, P.; Rivero, A.J.P. (Eds.): Artificial Neural Networks and Machine Learning – ICANN (2016), vol. 9887. Springer, cham (LNCS), pp. 63–71. https://doi.org/10.1007/978-3-319-44781-0_8

9. Wachter, S.; et al.: Counterfactual Explanations Without Opening the Black Box: Automated Decisions and the GDPR. Harvard Journal of Law & Technology **31**(2), (2018).https://doi.org/10.2139/ssrn.3063289

10. Kindermans, P.; et al. : The (Un)reliability of Saliency Methods. In Samek, W.; Montavon, G.; Vedaldi, A.; Hansen, L.K; Müller, K. (Eds.): Explainable AI: Interpreting, Explaining and Visualizing Deep Learning (2019), vol. 11700. Cham: Springer, cham. (LNCS), pp. 267–280. https://doi.org/10.1007/978-3-030-28954-6_14

11. Ribeiro, M. T.; et al.: "Why Should I Trust You?". In Krishnapuram, B.;et al. (Eds.): Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD '16: The 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. San Francisco California USA, (2016). New York, NY, USA: ACM, pp. 1135–1144 (2016).https://doi.org/10.1145/2939672.2939778

12. Apley, D. W.; Zhu, J.: Visualizing the effects of predictor variables in black box supervised learning models. Journal R. Stat. Soc. **82** (4), pp.1059–1086 (2020). https://doi.org/10.1111rssb.12377

13. Yang, L., Abdallah Shami, A.: On hyperparameter optimization of machine learning algorithms: Theory and practice. Neurocomputing, **415**,295–316 (2020) https://doi.org/10.1016/j.neucom.2020.07.061

14. Doshi-Velez, F., et al.: Accountability of ai under the law: The role of explanation (2017), Berkman Klein Center Working Group on Explanation and the Law, Berkman Klein Center for Internet & Society working paper. Available online at `http://nrs.harvard.edu/urn-3:HUL.InstRepos:34372584` https://doi.org/arXiv:1711.01134

15. Arrieta, A. B.; et al.: Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. Information Fusion **58**(3), pp.82–115 (2020). https://doi.org/10.1016/j.inffus.2019.12.012.

16. Mohseni, S.; Zarei, N.; Ragan, E. D.: A Multidisciplinary Survey and Framework for Design and Evaluation of Explainable AI Systems. In ACM Trans. Interact. Intell. Syst. **11** (3-4), pp.1–45 (2021). https://doi.org/10.1145/3387166

17. Zhou, J.; et al.: Evaluating the Quality of Machine Learning Explanations: A Survey on Methods and Metrics. Electronics **10** (5), 593–612 (2021). https://doi.org/10.3390/electronics10050593

18. Guidotti, R.; et al.: A Survey of Methods for Explaining Black Box Models. ACM Comput. Surv. **51** (5), 1–42 (2019). https://doi.org/10.1145/3236009

19. Belle, V.; Papantonis, I.: Principles and Practice of Explainable Machine Learning. In Frontiers in big data (2021). https://doi.org/10.3389/fdata.2021.688969.

20. Nguyen, A.; Martínez, M. R.: On quantitative aspects of model interpretability (2020). Available online at http://arxiv.org/pdf/2007.07584v1. https://doi.org/arXiv:2007.07584v1

21. Nguyen, A.; Martínez, M. R.: On quantitative aspects of model interpretability. (2020) Available online at http://arxiv.org/pdf/2007.07584v1. https://doi.org/arXiv:2007.07584v1

22. Visani, G.; et al.: Statistical stability indices for LIME: Obtaining reliable explanations for machine learning models. In Journal of the Operational Research Society **12**(2), pp.1–11 (2021). https://doi.org/10.1080/01605682.2020.1865846

23. Outcome-oriented Predictive Process Monitoring Benchmark- github, `https://github.com/irhete/predictive-monitoring-benchmark`. Last accessed 5 March 2021

24. Pedregosa, F.; et al: Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research **12**, pp. 2825–2830, (2011).

25. 4TU Centre for Research Data. `https://data.4tu.nl/Eindhoven_University_of_Technology`. last access: 07 April 2021.

26. My paper: RQ1-RQ3

27. Alibi Explain. `https://github.com/SeldonIO/alibi`. last access: 26 April 2021.

28. Weinzierl S.; et al.: XNAP: Making LSTM-Based Next Activity Predictions Explainable by Using LRP. In Ortega A. D. R., Leopold H., Santoro F. M. (Eds.): Business Process Management Workshops (2020). International Publishing (Lecture Notes in Business Information Processing), vol. 397, pp.129–141,Springer, Cham. https://doi.org/10.1007/978-3-030-66498-5_10

29. Galanti, R.; et al. (2020): Explainable Predictive Process Monitoring. 2nd International Conference on Process Mining (ICPM), Padua, Italy, (2020), IEEE, pp. 1–8. https://doi.org/10.1109/ICPM49681.2020.00012

30. Rizzi, W.; et al.: Explainability in Predictive Process Monitoring: When Understanding Helps Improving. In: Fahland D., Ghidini C., Becker J., Dumas M. (eds.): Business Process Management Forum. BPM (2020). Lecture Notes in Business Information Processing, vol. 392, pp.141–158. Springer, Cham. https://doi.org/10.1007/978-3-030-58638-6_9

31. Verenich, I.; et al.: Predicting process performance: A white-box approach based on process models. J Softw Evol Proc **31**(6), 26 pages(2019). https://doi.org/10.1002/smr.2170

32. Sindhgatta, R; e al.: Exploring Interpretable Predictive Models for Business Processes. In: Fahland D., Ghidini C., Becker J., Dumas M. (Eds.): Business Process

Management (2020), LNCS, vol. 12168, pp. 257–272. Cham: Springer International Publishing (LNCS). https://doi.org/10.1007/978-3-030-58666-9_15

33. Jain, S.; Wallace, B. C.: Attention is not explanation. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. pp. 3543–3556. Association for Computational Linguistics (2019). https://doi.org/10.18653v1N19-1357

34. Wiegreffe, S.; Pinter, Y.: Attention is not not Explanation. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, pp. 11–20,Association for Computational Linguistics (2019). https://doi.org/10.18653v1D19-1002

A   Appendix

This appendix reports the following:

- Execution times comparison over prefix-indexed event logs with respect to different prefix lengths(Figure 15).
- Comparison of Execution times of XAI methods over single-aggregated event logs classified based on predictive models(Figure 16).
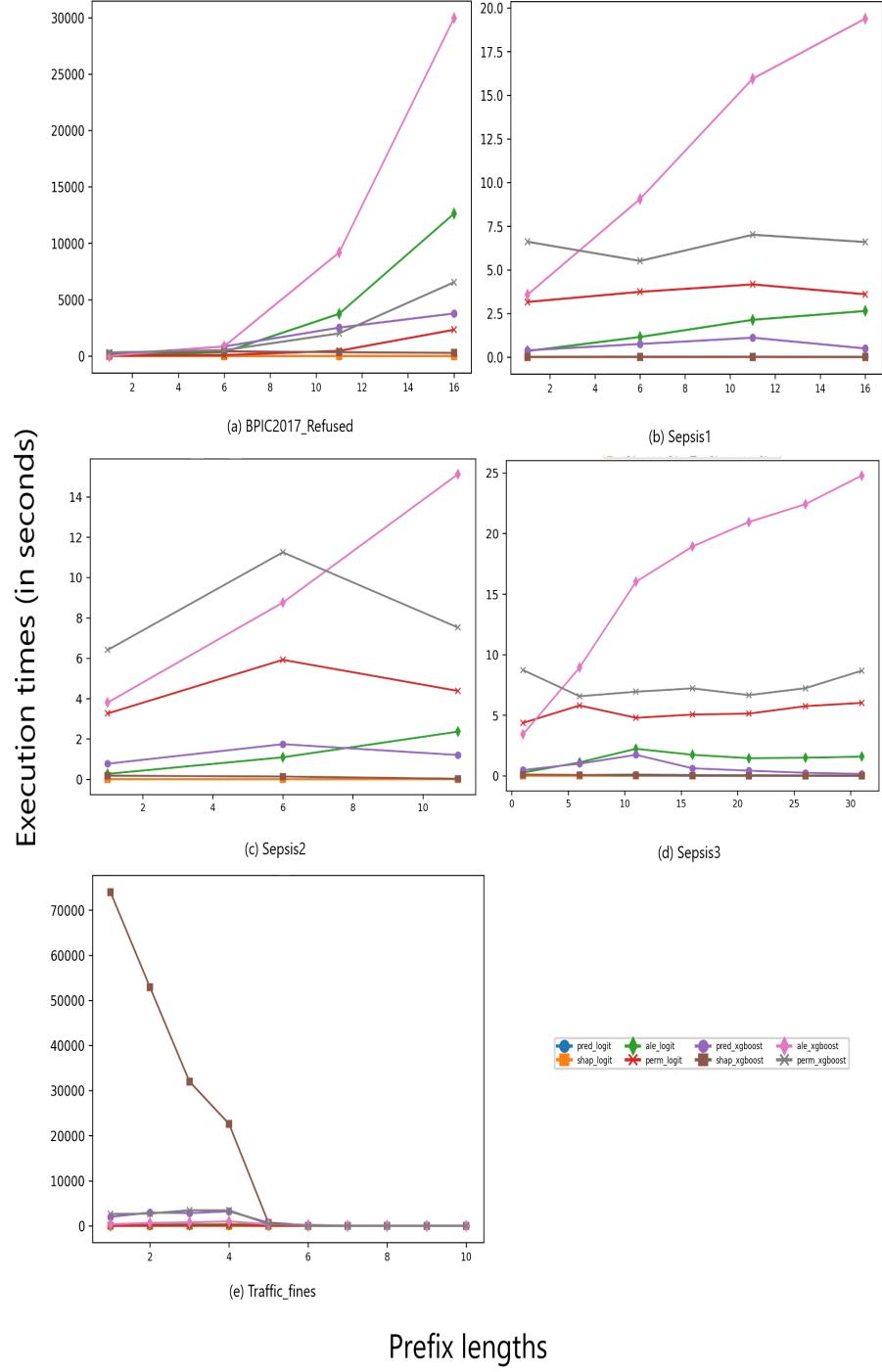
Fig. 15: Execution times (in seconds) of XAI methods on event logs preprocessed using prefix index combination.
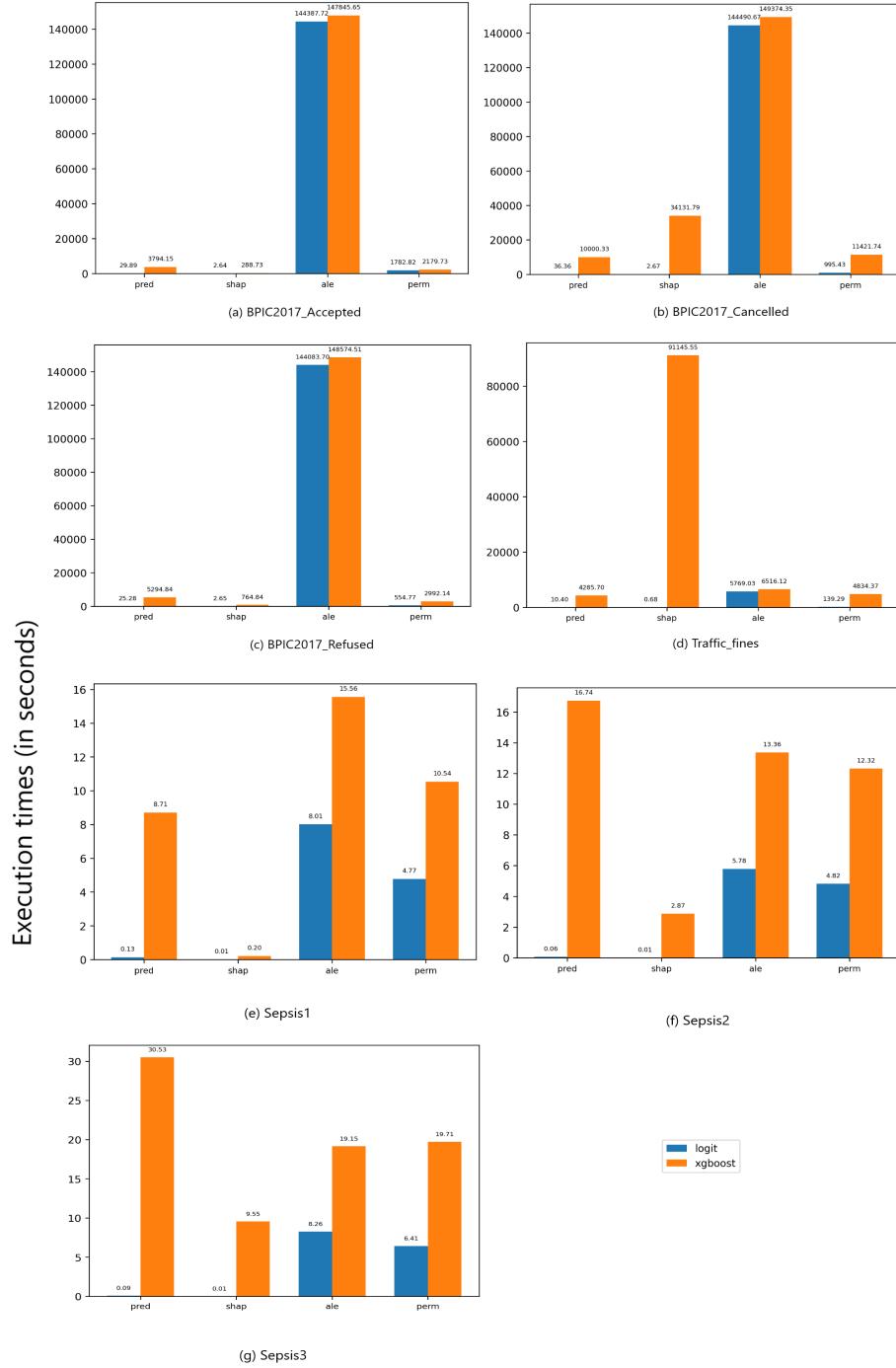
Fig. 16: Execution times (in seconds) of XAI methods on event logs preprocessed using single aggregation combination.