A Framework for Extracting and Encoding Features from Object-Centric Event Data

Jan Niklas Adams[1][0000−0001−8954−4925], Gyunam Park[1][0000−0001−9394−6513], Sergej Levich[3], Daniel Schuster[2,1][0000−0002−6512−9580], and Wil M.P. van der Aalst[1,2][0000−0002−0955−6940]

[1] Process and Data Science, RWTH Aachen University, Aachen, Germany
{niklas.adams,gnpark,schuster,wvdaalst}@pads.rwth-aachen.de
[2] Fraunhofer Institute for Applied Information Technology
[3] Information Systems Research, University of Freiburg, Freiburg, Germany
sergej.levich@is.uni-freiburg.de

Abstract. Traditional process mining techniques take event data as input where each event is associated with exactly one object. An object represents the instantiation of a process. Object-centric event data contain events associated with multiple objects expressing the interaction of multiple processes. As traditional process mining techniques assume events associated with exactly one object, these techniques cannot be applied to object-centric event data. To use traditional process mining techniques, object-centric event data are flattened by removing all object references but one. The flattening process is lossy, leading to inaccurate features extracted from flattened data. Furthermore, the graph-like structure of object-centric event data is lost when flattening. In this paper, we introduce a general framework for extracting and encoding features from object-centric event data. We calculate features natively on the object-centric event data, leading to accurate measures. Furthermore, we provide three encodings for these features: *tabular*, *sequential*, and *graph-based*. While tabular and sequential encodings have been heavily used in process mining, the graph-based encoding is a new technique preserving the structure of the object-centric event data. We provide six use cases: a visualization and a prediction use case for each of the three encodings. We use explainable AI in the prediction use cases to show the utility of both the object-centric features and the structure of the sequential and graph-based encoding for a predictive model.

Keywords: Object-Centric Process Mining · Machine Learning · Explainable AI.

1 Introduction

Process mining [1] is a branch of computer science producing data-driven insights and actions from event data generated by processes. These insights are typically grouped into three categories: process discovery, conformance checking, and enhancement. Process discovery techniques create process models describing the
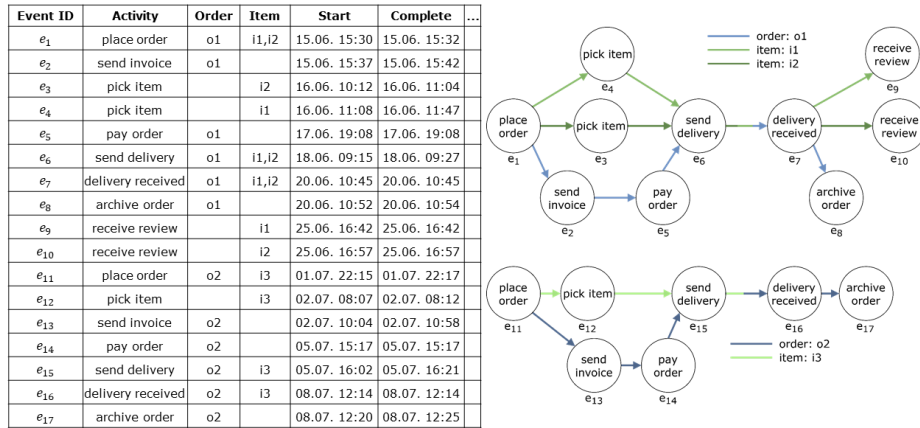
Event ID	Activity	Order	Item	Start	Complete	...
e_1	place order	o1	i1,i2	15.06. 15:30	15.06. 15:32	
e_2	send invoice	o1		15.06. 15:37	15.06. 15:42	
e_3	pick item		i2	16.06. 10:12	16.06. 11:04	
e_4	pick item		i1	16.06. 11:08	16.06. 11:47	
e_5	pay order	o1		17.06. 19:08	17.06. 19:08	
e_6	send delivery	o1	i1,i2	18.06. 09:15	18.06. 09:27	
e_7	delivery received	o1	i1,i2	20.06. 10:45	20.06. 10:45	
e_8	archive order	o1		20.06. 10:52	20.06. 10:54	
e_9	receive review		i1	25.06. 16:42	25.06. 16:42	
e_{10}	receive review		i2	25.06. 16:57	25.06. 16:57	
e_{11}	place order	o2	i3	01.07. 22:15	01.07. 22:17	
e_{12}	pick item		i3	02.07. 08:07	02.07. 08:12	
e_{13}	send invoice	o2		02.07. 10:04	02.07. 10:58	
e_{14}	pay order	o2		05.07. 15:17	05.07. 15:17	
e_{15}	send delivery	o2	i3	05.07. 16:02	05.07. 16:21	
e_{16}	delivery received	o2	i3	08.07. 12:14	08.07. 12:14	
e_{17}	archive order	o2		08.07. 12:20	08.07. 12:25	

Fig. 1: An object-centric event log and the underlying structure of events. The left-hand side depicts the event log. Events may be associated with multiple objects of different object types (here: Order and Item). The right-hand side shows the graph of directly-follows relationships for the events given by the objects. An event with multiple objects may have multiple predecessor events.

possible paths of actions in a process. Conformance checking techniques quantify and qualify the correspondence between a process model and event data. Process enhancement techniques take an encoding of features of the event data as input and deliver insights, predictions, or actions as output. Such enhancement techniques include process performance analysis [20,24], prediction [10,26,29] or clustering of similar process executions [25].

Generally, process enhancement techniques encode features of event data in either of two ways: as a table [18,9] or as a set of sequences [12,26,19]. In a tabular encoding, each row corresponds to feature values for, e.g., an event. This tabular encoding is used, for example, for regression, decision trees, and feed-forward neural networks. However, each process execution (also: case) is a timely ordered sequence of events. Therefore, summarizing event data to tabular encoding removes the sequential structure of the event data. Since this structure itself is meaningful, sequential encodings were developed [19]. These encodings represent each process execution as a sequence of feature values and are used for predictive models considering sequentially encoded data, such as LSTMs [26], or to visualize the variant of the process execution.

Traditional process mining builds on two central assumptions: Each event is associated with exactly one *object* (the case) and each object is of the same type. Each object is associated with a sequence of events. A traditional *event log*, therefore, describes a collection of homogeneously typed, isolated event sequences. This is a valid assumption when analyzing, e.g., the handling of insurance claims. In this example, each object describes an instantiation of the same type: an insurance claim. Events are associated to exactly one insurance claim. However, real-life information systems often paint another picture: Events may
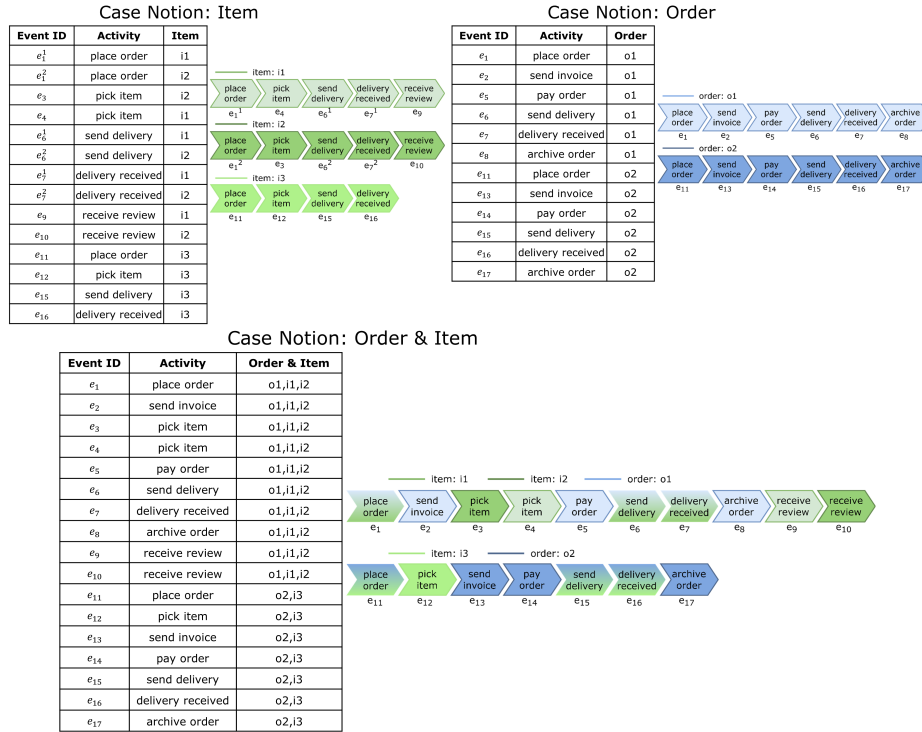
Case Notion: Item

Event ID	Activity	Item
e_1^1	place order	i1
e_1^2	place order	i2
e_3	pick item	i2
e_4	pick item	i1
e_6^1	send delivery	i1
e_6^2	send delivery	i2
e_7^1	delivery received	i1
e_7^2	delivery received	i2
e_9	receive review	i1
e_{10}	receive review	i2
e_{11}	place order	i3
e_{12}	pick item	i3
e_{15}	send delivery	i3
e_{16}	delivery received	i3

Case Notion: Order

Event ID	Activity	Order
e_1	place order	o1
e_2	send invoice	o1
e_5	pay order	o1
e_6	send delivery	o1
e_7	delivery received	o1
e_8	archive order	o1
e_{11}	place order	o2
e_{13}	send invoice	o2
e_{14}	pay order	o2
e_{15}	send delivery	o2
e_{16}	delivery received	o2
e_{17}	archive order	o2

Case Notion: Order & Item

Event ID	Activity	Order & Item
e_1	place order	o1,i1,i2
e_2	send invoice	o1,i1,i2
e_3	pick item	o1,i1,i2
e_4	pick item	o1,i1,i2
e_5	pay order	o1,i1,i2
e_6	send delivery	o1,i1,i2
e_7	delivery received	o1,i1,i2
e_8	archive order	o1,i1,i2
e_9	receive review	o1,i1,i2
e_{10}	receive review	o1,i1,i2
e_{11}	place order	o2,i3
e_{12}	pick item	o2,i3
e_{13}	send invoice	o2,i3
e_{14}	pay order	o2,i3
e_{15}	send delivery	o2,i3
e_{16}	delivery received	o2,i3
e_{17}	archive order	o2,i3

Fig. 2: Flattening an object-centric event log (cf. Fig. 1) such that it can be used for traditional process enhancement techniques. The event log is transformed into sequences of a chosen case notion. Due to deficiency, convergence and divergence, the features calculated on a flattened log might be misleading, e.g., through missing events. Furthermore, the graph-like structure of the original event log is lost.

be related to multiple objects of different types [3,4,11,28]. The most prominent example of information systems generating event data with multiple associated objects are ERP systems. Objects in such systems would correspond to, e.g., an order, different items of this order, and invoices in an order-to-cash process. Consider the simplified example of an order handling process depicted in Fig. 1. An event may be related to objects of type order, item, or both. An event with multiple objects may have multiple predecessor events. Therefore, the structure of an *Object-Centric Event Log (OCEL)* resembles a graph, not a sequential structure as is assumed in traditional process mining.

This gap between OCELs and traditional process enhancement techniques is currently bridged by *flattening* an event log [2], i.e., mapping an OCEL into traditional event log format by enforcing a homogeneous, sequential structure. This involves two steps: Choosing a *case notion* and duplicating events with multiple objects of that notion. All objects not included in this case notion are discarded. Flattening the event log of Fig. 1 is depicted in Fig. 2 for three different case notions. The first two are case notions of a single object type [2].
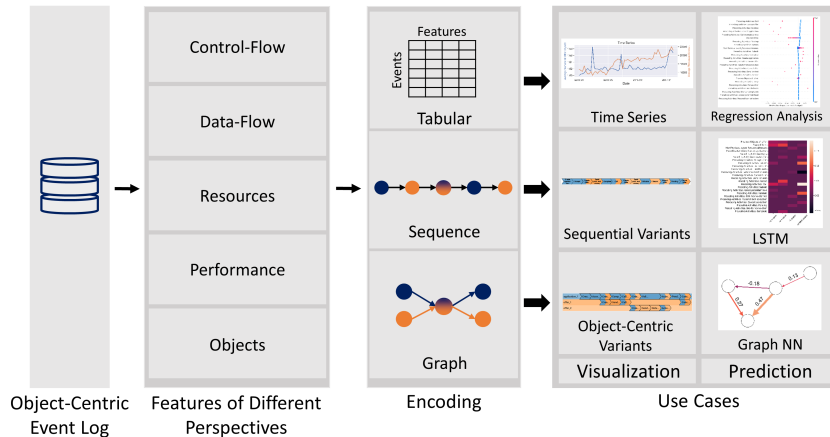
Fig. 3: Our framework enables accurate feature extraction for object-centric event data. Furthermore, we provide three encodings for object-centric features: tabular, sequential, and graph-based. We present a visualization and prediction use case for each encoding.

The third case notion is a composite case notion of co-appearing orders and items, i.e., an order and all corresponding items. The flattened event data may be used as input for traditional process enhancement techniques.

However, flattening manipulates the information of the object-centric event log. The problems related to flattening are *deficiency* (disappearing events) [3], *convergence* (duplicated events) [2] and *divergence* (misleading directly-follows relations) [28,2]. We showcase divergence using an example. One might use a composite case notion of order and item to flatten the event log (cf. Fig. 2 Case Notion: Order & Item). All orders and items related through events form one composite object, i.e., o1,i1,i2, and o2,i3. The events of these objects are flattened to one sequence, introducing inaccurate precedence constraints. E.g., events e_3 and e_4, which describe an item being picked, are now sequentially ordered, indicating some order between them. However, the original event data show that these two picking events are independent. The same holds for the relationship of pick item and pay order: The object-centric event data do not indicate any precedence constraint. However, the sequential representation enforces one.

The three problems of flattening have major consequences on the quality of the calculated features of the flattened OCEL: Due to missing events, duplicated events, or wrong precedence constraints, many features deliver incorrect results (cf. Sec. 4). Furthermore, the tabular or sequential encoding constructed from these features does not preserve the graph-like structure of the event data, removing important structural information. Therefore, features for OCELs can not accurately be extracted and encoded.

To solve the previously mentioned problem, an approach is necessary that calculates features natively on the object-centric event data and enables a graph-based encoding preserving the actual structure of the event log. In this paper, we

Table 1: Process enhancement techniques and supporting frameworks.

		Feature Extraction		Feature Encoding			Existing
		Object-centric	Flattened	Tabular	Sequential	Graph-based	work
(a) Process enhancement techniques	P1		✓	✓			[9,13]
	P2		✓		✓		[19,12,26]
	P3		✓			✓	[23,27,16]
(b) Frameworks	F1		✓	✓			[18]
	F2		✓		✓		[7,14]
	This paper	✓		✓	✓	✓	

introduce a general framework for extracting and encoding features for object-centric event data (cf. Fig. 3), providing two contributions: **1)** We translate the computation of the features introduced in the framework of de Leoni et. al [18] to the object-centric setting, providing accurate measures. **2)** We provide three different encodings to represent the extracted features for different algorithms and methods: tabular, sequential, and graph-based. Using features and encoding, we provide six use cases. These use cases showcase the generalizability of our framework to a plethora of different tasks. We use one visualization and one prediction use case for each encoding. In the prediction use cases, we depict how the different features and the structure of the encodings are utilized by predictive models, leveraging on explainable AI and SHAP values [21]. These contributions may be used as a foundation for new algorithms, new visualizations, new machine learning models, more accurate predictions, and more.

This paper is structured as follows. First, we discuss related work on feature extraction and encoding in Sec. 2. We introduce object-centric event data and process executions in Sec. 3. In Sec. 4, we provide an overview of native feature calculation for object-centric event data. In Sec. 5, we define three encodings for object-centric features. Sec. 6 depicts our six use cases for features and their encodings. We conclude this paper in Sec. 7.

2   Related Work

A plethora of process enhancement techniques exist in the literature, including process performance analysis, predictive process monitoring, and trace clustering [1]. Such techniques use encoded features extracted from an event log as input. Table 1(a) shows three categories of techniques using different feature extraction (i.e., feature extractions using 1. OCELs and 2. flattened event logs) and encoding (i.e., 1. tabular, 2. sequential, and 3. graph encoding) approaches with representative examples. First, *P1* represents the techniques using features extracted from flattened event logs and encoded as tabular formats. For instance, van Dongen et al. [9] use tabular encoding by transforming an event log into feature-outcome pairs to predict remaining times using non-parametric regression. Also, in [13], an event log is encoded into a tabular format with additional features on context, e.g., resource availability, to predict processing times. Second, techniques in *P2* also use features based on flattened event logs but encoded as sequential formats. Leontjeva et al. [19] propose *complex se-*

quence encoding to encode an event log to sequences to predict the outcome of an ongoing case. To predict the next activity of an ongoing case, Evermann et al. [12] encode control-flow features using *embedding* techniques, whereas Tax et al. [26] use *one-hot encoding*. Finally, *P3* consists of techniques using features extracted from flattened event logs and encoded as graph formats. Philipp et al. [23] encode an event log to a graph where each node represents an activity, and each edge indicates the relationship between activities. The graph is used to learn a *Graph Neural Network (GNN)* to predict process outcomes. Venugopal et al. [27] extend [23] by annotating nodes with temporal features. They use GNNs to predict the next activity and next timestamp of an event. Instead of representing a node as an activity, Harl et al. [16] uses one-hot encoding of an activity to represent a node to deploy *gated graph neural network* that provides the explainability based on *relevance score*.

Furthermore, to support the development of process enhancement techniques using different feature extraction and encoding, several frameworks have been proposed (cf. Table 1(b)). First, De Leoni [18] in *F1* suggest a framework to compute features using flattened event logs and encode them to tables. Second, Becker et al. [7] and Di Francescomarino et al. [14] in *F2* propose frameworks for techniques for sequentially encoding extracted features. To the best of our knowledge, no framework supporting graph encoding exists.

Despite the limitations of flattened event logs to extract misleading features, no study has been conducted to develop process enhancement techniques using features based on OCELs. In this work, we provide a framework for extracting and encoding features based on OCELs, with the goal of facilitating the development of object-centric process enhancement approaches. Our proposed framework supports all existing encoding formats, i.e., tabular, sequential, and graph, to be used for different algorithms and methods.

3   Object-Centric Event Data

Given a set X, the powerset $\mathcal{P}(X)$ denotes the set of all possible subsets. A sequence $\sigma : \{1, \ldots, n\} \to X$ of length $len(\sigma) = n$ assigns order to elements of X. We denote a sequence with $\sigma = \langle x_1, \ldots, x_n \rangle$ and the set of all sequences over X with X^*. We overload the notion $x \in \sigma$ to express $x \in range(\sigma)$.

A graph is a tuple $G = (V, E)$ of nodes V and edges $E \subseteq V \times V$. The set of all subgraphs of G is given by $sub(G) = \{(V', (V' \times V') \cap E) \mid V' \subseteq V\}$. A path connects two distinct nodes through edges. The set of paths between two nodes $v, v' \in V, v \neq v'$ is defined by $path_G(v, v') = \{\langle (v, v_1), (v_1, v_2), \ldots, (v_k, v') \rangle \in E^*\}$. Two distinct nodes are connected if the set of paths between them is not empty $path_G(v, v') \neq \emptyset$. The distance between two nodes is the length of the shortest path $dist_G(v, v') = len(\sigma_d)$ such that $\sigma_d \in path_G(v, v') \wedge \neg \exists_{\sigma'_d \in path_G(v, v')} len(\sigma_d) > len(\sigma'_d)$. A graph $G = (V, E)$ is connected iff a path exists between all edges $\forall_{v, v' \in V} \ v \neq v' \wedge path_G(v, v') \neq \emptyset$. The set of connected subgraphs of $G = (V, E)$ is defined as follows $consub(G) = \{G' \in sub(G) \mid G' \text{ is connected}\}$.

An event log is a collection of events associated with objects. Each event contains an activity, describing the executed action, a start and complete timestamp and additional attributes. Each object is associated to a sequence of events.

Definition 1 (Event Log). *Let $\mathcal{E}$ be the universe of events, $\mathcal{O}$ be the universe of objects, $\mathcal{OT}$ be the universe of object types, $\mathcal{A}$ be the universe of activities, $\mathcal{C}$ be the universe of attributes and $\mathcal{V}$ be the universe of attribute values. Let $A \subseteq \mathcal{A}$ be a set of activities and $C \subseteq \mathcal{C}$ be a set of attributes. Each object is mapped to exactly one object type $\pi_{type} : \mathcal{O} \rightarrow \mathcal{OT}$. An event log $L = (E, O, OT, \pi_{ct}, \pi_{st}, \pi_{trace}, \pi_{act}, \pi_{att})$ is a tuple composed of*

- *events $E \subseteq \mathcal{E}$, objects $O \subseteq \mathcal{O}$, and object types $OT \subseteq \mathcal{OT}$,*
- *two time mappings for the completion $\pi_{ct} : E \rightarrow \mathbb{R}$ and the start $\pi_{st} : E \rightarrow \mathbb{R}$ of an event such that $\pi_{st}(e) \leq \pi_{ct}(e)$ for any $e \in E$,*
- *a mapping $\pi_{trace} : O \rightarrow E^*$ mapping each object to a sequence of events such that $\forall_{o \in O} \ \pi_{trace}(o) = \langle e_1, \ldots, e_n \rangle \wedge \forall_{i \in \{1, \ldots n-1\}} \ \pi_{ct}(e_i) \leq \pi_{ct}(e_{i+1})$,*
- *an activity mapping $\pi_{act} : E \rightarrow A$ and,*
- *an attribute mapping $\pi_{att} : E \times C \nrightarrow \mathcal{V}$.*

The table in Fig. 1 depicts an example of an OCEL. A row corresponds to one event. Sorting the events of an object in timely order, we retrieve the event sequence for the object, e.g., $\pi_{trace}(\text{i3}) = \langle$place order, pick item, send delivery, delivery received$\rangle$. The relationships between objects can be expressed in the form of a graph, connecting objects that share events.

Definition 2 (Object Graph). *Let $L = (E, O, OT, \pi_{ct}, \pi_{st}, \pi_{trace}, \pi_{act}, \pi_{att})$ be an event log. We denote the objects of an event $e \in E$ with $\pi_{obj}(e) = \{o \in O \mid e \in \pi_{trace}(o)\}$. The object graph $OG_L = (O, I)$ is an undirected graph of nodes O and edges of object interactions $I = \{\{o, o'\} \subseteq O \mid o \neq o' \wedge \exists_{e \in E} \ \{o, o'\} \subseteq \pi_{obj}(e)\}$.*

Objects which are directly or transitively connected in the object graph depend on each other by sharing events. In traditional process mining, a process execution (case) is the event sequence of one object. We use the definitions of process executions [6] and generalize this notion such that a process execution is the set of events for multiple, connected objects.

Definition 3 (Process Execution). *Let $L = (E, O, OT, \pi_{ct}, \pi_{st}, \pi_{trace}, \pi_{act}, \pi_{att})$ be an event log and $OG_L = (O, I)$ be the corresponding object graph. A process execution $p = (O', E')$ is a tuple of objects $O' \subseteq O$ and events $E' \subseteq E$ such that $e' \in E' \Leftrightarrow \pi_{obj}(e') \subseteq O'$ and O' forms a connected subgraph in OG_L.*

We define two techniques to extract process executions from an OCEL. These two techniques are two out of many possible process execution extraction techniques. The first technique extracts process executions based on the connected components of the object graph. All transitively connected objects form one process execution. This might lead to large executions for entangled event logs. Therefore, we introduce the leading type extraction. A process execution is constructed for each object of a chosen leading object type. Connected objects are added to this process execution unless a connected object of the same type has

a lower distance to the leading object. This limits executions in size but also removes dependencies.

Definition 4 (Execution Extraction). *Let $L=(E, O, OT, \pi_{ct}, \pi_{st}, \pi_{trace}, \pi_{act}, \pi_{att})$ be an event log. An execution extraction $EX \subseteq consub(OG_L)$ retrieves connected subgraphs from the object graph. A subgraph $ex = (O', I') \in EX$ is mapped to a process execution $f^{extract}(ex, L) = (O', E')$ with $E' = \{e \in E \mid O' \cap obj(e) \neq \emptyset\}$. We define two extraction techniques:*

- *$EX_{comp}(L) = \{G \in consub(OG_L) \mid \neg\exists_{G' \in consub(OG_L)} \ G \in sub(G')\}$, and*
- *$EX_{lead}(L, ot) = \{G \in lead\_graphs = \{G' = (O', I') \in consub(OG_L) \mid \exists_{o \in O'} \ \pi_{type}(o) = ot \land \forall_{o' \in O'} \ \neg\exists_{o'' \in O'} \ o'' \neq o' \land \pi_{type}(o'') = \pi_{type}(o') \land dist_{G'}(o, o') > dist_{G'}(o, o'')\} \mid \neg\exists_{G'' \in lead\_graphs} G \in sub(G'')\}$ for $ot \in OT$.*

When looking at the example of Fig. 1, the process executions retrieved by applying EX_{comp} would be based on the connected components of the object graph, i.e., $\{o1, i1, i2\}$ and $\{o2, i3\}$. Using the leading type order, we would retrieve the same executions. Using item as the leading type, we would retrieve $\{i1, o1\}$, $\{i2, o1\}$ and $\{i3, o2\}$.

4   Object-Centric Features

This section deals with the problem resulting from flattening OCELs to apply process enhancement techniques: Features are calculated on the manipulated, flattened event data. Therefore, they might be inaccurate. We propose an object-centric adaptation of the features introduced by the seminal machine learning framework of de Leoni et al. [18]. We calculate them natively on the OCEL. Furthermore, we provide several new features recently introduced in the literature on object-centric process mining. A feature is, generally, calculated for an event. It might describe a measure for the single event, in relationship to its process executions, or the whole system.

Definition 5 (Features). *Let $L = (E, O, OT, \pi_{ct}, \pi_{st}, \pi_{trace}, \pi_{act}, \pi_{att})$ be an event log and $EX \subseteq consub(OG_L)$ be a set of extracted process executions. A feature $f_L{:}E \times EX \nrightarrow \mathbb{R}$ maps an event and a process execution onto a real number.*

The primary need for adapting traditional feature calculation arises from two main differentiations between object-centric and traditional event data: First, each event can have multiple preddecesors/successors, one for each object. Second, each event might have multiple objects of different types. The computation of features that are depended on previous and following behavior has to be adapted to the graph structure. The most obvious example are preceding activities: In traditional feature extraction, there is only one preceding activity for each event. In object-centric feature extraction, there are multiple preceding activities, one for each object. The graph-structure as well as the multiplicity of objects also enables the definition of new features leveraging on the graph structure and object (type) associations. Previous (i.e., all events that happened before the considered event in an execution) and following events can be adapted
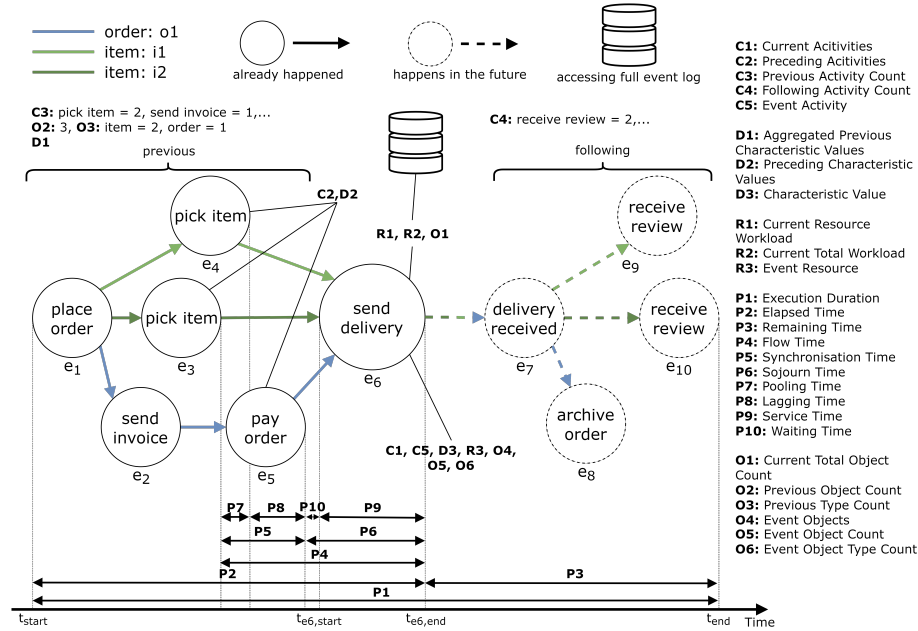
Fig. 4: Overview of the features that can be extracted for event e_6. These features are the object-centric adaptations of [18].

in two ways: time-based (using the event's timestamp) and path-based (using path information of the graph). We use a simple time-based adaptation. However, the graph-based adaptation might give interesting new research directions.

An overview of the features collected from an object-centric adaptation of de Leoni et al.'s framework [18] and features recently introduced in the literature [3,22] is depicted in Fig. 4. Similar to de Leoni et al., we group features according to different perspectives: **C**ontrol-Flow, **D**ata-Flow, **R**esource, **P**erformance and **O**bjects. We, now, discuss the different perspectives and the adaptations that are necessary to apply them to the object-centric setting. Table 2 provides a qualitative evaluation of the impact of flattening on the resulting feature value: Features can be equal, they can be misleading/incorrect after flattening, and not be available for flat event data.

The main adaptations of the control-flow perspectives are concerned with the switch from sequential to graph-like control-flow. Multiple preceding activities (**C2**) as well as multiple current activities (**C1**) (endpoints of the current execution graph) are possible. For previous and following activities (**C3**, **C4**), we use a simple time-based adaptation.

The data-flow perspective needs slight adaptations for preceding characteristic values (**D2**). Since there might be multiple preceding values, these need to be aggregated. Previous characteristic values (**D1**) are adapted on a time basis, and the characteristic value (**D3**) needs no adaptation.

Table 2: Impact of flattening on calculated feature values. Calculating a feature for an event on object-centric vs. flattened data can lead to correct or misleading results. Some features only exist on object-centric event data. Most features are misleading due to the graph structure and object multiplicity.

Features		Impact of flattening		Only available for OCEL
		Correct	Misleading	
Control-Flow	**C1**, **C2**, **C3**, **C4**		✓	
	C5	✓		
Data	**D1**, **D2**		✓	
	D3	✓		
Resource	**R1**		✓	
	R2, **R3**	✓		
Performance	**P1**, **P2**, **P3**, **P6**, **P10**		✓	
	P4, **P5**, **P7**, **P8**			✓
	P9	✓		
Objects	**O1**, **O2**, **O3**, **O4**, **O5**			✓

The resources perspective's features are mainly concerned with system-wide measurements, such as the workload of the current resource (**C1**) or the total system workload (**C2**). Therefore, this perspective remains mostly unaffected by a move to object-centricity. Future research might investigate new features derived from resource multiplicity per event.

The performance perspective has recently been studied for new object-centric features [22]. Due to an event having multiple predecessors, the established performance measures can be extended by several features expressing the time for synchronization between objects (**P5**), the pooling time of an object type (**P7**), or the lag between object types before the event (**P8**).

Finally, a new feature perspective concerning objects opens up. The paper introducing the discovery of object-centric Petri nets [3] introduces some basic features of the object perspective. For example, an event's number of objects (**O5**), the event's number of objects of a specific type (**O6**), or the current system's total object count (**O1**). Investigations of additional features in this perspective, e.g., quantifying the relationships between objects through graph metrics on the object graph, might also be an interesting research direction.

5   Feature Encodings

In this section, we tackle the absence of feature encodings that represent the graph-like structure of object-centric event data. We extend the currently used tabular and sequential encodings with a graph-based one and introduce all three encodings formally. Together with the formal definition of each encoding, we provide some common use cases, advantages, disadvantages and a continuation of our running example from Fig. 1. As an example of extracted features we choose the number of previous objects (**O2**), the synchronization time (**P5**) and the remaining time (**P3**). The execution extraction for our example is the connected
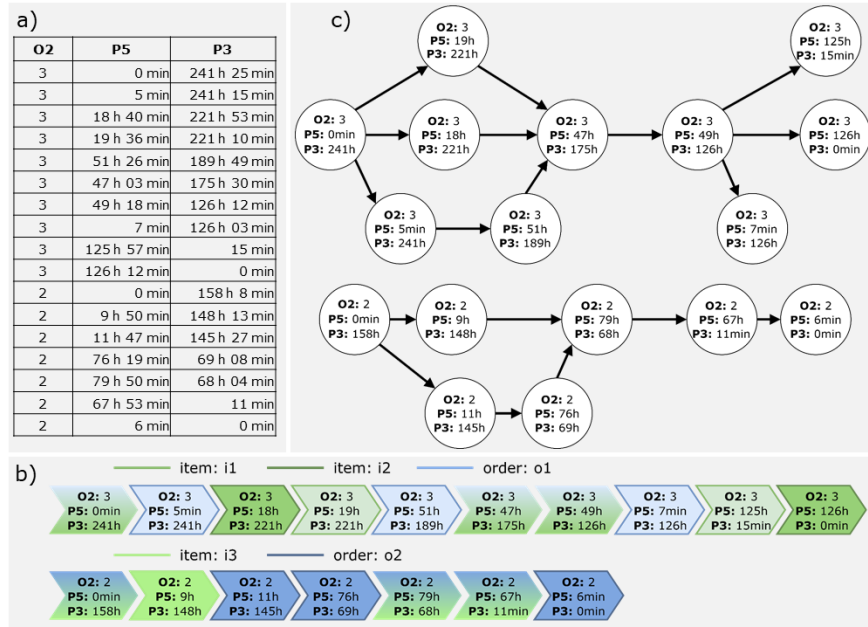
Fig. 5: Example of tabular a), sequential b) and graph-based c) feature encodings for the running example in Fig. 1. The graph-based encoding preserves the structural information from the OCEL.

components extraction EX_{comp}. A tabular encoding is a common representation of data points used for many use cases, such as regression analysis, clustering, different data mining tasks, etc.

Definition 6 (Tabular Encoding). *Let $L = (E, O, OT, \pi_{ct}, \pi_{st}, \pi_{trace}, \pi_{act}, \pi_{att})$ be an event log and $EX \subseteq consub(OG_L)$ be a set of process executions. Let $F_L \subseteq E \times EX \nrightarrow \mathbb{R}$ be a set of features. The event feature table is defined by $tab(e, f_L) = f_L(e, ex)$ for all $e \in E$, $ex \in EX$ (rows) and all $f_L \in F_L$ (columns).*

We depict an example of tabular encoding in Fig. 5 a). Such an encoding is easily readable and versatile usable, however, the structural order information of the event log is lost in the process of tabular encoding. A sequential encoding is commonly used in sequence visualization, clustering, classification or next value predictions (cf. Sec. 2).

Definition 7 (Sequential Encoding). *Let $L = (E, O, OT, \pi_{ct}, \pi_{st}, \pi_{trace}, \pi_{act}, \pi_{att})$ be an event log and $EX \subseteq consub(OG_L)$ be a set of extracted process executions. Let $F_L = \{f_{L,1}, \ldots, f_{L,m}\} \subseteq E \times EX \nrightarrow \mathbb{R}$ be a set of features. The sequential encoding of an execution $ex \in EX$ is defined by $seq(ex, F_L) = \langle (f_{L,1}(e_1, ex), \ldots, f_{L,m}(e_1, ex)), \ldots, (f_{L,1}(e_n, ex), \ldots, f_{L,m}(e_n, ex)) \rangle$ with with $(O', \{e_1, \ldots, e_n\}) = f^{extract}(ex, L)$ and $\pi_{ct}(e_1) \leq \cdots \leq \pi_{ct}(e_n)$.*

We depict a sequential encoding of the running example in Fig. 5 b). The events for process executions are ordered according to the complete timestamp of

Table 3: Results for the different models based on different encodings.

	Regression	LSTM	GNN
Baseline MAE	0.7598		
Train MAE	0.5101	0.4717	0.4460
Validation MAE	NA	0.4625	0.4534
Test MAE	0.5087	0.4568	0.4497

the event. The resulting sequence is attributed with the different feature values for each event. This encoding respects the timely order of events. *However, it does not respect the true precedence constraints of the event log*: By merging all events into one sequence, some event pairs are forced into a precedence relationships they did not exhibit in the event log (cf. Sec. 1). A graph encoding of features may be used for extensive visualization, applying graph algorithms or for utilizing graph neural networks [30].

Definition 8 (Graph Encoding). *Let $L=(E, O, OT, \pi_{ct}, \pi_{st}, \pi_{trace}, \pi_{act}, \pi_{att})$ be an event log $EX \subseteq consub(OG_L)$ be a set of extracted process executions. Let $F_L = \{f_{L,1}, \ldots, f_{L,m}\} \subseteq E \times EX \nrightarrow \mathbb{R}$ be a set of features. For an extracted execution $ex \in EX$, the graph of the corresponding process execution $p = (O', E') = f^{extract}(ex, L)$ is defined by $G_p = (E', K)$ with edges $K = \{(e, e') \in E' \times E' \mid e \neq e' \wedge o \in O' \wedge \langle e_1, \ldots, e_n \rangle \in \pi_{trace}(o) \wedge e = e_i \wedge e' = e_{i+1} \wedge i \in \{1, \ldots, n-1\}\})$. The graph encoding is defined by $G_{feat}(p, F_L) = (E', K, l)$ with a node labeling function $l(e) = \{f_L(e, ex) \mid f_L \in F_L\}$ for any $e \in E'$.*

An example of the graph-based feature encoding for our running example is depicted in Fig. 5 c). Each process execution is associated with a graph. Each node of the graph represents the feature values of an event.

6   Use Cases

In this section, we evaluate our framework by providing six use cases. We pursue two evaluation goals with this approach: First, we aim to showcase the generalizability of the framework by providing a collection of common process mining tasks the framework can be applied to. Second, we aim to showcase the feature's and encoding's effectiveness in the use cases. Over the last years, explainable AI has been increasingly employed to make predictive process monitoring transparent [15,17]. Through the use of SHAP [21] values, we are able to quantify feature importance as well as structural importance of sequential and graph-based encoding.

The use cases are split into two parts: three visualization and three prediction use cases. We use a real-life loan application event log [8] as an OCEL. An event can be related to an application and multiple loan offers as objects. We use tabular, sequential, and graph-based encoding to gain insights into the process through the visualization use cases. The prediction use cases aim at predicting the remaining time of an event's process execution (**P3**) using three different techniques for the different encodings: regression (tabular), LSTM neural networks (sequential), and GNNs (graph-based). We use the same features for each
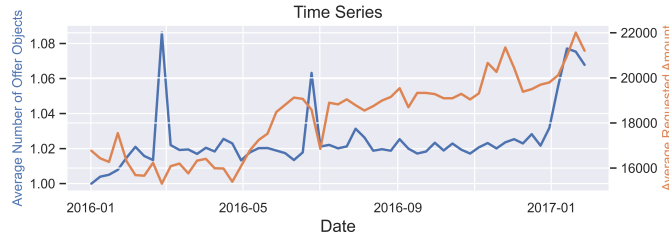
Fig. 6: Time series describing two features over time: the weekly average number of loan offers per event and the weekly average requested amount for each application. Using this evaluation some initial insights can be generated, e.g., the gradual increase in requested amount over time.

encoding: Preceding activities (**C2**), average previous requested amount (**D1**), the elapsed time (**P2**), and the previous number of offers (**O3**). We use a 0.7/0.3 train/test split of the same events for each model for comparability reasons. We set aside 20% of the training set as a validation set. The performance is assessed using the *Mean Absolute Error (MAE)* of the normalized target variable. Furthermore, we provide a baseline MAE achieved by predicting the training set's average remaining time. The summarized results are depicted in Table 3.

We provide an open-source python implementation of our framework[1]. Our experiments can ge reproduced through a GitHub repository[2]. The framework can be extended with new features and adapted algorithms.

6.1   Tabular Encoding

Visualization We split the event log into subsequent sublogs containing the events of one week each. For each sublog, we extract the average requested amount (**D3**) and the number of offers per event (**O6**). The resulting time series is depicted in Fig. 6. We can observe the dynamics of the process over time, e.g., the increase in the requested amount over time. Furthermore, we can observe that the number of offers is stable, except for a few short spikes.

Prediction We use a linear regression model to predict the remaining time based on the tabular encoding (cf. Table 3). This is an object-centric adaption of use cases [9,13]. We generate the SHAP values, i.e., the impact of different features on the individual model prediction, for 1000 predictions of the test set. The resulting bee swarm plot is depicted in Fig. 7 (left side). Red points indicate a high feature value. The more they are positioned to the left, the more the feature value reduces the model's prediction. Therefore, the combination of color and position gives insights into the feature value's impact on the model output. We can, e.g., observe a high decreasing impact of the existence of the *Call* activity in the preceding activities to the predicted remaining time. One can also observe an impact of the new object-centric feature of the number of previous objects

---

[1] https://github.com/ocpm/ocpa
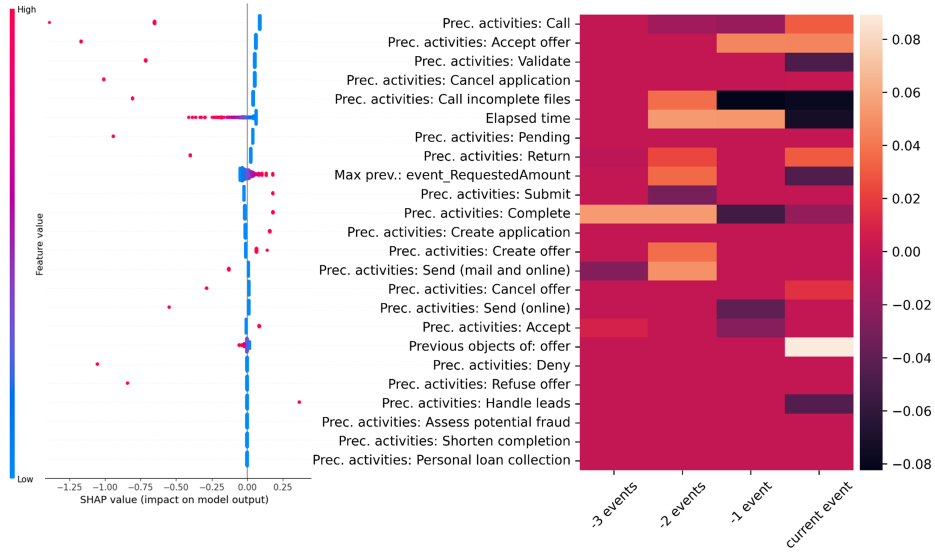[2] https://github.com/niklasadams/OCELFeatureExtractionExperiments

Fig. 7: Left: Bee swarm plot of SHAP values for the regression model, showing the aggregated importance of different features to the predictions. Right: SHAP values of one LSTM prediction, visualized for the different positions of the input sequence.

Fig. 8: Sequential variant visualization of a process execution enriched with the object information (blue = application, orange = first and second offer).

of type offer on the predicted remaining time: the more offers were previously recorded in a case, the lower the predicted remaining time. In conclusion, the selected set of object-centric feature adaptations yields valuable information for a predictive model.

6.2 Sequential Encoding

Visualization We choose one specific process execution and extract the sequential encoding for the current event's activity (**C5**) and the event's objects (**O4**) features. The result is a variant enriched by object information, depicted in Fig. 8. Even though such a visualization might have misleading causality information for events between objects, one can already retrieve some valuable insight into the intra-object order and the overall activities of an execution.

Prediction We use a neural network with two 10-hidden-node LSTM layers to predict the remaining time of the sequentially encoded features. We use subsequences of length four (cf. Table 3 for results). This is an object-centric adaption of use cases [26,19]. The regression use case already covered the importance of features for the prediction. Therefore, we focus on the importance of the sequential encoding in this use case. We use SHAP values for each feature of the
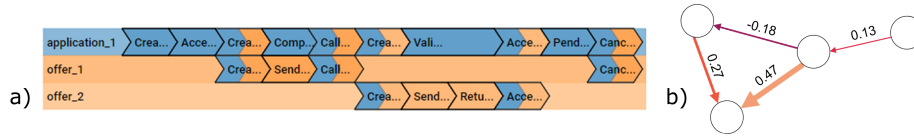
Fig. 9: Use cases for the graph encoding: a) activities and objects of one process execution. Shared events between objects are colored with multiple colors. b) shows the importance of different edges of one instance graph when predicting.

four positions in the sequential encoding. The calculated feature impacts for an individual prediction are depicted in Fig. 7 (right side). The more the value diverges from zero, the higher the feature's impact on the model's output. We observe features with high importance among all four positions of the sequence. Therefore, the model utilizes the sequential encoding of the features, showcasing its usefulness.

6.3 Graph Encoding

Visualization Fig. 9 a) depicts the graph-based variant visualization retrieved from OCπ [5] of the same process execution as Fig. 8. Using the graph, one can place concurrent events in two different lanes according to their objects, not indicating any precedence between them. One can intuitively determine the concurrent paths in the variant and the interaction of different objects. For large process executions, this provides structured access to the control-flow of the underlying process.

Prediction We use the graph-based feature encoding as an input for a GNN. The GNN contains two graph convolution layers. Each node in both layers has a size of 24. Input graphs are constrained to four nodes (cf. LSTM use case). We read the graphs out by averaging over the convoluted values, summarizing to one predicted remaining time (cf. Table 3 for results). This is an object-centric adaptation of use cases [23,27]. We adapt SHAP values to determine the importance of graph edges to the predicted remaining time. Fig. 9 b) depicts the calculated values for one graph instance. The more the value of an edge diverges from zero, the higher its existence impacts the model's prediction. We observe substantially different values for all edges: While some edges have a relatively low negative or positive impact on the model's output, the presence of other edges heavily impacts the predicted remaining time. Therefore, the graph structure itself yields important information for predicting the remaining time.

7 Conclusion

We introduced a general framework to extract and encode features from OCELs. Currently, object-centric event data needs to be flattened to apply process enhancement techniques to the data. This leads to inaccurate features. Additionally, no feature encoding is available to express the graph-like structure of object-centric event data. Our framework calculates features natively on the object-centric event data, leading to accurate features. Furthermore, we provide

a graph-based encoding of the features, preserving the underlying structure of an OCEL. We show the utility of the features and encodings in six use cases, a visualization and prediction use case for each of the three encodings. This framework lays a foundation for future machine learning approaches utilizing object-centric event data and new algorithms using our encodings as a basis.

We provide a collection of use cases showing the applicability of our framework for extracting and encoding features. For each of our framework steps, interesting future research directions are present: Which feature work well with which encoding? What are the best prediction techniques for which encoding? How to optimize existing network architectures to achieve maximum results? Furthermore, investigations of new features derived from the graph structure and object-multiplicity as well as further traditional features not included in de Leoni et al.'s framework [18] is an interesting direction for future research.

References

1. van der Aalst, W.M.P.: Process mining: Data science in action. Springer (2016)
2. van der Aalst, W.M.P.: Object-centric process mining: Dealing with divergence and convergence in event data. In: SEFM. pp. 3–25. Springer (2019). https://doi.org/10.1007/978-3-030-30446-1_1
3. van der Aalst, W.M.P., Berti, A.: Discovering object-centric Petri nets. Fundam. Informaticae **175**(1-4), 1–40 (2020). https://doi.org/10.3233/FI-2020-1946
4. Adams, J.N., van der Aalst, W.M.P.: Precision and fitness in object-centric process mining. In: ICPM. pp. 128–135. IEEE (2021). https://doi.org/10.1109/ICPM53251.2021.9576886
5. Adams, J.N., van der Aalst, W.M.P.: Ocπ: Object-centric process insights. In: PETRI NETS. vol. 13288, pp. 139–150. Springer (2022). https://doi.org/10.1007/978-3-031-06653-5_8
6. Adams, J.N., Schuster, D., Schmitz, S., Schuh, G., van der Aalst, W.M.P.: Defining cases and variants for object-centric event data. CoRR **abs/2208.03235** (2022). https://doi.org/10.48550/arXiv.2208.03235
7. Becker, J., Breuker, D., Delfmann, P., Matzner, M.: Designing and implementing a framework for event-based predictive modelling of business processes. In: EMISA. pp. 71–84. GI (2014)
8. van Dongen, B.: BPI Challenge 2017 (2017). https://doi.org/10.4121/uuid:5f3067df-f10b-45da-b98b-86ae4c7a310b
9. van Dongen, B.F., Crooy, R.A., van der Aalst, W.M.P.: Cycle time prediction: When will this case finally be finished? In: OTM. pp. 319–336. Springer (2008). https://doi.org/10.1007/978-3-540-88871-0_22
10. Ehrendorfer, M., Mangler, J., Rinderle-Ma, S.: Assessing the impact of context data on process outcomes during runtime. In: ICSOC. pp. 3–18. Springer (2021). https://doi.org/10.1007/978-3-030-91431-8_1
11. Esser, S., Fahland, D.: Multi-dimensional event data in graph databases. J. Data Semant. **10**(1-2), 109–141 (2021). https://doi.org/10.1007/s13740-021-00122-1
12. Evermann, J., Rehse, J., Fettke, P.: Predicting process behaviour using deep learning. Decis. Support Syst. **100**, 129–140 (2017). https://doi.org/10.1016/j.dss.2017.04.003
13. Folino, F., Guarascio, M., Pontieri, L.: Discovering context-aware models for predicting business process performances. In: OTM. pp. 287–304. Springer (2012). https://doi.org/10.1007/978-3-642-33606-5_18

14. Francescomarino, C.D., Dumas, M., Federici, M., Ghidini, C., Maggi, F.M., Rizzi, W.: Predictive business process monitoring framework with hyperparameter optimization. In: CAiSE. pp. 361–376. Springer. https://doi.org/10.1007/978-3-319-39696-5_22

15. Galanti, R., Coma-Puig, B., de Leoni, M., Carmona, J., Navarin, N.: Explainable predictive process monitoring. In: ICPM. pp. 1–8. IEEE (2020). https://doi.org/10.1109/ICPM49681.2020.00012

16. Harl, M., Weinzierl, S., Stierle, M., Matzner, M.: Explainable predictive business process monitoring using gated graph neural networks. Journal of Decision Systems **29**(sup1), 312–327 (2020). https://doi.org/10.1080/12460125.2020.1780780

17. Huang, T., Metzger, A., Pohl, K.: Counterfactual explanations for predictive business process monitoring. In: EMCIS. vol. 437, pp. 399–413. Springer (2021). https://doi.org/10.1007/978-3-030-95947-0_28

18. de Leoni, M., van der Aalst, W.M.P., Dees, M.: A general process mining framework for correlating, predicting and clustering dynamic behavior based on event logs. Inf. Syst. **56**, 235–257 (2016). https://doi.org/10.1016/j.is.2015.07.003

19. Leontjeva, A., et al.: Complex symbolic sequence encodings for predictive monitoring of business processes. In: BPM. pp. 297–313. Springer (2015). https://doi.org/10.1007/978-3-319-23063-4_21

20. Li, C., van Zelst, S.J., van der Aalst, W.M.P.: Stage-based process performance analysis. In: ICSOC Workshops. pp. 349–364. Springer (2020). https://doi.org/10.1007/978-3-030-76352-7_34

21. Lundberg, S.M., Lee, S.: A unified approach to interpreting model predictions. In: NeurIPS. pp. 4765–4774 (2017)

22. Park, G., Adams, J.N., van der Aalst, W.M.P.: OPerA: Object-centric performance analysis. CoRR **abs/2204.10662** (2022). https://doi.org/10.48550/arXiv.2204.10662

23. Philipp, P., et al.: Analysis of control flow graphs using graph convolutional neural networks. In: ISCMI. pp. 73–77 (2019). https://doi.org/10.1109/ISCMI47871.2019.9004296

24. Rogge-Solti, A., Weske, M.: Prediction of remaining service execution time using stochastic petri nets with arbitrary firing delays. In: ICSOC. pp. 389–403. Springer (2013). https://doi.org/10.1007/978-3-642-45005-1_27

25. Sun, Y., Bauer, B., Weidlich, M.: Compound trace clustering to generate accurate and simple sub-process models. In: ICSOC. pp. 175–190. Springer (2017). https://doi.org/10.1007/978-3-319-69035-3_12

26. Tax, N., Verenich, I., Rosa, M.L., Dumas, M.: Predictive business process monitoring with LSTM neural networks. In: CAiSE. pp. 477–492. Springer (2017). https://doi.org/10.1007/978-3-319-59536-8_30

27. Venugopal, I., Töllich, J., Fairbank, M., Scherp, A.: A comparison of deep-learning methods for analysing and predicting business processes. In: IJCNN. pp. 1–8 (2021). https://doi.org/10.1109/IJCNN52387.2021.9533742

28. Waibel, P., Pfahlsberger, L., Revoredo, K., Mendling, J.: Causal process mining from relational databases with domain knowledge. CoRR **abs/2202.08314** (2022)

29. Wang, C., Cao, J.: Interval-based remaining time prediction for business processes. In: ICSOC. Springer (2021). https://doi.org/10.1007/978-3-030-91431-8_3

30. Wu, Z., et al.: A comprehensive survey on graph neural networks. IEEE Trans. Neural Networks Learn. Syst. **32**(1), 4–24 (2021). https://doi.org/10.1109/TNNLS.2020.2978386