Predictive Process Monitoring

Chiara Di Francescomarino and Chiara Ghidini[✉]

Fondazione Bruno Kessler, Trento, Italy
{dfmchiara,ghidini}@fbk.eu

1 Introduction

Predictive Process Monitoring [29] is a branch of process mining that aims at predicting the future of an ongoing (uncompleted) process execution. Typical examples of predictions of the future of an execution trace relate to the outcome of a process execution, to its completion time, or to the sequence of its future activities.

Being able to predict in advance the outcome of a process execution, the time that a process instance will require to complete, or the activities that will be executed next can be extremely valuable in several domains and scenarios, e.g., for production processes, allowing organizations to prevent undesired outcomes, issues and delays. Indeed, differently from the problem of monitoring business processes in a *reactive* way [28], i.e., so that the violation or the delay is identified only after its occurrence, predicting the violation or the issue before it occurs, would allow for supporting users and organizations in *preventing* it by taking the appropriate preventive countermeasures. Fueled also by the wave of technical developments in Data Science, Predictive Analytics, and data driven Artificial Intelligence, the development of predictive techniques tailored to the field of Process Mining has rapidly established itself both as a vibrant research topic and as an impactful functionality with a direct application in innovative organizational contexts and process mining tools, which often go hand in hand. Examples are the development of new Predictive Process Monitoring pipelines for specific organizations (such as hospitals) [3] and the investigation of explainable Predictive Process Monitoring techniques performed together with leading Process Mining companies such as myInvenio[1] [18] with the aim of incorporating the features within their Process Mining tools (see also [36]).

Predictive Process Monitoring approaches usually leverage past historical complete executions in order to provide predictions about the future of an ongoing (incomplete) case. They usually have two phases: a *training or learning phase*, in which a predictive model is learned from historical (complete) execution traces and a *runtime or prediction phase*, in which the predictive model is queried for predicting the future of an ongoing case.

[1] Recently acquired by IBM as part of the IBM Process Mining suite. See www.ibm.com/cloud/cloud-pak-for-business-automation/process-mining/.

W. M. P. van der Aalst and J. Carmona (Eds.): Process Mining Handbook, LNBIP 448, pp. 320–346, 2022.
https://doi.org/10.1007/978-3-031-08848-3_10

The chapter is structured as follows:[2] after an introduction of a simple explanatory example (Sect. 2) and of the main dimensions characterizing the family of the Predictive Process Monitoring approaches for business processes (Sect. 3), the typical encodings and approaches used for the prediction of outcomes (Sect. 4), numeric values (Sect. 5), and sequences of activities - and related payloads - (Sect. 6), are described in the next three sections, respectively. Finally, Sect. 7 presents new relevant trends in the context of operational support techniques based on Machine Learning and Sect. 8 introduces the main available open source tools supporting Predictive Process Monitoring tasks. We assume as prerequisite for the next sections that the reader has some machine/deep learning knowledge, especially on classification and regression algorithms, as well as on recurrent neural networks. The interested reader can refer to [5,19,20].

2 Running Example

During the execution of a business process, process participants cooperate to satisfy certain business constraints. At any stage of the process enactment, decisions are taken aimed at achieving the satisfaction of these constraints. Being able to predict in advance certain aspects of a process execution allows organizations to take advantage or adapt to desirable future enfolding or to react and be able to prevent an undesirable scenario by taking the appropriate preventive countermeasures.

In this chapter we will illustrate the potential and characteristics of Predictive Process Monitoring by means of a running example in a healthcare scenario.[3] The example describes the process of a patient going to a hospital to perform a radiology exam and related medical checks. The process covers both the clinical aspects, such as the visit(s) and the radiology exam(s) and administrative issues, such as the admission to the radiology department, the computation of the medical bill and its payment. During the process execution, the doctor has to make decisions on whether further exams are required, and - if possible - issued. Depending on the examinations visits can precede and/or follow the radiology exam, which vary in range from Ultrasound, to X-ray, to Pet, MRI, Breast Imaging, and so on. The process typically starts with the admission, the execution of the medical activities (exams and visits) and the computation and paying of the bills. Different executions are nonetheless possible such as a payment in advance, before the visit.

In this scenario, historical information about past executions of the process, and in particular data related to the clinical history of other patients with similar characteristics, could be used to support the hospital predicting the unfolding of a certain execution. As an example, at a certain time during the process execution, one could predict whether a certain patient will require ultrasounds

[2] In this chapter we mostly focus on the main pipeline, omitting aspects mainly related to the preprocessing and evaluation phase.

[3] This example and its instantiations in the following sections are taken and inspired from the running example used in [25].

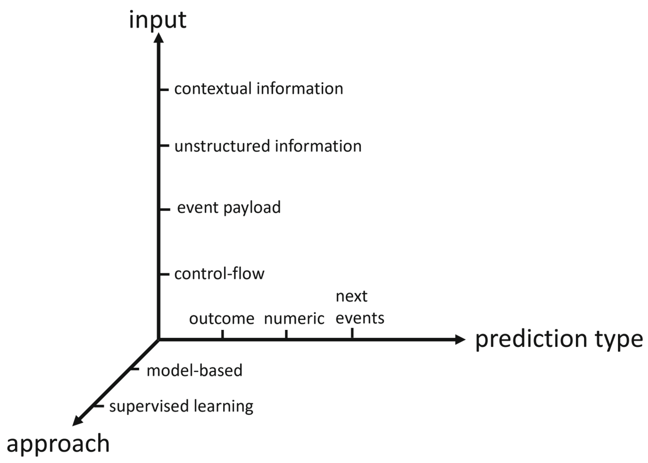

Fig. 1. Predictive Process Monitoring along three dimensions.

and/or at what time. This may be used by the hospital staff to improve or adapt the scheduling of their facilities.

3 The Family of Predictive Process Monitoring Approaches

Although Predictive business Process Monitoring is a relatively young field, it has been growing fast in the latest years, as it is also witnessed by recent surveys on the topic [13,31]. As depicted in Fig. 1, the literature on predictive business process monitoring can be roughly classified along three main dimensions:

- type of prediction (i.e., the type of predictions provided as output);
- type of adopted approach and technique;
- type of information exploited in order to get predictions (i.e., the type of information taken as input).

Concerning the type of prediction, according to the literature [13,46], we can classify the existing prediction types into three main big categories:

- predictions related to predefined categorical or boolean outcome values (*outcome-based predictions*);
- predictions related to measures of interest taking numeric or continuous values (*numeric value predictions*);
- predictions related to sequences of future activities and related data payloads (*next event predictions*).

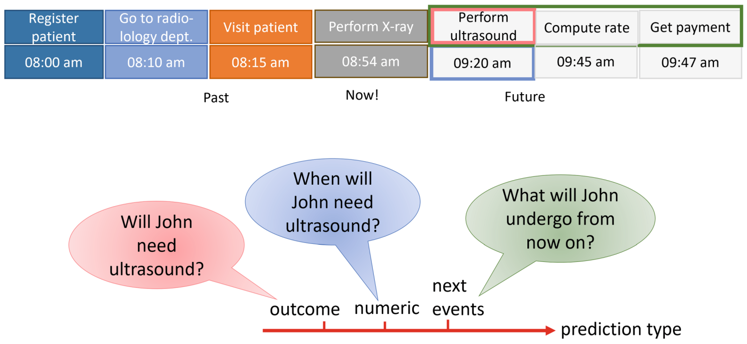

Fig. 2. Types of predictions.

Figure 2 shows an example of an execution trace describing the activities carried out by John. Let us assume that it is 8:54 a.m now. At 8:00 a.m. John has registered to the hospital to undergo some health checks, at 8:10 he was taken to the radiology department where he was visited at 8:15 and he is now having X-rays. Predictive Process Monitoring would allow us to answer different types of questions on the future of John. For instance, we could predict whether John will undergo an ultrasound scan in the future. The answer to this specific question will be a boolean value (e.g., it is true that John will undergo an ultrasound in the future). This is a typical example of an outcome-based prediction. However, this class of predictions also includes predictions assuming categorical values, that is, values that range in a limited and fixed number of possible options. Examples are the class of discount that will be applied to a customer at the end of his shopping, the class of risk of a given execution, or, in our scenario, the specific exam out of a number of options. Another typical question Predictive Process Monitoring could allow us to answer about John's future is, once we know that he will undergo an ultrasound, in how much time he is going to have it. The answer to this question is generally provided in terms of a numeric value (e.g., John is going to have an ultrasound exam in 26 min) and is an example of a numeric-value prediction. Typical examples in this settings are predictions related to the remaining time of an ongoing execution, predictions related to the duration or to the cost of an ongoing case. Finally, we could even predict what John is going to do from now on. The answer to this question is a sequence of future activities (e.g., John will undergo an ultrasound, will ask for his bill and will pay it). Typical examples of predictions falling under this category refer to the prediction of the sequence of the future activities (and of their data payloads) of a process case upon its completion.

Predictive Process Monitoring approaches are usually characterized by two phases. In a first phase, the *training or learning* phase (see the light blue part in Fig. 3), one or more models are built or enriched by leveraging the information contained in the execution log. In the second phase, the *runtime or prediction phase* (see the light green part in Fig. 3), the learned model(s) is(are) exploited

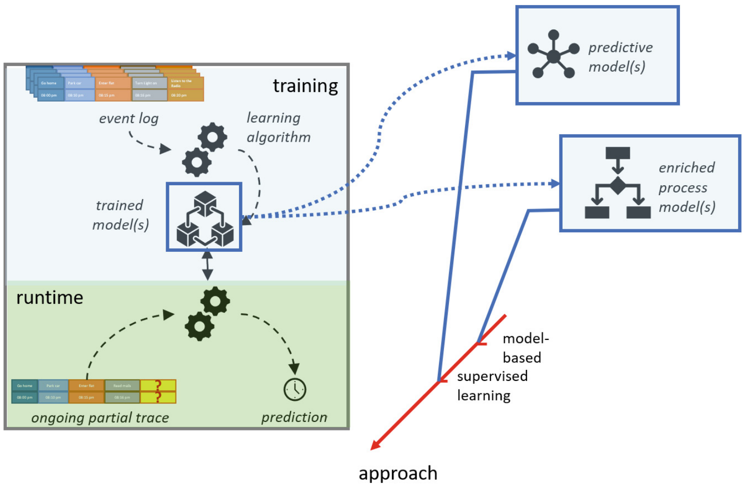

Fig. 3. Types of PPM approaches.

in order to get predictions related to an ongoing execution trace. We can identify two main groups of approaches dealing with the prediction problem:

- approaches relying on an explicit model (*model-based approaches*), e.g., annotated transition systems. The explicit model can either be discovered from the event log and then enriched with the information the log contains or directly be enriched, if an explicit model is already available. In model-based approaches, the model that is then leveraged at runtime in order to get predictions is an (enriched) model in which the process control flow is somehow made explicit (see the blue box in the middle on the right of Fig. 3).
- approaches leveraging machine learning and statistical techniques, e.g., classification and regression models, as well as neural networks. These approaches only rely on (implicit) predictive models built by encoding event log information in terms of features to be used as input for machine/deep learning techniques (see the blue box at the top on the right of Fig. 3).

Finally, we can identify four different types of information that can be used as input to the Predictive Process Monitoring approaches, e.g., for building a model annotated with execution information or for building the features to be used by machine learning approaches:

- information related to the control flow - i.e., the sequence of events. As depicted in the fourth row of Fig. 4, in the example of John's history this is the information related to the activities carried out by John (e.g., check in to the hospital, go to the radiology department, ...).
- information related to the structured data payload associated to the events. This information usually include the timestamp of the events, but it can also include other types of data attributes. For instance, in John's history, besides

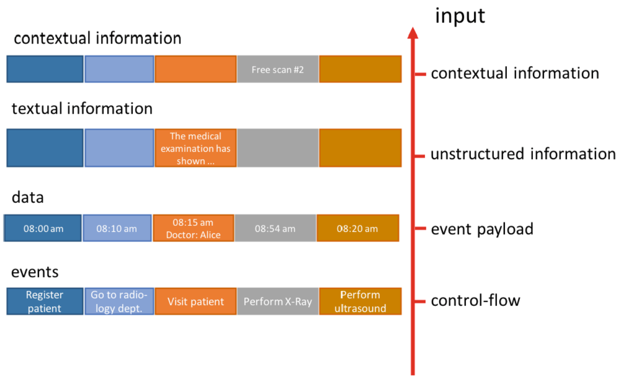

Fig. 4. Information used for making predictions.

the timestamp associated to each event, the data payload of the event `Visit patient` also includes the doctor who has visited John, i.e., *Alice* (see the third row in Fig. 4).

- information related to unstructured (textual) content, which can be available together with the event log. Indeed, it often happens that, together with the structured information related to the events and data payload, some unstructured information is also available. In John's example, for instance, the text of Alice's medical report is available together with the event `visit patient` (see the second row in Fig. 4) and could provide useful information on what John is going to do later on.
- information related to process context, such as workload or resource availability. In John's example, this kind of information could be related for instance to the availability of free ultrasound scan machines (first row in Fig. 4). Contextual information could provide useful information on what John is going to do later on and when. For example, the time required to John to perform an ultrasound could be related to the immediate availability of a scan equipment.

In several approaches, more than one of these types of information is used in order to learn from the past.

After reporting few more details on the model-based approaches and approaches leveraging machine learning in the next subsection, in the following sections, we will mainly focus on machine-learning approaches and on encodings taking into account event and data payload features. We will look in more detail at each of the three prediction type macro-categories, i.e., predicting outcomes, numeric values and sequences of activities (and related payloads), respectively.

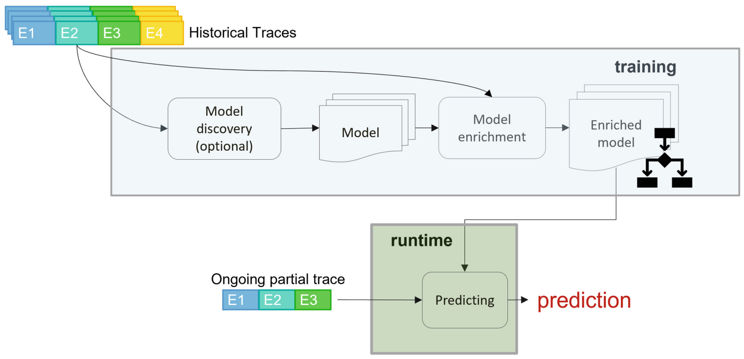

Fig. 5. Overview of the typical phases of model-based approaches.

σ_1 (Visit patient (VP) {*07:00*}, Register patient (RP) {*08:00*}, Compute rate (CR) {*11:00*}, Get payment (GP) {*12:00*}, Perform ultrasound (PU) {*18:00*})
σ_2 (Visit patient (VP) {*08:00*}, Compute rate (CR) {*10:00*}, Perform X-Ray (XR) {*13:00*}, Get payment (GP) {*16:00*})
σ_3 (Compute rate (CR) {*09:00*}, Visit patient (VP) {*13:00*}, Perform X-Ray (XR) {*14:00*}, Get payment (GP) {*15:00*})
σ_4 (Visit patient (VP) {*10:00*}, Register patient (RP) {*11:00*}, Compute rate (CR) {*14:00*}, Get payment (GP) {*15:00*}, Perform X-Ray (XR) {*17:00*})

Fig. 6. Running example model-based approaches.

3.1 Predictive Process Monitoring Approaches

We report here an overview of the main phases related to the two main families of Predictive Process Monitoring approaches, i.e., model-based approaches and approaches based on machine learning.

Figure 5 shows the main phases characterizing the model-based approaches. At training time, an explicit and conformant model (see [7]) can either be already available or can be discovered from the historical traces (see [2,4]) using the optional *Model discovery* phase in Fig. 5. The model is then enriched with information related to the data (*Model enrichment*), as for instance the remaining time extracted from the historical traces. At runtime, the enriched model is used in order to return a prediction.

One of the main model-based approaches leverages a transition system as explicit control-flow model (see Definition 1 in [4]). The transitions system is built based on a given abstraction of the representation of the events in the traces (e.g., the name of the activity), as well as of the representation of the state of the transition system, as for instance the *sequence* of activities executed so far or the *set* of activities occurred so far. For instance, let us consider the simple event log reported in Fig. 6 related to the example described in Sect. 2. Each case relates to a different patient and the corresponding sequence of events indicates the activities executed for a medical treatment of that patient. Given the variability of the process, different interplays are possible between the clinical and administrative activities. In particular in sequence σ_1 the process starts

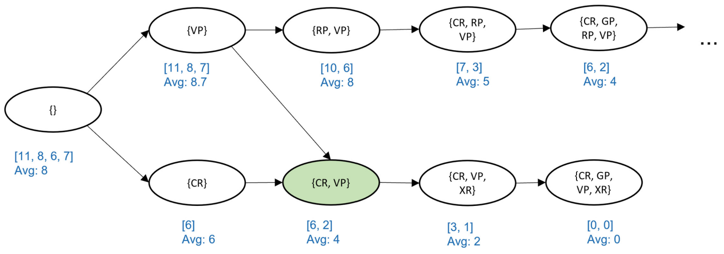

Fig. 7. Annotated transition system obtained from the log reported in Fig. 6.

directly with a visit (possibly due to urgency), while the administrative part is executed in the middle of the process; instead in sequence σ_3, the process starts with a computation of the overall price (possibly due to the request of having a quote) before proceeding further. The event timestamp of each event is reported among brackets nearby the activity. For example, trace σ_2 refers to a process execution in which the activity `Visit patient` is executed at time 08:00, the activity `Compute rate` at time 10:00 and so on. Figure 7 shows the transition system computed using as event representation abstraction the name of the activity and as state representation the activity *set*.

The transition system is then annotated, given a certain measurement function, as for instance the elapsed time or the remaining time, with the corresponding information extracted from the event log. For instance, information about the remaining time can be extracted from the traces and reported for each state of the transition system. This information is then used for making predictions, e.g., on the completion time of an ongoing trace, given a certain prediction function, as for example the average remaining execution time. The transition system in Fig. 7, for instance, is annotated (in blue) with the remaining time of each trace in the event log of Fig. 6. Moreover, for each state, the average of these values is also computed and reported. For example, the state corresponding to the empty set of activities , is annotated with the remaining time of each trace at the beginning of the execution, i.e., 11 h for σ_1, 8 h for σ_2 and so on.

When, at runtime, a prediction about the completion time of a new ongoing trace is required, the annotated transition system can be queried by looking at the state of the transition system corresponding to the ongoing case, and the value of the chosen prediction function returned. For instance, let us assume we want to predict the completion time of an ongoing case $\sigma_t =$ (`Compute rate (CR)` {*12:00*}, `Visit patient (VP)` {*13:00*}). Two measurements are associated to the corresponding state of the transition system in Fig. 7 (see the state in light green), i.e., 6 and 2 hours. Considering the average as prediction function, the average value of the measurements (4 hours) can be used to compute the predicted completion time, i.e., according to the prediction, the patient will complete his process at 17:00.

Several extensions have been proposed to the original approach, such as annotating the transition systems with machine learning models like Naïve Bayes

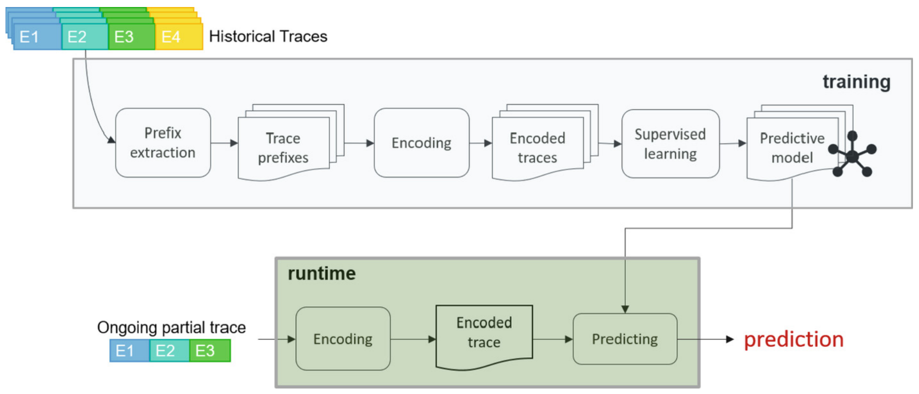

Fig. 8. Overview of the typical phases of approaches based on machine learning.

and Support Vector Regression models [34], taking into account also data payloads [35], combining the annotated transition systems with a context-driven predictive clustering approach [16,17]. Other model-based approaches consider, instead *sequence trees* [9] or stochastic Petri nets [40,41] as explicit models to predict the remaining execution time of a process instance.

Figure 8 sketches the main phases of the typical approaches based on machine learning. These approaches usually require that trace prefixes are extracted from the historical execution traces (*Prefix extraction* phase). This is due to the fact that at runtime predictions are made on incomplete traces, so that correlations between incomplete traces and what we want to predict (*target variables* or *labels*) have to be learned in the training phase. After prefixes have been extracted, prefix traces and labels (i.e., the information that has to be predicted) are encoded in the form of feature vectors (*Encoding* phase). Encoded traces are then passed to the (supervised learning) techniques in charge of learning from the encoded data one (or more) predictive model(s) (*Encoding* phase). At runtime, the incomplete execution traces i.e., the traces whose future is unknown, should also be encoded as feature vectors and used to query the predictive model(s) so as to get the prediction (*Predicting* phase).

In this chapter we will mainly focus on approaches leveraging machine learning - and in particular supervised learning - techniques.

4 Predicting Outcomes

Outcome predictions are predictions related to (categorical) case outcomes [46]. Typical examples of outcome predictions in the Predictive Process Monitoring literature are predictions related to risks or related the fulfilment of a predicate [11,29].

Given an event log L and a prefix execution trace $\sigma_i^m = <e_1, \ldots, e_m>$ of length m, the overall idea is learning a function $f_c(L, \sigma_i^m)$ returning a categorical

σ_1 (`Visit patient` {*33*, *clinic*},...,`Perform ultrasound` {*33*, *radiology*}): false

...

σ_k (`Compute rate` {*56*, *general lab*},..., `Get Payment` {*56*, *admin*}): true

Fig. 9. Running example with an outcome label

value $\overline{label_i}$, which is as close as possible to $label_i$, i.e., the actual (categorical) value of the variable that we aim to predict (e.g., whether the predicate will be actually fulfilled).

As described in the previous section, when dealing with approaches based on machine learning, one of the main steps to be carried out deals with encoding the information contained in (prefix) execution traces and corresponding labels in a format that is understandable by machine learning techniques. This would allow the technique to train, and hence learn, from encoded data a predictive model. In order to train a model, each (prefix) execution trace σ_i, (and its corresponding label) have to be represented through a feature vector $g_i = (g_{i1}, g_{i2}, ...g_{ih}, label_i)$.

In this section (and in the next two sections) we will present first the typical encodings used with the corresponding type of predictions[4] and then the main (machine-learning) pipelines/approaches used to build the predictive model and query it.

4.1 Typical Data Encodings

To exemplify the different data encoding techniques, we consider the very simple log in Fig. 9 pertaining to our running example of Sect. 2. Similarly to the log used in Sect. 3.1, also in this log each case relates to a different patient and the corresponding sequence of events indicates the activities executed for a medical treatment of that patient. `Visit patient` is the first event of sequence σ_1. Its data payload "{*33*, *radiology*}" corresponds to the data associated to attributes *age* and *department*[5]. Note that the value of *age* is static: it is the same for all the events in a case, while the value of *department* is different for every event. In the payload of an event, the entire set of attributes available in the log is considered as well. In case for some event the value for a specific attribute is not available, the value $\perp$ (*unknown*) is specified for it.

Given a case prefix, we aim at predicting whether the patient will recover soon (*true*), or not (*false*). We report the corresponding value, i.e., the corresponding label, for each case after the semicolon in Fig. 9.

Boolean Encoding. In the *boolean encoding* sequences of events are represented as feature vectors, in such a way that each feature corresponds to an event class (an activity) from the log. In particular, the boolean encoding represents a sequence

[4] Please note that some types of encodings can be used for different types of predictions. For instance encodings related to outcome-based and numerical predictions are exactly the same - except for the type of the label.

[5] We omit here for simplicity the information related to timestamps.

Table 1. Typical outcome-based encodings for the example in Fig. 9.

	`Visit patient`	`Perform ultrasound`	...	`Get Payment`	label
σ_1	1	1	...	0	false
...					
σ_k	0	0	...	1	true

(a) *boolean* encoding.

	`Visit patient`	`Perform ultrasound`	...	`Get Payment`	label
σ_1	2	1	...	0	false
...					
σ_k	0	0	...	4	true

(b) *frequency-based* encoding.

	event_1	...	event_m	label
σ_1	`Visit patient`		`Perform ultrasound`	false
...				
σ_k	`Compute rate`		`Get Payment`	true

(c) *simple-index* encoding.

	age	department_last	label
σ_1	*33*	*radiology*	false
...			
σ_k	*56*	*admin*	true

(d) *latest-payload* encoding.

	age	event_1	...	event_m	...	department_last	label
σ_1	*33*	`Visit patient`		`Perform ultrasound`	...	*radiology*	false
...							
σ_k	*56*	`Compute rate`		`Get Payment`	...	*admin*	true

(e) *index latest-payload* encoding.

	age	event_1	...	event_m	...	department_1	...	department_m	label
σ_1	*33*	`Visit patient`		`Perform ultrasound`		*clinic*		*radiology*	false
...									
σ_j	*56*	`Compute rate`		`Get Payment`		*general lab*		*admin*	true

(f) *complex index-based* encoding.

σ_i through a feature vector $g_i = (g_{i1_A}, g_{i2_A}, ... g_{ih_A}, label_i)$, where h_A is the size of the event class alphabet $A = \{a_{1_A}, \ldots, a_{h_A}\}$ and if g_{ij_A} corresponds to the event class $a_{j_A} \in A$ then:

$$g_{ij} = \begin{cases} 1 & \text{if } a_{j_A} \text{ occurs in } \sigma_i \\ 0 & \text{if } a_{j_A} \text{ does not occur in } \sigma_i \end{cases}$$

For instance, the encoding of the example reported in Fig. 9 with the *boolean* encoding is shown in Table 1a.

Frequency-Based Encoding. The *frequency-based* encoding, instead of boolean values, represents the control flow in a case with the frequency of each event class in the case. The frequency-based encoding $g_i = (g_{i1_A}, g_{i2_A}, ... g_{ih_A}, label_i)$ of σ_i, is such that, if g_{ij_A} corresponds to the event class $a_{j_A} \in A$ then:

$$g_{ij} = \begin{cases} n & \text{if } a_{j_A} \text{occurs } n \text{ times in } \sigma_i \\ 0 & \text{if } a_{j_A} \text{does not occur in } \sigma_i \end{cases}$$

Table 1b shows the *frequency-based* encoding for the example in Fig. 9, assuming that `Visit patient` occurs two times in σ_i and `Get Payment` occur four times in σ_k.

Simple-Index Encoding. Another way of encoding a sequence is by taking into account also information about the order in which events occur in the sequence, as in the *simple-index* encoding. Here, each feature corresponds to a position in the sequence and the possible values for each feature are the event classes. The resulting feature vector g_i of the simple-index encoding of an execution trace σ_i of length m is $g_i = (a_{i1}, a_{i2}, ...a_{im}, label_i)$, such that a_{ik} corresponds to the event class of the event at position k in σ_i. By using this type of encoding the example in Fig. 9 would be encoded as reported in Table 1c.

Latest-Payload Encoding. The *latest-payload* encoding takes into account both the static and the dynamic data attributes of the traces. The value of static attributes (trace attributes) is the same for all the events in the sequence, while the value of dynamic data attributes (event attributes) changes for different events. However, in this encoding, data attributes, also the dynamic ones, are all treated as static features without taking into consideration their evolution over time. Indeed, the latest-payload encoding encodes the data attributes and the data of the latest payload. The latest-payload encoding g_i of an execution trace σ_i of length m is $g_i = (s_i^1, \dots, s_i^u, d_{im}^1, \dots, d_{im}^r, label_i)$, where each s_i is a static feature and each d_{im} is a dynamic feature associated to the last event, i.e., the event at position m. Table 1d shows this encoding for the example in Fig. 9.

Index Latest-Payload Encoding. The *index latest-payload* encoding adds the latest encoding to the simple-index encoding. The resulting feature vector g_i, for a sequence $g_i = \sigma_i$, is $g_i = (s_i^1, \dots, s_i^u, a_{i1}, a_{i2}, \dots, a_{im}, d_{im}^1, \dots, d_{im}^r, label_i)$, where each s_i is a static feature, each a_{ij} is the event class at position j and each d_{im} is a dynamic feature associated to the event at position m. Table 1e reports this encoding for the example in Fig. 9.

Complex Index-Based Encoding. In the *complex-based* encoding, the dynamic nature of the dynamic information is considered and its evolution over time is taken into account. The resulting feature vector g_i, for a sequence σ_i, is $g_i = (s_i^1, .., s_i^u, a_{i1}, a_{i2}, ..a_{im}, d_{i1}^1, d_{i2}^1, \dots, d_{im}^1, \dots, d_{i1}^r, d_{i2}^r, ...d_{im}^r, label_i)$, where each s_i is a static feature, each a_{ij} is the event class at position j and each d_{ij} is a dynamic feature associated to an event. The example in Fig. 9 is transformed into the encoding shown in Table 1f.

4.2 Mostly Used Approaches: Classification-Based Approaches

Different pipelines and frameworks have been proposed for providing outcome predictions. Most of them relies on classification techniques[6] (e.g., Decision Tree, Random Forest, Support Vector Machine) for the *supervised learning* phase [12, 23, 25, 29]. Moreover, most of these pipelines have been enriched with a *Bucketing* phase [46] (see the orange blocks in Fig. 10). The idea is that at training time

[6] Note that deep learning techniques can also be used for predicting outcomes [52], however we focus here on the mostly used approaches.

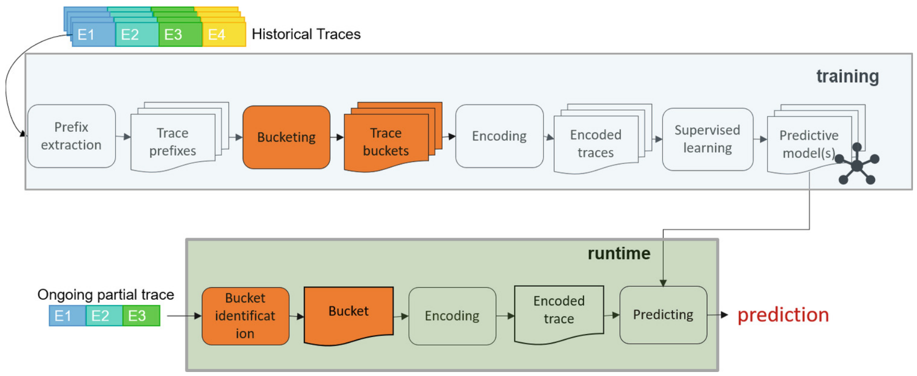

Fig. 10. Typical outcome-based pipeline

multiple predictive models are trained. Specifically, the log of prefix traces is divided in multiple buckets and each bucket is used to train a different classifier. At runtime, the most suitable bucket is identified and the corresponding classifier used for predicting the outcome.

The *Bucketing* phase has been instantiated in different ways in the Predictive Process Monitoring literature. For instance, in [12] trace clustering has been used to group prefix traces. Specifically, at training time, a clustering algorithm has been leveraged to cluster together prefix traces sharing a similar control flow. For each cluster, the data payload of the prefix traces in the cluster, once encoded in the proper format, has then been used to train a classifier. At runtime, the cluster of the incomplete ongoing trace is identified, i.e., the cluster containing the trace prefixes closest to the current incomplete trace, and the corresponding classifier queried in order to get the prediction. In [25], instead, a bucket consists of a set of prefix traces of the same length. Also in this case, at training time, a classifier for each prefix length k is built by learning from all prefix traces of length k. At runtime, the classifier of the same length of the ongoing trace is identified and the prediction returned.

5 Predicting Numeric Values

Numeric value predictions are predictions related to quantitative measures of interest of business process executions. Typical examples of numeric predictions in the Predictive Process Monitoring literature are predictions related to time, cost or generic process performance [1,8,48].

Given an event log L and a prefix execution trace $\sigma_i^m = <e_1, \ldots, e_m>$ of length m, the overall idea is learning a function $f_n(L, \sigma_i^m)$ returning a numerical value $\overline{label_i}$, which is as close as possible to $label_i$, i.e., the actual (numerical) value of the variable that we aim to predict (e.g., the remaining cycle time until the completion of the execution).

σ_1 (`Visit patient` {*33*, *clinic*}, ..., `Perform ultrasound` {*33*, *radiology*}): 3.5
...
σ_k (`Compute rate` {*56*, *general lab*}, ..., `Get Payment` {*56*, *clinic*}): 5

Fig. 11. Running example with a numeric label

Table 2. Typical numeric-based encodings for the example in Fig. 9.

	`Visit patient`	`Perform ultrasound`	...	`Get Payment`	label
σ_1	1	1	...	0	3.5
...					
σ_k	0	0	...	1	5

(a) *boolean* encoding.

	`Visit patient`	`Perform ultrasound`	...	`Get Payment`	label
σ_1	2	1	...	0	3.5
...					
σ_k	0	0	...	4	5

(b) *frequency-based* encoding.

	event_1	...	event_m	label
σ_1	`Visit patient`		`Perform ultrasound`	3.5
...				
σ_k	`Compute rate`		`Get Payment`	5

(c) *simple index* encoding.

	age	department_last	label
σ_1	*33*	*radiology*	3.5
...			
σ_k	*56*	*clinic*	5

(d) *latest payload* encoding.

	age	event_1	...	event_m	...	department_last	label
σ_1	*33*	`Visit patient`		`Perform ultrasound`	...	*radiology*	3.5
...							
σ_k	*56*	`Compute rate`		`Get Payment`	...	*clinic*	5

(e) *index latest payload* encoding.

	age	event_1	...	event_m	...	department_1	...	department_m	label
σ_1	*33*	`Visit patient`		`Perform ultrasound`		*clinic*		*radiology*	3.5
...									
σ_j	*56*	`Compute rate`		`Get Payment`		*general lab*		*clinic*	5

(f) *complex index-based* encoding.

5.1 Typical Data Encodings

Let us consider the running example of Fig. 9 and let us assume that this time we would like to predict the time required for completing the execution (reported in Fig. 11 after the semicolon).

Encodings typically used for numeric predictions are the same as the ones used for categorical predictions, except for the label, which is a numerical value rather than a boolean or a categorical value. Table 2 summarizes the boolean, frequency, simple-index, latest-payload, index latest-payload and complex-index encodings for numeric-based predictions.

5.2 Mostly Used Approaches: Regression-Based Approaches

Pipelines and frameworks proposed for numeric predictions are quite similar to the ones for outcome predictions. Most of them relies on regression techniques[7] (e.g., Regression Trees, Random Forest, XGBoost) for the *supervised learning* phase [23,29].

6 Predicting Next Events

Next event predictions are predictions related to the unfolding of the future events - until the end - of an incomplete ongoing trace [45]. Next event predictions can be related to the sequence of next event classes, but also to the next data payloads associated to the events, as for instance, the timestamps or the resources associated to the next event(s).

In case of activity predictions, given an event log L and a prefix execution trace $\sigma_i^m = <e_1, \ldots, e_m>$ of length m, the overall idea is learning a function $f_{sa}(L, \sigma^m)$ returning a sequence of next event classes that is as close as possible to $a_{m+1}, \ldots, \omega$, i.e., to the activity suffix of the current ongoing trace.

Most of the approaches for next activity predictions typically first learn a function f_{1a} that, given the first m events of a trace σ_i^m, predicts the next event class, i.e., the event class that will occur at time step $m + 1$. The suffix of the ongoing trace σ_i^m is then predicted until the last event ω, by predicting the next event iteratively, that is by learning the function f_{sa}:

$$f_{sa}(L, \sigma_i^m) = \begin{cases} f_{1a}(\sigma^m) & \text{if } f_{1a}(L, \sigma_i^m) = \omega \\ f_{sa}(L, < e_1, e_2, ..., e_m, e >) & \text{otherwise} \\ \quad \text{with } f_{1a}(L, \sigma_i^m) \text{ as } e\text{'s event class} & \end{cases} \tag{1}$$

Similarly, when predicting the values of the next events' data attribute x, e.g., the next timestamps, the idea is learning a function $f_{sx}(L, \sigma^m)$ returning a sequence of values of the data attribute x that is as close as possible to the sequence of values actually held by the attribute x in the next events of the ongoing trace.

In the next subsection describing the typical data encodings, we mainly focus on the encoding for the next event class prediction. The results can then be extended to the prediction of other data attributes related to the next event, as well as to predictions related to next events, as described in (1).

6.1 Typical Data Encodings

Let us consider the running example described in Fig. 9 enriched with timestamp information and let us assume that we want to predict the next activity related to the next time step (i.e., the activity at time step $m+1$). The actual activity at time step $m + 1$ is reported after the semicolon for the training traces in Fig. 12.

[7] Note that deep learning techniques can also be used for predicting numeric predictions [45], however we focus here on the mostly used approaches.

σ_1 (`Visit patient` {*33*, *08:00*, *clinic*}, ..., `Perform ultrasound` {*33*, *11:00*, *radiology*}): `Check X-ray`
...
σ_k (`Compute rate` {*56*, *13:00*, *general lab*} ,..., `Get payment` {*56*, *18:00*, *admin*}): `Emit receipt`

Fig. 12. Running example with next activity as label

Table 3. Typical sequence-based encodings for the example in Fig. 12.

	event_1	...	event_m	label
σ_1	1 0 0 0 0 0	...	0 1 0 0 0 0	0 0 0 0 1 0
...				
σ_k	0 0 1 0 0 0	...	0 0 0 1 0 0	0 0 0 0 0 1

(a) *one-hot* encoding

	event_1				...	event_m				label
	a_1	δ_1	h_1	w_1		a_m	δ_m	h_m	w_m	
σ_1	1 0 0 0 0 0	0	8	Mon	...	0 1 0 0 0 0	1	11	Mon	0 0 0 0 1 0
...										
σ_k	0 0 0 0 1 0	0	13	Sat	...	0 0 0 0 0 1	2	18	Sat	0 0 0 0 0 1

(b) *one-hot with temporal features* encoding

One-Hot Encoding. The *one-hot encoding* allows categorical data to be transformed into a numeric format. It relies on the existence of an alphabet of activities. Given the set $A = \{a_{1_A}, \dots a_{h_A}\}$ of all possible activities, an ordering function $idx : A \rightarrow \{1, \dots, |A|\} \subseteq \mathbb{N}$ is defined on it, such that $a_{i_A} <> a_{j_A}$ if and only if $i_A <> j_A$, i.e., two activities have the same A-index if and only if they are the same activity.

For instance, in the example in Fig. 12, if the activity alphabet is $A = \{$`Visit patient`, `Perform ultrasound`, `Compute rate`, `Get Payment`, `Check X-ray`, `Emit receipt`$\}$, the function idx $:A \rightarrow \{1, 2, 3, 4, 5, 6\}$ can be defined such that $idx($`Visit patient`$) = 1$, $idx($`Perform ultrasound`$) = 2$, $idx($`Computerate`$) = 3$ and so on. Each event $e_{ij} \in \sigma_i$ is then encoded as a vector (A_{ij}) where the features are all set to 0, except the one occurring at the index of its event class, which is set to 1. In the training phase, the event class of the next event e_{m+1}, which represents the target variable or label, is also encoded in the corresponding vector (A_{im}). The trace is finally encoded by composing the vectors obtained from all activities in the trace and the next activity into a matrix. The encoding of the trace σ_i is hence given by $g_i = ((A_{i1}), ..., (A_{im}), (A_{im+1}))$. The one-hot encoding related to the example in Fig. 12 is reported in Table 3a.

One-Hot Encoding with Temporal Features. The one-hot encoding, which takes into account only the activities, can be enriched with other information. For instance, another encoding used with activity sequences combines the one-hot encoding of features related to event classes and features related to time [45]. In the *one-hot encoding with temporal features*, given the set $A = \{a_{1_A}, \dots a_{m_A}\}$ of all possible activities, each event $e_{ij} \in \sigma_i$ is encoded as the one-hot encoding of its event class enriched with three additional features pertaining to time. The first one relates to the time difference between the considered event and the one of the previous event (δ_i), the second one reports the time since midnight (h_i), thus allowing for distinguishing between working and night time, and the last one refers to the time since the beginning of the week (w_i), thus allowing for distinguishing between business and non-working days. Also in this case, in the training phase, the label, i.e., the event class of the next event e_{m+1} is also encoded with the one-hot encoding. The one-hot encoding with temporal features related to the example in Fig. 12 is reported in Table 3b.

Embedding-Based Encoding. The *embedding-based encoding* is typically used when the number of the possible values of one or more categorical variables is high and the one-hot encoding may cause an exponential growth of the feature vector dimensionality. In the embedding-based encoding, categorical data with an alphabet of possible values of size m is mapped into a n-dimensional embedding space (where n is the chosen dimensionality of the embedded space) that encodes the values of the categorical attribute so that values that are closer in the vector space are expected to be similar.

6.2 Mostly Used Approaches: LSTM-Based Approaches

Most of the approaches for next event predictions rely on Recurrent Neural Networks and, more specifically, on LSTM (Long-Short Term Memory) architectures [6,26,45].[8] This type of deep learning approaches, by using recurrent connections in a single block (LSTM cell), is indeed particularly suitable to deal with sequence problems. Different types of LSTM architectures have been proposed in the literature for predicting the label associated to the next event and its data attributes.

For instance in [45] three types of architectures have been proposed in order to predict both next activity and the timestamp of the next event and then, iteratively, suffix prediction and remaining cycle time: a first type with separate layers for activity and timestamp prediction, a second type with shared LSTM layers for both activity and timestamp prediction and finally a third one with some shared and some separate layers. The architecture proposed in [6] for predicting the next activity and its timestamp and the remaining cycle time and suffix for a running case is a composition of LSTMs and feedforward layers. In [26] an encoder-decoder framework based on LSTMs is proposed to predict the next activity and the suffix of an ongoing case. The encoder maps an input sequence into a set of high dimensional vectors and the decoder returns it back into new sequence that can be used for prediction tasks.

7 New Trends in ML-Driven Operational Support

Besides the mainstream works in the field of Predictive Process Monitoring, new research trends and directions focusing on ML-driven operational support have recently started being investigated and developed. Some of these new trends are summarised in the following subsections.

[8] Note that the usage of LSTM architectures is not limited to next event predictions - they are indeed used also for outcome and numerical predictions - nevertheless it has been widely used in the literature for this type of predictions.

Table 4. Simple-index encoding enriched with some inter-case features for the example in Fig. 9.

	event_1	event_m	simult. trace #	avg. duration	label
σ_1	`Visit patient`	`Perform ultrasound`	10	6	False
σ_k	`Compute rate`	`Get Payment`	18	8	True

7.1 Intercase Predictions

In classical works, Predictive Process Monitoring methods assume that the predicted value of interest of an ongoing case only depends on intra-case information, as for instance on the execution history of that specific case. This assumption results in encodings that include past events, inter-event durations, and other case-related attributes. However, the only intra-case assumption does not hold in many real-life scenarios. For example, in situations where cases share limited resources, the completion time of a case heavily depends on other cases that are running at the same time [42,43].

Inter-case information can be encoded in different ways, as for instance by aggregating data related to traces running simultaneously. Examples of aggregated inter-case information that can be encoded together with the intra-case features are the number of traces and the average duration of traces being executed in the same time window in which the considered trace (prefix) is being executed, e.g., the number of traces and the average duration of traces executed in the same day of the current prefix trace. Table 4 shows an example of a simple-index encoding enriched with these two simple inter-case features related to the example reported in Fig. 9, where we assume that 10 other traces are running the same day in which σ_1^m is being executed and that their average duration is 6 hours, while 18 traces are running simultaneously to σ_k^m with an average duration of 8 hours.

Taking into account the inter-case dimension is a challenging problem, since, on the one hand, we would like to take into account as much inter-case information as possible as the levels of dependencies among cases can greatly vary in different scenarios and, on the other hand, encoding several features for a large number of simultaneously running cases may lead to a feature space explosion.

7.2 Explainable Predictions

In many applications of Predictive Process Monitoring techniques, users are asked to trust a model helping them making decisions. However, users would need a certain level of trust towards the predictive model: a doctor will not operate on a patient simply because the operation has been predicted or recommended by the model. Understanding the rationale behind predictions would certainly help users decide when to trust or not to trust them.

Explainability techniques are a way to implement responsible process decision making (see [30]) and can help us to this aim. Different explainability techniques

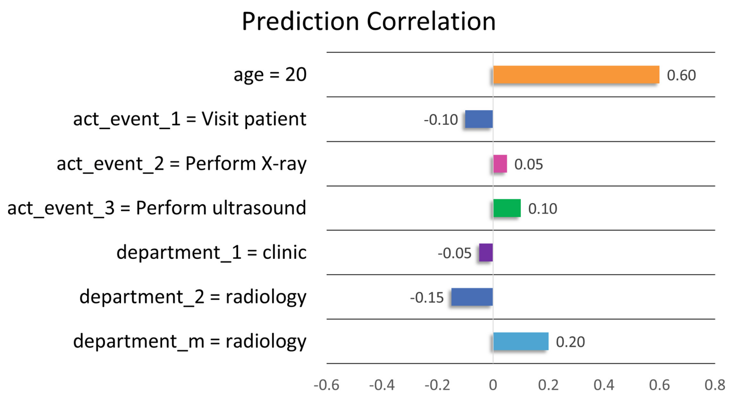

Fig. 13. Example of an explanation plot related to the prediction for σ_j.

have been proposed in the XAI (Explainable Artificial Intelligence) literature. Some of these techniques have already been experimented in the field of Predictive Process Monitoring in order to support users in understanding the overall predictive model [33] or the specific predictions it provides [18,44,51]; with model-agnostic techniques, i.e., techniques that can be applied to any predictive model, as in the case of [18] or with techniques specific to the predictive model used, as in the case of XNAP [51] and the attention layer [44] for neural networks.

As an example of prediction explanation related to a trace instance,[9] let us assume that we have trained our predictive model by encoding the training set of the example reported in Fig. 9 with the complex-index encoding (see Table 1f) and that, for our current ongoing trace σ_j (`Visit patient` {*20*, *clinic*}, `Perform X-Ray` {*20*, *radiology*}, `Perform ultrasound` {*20*, *radiology*}), which we have observed up to the event 3, the prediction of our predictive model is that the patient will recover soon. In order to understand whether we can trust or not the prediction, we would need to understand why our predictive model has returned such a prediction. Figure 13 shows an example of a possible explanation returned by a prediction explainer as LIME[37] or SHAP[27] applied to our specific Predictive Process Monitoring problem. The plot shows the impact of each feature (and related value) towards (in case of positive values) or against (in case of negative values) the fast recovery of the patient.[10] In the example, the feature

[9] Note that we provide here the idea of prediction explanations focusing on those related to a trace instance. However, aggregated trace prediction explanations (event log explanations) [18], as well as prediction model explanations [44] have also been investigated in the literature.

[10] Note that the semantics of the values on the x axis changes according to the explanation technique used for the plot. For instance, in the case of SHAP, the values on the x axis represent the SHAP values of the feature (and the related value) for the specific instance, that is the contribution of the feature towards the prediction with respect to the average value.

that has impacted most on the prediction of the fast recovery of the patient is her young age.[11]

Furthermore, the explanations used for making predictions more trustable to the users can be eventually used also for understanding the reasons why a predictive process model is wrong and hence use them to improve the model accuracy [38].

7.3 Predictions with A-Priori Knowledge

Past event logs, or more in general knowledge about the past, is not the only important source of knowledge that can be leveraged to make predictions. In many real life situations, cases exist in which, together with past execution data, some case-specific additional knowledge (*a-priori knowledge*) about the future is available and can be leveraged for improving the predictive power of a Predictive Process Monitoring technique. Indeed, this additional a-priori knowledge is what characterizes the future context of the process executions that will affect the development of the currently running cases.

We can think for instance to the occurrence of a strike, which may cause the delay or the cancellation of a flight in the travel process of a passenger, or to the temporary unavailability of a surgery room, which may delay or even rule out the possibility of executing certain activities in a patient treatment process. In this kind of scenarios, the information about the strike or about the unavailability of the surgery room is often available in advance. However, traditional Predictive Process Monitoring approaches, which only learn from the most frequent observed behaviours, are not able to take into account this knowledge. They will predict that the next activities of the passenger will be the usual ones, as if there is no strike, e.g., having the security check, moving to the boarding gate 3, boarding, ... While it is impractical to retrain the predictive algorithms to take into consideration this additional knowledge every time it becomes available, it is also reasonable to assume that considering it in some way would allow the Predictive Process Monitoring algorithm to predict for instance that the passenger will be moved to gate 2 and that there will be no boarding, and hence to improve the accuracy of the predictions on an ongoing case.

A possibility to deal with a-priori knowledge is to take into account this knowledge K at prediction time by guiding the Predictive Process Monitoring algorithm towards a solution that is compliant to the a-priori knowledge [14]. In [14] for instance, an approach using LSTM for predicting the next activities has been enriched with a mechanism able to take into account background knowledge K expressed in terms of LTL formulae in order to guide the LSTM algorithm to make predictions compliant with the a-priori knowledge. The LSTM approach keeps returning likely predictions on the suffix of the current ongoing trace (up to the last event ω) until it does not find a suffix that is compliant with K. More in detail, the LSTM network uses a beam search algorithm for

[11] Note that different types of explanations can be returned depending on the type of encoding that has been used.

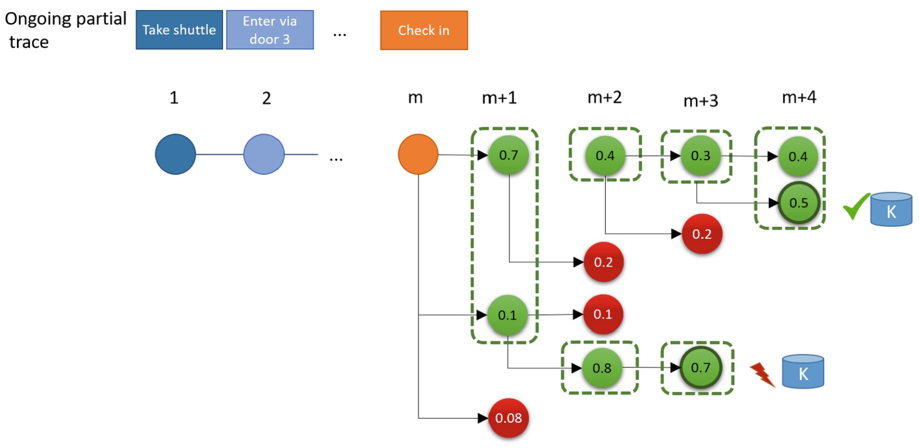

Fig. 14. Beam search in the a-priori approach.

considering at each time step the top beam-width *bw* most likely next events. Figure 14 shows the idea of the beam-search approach with $bw = 2$. $\sigma^m = <$`Take shuttle`, `Enter via door 3`, `Check in`$>$ is the current ongoing trace at time step m. At time step $m+1$, among the three possible next events we take the *bw* most likely next events (the green nodes in Fig. 14) and keep exploring those future paths. At time step $m+2$, we again select the 2 most likely next events and keep exploring the next events of these sequences. Whenever we find a sequence that is not compliant with `K`, as at time step $m+3$, we discard that path and we keep on exploring *bw* compliant paths. We stop whenever we predict the last event ω (see the circle with the thicker border) and the considered trace is still compliant with `K`.

7.4 Prescriptive Process Monitoring

Predictive Process Monitoring techniques are able to predict the likelihood of a positive outcome, the time required for completing an execution or the next activities that will be executed. However, all these techniques, are limited to the prediction. They do not support further stakeholders in making decisions on whether it is worth to intervene to avoid undesired outcomes and what to do next to optimize a given Key Process Performance Indicator (KPI) [24,32,47,50].

Prescriptive Process Monitoring aims to overcome this limit of Predictive Process Monitoring by supporting or prescribing stakeholders with decisions on whether to take actions in order to prevent or mitigate the occurrence of an undesired outcome [32,47] or on the activities to take for optimizing a certain measure of interests [24,50].

In the first scenario [32,47], predictions are used in order to evaluate through a cost model the tradeoffs between the cost of intervention to mitigate undesired outcomes and the cost of compensating unnecessary interventions. For instance, in the example related to the patient recovery described in Sect. 4, if the prediction related to an ongoing trace is that the patient will not recover soon, a surgery may increase the likelihood that the patient will recover soon and hence reduce anyway the cost for the hospital. However, the surgery has a cost, so that if the surgery has been planned because of a wrong prediction, then the cost of the surgery is unnecessary and hence should be avoided.

In the second scenario [24,50], predictions are used to uncover the future of different continuations of the current trace, so as to identify and hence recommend the one(s) leading to the best value for the KPI of interest. For instance, we can consider the example of the patient recovery described in Sect. 5. If the aim is recommending next activities to minimize the remaining cycle time until the completion of the execution of an ongoing trace σ^m of length m, possible next activities at step $m+1$ can be considered. For each possible continuation of σ^m, σ^{m+1}, the remaining time until the end of the execution can be predicted and the next activity corresponding to the minimum cycle time recommended.

8 Tool Support

The research related to Predictive Process Monitoring has been paired with the development of non-commercial plugins and tools with the purpose to be used and improved by the research community. We briefly illustrate in the following three among the main open-source tools supporting Predictive Process Monitoring.

8.1 Predictive Process Monitoring in ProM

ProM [15] is one of the most used and known tool in Process Mining. It is a framework collecting a number of plugins, working independently one from the other, and each focused on implementing a specific task. Among its variety of plugins, ProM also collects several plugins implementing techniques for the prediction of outcomes (e.g., [8,29]), for the prediction of numerical values (e.g., [1,10,16,23]), as well as for the prediction of next activity sequences (e.g., [35]). Some of them leverage model-based approaches (e.g. [1]), while others rely on machine-learning solutions (e.g., [10]).

8.2 Predictive Process Monitoring in Apromore

Apromore [22] is a well known and established tool. It is an advanced process model repository that allows to hold, analyse, and re-use large sets of process models. The tool is web-based and therefore it allows the easy integration of new plug-ins in a service oriented manner. This tool aims both at allowing practitioners to deal with the challenges of stakeholders of processes, and researchers to develop and benchmark their own techniques with a strong emphasis on the separation of concerns. The only plug-in performing Predictive Process Monitoring related challenges in Apromore is the one described in [49]. This plug-in performs outcome-based, numeric-based prediction, as well as next event predictions.

8.3 Predictive Process Monitoring in Nirdizati

Nirdizati [21,39] is a web-based application for supporting users in building, comparing, and analyzing predictive models that can then be used to perform predictions on the future development of an ongoing case. Differently from the other tools, Nirdizati specifically addresses Predictive Process Monitoring problems. Nirdizati, which collects a rich set of different state-of-the-art approaches based on machine learning algorithms, supports users to deal with different predictive monitoring tasks: outcome-based, numeric and next activities predictions. Moreover, it provides services for supporting the users in tuning the hyperparameters of the specific technique, the possibility of adding some simple intercase features in the encodings, as well as some incremental algorithms, so as to be able to incrementally update the predictive model as soon as new execution traces are available. Finally, it also offers several plots for the results visualisation, thus supporting the users in the predictive model comparison tasks.

Acknowledgements. The work described in this chapter describes the effort of a number of people. We would like to thank Marlon Dumas, Marcello La Rosa, Anna Leontjeva, Fabrizio Maria Maggi, Williams Rizzi, Arik Senderovich, Irene Teinemaa, Ilya Verenich, and Anton Yeshchenko for their precious cooperation and all the (also joint) work that led to this chapter.

References

1. van der Aalst, W.M.P., Schonenberg, M.H., Song, M.: Time prediction based on process mining. Inf. Syst. **36**(2), 450–475 (2011)
2. Aalst, W.: Foundations of Process Discovery. In: van der Aalst, W.M.P., Carmona, J. (eds.) Process Mining Handbook. LNBIP, vol. 448, pp. 37–75. Springer, Cham (2022)
3. Aringhieri, R., et al.: Leveraging structured data in predictive process monitoring: the case of the ICD-9-CM in the scenario of the home hospitalization service. In: Proceedings of "Towards smarter health Care: Can Artificial Intelligence Help? (SMARTERCARE)" Workshop. CEUR Workshop Proceedings, CEUR-WS.org (2021)
4. Augusto, A., J. Carmona, Verbeek, E.: Advanced Process Discovery Techniques. In: van der Aalst, W.M.P., Carmona, J. (eds.) Process Mining Handbook. LNBIP, vol. 448, pp. 76–107. Springer, Cham (2022)
5. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer, Newyork (2006). https://doi.org/10.1007/978-1-4615-7566-5
6. Camargo, M., Dumas, M., González-Rojas, O.: Learning accurate LSTM models of business processes. In: Hildebrandt, T., van Dongen, B.F., Röglinger, M., Mendling, J. (eds.) BPM 2019. LNCS, vol. 11675, pp. 286–302. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26619-6_19
7. Carmona, J., Dongen, B., Weidlich, M.: Conformance checking: foundations, milestones and challenges. In: van der Aalst, W.M.P., Carmona, J. (eds.) Process Mining Handbook. LNBIP, vol. 448, pp. 155–190. Springer, Cham (2022)
8. Castellanos, M., Salazar, N., Casati, F., Dayal, U., Shan, M.: Predictive business operations management. IJCSE **2**(5/6), 292–301 (2006). https://doi.org/10.1504/IJCSE.2006.014772

9. Ceci, M., Lanotte, P.F., Fumarola, F., Cavallo, D.P., Malerba, D.: Completion time and next activity prediction of processes using sequential pattern mining. In: Džeroski, S., Panov, P., Kocev, D., Todorovski, L. (eds.) DS 2014. LNCS (LNAI), vol. 8777, pp. 49–61. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11812-3_5
10. Chamorro, A.E.M., Resinas, M., Cortés, A.R., Toro, M.: Run-time prediction of business process indicators using evolutionary decision rules. Expert Syst. Appl. **87**, 1–14 (2017). https://doi.org/10.1016/j.eswa.2017.05.069
11. Conforti, R., Fink, S., Manderscheid, J., Röglinger, M.: PRISM - A Predictive Risk Monitoring Approach for Business Processes, pp. 283–400. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45348-4_22
12. Di Francescomarino, C., Dumas, M., Maggi, F.M., Teinemaa, I.: Clustering-based predictive process monitoring. IEEE Trans. Serv. Comput. **12**(6), 896–909 (2019). https://doi.org/10.1109/TSC.2016.2645153
13. Di Francescomarino, C., Ghidini, C., Maggi, F.M., Milani, F.: Predictive process monitoring methods: which one suits me best? In: Weske, M., Montali, M., Weber, I., vom Brocke, J. (eds.) BPM 2018. LNCS, vol. 11080, pp. 462–479. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-98648-7_27
14. Di Francescomarino, C., Ghidini, C., Maggi, F.M., Petrucci, G., Yeshchenko, A.: An Eye into the Future: Leveraging A-priori Knowledge in Predictive Business Process Monitoring, pp. 252–268. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-65000-5_15
15. van Dongen, B.F., de Medeiros, A.K.A., Verbeek, H.M.W., Weijters, A.J.M.M., van der Aalst, W.M.P.: The ProM framework: a new era in process mining tool support. In: Ciardo, G., Darondeau, P. (eds.) ICATPN 2005. LNCS, vol. 3536, pp. 444–454. Springer, Heidelberg (2005). https://doi.org/10.1007/11494744_25
16. Folino, F., Guarascio, M., Pontieri, L.: Discovering context-aware models for predicting business process performances. In: Meersman, R., et al. (eds.) OTM 2012. LNCS, vol. 7565, pp. 287–304. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33606-5_18
17. Folino, F., Guarascio, M., Pontieri, L.: Discovering High-Level Performance Models for Ticket Resolution Processes, pp. 275–282. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-41030-7_18
18. Galanti, R., Coma-Puig, B., de Leoni, M., Carmona, J., Navarin, N.: Explainable predictive process monitoring. In: van Dongen, B.F., Montali, M., Wynn, M.T. (eds.) 2nd International Conference on Process Mining, ICPM 2020, Padua, Italy, 4–9 October 2020, pp. 1–8. IEEE (2020). https://doi.org/10.1109/ICPM49681.2020.00012
19. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press, Cambridge (2016). http://www.deeplearningbook.org
20. Hastie, T., Tibshirani, R., Friedman, J.: Overview of Supervised Learning, pp. 9–41. Springer, New York (2009). https://doi.org/10.1007/978-0-387-84858-7_2
21. Jorbina, K., et al.: Nirdizati: a web-based tool for predictive process monitoring. In: Proceedings of the BPM Demo Track and BPM Dissertation Award co-located with 15th International Conference on Business Process Modeling (BPM 2017), Barcelona, Spain, 13 September 2017 (2017)
22. La Rosa, M., Reijers, H.A., van der Aalst, W.M.P., Dijkman, R.M., Mendling, J., Dumas, M., García-Bañuelos, L.: APROMORE: an advanced process model repository. Expert Syst. Appl. **38**(6), 7029–7040 (2011)

23. 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
24. de Leoni, M., Dees, M., Reulink, L.: Design and evaluation of a process-aware recommender system based on prescriptive analytics. In: van Dongen, B.F., Montali, M., Wynn, M.T. (eds.) 2nd International Conference on Process Mining, ICPM 2020, Padua, Italy, 4–9 October 2020, pp. 9–16. IEEE (2020). https://doi.org/10.1109/ICPM49681.2020.00013
25. Leontjeva, A., Conforti, R., Di Francescomarino, C., Dumas, M., Maggi, F.M.: Complex symbolic sequence encodings for predictive monitoring of business processes. In: Motahari-Nezhad, H.R., Recker, J., Weidlich, M. (eds.) BPM 2015. LNCS, vol. 9253, pp. 297–313. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-23063-4_21
26. Lin, L., Wen, L., Wang, J.: MM-Pred: a deep predictive model for multi-attribute event sequence. In: Berger-Wolf, T.Y., Chawla, N.V. (eds.) Proceedings of the 2019 SIAM International Conference on Data Mining, SDM 2019, Calgary, Alberta, Canada, 2–4 May 2019, pp. 118–126. SIAM (2019). https://doi.org/10.1137/1.9781611975673.14
27. Lundberg, S.M., Lee, S.: A unified approach to interpreting model predictions. In: Guyon, I., et al. (eds.) Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4–9 December 2017, Long Beach, CA, USA, pp. 4765–4774 (2017). https://proceedings.neurips.cc/paper/2017/hash/8a20a8621978632d76c43dfd28b67767-Abstract.html
28. Ly, L.T., Maggi, F.M., Montali, M., Rinderle-Ma, S., van der Aalst, W.M.P.: Compliance monitoring in business processes: Functionalities, application, and tool-support. Inf. Syst. **54**, 209–234 (2015). https://doi.org/10.1016/j.is.2015.02.007
29. Maggi, F.M., Di Francescomarino, C., Dumas, M., Ghidini, C.: Predictive monitoring of business processes. In: Jarke, M., et al. (eds.) CAiSE 2014. LNCS, vol. 8484, pp. 457–472. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-07881-6_31
30. Mannhardt, F.: Responsible process mining. In: van der Aalst, W.M.P., Carmona, J. (eds.) Process Mining Handbook. LNBIP, vol. 448, pp. 373–401. Springer, Cham (2022)
31. Márquez-Chamorro, A.E., Resinas, M., Ruiz-Cortés, A.: Predictive monitoring of business processes: a survey. IEEE Trans. Serv. Comput. **11**(6), 962–977 (2018)
32. Metzger, A., Neubauer, A., Bohn, P., Pohl, K.: Proactive process adaptation using deep learning ensembles. In: Giorgini, P., Weber, B. (eds.) CAiSE 2019. LNCS, vol. 11483, pp. 547–562. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-21290-2_34
33. Pauwels, S., Calders, T.: Bayesian network based predictions of business processes. In: Fahland, D., Ghidini, C., Becker, J., Dumas, M. (eds.) BPM 2020. LNBIP, vol. 392, pp. 159–175. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58638-6_10
34. Polato, M., Sperduti, A., Burattin, A., de Leoni, M.: Data-aware remaining time prediction of business process instances. In: 2014 International Joint Conference on Neural Networks (IJCNN), pp. 816–823, July 2014
35. Polato, M., Sperduti, A., Burattin, A., Leoni, M.: Time and activity sequence prediction of business process instances. Computing **100**(9), 1005–1031 (2018). https://doi.org/10.1007/s00607-018-0593-x

36. Reinkemeyer, L.: Status and future of process mining: from process discovery to process execution. In: van der Aalst, W.M.P., Carmona, J. (eds.) Process Mining Handbook. LNBIP, vol. 448, pp. 405–415. Springer, Cham (2022)
37. Ribeiro, M.T., Singh, S., Guestrin, C.: Why should I trust you?: Explaining the predictions of any classifier. CoRR abs/1602.04938 (2016). http://arxiv.org/abs/1602.04938
38. Rizzi, W., Di Francescomarino, C., Maggi, F.M.: Explainability in predictive process monitoring: when understanding helps improving. In: Fahland, D., Ghidini, C., Becker, J., Dumas, M. (eds.) BPM 2020. LNBIP, vol. 392, pp. 141–158. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58638-6_9
39. Rizzi, W., Simonetto, L., Di Francescomarino, C., Ghidini, C., Kasekamp, T., Maggi, F.M.: Nirdizati 2.0: new features and redesigned backend. In: Depaire, B., et al. (eds.) Proceedings of the Dissertation Award, Doctoral Consortium, and Demonstration Track at BPM 2019 co-located with 17th International Conference on Business Process Management, BPM 2019, Vienna, Austria, 1–6 September 2019, CEUR Workshop Proceedings, vol. 2420, pp. 154–158. CEUR-WS.org (2019)
40. Rogge-Solti, A., Weske, M.: Prediction of remaining service execution time using stochastic petri nets with arbitrary firing delays. In: Basu, S., Pautasso, C., Zhang, L., Fu, X. (eds.) ICSOC 2013. LNCS, vol. 8274, pp. 389–403. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-45005-1_27
41. Rogge-Solti, A., Weske, M.: Prediction of business process durations using non-Markovian stochastic petri nets. Inf. Syst. **54**(Suppl C), 1–14 (2015)
42. Senderovich, A., Di Francescomarino, C., Maggi, F.M.: From knowledge-driven to data-driven inter-case feature encoding in predictive process monitoring. Inf. Syst. **84**, 255–264 (2019). https://doi.org/10.1016/j.is.2019.01.007
43. Senderovich, A., Weidlich, M., Gal, A., Mandelbaum, A.: Queue mining for delay prediction in multi-class service processes. Inf. Syst. **53**(C), 278–295 (2015). https://doi.org/10.1016/j.is.2015.03.010
44. Sindhgatta, R., Moreira, C., Ouyang, C., Barros, A.: Exploring interpretable predictive models for business processes. In: Fahland, D., Ghidini, C., Becker, J., Dumas, M. (eds.) BPM 2020. LNCS, vol. 12168, pp. 257–272. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58666-9_15
45. Tax, N., Verenich, I., La Rosa, M., Dumas, M.: Predictive business process monitoring with LSTM neural networks. In: Dubois, E., Pohl, K. (eds.) CAiSE 2017. LNCS, vol. 10253, pp. 477–492. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-59536-8_30
46. Teinemaa, I., Dumas, M., Rosa, M.L., Maggi, F.M.: Outcome-oriented predictive process monitoring: review and benchmark. ACM Trans. Knowl. Discov. Data **13**(2), 1–57 (2019). https://doi.org/10.1145/3301300
47. Teinemaa, I., Tax, N., de Leoni, M., Dumas, M., Maggi, F.M.: Alarm-based prescriptive process monitoring. In: Weske, M., Montali, M., Weber, I., vom Brocke, J. (eds.) BPM 2018. LNBIP, vol. 329, pp. 91–107. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-98651-7_6
48. Tu, T.B.H., Song, M.: Analysis and prediction cost of manufacturing process based on process mining. In: 2016 International Conference on Industrial Engineering, Management Science and Application (ICIMSA), pp. 1–5, May 2016
49. Verenich, I., et al.: Predictive process monitoring in Apromore. In: Mendling, J., Mouratidis, H. (eds.) CAiSE 2018. LNBIP, vol. 317, pp. 244–253. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-92901-9_21

50. Weinzierl, S., Dunzer, S., Zilker, S., Matzner, M.: Prescriptive business process monitoring for recommending next best actions. In: Fahland, D., Ghidini, C., Becker, J., Dumas, M. (eds.) BPM 2020. LNBIP, vol. 392, pp. 193–209. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58638-6_12
51. Weinzierl, S., Zilker, S., Brunk, J., Revoredo, K., Matzner, M., Becker, J.: XNAP: making LSTM-based next activity predictions explainable by using LRP. In: Del Río Ortega, A., Leopold, H., Santoro, F.M. (eds.) BPM 2020. LNBIP, vol. 397, pp. 129–141. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-66498-5_10
52. Weytjens, H., De Weerdt, J.: Process outcome prediction: CNN vs. LSTM (with Attention). In: Del Río Ortega, A., Leopold, H., Santoro, F.M. (eds.) BPM 2020. LNBIP, vol. 397, pp. 321–333. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-66498-5_24