TAILORING MACHINE LEARNING FOR PROCESS MINING

Paolo Ceravolo
Computer Science Department
Università degli Studi di Milano, Italy
paolo.ceravolo@unimi.it

Sylvio Barbon Junior
Department of Engineering and Architecture
University of Trieste, Italy
sylvio.barbonjunior@units.it

Ernesto Damiani
Center for Cyber-Physical Systems
Khalifa University, Abu Dhabi, UAE
ernesto.damiani@ku.ac.ae

Wil van der Aalst
Process and Data Science
RWTH Aachen University
wvdaalst@pads.rwth-aachen.de

ABSTRACT

Machine learning models are routinely integrated into *process mining* pipelines to carry out tasks like data transformation, noise reduction, anomaly detection, classification, and prediction. Often, the design of such models is based on some *ad-hoc* assumptions about the corresponding data distributions, which are not necessarily in accordance with the *non-parametric* distributions typically observed with process data. Moreover, the learning procedure they follow ignores the constraints *concurrency* imposes to process data. Data *encoding* is a key element to smooth the mismatch between these assumptions but its potential is poorly exploited. In this paper, we argue that a deeper insight into the issues raised by training machine learning models with process data is crucial to ground a sound integration of process mining and machine learning. Our analysis of such issues is aimed at laying the foundation for a methodology aimed at correctly aligning machine learning with process mining requirements and stimulating the research to elaborate in this direction.

1 Introduction

Process Mining (PM) is a consolidated discipline grounded on *data mining* and *business process management*. The exploitation of traditional PM tasks (*discovery*, *conformance checking*, and *enhancement*) is today a reality in many organizations [1, 2]. In the last decade, a wave of new results in *artificial intelligence* has triggered the interest of the PM research community in using supervised or unsupervised Machine Learning (ML) techniques for gaining insight into business processes and providing advice on how to improve their inefficiencies.

In today's practice, ML models are routinely integrated into PM data pipelines [3] to carry out tasks like data transformation, noise reduction, anomaly detection, classification, and prediction. For example, ML is playing a key role in the interface between PM and sensor platforms. Advances in sensing technologies have made it possible to deploy distributed monitoring platforms capable of detecting fine-grained events. The granularity gap between these events and the activities considered by classic PM analysis has often been bridged using ML models [4, 5] that compute virtual activity logs, a problem which is also known as *log lifting* [6]. ML has been proposed as a key technology to *strengthen* existing techniques, for example, using trace clustering to reduce the diversity that a process discovery algorithm must handle in analyzing an event log [7, 8, 9, 10], to simplify the discovered models [11, 12, 13], or to

support real-time analysis on event streams [14, 15, 16]. ML is adopted to apply predictive models to the executing cases of a process. This research area, known as *predictive process monitoring*, exploits event log data to foresee future events, remaining time, or the outcome of cases, in support of decision making [17, 18, 19]. Root cause analysis [20] and data explainability [21] are other tasks that can be applied to event log data using ML techniques, in order to improve our understanding of a business process. ML models have also been used in addition to (or in lieu of) classic linear programming [22] to *optimize* business processes' resource consumption and to provide insights to process *re-design* [23]. Computational support for PM tends to converge with the one available for ML models also from the technology standpoint [24, 25]. This makes their integration seem straightforward.

In fact, it is not. When PM tasks are mapped to ML tasks, business process-specific assumptions should drive the construction of training functions and hyperparameters selection. Some of these assumptions stem from the very nature of human social systems. For example, it is well known that process variants are shaped by *non-parametric* distributions [26]. Quite the contrary, data normality is beneficial for many ML models, moreover, if the data distribution is skewed, ML models may be biased toward a particular outcome. In addition, the ML view on event log data is often oversimplified. The correct encoding of the procedural nature of event log traces is challenging. Often, the sequence of executed events is simply captured by a prefix of fixed length. Even more problematic is encoding *concurrency* and the *interactions* constraining the events in the business process. Encoding event log data into a feature space compatible with ML algorithms is a critical design choice in other concerns [27]. It impacts the *sample complexity*, the *data distribution*, and the relevance of the features to put under analysis, for example, to detect *concept drift* or to support *zero-shot learning* [28].

Today, much of the research on integrating ML with PM focuses on developing ML models to attain high performance in specific business process management scenarios. Less attention has been paid to designing a general methodology to select and adapt ML models based on the nature of the PM problem, taking into account the specific properties of the process data. We argue that, when using ML models in PM pipelines, it is important to prevent any *mismatch* between the assumptions on input data distributions underlying the ML models and the statistics of the event logs used to feed them [29]. Arbitrarily selecting algorithms leads to unfair evaluation and sub-optimal solutions. For example, a given model cannot be compared with another if their implementations consider different feature spaces [30]. It is also important to make sure that ML models are exposed to process-specific information, such as the processes' control-flow constraints. In this paper, we attempt to identify some of the causes of this mismatch and suggest how to remove them, with the aim of fostering research on a sound methodology to address the integration between PM and ML.

We believe that an effort on these aspects must be jointly made by the PM and Artificial Intelligence research communities. This call to collaboration is valid in general but particularly in business process management, where data analysis has to leave the safe harbor of experimental science to sail into the open sea of decision science. In this paper, we discuss the challenges in a specific direction, i.e., from PM to ML. More specifically, in Section 2 we discuss the issues leading to the PM-to-ML mismatch. In Section 3 we introduce some basic PM notions. In Section 4 we link them to ML principles. Section 5 clarifies the discussion by presenting a couple of samples. Section 6 proposes research lines for advancing in the direction of a general methodology that integrates ML models into PM pipelines. Section 7 closes the paper.

2 The Issues Landscape

An important problem underlying our discussion is how to take into account process data specificity in ML model selection and (hyper-) parameter tuning. Of course, processing event logs poses all the usual challenges of data pre-processing and preparation. We will not discuss standard data pre-processing techniques such as outlier removal [31, 32], noise filtering [33, 34], and missing entries recovery [35] as they can be tackled by current statistical techniques. Rather, we will focus on issues that are specific to process data, including their statistical distribution and event concurrency. Indeed, careless assumptions on the encoding of input data may result in biased models with reduced generalization capability.

2.1 Data Distribution

When choosing an ML model for a PM task, it is tempting to assume that the process data fed to the model will follow a normal distribution. Indeed, data normality is beneficial for many types of ML models. Models like Gaussian, naive Bayes, logistic and linear regression explicitly rely on the assumption that the data distribution is bi-variate or multivariate normal. Many phenomena of interest for business process analysis, such as the duration of some activities,

are known to follow normal or log-normal distributions [1]. For other PM data, however, assuming normality is not always advisable. For example, process variants are specific activity sequences that occur through a process from start to end. Variants' occurrence in an activity log is typically following a *non-parametric* trend that complies with the *Pareto principle* [26]. A normal distribution cannot always be assured also for the pairwise dependency relationship between activities, a key statistical information exploited by process discovery algorithms [36]. Indeed, in this case, the normality assumption has been verified for some event logs, including some popular benchmarks we will discuss in Section 5 (the "Road traffic fines" [37] and "Receipt phase of an environmental permit application process" [38]). However, the normality of dependencies in less regular, "spaghetti" like, processes is not observed, as in the "BPI Challenge 2015 Municipality 1"[39]. There are reasons to believe that dependencies in loosely specified logs may follow some power-law trend as well, and require careful parameter fitting in statistical analysis. Imbalanced data sets or non-stationary environments may also cause serious difficulties. For example, if the training data is skewed towards a particular class or outcome, the model may be more likely to predict that class or outcome even when it is not the most likely one. Independent component analysis [40] provides ways to reveal Gaussianity and non-Gaussianity. Of course, non-normal distributions can be transformed to normal ones using Box-Cox transformations [41], and unbalanced data sets can be balanced [42, 43] but, as we shall see, such data transformations should be applied with caution, as they have consequences on the performance of the models.

In any case, PM data regarding distributions of variants cannot be expected to always follow a Gaussian behavior, demanding estimation techniques to sample from the sequential process underlying log generation. *Markov Chain Monte Carlo* (MCMC) techniques are sometimes used for sampling from an unknown probability distribution (for instance, the distribution of variants) by using data to construct a Markov chain whose equilibrium distribution approximates the unknown one. MCMC techniques can be combined with Kalman filtering[44] to control uncertainty. Of course, an explicit estimate of the data distribution may not even be necessary. Some ML models work well also in the case of non-normally distributed data. Simple yet effective ML models like decision trees and random forests do not assume any normality and work reasonably well on raw event data. Also, linear regression is statistically effective if the model errors are Gaussian, an assumption less stringent for process data than the normality of the entire data set. Kernel methods, e.g., Gaussian processes and support vector machines, provide flexible models that are practical to work with but require proper hyperparameter variables to fit the data.

2.2 Concurrency

Another key attention point is concurrency. How to use ML to predict the behavior of highly concurrent systems and processes is still an open problem, and the research done in the AI community has only scratched its surface (see [45] for a recent review). Most ML approaches view event logs as merely sequential data [46], rather than sequential manifestations of a concurrent system. This may lead to under-sampling the log space and to insufficient training to handle apparently out-of-order event sequences [47]. To address this issue, it is important to provide ML models with control-flow information about the iterative or concurrent execution of tasks as additional context alongside event logs. One approach that has been explored is the use of Bi-directional Long-Short Term Memory (BiLSTM) architectures. Thapa et al. [48] leveraged BiLSTM to detect concurrent human activities in a smart home environment. Additionally, Thapa et al. [49] adapted the LSTM algorithm into a synchronous algorithm called sync-LSTM, enabling the model to handle multiple parallel input sequences and generate multiple synchronized output sequences. The field of predicting the behavior of highly concurrent systems using ML is rapidly evolving, as indicated by the recent survey conducted by Neu et al. [50]. Researchers are actively exploring new techniques and methodologies to improve the understanding and prediction of concurrency in various domains.

2.3 Non-stationary Behaviour

Even when the process data distributions can be fitted precisely, running processes, especially the ones involving resources that learn and age like people and equipment, change over time. This gives rise to *non-stationary* behavior. This problem is a critical one since ML models' learning capacity decreases under non-stationary conditions [16]. Concept drift detection techniques are therefore required. In traditional data mining applications, *concept drift* is identified when, at two separate points in time, a concept, i.e., the relation between a data instance and its associated class, changes [51]. In PM, many aspects of drift should be carefully monitored, including the appropriateness of the event trace with respect to the model, the dependency relationship between activities, and the interdependence between the activities and the available resources or cycle time. Each aspect should be appropriately encoded and monitored using statistical analysis [52].

---

[1]See, for instance, the "lunch break" duration distributions at `https://www.statista.com/statistics/995991/distribution-of-lunch-breaks-by-length-in-europe/`

2.4 Zero-shot Learning

A related topic is using ML models to identify solutions never observed during training, the so-called *zero-shot learning* [53]. There are several zero-shot learning approaches, but a commonality is that unstructured auxiliary information is encoded during the training process instead of using explicit labels. The training process aims to learn to connect new input elements to encodings that have the greatest similarity in terms of auxiliary information. In this way, the system can propose an outcome never observed during the training stage. Zero-shot is relevant in PM when the availability of labeled process data is limited, as the process may be recently developed, unused, or its outcomes inaccessible. In these situations, relying on historical observations to guide learning tasks is insufficient or erroneous. This scarcity has drawn the attention of the PM community to *contrastive learning*, a manner of unsupervised learning that learns representations by contrasting positive and negative pairs. Graph-related contrastive learning methods apply this notion to all types of graph data. Some popular unsupervised representation learning methods imply the idea of contrastive learning. For instance, DeepWalk [54] and node2vec [55] generate Markov chains of nodes based on random walking on graphs, forcing the neighboring nodes of a graph to have similar representations. More recent proposals such as DGI [56] and InfoGraph [57] combine contrastive learning with ordinary supervised training to maximize the mutual information of node and graph levels.

Much work is also being done on *generative* engines for logs based on likelihood-based models, like auto-encoders and Generative Adversarial Networks (GANs)[58]. However, ordinary GANs show some limitations when applied to generate "clean" process data, where low confidence variants are due to failures of the monitoring context rather than to adversarial constructions [59]. In addition, the GANs' *objective function*, i.e. the difference between the generated and the original distribution of traces in the event log, is not always suitable for evaluating the quality of the generated process variants, and even less for comparing different generators. Performance measures should be used instead, and the trained algorithms should be able to provide an answer with different information details, for example, predicting a performance result knowing or not the availability of resources currently in use.

2.5 Data Encoding

Supervised ML algorithms are trained on collections of examples, each encoded as a vector in a multidimensional feature space. An appropriate encoding method can reduce the sample complexity and reduce the space or time complexity of the model [27]. In PM, even more, than selecting individual features, it is important to capture the interconnections between the different process dimensions. The event logs analyzed in PM contain information from several complementary dimensions, such as event data, executing traces, resource consumption, and cycle time. Each event can be described as a multidimensional object, but its value for the process execution lies in the interdependence with the other events composing the process case instance, the resources available in the system, and the temporal limits constraining the case, which in turn depend on the other cases executed, executing, or to be executed in the system. Therefore, capturing the constraints due to the alternative, optional or mandatory dependency between events is crucial in PM. Encoding methods should also identify features subject to *concept drift*. Extracting insights from this type of functional data is not straightforward; covariance control[60] is needed to take into account the hidden relationships between the different dimensions.

Despite all this, little effort has been spent by the PM community to study the impact of encoding methods on the performance of PM pipelines. Only a few comparative studies are available [61, 27, 62, 63]. Basic techniques, such as the one-hot encoding scheme [64] or frequency-based encoding [9], are often adopted. For numerical attributes, general statistics have been used, such as average, maximum, minimum, and sum [18]. The k-gram encoding schema [8] is also quite popular. Each activity in the trace is represented as the sequence of k activities executed to reach it. As an alternative, arrays encoding traces as the frequency of their activities at each position have been proposed [65]. These encoding techniques can incorporate some control-flow information, but cannot fully account for concurrency. To better capture dependency between activities, techniques borrowed from other domains have been proposed, including text mining [66, 67] and graph embedding [68, 69]. Graph embedding methods emerged from the necessity of representing graphs as low-dimensional vectors to be exploited by downstream ML models. These methods rely themselves on ML models (usually, supervised learners) to compute highly informative but low-dimensional vectors of fixed length [70]. When applied to event logs encoding, such methods outperform the others, at the cost of higher time complexity and loss of transparency, as the resulting vectors are organized in a latent space losing any reference to the event log attributes or their statistical properties [71]. In any case, the representation of the control-flow is purely sequential and concurrency is not captured by these methods too. Recently, emerging attention on techniques for encoding control-flow information into a feature space is observed, for example by representing the degree of parallelism or optionality of activities [72, 73]. Another trend is aimed at constructing multi-perspective views of traces, representing the data-flow and control-flow into the same encoding [74, 75]. However, the application of these methods is still limited.

4

Generally speaking, the encoding procedures used to map PM data to ML models are not documented enough in the PM literature. Sometimes, the feature space selected is not explicitly presented, the steps followed to encode data are not well specified, or the adopted code is not shared. *Ablation studies*, removing parts of the data representation and studying the removal's impact on performance, are still the exception rather than the norm. We argue that the formalization of the encoding procedure allows explaining this key design choice to be justified by the specific analytical goals and the assumptions applying to the algorithms considered. We will propose such a formalization in Section 4.

3   Basic Notions in PM

To make this paper self-contained, in this section we recall some of the basic concepts of PM. An *event log* is a collection of *events* generated in a temporal sequence and stored as *tuples*, i.e., recorded values from a set of *attributes*. Events are aggregated by *case*, i.e., the end-to-end execution of a business process. For the sake of classification, all cases following the same *trace*, i.e., performing the same sequence of business process activities, can be considered equal as they belong to the same process *variant*.

Definition 1 (Event, Attribute) *Let Σ be the event universe, i.e., the set of all possible event identifiers; Σ^* denotes the set of all finite sequences over Σ. Events have various attributes, such as* TIMESTAMP, ACTIVITY, RESOURCE, ASSOCIATED COST, *and others. Let $\mathcal{AN}$ be the set of attribute names. For any event $e \in \Sigma$ and attribute $A \in \mathcal{AN}$, the function $\#_A(e)$ returns the value of the attribute A for event e.*

The set of possible values of each attribute is restricted to a domain. For example, $\#_{\text{ACTIVITY}} : \Sigma \to \mathcal{A}$, where $\mathcal{A}$ is the set of the legal activities of a business process, e.g. $\mathcal{A} = \{a, b, c, d, e\}$. If e does not contain the attribute value for some $A \in \mathcal{AN}$, then $\#_A(e) = \bot$. It follows that an event can also be viewed as a tuple of attribute-value pairs $e = (\mathcal{A}_1, ..., \mathcal{A}_m)$, where m is the cardinality of $\mathcal{AN}$.

Definition 2 (Sequence, Sub-sequence) *In a sequence of events $\sigma \in \Sigma^*$, each event appears only once and time is non-decreasing, i.e., for $1 \le i \le j \le |\sigma| : \#_{\text{TIMESTAMP}}(e_i) \le \#_{\text{TIMESTAMP}}(e_j)$. Thus $\langle e_1, e_2, e_3 \rangle$ denotes three subsequent events. A sequence can also be denoted as a function generating the corresponding event for each position in the sequence: $\sigma(i \to n) \mapsto \langle e_i, ..., e_n \rangle$, with e_n the last event of a sequence. In this way, we can define a sub-sequence as a sequence $\sigma(i \to j)$ where $0 \le i < j < n$.*

Definition 3 (Case, Event Log) *Let $\mathcal{C}$ be the case universe, that is, the set of all possible identifiers of a business case execution. $\mathcal{C}$ is the domain of an attribute $\#_{\text{CASE}} \in \mathcal{AN}$. We denote a case $c \in \mathcal{C}$ as $\langle e_1, e_2, e_3 \rangle_c$, meaning that all events are in a sequence and share the same case. For a case $\langle e_1, e_2, e_3 \rangle_c$ we have $\#_{\text{CASE}}(e_1) = \#_{\text{CASE}}(e_2) = \#_{\text{CASE}}(e_3) = c$. An event log L is a set of cases $L \subseteq \Sigma^*$ where each event appears only once in the log, i.e., for any two different cases, the intersection of their events is empty. When the case identifier is not used as a grouping attribute, an event log $\hat{L}$ can be simply viewed as a set of events, thus $\hat{L} \subseteq \Sigma$.*

Definition 4 (Variant, Event Log) *The cases c_1 and c_2 follow the same variant if $\langle e_1, e_2, e_3 \rangle_{c_1}$ and $\langle e_4, e_5, e_6 \rangle_{c_2}$ have the same sequence of activities, e.g. $\#_{\text{ACTIVITY}}(e_1) = \#_{\text{ACTIVITY}}(e_4) = a$, $\#_{\text{ACTIVITY}}(e_2) = \#_{\text{ACTIVITY}}(e_5) = b$, $\#_{\text{ACTIVITY}}(e_3) = \#_{\text{ACTIVITY}}(e_6) = a$. We call this sequence a trace. This implies an event log can also be viewed as a multi-set of traces. We denote an event log as a multi-set by writing $\overline{L} = [\langle a, b, c \rangle^3, \langle a, b, a \rangle^{11}, \langle a, c, b, a \rangle^{20}]$. The superscript number of a trace details the number of cases following this variant. For example, $\langle a, b, a \rangle^{11}$ means we have a variant with 11 cases following the trace $\langle a, b, a \rangle$.*

4   A Formalisation of PM Data Encoding

Despite the variety of encoding methods discussed in Section 2, we argue that available approaches fail to capture key process-level information such as the interplay between cases, or between activity execution and availability of resources. Most of the encoding methods in use today focus on the *control-flow*, according to an *inter-case* view. Methods focusing on the *intra-case* view have been proposed but are rarely applied [76]. Similarly, proposals for encoding the *data-flow* [77] are available in the literature, but never adopted in comparative studies or surveys. Another recent trend is stressing the need of capturing constraints connected to concurrency [72, 73]. In this section, we discuss in detail how PM data is encoded to suit ML models' training procedures. For the sake of space, we limit our discussion to supervised learning, probably the most widely applied ML approach. Generally speaking, supervised techniques train models to compute functions $f : \mathbb{R}^d \to \mathbb{R}^{d'}$ where the input is a d-dimensional vector $\mathbf{x}$ and the output is a d'-dimensional vector $\mathbf{y}$. Each dimension is a measurable piece of data, a.k.a feature or attribute. For popular ML tasks, the output is mono-dimensional. In regression, the output is a real-valued scalar value, while in classification,

the output is a natural number indexing a "class". However, nothing prevents having multidimensional vectors in output. In structured learning, input and output may be a structure like a block matrix, divided into sub-matrices to represent algebraic entities such as graphs, tensors, etc. The training process to approximate f requires a set of examples $\{(\mathbf{x}_1, \mathbf{y}_1), ..., (\mathbf{x}_n, \mathbf{y}_n)\}$ where inputs and outputs are paired. We can then define this training set as an example matrix $\mathbf{X} := [\mathbf{x}_1, ..., \mathbf{x}_n]^\top \in \mathbb{R}^{n \times d}$ and a label matrix $\mathbf{Y} := [\mathbf{y}_1, ..., \mathbf{y}_n]^\top \in \mathbb{R}^{n \times d'}$, given by the number n of vectors and the number d of dimensions in the vector space.

In their original format, PM log entries do not belong to a vector space. This is because the events in an event log are grouped by case and this grouping is essential to keep a connection with business process execution.

Our goal here is to formalize the procedure to encode the cases into vectors in a way that can be used as a template to describe the specific encoding chosen for a PM application. Our starting point is $\hat{L} \subseteq \Sigma$, a log view as a set of event identifiers. This representation can be mapped into a vector space $\mathbf{X}$ by applying a suitable *transformation function* grouping event by case and returning vectors of size equal to or less than the event size.

Definition 5 (Encoding function) *Given an event log $\hat{L} \subseteq \Sigma$, an* encoding function *$\Gamma : \Sigma \to \mathbf{X}^{n \times d}$ represents $\hat{L}$ in the vector space $\mathbf{X}$. The encoding function Γ is valid if it defines a transformation where two elements of Σ, e_i and e_j are aggregated on the same element $\mathbf{x} \in \mathbb{R}^d$ if $\#_{\text{CASE}}(e_i) = \#_{\text{CASE}}(e_j)$, with $n \leq |\mathcal{C}|$, i.e. the vectors in $\mathbf{X}$ are a subset of the cases in $\mathcal{C}$.*

We propose a canonical representation of Γ as a composition of a *filtering function* π, a *dimensioning function* ρ, a *grouping function* η, and a *valuation function* ν, i.e., $\Gamma = \nu \circ \eta \circ \rho \circ \pi$. One or more of these components can implement the identity function with null effects.

In particular, $\pi : \Sigma \to \Sigma_\alpha$ imposes a condition on the events' attributes or the attributes' values, $\forall e \in \hat{L} \wedge \text{A} \in \mathcal{AN} : P(\#_\text{A}(e))$, where P is a predicate, thus $|\Sigma_\alpha| \leq |\Sigma|$. For example, filtering the events by their timestamp $\forall e \in \hat{L} : \texttt{YYYY-MM-DD} \geq \#_{\text{TIMESTAMP}}(e) \leq \texttt{YYYY-MM-DD}$. The function $\rho : \Sigma_\alpha \to D$ defines the dimensions of the vector space, creating new dimensions based on a range of values in the original dimensions or, less commonly, grouping multiple dimensions into a single one. Often, the set D is the union of multiple attribute domains, i.e. $D = \mathcal{A}_{k=1} \cup \mathcal{A}_{k=2} \cup \cdots \mathcal{A}_{k=l}$. The function $\eta : \Sigma_\alpha \to \mathbf{X}_\alpha^{n \times d}$, with $d = |D|$, assigns to $\mathbf{X}_\alpha$ the values of the attributes in e and groups events by case so that $\forall \mathbf{x} \forall A_k : \mathbf{x}_{i,j} = \#_{\text{A}_k}(e) \iff \#_{\text{A}_k}(e) = D_j \wedge \#_{\text{CASE}}(e) = c_i$. The number of elements in the vector space equals the number of cases to include in the example matrix, thus $n \leq |\mathcal{C}|$. Because the sets Σ_α and D can be view as columnar matrices $M_{\Sigma_\alpha}^{n \times 1}$ and $M_D^{d \times 1}$, the size of $\mathbf{X}_\alpha$ is equal to $M_{\Sigma_\alpha} \times M_D^\top$, i.e. the set of events we selected with π is multiplied by the dimensions we identified with ρ. It is worth mentioning that, when grouping is applied, each vector component becomes an array of attribute values rather than a single value. The function ν aims at transforming these arrays of attribute values into real-valued scalar values. We define $\nu : \mathbf{X}_\alpha^{n \times d} \to \mathbf{X}^{n \times d}$ to clarify the components of the two matrices are valuated differently.
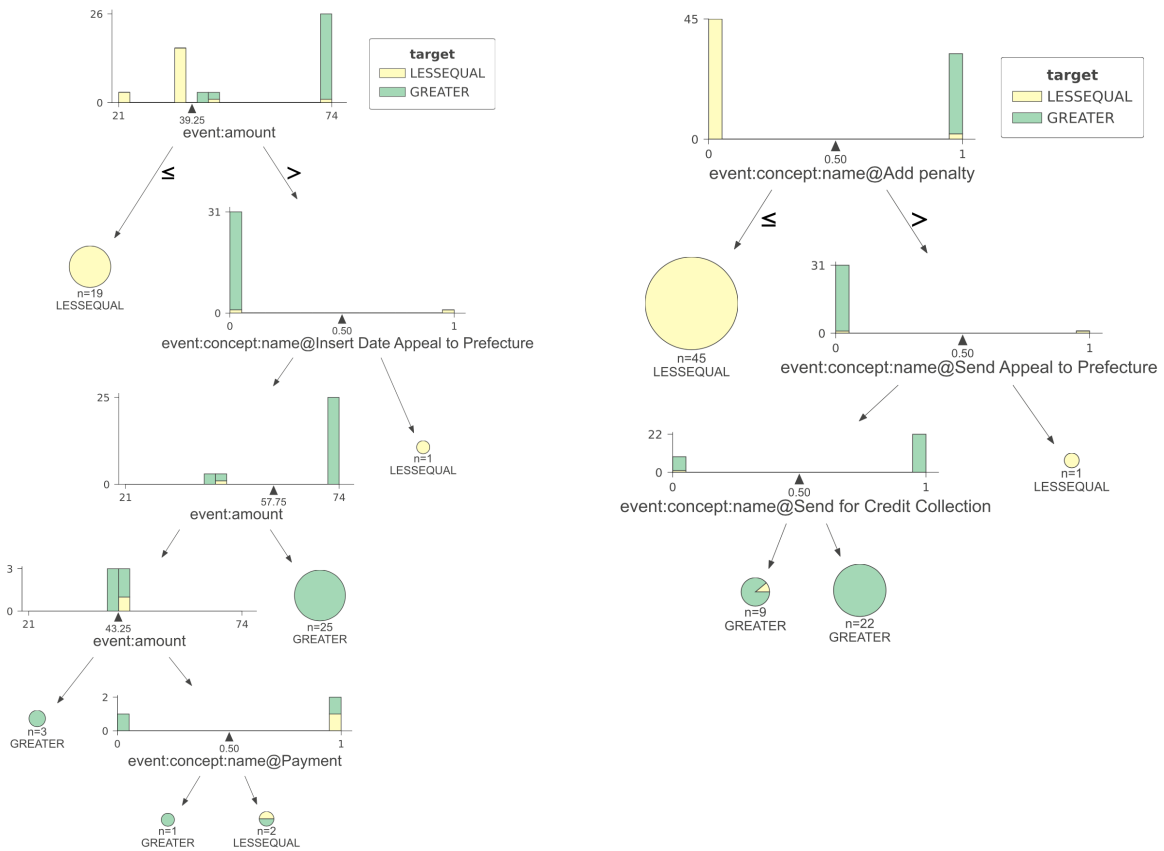
For example, the basic *one-hot* encoding schema corresponds to a null π, a ρ with $D = \bigcup_{k=1}^{l} \mathcal{A}_k$, an η for grouping the events of a same case, and a $\nu : \mathbf{X}_\alpha^{n \times d} \to \{0, 1\}^{n \times d}$, returning $\mathbf{x}_{i,j} = 1$ if at least a value $\#_{\text{A}_k}(e) = D_j$ is observed for the case $\#_{\text{CASE}}(e) = c_i$, and 0 if not. The popular *activity profile* schema [7] encodes an event log into a vector of activity values by simply counting all events of a case that include that activity. The encoding function maps the events in $\hat{L}$ into $\mathbf{X}$ by executing the four canonical transformations as follows. First, it verifies to consider only events associated with activity values $\forall e \in \Sigma : \#_{\text{ACTIVITY}}(e) \neq \bot$. Then it defines the dimensions of $\mathbf{X}$ with ρ so that $D = \mathcal{A}$, where $\mathcal{A}$ is the set of legal business process activities. Third, it aggregates the data by case with η. Finally, it performs the evaluation with ν, assigning the count of the components in $\mathbf{x}_{i,j}$ for each case c_i. For instance, the log $\overline{L} = [\langle a, b, c\rangle^3, \langle a, b, a\rangle^{11}, \langle a, c, b, a\rangle^{20}]$, is transformed in the first matrix in 1 with π, in the second matrix with ρ, in the third matrix in with η, to finally get the fourth matrix in 1 with ν.

$$
\begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \\ e_5 \\ ... \end{bmatrix}
\begin{bmatrix} a \\ b \\ c \end{bmatrix}
\begin{bmatrix} a & b & c \\ a & b & c \\ a & b & c \\ [a,a] & b & \bot \\ [a,a] & b & \bot \\ ... \end{bmatrix}
\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 2 & 1 & 0 \\ 2 & 1 & 0 \\ ... \end{bmatrix}
\tag{1}
$$

We believe that if the PM community would get used to clarifying the definition of the following functions when defining an encoding procedure, the literature will benefit in terms of the comparability of the results. For example, a *data-fow* approach will require clarifying the contribution of the different dimensions in encoding cases. An *intra-case* approach will require modifying the η function to encode multiple cases into a single vector.

Cases	Number of Variants	Coverage of Cases
56482	1	37,6%
102853	2	68,4%
132758	4	88,3%
142926	7	95,0%
148887	17	99,0%
150270	131	99,9%
150370	231	100,0%

Table 1: *Managing Road Traffic Fines* Event Log

(a) Unbalanced event log   (b) Balanced event log

Figure 1: Two *decision trees* generated from our sample event log. In 1a the data in input conforms to the case distribution observed in the event log. As a consequence, the most frequent variants take the lion's share and the numeric feature *amount* decides multiple split points. In 1b data is balanced oversampling those variants with low occurrence. The split points in the tree use categorical features only. Decision tree is an example of an algorithm significantly affected by uncritical training using the case distribution of event logs.
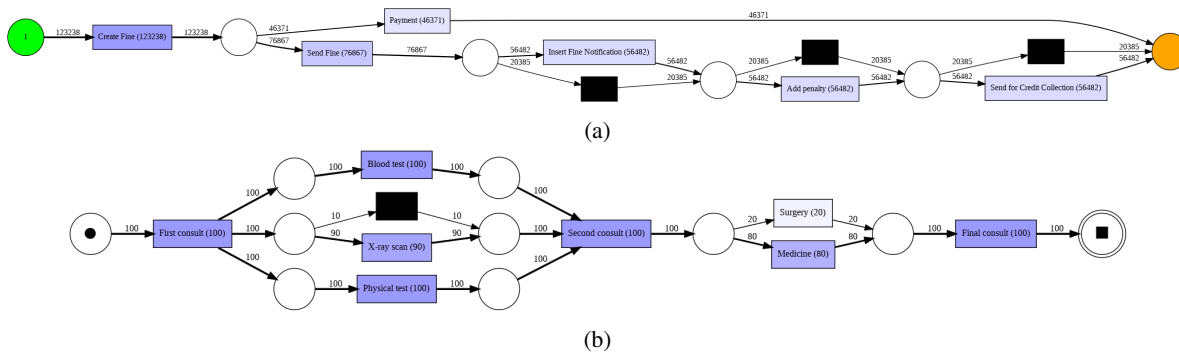
(a)

(b)

Figure 2: (a) The Heuristic Miner Algorithm [79] was used to discover a model from the "Road Traffic Fines" [37] event log. The discovered model specifies alternative routes that can be followed to complete the process. In particular, executing `Payment` or `Send Fine` implies the following alternative paths. (b) The Heuristic Miner Algorithm [79] is used to discover a model from the "Artificial Patient Treatment" [80] event log. The discovered model specifies that `Blood test`, `X-ray scan`, and `Physical test` are executed in parallel. Any order can be followed in executing these activities.

5   Illustrative Examples

We will now use two examples to illustrate the concepts introduced above.

The first example refers to the real-live event log of road traffic fines [37]. The events captured in the event log include creating a fine notice, recording the penalty amount, verifying if the payment is received, registering an appeal to the prefecture, and others. The reader interested in more details is referred to [78]. As illustrated in Table 1, the occurrence of trace variants follows a Pareto distribution with only 4 variants covering more than 88% of the recorded cases and with 100 variants that have a single occurrence. The most occurring variant is $\langle Create\ Fine,\ Send\ Fine,\ Insert\ Fine\ Notification,\ Add\ Penalty,\ Send\ for\ Credit\ Collection \rangle^{56482}$, the second is $\langle Create\ Fine,\ Payment \rangle^{46371}$, the third is $\langle Create\ Fine,\ Send\ Fine \rangle^{20385}$, and so on.

Let us now try to develop predictive analytics on this event log. For example, we could ask ourselves why certain cases exhibit a duration that is significantly longer than others. To study the problem, we are interested in searching for patterns correlated to long duration. Using encoding, we can represent the cases in the event log as vectors composed of categorical data, such as the executed activities, and of numerical data such as the number of penalties and the trace duration[2]. A decision tree can then be used to highlight the factors influencing case duration. We express it as a simple binary problem: being below or above a threshold of 200 days. Figure 1 illustrates the results we obtain. Figure 1a presents a decision tree conforming to the case distribution observed in the event log. The entire set of cases in L is encoded in $\mathcal{X}$. As a consequence, the most frequent variants take the lion's share of the examples used to train the decision tree. Figure 1b presents the decision tree obtained by balancing the case distribution among variants, oversampling those variants with low occurrence. This is, for example, achieved by creating $\mathcal{X}$ taking an equal number of occurrences to the traces in L.

Because the split points of the tree are chosen to best separate examples into two groups with minimum mixing, the cases with low occurrence tend to be ignored. Indeed, the tree in Figure 1a relies on the numeric feature *amount* to decide on multiple split points. On the contrary, the tree in Figure 1b defines the split points using categorical features only. This is due to the fact that the variants not associated with a penalty amount were quite rare, and by increasing their representation for balancing the data set we prevented the algorithm to use the penalty amount as a discrimination feature.

It is important to note that, in general, we cannot say if proactive balancing is better than using data as they are, and even which is the correct balancing factor to be applied. The strategy to be preferred strongly depends on our goal. If we want to analyse an event log in order to identify procedures that can be automated and learn the decision rule to be used, our interest is in the frequent behaviour. The real distribution of the event log, or even a distribution pruned from rare examples [26], must address the learning procedure we adopt. If our goal is anomaly detection [81] or root cause analysis [82] rare examples have to be represented.

---

[2]The methods used for encoding the event log in a vector space are available in the PM4PY library `https://pm4py.fit.fraunhofer.de/documentation#decision-trees`

Our next example is related to the need of capturing concurrency (Section 2). While cases included in an event log are described as sequences of activities, the behaviour they describe should be interpreted differently based on the model that generated them. To capture control-flow behaviour, one needs to encode the dependency relationships in event logs. By executing the Heuristic Miner algorithm [79] on the "Road Traffic Fines" [37] event log, we observe alternative paths can be followed to complete the process. If a case includes the execution of the `Payment` activity, it will not include `Send Fine` and the following activities. The same algorithm applied to the "Artificial Patient Treatment" [80] will reveal the concurrent execution of the `Blood test`, `X-ray scan`, and `Physical test` activities. All these activities are required to complete the diagnostic stage, except for `X-ray scan`, which may be skipped, but the order of execution is not relevant. Thanks to process models, PM techniques do consider concurrency. Two sequences $\langle a, b, c \rangle$ and $\langle a, c, b \rangle$ can have the same conformance to the model if the model describes b and c as concurrent activities, while the conformance value will be different if b and c are in sequence or relate to alternative paths. Unfortunately, most ML models view event logs merely as sequential data. When cases get encoded into a vector space, the inference the ML model can produce is based on the distance in this space. The distance between $\langle a, b, c \rangle$, and $\langle a, c, b \rangle$ is accounted in the same way in the vector space, and we cannot differentiate between the sequences based on the reference process model. This limitation impairs capturing concurrent behaviour that is not detected by simply matching the two sequences. In terms of our example, an ML procedure could effectively predict the lead time of a case knowing that the `Payment` activity was executed. Training an ML algorithm to predict the conformance to the diagnostic protocol of a delivered treatment is more complex, and will require a higher amount of training data, as the ML model needs to incorporate examples on the equivalence of the different orders of execution of the `Blood test`, `X-ray scan`, and `Physical test` activities. Encoding this equivalence in vector space spaces, for example, defining suitable pictograms to feed a CNN, is still an open challenge.

6 Toward an Integrated Methodology

Guided by the above considerations about encoding, we will now outline the strategy to be used to properly integrate PM and ML. In the previous sections, we argued that when PM tasks are mapped to ML tasks, PM-specific assumptions should drive the construction of training functions and hyper-parameters selection. Simple ML classification and regression algorithms model the data by a single Gaussian grounded on mean and co-variance. On the other hand, kernel methods like Gaussian Processes and Support Vector Machines, have opened the possibility of flexible models that are practical to work with, but require non-trivial hyper-parameter tuning to fit behavioural data[83].

Figure 3 provides a synoptic view of mapping PM tasks to ML ones.

As an example of non-trivial mapping, let us consider the non-linear relationship between data samples and the expected outcomes addressed by robust ML algorithms with adjusted hyper-parameters. At this point, linear projections as PCA are not effective as t-SNE visualisation [84] to obtain insights from the data. Other challenges with moderate difficulty are related to label availability and imbalanced scenarios [81]. In this case, semi-supervised ML techniques and generative models can tackle the label issue, as well as sampling or synthesising methods are the second ones. Problems related to data quality, in which the difficulty is to build an approximation to have a proper data distribution accentuated, can be solved by enlarging the training data and by a proper tuning of the ML algorithm. Alternatively, the training process can be enriched using generative models [85]. To handle the difficulties outlined in Section 1, when using non-pictorial traces representation "process-friendly" GANs can be considered, like Sequence GANs (SGANs), in which the adversarial samples are designed from discrete sequences, like events. The application of GANs is not limited to data augmentation, as it can be used also for improving data quality for process model generalisation [85]. Preliminary results are available on using GAN-generated data to improve predictive tasks (e.g., lead time of incomplete cases) under an adversarial framework [86].

Coming from non-stationary process behaviour, sampling methods are a promising way to reduce the impact of non-stationary distributions of event log data [87]. After bringing the data to at least a near-stationary behaviour, the business process can naturally change its pattern over time, leading to a burdensome problem called concept drift [88, 89, 52, 16]. In dealing with this problem, a significant part of the PM community has focused on detecting and managing its onset. Regardless of the success of these attempts, we still consider this problem an open issue, since the event data stream is modelled as a complete trace stream, known from the start to the end activity. In reality, the drift onset occurs at an arbitrary position of the event stream, well before the endpoint is reached and the rest of the trace is known. Some researchers are addressing this information deficiency by using statistical adaptations based on the Hoeffding Bounds [90]. In principle, it is possible to rely on statistical assumptions about the confidence interval of the data to make a decision on the drift onset. In other words, it is possible to create ML models and perform predictions supported by an approximated conjecture about the future, obtained from the available event log data. The use of
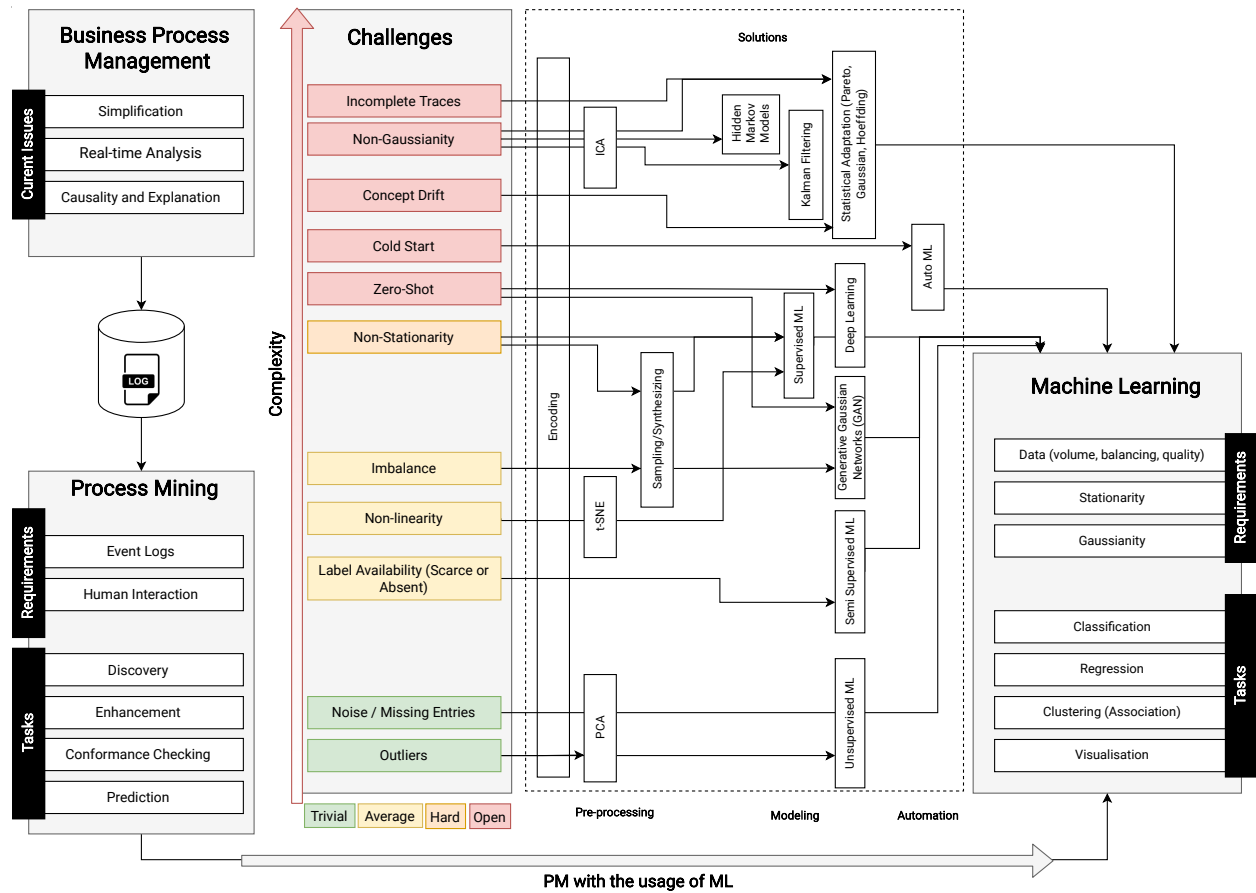
Figure 3: From task to Task, an overview of PM and ML relationship

"stateful" ML models with memory, in particular, Deep Learners based on the LSTM architecture, could enable handling drifts. However, this kind of challenge demands experienced ML practitioners and a robust computational structure.

6.1 Hyper-parameter Tuning

Once a class of ML models has been chosen, hyper-parameter tuning must be performed to instantiate the ML model that delivers the desired accuracy (and possibly some required non-functional properties, like explainability). Searching the model space by trial and error can be burdensome. Automated Machine Learning (AutoML) is a reasonable alternative to tackle these problems grounded on sharing previous knowledge for similar tasks. AutoML can help to handle the classification problems called Zero-shot [91, 92] or Cold-start [93], for which little context information (and even the complete list of classes) may not be available at the start of the training, by taking advantage of meta-features and information on similar models, akin to how human experts start an old-fashioned search for desirable models driven by their experience on related tasks [94]. Some PM research works based on AutoML discuss how to find a suitable PM pipeline by recommending steps. For example, [95] proposed a solution to suggest the encoding method, since the higher number of methods might lead to a tricky selection. Furthermore, there are encoding methods able to fit particular data. It is remarkable that traditional process mining tasks can be leveraged when matched with intelligent decision support approaches.

6.2 Final Recommendations

In this final section, we present a set of recommendations that aim to be valuable for both PM practitioners and researchers.

6.2.1 RECOMMENDATION 1: Choose data representation carefully

When working with PM data structures, it is crucial to carefully translate them into a metric feature space that can be manipulated by ML algorithms. Additionally, it is important to preserve context information, such as control-flow constraints, which are essential for process analysis. The choice of encoding techniques should align with problem-specific goals and constraints.

6.2.2 RECOMMENDATION 2: Fit the data distributions

PM often deals with non-Gaussian, non-stationary distributions. To achieve optimal performance in production, it is advisable to estimate the data distribution instead of relying on the best Gaussian mix approximation. Building training sets interactively poses a significant challenge in PM. Leveraging ML approaches such as AutoML and Active Learning can help reduce the manual burden and improve the process.

6.2.3 RECOMMENDATION 3: Do not assume the availability of a labelled training set

In business process environments, obtaining pre-existing labelled training sets for PM tasks is uncommon. Constructing a training set by correctly sampling the data space is essential, particularly due to the high diversity of process execution conditions in PM tasks.

6.2.4 RECOMMENDATION 4: Consider zero-shot learning

During the training of ML models, the complete set of possible outcomes (co-domain of f) may only be partially known. For instance, in process optimisation, the cost of certain sequences may not be available at the time of training the regression model. It is essential to assess the completeness of the available information when formulating the problem statement to ensure the quality of model inference.

6.2.5 RECOMMENDATION 5: Ensure minimum ML quality at an early stage via constraints

As the estimation of data distribution converges over time, an extended convergence period is unacceptable as it results in a high model error during training. It is possible to impose control flow constraints on ML models when they are known in advance based on domain requirements and regulations.

6.2.6 RECOMMENDATION 6: Incorporate domain knowledge

Domain knowledge plays a critical role in effective PM. Integrating domain-specific information and constraints into ML models can significantly enhance their performance and interpretability. It is important to actively involve domain experts in the feature engineering and model validation processes.

6.2.7 RECOMMENDATION 7: Evaluate model interpretability

PM tasks often require interpretable models to gain insights into process behaviour and make informed decisions. It is essential to evaluate the interpretability of ML models and choose algorithms that provide transparent explanations of their predictions. This becomes particularly crucial when dealing with critical processes or compliance and regulatory requirements.

6.2.8 RECOMMENDATION 8: Continuously monitor and update ML models

Process environments are dynamic, and changes over time can impact the performance of ML models. Establishing a framework for monitoring and evaluation allows the assessment of models' performance and facilitates their timely updates as needed. Continuous learning and retraining of models ensure their accuracy and relevance in evolving process scenarios.

6.2.9 RECOMMENDATION 9: Share knowledge and best practices

Promote knowledge sharing and collaboration within the PM community. Encourage the dissemination of successful case studies, research findings, and best practices to foster learning and advancement in the field. Engage in conferences, workshops, and online forums to connect with fellow practitioners and researchers and stay updated with the latest developments in PM.

By following these recommendations, PM practitioners and researchers can improve the effectiveness and efficiency of process mining applications, enabling better process understanding, optimisation, and decision-making.

7 Conclusions

The growing use of ML methods in PM necessitates a robust and comprehensive methodology for integrating these algorithmic techniques. This paper aimed to address the challenges associated with the ML/PM mapping and identify the fundamental principles for establishing a methodological foundation in this field. Through the analysis conducted in this study, we have provided a set of recommendations that can guide practitioners and researchers in effectively applying ML to PM tasks. These recommendations encompass various aspects of the PM process, from data representation to model evaluation and monitoring. By following these recommendations, PM practitioners and researchers can enhance the effectiveness and efficiency of their ML-driven process mining applications. It is important to acknowledge that the field of ML in PM is constantly evolving, and new challenges and opportunities will continue to arise. As such, ongoing research and collaboration among practitioners and researchers are crucial to refine and expand upon the proposed recommendations. By embracing a methodological foundation that integrates ML techniques in PM, we can unlock the full potential of process mining and leverage the power of data-driven insights to drive process understanding, optimisation, and decision-making in various domains and industries.

References

[1] Deloitte. Global process mining survey. Technical report, Deloitte, 2021.

[2] Wil Van Der Aalst, Arya Adriansyah, Ana Karla Alves De Medeiros, Franco Arcieri, Thomas Baier, Tobias Blickle, Jagadeesh Chandra Bose, Peter Van Den Brand, Ronald Brandtjen, Joos Buijs, et al. Process mining manifesto. In *International conference on business process management*, pages 169–194. Springer, 2011.

[3] Wil Van der Aalst and Ernesto Damiani. Processes meet big data: Connecting data science with process science. *IEEE Transactions on Services Computing*, 8(6):810–819, 2015.

[4] Niek Tax, Natalia Sidorova, Reinder Haakma, and Wil MP van der Aalst. Event abstraction for process mining using supervised learning techniques. In *Proceedings of SAI Intelligent Systems Conference*, pages 251–269. Springer, 2016.

[5] Sebastiaan J van Zelst, Felix Mannhardt, Massimiliano de Leoni, and Agnes Koschmider. Event abstraction in process mining: literature review and taxonomy. *Granular Computing*, 6(3):719–736, 2021.

[6] Ghalia Tello, Gabriele Gianini, Rabeb Mizouni, and Ernesto Damiani. Machine learning-based framework for log-lifting in business process mining applications. In *International Conference on Business Process Management*, pages 232–249. Springer, 2019.

[7] Minseok Song, Christian W Günther, and Wil MP Van der Aalst. Trace clustering in process mining. In *International conference on business process management*, pages 109–120. Springer, 2008.

[8] RP Jagadeesh Chandra Bose and Wil MP Van der Aalst. Context aware trace clustering: Towards improving process mining results. In *proceedings of the 2009 SIAM International Conference on Data Mining*, pages 401–412. SIAM, 2009.

[9] Annalisa Appice and Donato Malerba. A co-training strategy for multiple view clustering in process mining. *IEEE transactions on services computing*, 9(6):832–845, 2015.

[10] Gabriel Marques Tavares, Sylvio Barbon Junior, Ernesto Damiani, and Paolo Ceravolo. Selecting optimal trace clustering pipelines with meta-learning. In *Brazilian Conference on Intelligent Systems*, pages 150–164. Springer, 2022.

[11] Anna Kalenkova, Artem Polyvyanyy, and Marcello La Rosa. A framework for estimating simplicity of automatically discovered process models based on structural and behavioral characteristics. In *International Conference on Business Process Management*, pages 129–146. Springer, 2020.

[12] Arik Senderovich, Alexander Shleyfman, Matthias Weidlich, Avigdor Gal, and Avishai Mandelbaum. To aggregate or to eliminate? optimal model simplification for improved process performance prediction. *Information Systems*, 78:96–111, 2018.

[13] David Chapela-Campa, Manuel Mucientes, and Manuel Lama. Simplification of complex process models by abstracting infrequent behaviour. In *International Conference on Service-Oriented Computing*, pages 415–430. Springer, 2019.

[14] Ved Prakash Mishra, Balvinder Shukla, and Abhay Bansal. Analysis of alarms to prevent the organizations network in real-time using process mining approach. *Cluster Computing*, 22(3):7023–7030, 2019.

[15] Gabriel Marques Tavares, Paolo Ceravolo, Victor G Turrisi Da Costa, Ernesto Damiani, and Sylvio Barbon Junior. Overlapping analytic stages in online process mining. In *2019 IEEE International Conference on Services Computing (SCC)*, pages 167–175. IEEE, 2019.

[16] Paolo Ceravolo, Gabriel Marques Tavares, Sylvio Barbon Junior, and Ernesto Damiani. Evaluation goals for online process mining: a concept drift perspective. *IEEE Transactions on Services Computing*, 2020.

[17] Nicola Di Mauro, Annalisa Appice, and Teresa M. A. Basile. Activity prediction of business process instances with inception cnn models. In Mario Alviano, Gianluigi Greco, and Francesco Scarcello, editors, *AI*IA 2019 – Advances in Artificial Intelligence*, pages 348–361, Cham, 2019. Springer International Publishing.

[18] Vincenzo Pasquadibisceglie, Annalisa Appice, Giovanna Castellano, and Donato Malerba. Predictive process mining meets computer vision. In *International Conference on Business Process Management*, pages 176–192. Springer, 2020.

[19] Alfonso Eduardo Márquez-Chamorro, Manuel Resinas, and Antonio Ruiz-Cortés. Predictive monitoring of business processes: a survey. *IEEE Transactions on Services Computing*, 11(6):962–977, 2017.

[20] Zahra Dasht Bozorgi, Irene Teinemaa, Marlon Dumas, Marcello La Rosa, and Artem Polyvyanyy. Process mining meets causal machine learning: Discovering causal rules from event logs. In *2020 2nd International Conference on Process Mining (ICPM)*, pages 129–136. IEEE, 2020.

[21] Khadijah Muzzammil Hanga, Yevgeniya Kovalchuk, and Mohamed Medhat Gaber. A graph-based approach to interpreting recurrent neural networks in process mining. *IEEE Access*, 8:172923–172938, 2020.

[22] Tpb Wiel. Process mining using integer linear programming. 2010.

[23] Younis Al-Anqoudi, Abdullah Al-Hamdani, Mohamed Al-Badawi, and Rachid Hedjam. Using machine learning in business process re-engineering. *Big Data and Cognitive Computing*, 5(4):61, 2021.

[24] Wil van der Aalst. Academic view: Development of the process mining discipline. In *Process Mining in Action*, pages 181–196. Springer, 2020.

[25] Fabian Veit, Jerome Geyer-Klingeberg, Julian Madrzak, Manuel Haug, and Jan Thomson. The proactive insights engine: Process mining meets machine learning and artificial intelligence. In *BPM (Demos)*, 2017.

[26] Wil MP van der Aalst. On the pareto principle in process mining, task mining, and robotic process automation. In *DATA*, pages 5–12, 2020.

[27] Sylvio Barbon Junior, Paolo Ceravolo, Ernesto Damiani, and Gabriel Marques Tavares. Evaluating trace encoding methods in process mining. In *International Symposium: From Data to Models and Back*, pages 174–189. Springer, 2020.

[28] Benjamin Hilprecht and Carsten Binnig. One model to rule them all: towards zero-shot learning for databases. *arXiv preprint arXiv:2105.00642*, 2021.

[29] G MARQUES TAVARES et al. Meta learning in process mining: Toward a systematic approach to design data analytics pipelines with event logs. 2023.

[30] Efren Rama-Maneiro, Juan Vidal, and Manuel Lama. Deep learning for predictive business process monitoring: Review and benchmark. *IEEE Transactions on Services Computing*, pages 1–1, 2021.

[31] Mohammadreza Fani Sani, Sebastiaan J van Zelst, and Wil MP van der Aalst. Applying sequence mining for outlier detection in process mining. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 98–116. Springer, 2018.

[32] Mohammadreza Fani Sani, Sebastiaan van Zelst, and Wil MP van der Aalst. Repairing outlier behaviour in event logs using contextual behaviour.

[33] Weimin Li, Heng Zhu, Wei Liu, Dehua Chen, Jiulei Jiang, and Qun Jin. An anti-noise process mining algorithm based on minimum spanning tree clustering. *IEEE Access*, 6:48756–48764, 2018.

[34] Xiaoxiao Sun, Wenjie Hou, Dongjin Yu, Jiaojiao Wang, and Jianliang Pan. Filtering out noise logs for process modelling based on event dependency. In *2019 IEEE International Conference on Web Services (ICWS)*, pages 388–392. IEEE, 2019.

[35] Frank Fox, Vishal R Aggarwal, Helen Whelton, and Owen Johnson. A data quality framework for process mining of electronic health record data. In *2018 IEEE International Conference on Healthcare Informatics (ICHI)*, pages 12–21. IEEE, 2018.

[36] Alessandro Berti. Statistical sampling in process mining discovery. In *The 9th international conference on information, process, and knowledge management*, pages 41–43, 2017.

[37] M. De Leoni and F. Mannhardt. Road traffic fine management process, 2015.

[38] J.C.A.M. Buijs. Receipt phase of an environmental permit application process, 2014.

[39] B.F. van Dongen. Bpi challenge 2015 municipality 1, 2015.

[40] Te-Won Lee. Independent component analysis. In *Independent component analysis*, pages 27–66. Springer, 1998.

[41] Remi M Sakia. The box-cox transformation technique: a review. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 41(2):169–178, 1992.

[42] Albert Bifet, Gianmarco de Francisci Morales, Jesse Read, Geoff Holmes, and Bernhard Pfahringer. Efficient online evaluation of big data stream classifiers. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 59–68, 2015.

[43] Marco Roccetti, Giovanni Delnevo, Luca Casini, and Silvia Mirri. An alternative approach to dimension reduction for pareto distributed data: a case study. *Journal of big Data*, 8(1):1–23, 2021.

[44] Alexandre A Emerick and Albert C Reynolds. Combining the ensemble kalman filter with markov chain monte carlo for improved history matching and uncertainty characterization. In *SPE Reservoir Simulation Symposium*. OnePetro, 2011.

[45] Dominic A Neu, Johannes Lahann, and Peter Fettke. A systematic literature review on state-of-the-art deep learning methods for process prediction. *Artificial Intelligence Review*, pages 1–27, 2021.

[46] Wil MP van der Aalst. Concurrency and objects matter! disentangling the fabric of real operational processes to create digital twins. In *International Colloquium on Theoretical Aspects of Computing*, pages 3–17. Springer, 2021.

[47] Chiara Di Francescomarino, Chiara Ghidini, Fabrizio Maria Maggi, Giulio Petrucci, and Anton Yeshchenko. An eye into the future: leveraging a-priori knowledge in predictive business process monitoring. In *Business Process Management: 15th International Conference, BPM 2017, Barcelona, Spain, September 10–15, 2017, Proceedings 15*, pages 252–268. Springer, 2017.

[48] Keshav Thapa, Zubaer Md Abdullah Al, Barsha Lamichhane, and Sung-Hyun Yang. A deep machine learning method for concurrent and interleaved human activity recognition. *Sensors*, 20(20):5770, 2020.

[49] Keshav Thapa, ZubaerMd AI, Yang Sung-Hyun, et al. Adapted long short-term memory (lstm) for concurrent human activity recognition. *Computers, Materials & Continua*, 69(2), 2021.

[50] Dominic A Neu, Johannes Lahann, and Peter Fettke. A systematic literature review on state-of-the-art deep learning methods for process prediction. *Artificial Intelligence Review*, pages 1–27, 2022.

[51] Bartosz Krawczyk, Leandro L Minku, João Gama, Jerzy Stefanowski, and Michał Woźniak. Ensemble learning for data stream analysis: A survey. *Information Fusion*, 37:132–156, 2017.

[52] Lucas Baier, Josua Reimold, and Niklas Kühl. Handling concept drift for predictions in business process mining. In *2020 IEEE 22nd Conference on Business Informatics (CBI)*, volume 1, pages 76–83. IEEE, 2020.

[53] Martin Käppel, Stefan Schönig, and Stefan Jablonski. Leveraging small sample learning for business process management. *Information and Software Technology*, 132:106472, 2021.

[54] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, 2014.

[55] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

[56] Petar Velickovic, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. *ICLR (Poster)*, 2(3):4, 2019.

[57] Fan-Yun Sun, Jordan Hoffmann, Vikas Verma, and Jian Tang. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. *arXiv preprint arXiv:1908.01000*, 2019.

[58] Philippe Krajsic and Bogdan Franczyk. Variational autoencoder for anomaly detection in event data in online process mining. In *Proceedings of the 23rd International Conference on Enterprise Information Systems*, volume 1, pages 567–574, 2021.

[59] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in Neural Information Processing Systems*, 32, 2019.

[60] Jiayi Wang, Raymond KW Wong, and Xiaoke Zhang. Low-rank covariance function estimation for multidimensional functional data. *Journal of the American Statistical Association*, 117(538):809–822, 2022.

[61] Sylvio Barbon Jr, Paolo Ceravolo, Rafael S Oyamada, and Gabriel M Tavares. Trace encoding in process mining: a survey and benchmarking. *arXiv preprint arXiv:2301.02167*, 2023.

[62] Irene Teinemaa, Marlon Dumas, Marcello La Rosa, and Fabrizio Maria Maggi. Outcome-oriented predictive process monitoring: review and benchmark. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 13(2):1–57, 2019.

[63] Pieter De Koninck, Seppe vanden Broucke, and Jochen De Weerdt. act2vec, trace2vec, log2vec, and model2vec: Representation learning for business processes. In Mathias Weske, Marco Montali, Ingo Weber, and Jan vom Brocke, editors, *Business Process Management (BPM)*, volume 11080 of *Lecture Notes in Computer Science*, pages 305–321. Springer, 2018.

[64] Niek Tax, Ilya Verenich, Marcello La Rosa, and Marlon Dumas. Predictive business process monitoring with LSTM neural networks. In Eric Dubois and Klaus Pohl, editors, *Conference on Advanced Information Systems Engineering (CAiSE)*, volume 10253 of *Lecture Notes in Computer Science*, pages 477–492. Springer, 2017.

[65] Paolo Ceravolo, Ernesto Damiani, Mohammadsadegh Torabi, and Sylvio Barbon. Toward a new generation of log pre-processing methods for process mining. In *International Conference on Business Process Management*, pages 55–70. Springer, 2017.

[66] Sholom M. Weiss, Nitin Indurkhya, and Tong Zhang. *Fundamentals of Predictive Text Mining, Second Edition*. Texts in Computer Science. Springer, 2015.

[67] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, page II–1188–II–1196. JMLR.org, 2014.

[68] Aditya Grover and Jure Leskovec. Node2vec: Scalable feature learning for networks. In *International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, KDD '16, page 855–864, New York, NY, USA, 2016. Association for Computing Machinery.

[69] Bryan Perozzi, Vivek Kulkarni, Haochen Chen, and Steven Skiena. Don't walk, skip! online learning of multi-scale network embeddings. In *International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, ASONAM '17, page 258–265, New York, NY, USA, 2017. Association for Computing Machinery.

[70] Antonia Azzini, Sylvio Barbon Jr., Valerio Bellandi, Tiziana Catarci, Paolo Ceravolo, Philippe Cudré-Mauroux, Samira Maghool, Jaroslav Pokorny, Monica Scannapieco, Florence Sedes, Gabriel Marques Tavares, and Robert Wrembel. *Advances in Data Management in the Big Data Era*, pages 99–126. Springer International Publishing, Cham, 2021.

[71] Gabriel Marques Tavares and Sylvio Barbon. Analysis of language inspired trace representation for anomaly detection. In *ADBIS, TPDL and EDA 2020 Common Workshops and Doctoral Consortium*, pages 296–308. Springer, 2020.

[72] Andrea Chiorrini, Claudia Diamantini, Laura Genga, Martina Pioli, and Domenico Potena. Embedding process structure in activities for process mapping and comparison. In Silvia Chiusano, Tania Cerquitelli, Robert Wrembel, Kjetil Nørvåg, Barbara Catania, Genoveva Vargas-Solar, and Ester Zumpano, editors, *New Trends in Database and Information Systems (ADBIS)*, volume 1652, pages 119–129. Springer, 2022.

[73] Mozhgan Vazifehdoostirani, Laura Genga, and Remco Dijkman. Encoding high-level control-flow construct information for process outcome prediction. In *2022 4th International Conference on Process Mining (ICPM)*, pages 48–55. IEEE, 2022.

[74] Vincenzo Pasquadibisceglie, Annalisa Appice, Giovanna Castellano, and Donato Malerba. A multi-view deep learning approach for predictive business process monitoring. *IEEE Transactions on Services Computing*, 2021.

[75] Manuel Camargo, Marlon Dumas, and Oscar González-Rojas. Learning accurate lstm models of business processes. In *International Conference on Business Process Management*, pages 286–302. Springer, 2019.

[76] Arik Senderovich, Chiara Di Francescomarino, Chiara Ghidini, Kerwin Jorbina, and Fabrizio Maria Maggi. Intra and inter-case features in predictive process monitoring: A tale of two dimensions. In Josep Carmona, Gregor Engels, and Akhil Kumar, editors, *Business Process Management (BPM)*, volume 10445 of *Lecture Notes in Computer Science*, pages 306–323. Springer, 2017.

[77] Massimiliano de Leoni and Wil M. P. van der Aalst. Data-aware process mining: discovering decisions in processes using alignments. In Sung Y. Shin and José Carlos Maldonado, editors, *Symposium on Applied Computing (SAC)*, pages 1454–1461. ACM, 2013.

[78] Felix Mannhardt, Massimiliano de Leoni, Hajo A. Reijers, and Wil M. P. van der Aalst. Decision mining revisited - discovering overlapping rules. In Selmin Nurcan, Pnina Soffer, Marko Bajec, and Johann Eder, editors, *Advanced Information Systems Engineering*, pages 377–392, Cham, 2016. Springer International Publishing.

[79] AJMM Weijters, Wil MP van Der Aalst, and AK Alves De Medeiros. Process mining with the heuristics miner-algorithm. *Technische Universiteit Eindhoven, Tech. Rep. WP*, 166(July 2017):1–34, 2006.

[80] Process mining in healthcare tutorial, 2020.

[81] Sylvio Barbon Junior, Paolo Ceravolo, Ernesto Damiani, Nicolas Jashchenko Omori, and Gabriel Marques Tavares. Anomaly detection on event logs with a scarcity of labels. In *2020 2nd International Conference on Process Mining (ICPM)*, pages 161–168. IEEE, 2020.

[82] Mahnaz Sadat Qafari and Wil van der Aalst. Root cause analysis in process mining using structural equation models. In *International Conference on Business Process Management*, pages 155–167. Springer, 2020.

[83] Arman Melkumyan and Fabio Ramos. Multi-kernel gaussian processes. In *Twenty-second international joint conference on artificial intelligence*, 2011.

[84] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[85] Julian Theis and Houshang Darabi. Adversarial system variant approximation to quantify process model generalization. *IEEE Access*, 8:194410–194427, 2020.

[86] Farbod Taymouri, Marcello La Rosa, Sarah Erfani, Zahra Dasht Bozorgi, and Ilya Verenich. Predictive business process monitoring via generative adversarial nets: the case of next event prediction. In *International Conference on Business Process Management*, pages 237–256. Springer, 2020.

[87] Wang Chi Cheung, David Simchi-Levi, and Ruihao Zhu. Learning to optimize under non-stationarity. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1079–1087. PMLR, 2019.

[88] RP Jagadeesh Chandra Bose, Wil MP van der Aalst, Indrė Žliobaitė, and Mykola Pechenizkiy. Handling concept drift in process mining. In *International Conference on Advanced Information Systems Engineering*, pages 391–405. Springer, 2011.

[89] Josep Carmona and Ricard Gavalda. Online techniques for dealing with concept drift in process mining. In *International Symposium on Intelligent Data Analysis*, pages 90–102. Springer, 2012.

[90] Pedro Domingos and Geoff Hulten. Mining high-speed data streams. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 71–80, 2000.

[91] Wei Wang, Vincent W Zheng, Han Yu, and Chunyan Miao. A survey of zero-shot learning: Settings, methods, and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–37, 2019.

[92] Zhong Ji, Guangwen Dai, and Yunlong Yu. Multi-modality adversarial auto-encoder for zero-shot learning. *IEEE Access*, 8:9287–9295, 2019.

[93] Houssem Chemingui, Ines Gam, Raul Mazo, Camille Salinesi, and Henda Ben Ghezala. Product line configuration meets process mining. *Procedia Computer Science*, 164:199–210, 2019.

[94] R. Lily Hu, Caiming Xiong, and Richard Socher. Correction networks: Meta-learning for zero-shot learning, 2019.

[95] Gabriel Marques Tavares and Sylvio Barbon Junior. Process mining encoding via meta-learning for an enhanced anomaly detection. In *European Conference on Advances in Databases and Information Systems*, pages 157–168. Springer, 2021.