Journal on Data Semantics (2021) 10:77–106
https://doi.org/10.1007/s13740-021-00121-2

ORIGINAL ARTICLE

AI-Empowered Process Mining for Complex Application Scenarios: Survey and Discussion

Francesco Folino[1] · **Luigi Pontieri**[1]

Received: 17 February 2020 / Revised: 23 November 2020 / Accepted: 15 February 2021 / Published online: 9 March 2021

Abstract
The ever-increasing attention of process mining (PM) research to the logs of low structured processes and of non-process-aware systems (e.g., ERP, IoT systems) poses a number of challenges. Indeed, in such cases, the risk of obtaining low-quality results is rather high, and great effort is needed to carry out a PM project, most of which is usually spent in trying different ways to select and prepare the input data for PM tasks. Two general AI-based strategies are discussed in this paper, which can improve and ease the execution of PM tasks in such settings: (a) using explicit domain knowledge and (b) exploiting auxiliary AI tasks. After introducing some specific data quality issues that complicate the application of PM techniques in the above-mentioned settings, the paper illustrates these two strategies and the results of a systematic review of relevant literature on the topic. Finally, the paper presents a taxonomical scheme of the works reviewed and discusses some major trends, open issues and opportunities in this field of research.

Keywords Process mining · Artificial intelligence · Data quality · Augmented analytics · Informed machine learning · Structured literature review

1 Introduction

The young discipline of process mining (PM) has already produced a wide range of data analytics solutions for turning a log (i.e., a collection of process execution events) into novel and interesting process-oriented knowledge and insight and for offering operational support at run-time [89]. These solutions include both offline log data analytics methods (addressing *process discovery*, *conformance checking* and *enhancement* tasks) and operational-support methods (allowing for performing *detection*, *prediction* and *recommendation* tasks on ongoing process instances, at run-time).

Many success stories have shown these techniques to be quite effective and efficient in improving a business process, when (i) the available log data provide good-quality information and (ii) the process behavior is regular enough. Under these conditions, the full range of PM techniques can be exploited, possibly combined in a pipeline-like fashion (according to the "Extended L^* lifecycle" model [89]): from the induction and validation of a high-quality control-flow model to the enrichment of this model with stochastic modeling capabilities [76,78], up to the exploitation of these predictive capabilities for run-time support.

By contrast, many existing PM techniques have problems in dealing with less structured processes and with fine-grained (and possibly imprecise/incomplete) logs, like those that typically arise in important non-process-aware application contexts (e.g., related to legacy transactional systems, as well as to ERP, CRM, service/ message/ event-based systems and, more recently, IoT systems [47]) that have been attracting great attention from the PM community—this interest is justified by the wealth of log data that characterizes these contexts and by the need/opportunity of turning them into valuable process-level knowledge.

In fact, most of the efforts spent in the development of PM projects are usually devoted to iterated data extraction/preparation steps, typically performed across many consecutive "prepare-mine-evaluate" sessions, and refined on the basis of the quality (in terms of interpretability, insight

The original online version of this article was revised: The Acknowledgements section has been included.

✉ Luigi Pontieri
luigi.pontieri@icar.cnr.it

Francesco Folino
francesco.folino@icar.cnr.it

[1] Institute ICAR-CNR, Via P. Bucci 8/9C, 87036 Rende, CS, Italy

and actionability) of the results obtained at the end of each session. When dealing with heterogenous, incomplete and/or fine-grained logs, the number of PM iterations and the burden of both data extraction and data preparation steps tend to explode, since choosing the granularity and scope of the log data to be passed to PM tools becomes a critical task, which needs high levels of expertise in terms of both domain knowledge and data/process analysis skills. On the other hand, the effectiveness of each PM session is also determined by the ability of the analyst to tune effectively the parameters of the PM algorithms employed and to post-process their results.

In general, the exploitation of AI methods as a support to the internal decision making of PM algorithms can improve the effectiveness, robustness, usability and efficiency of PM projects in such challenging settings.

Two main families of AI-based strategies have been exploited to this end in the literature: (*A*) using domain knowledge to drive PM tasks and (*B*) addressing auxiliary AI tasks jointly with the target PM task. Both strategies can be regarded as a form of (AI-) *augmented analytics* [72] and have been shown to help reduce the efforts spent in each PM session (in terms of time and skills required to the analysts in the preparation of log data, the configuration and application of PM tools, and the evaluation of PM results), as well the total number of PM sessions needed to eventually obtain satisfactory achievements.

Goal, scope, contribution and novelty The main contribution of this work lies in offering a critical study of the combination of PM and AI techniques according to the emerging vision of augmented analytics, by pursuing two main objectives: *(i)* analyzing what has been done so far in the literature in this direction and *(ii)* reflecting on emerging trends and potentialities that have not yet been explored in full and on the issues that are still open.

Both kinds of analyses are specifically focused on the empowering of classical PM tasks (i.e., discovery, conformance checking, enhancement, detection, prediction, recommendation) through the adoption of complementary knowledge representation, learning and inference capabilities, as a way to better deal with the above-mentioned challenges (i.e., the low-level, incomplete and/or heterogenous nature of the given log data).

The concrete result of our retrospective study is a systematic (and replicable) literature review and a taxonomical classification of the relevant works reviewed. In performing this study, we excluded the works in the literature that only proposed solutions to log data pre-processing problems (such as event data extraction, integration, correlation, selection, transformation, augmentation, enrichment), which aim at supporting the analyst in preparing a collection of log data to the application of standard PM algorithms, but with no kind of direct interaction/integration with the latter. In fact, a recent survey of approaches (including some that leverage AI methods) to the extraction and preparation of log data is already available in the literature [26].

Despite the theoretical and practical importance of the topic considered in this manuscript, to the best of our knowledge, there is no systematic study in the literature that covers it. The existing literature review that looks the closest to our work is indeed the above-mentioned survey of log data extraction and preparation methods [26], which does not cover at all the efforts made in the literature for injecting, in a stronger and more direct way, complementary AI capabilities into core PM tasks.

Organization The remainder of this manuscript is structured as follows:

- Preliminaries on PM techniques and PM projects are provided in Sect. 2.
- Section 3 illustrates three major categories of log quality issues (namely, heterogeneity, incompleteness and hidden activities) that often complicate the application of PM solutions to real contexts and tend to make PM projects more demanding in terms of required time and human expertise.
- The two general AI-based strategies *A* and *B* mentioned above (relying on explicit domain knowledge and on performing synergistically multiple learning/reasoning tasks, respectively) are illustrated in Sect. 4), framed in the wider perspective of augmented analytics.
- Section 5 illustrates in detail both the research questions underlying our literature review and the search protocol employed in it.
- A critical structured discussion of the result of our literature review is given in Sects. 6 and 7, relatively to the adoption of strategies *A* and *B*, respectively.
- Two interesting emerging trends (concerning the usage of ensemble learning and deep learning methods in PM tasks) are discussed in detail in Sect. 8, which are both connected to Strategy B and are expected to find profitable applications in the future within the research field explored here.
- Section 9 summarizes the different classes of works reviewed with the help of an ad hoc taxonomical scheme, while providing the reader with a discussion of related work, and open opportunities and challenges.

2 Background

2.1 PM in Brief: General Aim and some Historical Notes

Following a widely accepted definition, *process mining* (PM) is a discipline at the intersection between data mining and machine learning, on one side, and business process modeling (BPM), on the other side [89]. The general goal of PM can be roughly stated as that of devising data analytics solutions/methods for extracting knowledge/insights from process logs, to eventually support the comprehension, modeling, monitoring, evaluation and improvement of business processes. In a sense, PM techniques exploit the evidence on the real behavior of a business process stored in log data to allow for better addressing the ultimate goal, shared with BPM approaches, of making the handling of a business process and of organizational resources more efficient, effective and aligned to business objectives and requirements (e.g., by inspiring actions for redesigning or reconfiguring the process or work-allocation policies).

In the last decade, PM has been a prominent prolific subfield of research in the BPM area and has attracted much attention from the industry. This is witnessed by the high number of papers concerning PM-related topics that have been published in top-class journals and conferences, which has continued growing from year to year [19], as well as by the wide variety of PM techniques developed in the academy (often as components of open-source frameworks like popular ProM [94], currently featuring more than 600 PM plug-ins) and many commercial tools [18]—e.g., Aris PPM, Celonis Discovery, Disco, Minit, Myivenio, Perceptive Process Mining, Process Gold, QPR ProcessAnalyzer, Signavio Process Intelligence, UpFlux to name a few. Further evidence of the momentum gained by PM is given by: (i) the popularity of the PM Manifesto [88] promoted in 2011 by the *IEEE Task Force on Process Mining*, (ii) the establishment in 2019 of a specific conference on the topic (namely, the *International Conference on Process Mining*), and (iii) the many success stories in public and private organizations of diverse sectors [18]. It is worth noting that the attention of PM researchers and practitioners has started recently spreading beyond the traditional application fields of BPM and started covering diverse relevant sectors, including, e.g., software engineering, healthcare, e-learning, IoT, and Cybersecurity.

2.2 Event Logs and Main Types of PM Techniques

Event logs Most PM techniques were (and still tend to be) conceived to work on a well-structured ("process-aware") *event log*, which conceptually consists of multiple (process execution) *traces*. Each trace is a list of (temporally ordered) events that represent the history of a single process instance (a.k.a. *case*) mainly in terms of the activities that performed during the unfolding of the process instance. Each event may be also associated with additional pieces of information, such as a temporal mark (i.e., a *timestamp*), properties of the *resource* involved in the execution of the activity, as well as other *payload* data (e.g., parameters/results of the performed activity, performance/cost measures, etc.).

Three fundamental properties are usually required to a process log, in order to allow for meaningful and effective applications of traditional PM techniques [59]:

(R1) each trace is explicitly associated with an identifier (*process ID*) of the process that produced it or, alternatively, the log only store traces of one process;
(R2) each event explicitly refers to an instance of the process (*case ID*);
(R3) each event refers, or it can be easily mapped, to an activity (*activity label*);
(R4) the log as a whole provides a sufficiently correct, complete and precise picture of the possible behaviors of the process that is being analyzed.

As noticed in [26] (and discussed in more detail later on) the very task of obtaining an event log that meets such requirements is quite a complicated task in many real-world application scenarios and a major obstacle for the adoption of the PM technology.

Main types of PM tasks (a.k.a. "use cases") A standard categorization of PM tasks and techniques differentiates: (i) *offline* tasks, to be performed on the "postmortem" traces of fully executed process instances, and (ii) *online* tasks, to be performed on the "pre-mortem," partial, traces of ongoing process instances [87].

Three foundational offline PM tasks considered in the literature are *process discovery*, *conformance checking* and *enhancement* [89]. Process discovery concerns the induction of a process model from a given log L. Conformance checking essentially relies on aligning the traces in a given log L with a model, in order to assess and measure the level of agreement between them and detect points of divergence. Enhancement techniques use the data stored in a log L to improve the quality/informativeness of an existing process model M, by suitably repairing or enriching M (e.g., with the addition of decision rules or time/performance annotations).

To provide operational support to the execution of an ongoing process instance c (based on its associated pre-mortem trace), three main online PM tasks have been considered in the literature: (i) the *detection* of deviances between c and a given reference model M; (ii) the *prediction* of some properties of c (e.g., the remaining execution time of c, the outcome of c, the next activity/activities performed for

c), based on predictive models previously extracted from historical log data; (iii) the *recommendation* of actions/choices concerning forthcoming steps of *c*, by possibly leveraging a predictive model. Aggregate-level prediction tasks were recently considered in the PM community, concerning the forecasting of measures/properties for process instances groups, or for the process as a whole and/or its surrounding environment [14,71].

2.3 PM Projects: Life Cycle and Inherent Complexity

Before discussing the structure and main characteristics of PM projects, and the life cycle models proposed for them, let us introduce some basic notions pertaining to the related class of Data Mining (DM) projects.

CRISP-DM model for DM projects Many conceptual models have been proposed in the literature and used in practice to describe and handle the life-cycle of a DM project. For the sake of concreteness, let us examine the one used in the *CRISP-DM* (***CR****oss-****I****ndustry* ***S****tandard* ***P****rocess for* ***D****ata* ***M****ining*) methodology [10]. As a matter of fact, the structuring of DM project life cycle into high-level phases that is adopted in CRISP is quite similar to those of other DM methodologies [60], if abstracting from naming differences and slight mismatches in the level of granularity of some phases.

In general, after a specification of the project goals and of related business questions/constraints (*Business Understanding*), and a preliminary exploration of the available data sources (*Data Understanding*), the actual analysis of relevant data instances starts with a *Data Preparation* phase. The latter typically consists of actions (e.g., collect, explore, clean, select and transform) that are meant to improve the quality of the selected raw data, in terms of relevance, completeness, precision and reliability, as well as to put these data into a form that better suit DM analyses. In the subsequent *Mining* phase, the analyst is in charge of selecting, configuring/tuning and running specific DM models and algorithms. The quality of the results obtained in the previous phase is studied in the *Evaluation* phase, which usually amounts to interpreting the models/patterns discovered, computing quality metrics, analyzing the errors, and estimating a model/pattern application' risks.

Until the quality of the discovered knowledge is not fully satisfying, a new iteration of the entire life cycle is performed. Usually, many interactive "try-and-evaluate" iterations are required, where the analyst often moves back and forth multiple times between phases in a non-sequential manner—e.g., the lessons learnt during a mining session may inspire new ways to prepare the data or even novel (more focused) business questions.

In cases where the discovered (validated) DM model features predictive/inference capabilities (as it happens, e.g., with data classification/forecasting/tagging models), a further *Deployment* phase can take place, where the model is suitably implemented and integrated into some operational system, in order to empower the latter with such "intelligent" data processing capabilities.

The PM^2 model for PM projects In principle, the above high-level conceptualization of DM projects could be adapted to PM settings, taking into account the peculiarities of the data, algorithms (addressing process-aware tasks like conformance checking, discovery, enhancement, detection, prediction, etc.) and evaluation metrics (e.g., concerning fitness, precision and generalization criteria for the case of control-flow models) that characterize these settings.

In fact, such a customization of the DM projects life cycle to the case of PM projects was proposed in [95] as part of a methodology, named PM^2, which specifically consists of the following project phases: *Planning*, *Extraction*, *Data Processing*, *Mining & Analysis*, *Evaluation*, *Process Improvement & Support*. Some of these phases have a single counterpart in the CRISP-DM model. By contrast, i.e., *Process Improvement & Support* (which may possibly involve model deployment actions) emphasizes the strong link that the results of PM analysis are required to have with the ultimate goal of improving and supporting the executions of business processes. Moreover, the Data Preparation phase of CRISP-DM is split into two phases in the PM^2 model: *Extraction* and *Data Processing*. Indeed, a preliminary process log is assumed to be derived from the data sources at hand in the Extraction phase, which mainly amounts to locating, extracting and consolidating relevant event-related data (possibly stored in multiple information systems [26]), and putting them into the form of traces satisfying the requirements mentioned in Sect. 2.2. This entails suitably choosing the *scope* of representation, *angle* and *granularity* [89] in the derivation of such traces, and may require accomplishing tricky event correlation and event abstraction tasks, to map each extracted event record to a process instance and a process activity, respectively, whenever these entities are not referred to in the record itself. Prior to the application of PM techniques, these data may undergo an ad hoc Data Processing phase, which consists of data transformation operations like trace/log filtering, abstraction or enrichment. In a sense, each instantiation of the Data Processing phase is meant to yield a collection of traces that represents a particular "process-oriented view" of the original event data, which is expected by the analyst to allow PM algorithms to discover interesting knowledge/models.

The L^* life-cycle model for PM projects A specialized, five-stage, pipeline-like model for PM projects, named *L^* life-cycle model*, was defined in [89], which mainly hinges on the

discovery, extension and use of control-flow oriented process models.

Specifically, the first two stages, named *Plan & justify* and *Extract*, basically correspond to the first two phases of the PM^2 model, respectively. The third stage, named *Create control-flow model and connect event log*, is devoted to obtaining a control-flow model that explains the input log accurately, by using process discovery and conformance checking techniques. This control-flow model is then enriched (through extension methods) with additional perspectives and/or predictive capabilities [76,78] in the fourth stage, named *Create integrated process model*. The final *Operational Support* stage consists in performing operational-support tasks on pre-mortem traces, based on a prediction-augmented process model.

The L^* model hence provides more structured guidelines for the design and implementation of a PM project. However, it can be applied in full only when the processes under analysis are *lasagna* processes, i.e., structured, regular, controllable and repetitive business processes. A popular rule-of-thumb for characterizing a lasagna process is that "with limited efforts it is possible to create an agreed-upon process model that has a fitness of at least 0.8" [89]. In application settings where the processes exhibit more flexible/chaotic behaviors, it is even difficult to build a control-flow model that is both sufficiently accurate and readable. This makes it prohibitive the application of classical model enhancement and operational-support methods. Things become even more complicated when the traces extracted from log data are affected by noise, inconsistencies and other data quality issues, or they are too heterogenous/fine-grained.

In general, preparing log data is a crucial and difficult task in PM projects (indeed, "finding, merging, and cleaning event data" was identified as an open challenge in the PM manifesto [88]), which can impact negatively on the success of PM initiatives, hence limiting the diffusion of PM techniques in real-life contexts. These issues are discussed in more detail in the following section.

3 Challenging Issues in Real-Life PM Projects

The success of a PM project, which typically requires many long applications of the life cycle phases described above, heavily depends on the expertise of PM analysts, in terms of both domain knowledge and process/data analytics skills, and on their ability (or good luck) in deriving, in a few prepare-mine-evaluate sessions, a log that allows for extracting valuable PM results (e.g., a meaningful/ interesting and conforming enough control-flow model). In particular, most of the efforts spent in a PM project are often devoted to data extraction and data preparation steps—this actually reflects a more general trend of data analytics projects, where such phases consume up to 80% and 50% of the total time and cost [99], respectively. Indeed, as the quality and relevance of the analyzed event log strongly impact on the quality (significance, interpretability, insight/actionability) of the results that can be obtained with PM tools, the construction of such a log is usually an iterative and interactive process in itself, which typically needs multiple data extraction, preparation and mining steps. In fact, data extraction/preparation steps are often refined several times, based on the quality of the results obtained by iteratively applying PM algorithms to different views (differing in scope, angle or granularity) of the same set of original log events.

Two main sources of complexity for PM projects can severely threaten the achievement of satisfactory results: (i) high-levels of variability characterizing the behavior of the process under analysis; (ii) data quality issues affecting the log data available for the analysis.

As to the latter point, different kinds of data quality issues may affect the process logs [5,81,88], which should be handled carefully in PM projects in order to avoid the a "garbage in-garbage out" effect. For example, some major quality dimensions for log data identified in the PM manifesto [88] are (i) *trustworthiness* (i.e., an event is registered only if it actually happened), (ii) *completeness* (i.e., no events relevant for a particular scope are lost), (iii) *semantics* (i.e., any event should be interpretable in terms of process concepts). Based on these dimensions, five levels of maturity for the event logs were defined in [88]. The lowest level of maturity (*) is assigned to logs containing events (e.g., recorded manually) featuring many wrong or incomplete entries, whereas the highest level (*****) is assigned instead to logs (typically coming from process-aware information systems) that are both complete and accurate. In fact, the quality of most real-life logs ranges in the middle of these two extremes [5,82].

The inherent complexity of PM projects is further exacerbated when dealing with the logs of non-process-aware applications (such as CRM and ERP systems, or legacy transactional information systems), which pose problems concerning both the quality of the data and the variability of the processes that generated them. An extreme, but increasingly frequent and important application scenario for PM projects regards the analysis of log data that are not clearly associated with well-defined processes/activities/cases, so that different alternative process-oriented views and models could be extracted from them for the sake of PM-based analyses (consider, e.g., the emerging application of employing PM techniques for analyzing patient histories in healthcare or logs of IoT/event-based systems).

When dealing with event data gathered in the above scenarios, even extracting an informative log (meeting the requirements R1–R4 of Sect. 2) entails a delicate process-oriented interpretation task and suitable data selection and transformation steps.

The rest of this section provides more details on three main issues that tend to cause a gap between the goals and expectancies of PM project analysts and stakeholders, thus complicating the project unfolding and threatening its success:

(I1) the log contains *heterogenous* traces, as a consequence of the fact that it was generated by multiple processes or by a single but unstructured and flexible process (Sect. 3.1);
(I2) the log consists of *low-level* events, which fail to represent process executions at an abstraction level that suits the PM tasks to be performed (Sect. 3.2);
(I3) the log is *incomplete*, in that it does not contain sufficient information (as concerns the PM tasks that are to performed) on the behaviors that the process under analysis has generated and can generate in general (Sect. 3.3).

The last two points above correspond to two well-known types of data quality issues, which affect the *semantics* and *completeness* quality dimensions of [88], respectively.

We pinpoint that log data may also suffer from other kinds of issues that have not been listed above (e.g., the presence of noisy or incorrect information and other problems undermining the *trustworthiness* of the data as well). Moreover, while we are assuming here that the log data under analysis already come in the form of (possibly low-level or incomplete) traces, in real-life scenarios the event records extracted from transactional data sources may lack information on which process instance generated them; therefore, some suitable "event correlation" tasks need to be performed, in order to recognize/define groups of event records that pertain to different process instances and turn each group into a distinct log trace. However, we believe that these additional log quality problems (different from the core issues I1, I2 and I3 above) are beyond the scope of this work, and we refer the reader to [26] for a recent comprehensive survey on these problems and existing solutions in the literature.

Note that all the above described data quality issues tend to occur very frequently when trying to combine Business Process Management (BPM) and Internet of Things (IoT) solutions, as was already noticed in [47], which constitutes a manifesto for such a novel topical area of research. A noticeable work in this area that looks closely related to issues I1-I3 is [53], which addresses the problem of extracting good-quality process models from sensor logs, generated by humans moving in smart spaces (i.e., IoT-enabled environments equipped with presence sensors). Three challenges are, indeed, acknowledged in [53]: *(i)* the high variability of human behaviors, which makes the log data not structured enough to be effectively described through a high-quality (and readable) process model; *(ii)* the abstraction gap between fine-granular sensor data and the human activities that should be analyzed; *(iii)* the need to partition the log into traces (or, in the specific case of modeling human habits, to recognize what really is a habit)—a simple common way to perform such a partitioning consists in applying a fixed time window (e.g., of a day), and regarding each resulting segment as a distinguished trace. Clearly enough, the former two challenges are closely linked to the above-defined general issues I1 and I2, respectively, while the last one might be related to event correlation issues (cf. [26]).

3.1 Issue I1: Heterogenous Logs

PM techniques have been shown to be very effective in settings where the given log data results from the execution of a single, well-structured process featuring a regular behavior. Indeed, in this case, it is possible to discover a good process model easily enough, and to apply enrichment and operational-support methods profitably (hence allowing for a complete application of the L^* life-cycle model described in Sect. 2.3).

However, there are many real-life applications where such an ideal situation does not hold, and the log to be analyzed describes rather heterogenous process instances. This can descend from two causes: (i) the log was generated by different processes, but this fact is not reflected by the presence of an explicit process identifier in the traces (allowing for separating them into different sub-logs); (ii) the log comes from just one process, but this process is handled in a flexible/unstructured manner, and its process instances follow rather diverse patterns of execution.

An illustrative example for the former situation is presented below.

Example 1 (Running example) Consider the case (inspired to a real-life application studied in [31]) of a phone company, where two business processes are carried out, one (W_1) for the activation of services and the other (W_2) for handling tickets, which both consist in performing a subset of the following activities: *Receive a request* (R), *Get more information from the client* (G), *Retrieve client's data* (I), *Send an alert to managers* (A), *Define a service package* (P), *Dispatch a contract proposal* (D), *Fix the issue* (F), *Notify the request outcome* (N). The actual link between these activities and the processes W_1 and W_2 is shown in Fig. 1 via process activity edges—please disregard, for the moment, the event types reported in the bottom of the figure, the meaning of which is clarified in the next subsection. Notice that many activities are shared by the two processes. Assume moreover that the processes are executed in a low structured way by using a non-workflow-based IT system, which does not maintain a process-aware log, but only a collection of traces without any identifier of the process (i.e., either $W1$ or $W2$, in this example) that triggered them. □

Clearly, a situation like that described in the above example does not occur frequently in traditional BPM settings, but it may not be so rare when analyzing the logs of non-process-aware systems. Anyway, the difficulties that descends from the presence of heterogeneous traces in the input log are similar, from conceptual and technical viewpoints, to those arising in contexts where the traces only regard a single process, which, however, features different (often context-dependent) execution scenarios (a.k.a. process variants, or use cases). As an example of the latter situation, consider, e.g., an e-commerce system where quite different procedures are used to handle the orders of gold customers, on the one hand, and those of the other customers, on the other.

When applying traditional process discovery methods to a log mixing different execution scenarios, there is a high risk that a useless "spaghetti-like" process model is obtained, featuring a great variety of execution paths across the process activities. Beside being difficult to read, such a model is very likely to be imprecise, as a consequence of the ambition of these methods to cover the behaviors of all/most of the traces in the log and of the limited expressivity of the model (owing to typical language biases). In other words, the capability of such a model to explain very heterogeneous behaviors usually comes at the cost of also modeling many extraneous execution patterns. In fact, an effective solution for better modeling heterogenous traces is to take care of the existence of different processes/variants, and to try to recognize and model each of them separately [42], as discussed later on.

3.2 Issue I2: Low-Level Logs

Often, the logs of non-process-aware systems do not refer explicitly to meaningful process activities. Such a circumstance occurs when the tracing/enactment system records "low-level" operations, instead of the corresponding "high-level" activities that the users (analysts or process stakeholders) are used to reasoning about. This causes a mismatch between the alphabet of symbols describing the actions in the log and the alphabet of these high-level activities.

Turning such a low-level log into traces referring to high-level process activities is a hard and time/expertise consuming task when the possible event-activity mappings are not one-to-one, at the levels of: (i) types (e.g., a low-level operation can be used as a shared functionality to perform different activities) and/or (ii) instances (e.g., there are *complex/composite* activities that can trigger multiple log events, as smaller "pieces of work", when executed). Notably, the latter case leads to a gap of representation granularity between the log and the users' vision, in addition to having mismatching events and activities alphabets.

A toy example of such a situation is shown in Fig. 1, and described in the example below.

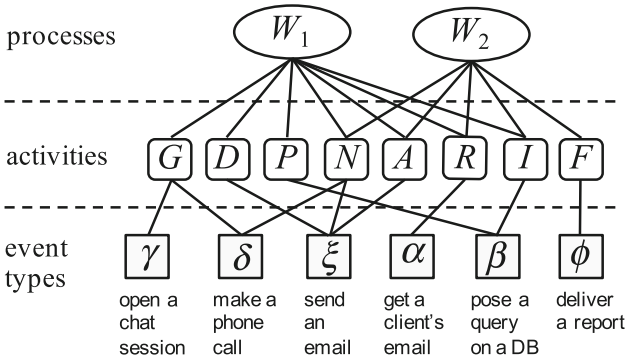

Fig. 1 Example of all possible mappings between activities and events in a low-level log (taken from [31])

Example 2 (... contd) Assume that all activities of the two processes in Example 1 are performed through the execution of generic operations such email exchanges, phone calls, db accesses, and that the log traces produced by the IT system just represent the history of a process instance in terms of such operations, rather than in terms of high-level activities. Figure 1 reports the different low-level event types that can occur in these traces, denoted with Greek letters, as well as their mapping to the (high-level) process activities. There, a link between an activity and an event means that the activity may generate an instance of that event when executed (e.g., an instance of activity N can produce an instance of event δ or ξ). Notably, the event-activity mapping is many to many: for example, activity G can produce an instance of either event δ or γ, while an instance of event δ can be generated by the execution of either activity G or N. Thus, any execution of a process activity generates a (potentially different) instance of one of the events associated with the activity itself. For example, a process instance Φ featuring the activity sequence $R\ I\ G\ P\ N$ might generate one of two sequences of events: $\alpha\ \beta\ \gamma\ \beta\ \xi$ or $\alpha\ \beta\ \delta\ \beta\ \delta$. Finally, in order to extend this scenario with an example of granularity representation gap, let us now assume that any execution of activity N may trigger both events δ and ξ, in addition to generating just one of them. Clearly, under this hypothesis, two further event sequences may be generated from Φ, in addition to the two above, namely: $\alpha\ \beta\ \gamma\ \beta\ \xi\ \delta$ and $\alpha\ \beta\ \delta\ \beta\ \delta\ \xi$. □

When directly applied to such logs, classic PM tools typically yield results of little use, such as incomprehensible/trivial process maps featuring no recognizable activities. Worst, some PM tasks like conformance checking cannot be performed at all.

The urgent need to extend PM approaches with the capability to deal with low-level logs is proven by many recent research proposals concerning the definition of (semi-)automated log abstraction methods [2,4,31,36,59], in two different settings: (1) no predefined activities are known, and

the only way to lift the representation of the events is to use unsupervised clustering/filtering methods [4,36]; (2) there is a conceptualization of process activities, in the stakeholders' minds or process documentation, which provide a sort of "supervised" event log abstraction [2,31,59,83].

3.3 Issue I3: Incomplete Logs

According to [98], an important context-oriented quality dimension for data is *completeness*, which generally pertains the capability of the data to have sufficient breadth and scope for the analysis task at hand. Hereinafter, a log L that does not contain sufficient information for carrying out some chosen PM task T will be said an *incomplete* log (relatively to T).

A direct application of this definition can also lead to classifying low-level logs as incomplete, seeing that the events in such logs mainly lack a reference to high-level process activities. On the other hand, heterogenous logs, mixing traces of different processes/variants, could be effectively handled if those traces were associated with the respective process/variant identifiers (which are instead missing), given that the analyst could select more homogeneous subsets of traces by way of such identifiers; thus, in principle, also heterogenous logs are related to some form of incompleteness. Finally, it is not so rare that a given log only gathers traces that were generated in special operating conditions (e.g., under particular resource allocation regimes, or for specific categories of cases), but this is not reflected explicitly in the form of context-oriented log/trace attributes.

Abstracting from the kinds of incompleteness mentioned here above (which have been partly covered by previous subsections), we next focus on two other challenging situations that make a log incomplete: (a) the log fails to cover the execution flows of a process (in the case of control-flow discovery tasks), and (b) the log data lacks ground-truth labels necessary for performing a supervised PM task (e.g., trace/event classification).

Control-flow incompleteness This kind of incompleteness occurs when a given log L: (i) lacks relevant types of the lifecycle transitions that can characterize the execution of process activities (e.g., it only contains *completion* events only, so that the activities look as if they were instantaneous), and/or (ii) represents activity sequences that do not satisfy the typical "trace completeness" assumptions [89] that underlie control-flow discovery algorithms.

As to the latter point, roughly speaking, approaches to control-flow discovery typically assume that, whenever an activity b depends on an activity a in a process, the log used to analyze the process should contain traces where a is (immediately) followed by b, as evidence for this dependence; by contrast, for any two concurrent activities x and y, there must be both traces where x (immediately) follows y and traces where the opposite happens. In practice, however, two concurrent activities might always appear in the same relative order in a log storing only completion events, owing to some kind of temporal bias—e.g., because one of the activities always finishes after the other, owing to the different durations of the two activities themselves or of some of their predecessor ones [42]).

On the other hand, log completeness might not hold just because the process under analysis features a high level of non-determinism, and an unreasonably large collection of traces would be required to cover the wide range of execution patterns that it could generate.

Finally, even when there are complete log data exist for a process, it can happen that the analysts only have restricted access to them, e.g., owing to privacy reasons, or to the difficulty of extracting and properly preparing these data (especially when the latter are stored in multiple heterogeneous legacy systems).

Lack of labeled examples Another kind of incompleteness descends from the scarcity of labeled data, which is likely to arise in certain outcome-oriented classification tasks (e.g., security breach detection [12]), where the classification model must be induced from example traces with the help of supervised learning methods, and the labels of these traces need to be assigned (or validated) manually by experts. Clearly, in such a setting, the number of labeled examples available for training could be insufficient to train an accurate classifier, especially when using data-hungry (e.g., deep learning) models, so that there is a high risk of overfitting the training set [34].

A different form of label scarcity can affect those log abstraction methods (e.g., [83]) that rely on supervised sequence-tagging schemes, where each example trace is equipped with per-event labels (representing the ground-truth activities that generated the event itself), and the goal consists in learning the hidden mapping from event sequences to activity sequences. Indeed, such per-event labels are difficult to obtain in practice, owing to the strong (and tedious) involvement of human operators/ experts that is required to assign them.

4 Augmented Analytics and Two Strategies for Pushing more AI into PM Sessions

As discussed above, PM projects are highly interactive and iterative and very demanding in terms of time and expertise required, especially when dealing with hard application settings featuring low-quality logs and/or complex/flexible processes. In particular, when dealing with low-level and/or heterogeneous logs, two log-preparation operations tend to

be performed multiple times by the analysts, in the attempt to derive an optimal (w.r.t. the PM analysis that is being carried out) log view: (i) data selection: a subset of events/traces is selected according to either a frequency-based criterion (e.g., retaining only frequent enough activities or paths) or to an application-dependent domain-oriented strategy (e.g., focusing on cases with specific values of their associated data properties); (ii) data abstraction: a (high-level) activity label is assigned to each low-level event by resorting to a predefined mapping/taxonomy.

Many existing commercial tools (e.g., Disco, Celonis Disc., Minit to name a few) provide the analyst with efficient functionalities that allow her/him perform many tentative "prepare-mine-evaluate" sessions, in the ultimate attempt of understanding both the log and the process that generated it. Most of these systems combine intuitive functionalities for visually and interactively select event/case, with rough process discovery algorithms that can quickly return an approximate (but simple) process map, as well as with other data visualization and reporting features borrowed from traditional *Business Intelligence* (BI) frameworks. Notably, the opportunity to provide the analyst with interactive process discovery algorithms has also been investigated in the research community, in order to support repeated process discovery sessions, while possibly trading lower models precision with higher speed and models readability. A noticeable example of such an algorithm is that underlying the popular ProM plug-in *Inductive Visual Miner* [51], which allows the analyst to produce expressive Petri net maps interactively, while letting her/him change the level of abstraction for the activities and execution paths represented.

A step ahead on the way of interactive PM analyses was made by the approach in [3] based on the concept of "Process Cubes," which tries to support the analyst efficiently and intuitively in both data selection and abstraction transformations (through customized versions of the traditional SLICE/DICE and ROLL-UP operators of OLAP, respectively). However, developing such a framework requires skilled experts to carry out a preliminary careful design of the (case/event) attribute aggregations that could be useful for the analysis.

On the other hand, although all the above inter-activity-bound solutions make PM sessions faster and easier, they still need a skilled user, who can exploit the data preparation/mining facilities available in a proper and smart way, as to eventually distillate informative log views and interesting PM patterns/models (possibly navigating through a predefined lattice of log views, in the case of OLAP-like frameworks).

As a more advanced alternative, specifically targeted by our paper, one can think of extending PM methods with smart AI capabilities that help them work better in a more effective, efficient and easier-to-use way, even in hard application scenarios. This is exactly the vision prospected in *Augmented Analytics* [72], which is the subject of the next subsection.

4.1 Augmented Analytics

The term augmented analytics (i.e., AI-powered analytics) refers to the attempt to instill smart assistance for the analyst and a higher level of automation into the entire life cycle of data analytics/mining processes by leveraging AI (and, in particular, of Machine Learning) methods and technologies. Among the various kinds of improvements that augmented analytics efforts aims at providing, let us mention the following major ones:

- allowing different kinds of stakeholders to easily collaborate with each other and with smart analytics tools and models, throughout the entire data analytics process;
- empowering the core Data Mining phase with additional AI capabilities allowing for enhancing the internal decision making of ML algorithms or for automatically tuning ML algorithms/ models (*AutoML*), or with "human-in-the-loop" ML schemes allowing for better controlling the quality of ML results;
- giving assistance/guidance in Data Extraction and Data Preparation phases (e.g., through smart services for data exploration/visualization and data quality assessment, or the suggestion of data transformation/ blending/enrichment/augmentation operations);
- providing assistance/guidance in the Evaluation phase (e.g., by letting the analyst easily navigate across the errors of a predictive model, and perform simulations with the model);
- extending the cognitive capabilities of decision-makers.

As noticed in [72], this perspective looks particularly valuable in complex scenarios involving the analysis of big and/or low-quality data, as well as a promising solution to the shortage of (skilled) data scientists and the need of enabling a more direct interaction (possibly in the guise of "citizen data scientists" [72]) of domain experts and business users with the analyst, the analytics tools and the analysis results.

In accordance with the perspective of augmented analytics, the following subsections specifically focus on two general strategies for empowering data analytics solutions with additional AI capabilities, which can be exploited in the context of PM projects to make the application of PM techniques smarter:

A) using declarative background knowledge, provided by domain experts, and expressing requirements, preferences, or constraints that can guide PM algorithms and PM analysts toward more effective and quicker analyses;

B) performing a main PM task synergistically with other auxiliary data analytics tasks (e.g., the discovery of structures or process-related concepts in the given log data) that can somehow help the former task obtain better and/or more interpretable results in fewer prepare-mine-evaluate sessions.

The following subsections offer a brief general description of these strategies.

4.2 Strategy *A*: Using Background Knowledge (BK)

Extending PM approaches with the capability to exploit domain knowledge (expressing, e.g., known physical/environmental constraints, user preferences, analyst's intuitions, requirements, norms) can be a valuable way to complement the information conveyed by the data available in the log, especially when this information is incomplete or of low quality. Indeed, such a source of high-level information may both help improve the quality of the models/patterns/knowledge derived from the log as well as better align their outcomes to the expectancies of domain experts/business users. In particular, when this background knowledge takes the form of hard/mandatory constraints, the mining of patterns/models can be sped up, thanks to fact that uninteresting portions of the search space can be pruned.

As a matter of fact, the attempt at having ML approaches and inference methods capable of using background knowledge has a long history in AI. In particular, logics-based representation and reasoning is natively embedded in *Inductive Logic Programming* (ILP) and *Statistical Relational Learning* (SRL) frameworks [41,49] (consider, e.g., the use of First-Order-Logics clauses in Markov Logic Networks), while many constraint-based methods have been successfully applied to several Data Mining tasks, such as pattern mining and clustering [73].

A recent effort to frame different ways of incorporating background knowledge coherently into ML-based data analytics processes is presented in [97], under the umbrella term of *Informed Machine Learning* (IML). Basically, IML refers to any kind of ML task where two different (but equally important) sources of information are available: data and *Background Knowledge* (BK). The latter can be expressed in different forms, such as logical rules, constraints, mathematical equations/inequalities/invariants, probability distributions, similarity measures, knowledge graphs and ontologies, to cite a few.

As sketched in Fig. 2, in an IML process, background knowledge can be used in three different phases:

- in the preparation of the data ($BK4Data$), which can so benefit from semantics-driven selection, enrichment, augmentation, and feature engineering;
- in the core learning/mining phase ($BK4Learning$), where the knowledge can be used to act either on the hypothesis space (e.g., by choosing a specific structure/architecture or specific hyper-parameters for deterministic ML models, or by assuming independency between certain variables in the case of probabilistic ML models), or on the training algorithm itself (e.g., by choosing specific loss functions in the search for model parameters, or specific priors in the case of probabilistic models);
- in the evaluation/analysis or the usage (e.g., for inference/prediction tasks) of the discovered patterns/models ($BK4Patterns$).

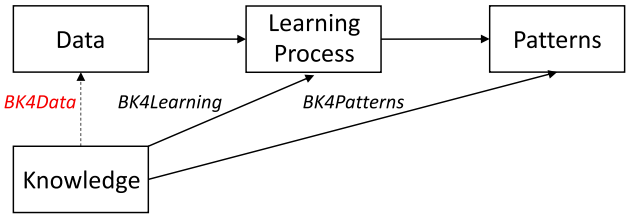

Fig. 2 Informed Machine Learning (IML) flow (based on an image of [97]). The dashed line between "Knowledge" and "Data" corresponds to the IML modality (namely, $BK4Data$) that has not been considered in our literature review

It is worth noting that the second, more advanced and intriguing form of IML above (i.e., $BK4Learning$) has been receiving renewed attention of late—and, in fact, it was the main target of the preliminary literature study conducted in [97]. In particular, several proposals for extending sub-symbolic ML methods with prior knowledge appeared recently [41], most of which rely on stating the learning task as an optimization problem where "fuzzy" constraints encoding both example-based supervision and prior knowledge must be satisfied, and/or on enforcing the prior constraints at testing time.

Strategy A in a PM scenario Different kinds of domain knowledge may be available in real-life PM projects, which can be used as valuable background knowledge for guiding PM tasks, especially when the quality of the log data at hand is not satisfactory. Two major forms of such knowledge are, for example:

- process-oriented ontologies/taxonomies [15] providing an articulated representation of business processes and/or of organizational resources;
- existing workflow models or declarative behavioral models (represented, e.g., via simple ordering constraints over process activities [31,42] or expressive temporal-logics models [96]), possibly induced from historical data with

one of the many process discovery techniques available in the literature [1], and suitably validated by experts.

Let us refer to such a (more or less structured) source of background information as *knowledge base* $\mathbb{B}$, and regard it as a complementary input for PM tasks, besides a log L, and possibly a reference model M (in the case of conformance/enhancement tasks).

As mentioned in Sect. 1, we decided not to consider the case where $\mathbb{B}$ is used for data preparation ($BK4Data$). The reason under this choice is that our work is interested in "smart" AI-based solutions addressing some common PM task (such as process discovery, conformance checking, predictive process monitoring, etc.) with the help of IML methods that exhibit a more direct and tight form of integration between the background knowledge and the target PM task itself.

More specifically, we will focus on two ways of using the background knowledge in $\mathbb{B}$: *(i)* using $\mathbb{B}$ in discovery tasks entailing some kind of inductive learning process ($BK4Learning$), and *(ii)* using $\mathbb{B}$ along with (automatically discovered or manually defined) models or patterns when performing inference, classification or verification tasks ($BK4Patterns$).

4.3 Strategy *B*: Performing Multiple AI Tasks

An alternative/complementary way to tame a difficult data analytics task T, consists in devising methods that can automatically extract and exploit "auxiliary" domain knowledge that may improve the expected quality of T's results (in terms of readability and/or meaningfulness/validity). Such hidden knowledge can play indeed as a sort of surrogate of expert-given domain knowledge, which may be particularly useful performing T on complex, low-level, or incomplete data.

In general data analytics settings, such knowledge could be captured (in a more or less interpretable form) by two different kinds of auxiliary (sub-)models: *(a)* fully fledged ML models, concerning correlated auxiliary tasks and trained in a joint and synergistic way with the primary task T (as in Multi-Task Learning and Learning-with-Auxiliary-Tasks works [8]); *(b)* "smart" data pre-processing/preparation or post-processing components having some level of integration with the core learning/inference/verification technique.

Such an approach grounds on the expectation that the knowledge or inductive bias coming from correlated tasks provides additional information and intelligence skills to an agent that must perform a learning/decision task on the basis of insufficient/unreliable data and can hence lead to better results in terms of generalization, stability and accuracy. Interestingly, the hidden knowledge extracted automatically via these auxiliary tasks can turn useful in performing T, even when it is left implicit (within the internal representation of an ML model), and it is inherently uncertain and approximate.

Strategy B in a PM scenario In principle, different kinds of auxiliary tasks could be beneficial to pursue when approaching a "main/target" PM task that is difficult to solve effectively, e.g., owing to the fact that the log data at hand are insufficient, too finely granular or unreliable.

For example, performing trace clustering [16] and event abstraction [2,31,83] tasks is widely reckoned as an effective means for obtaining high-quality collections of log traces. Indeed, trace clustering can help identify homogeneous and regular groups of traces in a heterogeneous log, which are easier to model and analyze (in that they represent sorts of hidden process variants or usage scenarios); indeed, the application of classic PM discovery tools to such clusters, rather than to the log as a whole, was often shown empirically to yield both better process models [16] (in terms of conformance, generalization and readability) and more accurate forecasts in predictive monitoring settings [93]. Event abstraction tasks allow instead for bringing low-level traces to a higher level of abstraction that is more suitable for traditional PM analyses (e.g., by mapping the given log events to instances of well-recognized process activities that were not explicitly associated with the events, but are assumed/known to have generated them). A further kind of task that can turn useful in process discovery consists in automatically detecting and purging infrequent behaviors and/or anomalous traces (usually referred to in the literature as *noise* and *outliers*, respectively) [11].

However, these three kinds of auxiliary tasks have been prevalently employed so far in separate data pre-processing steps, in order to transform a given low-quality collection of process execution data into an event log that better suits the application of standard PM tools. As mentioned in Sect. 1, this way of exploiting clustering, event abstraction and noise/outlier filtering methods, as a data preparation tool, is out of the scope of our work, which ultimately aims at studying more synergistic forms of hybridizing standard PM task with AI-based auxiliary tasks.

5 Literature Search Protocol

We now illustrate the procedure adopted in our literature study, identifying relevant works linked to the attempt to empower PM methods with complementary AI capabilities, according to the strategies A and B described before.

In order to enable a scientific, replicable and rigorous analysis, we defined a structured literature search protocol, inspired some major principles of *Systematic Literature Reviews* (SLR) [48]. Specifically, we first defined the research questions driving the search, and then performed

some search queries on a specific data source. Next, we applied a number of inclusion and exclusion criteria for selecting the a more focused set of truly relevant works, to be eventually examined in detail. The different components of our search protocol are described in the following, in separate subsections.

5.1 Research Questions

The general objective of our review study is investigating on whether and how standard PM tasks (namely, process discovery, conformance checking, model enhancement, detection, prediction, recommendation) have benefitted so far from the two AI-based strategies introduced in Sect. 4.1, in order to gain effectiveness and robustness in presence of the challenging issues I1-I3 defined in Sect. 3. In line with this goal, we formulated the following research questions:

(RQ1) What approaches to challenging PM problems are there in the literature that leverage Strategy A or Strategy B?
(RQ2) What forms of background knowledge (e.g., taxonomies, declarative constraints etc.) have been employed, and according to which kind of IML modality, in the PM approaches using Strategy A?
(RQ3) Which PM tasks have benefitted from using background knowledge, according to Strategy A?
(RQ4) What kinds of auxiliary tasks have been considered in the PM approaches adopting Strategy B?
(RQ5) Which PM tasks have benefitted from jointly pursuing auxiliary tasks, according to Strategy B?

Clearly, RQ1 is the fundamental research question driving our literature search, seeing as it is meant to identify a set of candidate examples for the two classes of AI-based PM approaches corresponding to Strategies A and B, respectively. The remaining questions are meant to structure these two broad classes by using some informative classification dimensions, namely: the target PM tasks (i.e., discovery, prediction, classification, etc.) addressed in works retrieved for either class (RQ3 and RQ5, respectively); the kind of background knowledge and IML modality (i.e., $BK4Learning$ or $BK4Patterns$, cf. Sect. 4.2) employed in the approaches adhering to Strategy A (RQ2); the kind of auxiliary tasks (e.g., clustering, abstraction) exploited in the approaches of Strategy B (RQ4). All of these classification dimensions are meant to give us a basis for describing the retrieved literature in a structured way (in Sects. 6 and 7), and for eventually providing the reader with a taxonomical scheme (in Sect. 9).

5.2 Search Procedure

The search process was performed on the *Scopus*[1] publication database, since, in our opinion, it ensures a better trade-off between the quantity and quality of indexed works, compared to *Google Scholar* (GS) and *Web of Science* (WoS). Indeed, as noticed in [62], while on the one hand, GS returns significantly more citations than both WoS and Scopus, about half of GS unique citations are not journal papers and include very many theses/dissertations, white papers, informal reports, and non-peer-reviewed papers of questionable value and impact from a scientific viewpoint. On the other hand, WoS is known to be far less inclusive than both GS and Scopus, which may lead to missing (too many) relevant publications.

In order to identify an initial, wide enough, range of potentially relevant works, we defined two distinct queries, one for Strategy A and the other for Strategy B, which are shown in Table 1, along with their associated strategy and the number of results returned. For the sake of presentation, we will refer hereinafter to each query by using the number associated with it in Table 1, i.e., either $Query$#1 or $Query$#2. Similarly, we will sometime refer to the answer set returned by the application of these two queries by using the notation $Group$#1 and $Group$#2, respectively.

Basically, the two queries were devised to have the same structure: three subqueries joined through AND operators, and consisting each of multiple alternative keywords (i.e., of a disjunction of multiple search terms). The former two subqueries define the specific research field (i.e., standard PM tasks) and the challenging scenario considered in our work (featuring incomplete, heterogeneous and/or low-level log data), respectively. As these two subqueries specify our reference "PM setting", they are instantiated identically in both $Query$#1 and $Query$#2. The remaining parts of $Query$#1 and $Query$#2 differ instead from one another, since they aim at defining the specific boundaries of Strategies A and B, respectively. [2]

Precisely, the first shared subquery of $Query$#1 and $Query$#2 (i.e., the first conjunct in the context-oriented preamble of both queries) contains the keywords "process discovery", "conformance checking", "predictive process monitoring", "predictive process model", "compliance checking" and "run-time prediction", which all correspond to well established PM tasks, as well as a few syntactical/semantical variants of the former that can help widen the set of relevant results retrieved—this is the case, e.g.,

[1] www.scopus.com

[2] The choice of performing separate searches for the two AI exploitation strategies mainly serves the objective of easing the interpretation, analysis and presentation of the two classes of works considered in our review.

Table 1 The two search queries employed, for the two AI-based strategies, in the literature review and the respective number of works returned by *Scopus*

#	Query	Strategy	#Works
1	("process discovery" OR "conformance checking" OR "predictive process monitoring" OR "predictive process models" OR "predictive process model" OR "compliance checking" OR "run-time prediction" OR "classifying business log traces" OR "classifying log traces") **AND** ("uncertainty" OR "low-level" OR "complex processes" OR "event abstraction" OR "incomplete" OR "high-level" OR "uncertain" OR "spaghetti-like" OR "noise" OR "infrequent" OR "flexible environments" OR "flexible processes") **AND** ("background knowledge" OR "a-priori knowledge" OR "prior knowledge" OR "domain knowledge" OR "predefined constraints" OR "possible mappings" OR "potential mappings" OR "activity mapping")	A	48
2	("process discovery" OR "conformance checking" OR "predictive process monitoring" OR "predictive process models" OR "predictive process model" OR "compliance checking" OR "run-time prediction" OR "classifying business log traces" OR "classifying log traces") **AND** ("uncertainty" OR "low-level" OR "complex processes" OR "event abstraction" OR "incomplete" OR "high-level" OR "uncertain" OR "spaghetti-like" OR "noise" OR "infrequent" OR "flexible environments" OR "flexible processes") **AND** ("abstraction" OR "outliers" OR "deviance-aware" OR "variants" OR "clustering" OR "groups")	B	82

of the term "classify(ing) (business) log traces", capturing a possible specialized way to carry out certain conformance checking tasks. The second subquery instead contains terms related to the data-related PM issues I1-I3 discussed in Sect. 3 (namely "uncertainty", "low-level", "complex processes", "event abstraction", "incomplete", "high-level", "infrequent", "noise", "flexible processes", and "spaghetti-like"), as well as some variations of these terms.

The rightmost conjuncts in $Query\#1$ and $Query\#2$ were chosen as to capture key distinctive aspects of the respective AI-based strategies, based on our personal background and knowledge of the subject matter. In particular, the last subquery of $Query\#1$ features terms that are likely used in works related to the adoption of Strategy A, namely: "background knowledge", "a-priori knowledge", "prior knowledge", "domain knowledge", "predefined constraints", "possible mappings", "potential mappings" and "activity mapping"—actually, the last three terms may occur in research works where the only kind of a-priori knowledge driving a PM task pertains to the mapping between (low-level) log events and (high-level) process activities.

Devising a specialized set of terms for the works related to Strategy B was not an easy task. Indeed, we noticed that a very small fraction of such works explicitly claimed to synergistically/jointly pursue multiple AI/PM tasks—and very few of them actually contained terms like "auxiliary task," "multiple tasks," "joint learning/training/mining," etc., which one would naturally expect to find instead. We thus had to leverage our knowledge of some major types of auxiliary tasks (e.g., trace clustering, event abstraction) that have been exploited in some popular works following Strategy B, and tried to extend this set of task types with a number of possible alternative ones. As a result, we eventually defined the third subquery of $Query\#2$ as the disjunction of the following terms: "outliers," "abstraction" and "clustering," as well as some terms that might witness the execution of complementary clustering or event abstraction/filtering tasks (namely, "variants," "groups" and "deviance-aware") in the case where such tasks are not referred to explicitly through the terms "outliers," "abstraction" and "clustering."

The search queries $Query\#1$ and $Query\#2$ were run on Scopus on October 30th, 2020, over the titles, keywords and abstracts of all the publications stored in the database. For the sake of fairness and reproducibility, we performed the search in an anonymous mode (i.e., accessing the Scopus website without using any personal account and disabling all cookies), to prevent any possible bias coming from our past searches.

The number of papers returned for these two queries were 48 and 82, respectively, as shown in Table 1.

5.3 Inclusion and Exclusion Criteria

According to [48], the next step in an SRL search protocol, after running the queries, consists in applying a number of well-specified inclusion and exclusion criteria, in order to obtain eventually a selection of really significant and relevant works, while ensuring that this selection

is both replicable and objective. The general rule followed for including/excluding a study in/from our review can be summarized as follows: a work must satisfy every inclusion criterion (IC) in order to be retained, whereas it is discarded whenever it satisfies any exclusion criterion (EC).

The following inclusion criteria were applied to any work returned by our searches:

(IC1) The work has already reached its final publication stage—no "in-press" works are allowed.
(IC2) The work was published as a journal article, as a contribution to conference proceedings, or as a book chapter—neither letters nor reviews were considered.
(IC3) The work is written in English.

In practice, all the above described basic inclusion criteria were enforced directly through the Scopus query engine. This reduced the number of works in $Group\#1$ and $Group\#2$ (from the values reported in Table 1) to 45 and 82, respectively—actually, the size of $Group\#2$ remained unchanged after the application of the inclusion criteria.

After that, all remaining works underwent a little more aggressive and semantic selection phase, devoted to filtering out works that might not be sufficiently significant/mature or not relevant to our scope. To this end, we sequentially applied the following exclusion criteria to any retrieved work satisfying the inclusion criteria:

(EC1) The work was published before 2020 and it has received less than 1 citation on average per year, starting from the year of publication till October 30th, 2020.[3]
(EC2) The work appeared in a workshop (seeing as such a kind of venue may accept proposals at an early stage of development).
(EC3) The work is a survey or an empirical/benchmark study (which hence does not propose any new technical solution), or it is a position/research-in-progress paper (so that any proposed solution is likely not to be mature enough from a technical viewpoint).
(EC4) The work is an abridged version of another work, by the same authors, published later (typically as a journal paper).
(EC5) The work proposes a solution that has no connection with (at least) one of the strategies A and B defined in Sect. 4.1.

[3] The cutoff value of 1 citation per year acts as a reasonable "survival threshold" for purging both obsolete and low-impact studies. This threshold is not applied to the works published in the current year 2020 for the sake of fairness, seeing as such works might still have no citations at all only due to their short life, independently of their quality and relevance.

The application of $EC1$ cut the size of $Group\#1$ from 45 to 35 works, and that of $Group\#2$ from 82 to 42—in fact, this also accounts for the removal (from $Group\#2$) of a paper, published in 2020, missing fundamental meta-data data (namely, the author names and work title) in Scopus.

By using $EC2$, we excluded no paper from $Group\#1$ and 7 papers from $Group\#2$, thus shrinking the size of both groups to 35 elements.

Criterion $EC3$ allowed us to further remove 4 works from $Group\#1$ and 2 from $Group\#2$. This way, the number of works reaching the next exclusion step was reduced to 31 for $Group\#1$ and 33 for $Group\#2$.

By way of $EC4$ we excluded 3 and 2 further works from $Group\#1$ and $Group\#2$, respectively, so coming to the size of 28 and 31, respectively.

The last exclusion criterion, namely $EC5$, directly descends from the research question $RQ1$ and serves the purpose of marking a semantic boundary for our search. For the sake of greater objectivity and fairness, the assessment of $EC5$ was performed independently by both authors of this manuscript, after carefully reading (at least the abstract, introduction and conclusions of) each remaining work. Any possible disagreement among the authors was resolved through discussion and deeper analyses. Enforcing this latter criterion caused the exclusion of further 19 (resp. 13) works from $Group\#1$ (resp. $Group\#2$).

The 9 works of $Group\#1$ and the 18 works of $Group\#2$ that survived the five exclusion stages above are shown in Tables 2 and 3 respectively, along with some major kinds of categorical information (e.g., the main PM task pursued) allowing for characterizing them in a finer way. These are the works that we considered for our literature review study.

For the sake of traceability and replicability, the complete set of works retrieved for Strategy A and Strategy B (after running their respective queries and applying the inclusion criteria) are reported, in a tabular form, in two separate CSV files (named *StrategyA.csv* and *StrategyB.csv*, respectively). The files indicating, for each excluded work, the first criterion that determined its exclusion, can be found in the online folder http://staff.icar.cnr.it/pontieri/papers/jods2021/.

The remaining collections of research works obtained for Strategy A and Strategy B are discussed in a structured form in the following two sections, respectively.

6 Review of PM Methods Using Strategy *A*

This section briefly discusses and categorizes all the relevant PM works that were reckoned to take advantage of explicit background knowledge, based on the results of our literature search. A summarized view of these approaches is offered in Table 2, where column *BK Usage* indicates, for each work, which of the two considered modes of exploiting the available

Table 2 Reviewed research works pertaining Strategy *A*. The works are ordered according to the average number of citations per year. For each work, the table reports the PM task addressed, the kind of background knowledge (BK) taken as input, and the informed machine learning (IML) mode according to which this knowledge is used

#	Work title	#Cit.	Year	Avg. cit.	IML mode	Used BK	PM task
1	Robust process discovery with artificial negative events [39]	123	2009	10.3	*BK4Learning*	Activity constraints	*Process discovery*
2	Online and offline classification of traces of event logs on the basis of security risks [32]	9	2018	3.0	*BK4Patterns*	Possible event-activity mappings and activity constraints	*Conformance checking*
3	Process discovery under precedence constraints [42]	17	2015	2.8	*BK4Learning*	Activity constraints	*Process discovery*
4	Domain-driven actionable process model discovery [103]	12	2016	2.4	*BK4Learning*	Activity constraints	*Process discovery*
5	Using domain knowledge to enhance process mining results [27]	7	2017	1.8	*BK4Learning*	Activity constraints (DECLARE model)	*Model enhancement*
6	Process discovery using prior knowledge [74]	8	2013	1.0	*BK4Learning*	Beliefs on activity dependencies	*Process discovery*
7	ProDiGy: Human-in-the-loop process discovery [28]	3	2018	1.0	*BK4Learning*	(Incrementally refined) process model (Petri net)	*Process discovery*
8	Interactive data-driven process model construction [29]	3	2018	1.0	*BK4Learning*	(Incrementally refined) process model (Petri net)	*Process discovery*
9	Efficient process conformance checking on the basis of uncertain event-to-activity mappings [90]	1	2020	1.0	*BK4Patterns*	Possible event-activity mappings	*Conformance checking*

Background Knowledge (BK)—i.e., either $BK4Learning$ or $BK4Patterns$ (cf. Sect. 4.2)—is adopted in the work, while column *Reference Task* reports the PM task addressed in the work.

6.1 Using BK for Process Discovery

To improve the quality of the process models (more precisely, control-flow models) discovered from incomplete/noisy logs, some of the works retrieved by our literature search propose to exploit a set of a-priori known activity constraints, providing a partial description of correct/forbidden process behaviors [27,39,42,103].

In particular, in [42] the discovery task is stated as the search for a process model (specifically represented as a C-net [89]) for a given log L that satisfies a set of precedence constraints provided by the expert. Essentially, these constraints are meant to express requirements (in terms of the existence or non-existence of edges and of paths) on the topology of the model, regarded as a dependency graph (while abstracting from local split/join constructs in the C-net). This problem is shown to be equivalent to a constraints satisfaction problem (CSP) [73] where these precedence constraints are complemented with log-driven precedence constraints, derived automatically from the traces of L. The problem is also shown to be tractable in two cases: when the background constraints only concern the absence of paths, or they contain any other kind of constraints but forbidden paths. Two different graph-based algorithms are proposed in [42] for these two cases, respectively, as well as an extension of the former algorithm that solves the general (intractable) case heuristically.

Fairly similar in the spirit to [42] is the approach proposed in [103]. The background knowledge is here expressed by an expert in terms of pairwise activity constraints (precedence/causality, parallelism) and designed start/end activities. These constraints are used to define an ILP problem, which also takes account of activity dependency measures extracted from the input log—in particular, a proximity score capturing both direct and indirect succession relationships. User's constraints are considered as strong constraints, which can correct log-driven relationships to some extent.

Table 3 Reviewed research works following Strategy *B*, still ordered by decreasing average numbers of citations per year. For each work, the table reports the (main) PM task addressed and the auxiliary AI tasks performed

#	Work title	#Cit.	Year	Avg. cit.	Main PM task	Auxiliary task(s)
1	Filtering out infrequent behavior from business process event logs [11]	59	2017	14.8	*Process discovery*	*Outlier/Noise filtering*
2	Active trace clustering for improved process discovery [20]	79	2013	9.9	*Process discovery*	*Clustering*
3	Data-driven process discovery—Revealing conditional infrequent behavior from event logs [58]	34	2017	8.5	*Process discovery*	*Classification*
4	Guided process discovery—a pattern-based approach [59]	20	2018	6.7	*Process discovery*	*Abstraction*
5	Behavioral process mining for unstructured processes [25]	28	2016	5.6	*Process discovery*	*Clustering*
6	Process mining based on clustering: A quest for precision [16]	60	2008	4.6	*Process discovery*	*Clustering*
7	Predictive monitoring of temporally aggregated performance indicators of business processes against low-level streaming events [14]	9	2019	4.5	*Prediction*	*Clustering + Abstraction*
8	Mining Predictive Process Models out of Low-level Multi-dimensional Logs [36]	26	2014	3.7	*Prediction*	*Clustering + Abstraction*
9	Discovering hierarchical process models using ProM [55]	29	2012	3.2	*Process discovery*	*Abstraction*
10	Mining usage scenarios in business processes: Outlier- aware discovery and run-time prediction [35]	32	2011	3.2	*Process discovery*	*Outlier/Noise filtering*
11	Online and offline classification of traces of event logs on the basis of security risks [32]	9	2018	3.0	*Conformance checking*	*Trace interpretation (activity inference)*
12	Controlled automated discovery of collections of business process models [38]	19	2014	2.7	*Process discovery*	*Clustering*
13	Leveraging process discovery with trace clustering and text mining for intelligent analysis of incident management processes [17]	21	2012	2.3	*Process discovery*	*Clustering + Classification*
14	Correlating activation and target conditions in data-aware declarative process discovery [52]	6	2018	2.0	*Process discovery*	*Clustering + Classification*
15	Mining multi-variant process models from low-level logs [37]	8	2015	1.3	*Process discovery*	*Clustering + Abstraction*
16	Process discovery from low-level event logs [33]	4	2018	1.3	*Process discovery*	*Abstraction*
17	Model checking as support for inspecting compliance to rules in flexible processes [54]	6	2015	1.0	*Conformance checking*	*Hypothetical reasoning + Data inference*
18	Partial order resolution of event logs for process conformance checking [86]	0	2020	0	*Conformance checking*	*Trace interpretation (event order inference)*

For example, even though an activity A never follows activity B in the log, the discovered model is allowed to include an edge from A to B in case the expert deems that B is directly caused by A, provided that the proximity score from A to B is high enough. Moreover, [103] presents a system that allows the user to revise the background knowledge in an incremental and interactive way: after discovering a model, the user can define novel constraints (allowing for obtaining a better process model), before running the discovery algorithm again.

Another example of the same category is the approach proposed in [39], where the discovery of a control-flow from a log L is stated as a multi-relational classification problem, in which pairwise temporal constraints over the activities (including both local and non-local causal dependencies, and parallelism) are known a priori and used as background knowledge. The approach works in four steps: (i) a set of temporal constraints is extracted from L; (ii) L and all the temporal constraints (both those provided by the expert and those derived from the given traces) are used to generate negative examples (precisely, for each prefix of any trace, negative events are generated stating which activities are not allowed to be executed later on in that trace); (iii) using both the log traces (as positive examples) and the artificially generated negative events, a logic program is induced that predicts whether a given activity can occur in a given position of a given trace; (iv) the logic program is converted into a Petri net.

The problem of discovering a control-flow model from a noisy log is faced in [74] by inducing a probabilistic graphical model taking the form of an information control net (ICN), the nodes of which represent different process activities. The proposed method takes as further input background knowledge expressed in terms of degrees of belief, associated with elements of the ICN. This knowledge is exploited according to a Bayesian inductive learning scheme, and it is shown to help improve the quality of the discovered process model.

Quite a different, interactive, approach to injecting the user's knowledge into a process discovery task is adopted in [29]. Here, the user is allowed to build a Petri net process model incrementally, by possibly combining her/his domain knowledge with statistical information extracted (by using PM techniques) from a given event log. This enables a human-in-the-loop scheme where the user and the process mining component can collaborate, while leaving total control to the former over the latter. More specifically, log-driven statistics concerning the ordering relationships between the process activities are derived from the log and presented to the user by visually projecting them onto the process model defined so far. This lets the user make informed decisions about where and how to accommodate a novel activity (i.e., an activity that appears in the log but not yet in the process model). To this end, only three predefined kinds of synthesis rules [21] can be used, which all ensure the soundness of the resulting model.

A similar incremental process discovery approach is proposed in [28], which describes a framework named *Prodigy*. Rather than just projecting log-driven statistics on the current process model, in this work the PM component is devised to recommend a number of alternative ways to refine the current version of the process model by inserting a novel activity (according to the same kinds of synthesis rules [21] as in [29]). These recommendations are ranked on the basis of how accurate the corresponding resultant process models are in describing the given log. The user can also leave the control to the system according to an "auto-discovery" operation mode (for a chosen number of refinement steps, or until the conformance scores of the model fall under some thresholds), which consists in automatically selecting and applying the top ranked recommendation.

It is worth noting that, differently from all the other works described in this subsection, the incremental PM schemes proposed in [28,29] do not require explicit domain knowledge to be available at the beginning of the PM session. However, whenever such knowledge is owned by the user, it will guide her/him (together with new statistics extracted from the log) in each incremental refinement of the process model. On the other hand, the updates made by the user at a certain time impact on those that can be performed subsequently, hence allowing for transferring the user's knowledge from one iteration of the approach to the following ones.

6.2 Using BK for Model Enhancement

A declarative process model encoded in language *DECLARE* [96] is used in [27], as an informative form of background knowledge, in order to improve an existing process model, previously discovered from a given log L, in a post-processing fashion. Specifically, assuming that the discovered model takes the form of a process tree, three alternative methods are proposed to enhance it, based on both L and the given DECLARE model: (i) brute-force search; (ii) a genetic programming scheme where candidate models are made evolving according to a fitness function that accounts for both log conformance metrics [89] and the fraction of a priori constraints fulfilled; and (iii) a heuristics algorithm that directly tries to correct a model based on the types of constraint it infringes. Although the post-processing nature of this solution might lead to regard it as a $BK4Patterns$ approach, since it comes back to the input data (which is, indeed, a peculiarity of the enhancement PM task), we believe that it is more appropriate to classify it as an instance of the $BK4Learning$ category.

6.3 Using BK for Conformance Checking

The core problem of checking whether low-level traces comply to high-level behavioral models is considered in [32,90] in the challenging setting where the ground-truth mapping between the events of the traces and the activities in the model is unknown and only uncertain information is available on it—actually, the models in [32] are meant to represent known security breach patterns.

Since pre-processing the traces with deterministic log abstraction methods [2,59,83] in such a setting can lead to misleading results [90][4], both proposals [32,90] adopt a probabilistic approach for evaluating the degree of compliance of each trace over either all of its admissible interpretations [90] or a representative subset of them, computed via Monte Carlo sampling [32]. In particular, in [90], conformance is analyzed at different levels of detail, across a hierarchy of (SESE) process fragments. An efficient computation approach is defined to help avoid enumerating all the possible interpretations ("translations") of a given log trace, so as to only focus on the portions of the trace that are really affected by mapping uncertainty and on the fragments that actually feature the activities involved in these portions.

Prior event-activity mapping probabilities are used in both works above to discard "meaningless" interpretations. Known activity-level precedence constraints, describing the behaviors of the processes in a loose partial fashion, are exploited as well (as a partial process model) in the Monte Carlo simulation of [32], to purge invalid interpretations of a given trace. Since the proposals in [32,90] can be viewed as informed inference-oriented PM methods, they have been both classified as instances of category $BK4Patterns$ in Table 2.

7 Review of PM Methods Using Strategy B

This section discusses the works that have been reckoned to take advantage of the "multi-task" Strategy B introduced in Sect. 4.3, within a PM setting, based on our literature search. Let us recall that these works rely on pursuing a number of auxiliary tasks, when approaching a "main/target" PM task (which would be difficult to solve effectively in isolation, e.g., owing to the fact that insufficient or unreliable log data are available).

[4] In fact, in a scenario affected by a high level of uncertainty (e.g., owing to the combined presence of flexible processes and of ambiguous event-activity mappings), selecting just one optimal interpretation for a trace leads to losing information whenever the trace can be explained via different similarly plausible alternative interpretations, and the analyst's expertise does not suffice to identify the "right interpretation" and definitely discard the other ones.

The works are described next in separate subsections, each of which is titled with the name of the main PM task, followed by those of the auxiliary ones. These two kinds of information are also reported in Table 3, for each to the works reviewed.

7.1 Process Discovery with Clustering

As discussed in the final part of Sect. 4.3, trace clustering task are usually performed before approaching a discovery/prediction task, in order to recognize subgroups of traces exhibiting more homogenous (and hence easier to describe and predict) behaviors. However, such a two-phase approach does not guarantee that the bias of the clustering task is aligned with the evaluation bias of the discovery/prediction task of interest.

The methods proposed in [16,20] deviate from this mainstream use case of clustering methods, and look well aligned to the essence of Strategy B. Indeed, the algorithm *ActiTrac* [20] reduces the above-mentioned dichotomy by addressing two joint clustering and control-flow (CF) discovery tasks, with a shared goal: maximizing the average conformance of the different control-flow models describing the discovered clusters. Precisely, the conformance of each cluster is evaluated by computing the fitness of the control-flow model induced for the cluster (through algorithm HeuristicsMiner [101]) with respect to the cluster traces. A greedy iterative clustering-plus-induction scheme, inspired to Active Learning methods [80], is employed to solve this problem heuristically. Experimental results confirmed that the control-flow models discovered this way are easier-to-read and more accurate in describing their associated traces, compared with a single process model induced from all the traces.

The problem of discovering a set of CF schemas, named *Disjunctive Workflow Schema (DWS)* from a given log L is considered in [16]. In fact, this work is a follow-up paper of [43], where the problem was considered originally, in a setting where the model is required to satisfy fitness and precision requirements. Since this problem is intractable, a heuristics solution was proposed in [43], which starts building a preliminary DWS consisting of a single CF schema, induced from the entire log L, and then iteratively refines the DWS through the following hierarchical clustering procedure: (i) the most imprecise/flexible CF schema W in the DWS is selected; (ii) the traces of W are split into clusters; (iii) W is replaced in the DWS with a collection of novel CF schemas, induced each from one of the clusters found. This approach is generalized in [16], where it is made both interactive and independent of the particular clustering/process discovery algorithms.

While the individual models discovered after clustering a given set of traces are usually simpler than one induced from all the traces, these models tend to share many duplicate/clone fragments, so that their overall description

complexity is rather high. A two-way process discovery approach is presented in [38], where a discovered process model can be both refined into more specific process variants based on trace clustering ("slicing"), and decomposed into hierarchically structured fragments by way of sub-processes extraction and merging ("dicing"). By slicing, mining and dicing recursively, this approach can eventually yield an easy-to-understand set of process models, satisfying user-specified constraints on the number and structural complexity of the discovered models. To let the user specify a minimal level of fitness for the returned process models, a top-down hierarchical variant of the fitness-aware clustering algorithm ActiTrac [20] is proposed for the splitting step.

A different approach to control-flow model discovery is proposed in [25], which relies on defining a methodology for mining out common sub-processes (representing typical behavioral patterns). These patterns are expected, indeed, to be better than an overall control-flow model at modeling complex unstructured domains. To this end, a hierarchical graph clustering method is exploited in the discovery of the most relevant (based on a description length criterion) sub-processes, and of their mutual hierarchical relations. The input of this clustering method is a set of instance graphs, each derived from a given log trace, based on the causal inter-activity dependencies detected by an existing (noise-aware) process discovery algorithm. An ad hoc repairing method is employed to solve structural anomalies that can affect the graphs of noisy instances.

7.2 Process Discovery with Outlier/Noise Filtering

The attempt to devise process discovery methods robust to noisy logs dates back to *Heuristics Miner* [100], and it has been pursued over time in several other discovery approaches, such as *Fodina* [91] and *Infrequent Inductive Miner* [50]. In fact, all of these process discovery methods perform the induction task over a collection of pairwise (e.g., directly follows and/or eventually follows) activity relations, providing a rather approximated representation of the log behaviors, and mainly deal with noise by filtering out the activity pairs in these relations that occur rarely in the log (according to user-given frequency thresholds).

A more sophisticated noise filtering approach is proposed in [11], where the identification of infrequent activity dependencies is combined with the discovery of a (rough) behavioral model for the log—even though the ultimate goal of this work is to help the analyst remove the manifestations of such infrequent behaviors from the log. The approach essentially relies on inducing an automaton from the log, where the nodes and transitions represent process activities and inter-activity dependencies, respectively, and then deriving a reduced version of this automaton purged of all the infrequent dependencies. By replaying the original log through this reduced automaton, it is possible to detect all the noisy events (as those that no longer fit the automaton), and possibly remove them all from the log.

Still in the context of process discovery, the problem of detecting outlier traces (representing exceptional process executions, determined, e.g., by system malfunctioning or anomalies) was also addressed in [35]. There, the idea of tightly combining a clustering of the log traces with the discovery of per-cluster control-flow models (pursued, e.g., in [20,43]), is extended with the attempt to make the clustering aware of outlying traces, so that the resulting control-flow models only describe typical execution scenarios of the process (and not outlier traces). To this end, a robust co-clustering method is devised that splits the traces based on their correlations with ad hoc behavioral patterns (encoding activity precedences and split/join constructs, and discovered automatically), and filters out all the (outlier) traces fallen into small clusters or no cluster at all.

7.3 Process Discovery with Abstraction

To deal with fine-granular log events, some control-flow discovery tools were devised to incorporate automated activity abstraction capabilities, such as the method proposed in [55] , which both discover hierarchies of process models based on automatically extracted abstraction relationships over the activities. In particular, in [55] , a two-phase process discovery strategy is used, where: (1) the given log is first brought to a desired level of granularity with the help of *Pattern Abstractions* extracted from the log itself, and then (2) a process map is discovered from the abstracted log (using the popular ProM plug-in *Fuzzy Miner* [45]). In its turn, the former phase consists of three steps: (i) finding common execution patterns (in the form of a sequential patterns over the activities, including tandem/maximal repeats); (ii) defining abstractions over these patterns; (iii) using these abstractions to pre-process the traces. Repeated applications of this method allow for deriving an activity abstraction hierarchy. Basically, at each iteration, for each defined abstraction x, a sub-log capturing the manifestation of that x' abstraction pattern is generated, and the Fuzzy Miner plug-in is applied on the "transformed" log to discover a process model for it.

The problem of discovering a good-quality behavioral model for a given low-level log is rephrased in [33] into that of inducing two different models: (i) a Hidden Markov Model (HMM) from the log describing the process execution flows in terms of (hidden) activities; and (ii) a high-level control-flow graph over the activities, which is built upon inter-activity dependency statistics extracted from the HMM model. Prior knowledge on activity dependencies and mappings can be used to guide the discovery of the HMM model (and, indirectly, of the high-level control-flow model). Thus,

in principle, the approach in [33] could be also seen as an Informed-PM method of category $BK4Learning$.

The solution proposed in [59] for the supervised abstraction of low-level logs can be regarded as a piece of a more comprehensive methodology for discovering abstract process models from low-level logs. This methodology, beyond the pre-processing steps meant to abstract low-level traces into higher-level (activity annotated) ones, includes, in fact, two other steps aimed at achieving a model recognizable by process stakeholders: (i) inducing a high-level process model from the abstracted version of the log, and (ii) computing alignment-based conformance measure between the original (low-level) event log and an expanded version of the discovered model (where each high-level activity is replaced with the associate activity pattern, describing the activity in terms of low-level events). Thus, the whole approach can be regarded as an example of hybrid approach that combines discovery and abstraction tasks (plus alignment-based conformance checking).

7.4 Process Discovery with Clustering and Abstraction

In order to facilitate the analysis of logs generated by low structured processes and that contain low-level traces, a twofold mining problem was introduced in [37]: (i) finding a conceptual (co-)clustering function that automatically abstracts log events into (non-predefined) event classes, while assigning each trace to a different process variant based on context data (e.g., properties of cases or environmental factors); (ii) inducing a specialized workflow schema for each single variant (expressed in terms of the discovered event clusters).

Notably, the event (resp. trace) clustering function is encoded via predictive (logical) rules over the event attributes (resp. the event classes). This allows, on the one hand, for providing the analyst with an interpretable description of both the discovered event classes and the trace clusters; on the other hand, to support the implementation of advanced runtime services (besides providing descriptive analyses only).

7.5 Process Discovery with Classification

A process discovery method (named *Data-aware Heuristic Miner*) is proposed in [58] that is meant to help reveal semantically relevant infrequent behavior correlated with process data properties. To this end, both activity labels and event data attributes are given as input to the control-flow discovery algorithm (built on Heuristics Miner [101]), which leverages a classifier induction method to capture conditional inter-activity dependencies, and embed them in the resulting process model.

7.6 Process Discovery with Clustering and Classification

A combination of trace clustering and text mining methods is proposed in [17] in order to enhance process discovery tasks in the flexible environment of incident management. Specifically, a number of accurate process models is discovered by using a fitness-driven pattern-based clustering method (implementing a semi-supervised learning strategy, based on both selective trace sampling and a greedy model optimization mechanism). Basically, the traces that do not fit well enough any of these clusters are kept apart, and made to undergo a combination of text mining and classifier induction methods, with the ultimate aim of finding non-functional patterns (i.e., patterns unrelated to control-flow aspects) from the data fields (including free text descriptions) of those atypical cases.

In [52], a method for inducing multi-perspective declarative process models is presented, which can discover conditions on (both categorical and numeric) data attributes related to the occurrence of log events pairs. Specifically, focusing on pairwise *DECLARE* [96] constraints, the method can infer correlated conditions on the payloads of the activation and of the target of such a constraint—hence allowing going beyond the pure modeling of control-flow aspects. The discovery of such correlated conditions leverages both clustering techniques and the induction of interpretable classifiers.

7.7 Prediction with Clustering and Abstraction

The combination of discovery and event abstraction capabilities was also exploited in both [36] and [37], in a predictive monitoring setting (aimed at forecasting the remaining time of a partial trace) and a traditional control-flow discovery setting, respectively. In both cases, the target task is pursued synergistically with two auxiliary tasks that are addressed to discovering two kinds of clusters: (i) event clusters (playing as hidden activities), and (ii) trace clusters (playing as process execution scenarios/variants). Both kinds of clusters are defined via logical classification rules, for interpretability and applicability reasons.

An approach similar to that in [36] is exploited in [14] for forecasting aggregate performances of a process, defined over time windows. Here, a further auxiliary task consists in the discovery and application of a time-series forecasting model, which is devoted to predict the performances of the instances, of a current time window w, that have not started yet is addressed. This task is performed jointly with the same two tasks of [36], now used to make single-instance forecasts on the ongoing low-level traces of w.

7.8 Conformance Checking with Trace Interpretation

As noticed in Sect. 6.3, the problem of checking to what extent a low-level trace complies to high-level behavioral models was studied in [32], in a scenario where the ground-truth mapping between the events of the traces and the activities in the model is unknown. In [32] such a conformance checking task is performed, on any given low-level trace, jointly with that of bringing the trace itself to the level of abstraction of the reference process activities. To this end, a representative set of activity-level interpretations is computed dynamically for the trace via a Monte Carlo sampling procedure. More details on this work are in Sect. 6.3, where it has been already reviewed as an application of Strategy A.

A very recent work [86] addresses the problem of performing a conformance checking task in a situation where the uncertainty pertains the ordering of the events in the traces—i.e., the events are only partially ordered, e.g., owing to synchronization problems, manual event recordings, or data corruption. Here the sub-problem of partial order resolution is defined, which amounts to estimating a probability distribution over all possible total orders of the events in a trace, and several alternative estimation models (based on different notions of behavioral abstraction) are proposed. In order to make conformance checking tasks feasible over a partially ordered trace, an approximation method is presented, which consists in sampling a subset of all the candidate order resolutions of the trace while ensuring a statistical error guarantee.

7.9 Conformance Checking with Data Inference and Hypothetical Reasoning

A conformance checking methodology is proposed in [54] that leverages complementary data inference and hypothetical reasoning capabilities. The methodology aims at enabling thorough verifications of a flexible process model, while tackling real-life processes issues like unpredictability, inconsistency and uncertainty. To this end a combination of Hybrid Logics and Description Logics methods is exploited, which support rich representations of the entities, data and context associated with a process instance. These additional knowledge representation and reasoning capabilities allow for possibly inferring missing information (e.g., missing connections between available evidence and previous inference results), which prove useful for the checking task, as well as supporting hypothetical reasoning (e.g., working with an updated model of the world). The checking results can be exploited to visualize and verbalize connections of the properties checked with their dependencies over time. Considering the ability of this approach to take knowledge on process-related business entities and context, it could also be seen as an instantiation of Strategy A. However, we prefer to assign it to Strategy B, owing to the flexible range of reasoning services that it offers to the analyst.

8 Emerging Trend: Using Deep/Ensemble Learning Methods for PM

As discussed in Sect. 4.3, Strategy B is ultimately meant to support a target learning/inference task (coinciding with one of the standard PM tasks in our specific setting) with hidden knowledge or inductive bias coming from some other auxiliary task, which is performed jointly with the target one. The systematic literature review described in Sect. 7 has revealed that the auxiliary tasks exploited in challenging PM settings often coincide with either traditional ML tasks (mainly, clustering and classification) or specialized inference/reasoning tasks (primarily, event abstraction and trace interpretation) that aim at providing the target task with a semantically lifted view of log data.

In particular, in the case of ML-oriented auxiliary tasks, researchers in the field of PM have typically resorted to the use of classic ML models and algorithms (e.g., decision trees/rules induction), combined with a preliminary application of ad hoc feature engineering methods—the types of features commonly adopted for this purpose include different types of sequence/set-based patterns extracted from the log traces [93].

The recent impressive development of Deep Learning (DL) [41] solutions has changed this landscape, by offering the opportunity to reduce the burden of manual feature engineering tasks, owing to the inherent ability of a Deep Neural Network (DNN) to learn expressive feature hierarchies for the input data automatically. And, in fact, the use of DNNs has been rapidly growing of late in the field of PM, especially in predictive process monitoring settings [22,93].

In the context of our study, the native representation learning capability of a DNN-based model, say M, can be regarded as an advanced form of auxiliary data preprocessing/preparation task, which is integrated tightly and transparently with M and solved in an integrated synergistic way with target task addressed by M—indeed, the feature abstraction (lower) layers of M are trained jointly with the (final) task-specific layers of M.

Another recent line of research related to Strategy B consists in exploiting Ensemble Learning methods as a form of Meta-Learning [92], where different base models are eventually integrated with some combination model. In a sense, the combiner sub-model acts as a sort of meta-model that reasons on the "high-level" views of the input data instances that result from projecting them onto the space of the base model decisions/predictions (possibly complemented with internal data representations computed by the base models). Training

the base models and the combiner model can be regarded as auxiliary tasks in the perspective of our study.

In other words, the research works combining PM techniques with deep/ensemble learning methods can be regarded as an emerging subclass of those exploiting Strategy *B*. However, we noticed that very few of these works explicitly relate the choice of combining ensemble/deep learning methods and PM tasks to the need of dealing with the log quality issues I1-I3 discussed in Sect. 3—despite the ability of both kinds of methods to work automatically on abstract data representation can alleviate the problems stemming from having low-level log data. Moreover, all of these works only focus on predictive PM tasks (mainly, run-time prediction, recommendation, and classifier-based conformance checking). Therefore, in order to extend our literature review to this important area of related research, we defined an ad hoc search query, which takes account of the peculiarities described above: *("event logs" OR "business process") AND ("run-time prediction" OR "predicting next event" OR "event sequence prediction" OR "classify process instances" OR "activity prediction" OR "process prediction" OR "predictive process monitoring" OR "predictive business process monitoring" OR "deviation detection")* ***AND*** *("ensemble learning" OR "neural networks" OR "deep learning" OR "deep predictive model" OR "LSTM").*

By using the search protocol described in Sect. 5 in combination with this query, we eventually obtained 18 works, summarized in Table 4. To be more precise, after running the query and applying all the inclusion criteria we obtained 43 works; 9, 1, 7, 3 and 5 of these works were then filtered out by sequentially enforcing the exclusion criteria EC1, EC2, EC3, EC4, and EC5, respectively. Details on the research works obtained after running this query on Scopus along with all the inclusion criteria, can be found in a CSV file, named *EmergingTrends.csv*, within the shared online folder: http://staff.icar.cnr.it/pontieri/papers/jods2021/.

The final collection of research works obtained this way are discussed in the following, within two separate subsections: one for the works leveraging ensemble learning methods (Sect. 8.1), and the other for those relying on deep learning methods (Sect. 8.2).

8.1 Use of Ensemble Learning Methods in PM Tasks

In [12], it was noticed that there is no consensus on which is the best among the many different kinds of patterns proposed [93], for deriving a vector space representation of log traces, while mixing multiple pattern families is likely to produce too heterogeneous and sparse representations. This motivated the definition in [12] of an ensemble learning method, where multiple base (deviance-oriented) trace classifiers are trained on different feature-based views of the log (each obtained by mapping the traces onto a distinguished collection of patterns) [12]. A probabilistic meta-learning (stacking) procedure is adopted to combine the discovered base classifiers into an overall "multi-view" classification model, which will classify any novel trace x on the basis of both the data features of x and the predictions returned for x by the discovered base models.

A different (heterogeneous) ensemble strategy is devised in [64], which consists in exploiting three different base learners to build up an ensemble model for predicting whether a process instance will eventually violate a QoS constraint. More specifically, the approach combines an artificial neural network (ANN), a constraint satisfaction technique (CSP), and the aggregation of QoS measurements into a (voting-based) ensemble, which is shown to improve the performances of the each of the base learners.

8.2 Use of Deep Learning Methods in PM Tasks

Different Deep Neural Network (DNN) architectures have been proposed recently (see Table 4) for inducing pre-mortem trace predictors, most of which rely on using Recursive (primarily, LSTMs) Neural Networks.

In particular, an LSTM-based model is defined in [84], for predicting both the next activity and the associated timestamp for a running process instance. The encoding of each trace event in input is simply obtained by concatenating a one-hot representation of the associated activity with a number of numerical features derived from the timestamp of the event. The resulting vectorial representation of an input trace is passed to a stack of LSTM layers, topped with a linear prediction layer.

A fairly similar architecture is adopted in [30] for predicting the next activity. The main difference from the one in [84] resides in the presence of ad hoc embedding layers for encoding the activity (and possibly the executor) associated with each input event.

A similar multi-layered LSTM architecture is still used in [77] to predict the next activity of a process instance, which can take account of both continuous and discrete event attributes as input.

Several alternative LSTM-based architectures are defined (and empirically compared) in [7] for the prediction of the activity, timestamp and resource of the next event of a process instance, which take into account both categorical and numerical event properties.

The problem of predicting all the categorical attributes of the next event (including the activity label) is addressed in [56], which proposes an ad hoc "multiway" LSTM-based model, including an attention-like mechanism for combining the outputs of different LTSM stacks (one for each event attribute).

A multi-step approach is proposed in [102] for recommending the next best action (w.r.t. a given target performance indicator) to be performed for a process instance. The

Table 4 Research works that have faced common PM tasks by exploiting deep learning or ensemble learning methods. Works are ordered based on their average number of citations per year. For each work, the table indicates both the (main) PM task addressed and kind of deep/ensemble learning architecture adopted

#	Work Title	#Cit.	Year	Avg. #cit.	PM Task	Approach (and architecture)
1	Predictive business process monitoring with LSTM neural networks [84]	113	2017	28.3	*Prediction (next event)*	*Deep learning (LSTM)*
2	Predicting process behavior using deep learning [30]	95	2017	23.8	*Prediction (next event)*	*Deep learning (LSTM)*
3	Comparing and combining predictive business process monitoring techniques [64]	66	2015	11.0	*Prediction (performances)*	*Ensemble learning (voting-based combination)*
4	An eye into the future: Leveraging a-priori knowledge in predictive business process monitoring [23]	19	2017	4.8	*Prediction (next event/suffix)*	*Deep learning (LSTM)*
5	Learning Accurate LSTM Models of Business Processes [7]	9	2019	4.5	*Prediction (next event/suffix)*	*Deep learning (LSTM)*
6	A Novel Business Process Prediction Model Using a Deep Learning Method [63]	4	2020	4.0	*Prediction (next event)*	*Deep learning (autoen-coders+FCC)*
7	Predicting performances in business processes using deep neural networks [67]	4	2020	4.0	*Prediction (aggregate performances)*	*Deep learning (LSTM,CNN,LSTM+CNN)*
8	MM-Pred: A deep predictive model for multi-attribute event sequence [56]	5	2019	2.5	*Prediction (next event/suffix)*	*Deep learning (LSTM+attention)*
9	Deep learning process prediction with discrete and continuous data features [77]	7	2018	2.3	*Prediction (next event)*	*Deep learning (LSTM)*
10	Activity Prediction of Business Process Instances with Inception CNN Models [24]	4	2019	2	*Prediction (next event)*	*Deep learning (CNN-based Inception net)*

Table 4 continued

#	Work Title	#Cit.	Year	Avg. #cit.	PM Task	Approach (and architecture)
11	Using convolutional neural networks for predictive process analytics [68]	3	2019	1.5	*Prediction (next event)*	*Deep learning (CNN)*
12	A Robust and Versatile Multi- View Learning Framework for the Detection of Deviant Business Process Instances [12]	7	2016	1.4	*Conformance checking (classification-based deviance detection)*	*Ensemble learning (stacking-based combination)*
13	Explainable predictive business process monitoring using gated graph neural networks [46]	1	2020	1	*Prediction (outcome)*	*Deep learning (GGNN)*
14	Predictive Analysis of Business Processes Using Neural Networks with Attention Mechanism [70]	0	2020	0	*Prediction (next event)*	*Deep learning (Transformer)*
15	Prescriptive business process monitoring for recommending next best actions [102]	0	2020	0	*Recommendation (next action)*	*Deep learning (LSTM)*
16	Exploring interpretable predictive models for business processes [79]	0	2020	0	*Prediction (next event)*	*Deep learning (LSTM+attention)*
17	Predictive process mining meets computer vision [69]	0	2020	0	*Prediction (next event)*	*Deep learning (CNN-based Inception net)*
18	Predictive business process monitoring via generative adversarial nets: The case of next event prediction [85]	0	2020	0	*Prediction (next event)*	*Deep learning (GAN)*

approach exploits a DNN as one of its main components, which exhibits essentially the same LSTM-based architecture as in [84].

A-priori knowledge encoded via LTL rules is leveraged in [23], in order to complement an LSTM-based next-activity predictor (previously discovered from historical log data) when forecasting the sequence of activities that will be performed in the future steps of a pre-mortem trace. These a-priori rules are exploited to prune out "invalid" candidate sequences, among those that are computed by iteratively applying the LSTM predictor within a beam search scheme. Notice that this approach could be also considered as an instantiation of Strategy *A* following the $BK4Patterns$ IML modality.

We found a few proposals in the reviewed literature that deviate from the mainstream pattern of using recurrent DNN architectures.

In particular, the approach proposed in [63] combines feature engineering and deep learning methods to improve the prediction of the next activity. To this end, a feed-forward neural network (composed of a stack of multiple dense layers plus a linear prediction one) is trained over fixed-length representations of the input traces, computed via a feature-hashing method (applied to the sequences of activities appearing in the traces).

The three works [24,68,69] all investigates the problem of applying a CNN-based architecture to a 2-D representation of the traces, in order to predict the next activity. Specifically, in [68] a standard convolution approach is applied to an image-like representation of the traces, extracted through an ad hoc featuring engineering step, whereas both [24,69] feature a more sophisticated convolutional architecture based on an Inception block. While both [24,68] convert each trace into a matrix where each column is associated with one event property (e.g., the activity or time), a different encoding is proposed in [69], inspired on the RGB-based scheme commonly adopted in computer vision (with each channel capturing a distinguished property of the trace).

An adversarial learning [40] framework is used in [85] for next activity prediction, which is meant to overcome the difficulty of standard deep learning schemes to generalize well when provided with insufficient training data or when using a sub-optimal network configuration/architecture. Interestingly, the worst-case accuracy of the proposed approach is shown to be at least equal to those of similar non-adversarial ones.

In [67], a method is proposed for predicting process performances, in terms of aggregate performance measures over a future range of time. Different alternative kinds of DNN predictors (based on CNN, LSTM and hybrid CNN-LSTM architectures) are exploited to learn such a prediction function, after discovering a state-based process model from a given log and annotating it with performance measurements (based on a log replay procedure). The annotated performance model is then used to extract a dataset encoding temporally aggregated state/transition performances, which is eventually employed to train the DNN predictor.

DNN models are typically used as complex, black-box function approximators [44], which learn effectively to associate a given input with some output, without providing any explanation/justification on what led to the final decisions. This is clearly bound to raise concerns in scenarios (such as medical diagnosis, planning, control) that need some degree of human supervision, and require machine outputs to be human-understandable (for the sake of experts' validation). In such a case, a DNN should provide interpretable justifications for its output, possibly supplying insights on its inner workings [9].

Special attention is paid to the interpretability of DNN predictors is payed in two recent approaches to predictive process monitoring [46,79]. In particular, an attention-based LSTM model for predicting the next activity of a process instance is proposed in [79], while exploring the possibility of providing insights into those model inputs that have influenced the prediction the most. Since such explanations are extracted directly from the weights of the model, it is claimed in [79] that they should reflect the prediction logics of the DNN better than the results of local post-hoc explanation techniques like LIME [75] and SHAP [57].

In the same vein as [79], the work in [46] tries to make process instances' forecasts more explainable by visualizing how much different activities have impacted on the prediction. To this end, a *Gated Graph Neural Network* (GGNN) is applied to graph-like representations of the process instances—in the graph of each process instance, the nodes and edges correspond to process activities and precedence relationships between activities, respectively. Notably, for any process instance x, information on the relevance of the activities to the prediction returned for x (extracted from the values that the weights of the discovered GGNN take when taking x's representation as input) is presented in the form of a process model.

9 Summary and Discussion: Literature Taxonomy, Opportunities and Open Issues

Summary: a taxonomy for the reviewed literature In this work, we have investigated on how classical PM tasks (i.e., discovery, conformance checking, enhancement, detection, prediction, and recommendation) have taken advantage of complementary AI capabilities (ranging from knowledge representation to machine learning and inference methods), while paying special attention to the challenging case where the log data at hand are low-level, incomplete and/or heterogenous.

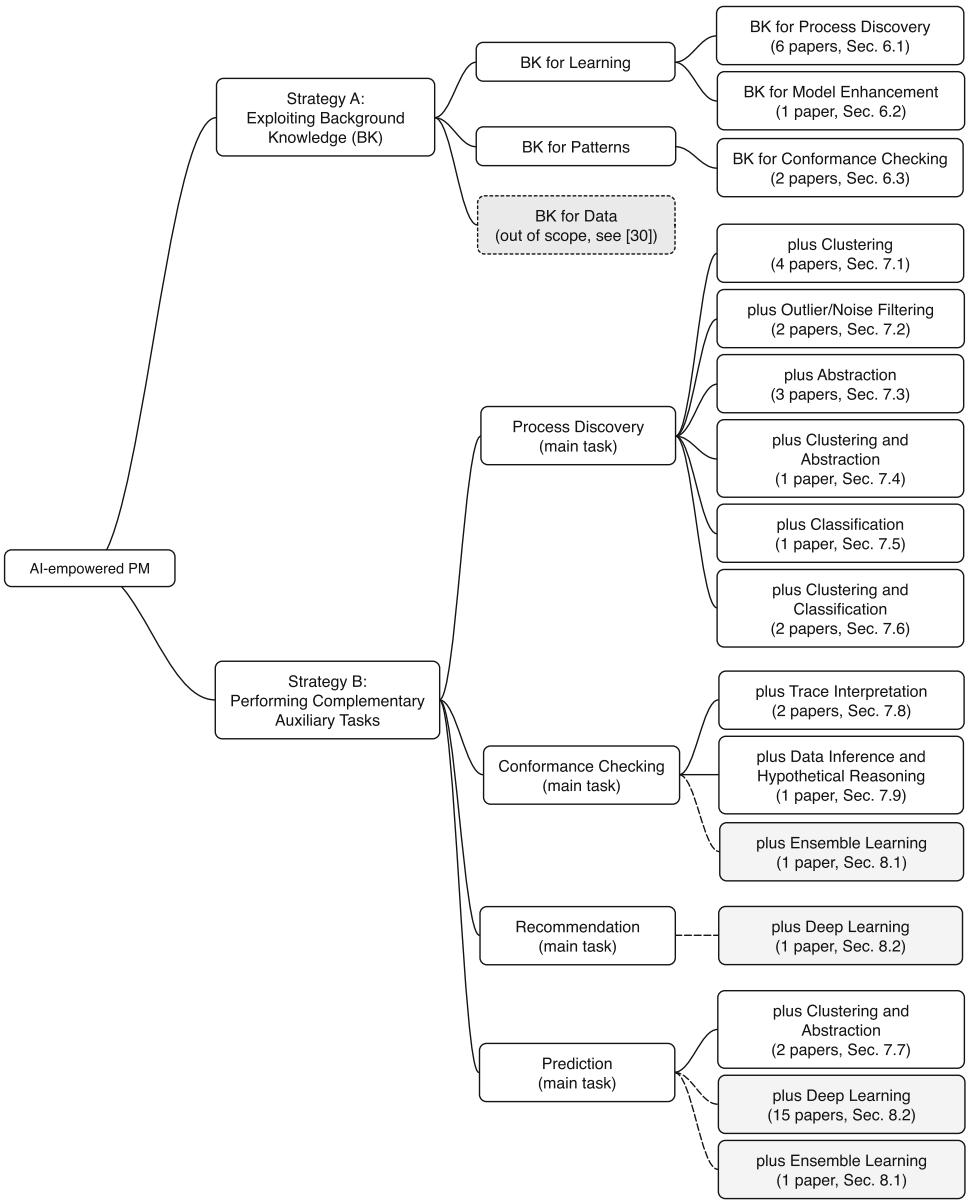

Fig. 3 A taxonomy of the reviewed literature works: shadowed nodes represents subclasses of works that either have not been covered by our study (namely, *BK for Data*, depicted with a dotted border) or constitute emerging kinds of solutions somehow related to certain subclasses of Strategy B. Each leaf class is annotated with the number of works reviewed for the class, and refers to the section of the manuscript that describe them

Specifically, we have focused on two different broad families of approaches, which hinge upon two different strategies for improving the performance of a target PM task: *A*) taking account of user-driven background knowledge; *B*) pursuing one or multiple auxiliary learning/inference tasks jointly with the target one.

A systematic literature review was carried out for both strategies, which entailed the retrieval and analysis of a number of major research works. This study confirmed that both kinds of strategies constitute a valuable means for improving the quality of the results that can be achieved on incomplete, low-level and heterogenous logs, and for enabling some level of operational support for low structured processes.

In order to provide the reader with a final compact view of the research field analyzed, in Fig. 3 we have depicted a taxonomy that summarizes the main different kinds of the approaches reviewed at decreasing (from left to right) levels of abstraction. The leaves in the taxonomy correspond to the homogenous groups of works, addressing a similar kind of problem and adopt similar ways to exploit additional AI capabilities.

For the reader's convenience, each of these leaf nodes refers to the section of the manuscript where the corresponding group of works has been discussed.

A special meaning is associated with the following two kinds of nodes, depicted as shadowed in the figure: *(i)* the subclass of Strategy *A* (namely, *BK for Data*) that has been left out of our review (but that has been covered by another recent survey [26]); *(ii)* groups of recent works, exploiting emerging deep/ensemble learning methods, which can be regarded as a sort of evolution of Strategy *B*.

Related areas, opportunities and open issues The idea of exploiting smart AI capabilities in PM settings links to a wider recent stream of research targeting the development of AI-enabled BPM solutions. For example, two noticeable emerging topics in this area regard: (i) extending BPM systems with Automated-Planning solutions [61], capable of supporting process design (e.g., suggesting new process models/variants), or enabling run-time case adaptation mechanisms (e.g., by reconfiguring the execution of a case to react to incidents/failures); (ii) leveraging prediction-oriented ML methods (currently mainly used to provide per-trace forecasts) in a more extensive way [71], to support better BPM configuration/design phases, or even foster the development of pro-active BPM systems, capable of performing decisions and optimization actions autonomously at run-time.

Such an ambitious goal, however, is difficult to pursue in contexts featuring low-quality event data, and limited/uncertain knowledge on the behavior of the process and of its surrounding environment. In particular, the desire to exploit a model discovered automatically as a basis for human/automated decisions and actions poses a number of delicate issues, which concern the validity (in terms of accuracy/correctness, reliability/robustness and accountability) of the model and of the entire data analytics process that produced it.

In fact, the validity of the models (especially in terms of generalization power) and of the inference results returned by PM tools strongly depends both on the quality of the log data used as input, and on the expertise of the analyst. Different kinds of bias can undermine the validity of a PM model in real-life projects and lead to faulty conclusions/predictions/decisions: (i) analyst's bias (e.g., pertaining to data preparation, or PM algorithm selection/configuration); (ii) statistical data-level bias (e.g., low representative logs); (iii) algorithmic/model bias (e.g., concerning the hypothesis space or search heuristics); (iv) evaluation bias (e.g., coming from the validation data/criteria used).

The first step toward making the use of PM tools more aware of these validity threats consists in trying to increase the level of transparency of PM analyses for the sake of validation/debugging, by both making the model biases explicit and producing explainable models/inferences. Interestingly, several works discussed in Sects. 6 and 7 have already moved in this direction.

In particular, some kinds of constraints (e.g., declarative process models [96]) or precedence constraints [32,33,39,42]) adopted in Strategy *A* methods constitute a formal way to put domain knowledge, business requirements and analyst's preferences/beliefs into a well-structured unambiguous form that can be easily inspected and validated. Moreover, as this background knowledge is used to guide the PM tasks (in the case of $BK4Patterns$ and $BK4Learning$ methods), it also helps interpret and evaluate the results of these tasks.

On the other hand, many existing methods following Strategy *B* can capture hidden structures in the data, such as trace/event clusters and activity patterns, which they mainly exploit as high-level features for the target PM task at hand. In several cases, such pieces of domain information discovered automatically are accompanied by readable descriptions, (e.g., the cluster-wise process models in [13,20,37,43], the trace/event clustering rules in [36,37], and the hierarchies of pattern abstractions in [6,55]), which can inspire new data preparation and mining steps (e.g., fine-grain analyses on interesting trace clusters), or serve as a basis for producing novel explicit background knowledge (e.g., event classifiers, activity taxonomies, process/context variants). This way, a virtuous circle is enabled, where knowledge discovered from log data can both inspire new data analytics steps and help extend/revise the base of domain knowledge, while the latter, in its turn, allows for better guiding and documenting novel PM sessions. Notably, the possibility of distilling novel data-driven knowledge from discovered models proves particularly useful in many real-life dynamic contexts where building up and maintaining a rich and updated enough base of background knowledge is not easy, owing to the lack of systematic, regular and effective interactions between domain experts, on the one hand, and data analysts and data analytics tools, on the other.

However, in dynamic/unstructured BPM scenarios it is unrealistic to assume all relevant human objectives/constraints to be encoded explicitly in a definite way, but it may be easier to get qualitative/quantitative feedback on the quality of PM results and of PM-based decisions/actions, directly from different process stakeholders (and possibly from the clients of process executions) or via performance/QoS indicators computed automatically. Methods developed in the field of Reinforcement Learning (RL) [80] offer a solid basis for devising ways to exploit feedbacks coming from BPM environment (including analysts, experts and process stakeholders) in a systematic way, in order to allow smart AI-based BPM tools to make their inferences, predictions, recommendations and decisions/actions as aligned as possible to the human desiderata and beliefs (even though the latter are not fully encoded in terms of objectives and constraints). Such an extension of PM/BPM frameworks has not received adequate attention in the literature so far, and it might be a valuable direction of future research.

Interestingly, the adoption of RL schemes and a synergistic use of learning and reasoning methods are two complementary technical solutions that can prove very useful in the appealing perspective of cognitive BPM systems [65], which are meant to continuously improve their levels of performance and of context-awareness through a continuous "plan-act-learn" cycle—where new knowledge and skills are learnt through the analysis of their interactions with the environment and of novel (event/sensor) data.

Finally, let us provide a brief discussion of some challenges and opportunities that specifically concern the design of Deep Learning (DL) methods for PM, which is gaining momentum, especially in the discovery of classification and forecasting models for predictive process monitoring (see the last part of Sect. 7), and in the detection of anomalous/noisy log data (see, e.g., [66]). As mentioned above, this success mainly descends from the capability of Deep Neural Networks (DNNs) to learn effective hierarchical features for complex data (as are log traces, because of their sequential and multi-dimensional nature) automatically. It is reasonable to expect this trend to grow in the near future, owing to the flexibility of DNN architectures, the availability of many consolidated DL libraries providing diverse reusable built-in neural network models/layers. In particular, these capabilities can prove very useful for devising novel more advanced approaches of Strategy *B* (e.g., implementing multi-task, transfer and/or semi-supervised learning schemes), compared to those described in Sect. 7.

However, as discussed in Sect. 8, the higher expressivity and flexibility of DNNs come at a cost: the internal data representation and decision logics of these models are hard to interpret and to control. This may be a serious obstacle for the acceptance of such solutions in real application contexts, considering the widespread demand for transparent, trustable and accountable AI/ML.

Interestingly, the rapidly growing body of research on Explainable DL [44] and on *interpretable deep neural networks* [9] is expected to help mitigate this critical issue—in fact, the two very recent works [46,79], reviewed in Sect. 8, constitute two initial steps in this direction within the field of Process Mining.

It is also worth noticing that recent advancements on the long-standing attempt to extend ANN methods with the capability to use Informed-PM methods that can exploit guidance from the analyst/expert.

Acknowledgements This work was partially supported by the European Commission funded project "Humane AI: Toward AI Systems That Augment and Empower Humans by Understanding Us, our Society and the World Around Us" (grant # 820437). The support is gratefully acknowledged.

References

1. Augusto A, Conforti R, Dumas M, La Rosa M, Maggi FM, Marrella A, Mecella M, Soo A (2018) Automated discovery of process models from event logs: review and benchmark. IEEE TKDE 31(4):686–705
2. Baier T, Mendling J, Weske M (2014) Bridging abstraction layers in process mining. Inf Syst 46:123–139
3. Bolt A, van der Aalst WM (2015) Multidimensional process mining using process cubes. In: Enterprise, business-process and information systems modeling, pp 102–116
4. Bose RJC, Van der Aalst WM (2009) Abstractions in process mining: a taxonomy of patterns. In: BPM, pp 159–175
5. Bose RJC, Mans RS, van der Aalst WM (2013) Wanna improve process mining results? In: IEEE Symposium CIDM, pp 127–134
6. Bose RJC, Verbeek EH, van der Aalst WM (2011) Discovering hierarchical process models using prom. In: International conference on advanced information systems engineering. Springer, pp 33–48
7. Camargo M, Dumas M, González-Rojas O (2019) Learning accurate LSTM models of business processes. In: International conference on business process management. Springer, pp 286–302
8. Caruana R (1997) Multitask learning. Mach Learn 28(1):41–75
9. Chakraborty S, Tomsett R, Raghavendra R, Harborne D, Alzantot M, Cerutti F, Srivastava M, Preece A, Julier S, Rao RM et al. (2017) Interpretability of deep learning models: a survey of results. In: 2017 IEEE SmartWorld, ubiquitous intelligence & computing, advanced & trusted computed, scalable computing & communications, cloud & big data computing, internet of people and smart city innovation, pp 1–6
10. Chapman P, Clinton J, Kerber R, Khabaza T, Reinartz T, Shearer CRH, Wirth R (2000) Crisp-dm 1.0: Step-by-step data mining guide
11. Conforti R, La Rosa M, ter Hofstede AH (2016) Filtering out infrequent behavior from business process event logs. IEEE Trans Knowl Data Eng 29(2):300–314
12. Cuzzocrea A, Folino F, Guarascio M, Pontieri L (2016) A robust and versatile multi-view learning framework for the detection of deviant business process instances. Int J Coop Inf Syst 25(4):1–56
13. Cuzzocrea A, Folino F, Guarascio M, Pontieri L (2017) Deviance-aware discovery of high quality process models. In: ICTAI, pp 724–731
14. Cuzzocrea A, Folino F, Guarascio M, Pontieri L (2019) Predictive monitoring of temporally-aggregated performance indicators of business processes against low-level streaming events. Inf Syst 81:236–266
15. De Medeiros AA, van der Aalst W, Pedrinaci C (2008) Semantic process mining tools: core building blocks. In: ECIS, pp 1953–1964
16. De Medeiros AKA, Guzzo A, Greco G, Van Der Aalst WM, Weijters A, Van Dongen BF, Saccà D (2007) Process mining based on clustering: A quest for precision. In: International conference on business process management, pp 17–29
17. De Weerdt J, vanden Broucke SK, Vanthienen J, Baesens B (2012) Leveraging process discovery with trace clustering and text mining for intelligent analysis of incident management processes. In: 2012 IEEE congress on evolutionary computation, pp 1–8
18. dos Santos Garcia C, Meincheim A, Junior ERF, Dallagassa MR, Sato DMV, Carvalho DR, Santos EAP, Scalabrin EE (2019) Process mining techniques and applications-a systematic mapping study. Expert Syst Appl
19. dos Santos Garcia C et al (2019) Process mining techniques and applications—a systematic mapping study. Expert Syst Appl 133:260–295
20. De Weerdt J, Vanden Broucke S, Vanthienen J, Baesens B (2013) Active trace clustering for improved process discovery. IEEE TKDE 25(12):2708–2720
21. Desel J, Esparza J (2005) Free choice Petri nets, vol 40. Cambridge University Press, Cambridge
22. Di Francescomarino C, Ghidini C, Maggi FM, Milani F (2018) Predictive process monitoring methods: Which one suits me best? In: BPM, pp 462–479
23. Di Francescomarino C, Ghidini C, Maggi FM, Petrucci G, Yeshchenko A (2017) An eye into the future: leveraging a-priori knowledge in predictive business process monitoring. In: BPM, pp 252–268

24. Di Mauro N, Appice A, Basile TM (2019) Activity prediction of business process instances with inception cnn models. In: International conference of the italian association for artificial intelligence, pp 348–361
25. Diamantini C, Genga L, Potena D (2016) Behavioral process mining for unstructured processes. J Intell Inf Syst 47(1):5–32
26. Diba K, Batoulis K, Weidlich M, Weske M (2019) Extraction, correlation, and abstraction of event data for process mining. WIREs Data Min Knowl Discov. https://doi.org/10.1002/widm.1346
27. Dixit P, Buijs JC, van der Aalst WM, Hompes B, Buurman J (2015) Using domain knowledge to enhance process mining results. In: SIMPDA, pp 76–104
28. Dixit PM, Buijs JC, van der Aalst WM (2018) Prodigy: Human-in-the-loop process discovery. In: 2018 12th international conference on research challenges in information science (RCIS), pp 1–12
29. Dixit PM, Verbeek H, Buijs JC, van der Aalst WM (2018) Interactive data-driven process model construction. In: International conference on conceptual modeling, pp 251–265
30. Evermann J, Rehse JR, Fettke P (2017) Predicting process behaviour using deep learning. Decis Support Syst 100:129–140
31. Fazzinga B, Flesca S, Furfaro F, Masciari E, Pontieri L (2018) Efficiently interpreting traces of low level events in business process logs. Inf Syst 73:1–24
32. Fazzinga B, Flesca S, Furfaro F, Pontieri L (2018) Online and offline classification of traces of event logs on the basis of security risks. J Intell Inf Syst 50(1):195–230
33. Fazzinga B, Flesca S, Furfaro F, Pontieri L (2018) Process discovery from low-level event logs. In: CAISE, pp 257–273
34. Folino F, Folino G, Guarascio M, Pontieri L (2019) Learning effective neural nets for outcome prediction from partially labelled log data. In: 31st IEEE international conference on tools with artificial intelligence (ICTAI)
35. Folino F, Greco G, Guzzo A, Pontieri L (2011) Mining usage scenarios in business processes: outlier-aware discovery and run-time prediction. Data Knowl Eng 70(12):1005–1029
36. Folino F, Guarascio M, Pontieri L (2014) Mining predictive process models out of low-level multidimensional logs. In: CAiSE, pp 533–547
37. Folino F, Guarascio M, Pontieri L (2015) Mining multi-variant process models from low-level logs. In: International conference on business information systems (BIS), pp 165–177
38. García-Bañuelos L, Dumas M, La Rosa M, De Weerdt J, Ekanayake CC (2014) Controlled automated discovery of collections of business process models. Inf Syst 46:85–101
39. Goedertier S, Martens D, Vanthienen J, Baesens B (2009) Robust process discovery with artificial negative events. J Mach Learn Res 10:1305–1340
40. Goodfellow IJ, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville AC, Bengio Y (2014) Generative adversarial nets. In: Annual conference on neural information processing systems (NIPS), pp 2672–2680
41. Gori M (2017) Machine Learning: a constraint-based approach. Morgan Kaufm
42. Greco G, Guzzo A, Lupia F, Pontieri L (2015) Process discovery under precedence constraints. TKDD 9(4):32:1-32:39
43. Greco G, Guzzo A, Pontieri L, Saccà D (2006) Discovering expressive process models by clustering log traces. IEEE TKDE 18(8):1010–1027
44. Guidotti R, Monreale A, Ruggieri S, Turini F, Giannotti F, Pedreschi D (2018) A survey of methods for explaining black box models. ACM Comput Surv (CSUR) 51(5):1–42
45. Günther CW, Van Der Aalst WM (2007) Fuzzy mining-adaptive process simplification based on multi-perspective metrics. In: BPM, pp 328–343
46. Harl M, Weinzierl S, Stierle M, Matzner M (2020) Explainable predictive business process monitoring using gated graph neural networks. J Decis Syst 1–16
47. Janiesch C, Koschmider A, Mecella M, Weber B, Burattin A, Di Ciccio C, Fortino G, Gal A, Kannengiesser U, Leotta F et al (2020) The internet of things meets business process management: a manifesto. IEEE Syst Man Cybern Mag 6(4):34–44
48. Kitchenham B (2004) Procedures for performing systematic reviews. Keele UK Keele University 33(2004):1–26
49. Koller D, Friedman N, Džeroski S, Sutton C, McCallum A, Pfeffer A, Abbeel P, Wong MF, Heckerman D, Meek C et al (2007) Introduction to statistical relational learning. MIT Press, Cambridge
50. Leemans SJ, Fahland D, van der Aalst WM (2013) Discovering block-structured process models from event logs containing infrequent behaviour. In: International conference on business process management, pp 66–78
51. Leemans SJ, Fahland D, van der Aalst WM (2014) Exploring processes and deviations. In: International conference on business process management. Springer, pp 304–316
52. Leno V, Dumas M, Maggi FM (2018) Correlating activation and target conditions in data-aware declarative process discovery. In: International conference on business process management, pp 176–193
53. Leotta F, Mecella M, Mendling J (2015) Applying process mining to smart spaces: perspectives and research challenges. In: Persson A, Stirna J (eds) Advanced information systems engineering workshops, pp 298–304
54. Letia IA, Goron A (2015) Model checking as support for inspecting compliance to rules in flexible processes. J Vis Lang Comput 28:100–121
55. Li J, Bose RJC, van der Aalst WM (2010) Mining context-dependent and interactive business process maps using execution patterns. In: International conference on business process management, pp 109–121
56. Lin L, Wen L, Wang J (2019) MM-ORED: a deep predictive model for multi-attribute event sequence. In: Proceedings of the 2019 SIAM international conference on data mining. SIAM, pp 118–126
57. Lundberg SM, Lee SI (2017) A unified approach to interpreting model predictions. In: Advances in neural information processing systems, pp 4765–4774
58. Mannhardt F, de Leoni M, Reijers HA, van der Aalst WM (2017) Data-driven process discovery-revealing conditional infrequent behavior from event logs. In: International conference on advanced information systems engineering, pp 545–560
59. Mannhardt F, de Leoni M, Reijers HA, van der Aalst WM, Toussaint PJ (2018) Guided process discovery-a pattern-based approach. Inf Syst 76:1–18
60. Mariscal G, Marban O, Fernandez C (2010) A survey of data mining and knowledge discovery process models and methodologies. Knowl Eng Rev 25(2):137–166
61. Marrella A (2017) What automated planning can do for business process management. In: International conference on business process management. Springer, pp 7–19
62. Martín-Martín A, Orduna-Malea E, Thelwall M, López-Cózar ED (2018) Google scholar, web of science, and scopus: a systematic comparison of citations in 252 subject categories. J Inf 12(4):1160–1177
63. Mehdiyev N, Evermann J, Fettke P (2018) A novel business process prediction model using a deep learning method. Bus Inf Syst Eng 1–15
64. Metzger A, Leitner P, Ivanović D, Schmieders E, Franklin R, Carro M, Dustdar S, Pohl K (2014) Comparing and combining predictive business process monitoring techniques. IEEE Trans Syst Man Cybern Syst 45(2):276–290

65. Nezhad HRM, Akkiraju R (2014) Towards cognitive BPM as the next generation BPM platform for analytics-driven business processes. In: International conference on business process management. Springer, pp 158–164
66. Nolle T, Luettgen S, Seeliger A, Mühlhäuser M (2019) Binet: multi-perspective business process anomaly classification. Inf Syst 101458
67. Park G, Song M (2020) Predicting performances in business processes using deep neural networks. Decis Support Syst 129:113191
68. Pasquadibisceglie V, Appice A, Castellano G, Malerba D (2019) Using convolutional neural networks for predictive process analytics. In: 2019 International conference on process mining (ICPM), pp 129–136
69. Pasquadibisceglie V, Appice A, Castellano G, Malerba D (2020) Predictive process mining meets computer vision. In: International conference on business process management (BPM), pp 176–192
70. Philipp P, Jacob R, Robert S, Beyerer J (2020) Predictive analysis of business processes using neural networks with attention mechanism. In: 2020 International conference on artificial intelligence in information and communication (ICAIIC), pp 225–230
71. Poll R, Polyvyanyy A, Rosemann M, Röglinger M, Rupprecht L (2018) Process forecasting: towards proactive business process management. In: International conference on business process management. Springer, pp 496–512
72. Prat N (2019) Augmented analytics. Bus Inf Syst Eng 61(3):375–380
73. Raedt LD, Nijssen S, O'Sullivan B, Hentenryck PV (2011) Constraint programming meets machine learning and data mining. Dagstuhl Rep 1(5):61–83
74. Rembert AJ, Omokpo A, Mazzoleni P, Goodwin RT (2013) Process discovery using prior knowledge. In: International conference on service-oriented computing, pp 328–342
75. Ribeiro MT, Singh S, Guestrin C (2016) "Why should i trust you?" explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, pp 1135–1144
76. Rogge-Solti A, Weske M (2013) Prediction of remaining service execution time using stochastic Petri nets with arbitrary firing delays. In: Basu S, Pautasso C, Zhang L, Fu X (eds) Service-oriented computing, pp 389–403
77. Schönig S, Jasinski R, Ackermann L, Jablonski S (2018) Deep learning process prediction with discrete and continuous data features. In: Proceedings of the 13th international conference on evaluation of novel approaches to software engineering, pp 314–319
78. Senderovich A, Shleyfman A, Weidlich M, Gal A, Mandelbaum A (2016) P3-folder: optimal model simplification for improving accuracy in process performance prediction. In: BPM, pp 418–436
79. Sindhgatta R, Moreira C, Ouyang C, Barros A (2020) Exploring interpretable predictive models for business processes. In: International conference on business process management, pp 257–272
80. Sugiyama M (2015) Statistical reinforcement learning: modern machine learning approaches. Chapman and Hall/CRC, Boca Raton
81. Suriadi S, Andrews R, ter Hofstede A, Wynn M (2017) Event log imperfection patterns for process mining. Inf Syst 64(C):132–150
82. Suriadi S, Andrews R, ter Hofstede AHM, Wynn MT (2017) Event log imperfection patterns for process mining: towards a systematic approach to cleaning event logs. Inf Syst 64:132–150
83. Tax N, Sidorova N, Haakma R, van der Aalst WM (2016) Event abstraction for process mining using supervised learning techniques. In: SAI IntelliSys, pp 251–269
84. Tax N, Verenich I, Rosa ML, Dumas M (2017) Predictive business process monitoring with LSTM neural networks. In: 29th International conference on advanced information systems engineering (CAISE'17), pp 477–492
85. Taymouri F, Rosa ML, Erfani S, Bozorgi ZD, Verenich I (2020) Predictive business process monitoring via generative adversarial nets: the case of next event prediction. In: Business process management
86. van der Aa H, Leopold H, Weidlich M (2020) Partial order resolution of event logs for process conformance checking. Decis Support Syst 136:113347
87. van der Aalst W (2010) Business process simulation revisited. In: Workshop on enterprise and organizational modeling and simulation, pp 1–14
88. van der Aalst W et al (2011) Process mining manifesto. In: BPM, pp 169–194
89. van der Aalst WMP (2011) Process mining: discovery, conformance and enhancement of business processes. Springer, Berlin
90. Van Der Aa H, Leopold H, Reijers HA (2019) Efficient process conformance checking on the basis of uncertain event-to-activity mappings. IEEE Trans Knowl Data Eng 32(5):927–940
91. vanden Broucke SKLM, Weerdt JD (2017) Fodina: a robust and flexible heuristic process discovery technique. Decis Support Syst 100:109–118
92. Vanschoren J (2019) Meta-learning. Springer, Berlin, pp 35–61
93. Verenich I, Dumas M, Rosa ML, Maggi FM, Teinemaa I (2019) Survey and cross-benchmark comparison of remaining time prediction methods in business process monitoring. ACM TIST 10(4):34
94. van Dongen BF et al. (2005) The ProM framework: a new era in process mining tool support. In: International conference on application and theory of petri nets, pp 444–454
95. van Eck ML, Lu X, Leemans SJ, van der Aalst WM (2015) Pm2: a process mining project methodology. In: International conference on advanced information systems engineering. Springer, pp 297–313
96. van Der Aalst WM, Pesic M, Schonenberg H (2009) Declarative workflows: balancing between flexibility and support. Comput Sci Res Dev 23(2):99–113
97. von Rueden L, Mayer S, Garcke J, Bauckhage C, Schuecker J (2019) Informed machine learning-towards a taxonomy of explicit integration of knowledge into machine learning. arXiv preprint arXiv:1903.12394
98. Wang RY, Strong DM (1996) Beyond accuracy: what data quality means to data consumers. J Manag Inf Syst 12(4):5–33
99. Watson HJ, Wixom BH (2007) The current state of business intelligence. Computer 40(9):96–99
100. Weijters A, Ribeiro J (2011) Flexible heuristics miner (FHM). In: 2011 IEEE symposium on computational intelligence and data mining (CIDM). IEEE, pp 310–317
101. Weijters AJMM, van der Aalst WMP (2003) Rediscovering workflow models from event-based data using Little Thumb. Integr Comput-Aided Eng 10(2):151–162
102. Weinzierl S, Dunzer S, Zilker S, Matzner M (2020) Prescriptive business process monitoring for recommending next best actions. In: International conference on business process management, pp 193–209
103. Yahya BN, Song M, Bae H, Sul So, Wu JZ (2016) Domain-driven actionable process model discovery. Comput Ind Eng 99:382–400