Journal of Intelligent Manufacturing (2023) 34:57–83
https://doi.org/10.1007/s10845-021-01903-y

Data-driven dynamic causality analysis of industrial systems using interpretable machine learning and process mining

Karim Nadim[1,2] · **Ahmed Ragab**[1,2,3] · **Mohamed-Salah Ouali**[1]

Received: 12 March 2021 / Accepted: 22 December 2021 / Published online: 11 May 2022

Abstract

The complexity of industrial processes imposes a lot of challenges in building accurate and representative causal models for abnormal events diagnosis, control and maintenance of equipment and process units. This paper presents an innovative data-driven causality modeling approach using interpretable machine learning and process mining techniques, in addition to human expertise, to efficiently and automatically capture the complex dynamics of industrial systems. The approach tackles a significant challenge in the causality analysis community, which is the discovery of high-level causal models from low-level continuous observations. It is based on the exploitation of event data logs by analyzing the dependency relationships between events to generate accurate multi-level models that can take the form of various state-event diagrams. Highly accurate and trustworthy patterns are extracted from the original data using interpretable machine learning integrated with a model enhancement technique to construct event data logs. Afterward, the causal model is generated from the event log using the inductive miner technique, which is one of the most powerful process mining techniques. The causal model generated is a Petri net model, which is used to infer causality between important events as well as a visualization tool for real-time tracking of the system's dynamics. The proposed causality modeling approach has been successfully tested based on a real industrial dataset acquired from complex equipment in a Kraft pulp mill located in eastern Canada. The generated causality model was validated by ensuring high model fitness scores, in addition to the process expert's validation of the results.

Keywords Causality analysis · Interpretable machine learning · Process mining · Petri nets · Discrete event systems · Supervisory control

Introduction

One of the major contributing factors to the higher rates of Greenhouse Gas (GHG) emissions of large-scale industrial facilities is inefficient monitoring, which leads to imprecise control activities causing higher frequency of faults and maintenance activities, poor adaptability to system uncertainties, and excessive energy consumption. Among these complex processes are large final emitters (LFEs) found in process industries such as oil refineries, steel & iron production, the cement industry, chemical processes, and pulp & paper mills (Talbot & Boiral, 2013). Those facilities require massive amounts of energy and they emit an average of 50,000 tons or more of GHG, specifically carbon dioxide (CO2) per annum (Climate Change Connection, 2018). In these large-scale facilities, building precise and representative causal models that can handle the fast-evolving dynamics and the high level of interactions between controllers becomes significantly challenging, time-consuming, and economically expensive in addition to the detailed and extensive subjective knowledge required from process experts and operators (Sun et al. 2021) and (Ragab et al., 2014).

Accordingly, accurate monitoring and root cause analysis tools are essential to promote more efficient supervisory

✉ Ahmed Ragab
ahmed.ragab@polymtl.ca; ahmed.ragab@canada.ca

Karim Nadim
karim.nadim@polymtl.ca; karim.nadim@canada.ca

Mohamed-Salah Ouali
mohamed-salah.ouali@polymtl.ca

[1] Department of Mathematics and Industrial Engineering, Polytechnique Montréal, Montréal, QC H3T 1J4, Canada

[2] CanmetENERGY – Natural Resources Canada (NRCan), Varennes, QC J3X 1P7, Canada

[3] Faculty of Electronic Engineering, Menoufia University, Menouf 32952, Egypt

control in these processes, which in return can have a positive impact on energy consumption rates, non-eco-friendly emissions, maintenance costs, and overall system safety (Naghoosi et al. 2013; Lindner et al., 2017). More specifically, there is a need to develop a concrete and representative causality modeling approach that captures the system's complex dynamics, states evolution, as well as the interactions between the controllers. The approach should also identify the different factors that cause failures, performance deterioration, and their root causes to enhance monitoring and control tasks and, finally, to allow energy-efficient operations.

Causality analysis has become a vital research topic in the industrial community with various applications including fault diagnosis and prognosis, and others (Chen et al., 2018; Chiang et al., 2015; Lindner et al., 2019). In general, several methods have been introduced in the literature for causality modeling, and they can be mainly divided into knowledge-based and data-based methods (Chiang & Braatz, 2003). The former depends on existing knowledge based on expertise and physical models of the system, and they usually require a deep understanding of the process in order to have an accurate model of the system (Ragab et al., Ragab, El Koujok, et al., 2019). However, as modern system architectures become more complex and the number of components significantly increases, the process of formulating an accurate model based on expertise and domain knowledge becomes very difficult, which deems most knowledge-based methods have some limitations when applied in large-scale industrial processes. The latter is based on the exploitation of available data to construct models by analyzing causality relationships between process variables. The data-driven causality analysis methods have attracted several industrial applications (Yang et al. 2014). However, based on an investigation of the literature, we found that existing data-driven causality analysis methods face challenges in complex industrial processes as follows.

- Firstly, most of these methods are static and face difficulties in capturing the temporal relations between events when applied to systems with a large number of variables and interacting components, where the computational demand becomes excessive (Nauta et al. 2018). Moreover, most of these methods make simplifying assumptions (Lee & Chien, 2020), causing inaccurate results (Elhefnawy et al. 2021).
- Most of the existing methods do not incorporate events' sequential information and they mostly focus on finding the direct root cause for a phenomenon of interest such as a fault in a machine. However, in an industrial process, we are more interested in finding the causes for any performance deterioration and not necessarily faults only. This is when the analysis of the sequence of events becomes crucial, as it can reveal the effect of the order in which specific events occurred on the state evolution of the system, which is important to accurately represent the system's dynamics and to infer the correct causality relationships (Kabir, 2017a). An example is the occurrence of an event, which might not cause deterioration, however, if that event occurs in a specific sequence with other events, then the overall performance of the process can be impacted.
- Based on insights from industrial partners, existing methods can become inefficient because it is insufficient to infer causality between variables separately (change in variable X1 causes a change in X2). The reason is that in complex processes such as recovery boilers, there are hidden interactions between controllers and it is extremely important to identify those interactions and analyze their effect on the system's KPIs (Ragab, El Koujok, et al., 2019; Ragab, Yacout, et al., 2019; Derigent et al. 2020).
- Existing methods deal with abnormal global events without incorporating normal operations (Li et al., 2016), which can be important to guide the operator to actions that avoid performance deterioration or that return the system back to normality.

These challenges cause the existing data-driven causality analysis methods to perform inefficiently in complex equipment found in many process industries. The recovery boilers, heat exchanger, evaporators and bleaching equipment in pulp & paper mills are examples of these kinds of equipment. This type of complex equipment has several interacted variables and are identified as highly dynamical nonlinear systems.

Accordingly, there is a clear need to develop accurate and representative causal models for such complex industrial systems. This paper proposes an innovative data-driven dynamic causality analysis approach based on interpretable machine learning (IML) (Molnar, 2020), process mining (PM) (Van der Aalst, 2016), and human expertise, to fill the discussed research gaps. The approach exploits historical data collected from the process to automatically construct an accurate and representative dynamic causal model.

The proposed approach aims at overcoming the above-mentioned challenges that face existing methods in industrial applications. This work is also motivated by solving one of the big challenges in the Artificial Intelligence (AI) and causality analysis communities, which is "*the discovery of high-level causal variables from low-level observations*" (Schölkopf et al., 2021). This approach focuses on building a dynamic causal model between high-level events rather than between the low-level variables. In fact, this can be beneficial to several industrial applications, where insights and reasoning are sought on important events that occur in complex processes and their effect on the system's key performance indicators (KPIs). We attempt to solve this challenge in an innovative way by adopting PM, to automatically build

causal models, without heavily involving the process expert. This allows the expert to analyze the temporal relations and the dependencies between events from a higher level. PM links data science and process science to model processes through generating state-event graph models, analyzing bottlenecks, and suggesting process improvement. It is based on exploiting discrete event log data solely, without the need for domain knowledge by analyzing the dependency relationship between events.

PM has been applied widely and successfully in business processes. However, to the best of our knowledge it has not been yet applied to complex industrial processes. One main reason is that most of these industrial systems do not possess well-structured discrete event logs. Generally, data collected in these processes contains low-level observations acquired from sensors with continuous values of KPIs, control setpoints, and measured variables. Therefore, to build discrete event logs that are exploitable by PM, we integrate IML in this work to extract patterns from those low-level continuous datasets. The patterns comprise combinations of conditions on the system's variables, including the controllable variables. Therefore, these patterns can model the interactions between the controllers and their effect on the system's KPIs. More importantly, these patterns actually represent the discrete events that occur in the process. Thus, the patterns are used as representative events for these complex processes, which can be exploited by PM techniques to build high-level causal models in the form of state-event graphs. In this paper, we build these models in the form of Petri nets (PNs) which are one of the powerful tools for modeling and visualizing discrete event systems (DES). More significantly, building the causal model in the form of PNs does not only allow inferring causal relations between important events but can also support tracking the state evolution of complex systems in real-time. This is due to the dynamic modeling capabilities of PNs, which allow monitoring of the system trajectory and state transitions. In addition, the reachability graph of the PN causal model can be regarded as an event-driven process, thus facilitating a suitable environment for applying intelligent supervisory control approaches using machine learning (ML) techniques such as reinforcement learning (RL). This can add an essential element of robustness and reasoning to the control approach, where more insights can be realized from analyzing the consequence of actions (or events) through the causal model (Schölkopf et al., 2021).

It is worth mentioning that this approach is bi-directional, where the causal model built from the low-level variables comprises the high-level discrete events and their sequential relations. At the same time, each event in the model can be broken down to its corresponding pattern and back to the low-level variables that define the pattern.

The proposed approach comprises two phases. In the first phase the dataset with continuous variables and KPIs collected from an industrial process is transformed into an event log. The event log is created by labeling the data using a classification criterion based on the KPIs, and afterwards, patterns are extracted using decision trees (Quinlan, 2014), which is one of the top ten machine learning algorithms (Wu et al., 2007), as an IML technique. These patterns represent the different events that occur in the system in a sequential manner. In the second phase, PM techniques are applied to automatically construct a dynamic causal model from the created event log. The dependency relationships are analyzed between the events in the log to incorporate the temporal information and a state-event diagram is generated. A Quantitative analysis is performed on the model to ensure high accuracy and to allow for causality inference.

The contributions of this paper are summarized as follows.

- IML and PM are integrated to generate high-level causal models for complex industrial processes in the form of state-event graphs (e.g. Petri nets). The causal model is built automatically from low-level sensory data without heavily involving the process expert going through difficult, labor-intensive and time-consuming tasks. The role of the expert is to supervise the modeling procedures, verify and validate the generated model.
- By using IML to extract the patterns from the continuous observations, the approach is able to model the interactions between the controllers and identify the most important events (both normal and abnormal) that affect the performance of the system. These patterns are exploited as events to form the causal model, thus inferring causality on a higher level and not between low-level variables.
- The exploitation of PM techniques allows generating a dynamic causal model in the form of PNs that incorporates temporal information, and the sequential relations between events, which are important in finding the causes for KPI-related changes. The PN model allows for tracking the system's state (based on KPIs) in real-time, as well as the events that cause state transitions.
- The generated model can be easily updated upon system changes and the acquisition of new data through the use of PM enhancement techniques, without depending on the process experts for tedious remodeling tasks. This allows for online real-time process simulation and fast remodeling tasks and also serves as a dashboard for operators. The approach is applied and validated on an industrial dataset collected from a recovery boiler in a pulp & paper mill.

The rest of the paper is organized into six sections as follows. Section 2 presents the related works for causality analysis in industrial processes. The preliminaries involving state-event graphs (Petri nets) and an illustration for the process mining approach are represented in Sect. 3. The proposed causality analysis methodology is detailed in Sect. 4.

Section 5 comprises a case study of a recovery boiler system of a pulp & paper mill located in eastern Canada and presents the results obtained from the proposed methodology. Discussion and future work directions are presented in Sect. 6, followed by a conclusion in Sect. 7.

Related work: causality analysis methods in industrial processes

Causality analysis, which is a major monitoring activity, has become a vital research topic in the industrial community (Yang et al., 2014). Causality analysis can provide the operators with the knowledge and process insights to support decision-making in fault prevention. In general, the traditional causality analysis approach was based on planned or randomized experiments, where for example, an input variable would be manipulated in a process, while other variables are fixed and the outcome (variable of interest) would be observed to conclude if there is a causal relationship between the input variable and the outcome. However, this approach is time-consuming and sometimes infeasible in complex processes, due to the massive number of variables and also for safety measures where experimentation is not tolerated (Spirtes, 2010). Consequently, attention was shifted to searching for causal models based on background knowledge and validating them using statistical tests, as well as data-driven methods that can infer causality relationships between process variables by analyzing sensory data using machine learning techniques.

This section presents a background on causality analysis methods in the literature. As mentioned in the introduction section, causality modeling methods in the literature can be divided into knowledge-based and data-based methods (Chiang & Braatz, 2003). Maurya et al. have devolved knowledge-based modeling approaches based on the system's mathematical equations (Maurya et al., 2003, 2004, 2006), as well as instrumentation diagrams (Thambirajah et al., 2007, 2009). Gil et al. (2011) have developed a topology-based method for process connectivity based on the physical description of systems. Several causal modeling techniques exist for constructing fault trees (FT) from process knowledge, which are named Model-Based Dependability Analysis (MBDA) techniques, and they depend on discovering dependency relationships from an existing system model (Aizpurua & Muxika, 2013; Sharvia et al., 2016; Kabir, 2017b). Another approach was developed by Leitner-Fischer and Leue (2013) to generate an FT expressing causality. The approach extracts counterexamples from the system's model, where some test conditions are utilized to discover events combinations that lead to different system states. Although knowledge-based methods have experienced some success in various applications, they are deemed ineffective when applied to large-scale systems and complex processes. This is more specifically due to the dependence of these methods on the existence of precise and representative models, which are extremely difficult to obtain in these large-scale systems and would require extensive effort by the experts and a very deep understanding of the process (Ge et al., 2017).

Therefore, it is appealing to learn causality models using data-driven techniques from historical data. Different techniques have been introduced to exploit historical data in the form of time-series observations to discover causality relations between the variables. Among the well-known data-driven techniques is the Granger causality (Granger, 1969). The Granger causality is based on the vector autoregressive model (VAR), which is a class of linear regressive models. A statistical significance is determined based on a G-causality index, which imposes a causality relation between two time-series if the cause can reduce the prediction error of the effect. One of the main applications of Granger causality was to establish root causes for plant-wide oscillations, which can propagate easily through the system due to the high level of interactions between components (Yuan & Qin, 2014). Several variants of the Granger causality have been developed. Landman et al. developed an approach that utilizes the Granger causality in combination with topology-based methods (Landman et al., 2014), where a causality matrix is initially constructed from the Granger causality, and afterward, each element in that matrix is validated for directness analysis using a connectivity matrix, which is captured from piping and instrumentation diagrams (P&IDs). Zhong et al., (2020) exploit data-driven Granger causality followed by further diagnosis using a topology-based model, which incorporates domain knowledge. The approach was tested to investigate solid oxide fuel cell system oscillations. Another method was proposed by Liu et al., (2020), which builds a simplified Granger causality map by determining its maximum spanning tree to overcome the complications of traditional causal maps such as process loops. Other variants were developed combining Granger causality with kernel entropy component analysis (KECA) for monitoring batch process (Fei et al. 2019), principal component analysis (PCA) for feature space reduction and early detection (Yuan & Qin, 2014), and with singular value decomposition (SVD) to amplify the contributing variables (Pyun et al., 2020). A topological causality approach has been introduced by Zhai and Yang, (2020), which builds a cross-map to infer causality and is validated using the Lorenz system and transfer entropy through causality index.

Various approaches were developed to exploit data and generate graphical causal models. One of the most common is causal Bayesian networks (CBNs), which have been applied widely (Neapolitan, 2004; Pearl, 2009). Li and Shi (2007) introduced an algorithm to generate casual Bayesian networks in rolling processes based on the integration of

domain knowledge for parameter learning, and a data-driven approach based on the Peter-Clark (PC) algorithm (Spirtes et al. 2000). The PC algorithm initially constructs a fully connected, undirected causal graph from the historical data as a first step and then determines the adjacencies between the variables and the directions. Peng et al. (2014) have proposed a novel approach based on multilogic probabilistic signed directed graphs (MPSDG) in the TEP, which consists of two modeling stages; an offline and online stage. In the former, the MPSDG is designed using historical frequency probabilities and logic gates describing the causality between variables. In the latter, the initial fault causes are discovered using a backtracking algorithm, and then the Bayesian inference is used to calculate the conditional probabilities of the causes. Moreover, Bauer and Thornhill (2008) have used a cross-correlation (CC) function to detect peaks and to estimate time delays between the system's variables in a large plant at the Eastman Chemical Company. The approach exploits historical data and the time delays to create a data-driven causality model verified by a statistical inference procedure to validate the peaks discovered from the CC function. Zhang and Geng (2015) have proposed an approach named dynamic uncertain causality graphs (DUCG), which deals with novel faults and represents uncertain causalities, by assessing an efficient probabilistic hypothesis. The approach is based on the decomposition of dynamic evidence into discrete time units and tests a global probabilistic hypothesis that is a combination of all-time units. Li and Yue, (2020) have modified DUCG to overcome difficulties in inferring the causality between events based on crisp numbers. The authors propose an intuitionistic fuzzy set based on dynamic uncertain causality graph (IFDUCG), which uses intuitionistic fuzzy sets to handle the uncertainty of events by integrating experienced knowledge into the traditional model. Jiuyong Li et al., (2016) proposed an approach based on causal decision trees (CDT). The approach exploits the advantages of decision trees to discover causality based on the "divide and conquer" strategy; however, the emphasis was on the distinction between the decision trees' correlation and causation. Causality is inferred in the algorithm based on the Mantel–Haenszel partial association test (Mantel & Haenszel, 1959). The output of the algorithm is in the form of causal decision trees, where non-leaf nodes are contributors to the outcome variable. It has also been concluded from the literature that fault trees are exploited significantly for extracting causality information (Kabir, 2017a; Malika et al., 2017; Waghen & Ouali, 2019; Zhou & Zhang, 2017). The hierarchy of events in a fault tree is used for interpretation of the dependency relations between basic and intermediate events up to the top-level event (fault).

An interesting approach has been developed by Bozorgi and et al., (2020), which gives recommendations for applying specific solutions based on uplift trees, which is a causal machine learning technique. The approach proposed in that work was applied to a financial business process. It was tested on loan applications problems, where the event logs are analyzed to increase the probability of the desired outcome. In that work, the authors assume an event log already exists, as well as candidate solutions to form action rules that solve a specific problem.

Nauta et al. (2018) have proposed the approach LIFT (learning fault trees from observational data) to generate data-driven static fault trees. Based on the Mantel–Haenszel statistical test, a system top event is analyzed to discover the sequence of contributing basic events by checking different combinations of AND/OR gates. The algorithm discovers causality between cause and effect and emphasizes data stratification to exclude the effect of other variables. Ragab et al., (2018) proposed a causality analysis methodology that combines human expertise, in the form of fault tree analysis (FTA), with additional knowledge extracted by a machine learning method. The proposed methodology was demonstrated using fault trees constructed for a reboiler system in a thermomechanical pulp mill.

Research has been conducted to use the state event graphs as powerful dynamic causality analysis tools, particularly by converting fault trees into state event graphs such as PNs, since these networks have more powerful mathematical modeling capacity than FTA, thus supporting more accurate quantitative analysis (Zhang et al. 2009; Steiner et al, 2012; Kabir et al., 2015). PNs are powerful visualization tools that can model the sequential and concurrency behaviors and allow for process simulation (David & Alla, 2010; Murata, 1989). These networks are convenient and efficient for building causal models for DES to analyze normal and faulty events temporal dependencies (Zhang et al., 2009). Therefore, we are exploiting PNs as the modeling framework in our proposed methodology. In what follows, the basic concepts and characteristics of PNs are presented.

State-event graphs and process mining: preliminaries

This section introduces the preliminaries of the state-event graphs and the process mining techniques. It presents the Petri nets models that illustrate these graphs and their building techniques. The objective is to give an overview to the readers who are not familiar with these concepts and help them understand how they are used in the proposed approach.

State event graphs: the petri net modeling approach

PN is a mathematical and graphical tool used extensively for the modeling and control of DES (David & Alla, 2010). These networks were first introduced by Carl Adam to model chemical processes (Petri, 1966). PNs serve as a promising tool for describing and analyzing event-driven processes that are characterized as non-deterministic, stochastic, concurrent, and/or time-dependent (Murata, 1989). Several extensions to PNs have been developed including timed PNs, hybrid continuous PNs, colored PNs, stochastic PNs, fuzzy PNs, etc. (David & Alla, 2010). Similar to flow charts, PNs can be utilized as a promising visual tool to simulate the dynamical evolution of interacting events in the system. Furthermore, PNs allow developing state and algebraic equations to serve as a mathematical analysis tool to model and analyze the system's behavior. In what follows, we present the PN background material necessary to follow the ideas presented in the paper. It can be read on a need-to-know basis. For further explanation of PNs concepts, elements, definitions, and applications, the interested reader is referred to (Murata, 1989; David & Alla, 2010; Mansour et al., 2013).

Graphically, PN is a directed bipartite diagram, which consists of nodes describing system states represented by circles called *Place*s, and rectangle-shaped bars called *transition*s. The *transition*s represent the events that occur in a system or a specific process, e.g. turning a machine on and off, or changing the set points of some controllers, which as a result can change the state of the system. The elements of the PNs are shown in Fig. 1. The nodes inside the PNs are connected by arcs, which can be weighted to describe the evolution of the system between the different states triggered by events that would occur as a result of specific conditions. The number of *Place*s and *transition*s inside a PN are finite and non-zero. The state of the system is determined by the number and distribution of tokens (dots) inside the *Place*s of the PN model, which is called marking. A marking is represented mathematically as a vector, where each entry indicates how many tokens are located in each *Place*. Changing the marking or the distribution of tokens will reflect a change in the state of the system. *Transition* nodes, which control the token dynamics, are enabled or ready to fire if and only if the number of tokens at each of its input *Place*s is greater than or equal to the weight of the directed arc connecting the *Place* to the *transition*. The firing of one *transition* moves a number of tokens determined by the weight of the input and output arcs, from the *transition*'s input *Place*s to its output *Place*s. Unweighted arcs have weights equal to one. A PN model, if built accurately, can be very helpful in explaining observational data collected from an industrial process. The *transition*s in the petri net can represent patterns in the data resembling changes in measured and controllable variables, which can be regarded as events occurring in the process. Moreover, the firing of the *transition*s would result in a change in the marking of the PN, which represents the current state of the system. Since the state of the system in reality can be determined by the KPIs, the marking can represent different ranges for those KPIs in the data. Mathematically, a PN is represented as $Q = \left(P, T, p^{+}, p^{-}, M_0\right)$, where P and T donate the sets of *Place*s and *transition*s, respectively. The notions p^{+} and p^{-} represent the arc weights from *Place*s to *transition*s, and from *transition*s to *Place*s, respectively and M_0 represents the initial marking. The incidence matrix [nxm] is denoted by $S = p^{+} - p^{-}$, while the firing vector σ_k [mx1] refers to the enabled *transition*s at epoch k. Finally, the evolution of states is given by $M(k+1) = M(k) + S\sigma_k$.

There are important properties that characterize PNs such as concurrency, conflict, liveness, boundedness, reachability, reversibility, and deadlock. Concurrency represents the occurrence of multiple events simultaneously, which signifies one of the essential aspects of PNs as a modeling tool. Conflict is a property of PNs that represents the non-determinism of events, where the firing of an event disables another event from firing due to the limitation of available resources as an example. Liveness refers to events that are continuously enabled or that could occur at any time during the process. Boundedness refers to the maximum number of tokens that any *Place* can hold in the network. A PN is considered "safe" if all *Place*s are bounded by only holding one token. Reachability refers to the possibility of reaching a specific marking from any other marking through a finite sequence of fired *transition*s or events. Reversibility is a property where the initial marking is reachable from any other marking. Finally, deadlock is a condition that occurs when no *transition*s are enabled to fire after reaching a particular marking; particularly, deadlock avoidance is one of the main challenges of PNs supervision.

The analysis of DES modeled as a PN can be conducted by multiple methods. The most common is the coverability graph analysis. It consists of a tree-like structure, which has a start node that represents the initial marking. The growth of the tree branches represents other markings that are reached from a preceding marking, and the arcs connecting the nodes represent the specific *transition* that caused the marking change and state evolution. An important note is that in the case where the PN is bounded, the coverability tree can be transformed into the simpler reachability graph, where the difference lies in the infinite markings, which only the coverability tree can represent (Murata, 1989).

Given the properties and modeling capabilities of PNs, these networks are exploited in this paper to model the causality relations between interacting events in industrial processes. Information about the sequence of events and their dependency relationships are realized through the analysis of coverability graphs to understand how the occurrence of a specific event could trigger other events. However, the con-

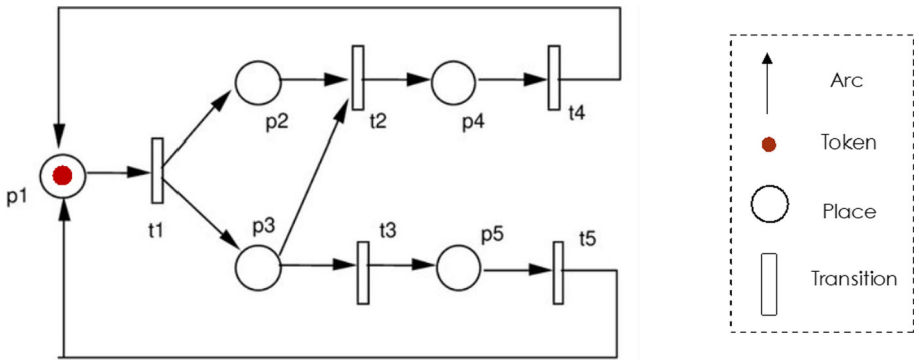

Fig.1 Elements of Petri Net

struction of PNs models is a tedious task that requires the involvement of human experts with deep domain knowledge about the system to be modeled. This becomes a significant challenge when modeling large-scale complex systems. However recently, the process mining approach (PM) (Van der Aalst, 2016) has emerged and has proven to be a promising tool in building PN models solely from data without the heavy dependence on domain knowledge and human effort. Therefore, we will exploit PM to build dynamic causal models in the form of PNs from the historical data that are collected from the monitored industrial systems.

Process mining

PM is an emerging approach that has gained a lot of interest. The availability of big data in most organizations' information systems represents an important asset that is not being fully exploited. PM supports model discovery from data, process enhancement as well as providing deep structural insights for business processes (BP), where a BP is a collection of sequences of events performed by equipment or individuals to provide a particular service or product for customers (Grisold et al. 2020; Weske, 2012). PM comprises a set of data-driven tools that discover, analyze and enhance process models to support decision-making, gain insights, apply verification of process models to locate errors in systems' procedures, and provide simulation techniques to conduct performance analysis and assistance in process redesign (Reinkemeyer, 2020; Van der Aalst, 2016). The essence of PM lies is in the exploitation of event data without the need for domain knowledge or expertise, as well as to uncover bottlenecks and non-compliance patterns that arise when event data and existing process models are compared and do not reflect the same behavior (Diba et al., 2020; Van der Aalst, 2016). The PM framework is mainly categorized into three tasks; process discovery, conformance checking, and process enhancement, which are depicted in Fig. 2. (Van der Aalst, 2016). Each of these tasks is dependent on the problem at hand. In the discovery task, process models are generated automatically from event logs using data-driven discovery techniques. The conformance task validates the generated models and measures their accuracy by comparing the behavior they allow with the event logs. The third task, edits and enhances the models upon the arrival of new event logs. In what follows, these tasks are briefly explained.

Process discovery

Process discovery constructs and discovers a model from event data logs by analyzing the dependency relationships between events that occur in particular sequences during executing different procedures in an organization. Examples of these dependency relations are; directly follows; never follow; parallelism; and loops. An event log, which is the input to process discovery techniques consists of a number of events, each assigned a *Case id*, and a timestamp, where a *Case* represents a specific time period where the system has been operating, such as an hour, day, or week depending on the problem at hand. An example of an event log for a machine operation is shown in Table 1. A *Case* here represents a day of operation. This event log is named E_1 and has five different events {*P1, P2, P3, P4, P5*} and three *Cases* corresponding to three days of operation.

The first PM discovery algorithm introduced was the alpha algorithm (Van Der Aalst et al., 2004). The alpha algorithm takes an event log and generates a workflow net (a class of Petri nets) that can probably replay the sequences found in the event logs correctly. Further improvements have been developed to the alpha miner algorithm by De Medeiros et al. (2003) such as the alpha + algorithm where pre- and pro-processing stages are added to deal with event loops. The heuristic miner (Weijters & Ribeiro, 2011), another process discovery algorithm, utilizes casual nets as the representation of the discovered models. The heuristic miner's framework is formulated by creating a dependency graph where the frequency of an event x followed by an event y is calculated and afterwards, the dependency graph is transformed into state-event graph after applying noise filtering based on specified thresholds. Another algorithm is the β-algorithm (Wen et al., 2009), which takes into consideration different event types

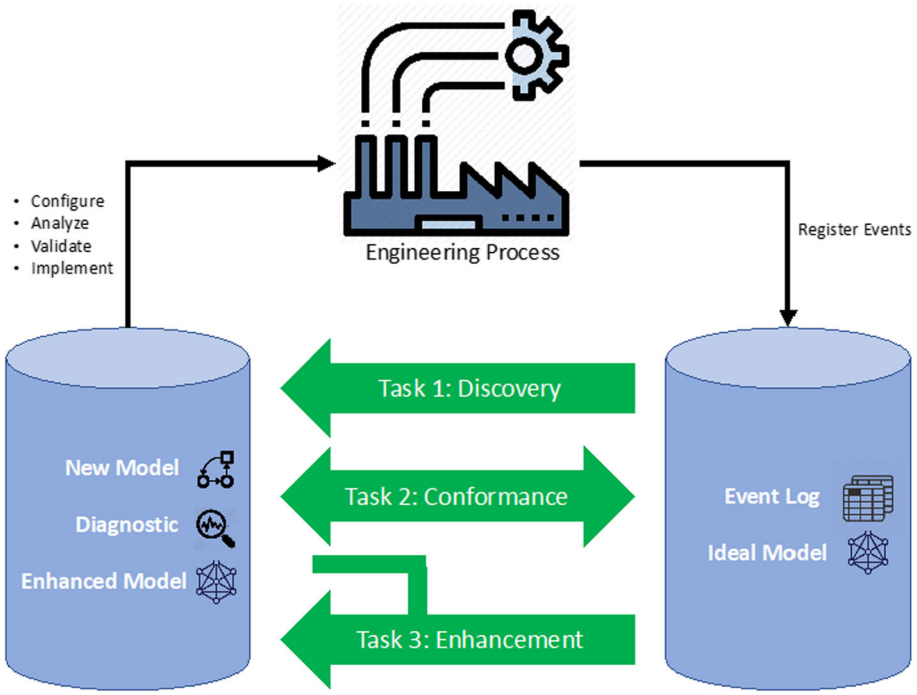

Fig. 2 The three tasks of Process Mining

Table 1 Event log E_1 of an industrial machine operation

Case Id	Timestamp	Event	Event ID
1	01/02/2019, 11:50	Machine Start	P1
	01/02/2019, 11:59	Set Point 1 Change	P2
	01/02/2019, 12:40	Set point 2 change	P4
	01/02/2019, 12:49	Machine Stop	P5
2	02/02/2019, 9:30	Machine Start	P1
	02/02/2019, 9:45	Set point 2 change	P4
	02/02/2019, 10:03	Set Point 1 Change	P2
	02/02/2019, 10:20	Machine Stop	P5
3	03/02/2019, 9:10	Machine Start	P1
	03/02/2019, 9:20	Set point 3 change	P3
	03/02/2019, 9:20	Machine Stop	P5

such as start and completion of tasks to build PN models with richer information. For more about PM techniques, the interested reader is referred to (Van der Aalst, 2016).

In this paper, we utilize one of the most exploited process mining discovery algorithms: the inductive miner (IM). This algorithm can deal with huge event data logs, interactive events, and noise filtering while preserving the correctness of the model and rediscoverability (Van der Aalst, 2016). The Inductive Miner generates what is called process trees, which can be easily converted to other notations such as PNs. The Inductive Miner is the one of the few process discovery techniques guaranteeing soundness (total execution of process phases, while reaching a defined end-state), high fitness (accuracy criterion of process models) in finite time (Leemans et al., 2013a). In what follows we explain how the IM algorithm works (Van der Aalst, 2016).

The inductive miner algorithm starts by creating what is called a directly follows graph, which represents the follows relations between the events in an event log. The algorithm comprises two steps, described in what follows.

Steps 1: {*Creation of Directly follows graph*}.

Let E be an event log, the directly follows graph of E is represented as $D(E) = (S_E, \mapsto_E, S_E^{start}, S_E^{end})$, where:

- S_E is the set of all events in E
- $\mapsto_E$ is the directly follows relations
- S_E^{start} is the set of start events
- S_E^{end} is the set of end events

The algorithm recursively divides the original event log into sub logs, and for each sub log a directly follows graph $D(E)$ is created using directly follows relations such as $x \mapsto_E y$, if x is followed directly by y anywhere in E. Also, $x \mapsto_E^+ y$, if there exists a path from x to y that is non-empty. To give an example, for the event log $E_1 = [\{P1, P2, P4, P5\}, \{P1, P4, P2, P5\}, \{P1, P3, P5\}]$ shown in Table 1, E_1 consists of three *cases* and five different events. Therefore, $S_{E1} = \{P1, P2, P3, P4, P5\}$, $S_{E1}^{start} = \{P1\}$, which is shown in Fig. 3 by the incoming arc and $S_{E1}^{end} = \{P5\}$, which is shown in the figure by the outgoing arc. And the directly follows relations are:

$P1 \mapsto_E P2$, $P1 \mapsto_E P4$, $P1 \mapsto_E P3$, $P2 \mapsto_E P4$, $P4 \mapsto_E P2$, $P2 \mapsto_E P5$, $P4 \mapsto_E P5$, $P3 \mapsto_E P5$, $P1 \mapsto_E^+ P5$. Based on these four elements, the directly

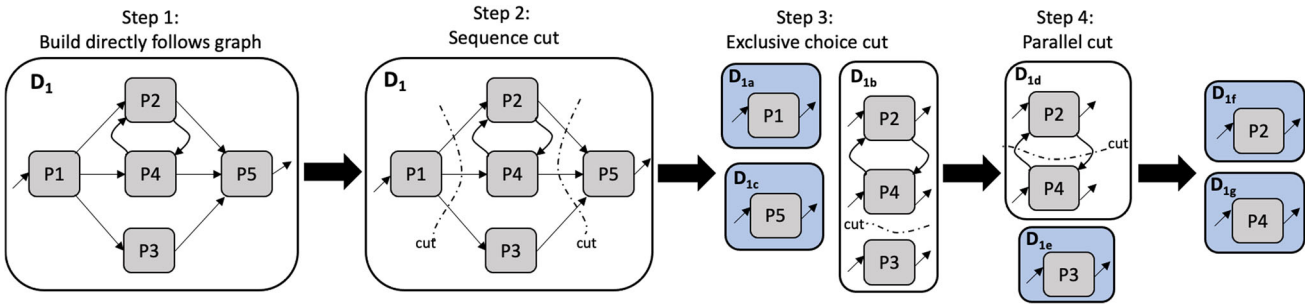

Fig. 3 Creating a directly follows graph for event log E_1 and finding cuts that represent the dependency relationships using the IM algorithm (Adopted from (Van der Aalst, 2016))

follows graph is constructed as shown in the first box on the left side in Fig. 3.

The inductive miner algorithm will divide E_1 iteratively until reaching sub logs that contain only one event, which are called base event sub logs. In order to execute these divisions, the algorithm applies cuts in the directly follows graph of the log or sub logs. In this work, we consider three kinds of cuts namely, *sequence cuts, exclusive choice cuts,* and *parallel cuts*, represented in process trees notation as $\rightarrow$, $\times$, and$\wedge$, respectively.

Step 2: {*Determination of Dependency Relation Cuts*}.

As mentioned, the directly follows graph of an event log E is represented as $D(E) = (S_E, \mapsto_E, S_E^{start}, S_E^{end})$, where S_E is the set of events in E. A cut will divide S_E into pairwise disjoint sets $S_1, S_2, \ldots, S_n$, where $S_i \cap S_j = \varnothing$ for $i \neq j$

- *Sequence cut* of $D(E)$ takes the form $(\rightarrow, S_1, S_2, \ldots, S_n)$, where $\forall_{i,\, j\in\{1,\ldots,n\}} \forall_{x\in S_i} \forall_{y\in S_j}\, i<j \implies x \mapsto_E^+ y$ and the opposite is false.
- *Exclusive choice cut* of $D(E)$ is represented as$(\times, S_1, S_2, \ldots, S_n)$, where $\forall_{i,\, j\in\{1,\ldots,n\}} \forall_{x\in S_i} \forall_{y\in S_j}\, i\neq j \implies x \mapsto_E y$ is false.
- *Parallel cut* of $D(E)$ takes the form $(\wedge, S_1, S_2, \ldots, S_n)$, where $\forall_{i\in\{1,\ldots,n\}} S_i \cap S_E^{start} \neq \varnothing\ AND\ S_i \cap S_E^{end} \neq \varnothing\ AND\ \forall_{i,\, j\in\{1,\ldots,n\}} \forall_{x\in S_i} \forall_{y\in S_j}\, i \neq j \implies x \mapsto_E y$

Returning to the example, after creating the directly follows graph D_1, the algorithm will cut it into three smaller graphs namely, D_{1a}, D_{1b}, *and* D_{1c} as shown in the figure, based on the *sequence cut*$(\rightarrow, \{P1\}, \{P2, P4, P3\}, \{P5\})$. This divides the set of events into three subsets where arcs connecting those subsets only go from left to right. Therefore, due to this *sequence cut*, three sub logs are created from the original event log E_1, namely $E_{1a} = [\{P1\}]$, $E_{1b} = [\{P2, P4\}, \{P4, P2\}, \{P3\}]$, $E_{1c} = [\{P5\}]$ and correspondingly, we have.

$\boldsymbol{IM}(E_1) = \rightarrow (\boldsymbol{IM}(E_{1a}), \boldsymbol{IM}(E_{1b}), \boldsymbol{IM}(E_{1c}))$, where $\boldsymbol{IM}(E_{1a}) = P1$, $\boldsymbol{IM}(E_{1c}) = P5$, and therefore.

$\boldsymbol{IM}(E_1) = \rightarrow (P1, \boldsymbol{IM}(E_{1b}), P5)$.

It is worth mentioning that an event cannot appear in more than one sub log. Now since E_{1a} and E_{1c} contain only one event, therefore, no cutting is further needed. Afterwards, D_{1b} is cut into two smaller directly follows graphs namely D_{1d}, D_{1e} as shown in the figure, based on an *exclusive choice cut* $(\times, \{P2, P4\}, \{P3\})$. This divides the set of events into two subsets where there are no arcs connecting the two subsets together. Due to this *exclusive choice cut,* two sub logs are created from E_{1b} namely$E_{1d} = [\{P2, P4\}, \{P4, P2\}]$, $E_{1e} = [\{P3\}]$, and we have.

$\boldsymbol{IM}(E_{1b}) = \times(\boldsymbol{IM}(E_{1d}), \boldsymbol{IM}(E_{1e}))$, where $\boldsymbol{IM}(E_{1e}) = P3$, and therefore $\boldsymbol{IM}(E_{1b}) = \times(\boldsymbol{IM}(E_{1d}), P3)$.

The sub log E_{1e} is a base case sub log, however, the sub log E_{1d} still needs more cuts. Therefore, D_{1d} is cut into two sub logs, namely D_{1f}, D_{1g} as shown in the figure, by the *parallel cut* $(\wedge, \{P2\}, \{P4\})$, which divides the events of into two subsets where every event in each subset is connected to all events in the other subsets. Due to this *parallel cut*, two sub logs are created from E_{1d}, namely$E_{1f} = [\{P2\}]$, $E_{1g} = [\{P4\}]$, and we now have.

$\boldsymbol{IM}(E_{1d}) = \wedge(\boldsymbol{IM}(E_{1f}), \boldsymbol{IM}(E_{1g}))$, where $\boldsymbol{IM}(E_{1f}) = P2$ and $\boldsymbol{IM}(E_{1g}) = P4$.

Finally, by substituting, we get $\boldsymbol{IM}(E_{1d}) = \wedge(P2, P4)$, then $\boldsymbol{IM}(E_{1b}) = \times(\wedge(P2, P4), P3)$, and then.

$\boldsymbol{IM}(E_1) = \rightarrow (P1, \times(\wedge(P2, P4), P3), P5)$. The equivalent process tree is depicted in Fig. 4. The conversion from process trees to Petri nets is based on the rules in Fig. 5.

Other variants exist for the framework to deal with noise and infrequent behavior such as the Inductive Miner Infrequent Behavior (IMf) (Leemans et al., 2013b). The authors show that the IMf is able to discover sound, deadlock-free PN models that can accurately display the observed behavior in the event log, while filtering infrequent behavior and noise as it analyzes the frequency of events, while maintaining high fitness scores, in addition to preserving an appropriate balance between model simplicity, precision, and generalization.

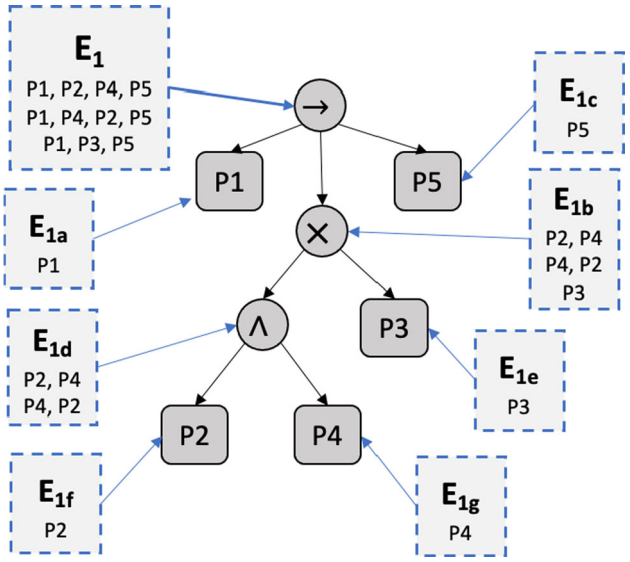

Fig. 4 Process tree obtained after applying the inductive miner algorithm on event log E_1

Conformance checking

Conformance checking is the analysis of an existing or discovered model in terms of accuracy, simplicity and generalization. Particularly, by comparing the event log with the model, the goal is to mark the similarities and key differences between both to analyze how accurate the model is in representing the observed behavior in the log. The main criterion for conformance checking is the fitness value, which is a model accuracy measure representing the percentage of *Cases* in the event log that could be modeled correctly by the discovered process model. Another conformance criterion is precision, which examines if the model allows for too much behavior that is not observed in the log. A number of techniques have been designed to calculate the fitness value. Token replay (Van der Aalst, 2016) is one of the methods for evaluating the accuracy of the PN models in representing the event log. For each *Case* in the log, the model is run to simulate those *Cases*, by placing a token in the start *Place*. A simulation for each *Case* begins and as the tokens progress, the behavior of the model is compared to the *Case* in the log. The comparison is conducted by mapping the event log behavior on top of the model for each *Case* and then calculating four variables related to the number of tokens; the number of **produced** tokens by the *transitions*; the number of **consumed** tokens by the *transitions*; the number of **missing** tokens due to an occurrence of an event in the event log, while there are not enough tokens for the corresponding *transition* to fire in the model; the number of **remaining** tokens in the

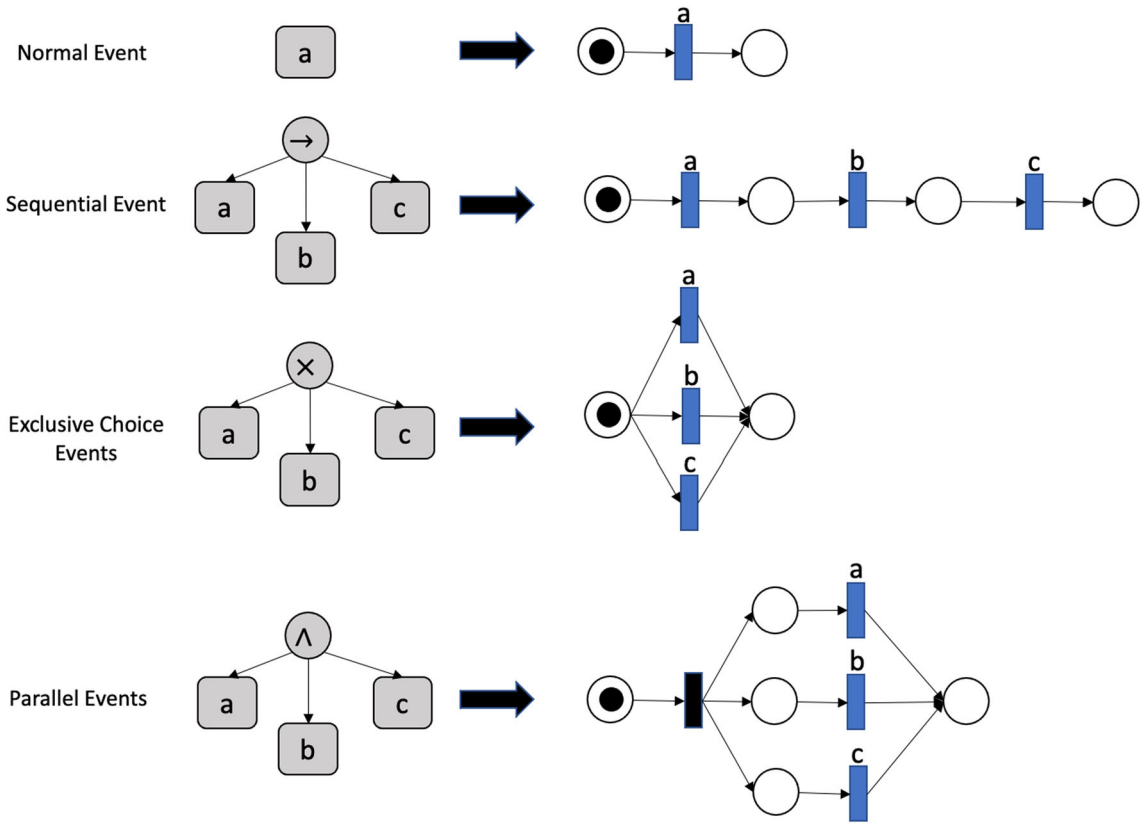

Fig. 5 Rules for converting process trees to Petri nets

PN if there are tokens in any *Place* in the PNs other than the end *Place* after the all events in a specific *Case* are completely executed. Normally, a perfect *Case* modeling (100% *Case* fitness) by the model would mean that; produced = consumed; missing = remaining = 0. In the situation of different values for the four mentioned variables; they are substituted into a formula to calculate the fitness of that *Case*. This step is repeated for all *Cases* in the event log and then averaged to calculate the total fitness of the model. Another conformance checking approach is the alignment method (Van der Aalst et al., 2012), where each *Case* in the log is simulated by the model and a corresponding alignment table is created. This approach can provide a detailed diagnostic procedure for each given *Case* and can be applied to any modeling process notation.

Process enhancement

The third task of Process Mining is process enhancement, which is the modification of an existing discovered model *X* that doesn't align with a certain event log to a new model *Y* that conforms with the event log while still capturing most of the structure of model *X*. This can be significantly beneficial for a smooth and automatic model update when a new event log is collected, resembling changes that occurred in the process. An approach was developed in (Fahland & Van Der Aalst, 2015), which applies the repair actions based on the fitness criteria calculated by conformance checking. By applying a threshold on the model's fitness value, one of three repair actions can be executed; (1) No repair due to perfect alignment; (2) Total event log-based rediscovery due to very low fitness and extreme divergence; or (3) Partial rediscovery only to the non-fitting parts to incorporate the new behavior found the updated event data, while preserving most of the original model's structure. Werner-Stark et al. (2011) applied the method for exploiting the process enhancement and conformance checking techniques to update a model of a discrete event parking system. A normal behavior model was initially built representing the system's reference model, and after acquiring new event data log which contained faults that occurred in the system, the logs were replayed on the reference model to detect and pinpoint the divergencies, which were used to modify the reference model to include the new abnormal events in the model.

Proposed methodology

This paper proposes a data-driven dynamic causality analysis approach based on AI and structure learning from low-level observational data. The methodology combines interpretable machine learning (decision trees in this paper), process mining, specifically the process discovery task for model generation and the conformance checking task for model validation and fitness calculation, and human expertise for defining the process' operating regions and verification to construct accurate and representative dynamic causal graphs for complex industrial processes. It can serve as a building block for different tools in process industries such as systems' trajectory visualization and supervisory control. The methodology can be applied in industrial plants to avoid abnormal events and to maintain the desired process behavior that achieves energy-efficient operations. The schematic diagram of the methodology is depicted in Fig. 6.

The detailed steps of the proposed methodology are explained in two main phases, namely, (1) operating region definition based on experienced KPI thresholding and pattern extraction using IML, (2) event data log preparation and causal model generation using process mining techniques.

Phase 1. Operating regions definition based on KPIs thresholding and pattern extraction using IML

In this phase, low-level observational data is collected by a data management system from an industrial plant, which comprises three types of time-sampled information: measured variables, controllable variables (variables that operators have direct control on), and KPIs (measured). The KPIs demonstrate how effectively an industrial facility is achieving its objectives. The KPIs can represent ratios and quantities of key production elements, the efficiency of energy consumption, as well as measurements of wastes and non-eco-friendly emissions.

Following the acquisition of data, the first step is to define the operating regions of the process based on the KPIs. Specifically, the aim of this step is to transform the KPIs in the data from continuous values to discrete classes representing how efficient the process is operating in each timestep. The process experts, who are deeply familiar with the plant, would construct a classification criterion particularly based on the KPIs to partition the state space of the system's operation into several operating regions representing normal and abnormal levels of operations. The definition of operating regions is mainly dependant on human experts and domain knowledge repositories, and in some situations, experts can rely on different data analytics. The process expert defines specific thresholds to be applied to the KPIs in order to define different operating regions. Each defined threshold is a cut point that would determine if a KPI value is normal, slightly abnormal, or severely abnormal, etc. As a result, the KPI values for each observation in the data would satisfy one of the defined operational classes. Examples of these classes are shown in the "KPI Class" column in the table of Fig. 9. The class labels of observations are assigned automatically by projecting them on a set of deductive IF–THEN rules using the thresholds defined on KPIs based on the human expert.

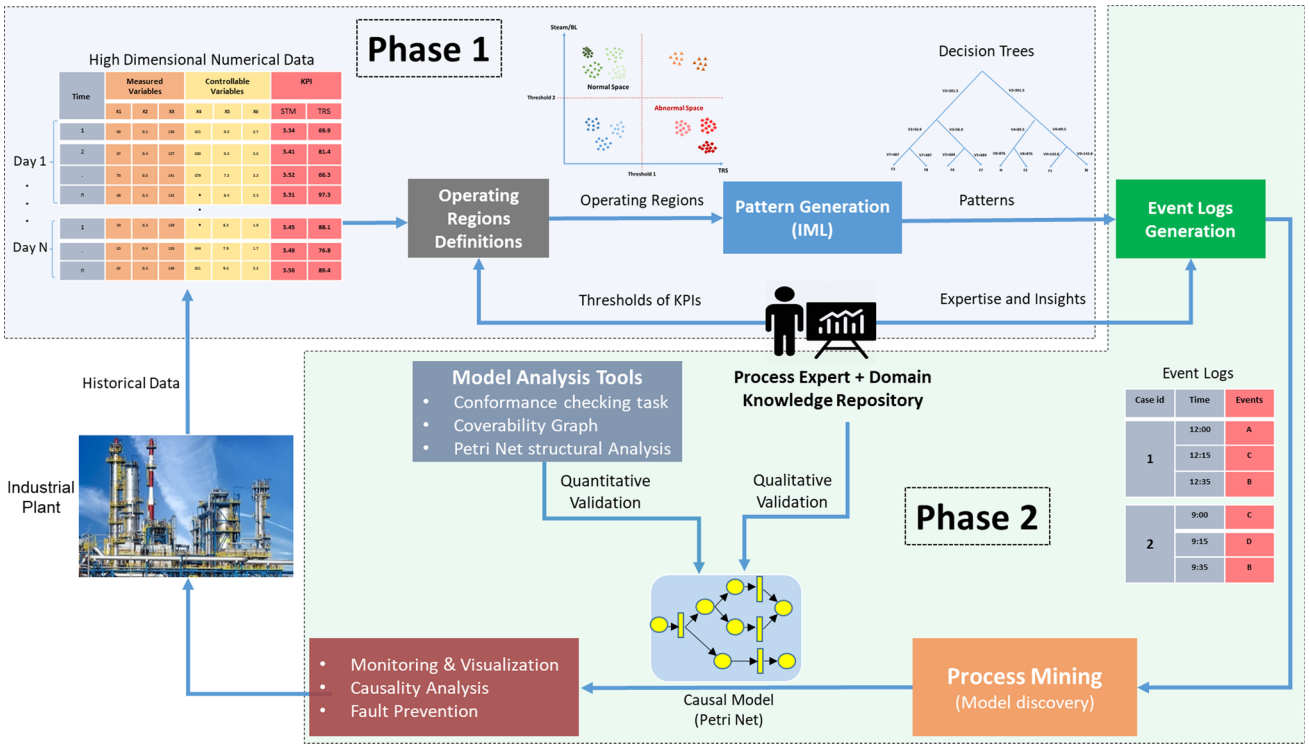

Fig. 6 Schematic of the proposed methodology

Thus, we can exploit those labels as target variables to apply supervised learning techniques, particularly IML, in order to gain some insights on the reasons for KPI class change, which would be the following step in this phase. It is important to note that the human expert is mainly involved to define and confirm those thresholds, as he/she is deeply familiar with the process and its KPIs' related targets or objectives. In fact, this task is important to extract accurate patterns from the data and this thresholding procedure is done with minimal effort from the expert who will be only involved in the validation of the resulting causal models.

In the second step, we utilize the DT, an inductive rule-based classifier, as the IML method. It is commonly used as a supervised learning method for extracting rules from data. DTs represent a flowchart structure, such that each node in the tree is an evaluation on a specific variable or feature and the result of that evaluation is represented by the output branches. These branches would link the tree nodes until reaching the leaf nodes. Each leaf node corresponds to the classification or decision label. Thus, the classification rules are represented by the paths from the top node to the leaf nodes. The quality of each split in the tree is evaluated by different criterions. Two of the most common are the Gini Index and Information Gain. The Gini Index will measure the frequency at which any observation in the dataset will be misclassified with random classification. While Information Gain measures the reduction in entropy after splitting a dataset is split based on variable.

In this approach, the input variables to the DT method are the measured and controllable variables in the data, while the target classes are the operating regions defined from step one. The main goal of using DTs is to exploit its interpretability to extract meaningful patterns that would explain the relation between each observation and its assigned class/label. Thus, acquiring information about the specific combinations of variables with different values that would lead to different KPI classes (e.g. normal KPI, slightly abnormal KPI, severely abnormal KPI, etc.). Each extracted pattern from the classification DT model is a combination of cut points on the measured variables, controllable variables, or both, with different value ranges, as illustrated in Fig. 7. The objective of the pattern extraction step is two folds (1) to identify the important events that occur in the process and affect the overall performance and their effect on the KPI, (2) to transform the original data, which comprises KPIs and continuous variables (measured and controllable) into discrete event logs, primarily because event logs are the appropriate input for PM discovery techniques.

To ensure highly accurate DT classification and the extraction of strong representative patterns, we apply an IML model enhancement approach based on the work done in (Dhurandhar et al., 2018) to integrate the accuracy of powerful models (such as ensemble of weak learners) and the inter-

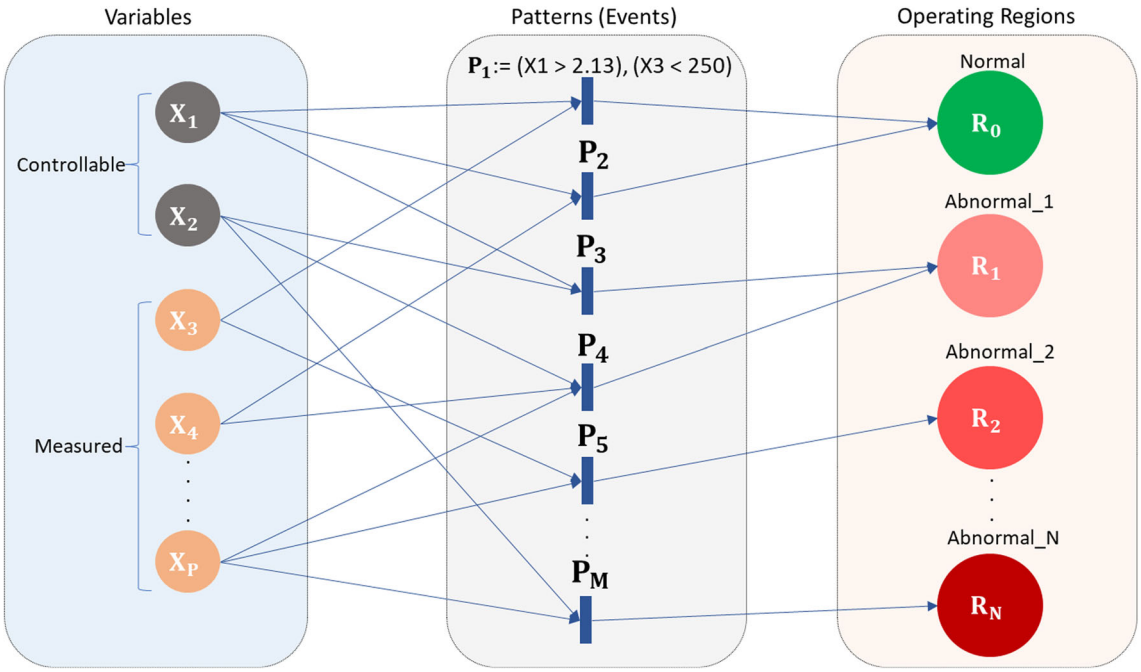

Fig. 7 Definition of extracted pattern

pretability of simple models. The main concept is to train a teacher model (powerful model), which has low interpretability; however, can achieve a high test accuracy on a specific dataset (Lee et al, 2021) and use this trained model to assign weights to each observation in the dataset, so that observations that are easy to classify would be assigned higher weights, while the ones that are difficult to classify are given lower weights. Specifically, for each observation in the dataset, the weight is calculated as the probability of assigning the correct (true) label by the trained teacher model. These probabilities can be easily acquired from the prediction probabilities of the trained teacher model for each and every observation. Afterwards, we train the student model (simple model having high interpretability but low test accuracy on the original dataset), using the weighted dataset. The observation weighting forces the simple classification model to focus on predicting the correct class of observations with high weights and now the model can learn while ignoring difficult observations (they are considered outliers) that will probably cause the simple model to have low accuracy. The student model can eventually obtain higher generalized accuracies while still maintaining its interpretability. The paper (Dhurandhar et al., 2018) uses a deep neural network as the teacher model in order to enhance the performance of a CART based on DTs. It starts by adding logistic classifiers to the intermediate layers of an accurate pre-trained deep neural network. Each classifier provides the prediction from the layer it is attached to. A confidence profile curve is generated which contains the confidence scores based on the logistic classifiers output and the true labels of the input. The observations in the original dataset are weighted after by using the area under curve (AUC) as a function of the confidence scores. In this paper, we adapt this method to use Random Forests as the teacher model instead of a deep neural network, and we use the prediction probabilities of the Random Forest to assign weights to each observation.

The technique is powerful since there is no trade-off between the interpretability and the accuracy of the simple model, i.e., the high interpretability of the simple model will not be affected and only its accuracy could be improved (Dhurandhar et al., 2018). The interpretability of the DT model (student) lies in its ability to extract patterns that are represented by the paths from the top node to the leaf nodes where each leaf node is assigned one of the target classes. After applying the IML enhancement method, the observations that are misclassified by the teacher model are assigned lower weights hence can be ignored by the student model. Consequently, after the student is trained based on the new weighted dataset, the output would still be an interpretable tree-like structure comprising more accurate patterns. Furthermore, there is no fixed boundary for accuracy improvement, since the efficiency of the technique depends on the performance of the teacher model and the complexity of the data in terms of data cleanliness, noise, and outliers. An illustration of the DTs model enhancement step is shown in Fig. 8.

By mapping the extracted patterns to the original dataset, sets of observations would be assigned to each pattern if they satisfy the pattern's rules i.e. satisfy the variables'

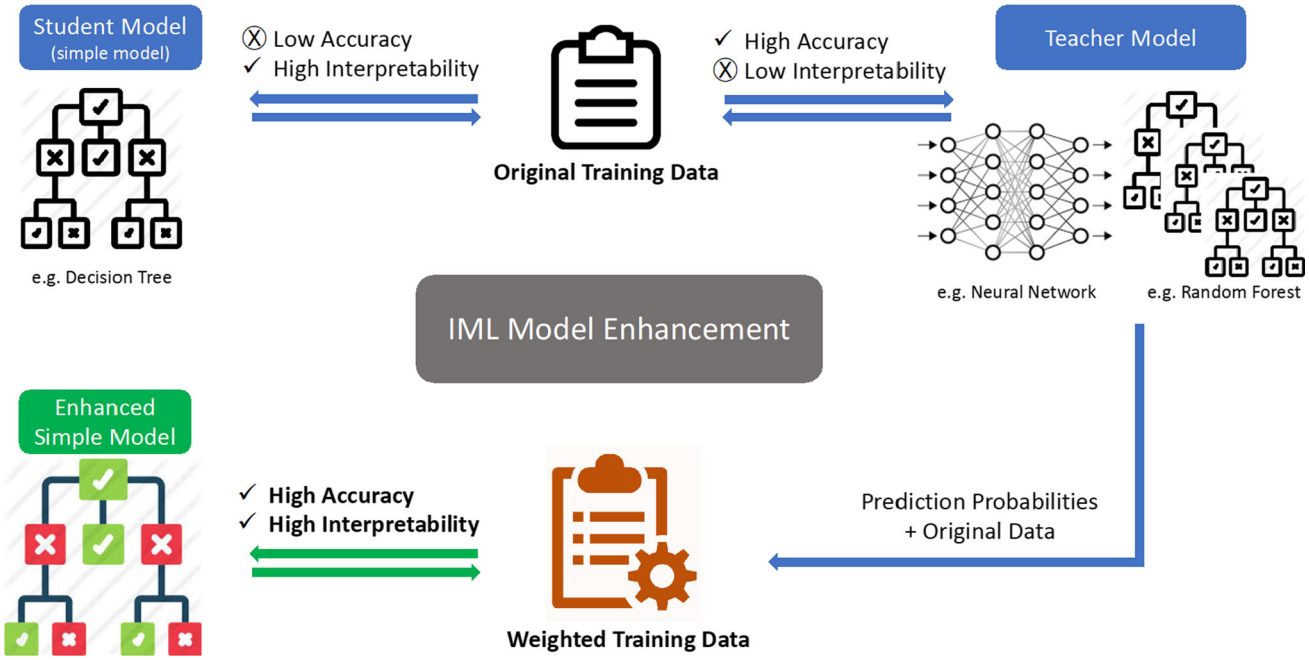

Fig. 8 Training of enhanced IML models

ranges defined by that pattern. At the same time, each pattern explains a specific operational class based on the DTs leaf nodes (one of the operating classes defined in *step 1*), which is the target label used in the training of the DTs classifier. The mapping is performed in the next phase.

Phase 2: Event data log preparation and causal model generation based on Process Mining

As the first step in this phase, the patterns that were extracted in the form of tree-like structure in Phase 1 using classification DTs are transformed into IF–THEN rules consisting of a set of cut points on the values of the measured or the controllable variables, or both. These rules are mapped to the dataset to assign each extracted pattern to a set of observations that satisfy this pattern. Figure 9. shows this step with an illustrative example of a small dataset consisting of four input variables (X1, X2, X3 and X4) and one KPI. There is one threshold on the KPI set by the expert to define two operating regions (Normal, or Abnormal) which are shown in the "KPI Class" column. The data listed in the table is used to train a classification DT model with the four variables as inputs and the KPI class as the target variable. After training the DT model, a tree structure is generated as shown on the right side of Fig. 9. Each path in the tree (from the top node to a leaf node) represents a pattern, which is translated to a set of nested IF–THEN rules. The pattern P3 is shown as an example. Afterward, each observation in data is run through the sets of IF–THEN rules to determine which pattern covers the observation then the covering patterns are added to the "Event/Pattern" column. Each pattern is an event that occurs during the process. We illustrate the relation between the event and the pattern by this example: by manipulating some of the controllable variables in a real process, the system can transition to a new state and, therefore, this would be considered an event. Also, satisfying the pattern's values regarding its controllable variables can be similarly considered an event that occurs in the real process, which would also transition the system to a new state.

Therefore, the original continuous dataset is now transformed into an event log, comprising discrete events (which are the extracted patterns), timestamps, and *Case ids* which would mark the start to the end of a process operation period, and all events that happened in between, such as all events that occurred during in one day of operation. In this paper, all events that happened in one day are assigned the same *Case id*. Therefore, the output of this step is an event log containing sequentially ordered events (the generated patterns mapped on the observations), which occurred during the process describing the system's evolution through its operational states. The event log, which is built based on the example in Fig. 9 is shown on the left side of Fig. 10.

In the second step, the generated event log is fed as input to the PM platform for preprocessing and causal model generation. Information about the log such as min, max, and the mean number of events per *Case*, the frequency of occurrence of each event, and frequency of *Cases* are inspected. Based on the cleanliness and complexity of the original data, some filtering might be needed after the analysis, where some events could be filtered to remove low-frequency events, which are

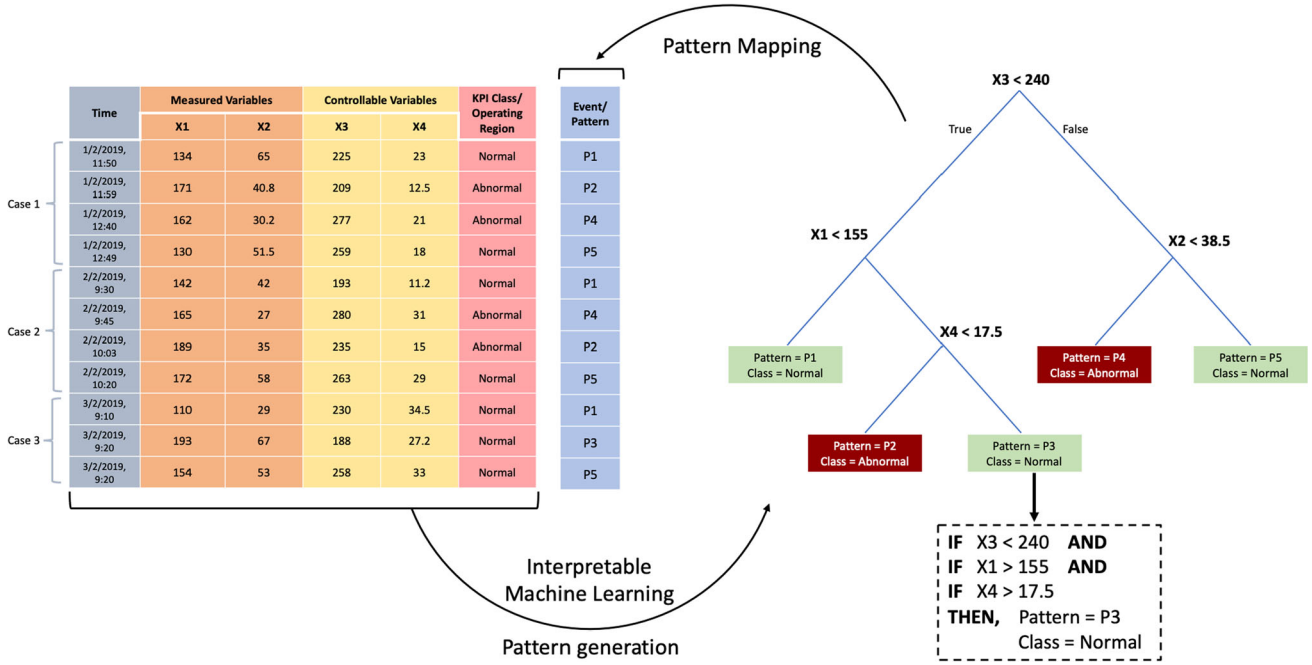

Fig. 9 Event log generation from low-level continuous observations

Event log

Case Id	Timestamp	Pattern/Event
1	1/2/2019, 11:50	P1
	1/2/2019, 11:59	P2
	1/2/2019, 12:40	P4
	1/2/2019, 12:49	P5
2	2/2/2019, 9:30	P1
	2/2/2019, 9:45	P4
	2/2/2019, 10:03	P2
	2/2/2019, 10:20	P5
3	3/2/2019, 9:10	P1
	3/2/2019, 9:20	P3
	3/2/2019, 9:20	P5

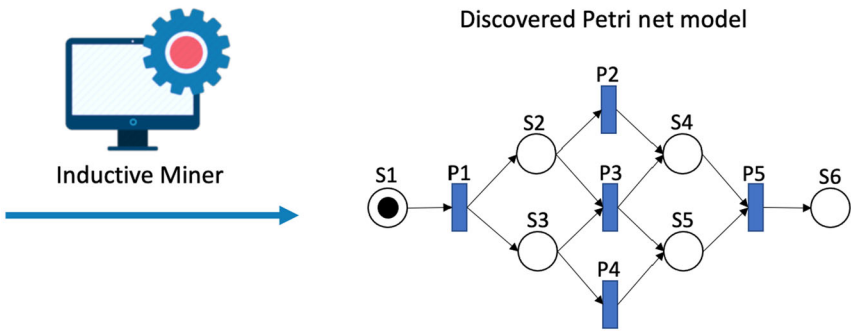

Fig. 10 Causal model discovered from the event log in Table 1

considered as outliers or sensor errors. Afterwards, the event logs are exploited to build an accurate and representative causal model for the process. For this task, we utilize the IMf as the PM discovery technique to process the event log data, analyze the dependency relationships between events, and generate state-event graphs models. The output would be a PN with states representing the operational classes and *transitions* representing the events in the log. The steps for building the Petri Net model from the event log using the IM algorithm are explained in Sect. 3.2.1. The output Petri net model for the event log listed in Table 1 is shown in the right side of Fig. 10.

In what follows, we explain how the three cases in the event log in Fig. 10 are properly modeled using the IM algo-

rithm. From the discovered PN model shown in the figure, it can be noticed that the event *P1* is a start event and *P5* is an end event, which indicates that the IM algorithm is able to properly represent the event log, since all *Cases* start with *P1* and end with *P5* in the event log. Furthermore, in the event log, *P1* can be followed by *P2* or *P3*, or *P4*, which is represented properly in the PN since the firing of the *transition P1* will remove a token from the *place S1* and add a token in places *S2* and *S3*, thus only transitions *P2*, *P3*, and *P4* are enabled to fire afterwards (see Sect. 3.1 for the enabling conditions of transition firing). If *P3* is fired then the tokens in places *S2* and *S3* will be removed and a token will be added to each of the places *S4* and *S5*, thus, *P2*, and *P4* are not enabled to fire while *P5* is the only enabled transition. This indicates that the PN model represents *Case 3* properly. In the situation where *P2* is fired instead of *P3*, then only the token in *S2* will be removed and a token will be added in *S4*, thus *P4* will be the only enabled transition. Afterwards, *P4* is fired and the token in *S3* will be removed, and a token is added in *S5* and therefore *P5* is enabled afterwards. This represents *Case 1* correctly. The same can actually happen if *P4* is fired instead of *P2*, which will only enable *P2* afterwards and then *P5*, which indicates the model represents *Case 2* properly.

The dynamics of the system and its trajectory through time can be tracked by observing the tokens' movements inside the discovered PN model. Causality relationships could be inferred by observing the event sequences and their temporal dependencies in the Petri net's coverability graph. The coverability graph is created during quantitative analysis, which shows all possible state changes and the *transition*s responsible in a sequential manner, due to the occurrence of an event or a group of events. By analyzing the graph, operators could gain insights into how certain events could eventually trigger other events causing state transitions. Following the process model discovery, we proceed to the second task of process mining, which is conformance checking. As mentioned previously, conformance checking measures the accuracy and precision of the models by comparing the behavior or event sequences that the model allows with the event log. More specifically, we are interested in measuring the fitness of the model, which is a measure of how many *Cases* in the event log could be modeled correctly by the discovered model. By ensuring a high fitness score, a representative and trustable model of the system can be possessed.

In a situation where new data is collected, the third task of process mining, which is process enhancement, is used to update the discovered PNs model by only editing to incorporate the novel event sequences observed in the new event log, while preserving most of the structure of the existing discovered model. This is significantly advantageous, as we are able to update the PNs model in an automated fashion whenever new data is collected, which would save time and effort rather than the traditional way of totally reconstructing the model from scratch. Finally, quantitative analysis is performed on the generated Petri Nets model to ensure the correctness of the model in terms of boundness, liveness, and being deadlock-free.

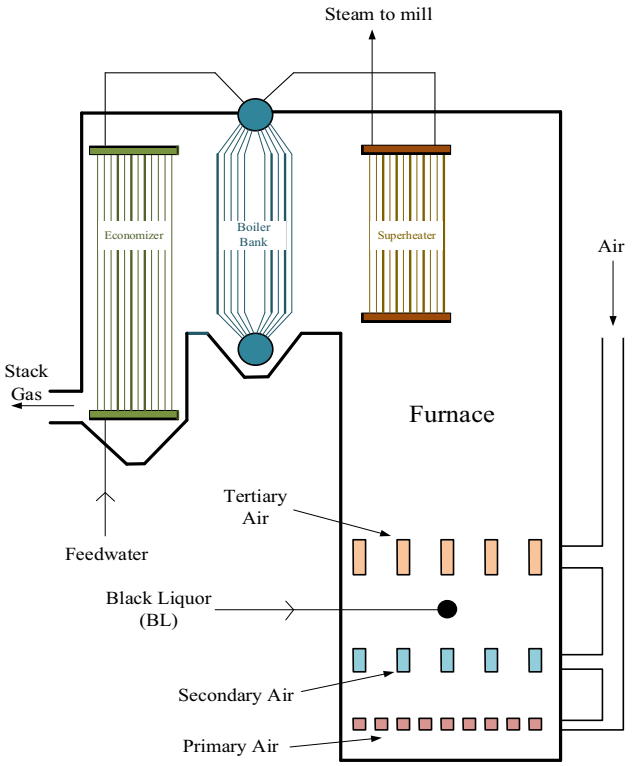

Fig. 11 A schematic for the BLRB, adopted from (Vakkilainen & others, 2005)

Case study: A black liquor recovery boiler in a Kraft pulp & paper mill

We implemented the proposed methodology on an industrial case study introduced in (Vakkilainen & others, 2005), involving a black liquor recovery boiler (BLRB) in a pulp & paper mill located in Eastern Canada. The BLRB is one of the major equipment in Kraft P&P mills, made up of several units and it is very important to maintain the BLRB process at a high level of efficiency and availability. The BLRB restores the used chemicals in the process, removes wastes and disposable by-products, and combusts the organic components from the black liquor for steam production. A schematic of the BLRB is depicted in Fig. 11.

The proposed causality analysis methodology exploits the sensory data collected from the BLRB in order to generate an event log comprising sequentially ordered events, and afterwards build a process model that represents the evolution of the system and causality relations. The data consists of

75,694 observations including 72 variables: 16 controllable and 56 measured. The observations were collected every five minutes for almost one year with an average of 275 observations per day. The dataset contains two KPIs: *Vapeur vs Solide* (m^3/Kg) (KPI-1), and *STR corrigé à 8% cheminée* (KPI-2). The KPI-1 is an economical indicator that indicates the ratio of the steam production to the black liquor consumption. The KPI-2 is an environmental indicator and measures the amount of total reduced Sulphide (TRS). The dataset was prepared and pre-processed (including removing noise and outliers) by the process expert and by using the EXPLORE software (NRCan, 2015).

Phase 1: definition of operating regions and pattern extraction from the BLRB dataset

In the initial phase, thresholds for the KPIs are set by the process expert to define the operating regions (classes). For KPI-1 the threshold was chosen to be **3.5** t vapeur/t solides and **8** pmm for KPI-2. Therefore, 4 operating classes are defined namely, *Normal, Abnormal_1, Abnormal_2, Abnormal_3*, which are used to label each of the 75,694 observations. The *Normal* class refers to both KPIs having acceptable values; *Abnormal 1* refers to only KPI-1 being acceptable; *Abnormal_2* indicates that only KPI-2 is acceptable; and *Abnormal_3* refers to neither KPIs having acceptable values. A graphical illustration of this step is shown in Fig. 12, considering the KPIs and the four operating classes defined. As mentioned, the output of this step is labeling the dataset where each observation is assigned one of the operating classes based on each observation's KPI values. The labels are distributed in the dataset as 35,322 observations for *Normal* class, 19,458 for *Abnormal_1* class, 10,854 for *Abnormal_2* class, and 10,060 for *Abnormal_3* class.

In the next step, the dataset is imported in Python for pattern generation using DT classifiers with the measured and controllable variables as the input variables, and the operating classes defined from the first step as the target variable. The DTs algorithm was applied on the dataset using the hold-out strategy with 80% for training, and 20% for testing using the **scikit-learn** package (Pedregosa et al., 2011), scoring a test accuracy of 84.25%. We applied the IML enhancement method found in (Dhurandhar et al., 2018) in order to enhance the performance of the DTs classifier (the student) while maintaining its interpretability characteristic to generate meaningful patterns. The method is explained in *Phase 1* of the proposed methodology and illustrated in Fig. 8. For the teacher model, we utilized the Random Forests classifier as a powerful model (teacher) to obtain a higher testing accuracy, scoring 92.71% on the testing data. The prediction probabilities from the Random Forest model were used to assign weights to each observation in the dataset (Pedregosa et al., 2011). The DTs model (the student) is trained again with the weighted dataset, to avoid difficult to classify observations, while focusing on the relatively easier observations that would contribute to better generalization performance and higher testing accuracy.

Afterwards, we perform hyperparameter optimization with GridSearch and post-pruning to obtain high-quality patterns that can represent the data and the different operating states of the system. Four hyperparameters were optimized with Gridsearch:

- min_samples_leaf ranging from (1–10);
- max_depth ranging from (1–14);
- min_samples_split ranging from (2–10);
- criterion = [gini, entropy]

The DT with the highest accuracy was found from the Gridsearch with the following hyperparameters: {min_samples_leaf = 5; max_depth = 5; min_samples_split = 2; criterion = gini}. Post-pruning was performed using the cost_complexity_pruning_path function to avoid overfitting to the training set and to improve the model generalization. The final model obtained after applying the IML enhancement method in (Dhurandhar et al., 2018), hyperparameter optimization, and post-pruning scored 89.95% on the test set. The values of precision recall and F1-Score for each class are shown in the confusion matrix given in Fig. 13. These values are calculated as follows: $precsion = \frac{TP}{TP+FP}$, $recall = \frac{TP}{TP+FN}$. The F1-score is calculated based on the average of both precision and recall, defined as: $F1 = 2 \cdot \frac{precsion \cdot recall}{precsion+recall} = \frac{2TP}{2TP+FP+FN}$, where TP (true positives), FP (false positives), TN (true negatives) and FN (false negative). The readers may be referred to (Witten et al., 2016) for more details.

Furthermore, we used the *k*-fold cross-validation method which splits the data into *k* smaller independent sets (folds) to avoid overfitting (Witten et al., 2016). In that method, the classification model is trained *k* times, each time with k-1 folds for training and the remaining fold is used for testing. The result is the average of the computed accuracies for all testing folds. In this work, we set k = 5 and the obtained testing accuracies were 90.4%, 88.8%, 91.1%, 89.7%, and 88.7%, with a mean of 89.7%. The model generated 32 patterns including 8 '***Normal***' patterns, 9 '***Abnormal_1***' patterns, 9 '***Abnormal_2***' patterns, and 6 '***Abnormal_3***' patterns. Some of the extracted patterns are represented in Table 2. As shown in the table, each pattern is a combination of IF–THEN rules representing the paths from the top node of the DT classifier to each leaf node.

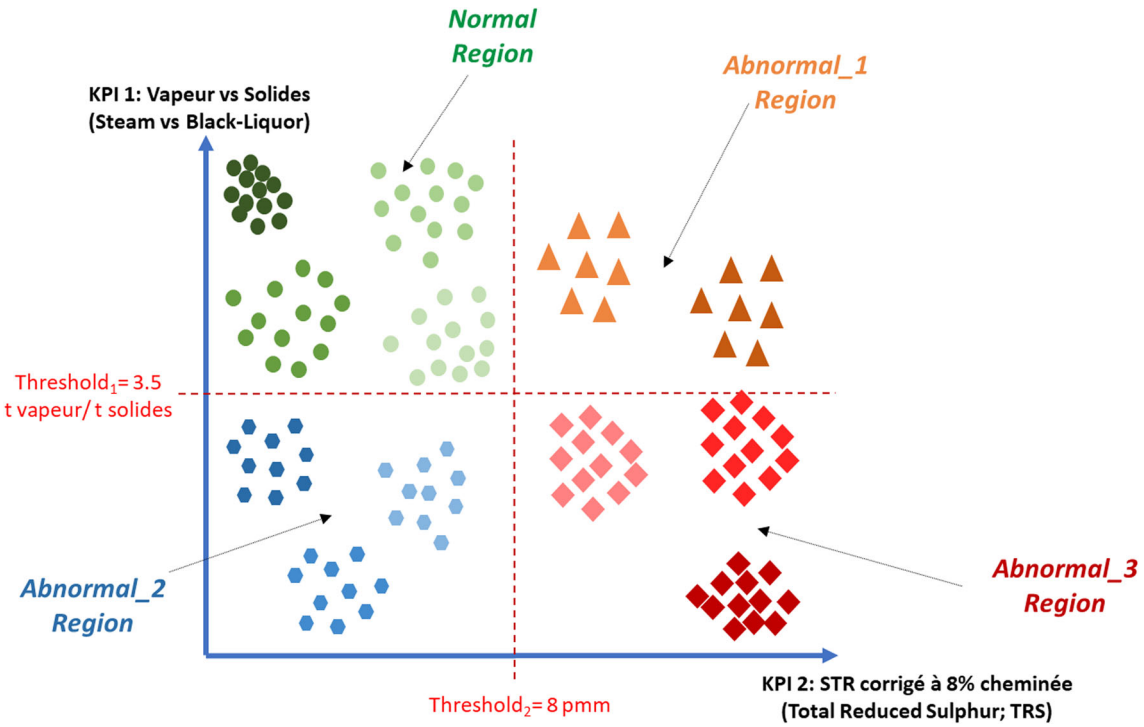

Fig. 12 Operating region definition based on the experienced KPI thresholding

	Normal	Abnormal_1	Abnormal_2	Abnormal_3	Recall	Precision	F1-Score
Normal	6109 (93.0%)	33 (0.5%)	421 (6.4%)	17 (0.3%)	92.8%	91.3%	92.1%
Abnormal_1	72 (4.5%)	1426 (88.0%)	17 (1.1%)	102 (6.3%)	88.1%	90.2%	89.2%
Abnormal_2	500 (15.0%)	8 (0.2%)	2795 (84.0%)	15 (0.5%)	84.2%	85.4%	84.8%
Abnormal_3	8 (0.4%)	113 (6.0%)	41 (2.2%)	1731 (91.0%)	91.4%	92.8%	92.1%

Fig. 13 Confusion matrix of the enhanced DT classifier

Phase 2: generating the event log and discovering the causal model for the Reboiler case study

The main objective of using the DTs was for interpretation, i.e., to extract the rules/patterns governing the classification, which is a significant strength of DTs. By applying an algorithm coded in Python, the patterns of the DTs are converted into IF–THEN rules to be mapped to the original dataset. The mapping would allow assigning each observation to one of the 32 extracted patterns, which satisfies its value ranges. The original dataset is now transformed into an event log, which comprises events (the extracted patterns), each with a timestamp and a *Case id*. In this event log, the *Case id* represents the events that occurred in one day of operation.

The constructed event log is imported into the ProM 6.9 software (Dongen et al., 2005) for analysis and process discovery. The ProM 6.9 software is one of the most famous open-source process mining tools. The software offers a wide variety of techniques for process discovery. Furthermore, the tool comprises plugins that execute conformance checking and process enhancement by providing new event logs to the existing model extracted in the discovery phase. By analyzing the event log, it was found that some noise exists especially for the *Case* start events, e.g., some *Cases* started with an event only one time through the whole event log and this can be considered as noise. Therefore, the ProM package, namely, "Filter Log Based on Heuristics," was utilized in order to remove noise from the starting events. The preprocessing step can be important, due to the significance of cleaning the event log and filtering the outliers in discovering accurate and representative models. The filtering threshold was adjusted to the default value of 0.8 based on the occurrence frequency. The frequencies of all event occurrence, start events, and end events are shown in Table 3. These frequencies can help the operators identify the persistent as well as the rare events.

Table 2 Examples of patterns extracted from the BLRB data using the DT classifier

Pattern	Variable description	Variable type	Pattern rules		Operating Class
P_{12}	X_{42}: Ratio air (m3)/ Solides (kg)	Controllable	$X_{42} \leq 3.18$	AND	*Abnormal_2*
	X_{41}: Solides Recup	Measured	$X_{41} \leq 1937.82$	AND	
	X_{30}:Débit air tertiaire (m^3/min)	Measured	$X_{30} > 278,035$	AND	
	X_8: Débit air secondaire nord (m^3/min)	Controllable	$X_8 \leq 696.49$	AND	
	X_{72}: Émanation CO becs coulée nord	Measured	$X_{72} \leq 0.4$		
P_{16}	X_{42}: Ratio air (m3)/ Solides (kg)	Controllable	$X_{42} \leq 3.18$	AND	*Normal*
	X_{41}: Solides Recup	Measured	$X_{41} \leq 1937.82$	AND	
	X_{30}:Débit air tertiaire (m^3/min)	Measured	$X_{30} > 278,035$	AND	
	X_8: Débit air secondaire nord (m^3/min)	Controllable	$X_8 > 696.49$	AND	
	X_9: Débit air tertiaire (m^3/min)	Controllable	$X_9 > 380.01$		
P_{28}	X_{42}: Ratio air (m3)/ Solides (kg)	Controllable	$X_{42} \leq 3.18$	AND	*Abnormal_3*
	X_{41}: Solides Recup	Measured	$X_{41} > 1937.82$	AND	
	X_{37}:Ratio air primaire (%)	Controllable	$X_{37} > 33.83$	AND	
	X_{30}:Débit air tertiaire (m^3/min)	Measured	$X_{30} \leq 285, 452.16$	AND	
	X_3: Pression vapeur atomisation (kPa)	Controllable	$X_3 > 860.39$		
P_{36}	X_{42}: Ratio air (m3)/ Solides (kg)	Controllable	$X_{42} > 3.18$	AND	*Abnormal_1*
	X_{38}: Ratio air secondaire (%)	Controllable	$X_{42} \leq 3.32$	AND	
	X_{35}: Analyseur Oxygène (%)	Measured	$X_{38} \leq 32.54$	AND	
	X_4: Débit savon brûlé (lpm)	Controllable	$X_{35} \leq 1.55$	AND	
			$X_4 \leq 15.74$		

The next step is the model/process discovery, where the event log is processed by PM discovery techniques to generate causal models that allow conducting visual and mathematical analysis of the process. Therefore, we ran the IMf technique on the event log with a threshold parameter of 0.3. The Petri net model generated after running the IMf algorithm is depicted in Fig. 14.

As shown in the figure, the start *Place* is the circle with the blue color, which contains one token (black circle) to start the process. The end *Place* is the circle with the grey color. When a token reaches the end *Place*, it corresponds to the ending of a *Case*, thus a token in the end *Place* will leave all other *Places* empty. The discovered model comprises the different patterns (*transition*s in Petri net notations) represented by the colored rectangles. Each color represents the operating class of the pattern (*Normal, Abnormal _1, Abnormal_2, Abnormal_3*). The black *transition*s are silent *transition*s. These silent *transition*s do not represent events in the real process or the event log, however, they are important to allow the model to represent some complex sequences in the real process. Event loops and event skipping are examples of such sequences. Undetermenism can be observed in the model where there can be a choice or conflict between two or more events and only one can occur, such as the conflict between *P23* and *P12*. The firing of a *transition* corresponds to the occurrence of an event in the real process and will cause a change in the distribution of tokens. By simulating the model with a Petri net simulator software like Workflow Petri Net Designer (WoPeD), the dynamics of the system could be observed and the operator can track the system's trajectory in real-time through the tokens' movements (marking change). Since each event is classified with one of the operating classes, operators can gain insights into avoiding specific events that would lead to abnormal operation and, thus, maintaining normal KPI values.

Following the model discovery, we conducted conformance checking to calculate the fitness and precision of the generated model. Specifically, we utilized the 'Replay a log on Petri Net for Conformance Checking' package in ProM 6.9 in order to compare the *Cases* found in the log with the discovered Petri net model. The model scored a total *Case* fitness of 98.66% and a precision score of 100%, where the former measures the percentage of the event log *Cases* that aligns with the discovered model, while the latter measures how precise the model is in not allowing for extra behavior that does not exist in the event log.

An important trade-off can be found between the accuracy of the DT classifier and the number of extracted patterns. Ultimately, we aim to find a DT model with the highest accuracy, without generating a large or excessive number of patterns. Defining a large number of patterns is subjective to the processes being modeled, however, too many patterns can result

Table 3 Frequency of events

Event	All events		Start events		End events	
	Occurrence (absolute)	Occurrence (relative %)	Occurrence (absolute)	Occurrence (relative %)	Occurrence (absolute)	Occurrence (relative %)
P36	1849	19.14	29	12.90	24	10.67
P40	1372	14.20	8	3.56	9	4.00
P54	1219	12.62	21	9.33	17	7.56
P46	751	7.78	20	8.90	18	8.00
P13	640	6.63	17	7.56	15	6.67
P55	571	5.91	63	28.00	50	22.22
P43	507	5.25	12	5.33	10	4.44
P12	443	4.59	–	–	–	–
P6	306	3.17	9	4.00	7	3.11
P9	247	2.56	21	9.33	19	8.44
P5	237	2.45	8	3.56	8	3.56
P16	232	2.40	–	–	–	–
P44	163	1.69	8	3.56	–	–
P28	160	1.66	9	4.00	–	–
P39	159	1.65	-	–	–	–
P15	132	1.37	–	–	–	–
P8	128	1.33	–	–	–	–
P21	79	0.82	–	–	–	–
P37	75	0.78	–	–	–	–
P27	73	0.76	–	–	8	3.56
P47	65	0.67	–	–	–	–
P30	64	0.63	–	–	–	–
P23	50	0.52	–	–	–	–

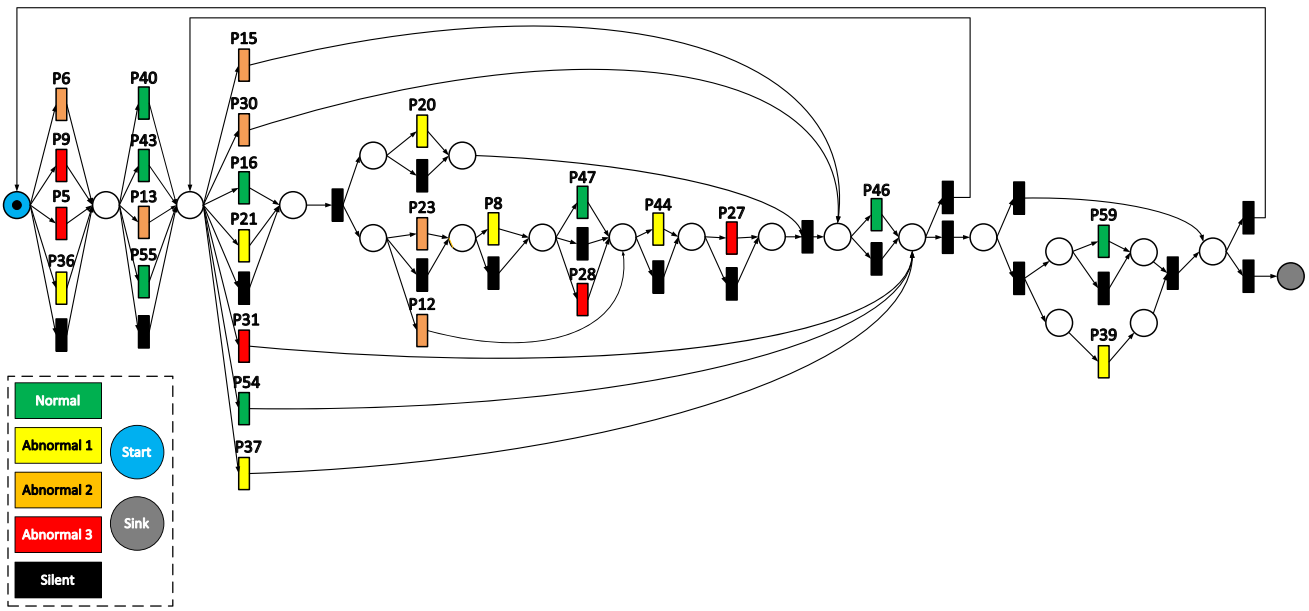

Fig. 14 The Petri Net discovered from the BLRB dataset using the inductive miner infrequent behavior technique

in the causal model discovered from process mining being too complex, difficult to visualize and track the system's trajectory, and in some cases having a low model fitness (accuracy) score. Thus, a minimal number of meaningful and representative patterns is desired, while maintaining a high accuracy from the classification DT model. However, most importantly, these patterns have to accurately reflect the events that happen in the process.

Based on the process expert's validation of the generated model, the approach was able to identify the effects of manipulating the controllers' set points on the measured variables and the system's state, which was displayed in the extracted patterns. Moreover, the Petri net model revealed the dynamics of the process in terms of the state evolution, which helped the expert identify some of the persistent trends regarding event sequences that occur frequently during the process operation. We would like to share with the readers a few, among many, insights obtained by the process expert through the analysis of the discovered PN model.

This model allowed the expert to pinpoint some of the events or control setpoints that have a high probability of transitioning the system to low KPIs operating regions, e.g. *P27* (*Abnormal_3 transition*), where setting the controllable variable $X3 < 860.38$, while the measured variable $X42 < 3.19$ will cause an abnormal operating state. Similarly, some of the events or control set point ranges are realized that can maintain the process under normal behavior, e.g. *P16*, where setting the two controllable variables $X8 > 696.5$, $X9 > 380$, while the measred variables $X42 < 3.19$, $X41 < 1937.83$ can maintain or transition the system to a normal operating state. Particularly, *P16* is an example of the approach's ability to identify the interactions between the controllers and their effect on the system's KPIs.

According to the model, if *P44* occurred, which is defined as $\{3.18 < X42 <= 3.32, X38 > 32.5, X30 <= 278{,}961.4, X41 > 1884\}$, the KPIs' values fall in the *Abnormal_1* region. The variable *X30* only is the only measured variable and the rest are controllable, therefore, this is a controllable event. Based on the model, when *P44* occurs, *P27,* or *P46* can most probably occur afterwards and both are controllable events. The event *P27* is defined as $\{3.18 < X42 <= 3.32, X38 > 32.5, X30 <= 278{,}961.4, X41 <= 1800\}$, while the event *P46* is defined as $\{3.18 < X42 <= 3.32, X38 > 32.5, X30 <= 278{,}961.4, 1800 < X41 <= 1884\}$. The event *P27* leads the system to *Abnormal_3* region, while *P46* returns it to the *Normal* region. It can be observed from the definition of those two events that the difference is in the controllable variable *X41*. Therefore, based on dependency information from the model regarding these three events and their definition, the expert got this important conclusion:

• ***If P44 occurred, setting the controllable variable X41 afterward to a value ≤ 1800 would further deteriorate the performance from Abnormal_1 to Abnormal_3, while if X41 is set to be between 1800 and 1884, the system can be returned back to the Normal region.***

Based on the dependency relationships in the discovered PN model, the event *P12* can mostly occur after *P21* occurs. The definition of *P12* is $\{X42 \leq 3.18, X41 \leq 1937.82, X30 > 278{,}035, X8 \leq 696.49, X72 \leq 0.4\}$, *where X30 and X72* while the definition *of P21* is $\{X42 > 4.32, X41 \leq 1937.82, X30 > 278{,}035, X8 \leq 696.49, X72 \leq 0.4\}$. It can be seen that the difference between both events is the range of the controllable variable *X42*. Once this condition $\{X41 \leq 1937.82, X30 > 278{,}035, X8 \leq 696.49, X72 \leq 0.4\}$ is satisfied and *X42* is set to a value ≤ 3.18 (event *P12*), the system would transition to the abnormal region *Abnormal*_2. However, based on the PN model, this specific event sequence *(P21 then P12)* could be avoided, if instead of executing *P21*, the operator can execute *P54,* which leads to the *Normal* region, since there is a choice between *P54 and P21*. The definition of the event P54 is $\{X41 \leq 1937.82, X30 > 278{,}035, X8 > 700, X72 \leq 0.4\}$. It can be observed that the difference between the two events *P54 and P21* is in the controllable variable *X8*. Therefore, the expert got a second important conclusion:

• ***When this condition {X41 ≤ 1937.82, X30 > 278,035, X72 ≤ 0.4} is satisfied, setting X8 to a value > 700 would transition the system to a Normal region, and avoid other events that can lead to abnormality such as the event P12.***

Finally, we conducted a quantitative analysis on the Petri net model. For such analysis, we exploited some packages in ProM 6.9 such as the 'Analyze Behavioral Property of Petri Net' package, as well as the WoPeD (Freytag, 2005) software for simulation. Based on the analysis, we found that the discovered model is safely bounded where no *Place* can hold more than one token; the model has no dead *transition*s; the model is not live, i.e. the dynamics of the model does not continue infinitely, where a *Case* has to begin from an initial marking and has to terminate by reaching the end marking or sink *Place*. Furthermore, the model doesn't comprise dead markings (except the sink/end *Place*), and it is deadlock-free, where it is always guaranteed that at least one *transition* will be enabled to fire from any marking. Finally, using the WoPeD software we generated the coverability graph to have a detailed analysis of the model's dynamic. The graph is shown in Fig. 15.

As shown in the figure, the colored rectangles represent the different markings, which are interpreted as the different states of the system. The directed arrows represent the dynamics of the system. Each arrow has a label comprising a single or a group of *transition*s connected with OR gates. The occurrence of one of the events on the label will cause the transition from the previous state (beginning of the arrow) to the next state (tip of the arrow). Causality can be inferred by analyzing the sequence of *transition*s, i.e. how specific events always follow other events. Furthermore, we are able to conduct a model simulation on the software using the "To-

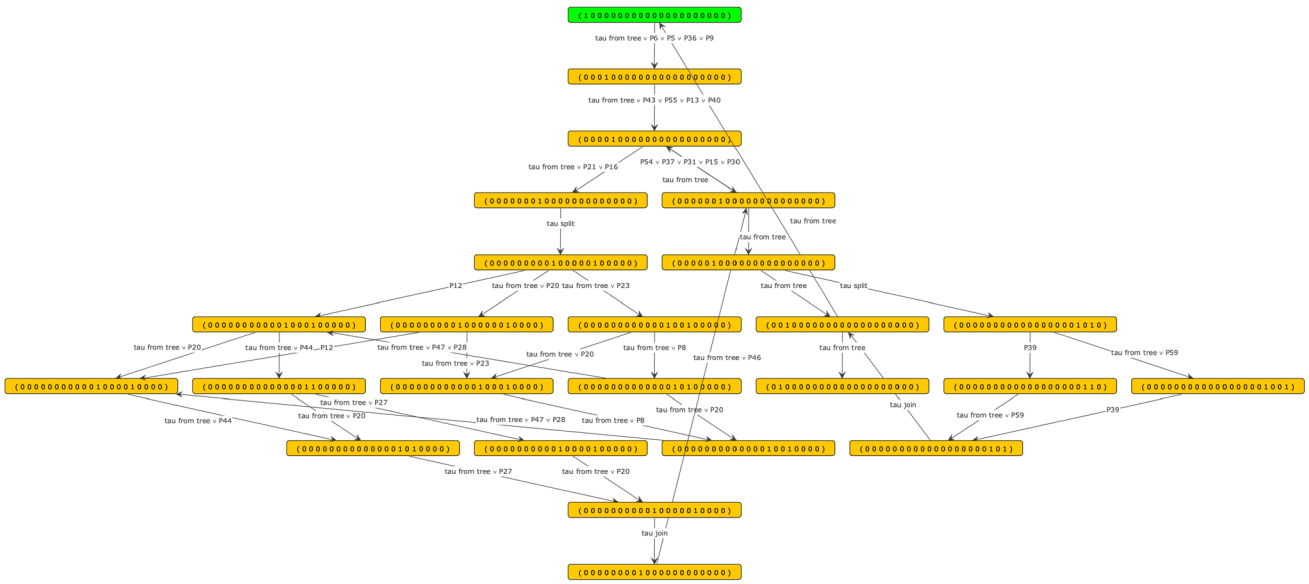

Fig. 15 Coverability graph of the discovered Petri net model

kenGame" option in WoPeD displaying visual animation of the progression of the tokens inside the Petri net *Places* by firing the *transition*s (events/patterns). The simulation serves as a dashboard for operators to track the system's dynamics, providing a complete overview of the process in real-time. The generated causal model obtained is useful for the process operator, allowing him to deeply understand the different causes for the abnormal situations related to the deterioration of both KPIs. Consequently, the operator gained insights in selecting the appropriate corrective actions that align the process performance with the desired objectives towards energy efficiency and the reduction of GHG emissions.

Discussion and future work

The increasing availability and development of sensor technology and data acquisition systems have introduced huge monitoring capabilities for industrial systems, serving several crucial activities such as fault diagnosis, supervisory control, predictive maintenance, and others (Reis & Gins, 2017). One of the main goals of system monitoring is to develop analytic tools that provide operators with estimates of the system's states and conditions based on causal models that reflect the actual behavior of the monitored system and detect abnormalities, along with their root causes (Yang et al., 2014). The efficiency of monitoring techniques in these complex industries is heavily dependent on the possession of precise models. Therefore, we propose a multistep methodology that integrates IML, PM, and human expertise to generate causal models that can accurately describe the system's dynamics and state evolution. We believe that this methodology could have a significant positive impact on industrial applications, particularly for complex processes.

Our approach is data-driven, where historical data that is collected from sensors inside the actual process is exploited without the dependence on domain knowledge and existing models. In complex processes such as pulp and paper mills, this introduces a great advantage by conserving the experts' time and efforts to construct accurate and reliable models, which is rarely feasible in such large-scale processes. However, it is important to mention that our goal is not to omit human involvement or responsibilities. What we seek is to harness our AI-based methodology to support the human experts in such complicated tasks, where the expert's indispensable background and knowledge will be used to verify, validate and supervise the model generated by our proposed approach.

The concept we used in our methodology where we extract patterns from measured and controllable continuous variables to be employed as process events allows for analyzing and identifying the interactions between the controllers. This can support operators in the development of enhanced control strategies that allow controllers to work more cooperatively, rather than competitively.

It is important to mention the effect of the IML performance on the causal model generation. In order to effectively inspect the events that contribute to changing or maintaining operation performance, we need to accurately define those events, which translates to extracting patterns. The reason is that in the real process, a slight change of one or more variables whether they are manipulated or measured could cause a change in the KPIs. In this approach, the patterns extracted from the data using the IML represent the events that occur

in the process. Therefore, it is essential to ensure generating strong and representative patterns, as they will be used to construct the event log. This event log will be the input to process mining techniques to build the casual model. Thus, the accuracy of the patterns increases the creditability of the final causal model, where these patterns would be more reliable in describing the actual events that happen in reality and affect the system's KPIs.

Our approach constructs dynamic causal models by inspecting the temporal information between the interactive process variables to infer causality relationships and allow for sequential event-based modeling. Based on the literature, we found that the majority of existing methods do not incorporate such temporal relations, which can significantly enhance the modeling accuracy and credibility. We achieve this in our approach by utilizing PM, which has emerged as an exceptional framework for discovering and enhancing business processes. According to (Van der Aalst, 2016) who is a PM pioneer, PM has been successfully implemented in over 100 organizations, varying from government departments like Justice Department, and Centraal Justitieel Incasso Bureau, media agencies like Winkwaves, Health care systems like Catharina hospital and AMC hospital, insurance companies, and banks. These organizations are classified as Lasagna processes, which are characterized by a relatively clear structure. However, we have not found applications of PM in large-scaled industrial systems. Therefore, we are aiming to be the first to apply PM for modeling complex industrial systems.

The dependency on KPIs for defining the operating regions allows for the process events to be classified as normal or abnormal from an environmental and economical perspective. Since PM generates state-graph models such as PNs, the inclusion of KPI information during the model discovery allows for normal and abnormal behaviors to be incorporated in the generated model, which can be utilized as an efficient visualization tool that serves as a dashboard for process operators to track the system's trajectory and allow for real-time process simulation. This can also allow for crucial applications such as fault diagnosis and prognosis, and supervisory control.

Furthermore, the PM framework includes model enhancement features based on augmented data. Therefore, updating the model can be performed automatically whenever new data is collected. The update process is data-driven and thus, reduces the human effort needed for reconstructing the model in the case of complex processes. As a result, the approach can serve as an important building block in the formation of digital twins. Particularly, the modeling and simulation of a digital twin comprise four primary components, namely, planning simulation (design simulation), simulation for virtual commissioning, training simulation (OTS: operator training system), and simulation during operation (Oppelt et al. 2020). The proposed methodology can contribute to the fourth component, which focuses on the simulation of the actual process through the integration of the virtual model and system operation in real-time. This integration can provide important insights about the actual process and allows for improved monitoring and enhanced real-time decision support, which can elevate the system's performance.

In this work, defining the operating regions is mainly dependent on the KPI-related thresholds set by the expert. However, in some industrial data, this task needs careful attention to ensure that the specified KPIs partition the data properly. Therefore, one of our future research directions is to automatically partition the data using unsupervised machine learning methods. Of course, everything related to data partitioning will be done under the full supervision of the human expert who will verify and validate the results obtained from these unsupervised methods. In our approach, we have raw low-level continuous data coming from a complex industrial process and we use IML to identify the representative patterns and convert them into high-level events to build the event log. Also, in this work, we treat the events in the event log data as discrete. However, the involvement of timed events can be one of our future research directions.

Other, future work involves fault prognosis, since the generated model allows visualizing the system's state evolution in real time, and thus operators can predict the transition to faulty states before they occur. Another future work is the application of predictive maintenance by incorporating maintenance activity datasets to support improved planning & scheduling and to increase the lifetime of equipment. Furthermore, since powerful supervisory control systems for industrial systems rely heavily on accurate and representative models, building precise causal models from data is a cornerstone for applying efficient supervisory control methodologies to prevent abnormal events and enhance performance. Particularly, the generated state-event graph from our approach is classified as a Markov decision model, which can be the platform to apply an AI-based supervisory control approach. In that case, the KPIs would be key for formulating optimization objective functions to modify control setpoints. Moreover, some of the extracted patterns may contain value ranges for measured variables only, and such patterns represent real events that are uncontrollable in the process (they do not comprise controllable variables for the operators to control). Therefore, the supervisory control approach in this case can attempt to avoid these uncontrollable events beforehand. Finally, we are looking forward to conducting a quantitative analysis to measure the direct environmental and economic impacts of applying our approach to process industries. Our main goal for proposing this approach is developing reliable tools that can identify the main factors for inefficient control and energy consumption to increase industrial profits and to reduce GHG emissions.

Conclusion

The majority of existing data-driven causality analysis methods face challenges when applied to industrial applications such as the ability to model interactions between controllers, incorporating temporal information, and identifying events' sequential relations that lead to process performance deterioration and not just equipment faults. Therefore, in this paper, we have proposed an approach to construct high-level dynamic causal models from low-level observations based on the exploitation of historical operation data in industrial processes. The approach integrates interpretable machine learning (IML), process mining (PM), and human expertise to build an accurate and representative dynamic causal model that is able to analyze events' sequential relations, deal with a high number of variables, identify the interactions between system components, and incorporate the normal and abnormal behavior of the system. The most common industrial data is numerical, which contains continuous observations of measured and controllable variables in addition to key performance indicators (KPIs). The approach realizes the important events from these observations through the use of IML techniques. These events are identified by extracting patterns from the input data, based on different KPI-related operating regions set by the expert. Furthermore, an IML enhancement technique based on transfer learning is adapted to ensure the generation of accurate and representative patterns. These patterns can model the interactions between the controllers (controllable events) as well as uncontrollable events. The identified events are then used to generate a discrete event log for the process. Afterward, the dynamic casual model is built using PM discovery techniques. More specifically, the PM takes as input the discrete event log and finds the appropriate cuts that explain the dependency relationships between the events. These relationships are represented in a state-event graph such as Petri nets, which allows for process simulation and causality inference. Process experts can use the generated Petri net model as a dashboard to track the system's trajectory and state evolution. The approach is tested successfully on an industrial dataset collected from a recovery boiler in a pulp & paper mill and validated using conformance checking (a class of process mining tasks) to measure the discovered model's fitness (accuracy) and precision in representing the event log. In addition, the modeling procedure and the results have been validated and verified by the process expert. A quantitative analysis of the generated model is conducted to ensure model appropriateness as well as to infer causal relationships between events, allowing for better decision-making in fault prevention. This approach can serve various future applications that are critical to the industry, such as supervisory control, predictive maintenance, and fault prognosis. In addition, an automatic model update is facilitated whenever new data is collected, which can have a significant role in building a virtual digital twin for the system.

Acknowledgements This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) under research grant number 231695 and the Natural Resources Canada's OERD (Office for Energy Research and Development) program.

Funding Open access funding provided by Natural Resources Canada. Another funding by Office of Energy Research and Development (OERD).

References

Aizpurua, J. I., & Muxika, E. (2013). Model-based design of dependable systems: limitations and evolution of analysis and verification approaches. *International Journal on Advances in Security*, *6*(1).

Bauer, M., & Thornhill, N. F. (2008). A practical method for identifying the propagation path of plant-wide disturbances. *Journal of Process Control, 18*(7–8), 707–719. https://doi.org/10.1016/j.jprocont.2007.11.007

Bozorgi, Z. D., Teinemaa, I., Dumas, M., La Rosa, M., & Polyvyanyy, A. (2020). Process Mining Meets Causal Machine Learning: Discovering Causal Rules from Event Logs. *2020 2nd International Conference on Process Mining (ICPM)*, 129–136. IEEE.

Chen, H.-S., Yan, Z., Yao, Y., Huang, T.-B., & Wong, Y.-S. (2018). Systematic procedure for Granger-causality-based root cause diagnosis of chemical process faults. *Industrial & Engineering Chemistry Research, 57*(29), 9500–9512.

Chiang, L. H., & Braatz, R. D. (2003). Process monitoring using causal map and multivariate statistics: Fault detection and identification. *Chemometrics and Intelligent Laboratory Systems, 65*(2), 159–178. https://doi.org/10.1016/S0169-7439(02)00140-5

Chiang, L. H., Jiang, B., Zhu, X., Huang, D., & Braatz, R. D. (2015). Diagnosis of multiple and unknown faults using the causal map and multivariate statistics. *Journal of Process Control, 28*, 27–39.

Connection, C. C. (2018). MANITOBA LARGE FINAL EMITTERS (LFE). Retrieved from https://climatechangeconnection.org/emissions/manitoba-ghg-emissions/manitoba-large-final-emitters-lfe/

David, R., & Alla, H. (2010). Discrete, continuous, and hybrid Petri nets (Vol. 1, pp. 17–130). Berlin: Springer.

De Medeiros, A. K. A., Van Der Aalst, W. M. P., & Weijters, A. J. M. M. (2003). Workflow Mining: Current Status and Future Directions. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *2888*(June 2014), 389–406. https://doi.org/10.1007/978-3-540-39964-3

Derigent, W., Cardin, O., & Trentesaux, D. (2020). Industry 4.0: contributions of holonic manufacturing control architectures and future challenges. *Journal of Intelligent Manufacturing*, 1–22.

Dhurandhar, A., Shanmugam, K., Luss, R., & Olsen, P. A. (2018). Improving simple models with confidence profiles. *Advances in Neural Information Processing Systems*, 10296–10306.

Diba, K., Batoulis, K., Weidlich, M., & Weske, M. (2020). Extraction, correlation, and abstraction of event data for process mining. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 10*(3), e1346.

Dongen, B., Verbeek, H., Weijters, A., & Aalst, W. (2005). The ProM Framework: A New Era in Process Mining Tool Support. In *Lecture Notes in Computer Science* (Vol. 3536). https://doi.org/10.1007/11494744_25

Elhefnawy, M., Ragab, A., & Ouali, M.-S. (2021). Fault classification in the process industry using polygon generation and deep learning. *Journal of Intelligent Manufacturing*, 1–14.

Fahland, D., & Van Der Aalst, W. M. P. (2015). Model repair - Aligning process models to reality. *Information Systems, 47*, 220–243. https://doi.org/10.1016/j.is.2013.12.007

Fei, H., Chaojun, W., & Shu-Kai, S. F. (2019). fault detection and root cause analysis of a batch process via novel nonlinear dissimilarity and comparative granger causality analysis. *Industrial & Engineering Chemistry Research, 58*(47), 21842–21854.

Freytag, T. (2005). Woped–workflow petri net designer. *University of Cooperative Education*, 279–282.

Ge, Z., Song, Z., Ding, S. X., & Huang, B. (2017). Data mining and analytics in the process industry: The role of machine learning. *IEEE Access, 5*, 20590–20616. https://doi.org/10.1109/ACCESS.2017.2756872

Gil, G. J. D. G., Alabi, D. B., Iyun, O. E., & Thornhill, N. F. (2011). Merging process models and plant topology. *International Symposium on Advanced Control of Industrial Processes (ADCONIP), 2011*, 15–21.

Granger, C. W. J. (1969). Investigating causal relations by econometric models and cross-spectral methods. *Econometrica: Journal of the Econometric Society*, 424–438.

Grisold, T., Wurm, B., Mendling, J., & Vom Brocke, J. (2020). Using process mining to support theorizing about change in organizations. *Proceedings of the 53rd Hawaii International Conference on System Sciences*.

Kabir, S. (2017a). An overview of fault tree analysis and its application in model based dependability analysis. *Expert Systems with Applications, 77*, 114–135.

Kabir, S. (2017b). An overview of fault tree analysis and its application in model based dependability analysis. *Expert Systems with Applications, 77*, 114–135. https://doi.org/10.1016/j.eswa.2017.01.058

Kabir, S., Papadopoulos, Y., Sharvia, S., & Walker, M. (2015). *Model-based dependability analysis: State-of-the-art, challenges and future outlook*. Amsterdam: Elsevier.

Landman, R., Kortela, J., Sun, Q., & Jämsä-Jounela, S. L. (2014). Fault propagation analysis of oscillations in control loops using data-driven causality and plant connectivity. *Computers and Chemical Engineering, 71*, 446–456. https://doi.org/10.1016/j.compchemeng.2014.09.017

Lee, C.-Y., & Chien, C.-F. (2020). Pitfalls and protocols of data science in manufacturing practice. *Journal of Intelligent Manufacturing*, 1–19.

Lee, M., Jeon, J., & Lee, H. (2021). Explainable AI for domain experts: a post Hoc analysis of deep learning for defect classification of TFT–LCD panels. *Journal of Intelligent Manufacturing*, 1–13.

Leemans, S. J. J., Fahland, D., & van der Aalst, W. M. P. (2013a). Discovering block-structured process models from event logs-a constructive approach. *International Conference on Applications and Theory of Petri Nets and Concurrency*, 311–329. Springer.

Leemans, S. J. J., Fahland, D., & van der Aalst, W. M. P. (2013b). Discovering block-structured process models from event logs containing infrequent behaviour. *International Conference on Business Process Management*, 66–78. Springer.

Leitner-Fischer, F., & Leue, S. (2013). Probabilistic fault tree synthesis using causality computation. International Journal of Critical Computer-Based Systems 30, 4(2), 119–143.

Li, J., Ma, S., Le, T., Liu, L., & Liu, J. (2016). Causal decision trees. *IEEE Transactions on Knowledge and Data Engineering, 29*(2), 257–271.

Li, J., & Shi, J. (2007). Knowledge discovery from observational data for process control using causal Bayesian networks. *IIE Transactions (institute of Industrial Engineers), 39*(6), 681–690. https://doi.org/10.1080/07408170600899532

Li, L., & Yue, W. (2020). Dynamic uncertain causality graph based on Intuitionistic fuzzy sets and its application to root cause analysis. *Applied Intelligence, 50*(1), 241–255.

Lindner, B., Auret, L., & Bauer, M. (2017). Investigating the impact of perturbations in chemical processes on data-based causality analysis. Part 1: Defining desired performance of causality analysis techniques. *IFAC-PapersOnLine, 50*(1), 3269–3274.

Lindner, B., Auret, L., Bauer, M., & Groenewald, J. W. D. (2019). Comparative analysis of Granger causality and transfer entropy to present a decision flow for the application of oscillation diagnosis. *Journal of Process Control, 79*, 72–84.

Liu, Y., Chen, H.-S., Wu, H., Dai, Y., Yao, Y., & Yan, Z. (2020). Simplified Granger causality map for data-driven root cause diagnosis of process disturbances. *Journal of Process Control, 95*, 45–54.

Malika, M., Khochmane, L., Bouzaouit, A., & Bennis, O. (2017). Transformation of fault tree into Bayesian network methodology for fault diagnosis. *Mechanics, 23*(6), 891–899.

Mansour, M. M., Wahab, M. A. A., & Soliman, W. M. (2013). Petri nets for fault diagnosis of large power generation station. *Ain Shams Engineering Journal, 4*(4), 831–842. https://doi.org/10.1016/j.asej.2013.04.006

Mantel, N., & Haenszel, W. (1959). Statistical aspects of the analysis of data from retrospective studies of disease. *Journal of the National Cancer Institute, 22*(4), 719–748.

Maurya, M. R., Rengaswamy, R., & Venkatasubramanian, V. (2003). A systematic framework for the development and analysis of signed digraphs for chemical processes. 1. Algorithms and analysis. *Industrial and Engineering Chemistry Research, 42*(20), 4789–4810. https://doi.org/10.1021/ie020644a

Maurya, M. R., Rengaswamy, R., & Venkatasubramanian, V. (2004). Application of signed digraphs-based analysis for fault diagnosis of chemical process flowsheets. *Engineering Applications of Artificial Intelligence, 17*(5), 501–518. https://doi.org/10.1016/j.engappai.2004.03.007

Maurya, M. R., Rengaswamy, R., & Venkatasubramanian, V. (2006). A signed directed graph-based systematic framework for steady-state malfunction diagnosis inside control loops. *Chemical Engineering Science, 61*(6), 1790–1810. https://doi.org/10.1016/j.ces.2005.10.023

Molnar, C. (2020). *Interpretable Machine Learning: A Guide for Making Black Box Models Explainable*. Lulu. com.

Murata, T. (1989). Petri nets: Properties, analysis and applications. *Proceedings of the IEEE, 77*(4), 541–580. https://doi.org/10.1109/5.24143

NRCan, N. R. C. (2015). *Improving Process Operation Using the Power of Advanced Data Analysis*.

Naghoosi, E., Huang, B., Domlan, E., & Kadali, R. (2013). Information transfer methods in causality analysis of process variables with an industrial application. *Journal of Process Control, 23*(9), 1296–1305. https://doi.org/10.1016/j.jprocont.2013.02.003

Nauta, M., Bucur, D., & Stoelinga, M. (2018). LIFT: Learning fault trees from observational data. *International Conference on Quantitative Evaluation of Systems*, 306–322. Springer.

Neapolitan, R. E. (2004). *Learning bayesian networks* (Vol. 38). Pearson Prentice Hall Upper Saddle River, NJ.

Oppelt, M., Lorenz, O., Leingang, C., & Pfeiffer, B.-M. (2020). Evolution of a digital twin. Part 1: the concept, Part 2: Use of the digital twin. *Petroleum Technology Quarterly*, *2020*, 35ff, 29ff.

Pearl, J. (2009). *Causality*. Cambridge: Cambridge University Press.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., & Dubourg, V. (2011). Scikit-learn: Machine learning in Python. *The Journal of Machine Learning Research, 12*, 2825–2830.

Peng, D., Geng, Z., & Zhu, Q. (2014). A Multilogic Probabilistic Signed Directed Graph Fault Diagnosis Approach Based on Bayesian Inference. *Industrial & Engineering Chemistry Research, 53*(23), 9792–9804. https://doi.org/10.1021/ie403608a

Petri, C. A. (1966). *Communication with automata: Volume 1 supplement 1*. APPLIED DATA RESEARCH INC PRINCETON NJ.

Pyun, H., Kim, K., Ha, D., Lee, C.-J., & Lee, W. B. (2020). Root causality analysis at early abnormal stage using principal component analysis and multivariate Granger causality. *Process Safety and Environmental Protection, 135*, 113–125.

Quinlan, J. R. (2014). *C4. 5: programs for machine learning*. Amsterdam: Elsevier.

Ragab, A., El-Koujok, M., Poulin, B., Amazouz, M., & Yacout, S. (2018). Fault diagnosis in industrial chemical processes using interpretable patterns based on Logical Analysis of Data. *Expert Systems with Applications, 95*, 368–383.

Ragab, A., El Koujok, M., Ghezzaz, H., Amazouz, M., Ouali, M. S., & Yacout, S. (2019). Deep understanding in industrial processes by complementing human expertise with interpretable patterns of machine learning. *Expert Systems with Applications, 122*, 388–405. https://doi.org/10.1016/j.eswa.2019.01.011

Ragab, A., Ouali, M.-S., Yacout, S., & Osman, H. (2014). Remaining useful life prediction using prognostic methodology based on logical analysis of data and Kaplan-Meier estimation. *Journal of Intelligent Manufacturing*. https://doi.org/10.1007/s10845-014-0926-3

Ragab, A., Yacout, S., Ouali, M.-S., & Osman, H. (2019). Prognostics of multiple failure modes in rotating machinery using a pattern-based classifier and cumulative incidence functions. *Journal of Intelligent Manufacturing, 30*(1), 255–274.

Reinkemeyer, L. (2020). *Process Mining in Action Principles, Use Cases and Outlook: Principles, Use Cases and Outlook*. https://doi.org/10.1007/978-3-030-40172-6

Reis, M. S., & Gins, G. (2017). Industrial process monitoring in the big data/industry 4.0 era: From detection, to diagnosis, to prognosis. *Processes*. https://doi.org/10.3390/pr5030035

Schölkopf, B., Locatello, F., Bauer, S., Ke, N. R., Kalchbrenner, N., Goyal, A., & Bengio, Y. (2021). Toward causal representation learning. *Proceedings of the IEEE, 109*(5), 612–634.

Sharvia, S., Kabir, S., Walker, M., & Papadopoulos, Y. (2016). Model-based dependability analysis: State-of-the-art, challenges, and future outlook. State-of-the-art, challenges, and future outlook. In *Software Quality Assurance: In Large Scale and Complex Software-intensive Systems* (pp. 251–278). https://doi.org/10.1016/B978-0-12-802301-3.00012-0

Spirtes, P. (2010). Introduction to causal inference. *Journal of Machine Learning Research, 11*(May), 1643–1662.

Spirtes, P., Glymour, C. N., Scheines, R., & Heckerman, D. (2000). *Causation, prediction, and search*. Cambridge: MIT Press.

Steiner, M., Keller, P., & Liggesmeyer, P. (2012). Modeling the effects of software on safety and reliability in complex embedded systems. *International Conference on Computer Safety, Reliability, and Security*, 454–465. Springer.

Sun, Y., Qin, W., Zhuang, Z., & Xu, H. (2021). An adaptive fault detection and root-cause analysis scheme for complex industrial processes using moving window KPCA and information geometric causal inference. *Journal of Intelligent Manufacturing*, 1–15.

Talbot, D., & Boiral, O. (2013). Can we trust corporates GHG inventories? An investigation among Canada's large final emitters. *Energy Policy, 63*, 1075–1085. https://doi.org/10.1016/j.enpol.2013.09.054

Thambirajah, J., Benabbas, L., Bauer, M., & Thornhill, N. F. (2007). Cause and effect analysis in chemical process utilizing plant connectivity information. *Advances in Process Control, 8*, 1–5.

Thambirajah, J., Benabbas, L., Bauer, M., & Thornhill, N. F. (2009). Cause-and-effect analysis in chemical processes utilizing XML, plant connectivity and quantitative process history. *Computers and Chemical Engineering, 33*(2), 503–512. https://doi.org/10.1016/j.compchemeng.2008.10.002

Van der Aalst, W., Adriansyah, A., & Van Dongen, B. (2012). Replaying history on process models for conformance checking and performance analysis. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 2*(2), 182–192. https://doi.org/10.1002/widm.1045

Van Der Aalst, W., Weijters, T., & Maruster, L. (2004). Workflow mining: Discovering process models from event logs. *IEEE Transactions on Knowledge and Data Engineering, 16*(9), 1128–1142. https://doi.org/10.1109/TKDE.2004.47

Vakkilainen, E., & others. (2005). *Kraft recovery boilers--Principles and practice*.

Van der Aalst, W. (2016). Process mining: Data science in action. In *Process Mining: Data Science in Action*. https://doi.org/10.1007/978-3-662-49851-4

Waghen, K., & Ouali, M.-S. (2019). Interpretable logic tree analysis: A data-driven fault tree methodology for causality analysis. *Expert Systems with Applications, 136*, 376–391.

Weijters, A. J. M. M., & Ribeiro, J. T. S. (2011). Flexible heuristics miner (FHM). In *IEEE SSCI 2011: Symposium Series on Computational Intelligence - CIDM 2011: 2011 IEEE Symposium on Computational Intelligence and Data Mining*. https://doi.org/10.1109/CIDM.2011.5949453

Wen, L., Wang, J., Aalst, W., Huang, B., & Sun, J. (2009). A novel approach for process mining based on Event Types. *Journal of Intelligent Information Systems, 32*, 163–190. https://doi.org/10.1007/s10844-007-0052-1

Werner-Stark, A., Gerzson, M., & Hangos, K. M. (2011). Model-based fault detection and isolation using process mining. *World Academy of Science, Engineering and Technology, 7*(73), 851–856.

Weske, M. (2012). Business process management: Concepts, languages, architectures, second edition. In *Business Process Management: Concepts, Languages, Architectures, Second Edition*. https://doi.org/10.1007/978-3-642-28616-2

Witten, I. H., Frank, E., Hall, M. A., & Pal, C. J. (2016). *Data Mining: Practical Machine Learning Tools and Techniques*. Hamilton: University of Waikato. Morgan Kaufmann.

Wu, X., Kumar, V., Quinlan, R., Ghosh, J., Yang, Q., Motoda, H., et al. (2007). Top 10 algorithms in data mining. *Knowledge and Information Systems*, *14(1)*. https://doi.org/10.1007/s10115-007-0114-2

Yang, F., Duan, P., Shah, S. L., & Chen, T. (2014). *Capturing connectivity and causality in complex industrial processes*. Berlin: Springer.

Yuan, T., & Qin, S. J. (2014). Root cause diagnosis of plant-wide oscillations using Granger causality. *Journal of Process Control, 24*(2), 450–459. https://doi.org/10.1016/j.jprocont.2013.11.009

Zhai, L., & Yang, J. (2020). Topological causality analysis of horizontal gas–liquid flows based on cross map of phase spaces. *Physics Letters A, 384*(27), 126693.

Zhang, Q., & Geng, S. (2015). Dynamic Uncertain causality graph applied to dynamic fault diagnoses of large and complex systems.

IEEE Transactions on Reliability, 64(3), 910–927. https://doi.org/10.1109/TR.2015.2416332

Zhang, X., Miao, Q., Fan, X., & Wang, D. (2009). Dynamic fault tree analysis based on Petri nets. *2009 8th International Conference on Reliability, Maintainability and Safety*, 138–142. IEEE.

Zhong, X., Xu, Y., Liu, Y., Wu, X., Zhao, D., Zheng, Y., et al. (2020). Root cause analysis and diagnosis of solid oxide fuel cell system oscillations based on data and topology-based model. *Applied Energy*, *267*, 114968.

Zhou, Z., & Zhang, Q. (2017). Model event/fault trees with dynamic uncertain causality graph for better probabilistic safety assessment. *IEEE Transactions on Reliability, 66*(1), 178–188.