

# Research Repository

## **Multi-level CEP Rules Automatic Extraction Approach for Air Quality Detection and Energy Conservation Decision Based on AI Technologies**

Accepted for publication in Applied Energy.

**Research Repository link:** <https://repository.essex.ac.uk/38672/>

### **Please note:**

Changes made as a result of publishing processes such as copy-editing, formatting and page numbers may not be reflected in this version. For the definitive version of this publication, please refer to the published source. You are advised to consult the [publisher's version](#) if you wish to cite this paper.

# Multi-level CEP Rules Automatic Extraction Approach for Air Quality Detection and Energy Conservation Decision Based on AI Technologies

Yuan Liu<sup>a,b,c</sup>, Wangyang Yu<sup>a,b,c,\*</sup>, Xiaojun Zhai<sup>d</sup>, Beiming Zhang<sup>a,b,c</sup>, Klaus D. McDonald-Maier<sup>d</sup> and Maria Fasli<sup>d</sup>

<sup>a</sup>Key Laboratory of Intelligent Computing and Service Technology for Folk Song, Ministry of Culture and Tourism, Shaanxi Normal University, No. 620, West Chang'an Street, Chang'an District, Xi'an, 710119, China

<sup>b</sup>School of Computer Science, Shaanxi Normal University, No. 620, West Chang'an Street, Chang'an District, Xi'an, 710119, China

<sup>c</sup>Anhui Province Engineering Laboratory for Big Data Analysis and Early Warning Technology of Coal Mine Safety, Anhui University of Science and Technology, An'hui, 232001, China

<sup>d</sup>School of Computer Science and Electronic Engineering, University of Essex, Colchester CO4 3SQ, U.K.

## ARTICLE INFO

### Keywords:

Air Pollution Detection  
Machine Learning  
Early Warning  
Energy Conservation  
CEP  
Petri net

## ABSTRACT

Energy conservation is a fundamental requirement in achieving sustainable development, enhancing environmental protection and improving economic and social wellbeing by reducing energy consumption. Carbon emissions are a significant manifestation of energy consumption. Real-time monitoring and forecasting of air quality plays a crucial role in effectively controlling carbon emissions and improving energy conservation measures. Complex Event Processing (CEP) is a technical framework for streaming data processing based on predefined rules and is widely used for real-time detection. Yet as the diversity of data increases, it becomes exceptionally difficult for experts to formulate the CEP rules. Rule-based algorithms can address this issue to a certain extent. However, the CEP rules extracted through these methods are obtained from static data and are often simple and independent of each other. Here, we regard such rules as direct rules, which ignore possible connections between events that usually imply deeper rules. In contrast, we refer to rules that reflect interconnections between events and reveal higher-level principles as indirect rules. In this paper, we propose a methodology fusing interpretable machine learning and decision mining to extract direct and indirect multi-level CEP rules from air pollution datasets for real-time detection, prediction, and early warning. First, we utilize the decision tree algorithm to extract direct rules to detect air quality in real-time. Next, we utilize the process mining algorithm to model the changes in air states and extract the Petri net model from it that reflects the changes in air quality. Then, we utilize our proposed decision mining algorithm to extract indirect CEP rules. Indirect rules can help to understand how pollutants in the air change over time. In addition, they can reveal underlying trends and patterns of air quality change for prediction and early warning. Finally, we validate the accuracy and effectiveness of our approach using a real air pollution dataset. This work contributes to the field of air pollution detection and can help promote the optimization of energy conservation, while it can also support the development of regulatory strategies.


## 1. Introduction

Air quality detection, prediction, and early warning play a key role in managing carbon emissions and promoting energy conservation [1]. By monitoring air pollutants such as carbon dioxide and other greenhouse gases, we can identify major problem areas in energy consumption [2, 3]. This data helps decision-makers and businesses understand which industrial processes or lifestyles are the most energy-intensive, optimize energy management, and reduce waste accordingly. In addition, air quality data can raise public awareness of the importance of reducing carbon emissions and energy conservation, and motivate people to adopt more environmentally friendly behaviors [4]. Government departments may employ this data to formulate more effective environmental policies, such as improving energy efficiency

standards, promoting clean energy, and reducing overall energy consumption. Thus, air quality detection helps to improve the environment and is an essential way to achieve energy conservation and sustainable development [5, 6].

The primary cause of air pollution is the large amount of harmful gases and particulate matter produced by human activities [7, 8]. Exhaust gases released from industrial production contain large quantities of sulfur dioxide, nitrogen oxides, and volatile organic compounds, and carbon monoxide and particulate matter emitted from vehicle exhausts are also among the major sources of pollution [9, 10]. Fertilizers and pesticides used in agricultural activities also release substances that are harmful to air quality [11, 12]. Rapid urbanization and industrialization have led to an increase in these sources of pollution, which have become the main driving force behind the deterioration of air quality [13]. Thus, real-time detection and early warning of these harmful gases and particles in the air are critical to detecting carbon emissions, mitigating air pollution, and promoting energy conservation [14, 15].

\*Corresponding author.

 liu\_yuan@snnu.edu.cn (Y. Liu); ywy191@snnu.edu.cn (W. Yu);  
xzhai@essex.ac.uk (X. Zhai); mingbeizhang@snnu.edu.cn (B. Zhang);  
kdm@essex.ac.uk (K.D. McDonald-Maier); mfasli@essex.ac.uk (M. Fasli)  
ORCID(s):

Complex Event Processing (CEP) plays a critical role in real-time air detection [16]. CEP is a highly automated IoT technology that monitors and analyzes large volumes of complex, dynamic data streams in real-time according to domain-specialized predefined CEP rules in order to quickly identify and respond to specific event patterns [17]. In real-time air monitoring, a CEP system is able to instantly capture a large amount of data from various sensors and monitoring devices, including concentrations of air pollutants such as sulfur dioxide, nitrogen oxides, volatile organic compounds, and other relevant information such as meteorological conditions. By processing these data streams in real-time, a CEP system is able to quickly identify anomalies and potential pollution events [18].

Yet, there are several issues in CEP systems. For example, the formulation of the CEP rules requires detailed and in-depth domain expertise and CEP rules are still manually formulated by domain experts [19]. However, in the face of the large amount of data that is constantly being generated, it becomes extremely difficult to formulate rules manually. Manually formulated rules often have many one-sided problems and suffer from subjectivity problems while there is also the risk that human experts may miss important rules which may apply in rare or unusual situations. Therefore, there is an urgent need for a method that can automatically generate rules to solve such issues.

With the rapid development of machine learning algorithms, data-driven real-time detection based methods have been successfully applied. Currently, rule-based classifier algorithms are one of the methods to automatically generate the CEP rules [20]. Facing the large amount of data generated, machine learning classifiers can automatically generate the CEP rules, which can effectively solve the problems faced by manual rule formulation. In this paper, the CEP rules automatically generated by machine learning algorithms are represented as the direct rules.

However, those machine learning based methods have limitations. They focus only on the data itself to extract the appropriate classification rules. As a result, the machine learning algorithms tend to ignore the changes between events represented by the data and the rules implicit in the changes. There may be connections between events, and these connections usually imply deeper principles. We call rules that reflect the interconnectedness of events and reveal higher-level laws indirect rules. In any domain of application, it is invaluable to be able to extract high-level principles as indirect rules by learning from real-time data. In the air pollution domain, indirect rules hidden in the variability of the data can help us understand how pollutants change over time and they can reveal underlying trends and patterns that provide a basis for early warning, effective prevention, and control measures.

In this paper, we propose an innovative methodology combining the decision tree [21] and process mining [22] algorithms to extract direct and indirect rules of air pollution to address the challenges of formulating CEP rules and capturing higher level rules in particular from real data while

also solving the problem of real-time air quality detection and early warning. The decision tree algorithm extracts direct rules from air quality data to detect air quality in real-time. The extracted direct rules as the states of the air quality are then mapped to the raw data and the raw data is converted to event log data. Then, the inductive mining algorithm [22] mines Petri net models that can reflect changes in air quality status from event log data. Next, we use the decision tree algorithm to perform decision analysis on each decision point in the extracted Petri nets to extract high-level indirect rules in real-time. Indirect rules can help to understand how pollutants in the air change over time. Moreover, they can reveal potential trends and patterns in air quality changes. By accurately grasping the trends in air quality changes, we can make more precise predictions and implement effective early warning measures. Finally, the extracted direct and indirect rules are used as predefined rules for the CEP system to detect and monitor air quality in real-time.

The main contributions of this paper are as follows.

- A novel approach in combining interpretable machine learning and process mining algorithms to automatically extract CEP rules addresses the difficulty of manually formulating CEP rules. This approach also enables detection and early warning of air quality.
- We utilize the decision tree algorithm in a novel way to analyze decision points in Petri nets and extract high-level indirect CEP rules in real-time, which further improves the accuracy of CEP monitoring and enables early warning.
- The methodology presented extracts multi-level CEP rules from air-quality data which has not been studied before.

The rest of this paper is organized as follows: in Section 2, we introduce the research in relevant areas; in Section 3, we present the general framework of the proposed methodology and related concepts; in Section 4, we illustrate our experimental process and results; and in Section 5, we conclude this paper.

## 2. Related work

Automatic extraction of CEP rules has been the focus of previous studies and different scholars have proposed different methods to automatically extract the CEP rules.

Rule-based classification algorithms are one of the main methods for automatically extracting the CEP rules. The authors in [23, 24] used rule-based machine learning classifiers (OneR classifier, PART classifier, Ripper classification) in different domains to automatically extract the CEP rules.

Algorithms based on unsupervised learning have also been studied in the field of the CEP rule automatic extraction. The works presented in [25, 26] both used the K-means unsupervised learning algorithms to automatically extract the CEP rules.

In addition, in [27], and our previous study [19], a framework is proposed to label the data as well as to extract the rules automatically. First, deep learning algorithms (e.g., LSTM, CNN, etc.) are used to process and label the dataset, followed by the automatic extraction of rules using a rule-based classifier approach.

The experimental results of the above methods show that the CEP rules can in principle be learned by rule-based classifiers.

Automatic extraction of the CEP rules based on intelligent algorithms (Bat Algorithm, Genetic Algorithm, etc.) has also been increasing in recent years. The authors in [20] proposed an automatic CEP rule extraction method based on evolutionary algorithms. The method could extract the temporal and causal relationships between various events in streaming data. The work of [28, 29] proposed an automatic CEP rule extraction method based on the bat algorithm. The method applies the standard Bat algorithm process to a set of candidate CEP rules and derives new candidate solutions via an advanced modification operator. This automatically generates the CEP rules.

As the work in the literature demonstrates, automatic extraction of the CEP rules remains very much an open problem. However, a major disadvantage of the aforementioned methods is that they only extract the CEP rules from the perspective of the data itself, and they do not capture rules implied by data changes. To the best of our knowledge, no study has yet focused on the extraction of the high-level CEP rules. Thus, this paper focuses on a novel way of combining interpretable machine learning and process mining algorithms for extracting the multi-level CEP rules. This comprises a novel approach to CEP rule formulation that has not been presented before. The implementation details of the method are described in Section 3.

### 3. Proposed methodology

In this section, we present the details of our proposed methodology, as well as the related basics.

#### 3.1. The overall design

In this paper, we propose a novel fusion of machine learning (enhanced decision tree algorithm) and decision mining as the basis for a real-time multi-level CEP rules automatic extraction methodology. The machine learning algorithm automatically extracts direct rules from raw air quality dataset. Each direct rule represents a state of air quality. Next, the direct rules are respectively mapped to the original dataset to generate event log data. Then, the inductive mining algorithm is applied to mine the Petri nets model that can reflect the change of air quality state from the continuously generated event log data. We first search for the decision points from the real-time updated Petri net model, and then we perform decision analysis for each decision point using the decision tree algorithm to extract the higher-level indirect rules. Finally, we feed the extracted direct and indirect rules into the CEP engine for real-time

air quality detection. The overall design diagram of our proposed methodology is shown in Figure 1.

Our proposed methodology can be divided into three phases. The first phase consists of the automatic extraction of direct rules and the generation of event log data. The second phase comprises the extraction of indirect rules using decision mining. The third phase is the formulation and use of the CEP rules. Next, we present the implementation details of these.

#### 3.2. Phase 1: extraction of direct rules and generation of event log

In this phase, we use the decision tree algorithm to automatically extract direct rules. Immediately after that, we map the direct rules to the raw data, transforming it into event log data.

##### 3.2.1. Extraction of direct rules

The decision tree algorithm is one of the most popular machine learning algorithms and has been widely used in data mining, classification, and other fields. The decision tree is a supervised learning method to classify data based on inductive rules. The process of decision trees is easy to understand and has better interpretability compared to black box models such as deep learning [30, 31]. The decision tree is a tree-like structure in which each internal node represents a judgment on an attribute, each branch represents the output of a judgment result, and finally, each leaf node represents a classification result. When training a decision tree model, the information of features and labels of the training data needs to be fed into the model. Then the model is trained by minimizing the loss function, and then the optimal single-branch classification rule is calculated. Finally, the decision tree is extended from the root node down to the leaf nodes and different sub-classification rules are continuously merged in order to get the final classification result. The two most commonly used classification metrics for decision tree models are information gain and Gini index. The information gain is calculated as the information entropy of the dataset as a whole minus the conditional entropy after splitting according to a certain feature. The higher the information gain, the more a feature can determine the classes of the data. The Gini index is used to evaluate and measure the probability that a randomly selected sample in a dataset is misclassified. A smaller Gini index means that the probability of a selected sample in the dataset being misclassified is smaller.

First, the reason we use the decision tree algorithm to automatically extract the direct rules is to use its interpretability to extract meaningful direct rules similar to those formulated by experts. These rules can explain the relationship between the results of the classification and the features. In other words, different combinations of values of the feature variables can lead to different classification results. The decision tree algorithm extracts the corresponding direct rules by classifying the input data. Then these direct rules are mapped to the original dataset. If any data satisfies one of the direct

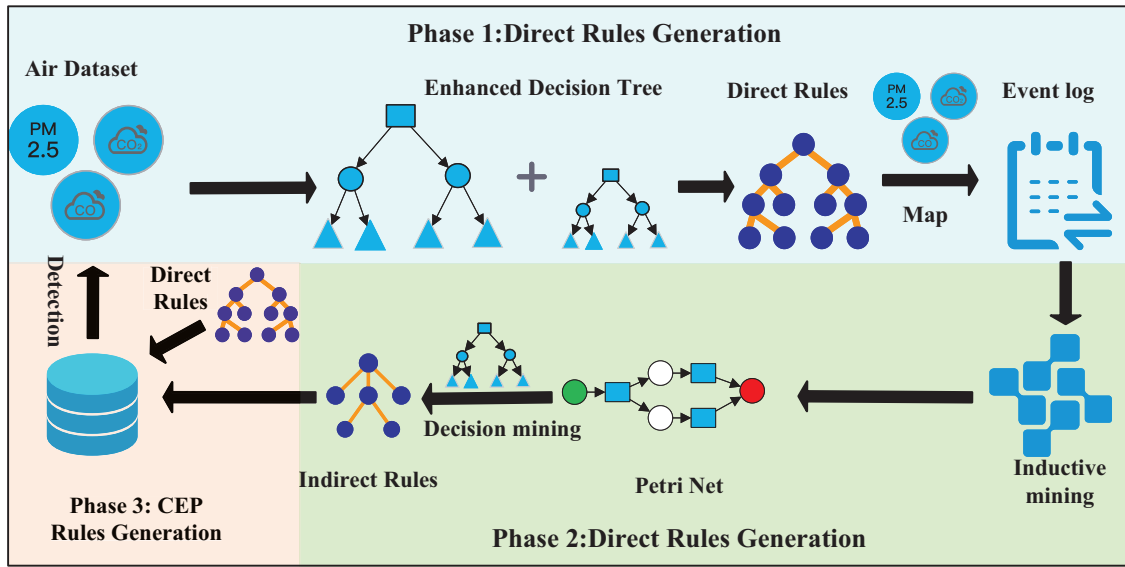


Figure 1: The overall proposed methodology.

rules, the data could be categorized into the corresponding category, such as *normal\_1*, *normal\_2*, *abnormal\_1*, etc.

To ensure that we can extract highly accurate and interpretable direct rules, we apply an enhancement of the interpretable machine learning model proposed by [32]. The approach integrated the accuracy of a strong model (e.g., weak learner integration) with the interpretability of a simple model. The primary objective of the approach is to train a powerful teacher model with low interpretability, and yet achieve high test accuracy on a particular dataset [33]. The trained model will assign weights to each dataset observation where those that are effortless to classify will be assigned higher weights, while the challenging ones will receive lower weights. Specifically, the weight of each observation in the dataset depends on the probability that a trained teacher model assigns it an accurate (true) label. The teacher model can easily obtain weights for each observation. We then employ the weighted dataset to train the student model. The student model is straightforward and easily understood, with high interpretability on the original dataset, but limited accuracy. Weighted observations allow the student model to prioritize the prediction of categories with true label observations with high weights, meanwhile ignoring outliers that may lead to lower prediction accuracy. With this approach, it is possible to ensure that the student model improves its predictive accuracy while maintaining its interpretability. The author in [32] uses deep neural networks as a teacher model to improve the performance of the decision tree. Logistic classifiers are added to the intermediate layers of a pre-trained deep neural network to initiate the process. Each classifier offers predictions based on the attached layer. The logistic classifiers' outputs and true input labels produce a confidence profile curve with confidence scores. The area under the curve (AUC) is deployed as a weight function for the original dataset observations. In this paper, we use Light Gradient Boosting

Machine (Lightgbm) [34] as a teacher model instead of a deep neural network and utilize the prediction probability of Lightgbm to assign weights to each observation. Lightgbm is an efficient gradient-boosting decision tree algorithm that is particularly good at handling large-scale data. It improves training speed and efficiency through innovative techniques such as gradient-based one-sided sampling (GOSS) and mutually exclusive feature bundling (EFB). Compared with the traditional decision trees, Lightgbm significantly improves operation speed, reduces memory usage, and supports a wide range of machine learning tasks, such as classification, regression, and ranking, which is suitable for various high-precision prediction scenarios. The schematic of the enhanced decision tree model is shown in Figure 2.

### 3.2.2. Generation of event log

In the previous step, we obtain direct rules that can categorize the raw data by training on the raw data. These rules consist of different combinations of values for each attribute of the data. In fact, each direct rule represents a state of the data (*normal\_1*, *normal\_2*, *abnormal\_1*, etc). In the case of air quality data, each direct rule represents the state of the air quality at the corresponding moment.

To generate event log data for air quality changes, we map direct rules to the original dataset. If any data satisfies one of the direct rules, the data is categorized into the appropriate category and added to the "State" column to form a new dataset. Next, in the new dataset, we assign the same "TraceID" to all the data in the same time interval, which is required to use the inductive mining algorithm. At this point, we complete the event log data construction, and the entire process is automatically performed.

### 3.3. Phase 2: extraction of indirect rules

In this phase, we first feed the generated event log data to the process mining algorithm to mine the Petri net model that



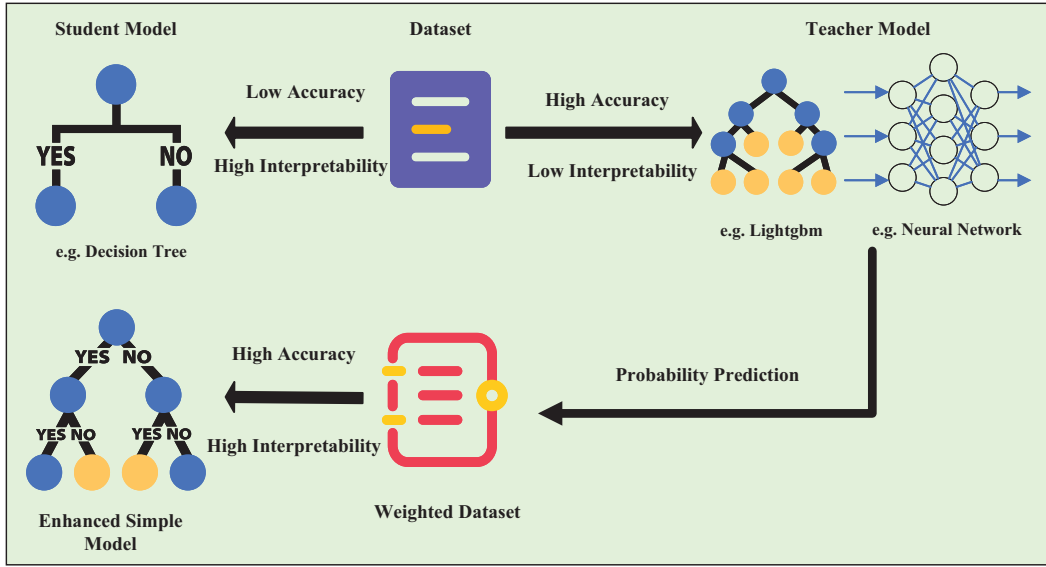


Figure 2: Training of enhanced simple models

can reflect the data state changes. Next, we perform decision analysis on each decision point in the Petri net model in order to mine the indirect rules that can reflect the connection between data states.

Here the process mining algorithm we used is the inductive mining algorithm. The inductive mining algorithm occupies an important position in the field of process mining, it is valued for its powerful data processing capabilities and wide range of application prospects. Compared to other process mining algorithms, such as alpha algorithm [35] and heuristic mining algorithm[36], etc., the unique feature of the inductive mining algorithm is its ability to handle large-scale event log data. It can efficiently handle huge amounts of event log data or interaction events at once, which makes the inductive mining algorithm exceptionally efficient when dealing with complex datasets. In addition, the inductive mining algorithm is designed with the practicality of the algorithm in mind and is able to mine process models with high quality and accuracy in a limited time. More importantly, the algorithm is able to avoid generating anomalies such as deadlocks during the mining process, ensuring that the mining process proceeds smoothly. These properties make the inductive mining algorithm an efficient and reliable tool in process mining [37]. A detailed description of the principles of the inductive mining algorithm can be found in [38].

The diagram of the process of mining the Petri net model from event log data by the inductive mining algorithm is shown in Figure 3.

A prototype Petri net model can be obtained by the inductive mining algorithm. The prototype Petri net is defined as follows.

**Definition 1 (The prototype Petri net[39]).** Let the four-tuple  $\Sigma = (P, T, F; M)$  be a prototype Petri net, where

- (1)  $P$  is a finite set of places, denoted by circles;

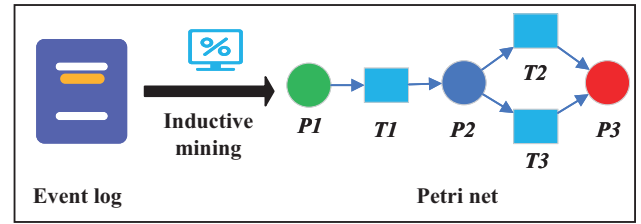


Figure 3: The process of mining Petri nets.

- (2)  $T$  is a finite set of transitions, denoted by rectangles or boxes;
- (3)  $P \cap T = \emptyset$  and  $P \cup T \neq \emptyset$ ;
- (4)  $F$  is a set of directed arcs, denoted by directed arcs, and  $F = (P \times T) \cup (T \times P)$ ;
- (5)  $M$  is a Marking of the net  $\sigma$  for a mapping:  $P \rightarrow \{0, 1, 2, 3, \dots\}$ . The initial Marking of the net  $\Sigma$  is denoted by  $M_0$ .

The predecessor and posterior sets for each place and transition in the Petri net are defined as follows. Analyzing the predecessor and posterior sets is crucial for us to extract the indirect rules next.

**Definition 2.** [39] Let  $\Sigma = (P, T, F; M)$  be a prototype Petri net,  $\forall x \in P \cup T$ , then

- (1) The predecessor set of  $x$  is:

$$\cdot x = \{y | (y \in P \cup T) \cap (y, x) \in F\}$$

- (2) The posterior set of  $x$  is:

$$x \cdot = \{y | (y \in P \cup T) \cap (x, y) \in F\}$$

After obtaining the Petri net model through the inductive mining algorithm, we need to perform decision analysis for

each decision point in the Petri net to extract indirect rules automatically. Each decision point is in fact a place. Decision points are defined as follows.

**Definition 3 (The decision point).** Let  $\Sigma = (P, T, F; M)$  be a prototype Petri net,  $p \in P$ ,  $p$  is a decision point if  $|p| > 1$ .

In the schematic we have given,  $P2$  is a decision point since  $|P2| > 1$ .

Next, we analyze each decision point using the decision tree algorithm, which provides us with the likelihood of each variation occurring at the decision point for the given feature. This is the indirect rule that we want to extract. Taking the schematic diagram we have given as an example, for the decision point  $P2$ , through the decision tree algorithm, we can get the respective probabilities of the two transitions  $T2$  and  $T3$  occurring on the decision point  $P2$ . In fact, we also get the probability that the occurrence of transition  $T1$  leads to the occurrence of transitions  $T2$  and  $T3$ .

As the event log continues to grow, the decision points may change at the same time. For example, there will be an increase in the number of decision branches, while new decision points may also appear. Thus, we need to continuously update the Petri net model based on the constant input of event log data and re-search for new decision points to achieve real-time extraction of CEP rules. Then, we propose an algorithm that can effectively solve the aforementioned issue. The specific steps of the algorithm 1 are as follows.

Algorithm 1 illustrates the basic steps for extracting indirect rules in real-time. First, each piece of data from the event stream is reformatted. The event stream data is usually in ".csv" format, and the plug-in converts it to ".xes" format that can be recognized by process mining algorithms. Next, the transformed data is added to the original event log data for re-mining by the process mining algorithms. Through this step, we achieve real-time updating of the Petri net model. Here, we regard the data with the same "TraceID" as the same batch of data. When the value of "event" of the new incoming data is "end", it means the end of this batch of data. At this point, we feed the new batch of data into the process mining algorithm at the same time as the previous data to mine the new Petri net model. Next, we need to determine whether the new generated Petri net is the same as the original Petri net. The specific comparison process is shown in Algorithm 2.

Algorithm 2 compares whether the places, transitions, and arcs of the new Petri net and the original Petri net are identical, respectively. If the places, transitions, and arcs of the new Petri net and the original Petri are the same, this means that the arrival of new data has not caused changes to the original Petri net. It also represents that the decision points of the Petri net have not changed, and it is only necessary to update the information of the decision points (stored in  $dps\_data$ ) and re-mining the rules to update the indirect rules. If there are differences between the places, transitions, and arcs of the new Petri net and the original Petri net, this means that the arrival of new data has changed

---

**Algorithm 1:** Real-time extraction of indirect rules.

---

**Input:** The event stream  
**Output:** *decision\_points*; Indirect rules

```

1 begin
2   while Data in the event stream do
3     new_event_log = pm4py.format_dataframe
      (Data)
4     if Data['Event'] == "end" then
5       newPetriNet = make_inductive_net
        (new_event_log)
6       if newPetriNet then
7         decision_points, dps_data =
          find_decision_points (newPetriNet)
8       end
9       if decision_points ≠ ∅ then
10        Indirect rules =
          mine_decision_rules (dps_data)
11      end
12    else
13      if decision_points ≠ ∅ then
14        for p in decision_point do
15          if Data['Event'] ∈ p and
            LastData['Event'] ∈ p then
16            dps_data [p].add(Data)
17          end
18        end
19      end
20    end
21  end
22 end

```

---



---

**Algorithm 2:** Comparison of old and new Petri nets.

---

**Input:** *newPetriNet*, *oldPetriNet*  
**Output:** The *newPetriNet* is *True* or *False*.

```

1 begin
2   if newPetriNet.places ≠ oldPetriNet.places then
3     return False
4   end
5   if newPetriNet.transitions ≠ oldPetriNet.
      transitions then
6     return False
7   end
8   if newPetriNet.arcs ≠ oldPetriNet.arcs then
9     return False
10  end
11  return True
12  oldPetriNet = newPetriNet
13 end

```

---

the original Petri net. At this point, we need to find the new decision points of the new Petri net and update the information of the decision points. The detailed steps for finding decision points are shown in Algorithm 3.

---

**Algorithm 3:** Finding decision points.

---

**Input:** *newPetriNet*  
**Output:** *decision\_points*

```

1 begin
2   for  $p$  in newPetriNet.places do
3     if  $|p| > 1$  then
4       decision_points.add(p)
5     end
6   end
7 end

```

---

Algorithm 3 iterates over each of the place of the new Petri net. When the number of elements in the posterior set of the iterated place is greater than 1, it means that we find a decision point. Then we add it to the set of decision points. After iterating over all the places, all the decision points in the new Petri net are obtained. Then, we perform a decision analysis on all decision points to extract the indirect rules we need. The specific steps for extracting indirect rules are shown in Algorithm 4.

---

**Algorithm 4:** Mining indirect rules.

---

**Input:** *decision\_points*, *dps\_data*  
**Output:** Indirect Rules.

```

1 begin
2   for  $p$  in the set of decision_points do
3     Indirect rules = tree.DecisionTreeClassifier
       (dps_data[p])
4   end
5 end

```

---

Algorithm 4 iterates over each decision point in turn and analyzes each decision point with a decision tree classification algorithm to extract the indirect rules.

When the value of "event" of the new incoming data is not "end", it means not the end of this batch of data. By this time, we need to continuously update the decision points information based on the new incoming data (*Data*) to prepare for the future extraction of indirect rules. Lines 13-17 of Algorithm 1 describe the process in detail. First we need to determine whether there are decision points. Then continue to determine, if the "event" of the current input event stream *Data* belongs to the posterior set of decision point  $p$ , and the "event" of the last input event stream *LastData* belongs to the posterior set of decision point  $p$ . This represents that the input event stream at this point is a branching transition of decision point  $p$ . Thus, the information (*dps\_data[p]*) of decision point  $p$  can be updated.

At this point, with the algorithms described above, we have achieved real-time extraction of indirect rules.

### 3.4. Phase 3: formulation and use of CEP rules

In the previous phases, we have automatically extracted direct and indirect rules in real-time by fusing machine learning and process mining algorithms. In this phase, we merge the direct and indirect rules extracted in the first two phases and feed them into the CEP engine at the same time. Then, in our case study, air quality data are monitored in real-time according to the merged CEP rules.

## 4. Case study and experimental results

In this section, we validate the effectiveness of our proposed method on a real air quality dataset. The air quality dataset we used is collected by the Pulse of the City EU FP7 project [40]. The Pulse of the City EU FP7 project provides not only urban air quality data, but also road traffic data, social events data and more [41]. The dataset we used in this paper is the original dataset that we labeled in our previous work [19]. This dataset is composed of four feature variables and a label. These four features record the Air Quality Index (AQI) values of particulate matter (*PM*), carbon monoxide (*CO*), sulfur dioxide (*SO<sub>2</sub>*) and ozone (*O<sub>3</sub>*) in the air. The label records whether the current air quality status is normal or not. The value of label is 1, which means the air is seriously polluted, and the value of label is 0, which means the air quality is good. This dataset contains 3494 data points, of which 894 data (25.58%) are in a normal state and 2600 data (74.42%) are in an abnormal state.

More statistical information is presented in Table 1, which includes the maximum, minimum, mean, and standard deviation (std) of the air pollution data. These statistical indicators not only help us to grasp the basic characteristics of air pollution, but also reveal the distribution and variability of the data, which provide important references for subsequent analysis of pollution trends and formulation of control strategies.

**Table 1**  
Basic information about the dataset.

	Maximum	Minimum	Mean	Std
<i>PM</i>	170	15	92.09	34.86
<i>CO</i>	213	16	119.99	40.88
<i>SO<sub>2</sub></i>	215	16	99.59	49.37
<i>O<sub>3</sub></i>	215	16	140.87	45.95

Next, we validate the methodology we proposed to automatically extract direct and indirect rules in real-time on this dataset.

### 4.1. Extraction of direct rules and generation of event log

In this phase, we adopt the enhanced decision tree (Lightgbm + decision tree) algorithm to extract direct rules from the air quality dataset.

We import the data into the Python compiler and train the data to extract the direct rules using the enhanced decision tree algorithm. The column *Label* in the data is used as the



target variable for training and all the feature variables are input variables. Before starting the training data, we divided the dataset into test set and training set. In this case, the training set data is 80% and the test set data is 20%.

First, in order to compare the accuracy with the enhanced decision tree algorithm, we use the decision tree algorithm from the **scikit-learn** package [42] on the training set. Finally, the accuracy of the test set using only the decision tree algorithm is 94.99%.

Next, we train the Lightgbm model on the training set. The Lightgbm classifier model we are using is invoked from the **lightgbm.sklearn** package. Here we optimize the Lightgbm model with hyperparameters using the GridSearch strategy to obtain more accurate direct rules.

We optimize four hyperparameters of Lightgbm with the GridSearch strategy, which are feature\_fraction, learning\_rate, max\_depth, and num\_leaves. These four hyperparameters play a crucial role in improving Lightgbm prediction accuracy. Their selection and adjustment can significantly affect the performance of the model. The explanation of these four hyperparameters and the principle of the Lightgbm adjustment parameter can be found in [34]. The search ranges for these four hyperparameters are as follows:

- feature\_fraction: [0.5, 1.5]
- learning\_rate: [0.05, 0.2]
- max\_depth: [1, 10]
- num\_leaves: [9, 15]

Then, we obtain the values of these four hyperparameters that make the Lightgbm model most accurate by using 5-fold cross-validation. The optimal parameter values are as follows: (feature\_fraction = 0.8 , learning\_rate = 0.13 , max\_depth = 3, num\_leaves = 10). After testing, Lightgbm achieved 97.71% accuracy on the test set. Next, the Lightgbm model's predictive probabilities are used to assign weights to each observation in the air quality dataset. Finally, we train the decision tree model again on the weighted air quality dataset. This avoids the problem of difficult classification of the original data and focuses the prediction on relatively easy observations, which will contribute to better generalization performance and higher test accuracy.

We use GridSearch to optimize four hyperparameters, which are: 1. criterion: used to determine the method of calculation of impurity, that is, to decide whether to use Entropy or Gini Impurity; 2. max\_depth: Limit the maximum depth of the tree and prune all branches that exceed the set depth. This is the most widely used pruning parameter, which is very effective in high dimension and low sample size. The decision tree grows one more layer and doubles the demand for sample size, so limiting the depth of the tree can effectively limit overfitting; 3. min\_samples\_split: restrict a node to contain at least min\_samples\_split training samples before the node is allowed to be branched, otherwise branching will not occur; 4. min\_samples\_leaf: limit a node to contain at least min\_samples\_leaf training samples in each

child node after branching, otherwise branching will not occur, or branching will occur in the direction of satisfying min\_samples\_leaf samples in each child node. The last three hyperparameters are the pruning strategy of the decision tree. The pruning strategy has a huge impact on the decision tree, and the correct pruning strategy is the core of the optimized decision tree algorithm. The search ranges for each of these four hyperparameters are as follows [43]:

- criterion: [entropy, gini]
- max\_depth: [2, 10]
- min\_samples\_leaf: [1, 8]
- min\_samples\_split: [1, 10]

Then, we obtain the values of these four hyperparameters that make the decision tree model most accurate by using 5-fold cross-validation. The optimal parameter values are as follows: (criterion = gini , max\_depth = 5 , min\_samples\_leaf = 3, min\_samples\_split = 6).

After the model is trained, we test the trained model on the test set and use the following metrics to measure the performance of the enhanced decision tree model: 1. Precision; 2. Recall; 3. F1-score. The formulas for these three metrics are as follows [43]:

- $precision = \frac{TP}{TP + FP}$
- $recall = \frac{TP}{TP + FN}$
- $F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} = \frac{2TP}{2TP + TP + FN}$

Where,  $TP$  represents true positives,  $FP$  represents false positives,  $TN$  represents true negatives and  $FN$  represents false negatives.

The Confusion Matrix and ROC of the enhanced decision tree model are shown in Figure 4.

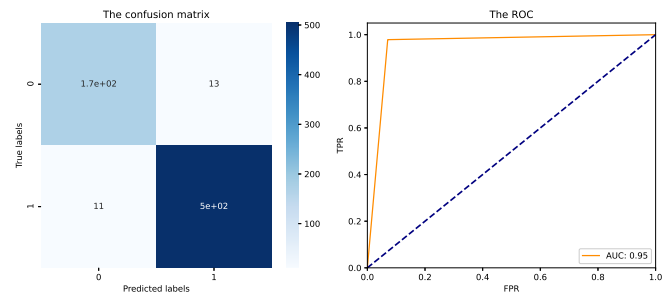


Figure 4: The Confusion Matrix and ROC.

According to Figure 4, we can see that the AUC of the decision tree model reaches 0.95 and according to the confusion matrix, we calculate the above three metrics, and the results are shown in Table 2.

Finally, after our testing, the enhanced decision tree algorithm achieved an average accuracy of 97.49%, an average

**Table 2**

Performance Metrics: Precision, Recall and F1.

	Precision	Recall	F1-score
0 (Normal)	0.94	0.93	0.93
1 (Abnormal)	0.98	0.98	0.98

Recall value of 95%, and an average F1-score of 96% on the test set. The performance of the enhanced decision tree is already close to that of the Lightgbm.

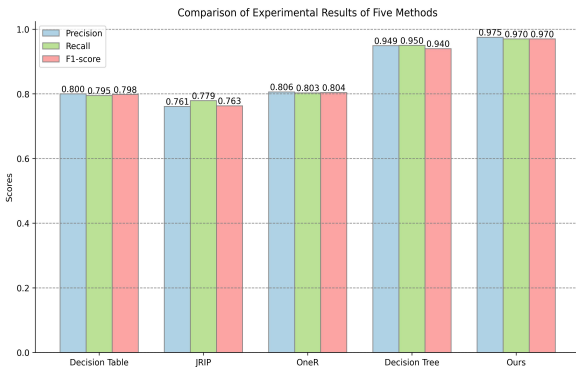
At the same time, we also performed experiment to compare with the rule-based CEP rule extraction approaches OneR, Decision Table, and JRIP [23, 24]. The results of the comparison are shown in Table 3.

**Table 3**

Comparison of experimental results of five methods.

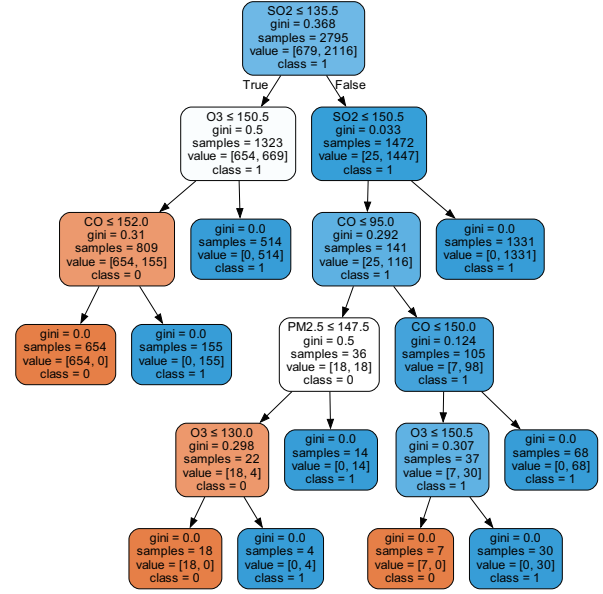
Methods	Class	Precision	Recall	F1-score
Decision Table	0	0.594	0.634	0.613
	1	0.871	0.851	0.861
	Average	0.800	0.795	0.798
JRIP	0	0.603	0.705	0.667
	1	0.815	0.909	0.860
	Average	0.761	0.779	0.763
OneR	0	0.609	0.640	0.624
	1	0.874	0.859	0.866
	Average	0.806	0.803	0.804
Decision Tree	0	0.860	0.960	0.910
	1	0.980	0.950	0.970
	Average	0.949	0.950	0.940
Ours	0	0.940	0.930	0.930
	1	0.978	0.98	0.98
	Average	0.975	0.970	0.970

In order to compare the experimental results of these algorithms more intuitively, we graphically show the average scores of these algorithms in the three evaluation metrics mentioned above, as shown in Figure 5.

**Figure 5:** Comparison of experimental results.

Comparative experimental results show that the enhanced decision tree model has the highest stability and accuracy in classifying normal and abnormal data.

The direct rules extracted using the enhanced decision tree model exhibit a tree-like structure, as shown in Figure 6.

**Figure 6:** Part of the direct rules.

With the enhanced decision tree algorithm, we extracted 10 direct rules. Each direct rule represents a state of air quality. Table 4 gives the composition of some of the direct rules and the state of air quality they represent.

**Table 4**

Sample of direct rules automatically extracted.

Direct Rules	Rules
$P1$	$SO_2 \leq 135.5$ And Normal $O_3 \leq 150.5$ And $CO \leq 152$
$P2$	$SO_2 \leq 135.5$ And Abnormal $O_3 \leq 150.5$ And $CO > 152$
$P6$	$SO_2 > 135.5$ And Abnormal $SO_2 \leq 150.5$ And $PM > 147.5$ And $CO \leq 95.0$

Next, we construct event log data from the data in the original air pollution dataset. First we map the transformed direct rules to the original air quality data, and each data will be assigned to a direct rule. Then, we add the state of each piece of data to the "State" column. We then assign the same *TraceID* to the data for the same time interval and keep the "datetime" column, all the feature columns, and the "State" column. In this way, we have constructed the event log data of air quality status.

## 4.2. Extraction of indirect rules

After constructing the event log data of air quality data, next, we feed it to our proposed algorithm for real-time extraction of indirect rules.

According to Algorithm 1, the format of the continuously fed event log data is first transformed into the data format that can be fed into the process mining algorithm. Then, the inductive mining algorithm is used to continuously mine the Petri net model that can reflect the change of air quality status from the input event log data. Here, the inductive mining algorithm we use is invoked from the **pm4py**<sup>1</sup> package.

Next, the indirect rules are updated by continuously updating the decision points and the information at each decision point based on the changing Petri net model that can be continuously updated.

As the event log data continues to be fed in, the mined Petri net model changes as follows.



Figure 7: The mined Petri net-1.

Each transition in the Petri net model obtained by the inductive mining algorithm represents an air quality state. The Petri net, illustrated in Figure 7, has not yet included the decision points. This represents that the air quality state has not changed over a period of time.

Continuing to feed the event log data in, the decision point occurs when the *TraceID* is 8. The Petri net model at this point is shown in Figure 8.

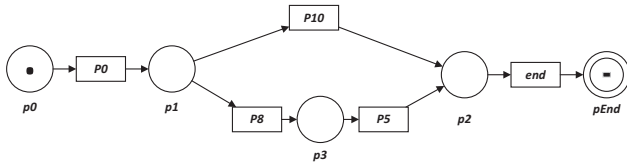


Figure 8: The mined Petri net-2.

At this point, our algorithm can find the decision point  $p1$ . Next, we continue to input the event log data to continuously mine the new Petri nets, and in this way, we can find new decision points and update the information of the new decision points. Finally, when all the event log data inputs are completed, the Petri net model we obtained is shown in Figure 9. In the final Petri net, the black transition is a silent transition that has no real meaning but ensures that the Petri net operates properly.

While continuously updating the Petri net model, we calculate its fitness and precision scores at the same time. Fitness measures the percentage of event log cases that are consistent with the model discovered, and precision

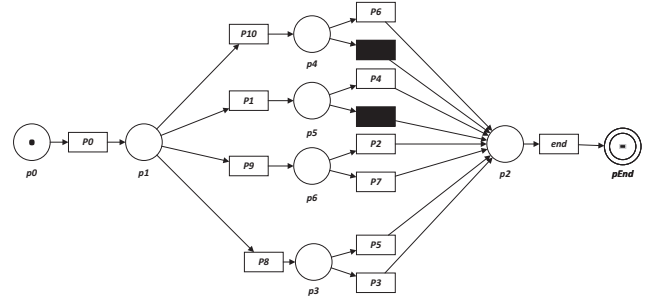


Figure 9: The mined Petri net-3.

measures how accurate the model is at not accounting for additional behavior that is not present in the event log. The algorithms for measuring fitness and precision are **fitness\_alignments()** and **precision\_token\_based\_replay()**, respectively, both from the **pm4py** package.

The changes in the scores of the mined Petri net models for these two metrics are shown in Figure 10.

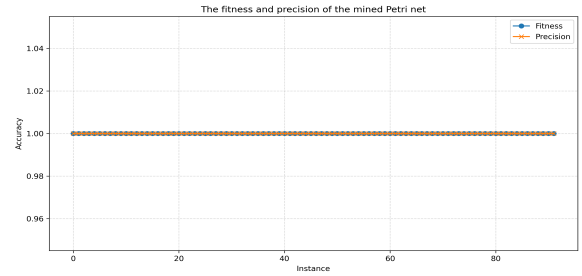


Figure 10: The fitness and precision of the mined Petri net model.

With the continuous input of event log data, the fitness and precision of the mined Petri net model are both 1, and there is no change. This indicates that our mined Petri net model is accurate and can reflect the change of air quality status effectively. This is crucial for us to find the decision points and mine out the indirect rules.

Next, we continue to use our algorithm to find decision points from the continuously updated Petri net model and calculate the prediction accuracy for each decision point.

In the end, we obtain five decision points, which are  $p1$ ,  $p3$ ,  $p4$ ,  $p5$ ,  $p6$ . Taking decision points  $p3$  and  $p6$  as an example, the change in prediction accuracy of the decision tree algorithm at these two decision points with the input of the event log is shown in Figures 11 and 12.

Finally, the decision tree algorithm has a prediction accuracy of 94.73% at decision point  $p3$  and 100% at the remaining decision points.

Next, after finding the decision points through our algorithm and continuously testing the prediction accuracy of the decision tree at each decision point, we also utilize our proposed algorithm to mine indirect rules from the decision points. Some of the indirect rules that we mined with our algorithm are as follows.

<sup>1</sup><https://pm4py.fit.fraunhofer.de/>

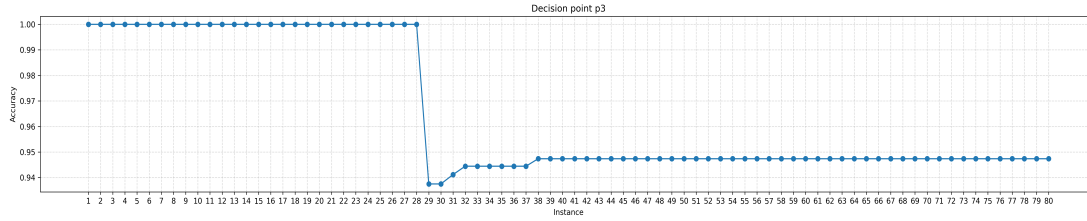


Figure 11: The decision point  $p_3$ .

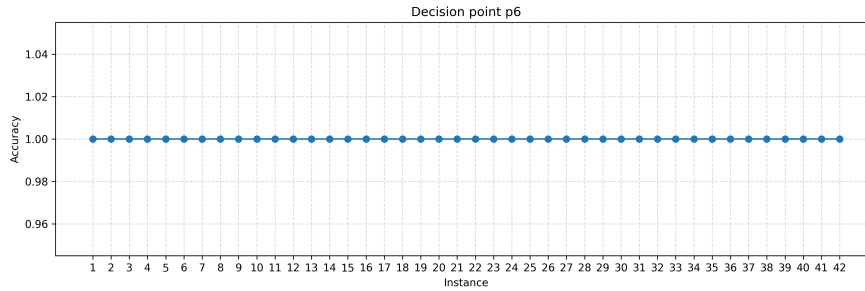


Figure 12: The decision point  $p_6$ .

When the AQI value of  $O_3$  is greater than 182.5, the air quality of "P8" will be changed to "P5". Both "P8" and "P5" air quality are in an abnormal state. According to the AQI value, the AQI value of  $O_3$  is greater than 182.5, which means that the air is severely polluted and will worsen the quality of the air. According to our algorithm, we also obtain that the predicted probability of "P8" developing into "P5" is 1 when the AQI value of  $O_3$  is greater than 182.5. The discovery of this information is crucial for early warning.

Similarly, when the AQI value of  $CO$  is greater than 152, the air quality of "P9" will change to "P2". The air quality of both "P9" and "P2" is abnormal. According to the AQI value, the AQI value of " $CO$ " is greater than 152, indicating that the air is severely polluted and the air quality will deteriorate. According to our algorithm, when the AQI value of  $CO$  is greater than 152, we also derive a predicted probability of 1 for "P9" to develop into "P2".

Thus, we organize the information extracted into indirect rules in the following format, as shown in Table 5.

**Table 5**  
Sample of indirect rules mined automatically.

Direct Rules	Rules	Probability
$P_8 \rightarrow P_5$	$O_3 > 182.5$	1.0
$P_9 \rightarrow P_2$	$CO > 152$	1.0

### 4.3. Formulation and use of CEP rules

In the previous two phases, we have extracted the direct and indirect rules of the air pollution dataset, respectively. Next, we merge the direct and indirect rules as predefined CEP rules. Then they are fed into the CEP engine for real-time air quality detection. The CEP engine we are using is

Flink CEP<sup>2</sup> which is a big data technology based on streaming relationships. Flink CEP rules are SQL-like statements that can specify the conditions of event attributes through pattern.where(), pattern.or() or pattern.until() methods.

Before utilizing direct and indirect rules to detect air quality data, we transform direct and indirect rules into CEP rules that can be recognized by the CEP engine, respectively. The transformed direct and indirect rules are shown in Table 6, Table 7 respectively.

**Table 6**  
Part of CEP direct rules.

Direct rules	Direct CEP rules
$P_1$	Pattern1 = Pattern.begin[Input]("begin") .where( $SO_2 \leq 135.5$ ) .where( $O_3 \leq 150.5$ ) .where( $CO \leq 152.0$ )
$P_2$	Pattern2 = Pattern.begin[Input]("begin") .where( $SO_2 \leq 135.5$ ) .where( $O_3 \leq 150.5$ ) .where( $CO > 152.0$ )

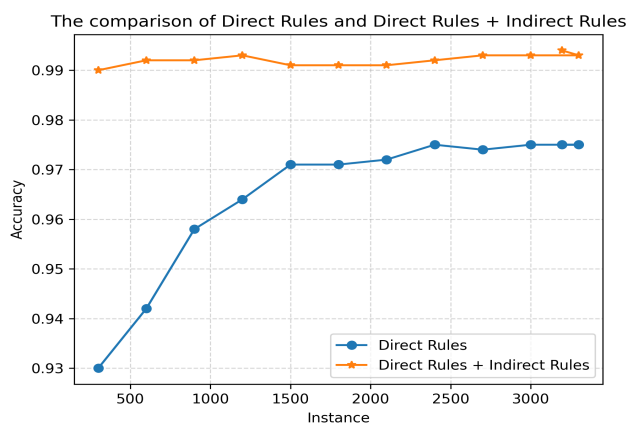
Next, we examine the air pollution dataset with the direct CEP rules as well as the merged rules that fuse the direct CEP rules and the indirect CEP rules, respectively. The comparison of the correct rate of detection and recognition between the two is shown in Figure 13.

Observing Figure 13, we can ascertain that as the data continues to be input, the identification accuracies of air quality detection using only direct rules as well as fusing direct and indirect rules are increasing for both methods. But the identification accuracy of this approach, which fuses

<sup>2</sup><https://flink.apache.org/>

**Table 7**  
Part of indirect CEP rules.

Indirect rules	CEP rules	Probability
<i>P8</i> (Abnormal) ↓ <i>P5</i> (Abnormal)	Indrect1 = Pattern. begin[Input]("begin") .where( $SO_2 \leq 150.5$ ) .where( $CO \leq 152.0$ ) .where( $PM \leq 147.5$ ) .where( $O_3 > 130$ ) .next("end") .where( $SO_2 \leq 150.5$ ) .where( $CO \leq 150.0$ ) .where( $O_3 > 150.5$ )	1.0



**Figure 13:** The comparison of Direct Rules and Direct rules + Indirect Rules

direct and indirect rules, is greater than the identification accuracy of using only direct rules. In the end, the identification accuracy using only direct rules is 97.5%, while the identification accuracy of fusing direct and indirect rules is 99.4%. The results of our experiments demonstrate that it is valuable to extract both direct and indirect rules and use them together to improve the accuracy of event detection.

Finally, in the CEP engine, the anomaly data filtered according to our extracted rules is shown in Figure 14.

```
Warning(Time: 13:03:01,particulate_matter: 131,ozone: 70,carbon_monoxide: 47.99999619,sulfure_dioxide: 156,label: 1,AlertData)
Warning(Time: 13:03:02,particulate_matter: 131,ozone: 73,carbon_monoxide: 51,sulfure_dioxide: 158,label: 1,AlertData)
Warning(Time: 13:03:03,particulate_matter: 132,ozone: 72,carbon_monoxide: 47.99999619,sulfure_dioxide: 165,label: 1,AlertData)
Warning(Time: 13:03:04,particulate_matter: 137,ozone: 72,carbon_monoxide: 53,sulfure_dioxide: 160,label: 1,AlertData)
Warning(Time: 13:03:05,particulate_matter: 137,ozone: 70,carbon_monoxide: 57,sulfure_dioxide: 159,label: 1,AlertData)
Warning(Time: 13:03:06,particulate_matter: 133,ozone: 75,carbon_monoxide: 52,sulfure_dioxide: 162,label: 1,AlertData)
```

**Figure 14:** Anomaly data filtered by the CEP engine.

## 5. Conclusions

Air quality detection is not only an important part of environmental monitoring, it is also a key and indispensable part of measures to promote sustainable development, carbon emission reduction and energy conservation. Through accurate air quality detection, we can understand the types

and concentrations of pollutants promptly, to formulate more effective pollution control strategies and emission reduction measures. This not only helps to improve air quality and protect public health, but also promotes the rational use and conservation of energy, reduces reliance on fossil fuels, and in turn reduces greenhouse gas emissions. This paper proposes a methodology that fuses machine learning and process mining to automatically extract the direct and indirect multi-level CEP rules from air quality datasets to detect air quality in real-time. The approach focuses on both the data itself and the rules implied by data changes. Our approach provides a novel idea for the automatic extraction of CEP rules. We apply our proposed approach to the field of air quality testing and validate the effectiveness of our approach on a real air quality dataset. The experimental results show that the multi-level CEP rules extracted by our approach achieve the highest detection accuracy in comparison with other CEP automatic extraction approaches. In addition, the CEP rules extracted by our approach not only help to identify pollution events but also, due to their interpretability, enable decision-makers to understand where the problems lie and where interventions are needed. It is significant in reducing carbon emissions and energy conservation.

In most cases, the data collected by sensors are usually unlabeled and their data volume is huge. Thus, in our future work, we plan to adopt a time series-based clustering approach, e.g., using advanced techniques such as transformer models, to effectively cluster these unlabeled data. With this approach, we expect to be able to extract valuable CEP rules from these large-scale datasets.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Data availability

Data will be made available on request.

## Acknowledgement

This work was supported in part by the Open Research Fund of Anhui Province Engineering Laboratory for Big Data Analysis and Early Warning Technology of Coal Mine Safety, China, under Grant No. CSBD2022-ZD05; in part by the Fundamental Research Funds for the Central Universities under Grant No. GK202205039; and in part by the Natural Science Foundation of Shaanxi Province under Grant No. 2021JM-205; in part by the Open Research Fund of Key Laboratory of Embedded System and Service Computing (Tongji University), Ministry of Education, China under Grant ESSCKF2023-02.

## References

- [1] L. Du, W. Lin, J. Du, M. Jin, M. Fan, Can vertical environmental regulation induce enterprise green innovation? a new perspective



- from automatic air quality monitoring station in china, *Journal of Environmental Management* 317 (2022) 115349.
- [2] L. Yu, X. Guo, C. Qin, L. Yang, W. Lu, R. Niu, K. Yuan, Q. Xue, Integrating synergistic control of pollutants and carbon dioxide into “three lines and one permit” in china, *Environmental Impact Assessment Review* 97 (2022) 106908.
- [3] K. Yan, X. Chen, X. Zhou, Z. Yan, J. Ma, Physical model informed fault detection and diagnosis of air handling units based on transformer generative adversarial network, *IEEE Transactions on Industrial Informatics* 19 (2) (2022) 2192–2199.
- [4] S. Zeng, A. Tanveer, X. Fu, Y. Gu, M. Irfan, Modeling the influence of critical factors on the adoption of green energy technologies, *Renewable and Sustainable Energy Reviews* 168 (2022) 112817.
- [5] S. Afshan, I. Ozturk, T. Yaqoob, Facilitating renewable energy transition, ecological innovations and stringent environmental policies to improve ecological sustainability: evidence from mm-qr method, *Renewable Energy* 196 (2022) 151–160.
- [6] J. Panneerselvam, L. Liu, N. Antonopoulos, Y. Bo, Workload analysis for the scope of user demand prediction model evaluations in cloud environments, in: 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing, IEEE, 2014, pp. 883–889.
- [7] C.-C. Lin, C.-C. Chiu, P.-Y. Lee, K.-J. Chen, C.-X. He, S.-K. Hsu, K.-C. Cheng, The adverse effects of air pollution on the eye: a review, *International Journal of Environmental Research and Public Health* 19 (3) (2022) 1186.
- [8] B. Alahmad, H. Khraishah, K. Althlaji, W. Borchert, F. Al-Mulla, P. Koutrakis, Connections between air pollution, climate change, and cardiovascular health, *Canadian Journal of Cardiology* (2023).
- [9] Y. Guo, L. Zhu, X. Wang, X. Qiu, W. Qian, L. Wang, Assessing environmental impact of nox and so2 emissions in textiles production with chemical footprint, *Science of The Total Environment* 831 (2022) 154961.
- [10] J. Zhou, X. Bai, J. Tian, Study on the impact of electric power and thermal power industry of beijing–tianjin–hebei region on industrial sulfur dioxide emissions—from the perspective of green technology innovation, *Energy Reports* 8 (2022) 837–849.
- [11] N. Boregowda, S. C. Jogigowda, G. Bhavya, C. R. Sunilkumar, N. Geetha, S. S. Udikeri, S. Chowdappa, M. Govarthanan, S. Jogaiah, Recent advances in nanoremediation: Carving sustainable solution to clean-up polluted agriculture soils, *Environmental Pollution* 297 (2022) 118728.
- [12] W. S. Darwish, L. A. Thompson, Soil, water, and air: potential contributions of inorganic and organic chemicals, in: *Present Knowledge in Food Safety*, Elsevier, 2023, pp. 26–43.
- [13] X. Zhang, L. Han, H. Wei, X. Tan, W. Zhou, W. Li, Y. Qian, Linking urbanization and air quality together: A review and a perspective on the future sustainable urban development, *Journal of Cleaner Production* 346 (2022) 130988.
- [14] V. Shakhov, A. Materukhin, O. Sokolova, I. Koo, Optimizing urban air pollution detection systems, *Sensors* 22 (13) (2022) 4767.
- [15] K. Yan, X. Zhou, J. Chen, Collaborative deep learning framework on iot data with bidirectional nlstm neural networks for energy consumption forecasting, *Journal of Parallel and Distributed Computing* 163 (2022) 248–255.
- [16] E. Brazález, H. Macià, G. Díaz, M. Baeza\_Romero, E. Valero, V. Valero, Fume: An air quality decision support system for cities based on cep technology and fuzzy logic, *Applied Soft Computing* 129 (2022) 109536.
- [17] W. Yu, Z. Ma, X. Zhai, Y. Zhou, W. Zhou, Y. Liu, Modeling and analyzing user behavior risks in online shopping processes based on data-driven and petri-net methods, *Computing and Informatics* 42 (2) (2023) 501–524.
- [18] J. Rosa Bilbao, J. Boubeta Puig, et al., Mode driven engineering for complex event processing: A survey (2022).
- [19] Y. Liu, W. Yu, C. Gao, M. Chen, An auto-extraction framework for cep rules based on the two-layer lstm attention mechanism: A case study on city air pollution forecasting, *Energies* 15 (16) (2022) 5892.
- [20] J. Lv, B. Yu, H. Sun, Cep rule extraction framework based on evolutionary algorithm, in: 2022 11th International Conference of Information and Communication Technology (ICTech), IEEE, 2022, pp. 245–249.
- [21] A. J. Myles, R. N. Feudale, Y. Liu, N. A. Woody, S. D. Brown, An introduction to decision tree modeling, *Journal of Chemometrics: A Journal of the Chemometrics Society* 18 (6) (2004) 275–285.
- [22] W. Van Der Aalst, Process mining: Overview and opportunities, *ACM Transactions on Management Information Systems (TMIS)* 3 (2) (2012) 1–17.
- [23] N. Mehdiyev, J. Krumeich, D. Enke, D. Werth, P. Loos, Determination of rule patterns in complex event processing using machine learning techniques, *Procedia Computer Science* 61 (2015) 395–401.
- [24] M. M. Naseri, S. Tabibian, E. Homayounvala, Intelligent rule extraction in complex event processing platform for health monitoring systems, in: 2021 11th International Conference on Computer Engineering and Knowledge (ICCKE), IEEE, 2021, pp. 163–168.
- [25] E. Petersen, M. A. To, S. Maag, T. Yamga, An unsupervised rule generation approach for online complex event processing, in: 2018 IEEE 17th International Symposium on Network Computing and Applications (NCA), IEEE, 2018, pp. 1–8.
- [26] M. U. Şimşek, S. Özdemir, Cep rule extraction from unlabeled data in iot, in: 2018 3rd International Conference on Computer Science and Engineering (UBMK), IEEE, 2018, pp. 429–433.
- [27] M. U. Simsek, F. Yildirim Okay, S. Ozdemir, A deep learning-based cep rule extraction framework for iot data, *The Journal of Supercomputing* 77 (2021) 8563–8592.
- [28] R. Bruns, J. Dunkel, N. Offel, Learning of complex event processing rules with genetic programming, *Expert Systems with Applications* 129 (2019) 186–199.
- [29] R. Bruns, J. Dunkel, Bat4cep: a bat algorithm for mining of complex event processing rules, *Applied intelligence* 52 (13) (2022) 15143–15163.
- [30] S. B. Kotsiantis, Decision trees: a recent overview, *Artificial Intelligence Review* 39 (2013) 261–283.
- [31] K. Yan, X. Zhou, Chiller faults detection and diagnosis with sensor network and adaptive 1d cnn, *Digital Communications and Networks* 8 (4) (2022) 531–539.
- [32] A. Dhurandhar, K. Shanmugam, R. Luss, P. A. Olsen, Improving simple models with confidence profiles, *Advances in Neural Information Processing Systems* 31 (2018).
- [33] M. Lee, J. Jeon, H. Lee, Explainable ai for domain experts: a post hoc analysis of deep learning for defect classification of tft–lcd panels, *Journal of Intelligent Manufacturing* (2021) 1–13.
- [34] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, T.-Y. Liu, Lightgbm: A highly efficient gradient boosting decision tree, *Advances in neural information processing systems* 30 (2017).
- [35] W. Van der Aalst, T. Weijters, L. Maruster, Workflow mining: Discovering process models from event logs, *IEEE transactions on knowledge and data engineering* 16 (9) (2004) 1128–1142.
- [36] A. Weijters, J. T. S. Ribeiro, Flexible heuristics miner (fhm), in: 2011 IEEE symposium on computational intelligence and data mining (CIDM), IEEE, 2011, pp. 310–317.
- [37] S. J. Leemans, D. Fahland, W. M. Van Der Aalst, Discovering block-structured process models from event logs—a constructive approach, in: *Application and Theory of Petri Nets and Concurrency: 34th International Conference, PETRI NETS 2013, Milan, Italy, June 24–28, 2013. Proceedings* 34, Springer, 2013, pp. 311–329.
- [38] J. Rudnickaia, Process mining. data science in action, University of Technology, Faculty of Information Technology (2016) 1–11.
- [39] W. Zhehui, Introduction to petri nets, Beijing, Press of Machinery and Industry (2006) 15–21.
- [40] T. Consortium, et al., Citypulse annual report, The CityPulse Consortium (2016).
- [41] M. I. Ali, F. Gao, A. Mileo, Citybench: A configurable benchmark to evaluate rsp engines using smart city datasets, in: *International semantic web conference*, Springer, 2015, pp. 374–389.

- [42] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al., Scikit-learn: Machine learning in python, the Journal of machine Learning research 12 (2011) 2825–2830.
- [43] I. H. Witten, E. Frank, A. Mark, Hall, and christopher j pal. data mining: Practical machine learning tools and techniques (2016).