Human-Interpretable Rules for Anomaly Detection in Time-series

Inès Ben Kraiem, Faiza Ghozzi, André Péninou, Geoffrey Roman-Jimenez, Olivier Teste

Human-Interpretable Rules for Anomaly Detection in Time-series

Ines Ben Kraiem
University of Toulouse - UT2J, IRIT
Toulouse, France
ines.ben-kraiem@irit.fr

Faiza Ghozzi
University of Sfax- ISIMS, MIRACL
Sfax, Tunisie
faiza.ghozzi@isims.usf.tn

Andre Peninou
University of Toulouse- UT2J, IRIT
Toulouse, France
andre.peninou@irit.fr

Geoffrey Roman-Jimenez
CNRS, IRIT
Toulouse, France
geoffrey.roman-jimenez@irit.fr

Olivier Teste
University of Toulouse- UT2J, IRIT
Toulouse, France
olivier.teste@irit.fr

ABSTRACT

Human-interpretable rules for anomaly detection refer to abnormal data presented in a format that cannot be intelligible to analysts. Learning these rules is a challenging issue, while only a few works address the problem of different types of anomalies in time-series. This paper presents an extended decision tree based on patterns to generate a minimized set of human-comprehensible rules for anomaly detection in univariate time-series. This method uses Bayesian optimization to avoid manual tuning of hyper-parameters. We define a quality measure to evaluate both the accuracy and the intelligibility of the produced rules. The performance of the proposed method is experimentally assessed on real-world and benchmark datasets. The results show that our approach generates rules that outperforms state-of-the-art anomaly detection techniques.

1 INTRODUCTION

Anomaly detection in time series is a widely studied issue in many areas such as financial markets, sensor networks, habitat monitoring, network intrusion, web traffic [1], and many others. Time series are often affected by unusual events or untimely changes (e.g., measurement error or faulty sensors) that need to be detected and processed by users for analysis and exploration [7].

In a real context, experts may observe some interesting local phenomena, which can be seen as remarkable points in time series. Using their domain knowledge, the experts investigate sequences of remarkable points to detect and locate anomalies. Experts can also build decision rules manually to detect future occurrences of these anomalies. However, as the amount of collected data is increasing, the decision rules become more complex to define which makes the analysis more difficult. Automatic rule extraction and detection of different types of anomalies can be of considerable interest to an expert, leading to appropriate action that can save a lot of time and value. To overcome this challenge, rule learning algorithms have been proposed [2, 12]. Deploying such systems might reveal comprehensible information to the users to explain the root cause of anomalies better than black-box algorithms.

To address these challenges, we provide a machine learning method to generate human-interpretable rules for anomaly detection in time-series, called Composition-based Decision Tree (CDT) [1]. This method uses patterns to identify remarkable points. The compositions of remarkable points existing into time-series are learned through a specific decision tree that finally produces intelligible rules. To avoid manual tuning, we use Bayesian hyper-parameter optimization to get the best hyper-parameters for our model. The approach aims at finding the best compromise between a high accuracy during anomaly detection and a minimized set of easily human-interpretable rules. We conduct experiments on three real-world datasets and three synthetic datasets. The results show the effectiveness of our method compared to both pattern-based methods and rule-based methods for anomaly detection.

2 RELATED WORK

Numerous works on anomaly detection in time-series had been covered under various surveys and reviews [3, 13]. Several fields of study are related to pattern-based time-series data mining and rule-learning methods for anomaly detection. Pattern-based methods aim to discover frequent [4, 5] or infrequent sub-sequences [16] from a time-series. In contrast to our method, these methods are less suitable for detecting multiple anomalies and can only find a specific type of anomaly. Rule-learning methods aim to find regularities in data that can be expressed in the form of IF-THEN-like rules [2, 9, 11]. In general, the rules are evaluated based on accuracy or the number of rules produced by these algorithms. In this paper, instead of only evaluating the rules based on these criteria, we introduce a quality measure, which takes into account the number of used patterns as well as the length of rules, to make the rules simpler to interpret. We also propose a function that seeks a compromise between the interpretability of the rules and their precision.

3 METHODOLOGY

In this section, we describe our Composition-based Decision Tree (CDT) method for anomaly detection and rule extraction.

3.1 Time-Series Preprocessing

Definition 1. An univariate *time-series* is defined as $Ts = \{x_1, ..., x_n\}$ where $\forall i \in [1..n], x_i \in \mathbb{R}$ such that values x_i are uniformly spaced in time and n is the size of Ts.

The time series are collected from different sensors and the values of measures are on different ranges. To achieve scale and offset invariance, we normalize each continuous time-series Ts to values within the range $[0, 1]$. Resampling could also be used to provide additional structure or to smooth time series and remove

[1] https://github.com/IBK-TLS/CDT

any noise; e.g., downsampling reduces the frequency of time-series observations.

3.2 Time-Series Labeling

To detect anomalies, experts first analyze the neighborhood of a point (unusual variations) such as the point which precedes it and follows it to decide if is a remarkable point. Based on this idea, we label each point of the time series by checking every three successive points using patterns.

Let us considers three successive points such as x_{i-1}, x_i, x_{i+1} of a time-series. Considering all possible variations between these three points, we define nine possible variations, which can occur between successive points, as listed in Table 1 namely, PP (Positive Peak), PN (Negative Peak), SCP (Start Constant Positive), SCN (Start Constant Negative), ECP (End Constant Positive), ECN (End Constant Negative), CST (Constant), VP (Variation Positive) and VN (Variation Negative). Each of these variations can have different magnitudes into [-1,1]. To identify fine variations of values between three points, we refine each variation by defining intervals of variations into [-1,1].

Hyper-parameter (δ). We denote δ the hyper-parameter used to distinguish the different magnitudes considered for each of the nine variations (PP, PN, SCP, SCN, ECP ECN, CP, VN, and CST). δ allows the introduction of fine amplitude shifts in the variations to capture the shape complexity in time-series, typically small or large amplitudes. δ represents the number of disjointed sub-intervals considered in [-1,1] and will be determined automatically using Bayesian optimization. For a given δ, we construct $2\delta+1$ intervals : i) δ sub-intervals for positive variations in $]0,1]$, ii) δ intervals for negative variations in $[-1,0[$, and iii) 1 special case for the absence of variation (equal to 0).

For the sake of simplicity, in the rest of the paper, we only consider notation with $\delta = 2$ (other values of δ will only result in a larger variety of intervals and patterns), and we denote the 5 resulting intervals as follows: Low (L) =]0,0.5], High (H) =]0.5,1], -Low (-L) = [-0.5,0[, -High (-H) = [-1,-0.5[, and the special case, Zero (Z) = 0.

Definition 2. We define a *pattern* annotated $P = (l, \alpha, \beta)$ where l is a name (or label) identifying the pattern, and α and β are two possible intervals from [-1,1]. For each successive points x_{i-1}, x_i, x_{i+1}, the point x_i is checked by a pattern only if $x_i - x_{i-1} \in \alpha \wedge x_i - x_{i+1} \in \beta$. In this case, x_i is labeled with l. For the rest of the paper, labels are denoted by the name of the variation (PP, PN, etc.).

Fig. 1 (left) illustrates an example of a pattern $PP_{L,H}$ that helps us to find one remarkable point from a time-series. This time-series represents the consumption data of a building's calorie sensor. Fig. 1 (right) shows examples of different magnitudes of pattern such as $PP_{L,H}$, $PP_{L,L}$ and $PP_{H,H}$. Using these patterns, we can automatically label each point in time series.

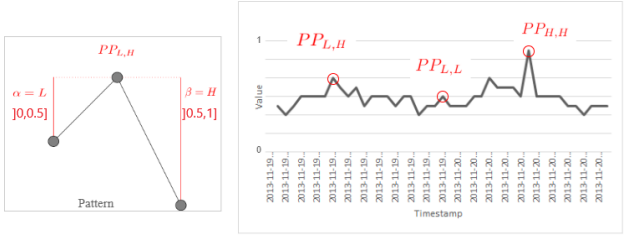

Figure 1: Example of different magnitudes of a pattern.

Definition 3. A *labeled times-series* annotated $Tsb = \{l_1, ..., l_N\}$ with $N = n - 2$ where each x_i point of the initial time-series (Ts) is replaced by the label of its corresponding pattern.

Table 1: Types of variation for labelling.

Variation	Definition	Pattern	Example
PP	$x_{i-1} < x_i \wedge x_i > x_{i+1}$		
	$\alpha, \beta \in \{L, H\}$	$PP_{\alpha,\beta}$	$PP_{L,H}$
PN	$x_{i-1} > x_i \wedge x_i < x_{i+1}$		
	$\alpha, \beta \in \{-L, -H\}$	$PN_{\alpha,\beta}$	$PN_{-H,-H}$
SCN	$x_{i-1} > x_i \wedge x_i = x_{i+1}$		
	$\beta \in \{Z\}$ $\alpha \in \{-L, -H\}$	$SCN_{\alpha,\beta}$	$SCN_{-L,0}$
SCP	$x_{i-1} < x_i \wedge x_i = x_{i+1}$		
	$\beta \in \{Z\}$ $\alpha \in \{L, H\}$	$SCP_{\alpha,\beta}$	$SCP_{H,0}$
ECN	$x_{i-1} = x_i \wedge x_i > x_{i+1}$		
	$\alpha \in \{Z\}$ $\beta \in \{L, H\}$	$ECN_{\alpha,\beta}$	$ECN_{0,L}$
ECP	$x_{i-1} = x_i \wedge x_i < x_{i+1}$		
	$\alpha \in \{Z\}$ $\beta \in \{-L, -H\}$	$ECP_{\alpha,\beta}$	$ECP_{0,-L}$
CST	$x_{i-1} = x_i \wedge x_i = x_{i+1}$		
	$\alpha, \beta \in \{Z\}$	$CST_{\alpha,\beta}$	$CST_{0,0}$
VP	$x_{i-1} < x_i \wedge x_i < x_{i+1}$		
	$\alpha \in \{L, H\}$ $\beta \in \{-L, -H\}$	$VP_{\alpha,\beta}$	$VP_{L,-L}$
VN	$x_{i-1} > x_i \wedge x_i > x_{i+1}$		
	$\alpha \in \{-L, -H\}$ $\beta \in \{L, H\}$	$VN_{\alpha,\beta}$	$VN_{-L,L}$

3.3 Composition-based Decision Tree

Given the time series labeling, we built an extension of the decision tree based on pattern compositions to produce rules able to detect anomalies.

Classically, a decision tree is induced from observations composed of feature values and a class label. It is built by splitting the training data into subsets by choosing the feature which best partitions the training data according to an evaluation criterion (e.g., Shannon's entropy, Gini index). This criterion characterizes the homogeneity of the subsets obtained by division of the data set. This process is recursively repeated on each derived subset until all instances in a subset belong to the same class label [10].

The classical decision tree considers features without any order when splitting the datasets. Conversely, in our approach, we would like to keep the order of the time series. Thus, our decision tree is built, by considering nodes as pattern compositions (ordered sequences of remarkable points) with the highest information gain. These compositions are calculated from a set of observations (sub-sequences of labeled time-series). The input of the tree is constructed by creating fixed sized sliding windows.

Definition 4. A *set of observations* annotated $D = \{d_1, d_2, ..., d_{N-\omega+1}\} = \{\{l_1, ..., l_\omega\}, \{l_2, ..., l_{\omega+1}\}, ..., \{l_{N-\omega+1}, ..., l_N\}\}$ represents the result of cutting Tsb by a sliding window of size ω, and using a fixed step size equal to one. Let M be the number of classes of observations. In our context, we consider two classes (M = 2): the abnormal class (observation with anomaly), or the normal class (observation without anomaly). Each d_i observation is associated with only one class annotated $class(d_i)$.

To determine a probability distribution of the observations over the classes, we introduce an impurity measure of a set of observations $D_j \subseteq D$. In our method, we opt to the Gini index. The *Gini impurity index*, annotated $G(D_j)$, provides a measure of the quality of D_j according to the distribution of the observations into the classes. The impurity metric is minimal (equal to 0) if a set contains only observations of one class, and it is maximal (equal to 0.5) if the set contains equally observations of all classes.

Hyper-parameter (ω). We denote $\omega <= N/2$ the window size to define observations. This hyper-parameter will be determined automatically using Bayesian optimization.

From an *observation with anomaly* we can define a composition used to split a node into two sub-nodes.

Definition 5. A *composition* annotated c is a sub-sequence of labels of an observation d_i. We denote $c \subseteq_o d_i$.

Example. Considering $d = \{l_1, l_2, l_3, l_4, l_5, l_6\}$, some compositions are $c = \{l_2, l_3, l_4\} \subseteq_o d$, $c = \{l_3, l_2, l_4\} \not\subseteq_o d$, and $c = \{l_1, l_2, l_3, l_4, l_5, l_6\} \subseteq_o d$.
We also introduce additional notations: $c \in_o D$ when $\forall d \in D, c \subseteq_o d$, and $c \notin_o D$ when $\forall d \in D, c \not\subseteq_o d$.

A decision tree is built based on features that have the highest Information Gain [10]. In CDT, the compositions are compared according to the information gain, noted IG, they provide.

The entire flow of the CDT approach we proposed is described by Algorithm 1. This algorithm builds a decision tree; we define a tree node as a quadruplet: *observations* (the set of observations considered in this node), a *composition* (used to split *observations* in two child nodes), *childTrue* (the node of observations satisfying the *composition*), and *childFalse* (the node of observations that do not satisfy the *composition*). An example corresponding to the root node is given in line 1. We introduce a function $list\_of\_all\_possible\_compositions()$ to compute all compositions deduced from a set D_j (line 6). For each composition, we calculate the information gain to split a node (line 7–15). At line 16, if $G(D_j) \neq 0$ means that the set of observations of the node is impure (observations are of different classes). Moreover, $maxGain \neq 0$ means that a composition that splits the set of observations has previously been found: in this case, we create a node N_{inc} that will determine the positive branch of the node ($c \in_o D_j$), whereas N_{exc} will be the negative one ($c \notin_o D_j$) (line 16–25). We repeat these steps until there are no more nodes to process (line 3–26).

Example. Fig. 2 illustrates an example of CDT result. The root-node is D_1, and it represents the whole training set of observations used for the construction of the CDT. The leave-nodes represent class labels and branches represent conjunctions of compositions that lead to those class labels. As shown in Fig. 2, the CDT is composed of 3 splits constructing a set of 3 leaves $\mathcal{S} = \{S_1, S_2, S_3\}$.

3.4 Rule Generation for Anomaly Detection

We convert the CDT into a set of decision rules. We only consider "pure leaf-nodes" leading to the anomaly class.

Definition 6. A *rule predicate*, annotated Rs is a branch of the decision tree leading to the anomaly class. It is constructed by combining (conjunction) the successive compositions c_i or $\neg c_i$ from the leaf-node to the root-node. For each positive branch ($c_i \in_o D_j$), the positive composition c_i is deduced whereas a negative composition $\neg c_i$ is deduced from a negative branch ($c_i \notin_o D_j$).

Algorithm 1 CDT: Composition-based Decision Tree

Input: $D = \{d_1, d_2, ..., d_{N-\omega+1}\}$ a set of observations
Output: N_{root} the root node of CDT

1: $N_{root} \leftarrow Node(D, null, null, null)$
2: $q \leftarrow [N_{root}]$ *// construct the queue of nodes to split*
3: **while** $q \neq \emptyset$ **do**
4: $N_j \leftarrow q.pop()$ *// dequeue the first node from the queue*
5: $D_j \leftarrow N_j.observations$
6: $C_j \leftarrow list\_of\_all\_possible\_compositions(D_j)$
7: $maxGain \leftarrow 0$
8: $c_{best} \leftarrow null$
9: *// Choose the composition that has the best Gain*
10: **for all** $c \in C_j$ **do**
11: **if** $IG(D_j, c) > maxGain$ **then**
12: $maxGain \leftarrow IG(D_j, c)$
13: $c_{best} \leftarrow c$
14: **end if**
15: **end for**
16: **if** $G(D_j) \neq 0$ and $maxGain \neq 0$ **then**
17: $D_{inc} \leftarrow \{d \in D_j | c_{best} \in_o d\}$
18: $D_{exc} \leftarrow \{d \in D_j | c_{best} \notin_o d\}$
19: $N_{inc} \leftarrow Node(D_{inc}, null, null, null)$
20: $N_{exc} \leftarrow Node(D_{exc}, null, null, null)$
21: $q.append(N_{inc})$ *// enqueue child nodes*
22: $q.append(N_{exc})$
23: $N_j.composition \leftarrow c_{best}$
24: $N_j.childTrue \leftarrow N_{inc}$
25: $N_j.childFalse \leftarrow N_{exc}$
26: **end if**
27: **end while**
28: **return** N_{root}

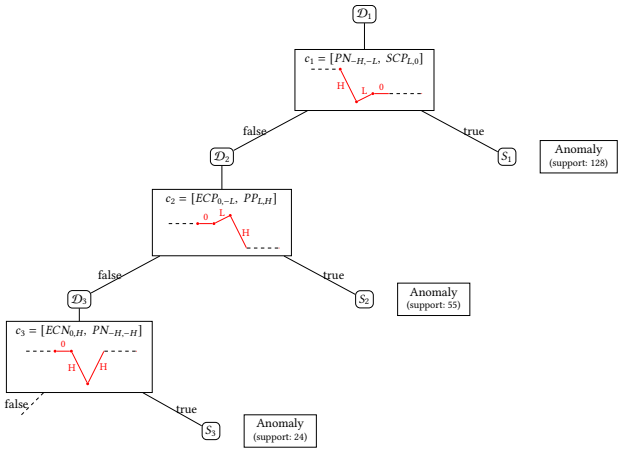

Figure 2: Illustration of a Composition-based Decision Tree (CDT).

Example. Three rules predicate are produced from the CDT in Fig. 2.

- $R_{S_1} : c_1 = [PN_{-H,-L},\ SCP_{L,0}]$
- $R_{S_2} : c_2 \wedge \neg c_1 = [ECP_{0,-L},\ PP_{L,H}] \wedge \neg[PN_{-H,-L},\ SCP_{L,0}]$
- $R_{S_3} : c_3 \wedge \neg c_2 \wedge \neg c_1 = [ECN_{0,H},\ PN_{-H,-H}] \wedge \neg[ECP_{0,-L},\ PP_{L,H}] \wedge \neg[PN_{-H,-L},\ SCP_{L,0}]$.

Using abusive notations, $c_i \wedge \neg c_j$ means that for an observation d on a time-series, we check $c_i \subseteq_o d \wedge c_j \not\subseteq_o d$.

Definition 7. A rule, annotated $\mathcal{R}$, is a disjunction of rule predicates. For instance, as shown in Fig. 2, $\mathcal{R} = R_{S_1} \vee R_{S_2} \vee R_{S_3} = (c_1) \vee (c_2 \wedge \neg c_1) \vee (c_3 \wedge \neg c_2 \wedge \neg c_1)$.

Rule Simplifications. One way to minimize CDT's rules is to post-process the produced rules through Boolean algebra simplifications. We aim to minimize the "sum-of-products" forms of Boolean functions. In our approach, this inter-branch simplification is applied until there is no longer any simplification to be made. This allows us to minimize the number of compositions in a rule. Using Boolean algebra, we can simplify the rule $\mathcal{R}$ generated from the tree in Fig. 2 as $\mathcal{R} = (c_1) \vee (c_2 \wedge \neg c_1) \vee (c_3 \wedge \neg c_2 \wedge \neg c_1) = (c_1) \vee (c_2) \vee (c_3)$.

3.5 Quality Measure

We aim to generate rules that are both accurate and comprehensible. According to experts opinion, a comprehensible rule should be short [2, 9] and should contain a minimized number of various labels. Therefore, we defined the following criteria to evaluate a quality of rules:

- $\mathcal{I}(c)$ to characterize the *quality of a composition c* depending on its length and the number of patterns used;
- $\mathcal{M}(\mathcal{I}_{R_S})$ to characterize the *quality of a rule predicate* R_S depending on the number of compositions and the quality of each one ($\mathcal{I}(c)$);
- $\mathcal{Q}(\mathcal{R})$ to characterize the *quality of a rule* $\mathcal{R}$ depending on the quality of rule predicate and its support.

We first calculate the interpretability of a composition as:

$$\mathcal{I}(c) = 1 - \frac{L_c.N_L}{\omega.MaxL} \tag{1}$$

where $L_c = |c|$ denotes the length of a composition c, N_L is the number of unique labels used in a composition, ω is the maximum window size, and $MaxL$ is the total number of labels. Then, we calculate the average interpretability of a rule predicate (conjunction of compositions) as:

$$\mathcal{M}(\mathcal{I}_{R_S}) = \frac{1}{Nc} \sum_{k=1}^{Nc} \mathcal{I}(c_k) \tag{2}$$

where Nc is the number of compositions in a rule predicate R_S. Finally, extracted rules' quality is calculated as:

$$\mathcal{Q}(\mathcal{R}) = \frac{1}{S} \sum_{i=1}^{nb} S_{R_{S_i}} . \mathcal{M}(\mathcal{I}_{R_{S_i}}) \tag{3}$$

where nb is the number of rule predicates in $\mathcal{R}$, $S_{R_{S_i}}$ is the support of rule predicate (true positive) and S is the support of all rules predicates (true positive and true negative). A rule predicate with high support is considered more important. Therefore, we multiplicate the average interpretability of a rule predicate with its support.

3.6 Hyper-Parameters selection

The manual tuning of hyper-parameters requires a prior knowledge. Automatic search algorithms such as grid search and random search could give good results. However, grid search is time consuming and random search might not find the optimal set. To address this problem, we use Bayesian Optimization [14] to efficiently get the best hyper-parameters (δ, ω) for our model. Indeed, we aim to find a configuration (that is, a set of parameters) that maximizes a performance metric or an objective function. Hence, we defined a search space and we try to find the hyper-parameters values of CDT that yield the highest result as measured on a validation set. Hyper-parameter optimization is presented as:

$$h^* = \arg\max_{h \in H} F(h) \tag{4}$$

where $F(h)$ represents an objective function to maximize, $h*$ is the set of optimized hyper-parameters (δ, ω) and h can take any value in search space H.

To optimize the trade-off between detection performance and good quality of rules, we defined the objective function $F(h)$ as the F-measure weighted by our $\mathcal{Q}(\mathcal{R})$ rules' quality measure.

$$F(h) = F1(h).\mathcal{Q}(\mathcal{R}) \tag{5}$$

where $F1(h)$ is the F-measure (the harmonic mean of the precision and recall) of the classification performance obtained with the set of parameters (h).

4 EXPERIMENTS

For our evaluation, we use real-world datasets from the Management and Exploitation Service (SGE) [6], and data from the Yahoo datasets [8]. We compare CDT with state-of-the-art pattern-based as well as rule-based methods for anomaly detection.

In SGE data sets, we aim to handle anomalies on calorie and electric consumption datasets. Calorie data consists of 25 consumption datasets from sensors deployed in different buildings managed by the SGE. These measurements are daily data for more than three years, about 33536 observations in total and they contain 586 anomalies. Electricity measurements are collected hourly for 10 years from one sensor (96074 in total). There are in total 10343 anomalies in the electricity dataset.

The Webscope S5 dataset, which is publicly available in [8], consists of 371 files divided into four categories, named A1/A2/A3/A4, each one containing respectively, 67/100/100/100 files. A1 Benchmark is based on real production traffic from actual web services while classes A2, A3, and A4 contain synthetic anomaly data. These datasets are represented by time-series in one-hour units. There are a total of 94778 traffic values in 67 different files and 1669 of these values are abnormal. The anomalies in the synthetic datasets are inserted at random positions. A2 Benchmark contains 142002 values with 466 anomalies while 168000 values exist in A3 and A4 Benchmarks with respectively 943 and 837 anomalies.

4.1 Evaluation Process and Metrics

The performance of all the methods is compared based upon the F1 score, the rules'quality metric $\mathcal{Q}$, and the objective function $F(h)$ used as defined in equation (5). We use the $F1$ score and $\mathcal{Q}$ score to compare the accuracy of our method against pattern-based methods. We use the $F(h)$ score to evaluate both the accuracy and the interpretability of the rules generated by our CDT method compared to those of rule learning methods. For evaluation, we split every dataset into three subsets: training set (60%), validation set (20%), and testing set (20%) ratio. We use the train and validation set to optimize the model's hyper-parameter values using Bayesian optimization. Then, we evaluate the optimized model on testing set.

Hyper-Parameters Optimization. To limit the search space of the Bayesian optimization, we constrained the parameter ω within [3,31] and δ within [1,21]. Table 2 shows the optimal hyper-parameters found with the Bayesian optimization. As we can see in Table 2, optimization on $F(h)$ tends to favors a small number of splits (δ) for patterns when compared to the $F1$ score optimization. This is due to the quality measure of rules $\mathcal{Q}(\mathcal{R})$,

which looks for short rules that include a minimum number of labels (δ). However, the size of observations (ω) needed to construct an optimal CDT remains to be comparable for $F1$ and $F(h)$ suggesting the need for presence of neighbors surrounding an anomaly to achieve good anomaly detection with CDT.

Table 2: Parameters of CDT for experiments.

Evaluation	F1-score		F(h)-score	
Dataset	ω	δ	ω	δ
SGE_Electricity	27	2	27	2
SGE_Calorie	5	4	21	1
Yahoo_A1	27	16	25	1
Yahoo_A2	17	2	17	1
Yahoo_A3	29	12	17	1
Yahoo_A4	25	8	21	1

4.2 Experiments with Pattern-based Algorithms

We employ the following three approaches as baseline methods to compare with our approach:

- Pattern-Based Anomaly Detection (PBAD) is an anomaly detection method based on frequent pattern mining techniques in mixed-type time-series [4].
- Matrix Profile (MP) is an anomaly detection method based on similarity-join to detect time series discords [15].
- Pattern Anomaly Value (PAV) is an anomaly detection algorithm based on pattern anomaly value. The anomalies are the infrequent linear pattern [16] .

To evaluate these methods, we used the implementation available in [4]. These algorithms are window-based approaches. Hence, we used the recommended settings for each of them. We split the time series into sliding windows of length 12 with a step size 6. These anomaly detection algorithms provide an anomaly score for each window. As these algorithms are unsupervised, we build the anomaly detection model on the full-time series data and we evaluate it using $F1$ score. For CDT, we used the appropriate values of hyper-parameters calculated using F1-score as provided in Table 2. All the data sets are normalized between 0 and 1 during the pre-processing phase. For Yahoo datasets and SGE-Electricity we downsampled these datasets from hours to days.

Result Analysis. Table 3 provides the $F1$ score obtained by each algorithm on each of the six univariate time-series datasets. The maximum values of $F1$ score for each dataset are given in bold type. We also calculate the average rank of each method.

Table 3: Evaluation of Anomaly Detection using F1-score.

Dataset \ Algorithm	CDT	PBAD	PAV	MP
SGE_Electricity	**0.76**	0.70	0.74	0.70
SGE_Calorie	0.85	0.80	0.88	**0.91**
Yahoo_A1	**0.92**	0.72	0.75	0.76
Yahoo_A2	**0.99**	0.65	**0.99**	0.76
Yahoo_A3	**1.0**	0.73	0.99	0.70
Yahoo_A4	**0.98**	0.75	0.93	0.96
Average	**0.92**	0.72	0.88	0.80

CDT outperforms the existing baselines in five of the six datasets. It can be observed in Table 3 that our method is more stable for different datasets than baselines. Note that for the competing algorithms, the data should be balanced otherwise, the detection results are poor. We have tested PBAD, PAV and MP on our initial datasets and on a balanced version and we have noticed that its performance highly degrades on the first case. They tend to focus on the accuracy of predictions from the majority class (normal class) which generates poor precision for the minority class (anomaly class).

4.3 Experiments with Rule Learning Algorithms

We compare our CDT method with the following state-of-the-art rule learning algorithms:

- PART is a combination of C4.5 and RIPPER rule learning to produce rules from partial decision trees using C4.5 algorithm [9].
- JRip implements a rule learner and incremental pruning to produce error reduction (RIPPER) [12]. Rules are formed by greedily adding conditions to the antecedent of a rule.

We compare CDT with PART and JRip based on $F1$ score, $Q(\mathcal{R})$ and $F(h)$ score (Table 4) and the number of rules produced by the classifiers (Figure 3). We evaluated these methods using WEKA. We use 10-fold cross validation to test and evaluate the PART and JRip with the standard default setting of WEKA. For CDT and each competitor, we use the hyper-parameters values obtained to maximize $F(h) - score$ as provided in Table 2.

Result Analysis. Table 4 shows the comparison results for each algorithm in the six datasets using $F1$, $Q(\mathcal{R})$ and $F(h)$ score. Note that the results of the F1 score for CDT in Table 4 are different from those in Table 3 because they are not evaluated with the same hyperparameter values (Table 2).

Overall, the average scores show that our approach has got the first position in ranking followed by PART and JRip. CDT outperforms PART and JRip in five of the six datasets in the $F1$ score, in three of the six datasets in the $Q(\mathcal{R})$ score and all datasets in the $F(h)$ score. We can observe that the performance of the rule algorithms varies depending on the number of attributes. Typically, with ω = 31 for the Yahoo_A1 dataset, PART and JRip get a lower $F1$ score, compared with the rest of the datasets. We can observe from the Table 4 that JRip has a high quality of rules $Q(\mathcal{R})$ in three datasets as well as CDT. This is due to the size of its generated rules that are quite short. However, it is less accurate than CDT and PART in almost all datasets. We can also see that none of the baseline algorithms has good $F(h)$ overall data sets. While CDT has the best tradeoff between $F1$ score and $Q(\mathcal{R})$ score.

Fig. 3 shows a summary of the number of rules produced by each method. CDT produces a fewer number of rules between 5 and 16 rules. It is followed by JRip that produced reasonably few rules between 15 and 30 rules. However, PART highly produces rules that are between 24 and 142. This is due to the specificity of the rules generated that have low support.

We present some examples of the generated rules by our CDT algorithm from SGE data sets to detect multiple anomalies in Table 5. As we can see, the rules with visualized patterns are easy to intuitively understand and can be easily interpreted by users. The experts give the following comments: the negative peak is considered an anomaly because the energy consumption in a building cannot be negative. The positive peak has occurred

Table 4: Evaluation of anomaly detection using the $F1$ score, the quality measure $Q(\mathcal{R})$ and the objective function $F(h)$.

Evaluation	F1-score			$Q(\mathcal{R})$			F(h)-score		
Algorithm / Dataset	CDT	PART	JRip	CDT	PART	JRip	CDT	PART	JRip
SGE_Electricity	**0.76**	0.71	0.72	0.67	0.67	**0.70**	**0.51**	0.48	0.50
SGE_Calorie	**0.99**	0.80	0.79	0.61	0.65	**0.69**	**0.60**	0.52	0.54
Yahoo_A1	**0.91**	0.70	0.69	0.48	0.50	**0.56**	**0.43**	0.35	0.39
Yahoo_A2	**0.99**	0.80	0.77	**0.69**	0.68	0.65	**0.68**	0.54	0.50
Yahoo_A3	**0.98**	0.78	0.71	**0.77**	0.69	0.70	**0.75**	0.54	0.50
Yahoo_A4	**0.97**	0.73	0.75	**0.70**	**0.70**	0.68	**0.68**	0.51	0.51
Average	**0.93**	0.75	0.74	**0.65**	0.64	0.64	**0.61**	0.49	0.49

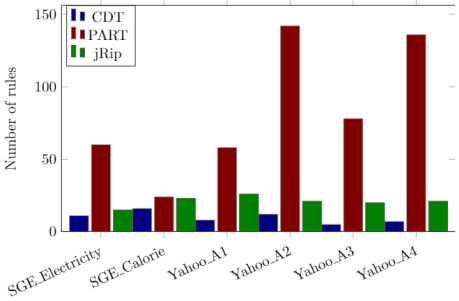

Figure 3: The number of rules generated for anomaly detection.

following overconsumption in the building. The collective anomalies present abnormal variations in successive points. This is due to a fault in the reading of the meters. Finally, the constant anomaly illustrated a stop of the meter.

Table 5: Example of rules generated for anomaly detection in the SGE_Calorie datasets.

Rule Predicate	Representation	Type of anomaly
$[PN_{-H,-H}, SCP_{-H,0}, CST, CST, CST]$ and $\neg\,[ECN_{0,-H}, SCN_{-H,0}]$		negative peak
$[ECP_{0,-L}, PP_{L,H}]$		positive peak
$[PN_{-H,-H}, PP_{H,H}]$		collective anomaly
$[SCN_{H,0}, CST, CST, CST, CST, CST]$ and $\neg\,[PN_{-H,-H}, SCP_{H,0}, CST]$		constant anomaly

5 CONCLUSION

We propose a machine learning method, CDT, that generates human-interpretable rules based on a formalisation of 9 general patterns of variations for multiple anomaly detection in time-series. The approach is based on a modified decision tree which considers nodes as pattern compositions. Using Bayesian Optimization, we optimized the hyper-parameters such that it maximizes both the rules'quality and the classification performances. The performance of the presented method was tested using the SGE and Yahoo datasets. Our approach appeared as robust compared to the existing algorithms in conducted experiments where their model accuracy decreases in case of multiple anomalies and in generating few interpretable rules.

Future work will concern the improvement of the generated rules. For instance, combine rules by a generalization and eliminate redundant rules. Moreover, we could investigate other hyper-parameters such as the size of down-sampling to improve the quality of the generated rules. We could also expand our method to suit multivariate time-series.

Acknowledgment. This PhD. was supported by the Management and Exploitation Service (SGE) of the Rangueil campus attached to the Rectorate of Toulouse and the research is made in the context of the neOCampus project (Paul Sabatier University, Toulouse).

REFERENCES

[1] Charu C. Aggarwal. 2015. Outlier analysis. In Data mining. Springer, Cham, 237–263.
[2] Nahla Barakat and Joachim Diederich. 2005. Eclectic rule-extraction from support vector machines. International Journal of Computational Intelligence 2, 1 (2005), 59–62.
[3] Sina Däubener, Sebastian Schmitt Hao Wang, Peter Krause, and Thomas Bäck. 2019. Anomaly Detection in Univariate Time Series: An Empirical Comparison of Machine Learning Algorithms. ICDM (2019).
[4] Len Feremans, Vincent Vercruyssen, Boris Cule, Wannes Meert, , and Bart Goethals. 2019. Pattern-based anomaly detection in mixed-type time series. Lecture Notes in Artificial Intelligence (2019).
[5] Zengyou He, Xiaofei Xu, Joshua Zhexue Huang, and Shengchun Deng. 2005. FP-outlier: Frequent pattern based outlier detection. Computer Science and Information Systems 2, 1 (2005), 103–118.
[6] Ines Ben kraiem, André Péninou Faiza Ghozzi, Geoffrey Roman-Jimenez, and Olivier Teste. 2020. Automatic Classification Rules for Anomaly Detection in Time-series. RCIS (2020).
[7] Ines Ben kraiem, André Péninou Faiza Ghozzi, and Olivier Teste. 2019. Pattern-based method for anomaly detection in sensor networks. International Conference on Enterprise Information Systems 1 (jan 2019), 104–113.
[8] Nikolay Laptev and Saeed Amizadeh. 2015. A labeled anomaly detection dataset S5 Yahoo Research, v1. https://webscope.sandbox.yahoo.com/catalog.php?datatype=s&did=70
[9] Daud Nor Ridzuan and Corne David Wolfe. 2009. Human readable rule induction in medical data mining. In Proceedings of the European Computing Conference (vol.1 ed.), Vol. 27 LNEE. 787–798.
[10] Jiang Su and Harry Zhang. 2006. A fast decision tree learning algorithm. In AAAI, Vol. 6. 500–505.
[11] William W.Cohen. 1995. Fast effective rule induction. In Machine learning proceedings 1995. Elsevier, 115–123.
[12] Ian H. Witten and Eibe Frank. 2005. Data Mining: Practical Machine Learning Tools and Techniques (2nd ed.). Morgan Kaufmann.
[13] Hu-Sheng Wu. 2016. A survey of research on anomaly detection for time series. In 2016 13th International Computer Conference on Wavelet Active Media Technology and Information Processing. IEEE, 426–431.
[14] Jia Wu, Xiu-Yun Chen, Hao Zhang, and al. 2019. Hyperparameter Optimization for Machine Learning Models Based on Bayesian Optimization. Journal of Electronic Science and Technology 17 (2019), 26.
[15] Chin-Chia Michael Yeh, Yan Zhu, Liudmila Ulanova, and al. 2016. Matrix profile I: all pairs similarity joins for time series: a unifying view that includes motifs, discords and shapelets. In 2016 IEEE 16th international conference on data mining (ICDM). 1317–1322.
[16] Xiao yun Chen and Yan yan Zhan. 2008. Multi-scale anomaly detection algorithm based on infrequent pattern of time series. J. Comput. Appl. Math. 214, 1 (2008), 227–237.