Anomaly detection, which aims to identify data instances that do not conform to the expected behavior, is a classic data mining task with numerous applications in system health monitoring, intrusion detection, anti-money laundering, social media analysis, etc. Over the past decades, numerous methods have been proposed to tackle this challenging problem in different application domains. Examples include one-class classification-based [45, 64, 69], nearest neighbor-based [10, 68], clustering-based [28, 77], isolation-based [40, 41], density-based [31, 39, 62] and deep anomaly detection models based on autoencoders [4, 12, 81, 82], generative adversarial networks [23, 37, 80], to name a few. We refer to [11, 55] for a comprehensive review. However, we realize that most existing studies exclusively focus on the detection task only and ignore the interpretability of the underlying models and their detection results. In fact, anomaly interpretation, which aims to provide explanation of why specific data instances are detected as anomalies, is equally critical for many real-world applications. For instance, when an anomaly is reported by a health monitoring system for a critical device in a factory, human experts need straightforward clues about why this data instance is identified as anomaly, and then can decide what next steps - such as ignorance, model adjustment, fault checking, predictive maintenance and system shutdown – should be taken.

2 RELATED WORK
In most anomaly detection applications, accuracy alone is not sufficient, interpretability and trustworthiness are equally critical [63]. However, building interpretable anomaly detection models is a challenging task due to the lack of anomaly-supervisory information and the unbounded nature of anomalies [52, 63]. Compare to the vast body of literature on the detection task, the research on interpretability in anomaly detection is very limited. There are a few existing works which focus on selecting discriminative features to explain the anomalous part of detected anomalies [6, 13, 53, 54]. Similar to feature selection, mining anomalous aspect such as finding the most outlying subspace of a given object [14, 72], looking for numerical attributes [5] and top feature pairs that best explain anomaly clusters [42] have also been explored. Moreover, considering the intrinsic explanation ability of decision trees, some anomaly detectors adopt decision forest to extract explanation rules [2, 33, 34]. Besides, some deep anomaly detection models leverage attention mechanisms [74], feature-wise reconstruction errors [65] and integrated gradients [67] to provide interpretation for detected anomalies. However, we believe the explanations provided by the existing methods are still not straightforward enough to help users locate/understand the cause of the detected anomalies which is crucial for many real-world applications.

Association rules have been used for anomaly detection since decades ago. Earlier examples include mining association rules for
network intrusion detection [43, 44], credit card fraud detection [8] and fault detection in spacecraft housekeeping data [76]. More recently, association rules are used to detect anomalies in data collected from heating substations [75], education records [60], gas concentration equipment [27], soil moisture probes [78], power systems [35], software systems [18] and clinical systems [24, 70]. A major advantage of association rule-based anomaly detection is its self-explainability, however, there are also certain disadvantages which significantly restrict their application in practice. First, association rules must be generated from a categorical dataset. Second, oftentimes there are too many rules that can be generated which leads to a high false positive rate if all of them are used for anomaly detection. To overcome these problems, several methods are designed to covert continuous data to meaningful categorical data via expert knowledge [51], auto-regressive models [47], clustering and regression models combined with domain-specific rules [15]. Moreover, only invariant rules which achieves 100% confidence are selected for anomaly detection which effectively reduces the false positive rate. However, these methods are still not general enough due to their dependence on the domain knowledge of the underlying data generation process.

3 PRELIMINARIES

3.1 Association Rule Mining

Association rule mining, one of the most important data mining techniques, is used to discover the frequently occurring patterns in the database. The main aim of association rule mining is to find out the interesting relationships and correlations among the different items of the database [29].

Specifically, let \( I = \{i_1, i_2, \ldots, i_m\} \) be a set of items and \( D \) be a set of transactions, where each transaction \( T \) is a set of items such that \( T \subseteq I \). An association rule is expressed as \( X \Rightarrow Y \) where \( X, Y \subseteq I \) and \( X \cap Y = \emptyset \). Furthermore, there are two basic measures for an association rule: support (s) and confidence (c). A rule has support \( s \) if \( s \% \) of the transactions in \( D \) contains \( X \cup Y \). A rule \( X \Rightarrow Y \) has confidence \( c \), if \( c \% \) of transactions in \( D \) that support \( X \) also support \( Y \). Given a set of transactions \( D \) (the database), the problem of mining association rules is to discover all association rules that have support and confidence greater than the user-specified minimum support (called minsup) and minimum confidence (called minconf) [26].

More specifically, association rules are commonly generated using the following two steps: 1) Find all the frequent items whose support is larger than minsup. 2) Based on these frequent items, association rules which have confidence above minconf are generated. The first step is much more difficult than the second step. Many algorithms have been developed to mine frequent itemsets, including Apriori [1], FP-Growth [21] and genetic algorithms [66], etc. Moreover, to make sure all the generated rules are not redundant, one can require all the mined frequent items to be closed [56]. Specifically, an itemset is closed if there exists no superset that has the same support as this original itemset. Association rules generated from closed frequent itemsets are not redundant meaning that if there is an association rule \( X \Rightarrow Y \) then there must not exist another rule \( U \Rightarrow W \), such that \( X \subseteq U \), \( Y \subseteq W \) and the support of \( X \cup Y \) equals to the support of \( U \cup W \). Among the algorithms for mining closed frequent itemsets are AprioriClose [56], LCM [71], CHARM [79], CLOSET+ [73] and FPClose [20].

In practice, the support of different items can vary significantly in the database, thus only using a single minsup can cause problems such as dominating by infrequent items if minsup is too low or only covering a very limited items if minsup is too high. To solve the problems, different rules may need to satisfy different minimum supports depending on what items are in the rules, and we need to mine frequent itemsets with multiple minimum supports. This problem is much harder because the downward closure property (all non-empty subsets of a frequent itemset must also be frequent), which is the basis to reduce the search space of frequent itemsets in aforementioned algorithms, is invalid. Fortunately, there are available algorithms for mining frequent itemsets with multiple minimum support thresholds. Among the examples are MSAPriori [38], CFPgrowth [26] and CFP-growth++ [32]. CFPgrowth and CFP-growth++ are more efficient than MSAPriori when dealing with large databases.

3.2 Decision Trees

Decision trees (DTs) are a family of machine learning algorithms primarily designed for classification and regression. Their representability and ability to produce rules with relevant attributes make them the most commonly used technique when seeking interpretable machine learning models [17, 49].

Specifically, let \( X \) be the input variables with \( M \) dimensions \( X_i \) where \( i = 1, \ldots, M \), \( Y \) be the output variable (\( Y \in \{1, \ldots, C\} \) for classification, \( Y \in \mathbb{R} \) for regression), \( D \) be the dataset formed by sampling from the unknown joint distribution \( P_{X,Y} \). A DT consists of a hierarchy of internal nodes with defined splitting rules based on \( X \), and a set of leaf nodes with predictions about \( Y \). Splitting rules can involve one variable each time leading to univariate DTs, or multiple variables each time leading to multivariate DTs. To promote interpretability, we consider univariate DTs in this work. Following [50], we denote a decision rule on a single variable \( X_i \) as \( f(x) = 1_{(r,\infty)}(x_i) \) where \( 1_{A}(x_i) \) is an indicator function, taking value 1 for \( x_i \in A \) and 0 otherwise. For numerical \( X_i \), we have: \( f(x) = 1_{(r,\infty)}(x_i) \) where \( r \) is the selected cut-off value. Intuitively, a 0/1 outcome directs the instance \( x \) to the left/right child node.

The learning of DTs is mainly composed by induction and pruning. Induction is about learning the DT structure and its splitting rules. [36] shows that learning an optimal DT that maximizes prediction performance while minimizing the size of the tree is NP-complete owing to the discrete and sequential nature of the splits. As a result, standard DT algorithms such as CHAID [30], CART [9], ID3 [58], and C4.5 [59] learn a DT by following locally optimal induction strategies. Specifically, locally optimal induction selects splitting rules that maximize an objective at each node, e.g., variance reduction for regression, maximizing information gain for classification, etc. The splitting procedure stops when a specific criterion, e.g., the maximum depth of the tree and the minimum number of samples required to be at a leaf node, is reached and a leaf node is created. By using this locally optimal search heuristic, learned greedy trees can be very accurate but significantly overfit. To avoid that, pruning techniques are commonly applied to find a
4 INVARIANT RULE DEFINITION AND MINING

Let \( X \) be a \( M \) dimensional variable where each dimension \( X_i \in \mathbb{R} \) is a continuous variable, \( U \) be a \( N \) dimensional variable where each dimension \( U_i \in \{1, \ldots, C\} \) is a categorical variable. We consider a process that generates a stream of multidimensional data instances \( \mathcal{D} = (d^1, d^2, \ldots, d^{[D]}) \), and each data point \( d = (x_1, \ldots, x_M, u_1, \ldots, u_N) \) is a vector consisting of a data instance for \( X \) and \( U \) respectively.

Furthermore, we assume there exist a set of predicates \( \mathcal{P} = \{p_1, p_2, \ldots, p_{|P|}\} \) (how the predicates are generated will be introduced later). For example, \( 5 \leq X_i < 10 \) is a predicate for a continuous variable, \( U_i = 1 \) is a predicate for a categorical variable. Each data point \( d \) may satisfy a subset of predicates in \( \mathcal{P} \).

We further denote the support of a predicate set \( \mathcal{S} \) by \( \sigma(\mathcal{S}) \), representing the fraction of data points where all the predicates in \( \mathcal{S} \) are satisfied. Then, we formally define an invariant rule as follows:

\[
S_1 \implies S_2 \text{ where } S_1, S_2 \subseteq \mathcal{P} \land S_1 \cap S_2 = \emptyset \land \frac{\sigma(S_1 \cup S_2)}{\sigma(S_1)} = 1
\]

where we call \( S_1 \) the antecedent predicate set, \( S_2 \) the consequent predicate set. The rule means whenever a data point satisfies the antecedent set \( S_1 \), it must also satisfy the consequent set \( S_2 \). For example, an invariant rule may look as follows:

\[
\{5 \leq X_1 < 10, X_2 > 20.4, U_1 = 0\} \implies \{X_3 < 7.1, U_2 = 2\}
\]

Notably, we can use the invariant rules for interpretable anomaly detection. Concretely, we can report a data point as an anomaly if the predicates in the antecedent set are satisfied while anyone in the consequent set is not. Since such rules are self-explainable, they are extremely useful for accountable decision-making regarding the detected anomalies. More details of how we utilize such rules for interpretable anomaly detection will be described in subsequent sections.

To guarantee the statistical significance of rules, similar to [15] we also require the support of a rule to be larger than a rule-specific minimum support threshold. Concretely, let \( S_1 \implies S_2 \) be an invariant rule, \( \mathcal{S} = S_1 \cup S_2 \), \( \{p_1, \ldots, p_{|S|}\} \) be all the predicates in \( \mathcal{S} \), we require:

\[
\sigma(\mathcal{S}) > \max \left( \theta, \gamma \min (\sigma(p_1), \ldots, \sigma(p_{|S|})) \right)
\]

where \( \gamma \in (0, 1) \) and \( \theta \in (0, 1) \) are user-defined thresholds. Intuitively, the above condition means that the support of the rule must be larger than a global threshold \( \theta \) to achieve a minimum statistical significance. Moreover, since the support of different predicates can vary significantly, we set a different minimum support threshold for each rule. More specifically, as \( \sigma(\mathcal{S}) \leq \min(\sigma(p_1), \ldots, \sigma(p_{|S|})) \) according to the anti-monotone property (the support of an itemset cannot exceed the support of its subset), we also require the support of the rule to be larger than its specific upper bound scaled by \( \gamma \).

Given the dataset \( \mathcal{D} \) and the predicate set \( \mathcal{P} \), mining such invariant rules is a well-studied problem in association rule mining. Concretely, we first find all frequent predicate sets with multiple minimum support thresholds from the dataset using algorithms such as CFP-growth [26] and CFP-growth++ [32]. Then, to ensure all the generated invariant rules are non-redundant, we only keep closed predicate sets in the frequent sets by filtering out predicate sets with immediate supersets of the same support. Lastly, for an arbitrary closed frequent predicate set \( \mathcal{S} \), we randomly partition it into two non-empty sets \( \mathcal{S} \) and \( \mathcal{S} \), an invariant rule \( \mathcal{S} \implies \mathcal{S} \) is generated if its confidence is 100%, i.e., \( \sigma(\mathcal{S})/\sigma(\mathcal{S}) = 1 \).

5 PREDICATE GENERATION

The quality of the predicate set \( \mathcal{P} \) is of vital importance for our method. Specifically, the generated predicates should have high likelihoods leading to invariant rules that are not self-evident. Furthermore, the form of generated predicates should be as simple as possible to maximize the interpretability of invariant rules. With the above points in mind, we propose two algorithms for generating predicates, one specifically for categorical variables and the other for continuous variables.

Before presenting the algorithms, it is worth noting that we say an invariant rule is self-evident if we know it cannot be violated without seeing the dataset. For example, \( X_1 < 5 \Rightarrow X_1 < 10 \) is obviously a self-evident invariant rule that should not be generated. Moreover, to ensure that any generated predicate \( p \) has nonzero likelihood to contribute to an invariant rule, it is required that \( \sigma(p) > \theta \). It can be easily found that if the support of a predicate \( p \) is less than \( \theta \), then \( p \) has zero likelihood to contribute to any invariant rule because any rule containing \( p \) cannot satisfy the minimum support condition.

5.1 Predicate Generation for Categorical Variables

The algorithm of generating predicates for categorical variables is rather straightforward. Let \( \{1, \ldots, C\} \) be the set of possible values for a categorical variable \( U_i \) and \( c \in \{1, \ldots, C\} \), we generate a candidate predicate \( p : U_i = c \). If \( \sigma(p) > \theta \), we add \( p \) to \( \mathcal{P} \). Otherwise, we add \( p \) to a set \( \mathcal{P} \) which stores candidate predicates whose support are less than the threshold. Then, we traverse all predicates in \( \mathcal{P} \) and use the \( \sigma(\mathcal{S}) \) operator to generate combined predicates until their supports are larger than the threshold. For example, \( \sigma(p_1) < \theta \) and \( \sigma(p_2) < \theta \), we generate a combined predicate \( p : p_1 \land p_2 ) \) if \( \sigma(p_1 \land p_2) > \theta \).

Algorithm 1 gives the details of generating predicates for categorical variables. It is worth noting that by the condition checking at Line 17, the algorithm guarantees that each predicate in \( \mathcal{P} \) will be included in a combined predicate as long as \( \sigma(p_1 \land \ldots \land p_j) > \theta \) assuming \( \mathcal{P} = \{p_1, \ldots, p_j\} \).

5.2 Predicate Generation for Continuous Variables

To maximize interpretability, we generate predicates for each continuous variable by a set of proposed cut-off values. For example, assuming there are three cut-off values \( r_1, r_2, r_3 \) for a variable \( X_i \) while \( r_1 < r_2 < r_3 \), we generate four predicates which are \( X_i < r_1 \), \( r_1 \leq X_i < r_2 \), \( r_2 \leq X_i < r_3 \) and \( X_i \geq r_3 \) if the support for each predicate is larger than \( \theta \). More importantly, predicates generated

---

1For convenience, we drop the superscript for sequential index when the context is clear.
Algorithm 1 Predicate generation for categorical variables

Require: The dataset $D$, the minimum support threshold $\theta$
1. $P \leftarrow \emptyset, \bar{P} \leftarrow \emptyset$
2. for $i = 1, \ldots, N$ do
3. Let $\{1, \ldots, C\}$ be the set of possible values for $U_i$
4. for $c = 1, \ldots, C$ do
5. Generate predicate $p: U_i = c$
6. if $\sigma(p) > \theta$ then
7. Add $p$ to $P$
8. else
9. Add $p$ to $\bar{P}$
10. end if
11. end for
12. end for
13. Assume $\Phi = \{p_1, \ldots, p_J\}$
14. $k \leftarrow 1$
15. for $j = 1, \ldots, J$ do
16. if $\sigma(p_k) > \theta$ then
17. if $\sigma(p_{k+1}) > \theta$ then
18. Generate predicate $p: p_k \ldots p_j$
19. Add $p$ to $P$
20. $k \leftarrow j + 1$
21. else
22. Generate predicate $p: p_k \ldots p_j$
23. Add $p$ to $P$
24. break
25. end if
26. end if
27. end for
28. return $\bar{P}$

Algorithm 2 Predicate generation for continuous variables

Require: The dataset $D$, the minimum support threshold $\theta$
1. $P \leftarrow \emptyset, \bar{P} \leftarrow \emptyset$
2. for $i = 1, \ldots, N$ do
3. Train a DT classification model $DT(X) \rightarrow U_i$
4. For each internal node $j$ in the trained DT model, add $(X_j, \tau_j)$ to $\mathcal{T}$ based on its split rule $1_{(\tau_j, \infty)}(x_j)$
5. end for
6. for $i = 1, \ldots, M$ do
7. Train a DT regression model $DT(X_{-i}) \rightarrow X_i$
8. For each internal node $j$ in the trained DT model, add $(X_j, \tau_j)$ to $\mathcal{T}$ based on its split rule $1_{(\tau_j, \infty)}(x_j)$
9. end for
10. for $j = 1, \ldots, N$ do
11. Get $X_i: (\tau_1, \ldots, \tau_j)$ from $\mathcal{T}$ where $\tau_1 < \ldots < \tau_j$
12. $k \leftarrow 0$
13. for $j = 1, \ldots, J$ do
14. if $k = 0$ and $\sigma(X_i < \tau_j) > \theta$ then
15. Generate predicate $p: X_i < \tau_j$
16. Add $p$ to $P$
17. $k \leftarrow j$
18. end if
19. if $k > 0$ and $\sigma(X_i > \tau_j) > \theta$ then
20. if $\sigma(X_i \geq \tau_j) > \theta$ then
21. Generate predicate $p: X_i \geq \tau_j$
22. Add $p$ to $P$
23. $k \leftarrow j$
24. else
25. Generate predicate $p: X_i \geq \tau_k$
26. Add $p$ to $P$
27. break
28. end if
29. end if
30. if $j = J$ then
31. Generate predicate $p: X_i \geq \tau_k$
32. Add $p$ to $P$
33. end if
34. end for
35. end for
36. return $\bar{P}$

5.3 Boundary Predicates and Rules

We also generate special predicates and rules for variables to detect values that are outside their normal range. Concretely, for each categorical variable $U_i$, we generate an invariant rule $\emptyset \Rightarrow U_i \in \{1, \ldots, C\}$, where $\{1, \ldots, C\}$ is the set of seen values for $U_i$ in $D$. For each continuous variable $X_i$, we generate an invariant rule $\emptyset \Rightarrow \min(\mu_i - 3\sigma_i, x_i) \leq X_i \leq \max(\mu_i + 3\sigma_i, x_i)$, where $\mu_i$ and $\sigma_i$ are the mean and standard deviation of $X_i$ in $D$. $x_i$ and $\bar{x}_i$ are the smallest value and largest value of $X_i$ in $D$.

6 ANOMALY DETECTION AND INTERPRETATION

Let $\mathcal{L} = \{l_1, l_2, \ldots, l_J\}$ be the derived rules in the training phase, we introduce how to calculate an anomaly score for each data point.
## Algorithm 3 Anomaly score calculation

**Require:** The invariant rules $\mathcal{L}$ and a data point $d$

1. Set anomaly score $s \leftarrow 0$
2. for $i = 1, 2, \ldots, |\mathcal{L}|$ do
3. Let $S^1_i$ be the antecedent predicate set and $S^2_i$ be the consequent predicate set of the rule $l_i$
4. if all the predicates in $S^1_i$ are satisfied by $d$ and there is at least one predicate in $S^2_i$ is not satisfied by $d$ then
5. $s \leftarrow s + \sigma(S^1_i \cup S^2_i)$
6. end if
7. end for
8. return $s$

---

*Figure 1:* (left) the number of generated invariant rules with different values for $\theta$ and $\gamma$ in the training phase of the SWAT case; (right) the AUC with different values for $\theta$ and $\gamma$ in detection phase of the SWAT case.

---

## 7 EXPERIMENTS

This section presents the detailed results for two sets of experiments designed to evaluate our proposed method. Before giving experimental details, we first give a brief time efficiency analysis of our method. The training process of our method consists of two steps: predicate generation and invariant rule mining. Since the predicate generation step has almost linear time complexity, the time cost during the training phase is dominated by the invariant rule mining step or the algorithm for mining frequent predicate sets with multiple minimum support thresholds to be more specific. The time complexity of such mining algorithms is largely impacted by the characteristics of the dataset, thus does not have a formal $O$ style formulation. In our experiment, we choose CFP-growth as the mining algorithm due to its implementation simplicity and proved efficiency. We refer to [26] for a more detailed efficiency analysis of the CFP-growth algorithm. In the detection phase, one can easily find that the time complexity is linear with the number of generated invariant rules and the process can be easily parallelized.

### 7.1 Impact of Hyperparameters

In the first experiment, we examine the impact of different values for $\theta$ and $\gamma$ on the anomaly detection performance using SWAT [19], a commonly used benchmark dataset for anomaly detection. More details of the SWAT dataset can be found in Table 1. Specifically, we experiment with $\theta = [0.1, 0.2, 0.3, 0.4, 0.5], \gamma = [0.3, 0.6, 0.9].$

It is conceivable that more invariant rules will be generated with smaller values for $\theta$ and $\gamma$, thus more anomalies can be potentially detected. However, the generated invariant rules will also have lower statistical significance which will lead to a higher false positive rate. Therefore, setting proper values for $\theta$ and $\gamma$ are of vital importance in our method. In Figure 1, we plot the number of generated invariant rules and the Area Under the Receiver Operating Characteristic Curve (AUC ROC) for anomaly detection performance evaluation with different values of $\theta$ and $\gamma$. Furthermore, we also report the F1 score and the corresponding precision and recall of anomaly detection when the anomaly threshold $\varphi$ is set to 0 using different values of $\theta$ and $\gamma$ in Figure 2. As can be seen from the figures, our method achieves good anomaly detection performance when $\theta \leq 0.2$ and $\gamma \geq 0.6$. This is because setting a small value for $\theta$ allows to generate more invariant rules which is beneficial for detecting anomalies in general. Additionally, it is also
rather important to control the “quality” of rules by setting a large value for $\gamma$. Specifically, consider that there is a predicate $p_1$ with a small support but still considerably larger than $\theta$, another predicate $p_2$ with a very large support. In this case, it is very easy to generate a rule $p_1 \Rightarrow p_2$ just because of coincidence. Such rules can be effectively filtered out by setting a large value to $\gamma$ because we require $\sigma(p_1, p_2) > \gamma \times \sigma(p_1)$, i.e., at least $\gamma$ proportion of the data points satisfying $p_1$ must also satisfy the rule. To sum up, we recommend to set a large value, e.g., 0.7, for $\gamma$ in general. Furthermore, we recommend to properly set the value for $\theta$ through cross validation (generating as many rules as possible given an acceptable false positive rate).

7.2 Performance Comparison with Other Models

In the second experiment, we compare the anomaly detection performance of our method with some popular anomaly detection models on various public benchmark datasets in different application domains.

7.2.1 Baseline models. Since it is impossible to cover all the existing anomaly detection models in our experiment and there is no such model that can outperform others statistically in different scenarios according to the recent study [22], we select the following three popular anomaly detection methods: Local Outlier Factor (LOF) [10], Isolation Forest (IF) [40] and Autoencoder (AE) as our baselines. Specifically, LOF is a conventional density-based algorithm based on k-nearest neighbors; IF is a well-known isolation-based algorithm based on tree ensembles; AE is the vanilla version of reconstruction-based method based on neural networks and it often plays as the backbone of more advanced deep anomaly detection models. We implement LOF and IF using Scikit-Learn [57] version 1.1.2, and the default hyperparameter values are used. For the AE model, the number of neurons in the bottleneck layer is set to $1/4$ of the input dimension, and other hyperparameters such as number of hidden layers, number of training epochs are tuned to minimize the reconstruction error using cross validation. Regarding our method, we set $\gamma$ to 0.7, and $\theta$ is tuned to generate as many rules as possible while the false positive rate when $\theta = 0$ is lower than 0.01 using cross validation.

7.2.2 Benchmark datasets. We select six benchmark datasets in three application domains: SWAT [19] and WADI [3] for industrial process condition monitoring, KDDCup99 [25] and Gas Pipeline [48] for network intrusion detection, Annthyroid [61] and Cardio [61] for disease detection. Like [16] that we downsample SWAT and WADI by a factor of 5 for convenience, and the data in the last day of WADI is ignored as they have a severe distribution shift problem compared with previous data. For KDDCup99, we use the 10 percent version for convenience. For all datasets which have anomaly ratio that is larger than 20%, we randomly drop anomalies until the anomaly ratio reaches 20%. Moreover, all the training set in our experiments contains no anomalies. More details of the benchmark datasets are given in Table 1.

7.2.3 Performance metrics and results. The AUC ROC is commonly used as the metric to compare the performance of different anomaly detection algorithms. However, one of the major practical drawbacks of the AUC is that it summarizes the entire ROC curve, including regions with a false positive rate that is far too high for practical applications. As a result, we also report the standardized partial AUC (pAUC) [46] with a false positive rate that is lower than 0.1 to compare the performance of algorithms with a low false positive rate.

The experiment results for the anomaly detection models on the benchmark datasets are reported in Table 2. We find that no single model can always achieve best performance on all datasets, and the AE model has the highest average performance. Regarding our method, we find that it can achieve comparable performance with LOF and IF in terms of AUC in most cases, and comparable performance with AE (better than LOF and IF) in terms of pAUC in most cases. Nevertheless, we would like to point out again that our work is not aimed to beat existing anomaly detection models in terms of AUC and pAUC, but to significantly improve the interpretability of detection results which we believe is an equally important task.
## CONCLUSION

We proposed a data-driven method which leverages decision tree learning and association rule mining to generate invariant rules for anomaly detection. Since our generated rules are self-explainable, they can provide straightforward clues to assist users to understand the cause of the detected anomalies which is of vital importance in practice. Furthermore, experimental results on the benchmark datasets in different application domains show that our method can also achieve comparable detection performance with popular algorithms. Like any study, our proposed method is not without limitation. First, our method currently only works for tabular data, we leave the extension of our method for image and time series data as future work. Second, our method works well for datasets with a considerable number of features, e.g., larger than 10. But its performance could be reduced if there are not enough features where we can derive enough predicates to generate invariant rules to cover the "normal profile" of the data generation process. Nevertheless, we believe our method is a strong addition to the anomaly detection community since providing tangible explanations of anomalies is imperative for this field [63].

---

### Table 1: The details of benchmark datasets.

| Domain                  | Train size | Test size | Num. cont. vars | Num. cat. vars | Anomalies(%) |
|-------------------------|------------|-----------|-----------------|---------------|--------------|
| SWAT                    | 99360      | 89984     | 25              | 15            | 12           |
| WADI                    | 241921     | 15701     | 67              | 26            | 7            |
| KDDCup99                | 78416      | 23577     | 32              | 6             | 20           |
| Gas pipeline            | 58138      | 48111     | 10              | 13            | 20           |
| Anthyroid               | 3998       | 2880      | 6               | 15            | 7            |
| Cardio                  | 1099       | 696       | 19              | 2             | 20           |

### Table 2: Results, in terms of AUC and standardized pAUC, for the anomaly detection models on the benchmark datasets.

| Domain     | LOF | IF  | AE  | Ours |
|------------|-----|-----|-----|------|
|            | AUC | pAUC| AUC | pAUC |
|            |     |     |     |      |
| SWAT       | 0.72 | 0.50  | 0.88 | 0.81  | 0.84 | 0.83  | 0.83 | 0.81  |
| WADI       | 0.53 | 0.61  | 0.71 | 0.56  | 0.71 | 0.65  | 0.72 | 0.70  |
| KDDCup99   | 0.93 | 0.63  | 0.94 | 0.74  | 0.99 | 0.96  | 0.81 | 0.81  |
| Gas pipeline | 0.78 | 0.70  | 0.53 | 0.53  | 0.60 | 0.65  | 0.64 | 0.64  |
| Anthyroid  | 0.73 | 0.54  | 0.67 | 0.52  | 0.77 | 0.62  | 0.60 | 0.59  |
| Cardio     | 0.91 | 0.74  | 0.92 | 0.76  | 0.94 | 0.80  | 0.90 | 0.82  |

| Avg.       | 0.77 | 0.62  | 0.77 | 0.65  | 0.81 | 0.75  | 0.75 | 0.73  |
| Avg. Rank  | 2.83 | 3.17  | 2.50 | 3.33  | 1.67 | 1.50  | 3.00 | 1.83  |

### REFERENCES

[1] Rakesh Agrawal, Ramakrishnan Srikant, et al. 1994. Fast algorithms for mining association rules. In Proc. 20th int. conf. very large data bases, VLDB, Vol. 1215, Citeseer, 487–499.
[2] Diana Laura Aguilar, Miguel Angel Medina Perez, Octavio Loyola-Gonzalez, Kim-Kwang Raymond Choo, and Edoardo Bucheli-Susarrey. 2022. Towards an interpretable autoencoder: a decision tree-based autoencoder and its application in anomaly detection. *IEEE Transactions on Dependable and Secure Computing* (2022).
[3] Chaudhry Mujeeb Ahmed, Venkata Reddy Palletti, and Aditya P. Mathur. 2017. WADI: a water distribution testbed for research in the design of secure cyber physical systems. In *Proceedings of the 3rd International Workshop on Cyber-Physical Systems for Smart Water Networks (CySWATER ’17)*. Association for Computing Machinery, New York, NY, USA, 25–28.
[4] Jinwon An and Sungzoon Cho. 2015. Variational autoencoder based anomaly detection using reconstruction probability. *Special Lecture on IE 2, 1 (2015), 1–18.
[5] Fabrizio Angiulli, Fabio Fassetti, Giuseppe Manco, and Luigi Palopoli. 2017. Outlying property detection with numerical attributes. *Data Mining and Knowledge Discovery 31*, 1 (Jan. 2017), 134–163.
[6] Fatemeh Azmandian, Jennifer G. Dy, Javed A. Aslam, and David R. Kaeli. 2012. Local Kernel Density Ratio-Based Feature Selection for Outlier Detection. In *Proceedings of the Asian Conference on Machine Learning*, PMLR, 49–64.
[7] Rodrigo C Barros, André CPLF De Carvalho, and Alex A Freitas. 2015. Automatic design of decision-tree induction algorithms. Springer.
[8] Rudiger Brause, T Langsdorf, and Michael Hepp. 1999. Neural data mining for credit card fraud detection. In *Proceedings 11th International Conference on Tools with Artificial Intelligence*, IEEE, 103–106.
[9] Leo Breiman, Jerome H Friedman, Richard A Olshen, and Charles J Stone. 2017. Classification and regression trees. *Routledge.*
[10] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. 2000. LOF: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*. 93–103.
[11] Varun Chandola, Arindam Banerjee, and Vipin Kumar. 2009. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41 (3), (2009), 1–58.
[12] Jinghui Chen, Saket Sathe, Charu Aggarwal, and Deepak Turaga. 2017. Outlier detection with autoencoder ensembles. In *Proceedings of the 2017 SIAM international conference on data mining*. SIAM, 90–98.
[13] Xuan Hong Dong, Barbara Micenková, Ira Assent, and Raymond T. Ng. 2013. Local Outlier Detection with Interpretation. In *Advanced Information Systems Engineering*. Vol. 7908. Springer Berlin Heidelberg, Berlin, Heidelberg, 304–320.
[14] Lei Duan, Guanting Tang, Jian Pei, James Bailey, Akiko Campbell, and Changjie Tang. 2015. Mining outgoing aspects on numeric data. *Data Mining and Knowledge Discovery 29*, 5 (2015), 1116–1151.
[15] Cheng Feng, Venkata Reddy Palletti, Aditya Mathur, and Deeph Chana. 2019. A Systematic Framework to Generate Invariants for Anomaly Detection in Industrial Control Systems. In *26th Annual Network and Distributed System Security Symposium*, NDSS 2019, San Diego, California, USA, February 24-27, 2019. The Internet Society.
[16] Cheng Feng and Pengwei Tian. 2021. Time series anomaly detection for cyber-physical systems via neural system identification and bayesian filtering. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 2858–2867.
[17] Alex A Freitas. 2014. Comprehensible classification models: a position paper. *ACM SIGKDD explorations newsletter 15*, 1 (2014), 1–10.
[18] Liang Ge. 2021. APT Detection Based on Association Rule Mining with Bayesian Inference. In *2021 IEEE Sixth International Conference on Data Science in Cyberspace*. 18–21.
Towards Interpretable Anomaly Detection via Invariant Rule Mining

[70] Raihan Ul Islam, Mohammad Shahadat Hossain, and Karl Andersson. 2018. A novel anomaly detection algorithm for sensor data under uncertainty. *Soft Computing* 22, 5 (March 2018), 1623–1639.

[71] Takeaki Uno, Masashi Kiyomi, Hiroki Arimura, et al. 2004. LCM ver. 2: Efficient mining algorithms for frequent/closed/maximal itemsets. In *Fimi*, Vol. 126.

[72] Nguyen Xuan Vinh, Jeffrey Chan, Simone Romano, James Bailey, Christopher Leckie, Kotagiri Ramamohanarao, and Jian Pei. 2016. Discovering outlying aspects in large datasets. *Data Mining and Knowledge Discovery* 30, 6 (Nov. 2016), 1520–1555.

[73] Junyong Wang, Jiawei Han, and Jian Pei. 2003. CLOSET+ searching for the best strategies for mining frequent closed itemsets. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, 236–245.

[74] Hongzuo Xu, Yijie Wang, Songlei Jian, Zhenyu Huang, Yongjun Wang, Ning Liu, and Fei Li. 2021. Beyond Outlier Detection: Outlier Interpretation by Attention-Guided Triplet Deviation Network. In *Proceedings of the Web Conference 2021 (WWW ’21)*. Association for Computing Machinery, New York, NY, USA, 1328–1339.

[75] Puning Xue, Zhigang Zhou, Xiumu Fang, Xin Chen, Lin Liu, Yaowen Liu, and Jing Liu. 2017. Fault detection and operation optimization in district heating substations based on data mining techniques. *Applied Energy* 205 (Nov. 2017), 926–940.

[76] Takehisa Yairi, Yoshiyuki Kato, and Koichu Hori. 2001. Fault detection by mining association rules from house-keeping data. In *Proceedings of the 6th International Symposium on Artificial Intelligence, Robotics and Automation in Space*, Vol. 18. Citeseer, 21.

[77] Dantong Yu, Gholamhosein Sheikholeslami, and Aidong Zhang. 2002. Findout: Finding outliers in very large datasets. *Knowledge and Information Systems* 4, 4 (2002), 387–412.

[78] Ziwen Yu, Alex Bedig, Franco Montalto, and Marcus Quigley. 2018. Automated detection of unusual soil moisture probe response patterns with association rule learning. *Environmental Modelling & Software* 105 (July 2018), 257–269.

[79] Mohammed J Zaki and Ching-Jui Hsiao. 2002. CHARM: An efficient algorithm for closed itemset mining. In *Proceedings of the 2002 SIAM international conference on data mining*. SIAM, 457–473.

[80] Houssam Zenati, Manon Romain, Chuan-Sheng Foo, Bruno Lecosut, and Vijay Chandrasekhar. 2018. Adversarially learned anomaly detection. In *2018 IEEE International conference on data mining (ICDM)*. IEEE, 727–736.

[81] Chong Zhou and Randy C Paffenroth. 2017. Anomaly detection with robust deep autoencoders. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, 665–674.

[82] Bo Zong, Qi Song, Martin Renqiang Min, Wei Cheng, Cristian Lumezanu, Daeki Cho, and Hairong Chen. 2018. Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In *International conference on learning representations*. 