Outlier Summarization via Human Interpretable Rules

Yuhao Deng
dyh18@bit.edu.cn
Beijing Institute of Technology

Yu Wang
yuw164@ucsd.edu
University of California

Lei Cao
lcao@csail.mit.edu
University of Arizona/MIT

Lianpeng Qiao
qiaolp@bit.edu.cn
Beijing Institute of Technology

Yuping Wang
wyp_cs@bit.edu.cn
Beijing Institute of Technology

Jingzhe Xu
xjz@bit.edu.cn
Beijing Institute of Technology

Yizhou Yan
yyan2@wpi.edu
Worcester Polytechnic Institute

Samuel Madden
madden@csail.mit.edu
MIT

ABSTRACT

Outlier detection is crucial for preventing financial fraud, network intrusions, and device failures. Users often expect systems to automatically summarize and interpret outlier detection results to reduce human effort and convert outliers into actionable insights. However, existing methods fail to effectively assist users in identifying the root causes of outliers, as they only pinpoint data attributes without considering outliers in the same subspace may have different causes.

To fill this gap, we propose STAIR, which learns concise and human-understandable *rules* to summarize and explain outlier detection results with *finer* granularity. These rules consider both attributes and associated values. STAIR employs an interpretation-aware optimization objective to generate a small number of rules with minimal complexity for strong interpretability. The learning algorithm of STAIR produces a rule set by iteratively splitting the large rules and is optimal in maximizing this objective in each iteration. Moreover, to effectively handle high dimensional, highly complex data sets that are hard to summarize with simple rules, we propose a *localized* STAIR approach, called L-STAIR. Taking data locality into consideration, it simultaneously partitions data and learns a set of localized rules for each partition. Our experimental study on many outlier benchmark datasets shows that STAIR significantly reduces the complexity of the rules required to summarize the outlier detection results, thus more amenable for humans to understand and evaluate.

PVLDB Reference Format:
Yuhao Deng, Yu Wang, Lei Cao, Lianpeng Qiao, Yuping Wang, Jingzhe Xu, Yizhou Yan, and Samuel Madden. Outlier Summarization via Human Interpretable Rules. PVLDB, 17(7): 1591 - 1604, 2024.
doi:10.14778/3654621.3654627
PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/baodaBBji/anonymous-Tech-Report.

Lianpeng Qiao and Yuping Wang are the corresponding authors.

Proceedings of the VLDB Endowment, Vol. 17, No. 7 ISSN 2150-8097.
doi:10.14778/3654621.3654627

1 INTRODUCTION

Motivation. Outlier detection is critical in enterprises, with applications ranging from preventing financial fraud in finance and defending network intrusions in cyber security, to detecting imminent device failures in IoT.

To turn outliers into actionable insights, users often expect an outlier detection system to produce human understandable information to explain the detected outliers. Otherwise, these outliers are just a set of isolated data objects without any indication of their significance to users. It is thus hard for the users to quickly diagnose the key factors that cause the outliers and fix the problems promptly. Further, outlier detection frequently returns a large number of outlier candidates. This raises the problem of how to best present results such that the users do not have to sift through a huge number of results to assess the validity of the outliers one by one.

If a system is able to *summarize* outlier detection results into groups and explicitly *explain* why each group of objects is considered to be *abnormal* or *normal*, this will greatly reduce the effort of users in evaluating outlier detection results.

State-of-the-Art. In the literature, some works target explaining outliers. Scorpion [69] produces meaningful explanations for anomalies in aggregation queries when the 'cause' of an outlier is contained in its provenance. Similar to Scorpion, Cape [47] aims to explain the outliers in aggregation queries, but using the objects that counterbalance the outliers. Both works do not tackle the problem of summarizing outliers. Macrobase [17] explains outliers by mining the association between the outliers and some external attributes which are *not* used to detect anomalies such as the location of the sensors, time of occurrence, software version, etc.

Similarly, LookOut [36] identifies some *attribute pairs* to explain the detected outliers. These attribute pairs, if used to detect outliers, will produce similar results to detecting outliers on all attributes, thus potentially the key factors that make the outliers 'abnormal'.

Intuitively, we could adapt LookOut to summarize the detected outliers, for example, by grouping the outliers together if they can be identified by the same set of attributes. However, summarizing and explaining outliers at the attribute level is problematic. Even if some outliers can be captured by analyzing the same set of attributes, potentially they could be caused by totally different problems and hence do not necessarily share similar properties. As an example,

when using outlier detection techniques to detect spoofing in social networks, although some anomalies can be detected by analyzing the event location and time, it is unlikely that the anomalies that occur at totally different locations or times are strongly correlated.

Therefore, explaining and summarizing outliers at this *coarse-grained*, attribute level is not sufficient to help the users quickly identify the key factors resulting in the detected outliers. In this work, we thus propose a *fine-grained* methodology that takes the *value* of the attributes into consideration. That is, in addition to a set of attributes, it also reveals to the users the *conditions* that these attributes should satisfy to make the outliers stand out.

Challenges. Summarizing and interpreting outliers at this fine-granularity is challenging because taking the values of the data objects into consideration will lead to a prohibitively large search space, exponential to the number of attributes and their distinct values. On the other hand, this summarization and interpretation still has to be easily understandable for the human.

Proposed Approach. In this work, we propose STAIR, which effectively produces a set of fine-grained human understandable abstractions, each describing the common properties of a group of detection results. This allows the users to efficiently verify a large number of outlier detection results and diagnose the key factors resulting in the potential outliers by only examining a small set of interpretable abstractions.

Rule-based Outlier Summarization and Interpretation. STAIR adapts the classical decision tree classification to learn a compact set of human understandable *rules* to summarize and explain the outlier detection results. Using the results produced by an outlier detection method as training data, STAIR learns a decision tree to accurately separate outliers and inliers in the training set. Each branch of the decision tree is composed of a set of *data attributes* with associated *values* that iteratively split the data. Therefore, it can be thought of as a *rule* that represents a subset of data sharing the same class (outlier or inlier) and that is easy to understand by humans.

Outlier Summarization and Interpretation-aware Objective. However, decision tree algorithms target maximizing the classification accuracy. Rules learned in this way do not necessarily have the properties desired by outlier summarization and interpretation. This is because when handling highly complex data sets, to minimize classification errors, decision trees often have to be *deep* trees with *many* branches and hence produce a lot of complex rules which are hard for humans to understand. Although some methods like CART [20] have been proposed to prune a learned decision tree in a post-processing step, they target avoiding overfitting and thus lifting the classification accuracy. They do not guarantee the simplicity of each rule.

To solve the above issues, we propose a new optimization objective customized to outlier summarization and interpretation. It targets producing a minimal number of rules that are as simple as possible, while still assuring the classification accuracy. However, the simplicity requirement of outlier summarization and interpretation conflicts with the accuracy requirement, while it is hard for the users to manually set an appropriate regularization term to balance the two requirements. STAIR thus introduces a learnable regularization parameter into the objective and relies on the learning algorithm to automatically make the trade-off.

Rule Generation Algorithm. We then design an optimization algorithm to generate the summarization and interpretation-aware rules. Similar to the classic decision tree algorithms [29], STAIR produces a rule set by iteratively splitting the decision node. In each iteration, STAIR dynamically adjusts the regularization parameter to ensure that it is always able to produce a valid split which increases the objective. We prove that the regularization parameter and the rule split that STAIR learns in each iteration as a combination is *optimal* in maximizing the objective. Note like the classical decision tree algorithms, STAIR is able to handle both numerical and categorical values, thus applicable to various types of datasets.

Localized Outlier Summarization and Interpretation. To solve the problem that one single decision tree with a small number of simple rules is not adequate to satisfy the accuracy requirement when handling high dimensional, highly complex data sets, we propose a *localized* STAIR approach, called L-STAIR. Taking data locality into consideration, L-STAIR divides the whole data set into multiple partitions and learns a localized tree for each partition. Rather than first partitioning the data and then learning the localized tree in two disjoint steps, L-STAIR jointly solves the two sub-problems. In each iteration, it optimizes the data partitioning and rule generation objectives alternatively and is guaranteed to converge to a partitioning that can be summarized with simple rules.

Contributions. The key contributions of this work include:

- To the best of our knowledge, STAIR is the first approach that summarizes the outlier detection results with human interpretable rules, and it is generally applicable for summarizing the prediction results of any binary classification models.
- We define an outlier summarization and interpretation-aware optimization objective which targets producing the minimal number of rules with the least complexity, while still guaranteeing the classification accuracy.
- We design a rule generation method that is optimal in optimizing the STAIR objective in each iteration.
- We propose a localized STAIR approach which jointly partitions the data and produces rules for each local partition, thus scaling STAIR to high dimensional, highly complex data.
- Our extensive experimental study using 10 datasets demonstrates that compared to 7 rule-based methods that explain outliers and machine learning prediction, STAIR significantly reduces the complexity and the number of rules required to summarize outlier detection results.

2 PRELIMINARY: DECISION TREE

In this section, we overview the decision tree classification problem and its classical learning algorithms.

Decision Tree Overview. Decision tree learning is a classical classification technique where the learned function can be represented by a decision tree. It classifies instances by sorting them down the tree from the root to the leaf node, which could predict the label of this instance. Each node in the tree denotes the test of the specific attribute, and the instance is classified by moving down the tree

branch from this node according to the value of the attribute in the given example.

Learning Algorithms. Most algorithms learn the decision trees in a top-down, greedy search manner such as ID3 [54] and its successor C4.5 [55]. The basic algorithm, ID3, will run a *statistical test* on choosing the instance attribute to determine how well it could classify the data points. From the root node, the algorithm will find the best attribute to form branches and then put all the training examples into the corresponding child nodes. It then repeats this entire process using the training data associated with the child nodes to select the appropriate attribute and value for the current node and form new branches from the child nodes.

Information Gain-based Statistical Test. There are several strategies for the statistical test in each step. One of the most popular tests is *information gain*, which measures how well a given attribute could separate the training examples. Before giving the precise definition of information gain, we need to give the definition of *entropy* first. Given a data collection S, containing positive and negative examples, the entropy of S is:

$$Entropy(S) = -p_+ \log_2 p_+ - p_- \log_2 p_- \tag{1}$$

where p_+ and p_- are the proportion of positive and negative examples in S, respectively.

Next, we give the formulation of the information gain of an attribute A with split value v, relative to a collection of examples S:

$$Gain(S, A, v) = Entropy(S) - \sum_{b \in Branches} \frac{|S_b|}{|S|} Entropy(S_b) \tag{2}$$

where *Branches* contains two branches, each of which has the training examples with attribute A smaller or larger than the value v, respectively. S_b refers to the collection of examples from branch b. The learning algorithm iteratively splits nodes and forms branches by maximizing Eq. 2 at each step.

Learning the decision tree in this way is equivalent to maximizing the global objective:

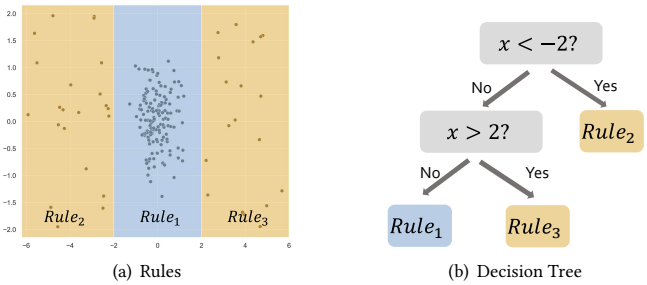
$$\max \sum_{l \in L} n_l (1 - Entropy(S_l)) \tag{3}$$

where S_l represents the collection of training examples in the leaf node l and n_l represents the number of examples falling into node l.

3 RULE-BASED SUMMARIZATION AND INTERPRETATION

In this section, we first give the definition of **rule** and then explain why rules are good at summarizing and interpreting outlier detection results.

Definition 3.1. Given a data set $\mathbb{D}$ in a N-dimensional feature space $[x_1, x_2, \cdots, x_N]$, a **Rule** R_i is defined as $R_i = (a_1 \le x_1^i \le b_1) \wedge (a_2 \le x_2^i \le b_2), \cdots, \wedge (a_j \le x_j^i \le b_j), \cdots, \wedge (a_L \le x_L^i \le b_L)$. $\forall$ clause $(a_j \le x_j^i \le b_j)$ of R_i, x_j^i corresponds to one attribute $x_j \in \{x_1, x_2, \cdots, x_N\}$; a_j and b_j $(a_j < b_j)$ fall in the domain range of attribute x_j. L indicates the number of attributes in rule R_i, or the **length** of R_i.



(a) Rules (b) Decision Tree

Figure 1: Example of rules and decision tree. The blue partition covered by $Rule_1$ represents inliers, while the brown partitions covered by $Rule_2$ and $Rule_3$ represent outliers.

By Def. 3.1, a rule R_i corresponds to a conjunction of domain value intervals, each with respect to some attribute x_j. Rule R_i covers a data subset $\mathbb{D}_i \subseteq$, where $\forall$ object $d_i \in \mathbb{D}_i$, the attributes of object d_i fall into the corresponding interval.

In the decision tree model [61], each branch corresponds to one rule. Fig. 1(a) shows a toy decision tree T_i learned from a 2-dimensional data set $\mathbb{D}$. T_i classifies the objects in $\mathbb{D}$ into outliers and inliers. It has three branches, corresponding to 3 rules: $R_1 = (-2 \le x_1 \le 2)$, $R_2 = (x_1 > 2)$, and $R_3 = (x_1 < -2)$. All rules only contain one attribute x_1. Rules R_2 and R_3 are lower bounded or upper bounded only. Thus the length of these rules is one.

Note the length of a rule is not equivalent to the depth of the tree. The depth of the decision tree in Figure 1(b) is two, while the lengths of the three rules are all one. Even if the decision tree gets deeper, the lengths of the rules could still be small. This is because a decision tree could use one attribute multiple times on one single branch (rule).

These rules classify the whole data set into three different partitions. Rule R_1 covers all inliers in $\mathbb{D}$, while both R_2 and R_3 represent outliers.

Rules effectively summarize and interpret the outliers and inliers in the data. The merit is twofold. First, each rule covers a set of inliers or outliers. Therefore, rather than exhaustively evaluating a large number of outliers or inliers one by one, the users now only have to evaluate a small number of rules, thus saving a huge amount of human effort. Second, the rules are human interpretable, helping the users easily understand why an object is considered as outlier or inlier and identify the root cause of the outliers. For example, rules R_2 and R_3 intuitively tell users that some objects are abnormal because their x_1 values are too large or too small.

4 THE OPTIMIZATION OBJECTIVE OF RULES GENERATION

4.1 The Insufficiency of Classic Decision Trees

Intuitively, to produce rules effectively summarizing and interpreting the outlier detection results, we could directly apply the classical decision tree algorithms such as ID3 [29]. That is, we use the output of the outlier detection method as ground truth labels to train a decision tree model and then extract rules from the learned decision tree.

However, decision tree algorithms target producing rules that maximize classification accuracy. The rules learned in this way do not necessarily have the desired properties when used in outlier summarization and interpretation, for the following reasons:

First, they may produce rules that contain many attributes and thus are too complicated for humans to evaluate. For example, humans can easily understand and reason on a rule with a couple of attributes such as the rules in Fig. 1, while it will be much harder for humans to obtain any meaningful information from a complicated rule with many attributes. For instance, the rule with 20 attributes $a_1 \leq x_1 \leq b_1, \cdots, a_{20} \leq x_{20} \leq b_{20}$ will be almost impossible for human to understand.

Second, to maximize the classification accuracy they may produce many rules. However, to reduce the human evaluation efforts, ideally, we want to produce as few rules as possible.

The above situations could happen when handling highly complex data sets which often require a *deep* tree with *many* branches.

4.2 Summarization and Interpretation-aware Objective

To address the above concerns, we design an optimization objective customized to outlier summarization and interpretation. It targets producing a minimal number of rules that are as simple as possible, while still guaranteeing the classification accuracy. The objective is composed of two sub-objectives, namely *length objective* and *entropy objective*.

Length Objective. To minimize the number of rules as well as bounding the complexity of each rule, we first introduce an objective with respect to the lengths of the rules in the rule set $\mathcal{R}$:

$$\min_{\mathcal{R}} \mathcal{L}(\mathcal{R}), \text{ where } \mathcal{L}(\mathcal{R}) = \sum_{r_i \in \mathcal{R}} L(r_i) \quad \text{s.t. } L(r_i) \leq L_m \tag{4}$$

In Eq. 4, $\mathcal{R}$ denotes a rule set. $L(r_i)$ denotes the length of a rule r_i in $\mathcal{R}$. L_m is the predefined maximal length of each rule that the users allow. Essentially, the total length of all rules represents the complexity of the learned model. Minimizing it will effectively reduce the number of rules, while at the same time simplifying each rule.

Entropy Objective. To maximize the classification accuracy of the derived model, we adopt the entropy-based optimization objective from the classical decision tree algorithms [29], i.e. ID3 and C4.5, as illustrated in Sec. 2.

$$\max_{\mathcal{R}} \mathcal{S}(\mathcal{R}), \text{ where } \mathcal{S}(\mathcal{R}) = \sum_{r_i \in \mathcal{R}} n_{r_i} E(r_i) \tag{5}$$

In Equation 5, $E(r_i)$ corresponds to $1 - Entropy(r)$. Maximizing Eq. 5 effectively maximizes the classification accuracy.

Combing Eq. 5) and Eq. 4), our summarization and interpretation-aware objective (Eq. 7) maximizes the classification accuracy, while at the same time minimizing the total length of the rules.

$$\max_{\mathcal{R}} \mathcal{S}(\mathcal{R}) = \frac{\sum_{r_i \in \mathcal{R}} n_{r_i} E(r_i)}{\sum_{r_i \in \mathcal{R}} L(r_i)} \quad \text{s.t. } L(r_i) \leq L_m, F1(\mathcal{R}) > F1_m \tag{6}$$

where L_m corresponds to the *maximal* length of a rule that the users allow, while $F1_m$ is a predefined requirement on classification accuracy which is measured by F1 score in the case of outlier detection.

Optimization Issue. However, in practice we observed that this objective caused issues in the optimization process. Maximizing the entropy objective typically will lead to more complex rules and in turn the increase of the length objective. However, the length objective often increases faster than the entropy objective. Therefore, the overall objective (Eq. 6) tends to stop increasing in a few iterations.

Final Objective: Introducing a Stabilizer. To solve this problem, we introduce a stabilizer M into the length objective – the denominator of Eq. 6:

$$\max_{\mathcal{R}, M} \mathcal{S}(\mathcal{R}, M) = \frac{\sum_{r_i \in \mathcal{R}} n_{r_i} E(r_i)}{\sum_{r_i \in \mathcal{R}} L(r_i) + M} \quad \text{s.t. } L(r_i) \leq L_m, F1(\mathcal{R}) > F1_m \tag{7}$$

The stabilizer M mitigates the impact of the quickly increasing length objective. It ensures that the length objective does not dominate our summarization and interpretation-aware objective. Intuitively, in the extreme case of setting M to an infinitely large value, the increase of the total rule length is negligible to the objective. Now maximizing Eq. 7 in fact is equivalent to the traditional entropy-based decision tree.

Auto-learning Stabilizer M. An appropriate value of M is critical to the quality of the learned rules. However, relying on the users to manually tune it is difficult. First, M can be any positive value and thus has an infinite number of options. Second, ideally, M should dynamically change to best fit the evolving rule set produced in the iterative learning process. Therefore, rather than make it a hyperparameter, M is a learnable parameter in our objective function Eq. 7.

5 STAIR: RULE GENERATION METHOD

This section introduces our SummarizaTion And Interpretation-aware Rule generation method (STAIR). Similar to the classic decision tree algorithms [29], STAIR produces a rule set by iteratively splitting the decision node. We prove that in each iteration STAIR is *optimal* in maximizing our objective in Eq. 7.

Below we first give the overall process of STAIR:

(1) Initialize the stabilizer M in Eq. 7 to zero;
(2) Increase the value of M;
(3) Find a node to split that could increase the objective in Eq. 7; go to step 2.

In short, STAIR iteratively increases the value of M and splits the nodes. Next, we first show that the value of M is critical to the performance of STAIR and then introduce a method to calculate the optimal value of M at each iteration.

5.1 The Value of M Matters

Given a set of rules $\mathcal{R}$, dividing a node n is tantamount to partitioning a rule r in $\mathcal{R}$ into two separate rules r_1 and r_2, where r_1 and r_2 correspondingly conclude at the two child nodes of node n. For a given M and a set of rules $\mathcal{R}$, we define a split $sp(\mathcal{R}, M)$ to be **valid** if $\mathcal{S}(\mathcal{R} \backslash \{r\} \cup \{r_1, r_2\}, M) > \mathcal{S}(\mathcal{R}, M)$. That is, a valid

split will increase the objective defined in Eq. 7. For the ease of presentation, we use $\mathcal{S}(\mathcal{R}', M)$ to denote $\mathcal{S}(\mathcal{R}\backslash\{r\} \cup \{r_1, r_2\}, M)$

Next, we show the smallest M that could produce a valid split is optimal in maximizing Eq. 7.

Theorem 5.1. ***Monotonicity Theorem.*** *Given a rule set $\mathcal{R}$, if $M_a > M_b$, then $\mathcal{S}(\mathcal{R}'_a, M_a)$ is guaranteed to be **smaller** than $\mathcal{S}(\mathcal{R}'_b, M_b)$, where $\mathcal{R}'_a, \mathcal{R}'_b$ denotes the rule set produced by a valid split on $\mathcal{R}$ that maximizes the objective given M_a or M_b.*

Proof. Because $M_a > M_b$, we have:

$$S(\mathcal{R}'_a, M_a) = \frac{\sum_{r\in\mathcal{R}'_a} n_r E(r)}{\sum_{r\in\mathcal{R}'_a} L(r) + M_a} < \frac{\sum_{r\in\mathcal{R}'_a} n_r E(r)}{\sum_{r\in\mathcal{R}'_a} L(r) + M_b} = S(\mathcal{R}'_a, M_b) \tag{8}$$

Because $\mathcal{R}'_b$ corresponds to the best split given M_b, we obtain:

$$S(\mathcal{R}'_a, M_b) \leq S(\mathcal{R}'_b, M_b) \tag{9}$$

From Eq. 8 and Eq. 9, we have:

$$\mathcal{S}(\mathcal{R}'_a, M_a) < \mathcal{S}(\mathcal{R}'_b, M_b) \tag{10}$$

This concludes our proof. □

5.2 Calculating the Optimal M

By Theorem 5.1, to maximize the objective at each iteration, it is necessary to search for the smallest value of M that could produce a valid split. Intuitively we could find the optimal M by gradually increasing the value of M at a fixed step size. However, this is neither effective nor efficient, because it is hard to set an appropriate step size. If it is too large, STAIR might miss the optimal M. On the other hand, if the step size is too small, STAIR risks incurring many unnecessary iterations not producing any valid splits.

To solve the above problem, we introduce a method that uses the concept of *boundary stabilizer* to directly calculate the optimal M. Moreover, the best splitting is discovered as the by-product of this step.

We use M_o to denote the optimal M. Because M_o is the smallest M that could produce a valid split, then $\forall r_0$ and $\forall r_1, r_2$, where r_1 and r_2 represent the rules produced by splitting rule r_0, Eq. 11 holds:

$$\mathcal{S}(\mathcal{R}, M) > \mathcal{S}(\mathcal{R}\backslash\{r_0\} \cup \{r_1, r_2\}, M), \forall M < M_o \tag{11}$$

Boundary Stabilizer M. To compute M_o, we first define a boundary M denoted as M_b which makes Equation 12 hold:

$$\mathcal{S}(\mathcal{R}, M_b) = \mathcal{S}(\mathcal{R}\backslash\{r_0\} \cup \{r_1, r_2\}, M_b) \tag{12}$$

By Eq. 12, setting the M to M_b will produce a split that does not change the objective. That is, under M_b no valid split will increase the objective. However, there exists a split that does not decrease the objective. So M_b is called the boundary M,

We then expand Eq. 12 as follows:

$$\frac{\sum_{r\in\mathcal{R}\backslash\{r_0\}} n_r E(r) + n_{r_0} E(r_0)}{\sum_{r\in\mathcal{R}\backslash\{r_0\}} L(r) + L(r_0) + M_b} = \frac{\sum_{r\in\mathcal{R}\backslash\{r_0\}} n_r E(r) + n_{r_1} E(r_1) + n_{r_2} E(r_2)}{\sum_{r\in\mathcal{R}\backslash\{r_0\}} L(r) + L(r_1) + L(r_2) + M_b} \tag{13}$$

We define $A = \sum_{r\in\mathcal{R}\backslash\{r_0\}} n_r E(r)$, $B = \sum_{r\in\mathcal{R}\backslash\{r_0\}} L(r)$, and $A_0 = \sum_{r\in\mathcal{R}} n_r E(r)$, $B_0 = \sum_{r\in\mathcal{R}} L(r)$, then Eq. 13 could be rewritten as:

$$\frac{A + n_{r_0} E(r_0)}{B + L(r_0) + M_b} = \frac{A + n_{r_1} E(r_1) + n_{r_2} E(r_2)}{B + L(r_1) + L(r_2) + M_b} \tag{14}$$

Then after some mathematical transformation, we obtain:

$$\begin{aligned} &M_b(n_{r_1} E(r_1) + n_{r_2} E(r_2) - n_{r_0} E(r_0)) \\ &= n_{r_0} E(r_0)(L(r_1) + L(r_2)) + A(L(r_1) + L(r_2) - L(r_0)) \\ &- B(n_{r_1} E(r_1) + n_{r_2} E(r_2) - n_{r_0} E(r_0)) \\ &- (n_{r_1} E(r_1) + n_{r_2} E(r_2)) L(r_0) \end{aligned} \tag{15}$$

Denoting $\Delta L = L(r_1) + L(r_2) - L(r_0)$ and $\Delta E = n_{r_1} E(r_1) + n_{r_2} E(r_2) - n_{r_0} E(r_0)$, we simplify Eq. 15 to:

$$\begin{aligned} M_b \Delta E &= n_{r_0} E(r_0)(L(r_1) + L(r_2)) + A\Delta L - B\Delta E \\ &- (n_{r_1} E(r_1) + n_{r_2} E(r_2)) L(r_0) \\ &= A\Delta L - B\Delta E + n_{r_0} E(r_0)\Delta L - L(r_0)\Delta E \\ &= (A + n_{r_0} E(r_0))\Delta L - (B + L(r_0))\Delta E \end{aligned} \tag{16}$$

$$M_b = A_0 \frac{\Delta L}{\Delta E} - B_0, \forall r_0 \in \mathcal{R}, \forall r_1, r_2 \tag{17}$$

$\forall M > M_b$, with the same r_0 and r_1, r_2 in Eq. 12, Eq. 15 becomes:

$$\begin{aligned} &M(n_{r_1} E(r_1) + n_{r_2} E(r_2) - n_{r_0} E(r_0)) \\ &> n_{r_0} E(r_0)(L(r_1) + L(r_2)) + A(L(r_1) + L(r_2) - L(r_0)) \\ &- B(n_{r_1} E(r_1) + n_{r_2} E(r_2) - n_{r_0} E(r_0)) \\ &- (n_{r_1} E(r_1) + n_{r_2} E(r_2)) L(r_0) \end{aligned} \tag{18}$$

Note that with expanding r_0 to r_1, r_2, the entropy of the rules must be lower, which means $n_{r_1} E(r_1) + n_{r_2} E(r_2) - n_{r_0} E(r_0) > 0$.

Then from Eq. 15 to Eq. 12, we easily obtain Eq. 19 from Eq. 18:

$$\mathcal{S}(\mathcal{R}, M) < \mathcal{S}(\mathcal{R}\backslash\{r_0\} \cup \{r_1, r_2\}, M), \forall M > M_b \tag{19}$$

That is, an M larger than M_b is guaranteed to produce a valid split – splitting rule r_0 to r_1 and r_2.

Calculating Optimal M. According to the Monotonicity theorem (Theorem 5.1), a smallest M is the best in maximizing the objective. Therefore, STAIR can directly calculate M_o using Eq. 20:

$$M_o > \min_{\Delta L/\Delta E} A_0 \frac{\Delta L}{\Delta E} - B_0, \forall r_0 \in \mathcal{R}, \forall r_1, r_2 \tag{20}$$

That is, STAIR first finds a rule r_0 from $\mathcal{R}$ that after split into two rules, produces the smallest $A_0 \frac{\Delta L}{\Delta E}$. STAIR then sets M_o as a value larger than $\frac{\Delta L}{\Delta E}$ - B_0. In this way, STAIR successfully calculates the optimal M and finds the best split in one step, making its learning process effective yet efficient.

5.3 STAIR Learning Algorithm

Algorithm 1 shows the learning process of STAIR. It starts with initializing M as 0 (Line 1) and uses a min heap structure H to keep all nodes. Similar to the decision tree algorithms, it initializes H to contain only the root node (Line 2). It then sets the rule set $\mathcal{R}$ to contain only one rule r_0 corresponding to the root node (Line 3). By default, rule r_0 classifies all training samples as inliers. Then based on Eq. 20, STAIR iteratively extracts a rule r_0, calculates ($\frac{\Delta L}{\Delta E}$), updates M, and splits r_0 into two rules r_1 and r_2. After each split, it calculates ($\frac{\Delta L}{\Delta E}$) with respect to r_1/r_2, refreshes the rule set $\mathcal{R}$

Algorithm 1: Learning Algorithm of STAIR

Input: Training data X, $F1$ score threshold $F1_m$
Output: The target rule set.

Initialize M to be zero;
Initialize the min heap H to contain only the root node;
Set the rule set $\mathcal{R} = \{r_0\}$;
$A_0 = nE(r_0), B_0 = 0$;
while *True* **do**
  extract from H a rule r_0 which has the minimal $\frac{\Delta L}{\Delta E}$;
  Set $M = A_0(\frac{\Delta L}{\Delta E})_{r_0} - B_0$;
  while *the minimal* $(\frac{\Delta L}{\Delta E})_{r_0}$ *from* $H \leq \frac{M+B_0}{A_0}$ **do**
    Extract rule r_0 with the minimal $(\frac{\Delta L}{\Delta E})_{r_0}$ from H;
    Split r_0 into r_1, r_2 ;
    Insert r_1, r_2 into $\mathcal{R}$ and H; Maintain the heap H according to $(\frac{\Delta L}{\Delta E})_{r_1}$ and $(\frac{\Delta L}{\Delta E})_{r_2}$;
    $A_0 \leftarrow A_0 + E(r_1) + E(r_2) - E(r)$;
    $B_0 \leftarrow B_0 + L(r_1) + L(r_2) - L(r)$;
  end
  Calculate the $F1$-score of the current rule set as $F1(\mathcal{R})$;
  if $F1(\mathcal{R}) > F1_m$ **then**
    Break;
end

and min heap H, and updates A_0 and B_0 accordingly. The learning process will terminate when the following conditions hold: (1) the accuracy reaches the requirement specified by users; and (2) the $\mathcal{S}(\mathcal{R}, M)$ does not increase in a few iterations.

Complexity Analysis. Compared to the classical decision tree algorithms, the additional overhead that STAIR introduces is negligible. In each iteration, STAIR extracts the rule r_0 from min heap H and inserts into H the new rules. Assume there are n nodes in the tree. Because the complexity of the min heap's retrieve and insert operations is $O(\log n)$, the additional complexity is $O(n \log n)$.

6 LOCALIZED STAIR: DATA PARTITIONING & RULE GENERATION

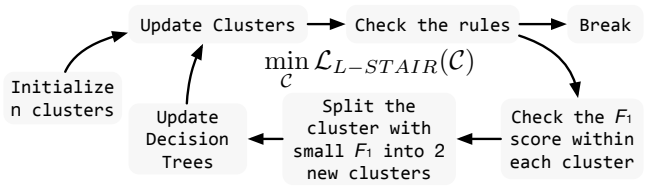


Figure 2: The Localized STAIR

As shown in our experiments (Sec. 7), although in general STAIR performs much better than the classical decision tree algorithms in producing summarization and interpretation friendly rules, its performance degrades quickly on high dimensional, highly complex data sets, for example on the *SpamBase* data set which has 57 attributes. This is because a single decision tree with a small number of simple rules is not powerful enough to model the complex distribution properties underlying these data sets.

To solve this problem, we propose a *localized* STAIR approach, so called L-STAIR. L-STAIR divides the whole data set into multiple partitions and learns a tree model for each partition. Taking the data locality into consideration, L-STAIR produces data partitions where the data in each partition share similar statistical properties, while different partitions show distinct properties. L-STAIR thus is able to produce localized, simple rules that effectively summarize and explain each data partition.

Next, we first introduce the objective of L-STAIR in Sec. 6.1 and then give the learning algorithm in Sec. 6.2.

6.1 Joint Optimization of Data Partitioning and Rule Generalization

Intuitively, L-STAIR could produce the localized rules in two disjoint steps: (1) partitioning data using the existing clustering algorithms such as k-means [37] or density-based clustering [30]; (2) directly applying STAIR on each data partition one by one. However, this two-step solution is sub-optimal in satisfying our objective, namely producing a minimal number of interpretable rules that are as simple as possible to summarize the outlier detection results. This is because the problems of data partitioning and rule generation are highly dependent on each other. Clearly, rule generation relies on data partitioning. To generate localized rules, the data has to be partitioned first. However, on the other hand, without taking the objective of rule generation into consideration, the clustering algorithm does not necessarily yield data partitions that are easy to summarize with simple thus interpretable rules. Therefore, L-STAIR solves the two sub-problems of data partitioning and rule generation jointly.

To achieve this goal, in addition to the summarization and interpretation-aware objective (Eq. 6) defined in Sec. 4.2, L-STAIR introduces a partitioning objective composed of *error objective* and *locality objective.*

Error Objective. We denote the partitions of a dataset as $C = \{C_i\}_{i=1}^n$, where n is the number of partitions and C_i represents the ith partition. DT_i denotes the decision tree learned for a data partition C_i. Decision tree DT_i produces a prediction with respect to each object x in data partition C_i, denoted as $DT_i(x)$.

Next, in Eq. 21 we define an *error metric* to measure how good a decision tree DT_i fits the data in C_i:

$$\sum_{x \in C_i} ||DT_i(x) - y_i||_2^2 \tag{21}$$

To ensure the classification accuracy, L-STAIR targets minimizing this error metric with respect to all data partitions, which yields the **error objective**:

$$\min_{C} \sum_{C_i \in C} \sum_{x \in C_i} ||DT_i(x) - y||_2^2 \tag{22}$$

where y indicates the ground truth label of object x.

Locality Objective. Although using the above error objective to learn the data partitioning and the corresponding decision trees will effectively minimize the overall classification with respect to the

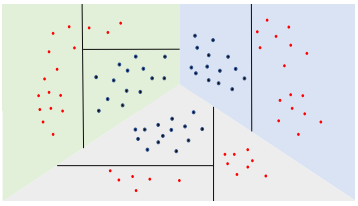


Figure 3: The intuition of locality: The black points are inliers and the red ones are outliers. The backgrounds with different colors refer to three clusters, while the straight lines in each cluster represent the rules.

whole dataset, the data partitions produced in this way do not preserve the locality of each data partition. Potentially one rule could cover a set of data objects that are scattered across the whole data space, thus is not amenable for humans to understand. As shown in Figure 3, when the locality is preserved, the rules are constrained within each cluster. This means there is no overlapping between the rules. Then the generated rules will be easier to understand.

Therefore, to ensure the data locality of each partition, we introduce the **locality objective**:

$$\min_{C} \sum_{C_i \in C} ||x - center(C_i)||_2^2 \tag{23}$$

Optimizing on the locality objective enforces the objects within each partition to be close to each other, similar to the objective of clustering such as k-means.

The Final L-STAIR Objective. Combining Eq. 23 and Eq. 22 together leads to the final partitioning objective:

$$\min_{C} \mathcal{L}_{L-STAIR}(C) = \sum_{C_i \in C} \sum_{x \in C_i} ||DT_i(x) - y||_2^2 + \lambda ||x - center(C_i)||_2^2 \tag{24}$$

Eq. 24 uses λ ($0 < \lambda < 1$) to balance these two objectives. Setting the λ to a small value will give error objective higher priority.

6.2 L-STAIR Learning Algorithm

L-STAIR generates interpretable and localized decision trees within distinct clusters. As shown in Algorithm 2, initially, the training data is divided into a predefined number of clusters using K-means clustering (line 1). The algorithm iteratively refines the clusters and builds decision trees for each cluster using the STAIR algorithm (Section 5.3) (line 3-9). The objective function (Eq. 24) guides the update of the clusters, preserving data locality within each partition (line 4). To enhance the interpretability, empty clusters are removed (line 5), and the overall model performance is assessed using the F_1 score(line 6). If the F_1 score surpasses a predefined threshold F_{1m}, it terminates immediately (line 8). Otherwise, clusters with insufficient F_1 scores undergo further refinement. That is, L-STAIR splits them into new clusters using the K-means algorithm (line 9). This iterative process continues until satisfying the desired interpretability threshold.

Algorithm 2: L-STAIR learning algorithm

Input: Training data X, cluster number n for initialization, F_1 score threshold F_{1m}
Output: Clusters C and decision tree for each cluster DT_i, $i \in \{1, \cdots, |C|\}$

1 Initialize n clusters using K-means;
2 **while** *True* **do**
3 Build n new decision trees DT_i, $i = \{1, \cdots, n\}$ using the algorithm introduced in Section 5.3 for n clusters;
4 Update the clusters C according to the objective Eq.(24);
5 Remove empty clusters;
6 Calculate the F_1 score of the predictions made by the MDTs(the rules), and denote it as f_1;
7 **if** $f_1 > F_{1m}$ **then**
8 break;
9 Check the F_1 score within each cluster, split each of the clusters with too small F_1 scores to n new clusters using K-means algorithm.
10 **end**

6.3 Dynamically Adjusting the Number of Partitions

As shown in Algorithm 2, L-STAIR uses the hyperparameter n to specify the number of partitions and initialize each data partition accordingly. It is well known that in many clustering algorithms such as k-means the number of clusters is a critical hyper-parameter that determines the quality of data partitioning, and it is hard to tune in many cases [31]. L-STAIR does not rely on an appropriate n to achieve good performance, because L-STAIR allows the users to set a small n initially and then dynamically adjusts it in the learning process.

Producing New Partitions. L-STAIR will produce new partitions by splitting some partitions that are too complicated to summarize and explain with simple rules. The partition is said to be too complicated when the obtained $F1$-score on it is not good enough, more specifically lower than $F1_m$. This indicates that simple rules could not fully explain this partition. After identifying a complicated partition, L-STAIR uses k-means again to split it into two partitions, and then build one decision tree for each new partition.

Removing Partitions. L-STAIR identifies the redundant partitions as those bearing large similarity to others such that merging them into other partitions does not degrade the partitioning objective. After identifying redundant partitions, L-STAIR will discard them and reassign their data points to other partitions.

Our experiments (Table 3, Sec. 7.5) on 10 datasets show that starting with a small n L-STAIR is always able to produce good results.

Table 1: Statistics of the 10 Datasets.

Dataset	# Instances	Outlier Fract.	# of Dims
PageBlock[3]	5473	10%	10
Pendigits[7]	6870	2.3%	16
Shuttle[6]	49097	7%	9
Pima[8]	768	35%	8
Mammography[1]	11873	2.3%	6
Satimage-2[2]	5803	1.2%	36
Satellite[9]	6435	32%	36
SpamBase[5]	4601	40%	57
Cover[4]	286048	0.9%	10
Thursday-01-03[10]	33110	28%	68

7 EXPERIMENTS

Our experimental study aims to answer the following questions:

- **Q1**: How do STAIR and L-STAIR compare against other methods in the total rule lengths given a F_1 threshold?
- **Q2**: How do STAIR and L-STAIR compare against other methods in F_1 score when producing rules with the similar complexity?
- **Q3**: How do the parameters L_m and $F1_m$ affect the performance of STAIR?
- **Q4**: How does the number of partition n affect the performance of L-STAIR?
- **Q5**: How good is L-STAIR at preserving the locality of the data?
- **Q6**: How does STAIR dynamically adjust the value of stabilizer M introduced in our summarization and interpretation-aware optimization objective?
- **Q7**: How does STAIR perform in multi-class classification?
- **Q8**: Are the rules indeed interpretable?

7.1 Experimental Settings

Datasets. We evaluate the effectiveness of STAIR and L-STAIR on ten benchmark outlier detection datasets. Table 1 summarizes their key statistics.

Hardware Settings. We implement our algorithm with Python 3.7. We use the decision tree algorithms in scikit-learn and implement STAIR with numpy. We train all models on AMD Ryzen Threadripper 3960X 24-Core Processor with 136GB RAM.

Baselines. We compare against seven rule-based methods:

- **ID3** [54]: The classic decision tree algorithm. To find the simplest decision tree that satisfies the accuracy threshold $F1_m$, we start with a small tree (depth 3) and iteratively increase its depth until the obtained tree could yield a $F1$ score larger than $F1_m$.
- **CART** [18]: CART uses post-processing to prune a learned decision tree. The goal is to minimize the complexity of the decision tree, while still preserving the accuracy. We first use ID3 to build a decision tree that is as accurate as possible and then continue to prune it until it is right above the F-1 score threshold.
- **RIPPERk** [24]: RIPPERk adopts a depth-first search to generate one rule from the dataset at each iteration. It greedily adds rule antecedents by comparing the information gain of each attribute and using the pruning technique to avoid overfitting. We continue running the algorithm until the obtained rules achieve an $F1$ score larger than $F1_m$.
- **CORELS** [13]: CORELS is an iterative method that takes a training dataset as input and produces a set of rules as output, which can be used to interpret the instances. We calculate the $F1$ score at each iteration until the $F1$ score is larger than the accuracy requirement $F1_m$.
- **FRL** [68]: FRL also takes a training dataset as input and learns a binary classification model consisting of a set of if-then-elseif rule lists. To be specific, in each rule, the "then" clauses corresponding to "elseif" clauses specify the probabilities of the outcome ("1"), which exhibit a monotonically decreasing trend. In this way, more important clauses are shown first. Similarly, we calculate the $F1$ score at each iteration until the $F1$ score exceeds $F1_m$.
- **Explanation Table** [34]: Explanation Table takes the original dataset as input and outputs an explanation table with the same attributes, where each row can be regarded as a rule to explain a binary attribute. To be specific, the rule contains the values of different attributes, and the value can be "$*$", indicating the corresponding attribute can take any values. We also calculate the $F1$ score at each iteration until the $F1$ score is larger than $F1m$. The number of non-"$*$" values in the obtained explanation table represents the total rule length.
- **HiCS** [39]: HiCS focuses on computing the contrasts of subsets of attributes, and the subset with the highest contrast is regarded as contributing the most to identifying the outliers. Unlike our work, HiCS does not produce rules to interpret and summarize the outliers. To be comparable to our work, we apply a decision tree algorithm (ID3) on the identified set of attributes to produce a tree that achieves a $F1$ score larger than $F1_m$.

Outlier Detection Algorithm. In the experiments we use LOF [21], a typical density-based method, as the outlier detection method.

Table 2: Total rule length under similar F_1 score (Q1).

Dataset	ID3	CART	FRL	RIPPERk	CORELS	Explanation Table	HiCS	STAIR	L-STAIR
PageBlock	97	88	88	89	67	91	105	50	**25**
Pendigits	290	328	303	270	257	330	390	187	**60**
Shuttle	1520	863	848	845	745	967	1800	697	**125**
Pima	20	12	15	18	14	16	38	12	**10**
Mammography	79	65	66	84	70	79	93	66	**24**
Satimage-2	151	117	125	128	110	132	163	93	**38**
Satellite	1263	471	545	897	461	980	1333	442	**70**
SpamBase	1546	1043	1086	1100	1088	1343	1616	1017	**150**
Cover	6616	4869	5124	5357	4689	5018	6787	4657	**402**
Thursday-01-03	4032	1393	2454	3864	2400	3135	4612	957	**440**

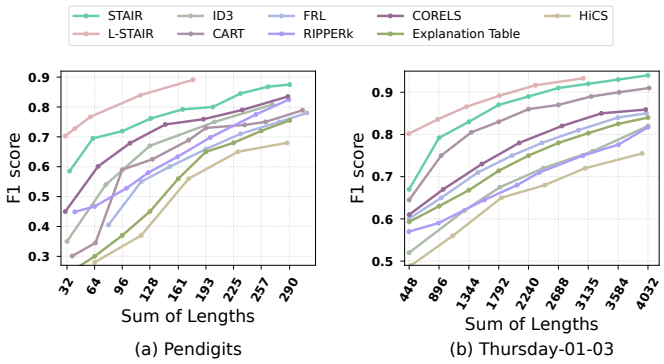
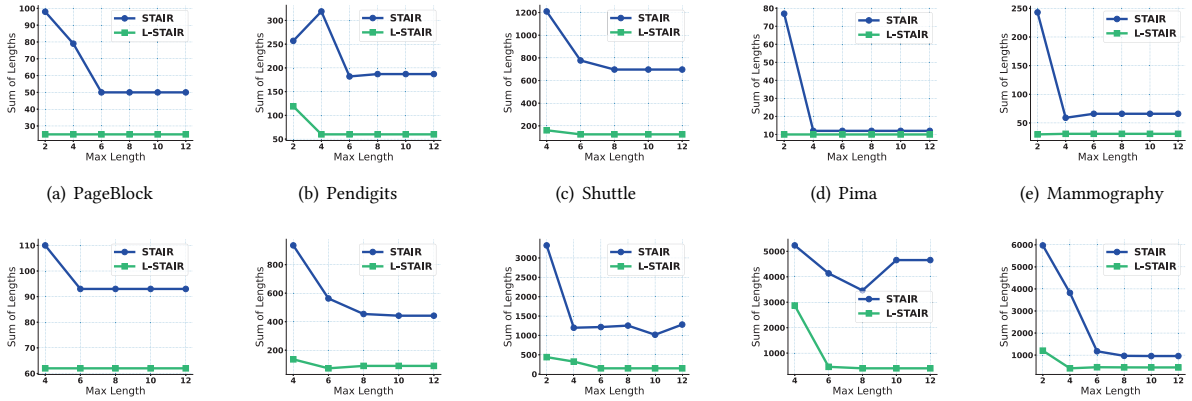


Figure 4: F_1 score with varying total rule lengths (Q2).

7.2 Comparison Against Baselines (Q1): Total Rule Length

We measure the total length of the rules produced by each algorithm when they achieve a similar F_1 score. We set the maximal length of the rules L_m to 10 and the F_1 score threshold $F1_m$ to 0.8. For the baselines that do not use the F_1 score threshold in their algorithms, we tune their hyper-parameters to produce the simplest tree with a F_1 score slightly higher than 0.8. This ensures that all algorithms have the **similar** F_1 score. L-STAIR automatically determines the number of data partitions with the initial partition number picked from {2, 4, 8}. We set the maximal iteration to 10.

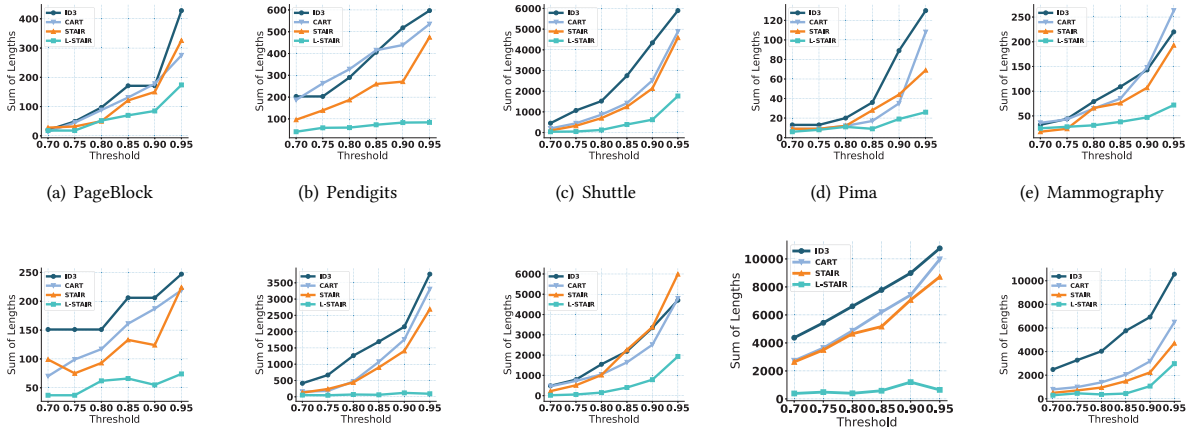
Based on the results shown in Table 2, we draw the following conclusions: (1) In comparison to all the baselines, STAIR produces



(a) PageBlock (b) Pendigits (c) Shuttle (d) Pima (e) Mammography

(f) Satimage-2 (g) Satellite (h) SpamBase (i) Cover (j) Thursday-01-03

Figure 5: The effects of the maximal length L_m on the total rule length (Q3).



(a) PageBlock (b) Pendigits (c) Shuttle (d) Pima (e) Mammography

(f) Satimage-2 (g) Satellite (h) SpamBase (i) cover (j) Thursday-01-03

Figure 6: The effect of the threshold $F1_m$ on all methods (Q3).

much simpler rules that are amenable for humans to evaluate, reducing the total length of the rules by up to 79.3% (the *Thursday-01-03* dataset). This is because the summarization and interpretation-aware optimization objective (Eq. 7) of STAIR simultaneously minimizes the complexity of the tree and maximizes the classification accuracy; (2) The performance of STAIR on the SpamBase dataset is not satisfying potentially due to its large dimensionality. SpamBase has 57 attributes. It thus might be too complex to use a single small tree to summarize the whole dataset; (3) L-STAIR which partitions the data and produces one tree for each data partition solves the problem mentioned in (2) and outperforms the basic STAIR by up to 91.37% on the dataset *Cover*.

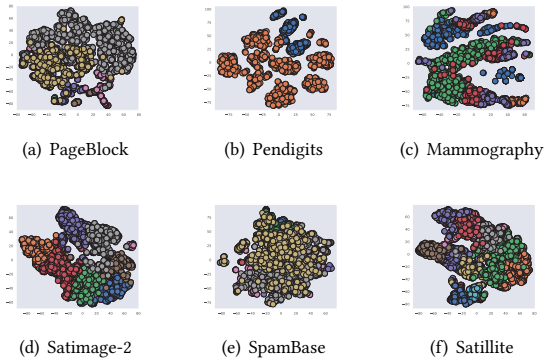
7.3 Comparison Against Baselines (Q2): F_1 Score

In this section, we evaluate the F_1 score of each algorithm when they produce a rule set with a similar total length. For each dataset, we vary the total rule length by selecting 10 numbers within a range from 0 to the total rule length result with respect to the ID3 algorithm in Table 2. For instance, in Table 2 the total rule length of ID3 on the dataset Pendigits is 290. We thus select ten numbers: 29, 58, $\cdots$, 290 as the candidate total lengths.

Subject to the constraint of total length l, we run the baselines and our methods in the following way to obtain the F_1 score: (1)

Table 3: The number of partitions n in L-STAIR (Q4).

	L-STAIR (n=2)			L-STAIR (n=4)			L-STAIR (n=8)		
Dataset	Length	# of R	# of C	Length	# of R	# of C	Length	# of R	# of C
PageBlock	31	26	9	25	20	8	67	31	8
Pendigits	60	20	2	86	37	4	68	39	8
Shuttle	125	52	9	309	112	10	357	108	9
Pima	10	6	2	17	12	4	27	20	8
Mammography	31	20	6	24	15	5	29	21	8
Satimage-2	62	27	4	38	19	4	49	31	8
Satellite	70	18	2	78	28	4	209	74	8
SpamBase	150	51	8	218	73	10	278	72	9
Cover	402	117	5	470	136	4	621	195	8
Thursday-01-03	477	183	11	479	182	11	440	169	11



(a) PageBlock (b) Pendigits (c) Mammography

(d) Satimage-2 (e) SpamBase (f) Satillite

Figure 7: Visualization of L-STAIR (Q5).

For ID3, we gradually increase the depth of the tree until it generates a rule set with a total length slightly higher than l; (2) For CART, we first build a decision tree that is as accurate as possible and then prune it until it has a length close to l; (3) For RIPPERk, CORELS, FRL, Explanation Table and HiCS, we run these algorithms continuously until the length of the rule sets generated by these algorithms reached l; (4) For STAIR/ L-STAIR, we update the breaking condition of Algorithm 1 / 2 such that it will terminate after reaching the length l. We run this experiment on the Pendigits and Thursday-01-03 datasets. As shown in Figure 4, STAIR is more accurate than other methods when they produce a rule set with a similar total length, indicating that given the same budget on the total rule length, STAIR produces rules with higher accuracy.

7.4 Effect of Hyper-parameters L_m and $F1_m$ (Q3)

In this set of experiments, we first study how the maximal length L_m affects STAIR and L-STAIR. We fix the F1 score threshold $F1_m$ as 80% and then vary L_m from 2 to 12 and measure how the total rule length changes. Note in some cases when L_m is too small, e.g. 2, the learned tree cannot meet the F1 score requirement. As shown in Figure 5, as L_m gets larger, the total rule length will get smaller. This is because with a looser constraint, STAIR gets a larger search space and hence a better chance to find a simple tree. When STAIR gets better, L-STAIR will also get better. Besides, we observe that L-STAIR could reach the minimal total rule length with smaller L_m. This shows the power and benefits of localization.

Next, we investigate how the F1 score threshold $F1_m$ affects STAIR and L-STAIR. We fix L_m to 10 and vary $F1_m$ from 0.70 to 0.95. As shown in Figure 6, in most of the cases STAIR outperforms ID3 and CART, while L-STAIR consistently outperforms all other methods in all scenarios by up to 94.0% as shown in the results on the *Cover* dataset when the threshold is set as 0.95. The larger the $F1_m$ threshold is, the more L-STAIR outperforms other baselines. This is because partitioning allows L-STAIR to get a set of localized trees, each of which produces high accurate classification results on the corresponding data subset.

Table 4: Multi-class classification: total rule length.

Dataset	ID3	CART	STAIR	L-STAIR
Wine Quality	5133	3217	2251	**1538**

7.5 Number of partitions in L-STAIR (Q4)

We study how the initial number of the partitions n affects L-STAIR. In this set of experiments, n is selected from {2, 4, 8}. In addition to the total rule length, we also report the number of rules in the final ruleset. From the results shown in Table 3, we have the following observations: (1) Compared to the results in Table 2, no matter what n L-STAIR starts with, it consistently outperforms other methods; (2) L-STAIR always performs well when starting with a small n compared to other initial n values, indicating that n is not a hyper-parameter that requires careful tuning.

Table 5: Multi-class classification: #-partitions n in L-STAIR

	L-STAIR (n=2)			L-STAIR (n=4)			L-STAIR (n=8)		
Dataset	Length	# of R	# of C	Length	# of R	# of C	Length	# of R	# of C
Wine Quality	1642	614	11	1692	620	13	1538	635	17

7.6 L-STAIR: Locality-Preserving (Q5)

We evaluate if the partitioning of L-STAIR is able to preserve the locality of the data. We show this by visualizing its data partitioning. Before visualization, We apply T-SNE to embed the data into 2D. We plot different partitions in different colors. Due to space limits, we only plot the partitioning of 6 datasets. As shown in Figure 7, on all datasets the partitioning of L-STAIR preserves the locality. This thus guarantees the interpretability of each localized tree.

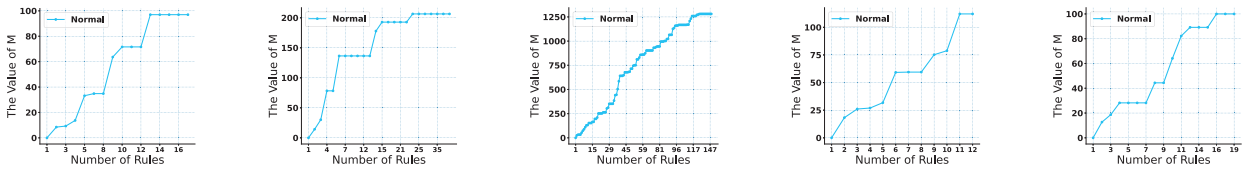
7.7 Dynamically Adjusting the Value of M (Q6)

In this set of experiment, we show how STAIR automatically adjusts the value of stabilizer M introduced in our summarization and interpretation-aware optimization objective (Sec. 4.2). To better understand the influence of a dynamically adjusting M, we use the number of rules produced in the training process as the reference variable, corresponding to the x-axis. From Figure 8, we observe: (1) The value of M continuously increases during the training process to split nodes and thus produce valid rules; (2) The values of M are different across different datasets, indicating that it is hard to get an appropriate M by manual tuning.

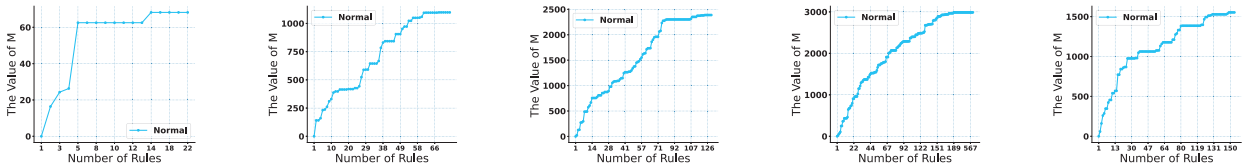
7.8 Multi-class classification Problems (Q7)

We use this set of experiments to show that STAIR and L-STAIR are generally applicable to the more complicated multi-class classification problems. We use one of the most popular classification datasets *Wine Quality*[2] [25], which contains 4898 instances and 12 attributes. We regard the attribute "score" as the target which corresponds to integers within the range from 0 to 10 and run a classification task on it. Our STAIR and L-STAIR could be easily

[2]https://archive.ics.uci.edu/ml/datasets/wine+quality

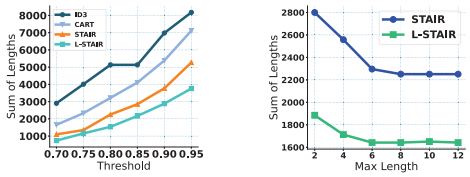


(a) PageBlock (b) Pendigits (c) Shuttle (d) Pima (e) Mammography

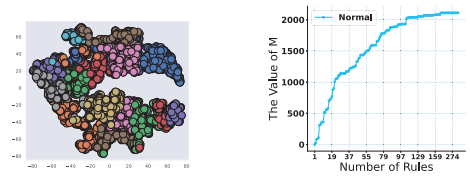


(f) Satimage-2 (g) Satellite (h) SpamBase (i) Cover (j) Thursday-01-03

Figure 8: The dynamic M during training (Q6).



(a) Different Thresholds (b) Different L_m



(c) Cluster Visualization (d) Increasing of M

Figure 9: Multi-class dataset *Wine Quality*: ablation study

extended to multi-class settings by replacing the F1 score with the *classification accuracy*.

As shown in Table 4, we report the total rule length, the same as the outlier detection scenario. We observe from the results: (1) L-STAIR and STAIR significantly outperform ID3 and Cart by up to 70.0%; (2) As shown in Table 5, the initial number n of the partitions make little difference to the resulted lengths, indicating L-STAIR is not sensitive to the hyper-parameter n; (3) As illustrated in Figure 9(a) and Figure 9(b), STAIR and L-STAIR always outperform the baselines no matter how the accuracy threshold and the maximal rule length threshold vary; (4) Figure 9(c) visualizes the partitions produced by L-STAIR. The locality of the data partitions is well-preserved; (5) As shown in Figure 9(d), the dynamic update of the value of stabilizer M is important in splitting the nodes and producing valid rules, similar to the case of outlier detection.

7.9 Case studies (Q8)

We conduct case studies to show the rules produced by STAIR are indeed interpretable. Table 6 shows the rules produced on Pima [8] and Cover [4]. Due to space constraints, we only show 6 rules with concise explanations. Please see our technical report [11] for more rules. In Pima, a patient with diabetes is considered as an outlier. The outlier algorithm is expected to capture these outliers based on some diagnostic measurements. Based on some medical common senses, the most important indications of diabetes are (1) fasting blood glucose; and (2) insulin levels. The example rules in Table 6 correctly reflect these common senses and thus well summarize the outliers. For example, Rule 1 represents the early Type-2 diabetes, which typically shows elevated fasting glucose, high insulin levels, and an increased BMI. Rule 2 corresponds to advanced Type-2 diabetes, showing elevated fasting glucose and low insulin concentrations.

The Cover [4] dataset classifies forest cover types in Northern Colorado's wilderness areas. Cache la Poudre's unique conditions make it an ideal habitat for four tree types: Ponderosa pine, cottonwood, willow, and Douglas-fir, labeled as outliers. The outlier algorithm, focused on cartographic variables, is expected to identify such outliers. Rules generated by STAIR effectively summarize outliers. For instance, Rule 4 correctly identifies cottonwood and willow as outliers due to their preference for low elevation and abundant water sources, while Rule 5 and Rule 6 distinguish areas suitable for Ponderosa pine, Douglas-fir, and willow based on distinct elevation and water source characteristics.

8 RELATED WORK

Outlier Summarization and Interpretation. To the best of our knowledge, the problem of summarizing and interpreting outlier detection results using human understandable rules has not been well exploited. Focused on a special type of outliers, Scorpion [69] targets a specific type of outliers in aggregation queries, explaining the outliers based on provenance. Objects that, when removed, significantly reduce the abnormality of an outlier are considered its

Table 6: Example rules.

ID	Example Rules of Pima Dataset [8]	Explanation
1	$Glucose > 150 \wedge Insulin > 120 \wedge BMI > 40.2$	Early Type-2 diabetes
2	$Glucose > 150 \wedge Insulin \leq 10 \wedge BMI > 40.2$	Advanced Type-2 diabetes
3	$Glucose > 150 \wedge Insulin \leq 10 \wedge BMI \leq 22.6 \wedge DiabetesPedigree > 0.875 \wedge Pregnancies > 2$	Type-1 diabetes
ID	**Example Rules of Cover Dataset [4]**	**Explanation**
4	$Dist\_Hydrology \leq 42 \wedge Elevation \leq 2231 \wedge Dist\_Roadways > 73 \wedge Aspect \leq 119.0$	Cottonwood, Willow
5	$60 < Dist\_Hydrology \leq 390 \wedge Elevation \leq 2231 \wedge Dist\_Roadways > 73 \wedge Aspect > 180.0$	Ponderosa pine, Douglas-fir
6	$42 < Dist\_Hydrology \leq 60 \wedge 2231 < Elevation \leq 2526 \wedge Hillshade\_9am > 206$	Willow

cause. Similarly, Cape [47] explains outliers in aggregation queries, using objects that counterbalance outliers as explanations. However, both Scorpion and Cape do not target summarizing outliers.

Macrobase [17] uses association rule mining to correlate outliers with external attributes, such as sensor location or occurrence time. However, this method relies on the presence of external attributes and only identifies potential outlier-causing attributes. In contrast, our STAIR produces rules that specify both the attributes and the conditions they must satisfy. Myrtakis et al.[49] evaluate outlier explanation methods like BEAM[50] and RefOut [40], which focus on explaining individual outliers rather than summarizing outliers.

LookOut [36] identifies attribute pairs contributing most to detected outliers, while HiCS [39] detects outliers in high-dimensional data by computing contrast for each subspace. However, these methods provide coarse-grained explanations, insufficient for users to quickly identify key factors behind detected outliers.

Interpretable AI. Some works [28, 53, 60] target on interpreting the machine learning models, such as LIME [57], Anchor [58], LORE [35], which provide explanations for individual predictions by learning local linear models around each prediction. However, the computational cost of generating explanations for each individual object limits the scalability of methods like LIME, particularly on large datasets. Some other methods [15, 35, 44, 58, 62] explain classification results in the similar fashion. Taking the explanations w.r.t all testing objects as input, Pedreschi et al. select a subset of the explanations to constitute a global explanation [52]. However, this work, yet to go through the peer review process, is not scalable to big dataset because it requires constructing the explanations for all testing objects. In addition, some techniques, including gradient-based [63, 66] and attention based [16] methods, focus on particular types of deep learning models, thus hard to be used in the outlier summarization scenario. Lakkaraju et al [42] proposed to build a prediction model that is more interpretable than deep learning models. Methods [48, 51, 67] have worked on augmenting the data to produce models with better interpretability. Instead of inventing new prediction models, our work focuses on explaining and summarizing the results produced by any outlier detection method. There also exist works explaining data management tasks using rules. For example, Singh et al. [64, 65] focus on utilizing general boolean formulas to automatically generate interpretable and concise entity matching rules. AIME [32, 33] focus on extracting rules from knowledge graphs considering the structural information.

Outlier Detection. Due to the importance of outlier detection, many unsupervised outlier detection methods have been proposed including the density based method LOF [21], the statistical-based Mahalanobis method [12], the distance-based methods [14, 41, 56], and Isolation Forest [43], which do not use any human-labeled data. As a crowdsourcing-based method, HOD [22] proposed to leverage human to improve the outlier detection performance in text data. It produces some questions which once answered by humans, could help verify the status of multiple outlier candidates returned by the unsupervised methods. However, the question-generation process of HOD is still time-consuming. In addition, instead of focusing on text data, STAIR is generally applicable to different types of data including numerical, categorical, and text data.

Decision Tree Algorithms. CART [20] proposed post-processing pruning for decision trees to prevent overfitting and improve generalization. However, unlike STAIR, which prioritizes simplicity and interpretability, CART's approach is less effective in minimizing rule complexity. Similarly, other decision tree algorithms [19, 38, 59] mostly prioritize classification accuracy over rule simplicity.

Error Summarization for Data Cleaning. In data cleaning, there exist various methods that use deep learning to clean the data. For example, Deng et al. [26, 27] present an innovative approach to iteratively detect mislabeled data instances by leveraging early loss signals, which achieves state-of-the-art performance on accuracy. Chai et al. [23] clean the data over a small coreset that can lead to competitive model performance over the full train data, which achieves a data-efficient training process and cost-effective cleaning efforts. Miao et al. pioneer the development of accuracy-controlled imputation acceleration mechanisms [45, 46, 70] which have the excellent ability to well deal with large-scale missing data. Although the above methods can achieve good results, but may lack interpretability, which can be regarded as a future work of STAIR.

9 CONCLUSION

This work targets reducing the human effort in evaluating outlier detection results by introducing STAIR, which employs an optimization objective for outlier summarization and utilizes an efficient learning algorithm. Experimental results demonstrate that STAIR effectively generates concise and interpretable rules, surpassing the complexity of rules produced by alternative rule-based methods.

ACKNOWLEDGMENTS

This paper is supported by NSFC(62102215) and CCF-Huawei Populus Grove Fund (CCF-HuaweiDB202306). Yuping Wang is supported by NSFC (U23A20297). Lei Cao is supported by NSF DBI-2327954.

REFERENCES

[1] 1993. Mammography. https://www.kaggle.com/datasets/kmader/mias-mammography.

[2] 1993. Satimage-2. https://odds.cs.stonybrook.edu/satimage-2-dataset/.

[3] 1995. PageBlock. https://archive.ics.uci.edu/dataset/78/page+blocks+classification.

[4] 1998. Covertype. https://archive.ics.uci.edu/dataset/31/covertype.

[5] 1999. Spambase. http://archive.ics.uci.edu/dataset/94/spambase.

[6] 2007. Shuttle. https://archive.ics.uci.edu/dataset/148/statlog+shuttle.

[7] 2014. Pendigits. https://datahub.io/machine-learning/pendigits#readme.

[8] 2016. Pima. https://www.dbs.ifi.lmu.de/research/outlier-evaluation/DAMI/semantic/Pima/Pima_35.html.

[9] 2017. Satellite. https://datahub.io/machine-learning/satellite#readme.

[10] 2018. Thursday-01-03. https://www.kaggle.com/datasets/karenp/original-network-traffic-thursday-01-03-2018-logs.

[11] 2023. https://github.com/baodaBBji/anonymous-Tech-Report/blob/main/Outlier_Tech_Report.pdf.

[12] Charu C. Aggarwal. 2017. *Outlier Analysis: Second Edition.* Springer.

[13] Elaine Angelino, Nicholas Larus-Stone, Daniel Alabi, Margo I. Seltzer, and Cynthia Rudin. 2017. Learning Certifiably Optimal Rule Lists for Categorical Data. *J. Mach. Learn. Res.* 18 (2017), 234:1–234:78. http://jmlr.org/papers/v18/17-716.html

[14] Fabrizio Angiulli and Clara Pizzuti. 2002. Fast Outlier Detection in High Dimensional Spaces. In *PKDD*. 15–26.

[15] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. 2015. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one* 10, 7 (2015), e0130140.

[16] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *ICLR*.

[17] Peter Bailis, Edward Gan, Samuel Madden, Deepak Narayanan, Kexin Rong, and Sahaana Suri. 2017. Macrobase: Prioritizing attention in fast data. In *Proceedings of the 2017 ACM International Conference on Management of Data*. ACM, 541–556.

[18] Mridula Batra and Rashmi Agrawal. 2018. Comparative Analysis of Decision Tree Algorithms. In *Nature Inspired Computing*, Bijaya Ketan Panigrahi, M. N. Hoda, Vinod Sharma, and Shivendra Goel (Eds.). Springer Singapore, Singapore, 31–36.

[19] David Biggs, Barry De Ville, and Ed Suen. 1991. A method of choosing multiway partitions for classification and decision trees. *Journal of applied statistics* 18, 1 (1991), 49–62.

[20] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. 1984. *Classification and Regression Trees.* Wadsworth and Brooks, Monterey, CA.

[21] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. 2000. LOF: Identifying Density-based Local Outliers. In *SIGMOD*. ACM, 93–104.

[22] Chengliang Chai, Lei Cao, Guoliang Li, Jian Li, Yuyu Luo, and Samuel Madden. 2020. Human-in-the-Loop Outlier Detection. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data* (Portland, OR, USA) *(SIGMOD '20)*. Association for Computing Machinery, New York, NY, USA, 19–33.

[23] Chengliang Chai, Jiabin Liu, Nan Tang, Ju Fan, Dongjing Miao, Jiayi Wang, Yuyu Luo, and Guoliang Li. 2023. GoodCore: Data-effective and Data-efficient Machine Learning through Coreset Selection over Incomplete Data. *Proc. ACM Manag. Data* 1, 2 (2023), 157:1–157:27. https://doi.org/10.1145/3589302

[24] William W Cohen. 1995. Fast effective rule induction. In *Machine learning proceedings 1995*. Elsevier, 115–123.

[25] Paulo Cortez, António Cerdeira, Fernando Almeida, Telmo Matos, and José Reis. 2009. Modeling wine preferences by data mining from physicochemical properties. *Decis. Support Syst.* 47, 4 (2009), 547–553.

[26] Yuhao Deng, Chengliang Chai, Lei Cao, Nan Tang, Ju Fan, Jiayi Wang, Ye Yuan, and Guoren Wang. 2024. MisDetect: Iterative Mislabel Detection using Early Loss. *Proc. VLDB Endow.* 17, 6 (2024), 1159–1172. https://doi.org/10.14778/3648160.3648161

[27] Yuhao Deng, Qiyan Deng, Chengliang Chai, Lei Cao, Nan Tang, Ju Fan, Jiayi Wang, Ye Yuan, and Guoren Wang. 2024. IDE: A System for Iterative Mislabel Detection. In *Companion of the 2024 International Conference on Management of Data, SIGMOD/PODS 2024, Santiago, Chile, June 9-15, 2024*. ACM.

[28] Jack Dunn, Luca Mingardi, and Ying Daisy Zhuo. 2021. Comparing interpretability and explainability for feature selection. *CoRR* abs/2105.05328 (2021).

[29] Halima Elaidi, Zahra Benabbou, and Hassan Abbar. 2018. A comparative study of algorithms constructing decision trees: ID3 and C4.5. In *Proceedings of the International Conference on Learning and Optimization Algorithms: Theory and Applications, LOPAL 2018, Rabat, Morocco, May 2-5, 2018*. 26:1–26:5.

[30] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise.. In *Kdd*, Vol. 96. 226–231.

[31] Wei Fu and Patrick O Perry. 2020. Estimating the number of clusters using cross-validation. *Journal of Computational and Graphical Statistics* 29, 1 (2020), 162–173.

[32] Luis Galárraga, Christina Teflioudi, Katja Hose, and Fabian M. Suchanek. 2015. Fast rule mining in ontological knowledge bases with AMIE+. *VLDB J.* 24, 6 (2015), 707–730. https://doi.org/10.1007/S00778-015-0394-1

[33] Luis Antonio Galárraga, Christina Teflioudi, Katja Hose, and Fabian M. Suchanek. 2013. AMIE: association rule mining under incomplete evidence in ontological knowledge bases. In *22nd International World Wide Web Conference, WWW '13, Rio de Janeiro, Brazil, May 13-17, 2013*, Daniel Schwabe, Virgílio A. F. Almeida, Hartmut Glaser, Ricardo Baeza-Yates, and Sue B. Moon (Eds.). International World Wide Web Conferences Steering Committee / ACM, 413–422. https://doi.org/10.1145/2488388.2488425

[34] Kareem El Gebaly, Parag Agrawal, Lukasz Golab, Flip Korn, and Divesh Srivastava. 2014. Interpretable and Informative Explanations of Outcomes. *Proc. VLDB Endow.* 8, 1 (2014), 61–72. https://doi.org/10.14778/2735461.2735467

[35] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Dino Pedreschi, Franco Turini, and Fosca Giannotti. 2018. Local Rule-Based Explanations of Black Box Decision Systems. *CoRR* abs/1805.10820 (2018).

[36] Nikhil Gupta, Dhivya Eswaran, Neil Shah, Leman Akoglu, and Christos Faloutsos. 2018. Beyond Outlier Detection: LookOut for Pictorial Explanation. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2018, Dublin, Ireland, September 10-14, 2018, Proceedings, Part I (Lecture Notes in Computer Science, Vol. 11051)*, Michele Berlingerio, Francesco Bonchi, Thomas Gärtner, Neil Hurley, and Georgiana Ifrim (Eds.). Springer, 122–138.

[37] T. Kanungo, D.M. Mount, N.S. Netanyahu, C.D. Piatko, R. Silverman, and A.Y. Wu. 2002. An efficient k-means clustering algorithm: analysis and implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24, 7 (2002), 881–892.

[38] Gordon V Kass. 1980. An exploratory technique for investigating large quantities of categorical data. *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 29, 2 (1980), 119–127.

[39] Fabian Keller, Emmanuel Müller, and Klemens Böhm. 2012. HiCS: High Contrast Subspaces for Density-Based Outlier Ranking. In *IEEE 28th International Conference on Data Engineering (ICDE 2012), Washington, DC, USA (Arlington, Virginia), 1-5 April, 2012*, Anastasios Kementsietsidis and Marcos Antonio Vaz Salles (Eds.). IEEE Computer Society, 1037–1048.

[40] Fabian Keller, Emmanuel Müller, Andreas Wixler, and Klemens Böhm. 2013. Flexible and adaptive subspace search for outlier analysis. In *22nd ACM International Conference on Information and Knowledge Management, CIKM'13, San Francisco, CA, USA, October 27 - November 1, 2013*. ACM, 1381–1390.

[41] Edwin M. Knorr and Raymond T. Ng. 1999. Finding Intensional Knowledge of Distance-Based Outliers. In *In VLDB*. 211–222.

[42] Himabindu Lakkaraju, Stephen H. Bach, and Jure Leskovec. 2016. Interpretable Decision Sets: A Joint Framework for Description and Prediction. In *KDD*. ACM, 1675–1684.

[43] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2008. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*. IEEE, 413–422.

[44] Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. *Advances in neural information processing systems* 30 (2017).

[45] Xiaoye Miao, Yangyang Wu, Lu Chen, Yunjun Gao, Jun Wang, and Jianwei Yin. 2021. Efficient and Effective Data Imputation with Influence Functions. *Proc. VLDB Endow.* 15, 3 (2021), 624–632. https://doi.org/10.14778/3494124.3494143

[46] Xiaoye Miao, Yangyang Wu, Lu Chen, Yunjun Gao, and Jianwei Yin. 2023. An Experimental Survey of Missing Data Imputation Algorithms. *IEEE Trans. Knowl. Data Eng.* 35, 7 (2023), 6630–6650. https://doi.org/10.1109/TKDE.2022.3186498

[47] Zhengjie Miao, Qitian Zeng, Chenjie Li, Boris Glavic, Oliver Kennedy, and Sudeepa Roy. 2019. CAPE: Explaining Outliers by Counterbalancing. *Proc. VLDB Endow.* 12, 12 (2019), 1806–1809.

[48] Yao Ming, Huamin Qu, and Enrico Bertini. 2019. RuleMatrix: Visualizing and Understanding Classifiers with Rules. *IEEE Trans. Vis. Comput. Graph.* 25, 1 (2019), 342–352.

[49] Nikolaos Myrtakis, Vassilis Christophides, and Eric Simon. 2021. A Comparative Evaluation of Anomaly Explanation Algorithms. In *Proceedings of the 24th International Conference on Extending Database Technology, EDBT 2021, Nicosia, Cyprus, March 23 - 26, 2021*. OpenProceedings.org, 97–108.

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

[51] Görkem Paçaci, David Johnson, Steve McKeever, and Andreas Hamfelt. 2019. "Why Did You Do That?" - Explaining Black Box Models with Inductive Synthesis. In *ICCS (5) (Lecture Notes in Computer Science, Vol. 11540)*. Springer, 334–345.

[52] Dino Pedreschi, Fosca Giannotti, Riccardo Guidotti, Anna Monreale, Luca Pappalardo, Salvatore Ruggieri, and Franco Turini. 2018. Open the Black Box Data-Driven Explanation of Black Box Decision Systems. *CoRR* abs/1806.09936 (2018).

[53] P Jonathon Phillips, Carina A Hahn, Peter C Fontana, David A Broniatowski, and Mark A Przybocki. 2020. Four principles of explainable artificial intelligence. *Gaithersburg, Maryland* (2020).

[54] J. Ross Quinlan. 1986. Induction of Decision Trees. *Mach. Learn.* 1, 1 (1986), 81–106.

[55] J. Ross Quinlan. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufmann.

[56] Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim. 2000. Efficient algorithms for mining outliers from large data sets. In *ACM SIGMOD Record*, Vol. 29. ACM, 427–438.

[57] Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *KDD*. ACM, 1135–1144.

[58] Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Anchors: High-Precision Model-Agnostic Explanations. In *AAAI*. AAAI Press, 1527–1535.

[59] Gilbert Ritschard. 2013. CHAID and earlier supervised tree methods. In *Contemporary issues in exploratory data mining in the behavioral sciences*. Routledge, 70–96.

[60] Cynthia Rudin. 2019. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature machine intelligence* 1, 5 (2019), 206–215.

[61] S.R. Safavian and D. Landgrebe. 1991. A survey of decision tree classifier methodology. *IEEE Transactions on Systems, Man, and Cybernetics* 21, 3 (1991), 660–674. https://doi.org/10.1109/21.97458

[62] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. 2017. Learning Important Features Through Propagating Activation Differences. In *ICML (Proceedings of Machine Learning Research, Vol. 70)*. PMLR, 3145–3153.

[63] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2014. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. In *ICLR (Workshop Poster)*.

[64] Rohit Singh, Venkata Vamsikrishna Meduri, Ahmed K. Elmagarmid, Samuel Madden, Paolo Papotti, Jorge-Arnulfo Quiané-Ruiz, Armando Solar-Lezama, and Nan Tang. 2017. Generating Concise Entity Matching Rules. In *Proceedings of the 2017 ACM International Conference on Management of Data, SIGMOD Conference 2017, Chicago, IL, USA, May 14-19, 2017*, Semih Salihoglu, Wenchao Zhou, Rada Chirkova, Jun Yang, and Dan Suciu (Eds.). ACM, 1635–1638. https://doi.org/10.1145/3035918.3058739

[65] Rohit Singh, Venkata Vamsikrishna Meduri, Ahmed K. Elmagarmid, Samuel Madden, Paolo Papotti, Jorge-Arnulfo Quiané-Ruiz, Armando Solar-Lezama, and Nan Tang. 2017. Synthesizing Entity Matching Rules by Examples. *Proc. VLDB Endow.* 11, 2 (2017), 189–202. https://doi.org/10.14778/3149193.3149199

[66] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic Attribution for Deep Networks. In *ICML (Proceedings of Machine Learning Research, Vol. 70)*. PMLR, 3319–3328.

[67] Madhumita Sushil, Simon Suster, and Walter Daelemans. 2018. Rule induction for global explanation of trained models. In *BlackboxNLP@EMNLP*. Association for Computational Linguistics, 82–97.

[68] Fulton Wang and Cynthia Rudin. 2015. Falling Rule Lists. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2015, San Diego, California, USA, May 9-12, 2015 (JMLR Workshop and Conference Proceedings, Vol. 38)*, Guy Lebanon and S. V. N. Vishwanathan (Eds.). JMLR.org. http://proceedings.mlr.press/v38/wang15a.html

[69] Eugene Wu and Samuel Madden. 2013. Scorpion: Explaining Away Outliers in Aggregate Queries. *Proc. VLDB Endow.* 6, 8 (2013), 553–564.

[70] Yangyang Wu, Jun Wang, Xiaoye Miao, Wenjia Wang, and Jianwei Yin. 2024. Differentiable and Scalable Generative Adversarial Models for Data Imputation. *IEEE Trans. Knowl. Data Eng.* 36, 2 (2024), 490–503. https://doi.org/10.1109/TKDE.2023.3293129