Journal of Intelligent Information Systems
https://doi.org/10.1007/s10844-025-00961-5

RESEARCH

Towards scalable action rule discovery: a structured vertical partitioning method

Aileen Benedict[1] · **Zbigniew W. Ras**[1,2]

Received: 2 December 2024 / Revised: 10 June 2025 / Accepted: 13 June 2025

Abstract

Purpose: The extraction of actionable insights is critical for intelligent systems and recommendation engines. However, traditional methods for action rule discovery face challenges in scalability and efficiency when applied to large datasets. This study introduces a correlation-based vertical partitioning method to improve the consistency and interpretability of action rules while addressing the limitations of random partitioning and unstructured approaches. **Methods:** The proposed method clusters flexible attributes using correlations, enabling structured partitions for parallel rule generation via hierarchical clustering. Comparative experiments evaluated its precision, runtime, lightness, and coverage against random and baseline partitioning approaches. **Results:** The correlation-based method outperformed random partitioning and significantly improved runtime efficiency over the baseline. It generates interpretable rules in a single iteration, avoiding variability and repeated runs, though challenges in rule combination efficiency suggest areas for improvement. **Conclusion:** The correlation-based vertical partitioning method strikes a balance between computational efficiency and rule quality, making it a promising solution for large-scale action rule discovery. Future work could enhance scalability further by improving the rule combination process and exploring hybrid or adaptive partitioning strategies to extend the method's applicability across diverse domains.

Keywords Action rules · Action rule discovery · Recommendation systems · Dataset partitioning · Distributed computing

1 Introduction

The extraction of actionable insights is essential for data-driven intelligent systems and recommendation engines, which rely on precise, targeted information to support decision-

✉ Zbigniew W. Ras
ras@charlotte.edu

Aileen Benedict
abenedi3@charlotte.edu

1 Computer Science Department, University of North Carolina at Charlotte, 9201 University City Blvd., Charlotte 28223, NC, USA

2 Institute of Computer Science, Polish-Japanese Academy of Information Technology, Koszykowa, 86, 02-008 Warsaw, Poland

Published online: 20 June 2025

making. Action rule discovery, introduced by Ras and Wieczorkowska (2000), is a pivotal technique in this process. Action rules identify transformations in flexible attributes (changeable attributes) that can lead to a desired change in a decision attribute, while stable attributes remain constant.

For example, consider an action rule aimed at enhancing customer satisfaction in a service-oriented context:

$$\Big[(\text{Service\_Type}, \text{Online}) \wedge (\text{Response\_Time}, \text{Average} \rightarrow \text{Fast})\Big] \Rightarrow (\text{Customer\_Satisfaction}, \text{Low} \rightarrow \text{High}), \quad (1)$$

This rule suggests that for customers with a stable attribute such as Service Type being Online, adjusting the flexible attribute Response Time from Average to Fast is likely to improve Customer Satisfaction from Low to High, the desired outcome. By uncovering such patterns, action rules enable targeted interventions, making them invaluable in domains where actionable, data-driven strategies are critical.

With the advent of distributed computing, there is growing interest in scalable solutions for action rule discovery to handle increasing data volumes. Traditional centralized approaches often face computational bottlenecks when analyzing vast datasets with complex attribute relationships. Distributed methods mitigate these constraints by enabling parallelized rule discovery. This study focuses on enhancing the efficiency and quality of distributed action rule discovery through vertical partitioning.

Previous research has proposed distributed algorithms like the MR-Random Forest algorithm for action rule discovery, which leverages the MapReduce framework and Random Forests to process large datasets efficiently. Such approaches highlight the benefits of combining ensemble learning with distributed computing for faster and more reliable action rule discovery across multiple nodes (Tzacheva et al., 2016). Vertical partitioning, a technique that divides data by columns, has been adopted in database optimization to improve performance by enhancing data access patterns. In action rule discovery, vertical partitioning enables focused and manageable rule generation, leading to faster processing and improved precision. However, existing vertical partitioning methods, such as random grouping of attributes (Tarnowska et al., 2022), introduce inconsistencies, often requiring multiple iterations to achieve stable results.

This study proposes a novel approach to distributed action rule discovery using correlation-based vertical partitioning. Unlike random splits, our method leverages attribute correlations to create structured partitions, ensuring related attributes are grouped together. This approach improves the consistency and relevance of generated rules by tailoring partitions to reflect attribute relationships. The result is a method that balances computational efficiency with high-quality, stable rule sets, addressing the limitations of existing strategies.

The contributions of this work are threefold:

1. We introduce a correlation-based vertical partitioning method that overcomes the inconsistencies of random partitioning by clustering attributes based on their correlations.
2. We implement this method using the Python Ray parallel processing framework, enabling efficient execution on large datasets.
3. We conduct a comparative analysis against traditional random partitioning and a baseline method without partitioning, evaluating performance metrics such as precision, coverage, lightness, and processing time.

This journal paper extends our prior work in two significant stages. Our earliest study (Benedict & Ras, 2024a) introduced the correlation-based vertical partitioning method and

provided preliminary experimental validation using RSES-based action rule generation. While promising, that version included limited methodological detail and early-stage results. We then refined the approach in a subsequent conference paper (Benedict & Ras, 2024b) presented at ISMIS 2024, where we improved the experimental design and presented more structured results. However, space constraints limited the depth of explanation, particularly around formal definitions and interpretation of results.

This journal article builds upon both works by significantly expanding the methodological detail, improving clarity, and deepening the experimental discussion. Key enhancements include:

1. Expanded Theoretical Foundations: We provide additional background on action rules, including examples and a formal definition with consistent notation.
2. Detailed Methodology Explanation: We offer a comprehensive description of our correlation-based vertical partitioning approach, supported by step-by-step pseudocode and section-level cross-references (Algorithm 1).
3. Comprehensive Evaluation Metrics: We include a detailed breakdown of the evaluation metrics used and their relevance to action rule discovery, improving result interpretability.
4. Refined Experimental Analysis: We extend our discussion of experimental results, exploring the nuances between different partitioning strategies and rule generation methods, with emphasis on rule consistency, computational efficiency, and practical impact.

By incorporating these improvements, this journal article provides a more rigorous and comprehensive examination of correlation-based vertical partitioning for action rule discovery, offering deeper insight into its advantages, limitations, and potential applications.

In the following sections, we provide an overview of existing approaches to action rule discovery, detail our methodology for correlation-based vertical partitioning, and discuss the results of our experiments. Our findings demonstrate that structured attribute partitioning based on correlations enhances both the efficiency and quality of action rule discovery, offering valuable insights for the development of intelligent systems and recommendation engines.

2 Background

2.1 Action rules and their applications

The extraction of actionable insights through action rules can be an essential step for data-driven intelligent systems and recommendation engines, enabling effective, informed decision-making. Action rules, introduced by Ras and Wieczorkowska (2000), provide a structured approach for identifying attribute transformations that lead to specific desired outcomes. By mapping out the necessary changes in attributes, action rules support intervention strategies across various domains, from improving customer experiences to refining therapeutic outcomes in healthcare.

2.2 Formal definition of an action rule

An action rule describes a transformation of flexible attributes that leads to a desired change in a decision attribute while stable attributes remain unchanged. Formally, an action rule r is

defined as:

$$r = \Big[(A_1 = a_1) \wedge (A_2 = a_2) \wedge \cdots \wedge (A_k = a_k) \\ \wedge (B_1, b_1 \rightarrow b_1') \wedge \cdots \wedge (B_m, b_m \rightarrow b_m')\Big] \Rightarrow (D, d \rightarrow d') \quad (2)$$

where:

- $A_1, A_2, \ldots, A_k$ are stable attributes (do not change).
- $B_1, B_2, \ldots, B_m$ are flexible attributes (modifiable to achieve a desired outcome).
- d and d' are the initial and desired values of the decision attribute D.
- The notation $(B_i, b_i \rightarrow b_i')$ represents a modification of the flexible attribute B_i from b_i to b_i', which contributes to the transition from d to d'.

For an action rule to be valid, it must satisfy support and confidence thresholds, ensuring that the transformation is statistically meaningful. This formalization aligns with the foundational work of Ras and Wieczorkowska (2000) and subsequent developments in action rule discovery.

In the context of action rules, support is defined as the number of distinct objects in the dataset that satisfy both the condition part (i.e., attribute transitions) and the decision transition of the rule. It is the number of rows that match "both the antecedent and consequent of a rule" (Sýkora & Kliegr, 2020). Unlike traditional association rule mining where support is often expressed as a proportion, we follow the convention of using absolute counts, as established in Ras and Wieczorkowska (2000). For example, a support value of 2 indicates that the action rule holds for two specific instances in the dataset. Confidence reflects the likelihood that the desired decision change occurs when the specified attribute transitions are applied. It is computed as the ratio between the number of records that satisfy both the antecedent and the decision change, and the number of records that satisfy only the antecedent (Sýkora & Kliegr, 2020) This formulation quantifies how reliably the suggested actions lead to the desired outcome, as originally described in Ras and Wieczorkowska (2000).

To illustrate an action rule, consider a business scenario where a company aims to improve customer satisfaction. Suppose *customer segment* and *product category* are stable attributes—these remain constant and define the context in which changes can be made. Meanwhile, attributes like *customer satisfaction level* and *product pricing* are flexible, meaning they can be adjusted to influence customer behavior.

For instance, an action rule might indicate that if a customer belongs to a specific segment and purchases a particular product category, then reducing the response time from *average* to *fast* and offering a discount could improve their satisfaction from *low* to *high*. Formally, this can be expressed as:

$$r = [[a_1 \wedge g_2 \wedge (B, b_1 \rightarrow b_2) \wedge (H, h_1 \rightarrow h_2)] \Rightarrow (D, d_1 \rightarrow d_2)], \quad (3)$$

where a_1 and g_2 represent the stable attributes (*customer segment* and *product category*), while $(B, b_1 \rightarrow b_2)$ and $(H, h_1 \rightarrow h_2)$ denote flexible attributes (*response time* and *discount level*) that can be adjusted to achieve the desired improvement in the decision attribute, $(D, d_1 \rightarrow d_2)$, representing *customer satisfaction* moving from *low* to *high*.

Action rules form the foundation of knowledge-based recommendation systems and have seen applications across various fields, including business, healthcare, education, music, and art (Tarnowska & Ras, 2021; Ras, 2022; Duan & Ras, 2022; Chatterjee et al., 2024; Ras & Dardzinska, 2011; Powell et al., 2021). In these domains, action rules enable targeted interventions that optimize outcomes based on data insights. For instance, in customer rela-

tionship management, an action rule may indicate how changes in response time can elevate customer satisfaction levels, supporting businesses in retaining a loyal customer base.

2.3 Action rule discovery approaches

Action rule discovery encompasses various approaches to uncovering attribute transformations that can yield desired outcomes. These discovery methods are primarily categorized into rule-based and object-based approaches. Rule-based approaches involve a two-step process: first, pattern detection through learning techniques that produce rules or clusters, followed by automated methods that derive action strategies from these patterns. These strategies indicate how to modify attribute values to reach a specific object classification. In contrast, object-based approaches extract actionable patterns directly from databases. Notable examples include the DEAR and DEAR2 systems (Tsay & Ras, 2003, 2005), which were early implementations of rule-based action rule extraction techniques.

The development of rule discovery methods has progressed further with foundational work by He et al. (2005) introducing a methodology for mining action rules from scratch. This approach formalized the action rule mining process without relying on pre-existing classification rules, ensuring completeness and correctness in discovered action rules under a support-confidence-cost framework. The authors' methodology provided significant improvements in rule discovery efficiency through a modified Apriori algorithm, allowing for a more comprehensive and targeted rule search. This advancement is particularly relevant for applications where decision-making objectives, such as optimizing profitability in customer relationship management, require a high degree of rule specificity.

Further advancements include the introduction of behavioral rules by Su et al. (2012), which emphasize not only actionable insights but also the probabilities of behavioral changes associated with these actions. Their framework introduces algorithms to predict behavior changes, adding an expected utility dimension that enhances the applicability of action rules in decision support systems. This framework is especially valuable in domains where influencing behavior is crucial, such as public policy or healthcare.

2.4 Distributed action rule discovery

The increasing volume of data in modern intelligent systems presents significant computational challenges for action rule extraction, particularly when dealing with large-scale datasets. Traditional methods often struggle with the resource demands of such data, necessitating distributed approaches to handle the complexities effectively. Distributed methods for action rule discovery aim to partition data in ways that optimize computational efficiency while maintaining the accuracy and relevance of the extracted rules.

One approach to handling large datasets is vertical partitioning, a well-established database design technique that divides tables into smaller segments based on attribute usage patterns. This concept, pioneered by Navathe et al. (1984), optimizes database performance by minimizing disk I/O operations, enhancing transaction processing efficiency. While originally designed for relational databases, the principles of vertical partitioning have been extended to action rule generation in large datasets, where attribute usage varies significantly.

In the context of action rule discovery, distributed methods like those proposed by Bagavathi et al. (2018) emphasize a hybrid approach, involving both horizontal and vertical partitioning. Their method partitions data into granules, handling rows (horizontal) and attributes (vertical) separately, with rules extracted from pairwise disjoint granules.

Tarnowska et al. (2022) further expanded this distributed approach by implementing random vertical partitioning using Spark, achieving reductions in processing time. However, the random partitioning method introduces variability, often requiring multiple iterations for consistent results.

Another study introduces the MR-Random Forest algorithm, designed to extract action rules in a distributed computing environment, addressing the challenges of scalability and efficiency (Tzacheva et al., 2016). The proposed approach leverages a combination of the MapReduce Framework, the Random Forest algorithm, and Action Rule Discovery methods like AroGS (Action Rule Discovery Based on Grabbing Strategy) and AAR (Association Action Rules). The use of the MapReduce framework ensured adaptability to large datasets.

In this study, we propose an alternative to random partitioning by using correlation-based clustering to achieve structured vertical partitions. By clustering flexible attributes based on their correlations, our method offers a more efficient and consistent approach to rule extraction, circumventing the need for multiple random runs. The clustering step provides a structured grouping that reduces variability in rule generation and enhances rule relevance, making it particularly suitable for large-scale data environments.

2.5 Other information systems

Action rule discovery is closely related to several research areas within information systems, particularly those that focus on extracting meaningful patterns from structured data. Here, we highlight three neighboring research endeavors: association rule mining, declarative process mining, and specification mining.

Association Rule Mining

Association rule mining is a fundamental technique in data mining that identifies relationships between itemsets within large datasets (Agrawal et al., 1993; Zhao & Bhowmick, 2003). Originally introduced for market basket analysis, it has since found applications across various domains, including healthcare, finance, and recommendation systems. The primary goal is to uncover frequent itemsets and generate association rules that meet predefined support and confidence thresholds, providing actionable insights into co-occurring patterns in data (Zhang & Wu, 2011). The Apriori algorithm remains a widely used method for association rule mining, leveraging a bottom-up approach to efficiently discover frequent itemsets and generate high-confidence rules (Borgelt & Kruse, 2002; Agrawal et al., 1993). Additionally, interestingness measures play a crucial role in ranking and filtering mined rules to improve their relevance and usability (Geng & Hamilton, 2006). These measures, which include confidence, lift, and conviction, help distinguish significant associations from coincidental correlations, making association rule mining an essential tool in knowledge discovery and decision support systems.

Declarative Process Mining in Information Systems

Declarative process mining is an approach to process discovery that focuses on identifying flexible, constraint-based representations of processes rather than rigid procedural workflows. Unlike traditional process discovery techniques that generate structured models (e.g., Petri nets or BPMN diagrams), declarative models emphasize behavioral constraints that must be satisfied while allowing for variability in execution paths (Maggi et al., 2018).

One of the key advantages of declarative process models is their suitability for processes operating in dynamic or unpredictable environments. Declarative models specify what is

required rather than how tasks must be performed, making them particularly useful for domains where procedural workflows may become overly complex (Leno et al., 2020). For example, in healthcare settings, patient treatment processes often exhibit high variability depending on individual cases, making a declarative approach more practical than enforcing strict procedural sequences.

A widely used framework for declarative process modeling is Declare, which employs logic-based constraints to define process behavior (Cecconi et al., 2022). Declare models specify rules that must be adhered to during process execution, such as precedence constraints (e.g., "X must occur before Y") or response constraints (e.g., "If A occurs, then B must eventually occur"). These constraints can be discovered automatically from event logs using declarative process mining techniques.

Recent research has focused on enhancing declarative process discovery with correlation-based techniques, which incorporate data-driven conditions into the constraints (Leno et al., 2020). Instead of purely control-flow-based constraints, these approaches analyze how process activities are influenced by data attributes, leading to more informative models. Additionally, efforts have been made to improve the efficiency of declarative process discovery algorithms by employing parallelization techniques, such as search space partitioning and database partitioning, to scale analysis to large datasets (Maggi et al., 2018).

Despite its advantages, declarative process mining faces challenges related to rule selection and interpretability. Unlike procedural models that offer clear visual representations of workflows, declarative models require domain experts to interpret constraints effectively. Furthermore, recent research has highlighted the need for better interestingness measures to assess the relevance of discovered rules and distinguish meaningful patterns from coincidental correlations (Cecconi et al., 2022).

Overall, declarative process mining offers a flexible and scalable alternative to traditional process discovery methods, making it an essential tool for analyzing complex and dynamic workflows in various domains.

Specification Mining

Specification mining infers formal specifications of software behavior from execution traces, aiding in verification, testing, and maintenance (Ammons et al., 2002). It is particularly useful when manually written specifications are incomplete or unavailable. A key approach is temporal specification mining, which extracts constraints in Linear Temporal Logic (LTL) to capture event ordering and system behaviors over time (Lemieux et al., 2015). Tools like Texada mine general LTL properties from logs, identifying cause-effect relationships and required execution sequences.

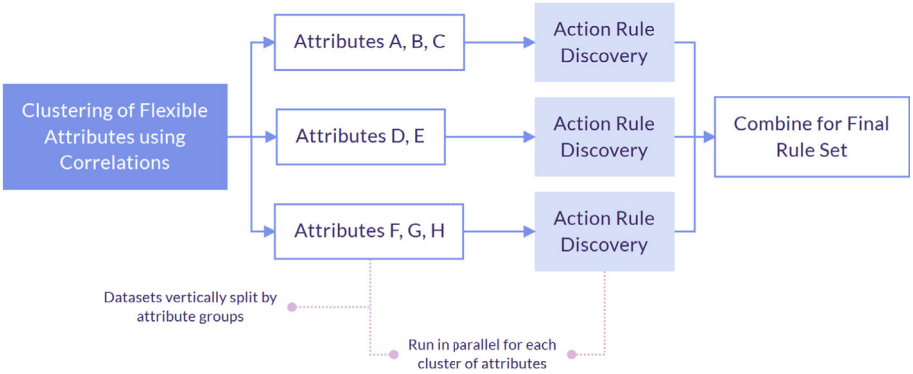
Dynamic specification mining observes program execution to infer API protocols (Yang et al., 2006). For instance, Perracotta detects temporal API usage rules even in noisy or incomplete traces, while state machine inference learns automata to model valid API behaviors (Ammons et al., 2002). These techniques help detect violations, such as improper resource handling in software systems. Challenges include scalability, trace completeness, and interpretability. Static analysis offers broad coverage but may introduce imprecision, whereas dynamic approaches yield precise insights but depend on trace quality. Hybrid methods combining both have been explored to improve precision and generalizability. As software grows in complexity, automated specification mining continues to evolve, enhancing software reliability by reducing the manual effort needed for formal verification.

3 Methodology

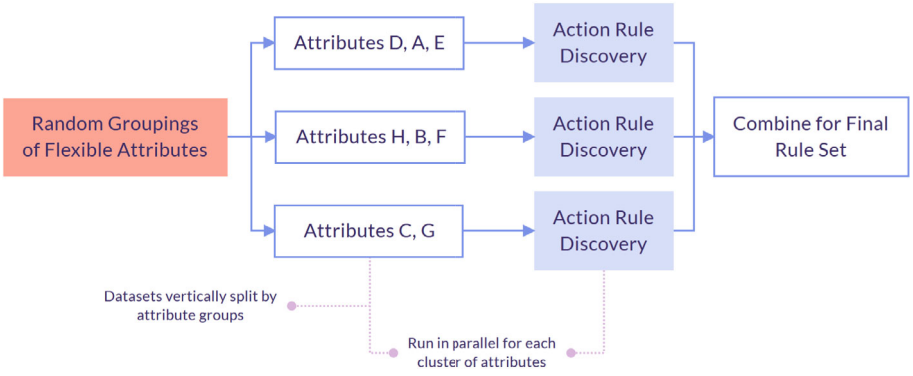
In this study, we propose an approach for distributed action rule discovery that leverages attribute correlation to perform vertical partitioning on datasets with numerous flexible attributes. We present a detailed explanation of our methodology in this section, highlighting how it contrasts with a random-based partitioning approach used for comparison. The differences are emphasized in Fig. 1.

3.1 Attribute correlation-based vertical partitioning

Our methodology employs correlation-based vertical partitioning to manage datasets with numerous flexible attributes. By clustering attributes based on their pairwise correlations, this approach creates partitions that group highly related attributes together, enabling more focused and efficient action rule discovery. The process iteratively examines each level of the resulting dendrogram, generating and combining rules for clusters within each level to identify the optimal set of action rules. Algorithm 1 outlines the key steps of this process.



(a) Method 1: Correlation-Based Partitioning



(b) Method 2: Random-Based Partitioning

Fig. 1 Comparison of action rule derivation using correlation-based vs. random-based partitioning. the only difference between the two figures is the partitioning method used, highlighted by different colored blocks

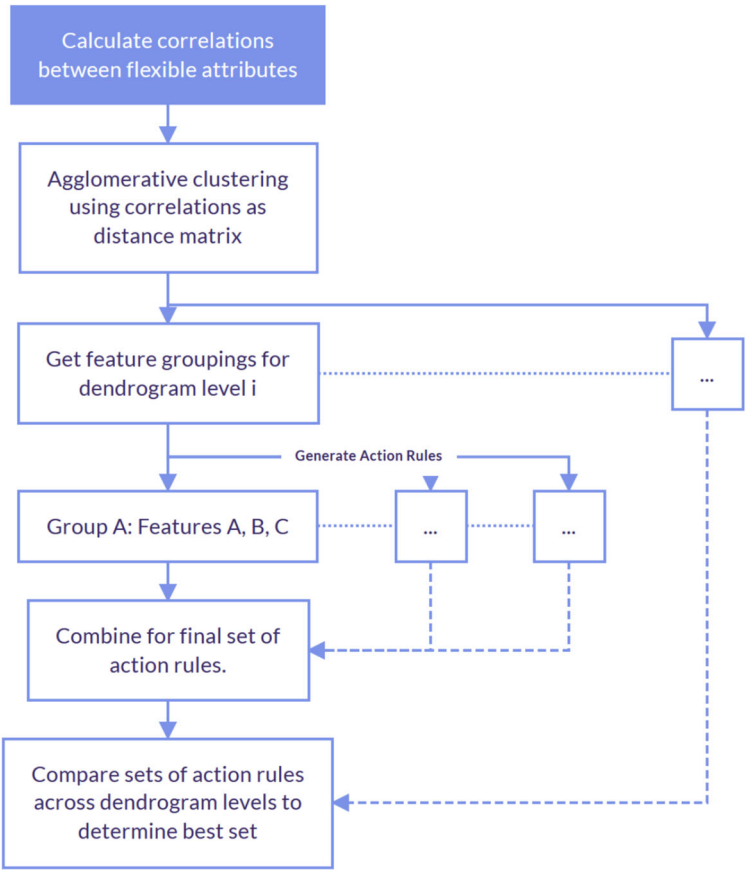


Fig. 2 Detailed overview of the action rule derivation process using correlation-based partitioning

Algorithm 1 Vertical partitioning for action rule discovery.

Require: Dataset with flexible attributes
Ensure: Optimal set of action rules based on correlation-based feature partitioning
1: Partition flexible attributes into clusters based on correlation, creating hierarchical levels. ▷ See Section 3.1.1
2: Initialize parallel processing ▷ See Section 3.1.2
3: **for** each level of partitions (dendrogram levels) **do**
4: **for** each cluster within the current level **do**
5: Generate action rules for the current cluster in parallel. ▷ See Section 3.1.2
6: **end for**
7: Combine rules from all clusters within the current level to form possible rule combinations. ▷ See Section 3.1.3
8: Store the combined rules for this level.
9: **end for**
10: Retrieve and compile the rules generated at all levels.
11: Determine the partition level with the best F-score and use its corresponding action rules as the final result. ▷ See Section 3.1.4

In addition to returning the level with the highest f-score as the final set of rules, for the sake of our experiments, we also present the results for every level of the dendrogram for analysis. This encompasses several performance metrics along with the enumerated rules that were derived (Fig. 2).

3.1.1 Hierarchical clustering of flexible attributes based on correlation

To calculate the correlations between the flexible classification attributes, we utilize the following methods based on the attribute types: Pearson's correlation coefficient for continuous attributes, Cramer's V for categorical attributes, and the Correlation Ratio for mixed attribute types. These measures are well-established in statistical literature for handling mixed-type data, as discussed in Akoglu's comprehensive overview of correlation coefficients (Akoglu, 2018). These similarity measures ensure that attributes frequently influencing each other are grouped together, leading to partitions that better reflect real-world relationships. Each method is detailed below:

Pearson's Correlation Coefficient:
Pearson's correlation coefficient, denoted as r, measures the linear relationship between two continuous variables X and Y:

$$r = \frac{\sum_{i=1}^{n}(X_i - \overline{X})(Y_i - \overline{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \overline{X})^2 \sum_{i=1}^{n}(Y_i - \overline{Y})^2}}, \tag{4}$$

where $\overline{X}$ and $\overline{Y}$ are the means of X and Y, respectively.

Cramer's V:
Cramer's V measures the association between two categorical variables and is defined as:

$$V = \sqrt{\frac{\chi^2}{n \cdot \min(k-1, r-1)}}, \tag{5}$$

where χ^2 is the chi-squared statistic, n is the total sample size, k is the number of columns, and r is the number of rows.

Correlation Ratio (η):
The Correlation Ratio measures the strength of the relationship between a continuous dependent variable Y and a categorical independent variable X. It is computed as:

$$\eta = \sqrt{\frac{\sum_{i=1}^{k} n_i(\overline{Y}_i - \overline{Y})^2}{\sum_{i=1}^{n}(Y_i - \overline{Y})^2}}, \tag{6}$$

where k is the number of categories, n_i is the number of samples in category i, $\overline{Y}_i$ is the mean of Y for category i, and $\overline{Y}$ is the overall mean of Y.

Next, to perform agglomerative clustering, we must first compute the distance matrix, which serves as the input for clustering. The distance matrix is calculated using the following equation:

$$\text{Distance Matrix} = 1 - \frac{|\text{associations}| + 1}{2} \tag{7}$$

Here, *associations* refer to the correlation values calculated in the previous step for pairs of flexible attributes. The absolute value of these associations is taken to ensure all values are positive, as the clustering process focuses solely on the strength of the relationships rather than their direction (positive or negative).

Agglomerative clustering was selected over alternative clustering methods due to its hierarchical nature, which allows for dynamic partitioning at different levels. Unlike partition-based methods such as k-means or DBSCAN, agglomerative clustering does not require pre-specifying the number of clusters. This is particularly useful in action rule discovery, where the optimal partitioning level is determined dynamically based on evaluation metrics. Additionally, hierarchical clustering preserves attribute relationships across different resolutions, making it more interpretable for action rule generation.

We use correlation-based similarity instead of traditional distance metrics (e.g., Euclidean distance) to better capture relationships between attributes. Correlation measures (Pearson's, Cramér's V, and correlation ratio) focus on dependencies between attributes rather than absolute differences, ensuring that attributes influencing each other are grouped effectively. Alternative similarity measures such as cosine similarity or Mahalanobis distance may be useful in specific contexts but were less suitable for the structured, categorical, and mixed-type data in this study.

Agglomerative clustering is then applied to the flexible attributes using the computed distance matrix, employing a single-linkage method. Single-linkage clustering merges clusters based on the smallest distance between any two points across clusters, resulting in the creation of a dendrogram. This dendrogram visually represents the hierarchical relationships between attributes and provides a framework for analyzing clustering at multiple levels.

Single-linkage clustering was chosen because it prioritizes preserving the strongest attribute relationships by linking the most correlated attribute pairs first. This ensures that highly interrelated attributes remain grouped together, forming coherent partitions for action rule discovery. Given the objective of grouping highly correlated attributes while maintaining flexibility in cluster shape, single-linkage clustering provided the most suitable structure for partitioning flexible attributes in our methodology.

From the dendrogram, we derive clusters of flexible attributes at each level. The purpose of clustering is to reduce the complexity of analyzing flexible attributes by grouping them into smaller, more manageable subsets. These clusters are then evaluated to identify the level of the dendrogram that yields the most effective action rules. Each level of the dendrogram, consisting of various clusters of flexible attributes, is processed to generate and evaluate action rules. To enhance efficiency, the processing of dendrogram levels, including rule generation and evaluation, is executed in parallel.

By providing a systematic approach to partitioning flexible attributes and evaluating each dendrogram level, this methodology ensures a thorough exploration of clustering configurations to identify the optimal grouping for action rule generation.

While our current approach is guided by domain-specific reasoning, future work could empirically compare different clustering techniques, similarity measures, and linkage methods to assess their impact on action rule discovery. A systematic evaluation across multiple datasets would provide insights into whether alternative methods yield improved partitions in terms of rule consistency, computational efficiency, and interpretability.

3.1.2 Parallel generation of action rules for current cluster

Subsequently, action rules were developed for each cluster of attributes at a specific level in the dendrogram. Similarly for efficiency, we can devise action rules for every cluster in

parallel. Our study employed a method for uncovering action rules utilizing the Rough Sets Exploration System (RSES) (Bazan & Szczuka, 2005) to facilitate comparative assessments. Algorithm 2 outlines the key steps of this process.

Algorithm 2 Action rule generation using RSES classification rule outputs.

Require: Flexible attributes F, Configuration parameters C
Ensure: Set of action rules R
Load classification rules CR from RSES output
Split CR into $target\_from$ and $target\_to$ buckets based on decision outcomes
Initialize compatible pairs $CP \leftarrow []$
for each $rule\_from \in target\_from$ **do**
  for each $rule\_to \in target\_to$ **do**
    if Stable attributes are compatible and support thresholds are met **then**
      $CP.append((rule\_from, rule\_to))$
    end if
  end for
end for
Initialize action rules $R \leftarrow []$
for each $(rule\_from, rule\_to) \in CP$ **do**
  Extract stable attributes S and flexible changes F'
  if F' is non-empty and thresholds are met **then**
    Create and add action rule AR to R
  end if
end for
return R

The process of generating action rules using the Rough Sets Exploration System (RSES) builds upon classification rules derived from RSES outputs. These classification rules encapsulate patterns or associations among attributes in the dataset. Each rule consists of stable attributes, which remain constant across observations, flexible attributes, which are allowed to vary, and decision attributes, representing the target outcome or category associated with the rule. To facilitate action rule discovery, these classification rules are parsed and grouped into distinct categories based on their decision outcomes, such as "passive" or "promoter." This grouping ensures that the subsequent steps focus on transitions between desired decision categories.

A critical step in this process involves identifying pairs of classification rules that are compatible based on their stable attributes. Compatibility is defined as the alignment of stable attributes between rules, such that corresponding stable attributes have identical values or are absent in one of the rules. By ensuring that stable attributes are consistent, the method isolates variations in flexible attributes, which serve as the basis for action rules. This alignment is essential for maintaining the interpretability and validity of the proposed actions.

Once compatible rule pairs are identified, action rules are generated by analyzing the differences in flexible attributes between the paired rules. These differences represent potential actions, such as a change in the value of a specific attribute, that are hypothesized to drive a transition from one decision category to another (e.g., from "passive" to "promoter"). For each pair, an action rule object is created, encapsulating the stable attributes, the identified changes in flexible attributes, and the desired decision transition. The process ensures that the action rules are both interpretable and actionable.

To validate the generated action rules, their support and confidence are computed. Support measures the prevalence of the rule in the dataset, while confidence quantifies the likelihood that the proposed action leads to the desired outcome. Only rules that meet predefined thresh-

olds for both metrics are retained, ensuring that the final set of action rules is both reliable and relevant.

The resulting action rules provide actionable recommendations for modifying flexible attribute values to achieve targeted decision outcomes. By systematically leveraging the distinctions between stable and flexible attributes, this method enables the discovery of interpretable rules that offer meaningful insights into the dataset. Furthermore, the integration of compatibility checks, targeted decision transitions, and validation metrics ensures that the generated rules are grounded in the dataset's underlying structure and tailored to address specific objectives.

3.1.3 Combining rules across clusters for potential rule generation

Once action rules are generated for each cluster within a dendrogram level, the next step involves combining these rules across clusters to form more comprehensive rules, as outlined in Algorithm 3. This process leverages a method specifically designed to balance efficiency and precision by incorporating confidence and support thresholds at intermediate steps.

Algorithm 3 Depth-controlled rule combination with threshold filtering.

Require: List of rule clusters $L = [C_1, C_2, ..., C_n]$, confidence threshold t_c, support threshold t_s, maximum depth d
Ensure: Set of valid combined action rules
Initialize empty list $R_{combined}$ to store resulting rules
Initialize empty dictionary $rule\_status$ to track processed combinations
for $depth = 1$ to d **do**
  for each combination of $depth$ clusters from L **do**
    for each rule combination from Cartesian product of selected clusters **do**
      Derive a key based on shared stable attributes among rules
      if key exists in $rule\_status$ and is marked as invalid **then**
        continue to next combination
      end if
      Attempt to merge rules
      if $depth = d$ **then**
        Evaluate combined rule against thresholds t_c and t_s
        if rule meets thresholds **then**
          Add rule to $R_{combined}$
          Mark key as valid in $rule\_status$
        else
          Mark key as invalid in $rule\_status$
        end if
      else
        Add rule to $R_{combined}$
      end if
    end for
  end for
end for
return $R_{combined}$

The algorithm incrementally explores combinations of increasing depth, ranging from single-cluster rules to combinations that span multiple clusters (up to a predefined maximum depth). To combine rules across clusters, a Cartesian product of the filtered rules is computed, exploring all possible combinations. Each combination is uniquely identified by a key derived

from the stable attributes shared among the rules in the combination. This key is used to avoid reprocessing combinations that have already failed to meet criteria in previous iterations.

The Cartesian product inherently results in exponential growth in the number of combinations as the number of clusters and the combination depth increase. To address this computational challenge, the methodology incorporates two key strategies: (1) depth constraints, which limit the number of rules combined at each step, and (2) threshold-based filtering, which significantly reduces the candidate set by excluding low-quality rules early in the process. These strategies balance computational demands with the thoroughness of the evaluation, ensuring the identification of high-quality, actionable insights.

The combination process is depth-controlled, progressing incrementally from single-rule combinations per cluster to a predefined maximum depth. At the maximum depth, the algorithm critically evaluates the merged rules, retaining only those that satisfy confidence and support thresholds. Rules failing to meet these criteria are pruned, thereby optimizing the final rule set for downstream analysis.

The resulting rule set for each dendrogram level encapsulates the most actionable and high-quality insights derived from the given clustering of flexible attributes. This systematic approach ensures the scalability of the methodology while maintaining the rigor required for generating meaningful action rules.

Future enhancements could include heuristic search strategies to focus on promising regions of the rule space or additional pruning techniques during the combination phase. These refinements aim to further improve scalability while preserving the depth and quality of the analysis.

3.1.4 Selecting the optimal dendrogram level

Once action rules have been generated for each level of the dendrogram, we evaluate their effectiveness using the F-score, a comprehensive metric that balances precision and recall. The dendrogram level with the highest F-score is selected as the optimal level, containing the final set of action rules. This study focuses on F-score as the primary evaluation metric; however, other metrics, such as lightness, coverage, or the total number of rules, could also be employed depending on the specific application. Alternatively, action rules from all levels could be retained for broader analysis.

The process of determining the optimal level is iterative and ensures a systematic evaluation of potential rule combinations. At each step, rules that fail to meet predefined thresholds for support and confidence are pruned, resulting in a refined set of action rules that are both actionable and meaningful.

This iterative and depth-controlled approach enhances both the quality and efficiency of the rule combination process. By systematically exploring the rule space and integrating threshold-based filtering, the methodology ensures the generation of high-quality action rules while maintaining computational feasibility. The final rule set provides robust insights for downstream decision-making and demonstrates the practical effectiveness of the vertical correlation partitioning framework.

To determine the optimal clustering level in the dendrogram, we compute the F-score for the cluster set at each level. The F-score balances precision and recall, helping to identify the most meaningful partitioning of attributes.

For each level, we evaluate the quality of the action rules at that level. The F-score for each level is computed using:

$$F = \frac{2 \times (\text{Precision} \times \text{Recall})}{\text{Precision} + \text{Recall}} \tag{8}$$

where precision is computed using the support and confidence of the action rules generated at a given level using:

$$\text{Precision} = \frac{\sum_i (\text{support}_i \times \text{confidence}_i)}{\sum_i \text{support}_i}. \tag{9}$$

Similarly, recall can be computed as:

$$\text{Recall} = \frac{\left|\bigcup_i \text{covered\_tuples}_i\right|}{|\{x \in \mathcal{D} \mid x[\text{decision attribute}] = \text{from\_value}\}|} \tag{10}$$

where:

- $\bigcup_i \text{covered\_tuples}_i$ represents the set of unique tuples covered by all action rules discovered in a level. This is computed by identifying all data points satisfying at least one rule's antecedent conditions and ensuring no duplicates are counted.
- $\{x \in \mathcal{D} \mid x[\text{decision attribute}] = \text{from\_value}\}$ is the total number of tuples in the dataset where the decision attribute is equal to the specified "from" value, or the antecedent.

These metrics are further discussed in Section 5.1.

3.1.5 Example calculation of precision, recall, and F-score

To illustrate the computation of precision, recall, and F-score, consider a dataset where the goal is to transition customers from a *Passive* satisfaction level to a *Promoter* level. The dataset consists of ten tuples, where the decision attribute *Satisfaction* takes values *Passive* or *Promoter*. Action rules are derived based on classification rules and evaluated using precision and recall. Table 1 presents a sample dataset used for action rule generation.

Suppose we extract the following action rules:

Table 1 Example dataset for action rule generation

ID	Customer Segment	Response Time	Discount Level	Satisfaction
1	Young Adults	Slow	None	Passive
2	Young Adults	Slow	Low	Passive
3	Young Adults	Fast	None	Promoter
4	Young Adults	Slow	High	Passive
5	Middle-Aged	Fast	High	Promoter
6	Middle-Aged	Slow	Low	Passive
7	Seniors	Fast	None	Promoter
8	Seniors	Slow	High	Passive
9	Seniors	Slow	None	Passive
10	Seniors	Fast	High	Promoter

- **Rule 1:** If *Customer Segment* = 'Young Adults' and *Response Time* = 'Slow' → 'Fast', then *Satisfaction* = 'Passive' → 'Promoter'.
  Support = 3, Confidence = 0.67
- **Rule 2:** If *Customer Segment* = 'Seniors' and *Discount Level* = 'None' → 'High', then *Satisfaction* = 'Passive' → 'Promoter'.
  Support = 2, Confidence = 0.50

Precision is computed using the support and confidence of the action rules:

$$\text{Precision} = \frac{\sum_i (\text{support}_i \times \text{confidence}_i)}{\sum_i \text{support}_i}. \tag{11}$$

Substituting the values:

$$\text{Precision} = \frac{(3 \times 0.67) + (2 \times 0.50)}{3 + 2} = \frac{2.01 + 1}{5} = \frac{3.01}{5} = 0.60. \tag{12}$$

Recall is calculated based on the number of unique tuples covered by the action rules, relative to the total number of tuples with the 'from' decision value (*Satisfaction* = 'Passive'):

$$\text{Recall} = \frac{\left|\bigcup_i \text{covered\_tuples}_i\right|}{|\{x \in \mathcal{D} \mid x[\text{Satisfaction}] = \text{Passive}\}|} \tag{13}$$

where:

- $\bigcup_i \text{covered\_tuples}_i$ represents the set of unique tuples covered by all action rules.
- $\{x \in \mathcal{D} \mid x[\text{Satisfaction}] = \text{Passive}\}$ is the total number of tuples in the dataset with *Satisfaction* = 'Passive'.

Given that Rule 1 covers 3 unique tuples (IDs: 1, 2, 4) and Rule 2 covers 2 unique tuples (IDs: 8, 9), the total number of covered tuples is 5. Since there are 6 tuples in the dataset where *Satisfaction* = 'Passive', recall is computed as:

$$\text{Recall} = \frac{5}{6} = 0.83. \tag{14}$$

The F-score balances precision and recall:

$$F = \frac{2 \times (\text{Precision} \times \text{Recall})}{\text{Precision} + \text{Recall}}. \tag{15}$$

Substituting the computed values:

$$F = \frac{2 \times (0.60 \times 0.83)}{0.60 + 0.83} = \frac{2 \times 0.498}{1.43} = \frac{0.996}{1.43} = 0.70. \tag{16}$$

This example demonstrates how precision measures the reliability of action rules, while recall evaluates their coverage over the dataset. The F-score provides a balanced assessment by incorporating both measures.

3.1.6 Example walkthrough of correlation-based vertical partitioning and action rule derivation

To illustrate the full pipeline of our method, we use a toy dataset presented in Table 2. This dataset consists of ten records representing customer interactions across several service dimensions, including a mixture of stable, flexible, and decision attributes.

In this example, we have the following attributes:

Table 2 Expanded toy dataset used for example walkthrough

ID	CS	RT	DL	Sup	DS	Sat
1	Young Adults	Slow	None	Poor	Late	Passive
2	Young Adults	Slow	Low	Poor	On Time	Passive
3	Young Adults	Fast	None	Excellent	On Time	Promoter
4	Young Adults	Slow	High	Average	Late	Passive
5	Middle-Aged	Fast	High	Excellent	Early	Promoter
6	Middle-Aged	Slow	Low	Poor	Late	Passive
7	Seniors	Fast	None	Excellent	On Time	Promoter
8	Seniors	Slow	High	Average	Late	Passive
9	Seniors	Slow	None	Poor	Late	Passive
10	Seniors	Fast	High	Excellent	Early	Promoter

Abbreviations: CS = Customer Segment (stable attribute), RT = Response Time, DL = Discount Level, Sup = Customer Support, DS = Delivery Speed (all flexible attributes), Sat = Satisfaction (decision attribute)

- Stable attribute: *Customer Segment*, which represents a demographic grouping and remains unchanged across rule transitions.
- Flexible attributes: *Response Time*, *Discount Level*, *Customer Support*, and *Delivery Speed*, all of which are modifiable and may influence customer satisfaction.
- Decision attribute: *Satisfaction*, the outcome we aim to improve from *Passive* to *Promoter* via actionable changes.

We note that the categorization of attributes as stable, flexible, or decision-related is typically determined by the domain expert or user prior to applying the method. In practice, this designation reflects real-world constraints — for instance, demographic information like *Customer Segment* is considered stable and not subject to change, while service-related factors such as *Response Time* or *Discount Level* are treated as flexible since they can be modified through intervention.

First, we must calculate the correlation matrix. Using appropriate statistical measures (e.g., Cramér's V for categorical data), we compute the pairwise relationships between the flexible attributes. The flexible attributes in this dataset are abbreviated as follows: RT = Response Time, DL = Discount Level, Sup = Customer Support, and DS = Delivery Speed. A sample correlation matrix for this toy dataset is shown below:

	RT	DL	Sup	DS
RT	1.00	0.48	0.72	0.55
DL	0.48	1.00	0.41	0.38
Sup	0.72	0.41	1.00	0.60
DS	0.55	0.38	0.60	1.00

This matrix reflects the degree of association between each pair of flexible attributes. For example, RT and Sup have a strong correlation (0.72), indicating they tend to co-vary across the dataset and are likely to be grouped in the same cluster during partitioning.

Based on the correlation matrix, we then apply agglomerative hierarchical clustering with single-linkage to group related attributes. The resulting dendrogram is visualized in Fig. 3.

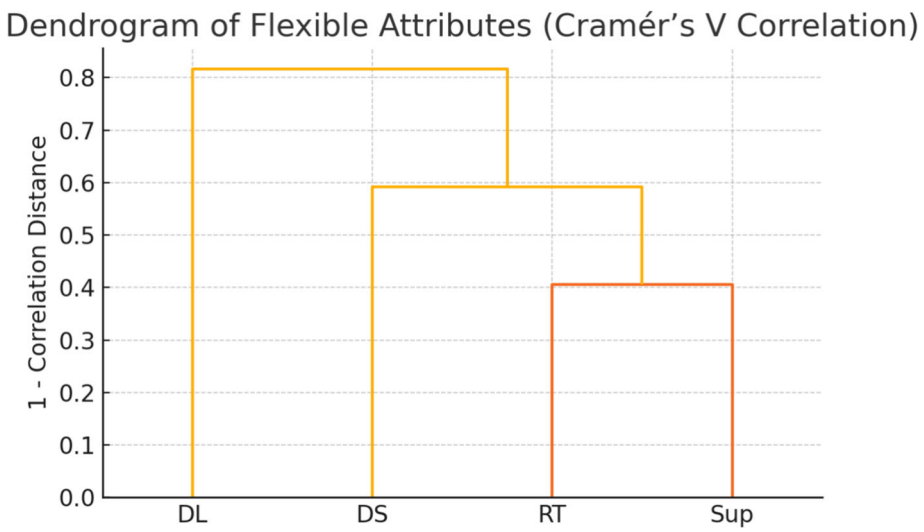


Fig. 3 Dendrogram of flexible attributes (RT, DL, Sup, DS) based on Cramér's V correlation

In this toy example, the strongest correlation is between Response Time (RT) and Customer Support (Sup), which are merged early in the clustering process. Delivery Speed (DS) joins this cluster next, while Discount Level (DL) is the most weakly correlated and is grouped last. These clusters are used to define partitions for parallel rule generation in the next step.

Once the attributes are clustered, action rule generation is run in parallel for each cluster. Consider a cluster formed by RT (Response Time) and Sup (Customer Support). Using the RSES system, we may extract the following classification rules:

- Rule 1 (Passive): If CS = 'Young Adults', RT = 'Slow', Sup = 'Poor', then Sat = 'Passive'
- Rule 2 (Promoter): If CS = 'Young Adults', RT = 'Fast', Sup = 'Excellent', then Sat = 'Promoter'

Since both rules share the same stable attribute (CS = 'Young Adults'), they are considered compatible. The transformation between their differing flexible attributes produces the following action rule:

$$\begin{aligned}&(\text{CS} = \text{Young Adults}) \wedge (\text{RT}, \text{Slow} \rightarrow \text{Fast})\\ \wedge&(\text{Sup}, \text{Poor} \rightarrow \text{Excellent}) \Rightarrow (\text{Sat}, \text{Passive} \rightarrow \text{Promoter})\end{aligned} \tag{17}$$

This rule suggests that for young adult customers who are currently passive in satisfaction, improving response time from slow to fast and enhancing customer support from poor to excellent may lead them to become promoters.

Once rules are generated for each cluster, we explore possible combinations across clusters in a depth-controlled manner. Our method first constructs combinations of rules incrementally across depths — starting from pairs and progressing up to a predefined maximum depth. At each step, combinations are only considered if the involved rules share the same stable and decision attributes. This compatibility is tracked using a unique key derived from the stable attributes, ensuring that incompatible rule combinations are pruned early.

At the maximum depth, the combined rules are evaluated for their overall confidence and support. Only those that meet the defined thresholds are retained, ensuring that the resulting

extended action rules are both meaningful and statistically valid. For example, consider the following rules from different clusters:

For example, consider the following rules discovered from three different clusters:

- Rule A (Cluster 1 - RT and Sup): (CS = Young Adults) $\wedge$ (RT, Slow $\rightarrow$ Fast) $\wedge$ (Sup, Poor $\rightarrow$ Excellent) $\Rightarrow$ (Sat, Passive $\rightarrow$ Promoter)
- Rule B (Cluster 2 - DS): (CS = Young Adults) $\wedge$ (DS, Late $\rightarrow$ Early) $\Rightarrow$ (Sat, Passive $\rightarrow$ Promoter)
- Rule C (Cluster 3 - DL): (CS = Young Adults) $\wedge$ (DL, None $\rightarrow$ High) $\Rightarrow$ (Sat, Passive $\rightarrow$ Promoter)

Since these rules share the same stable and decision attributes, they are merged to form the following extended action rule:

$$\begin{aligned}&(\text{CS} = \text{Young Adults}) \wedge (\text{RT, Slow} \rightarrow \text{Fast}) \wedge (\text{Sup, Poor} \rightarrow \text{Excellent})\\ &\wedge(\text{DS, Late} \rightarrow \text{Early}) \wedge (\text{DL, None} \rightarrow \text{High}) \Rightarrow (\text{Sat, Passive} \rightarrow \text{Promoter})\end{aligned} \tag{18}$$

This approach enables the discovery of richer action rules that synthesize insights from multiple attribute clusters, while still maintaining interpretability and precision. Each combined rule is evaluated against the original dataset to calculate its support and confidence. Only those that meet the minimum thresholds are retained, ensuring that final rules are both meaningful and statistically valid.

Finally, each dendrogram level (partition granularity) results in a rule set. The level producing the highest F-score (balancing coverage and confidence) is selected as the final action rule output. In this simplified example, only one level is shown, but in practice, this step evaluates all clustering levels in parallel.

This walkthrough illustrates each step of the proposed method using a realistic, interpretable dataset. It reinforces how attribute correlation structures the rule discovery process and supports the generation of meaningful, actionable recommendations.

3.2 Random vertical partitioning

Random vertical partitioning offers a contrasting approach to our proposed correlation-based partitioning. Unlike the structured grouping derived from attribute correlations, random partitioning arbitrarily divides flexible attributes into clusters, without considering their relationships or interdependencies. This method follows the approach introduced by Tarnowska et al. (2022).

In random partitioning, flexible attributes are shuffled and split into varying numbers of groups, ranging from two to the total number of attributes (exclusive). These groups emulate the hierarchical levels in correlation-based dendrograms, but without the guidance of attribute correlation. For each random partition, action rules are generated for clusters of attributes, and all possible combinations of rules are created, filtered by support and confidence thresholds.

This method explores diverse partitions of the dataset, providing a baseline for comparison. By evaluating the F-score for each partitioning level, the partition with the highest score is selected as the optimal set of action rules. The process, while computationally similar, lacks the targeted refinement of correlation-based partitioning, highlighting the benefits of structured attribute grouping in our proposed methodology.

3.2.1 Comparison with random vertical partitioning

To highlight the advantages of correlation-based vertical partitioning, we briefly contrast it with a random partitioning approach using the same toy dataset from Section 3.1.6. In random partitioning, flexible attributes are grouped arbitrarily rather than based on their statistical relationships. For instance, *Response Time* might be grouped with *Discount Level*, while *Customer Support* and *Delivery Speed* are split into separate partitions.

This arbitrary grouping increases the risk of separating highly interdependent attributes, such as *Response Time* and *Customer Support*, into different partitions. As a result, strong action rules involving both attributes cannot be discovered during the same iteration, leading to weaker intermediate rules.

For example, one might derive an action rule such as:

$$(\text{CS} = \text{Young Adults}) \wedge (\text{RT}, \text{Slow} \rightarrow \text{Fast}) \Rightarrow (\text{Sat}, \text{Passive} \rightarrow \text{Promoter}) \tag{19}$$

This rule may hold, but it lacks the additional context provided by other correlated flexible attributes (e.g., *Customer Support* or *Delivery Speed*), which may significantly improve the rule's confidence and support.

Because attribute groupings are not optimized, random partitioning typically requires more iterations and re-evaluation of multiple partition configurations before discovering high-quality action rules. In contrast, correlation-based partitioning—guided by hierarchical clustering—groups related attributes together from the start, resulting in faster convergence and more interpretable, high-confidence rules.

4 Data preparation

4.1 Dataset source

Our study investigates customer feedback data provided by The Daniel Group, focusing on the heavy equipment repair industry in the United States and Canada. This dataset encompasses structured survey responses collected via phone interviews across 38 companies and includes data from over 340,000 customers. Each survey captures both structured numerical ratings and unstructured textual feedback, offering a comprehensive view of customer experiences.

The structured portion of the dataset consists of benchmarks, which are numerical ratings (0-10) evaluating various aspects of customer interactions, such as service quality, communication, and ease of completing orders. These benchmarks serve as flexible attributes in our analysis. Meanwhile, the decision attribute, PromoterStatus, classifies customers into one of three categories: Detractors (scores of 0–6), Passives (scores of 7–8), or Promoters (scores of 9–10), based on their likelihood of recommending the company.

To narrow the scope for actionable analysis, we utilized survey data from four companies recorded during 2015. The dataset was preprocessed to address inconsistencies across companies and prepare the data for action rule mining. Specifically, stable attributes such as division, survey type, and channel type were consistent across the datasets, while flexible attributes varied based on the benchmarks surveyed. The aim of this study was to identify rules that could transition Passive customers into Promoters, thereby boosting customer satisfaction and loyalty.

4.2 Data cleaning

Inconsistencies in the semantics of the dataset in 38 companies posed challenges to the confidence of rule extraction. To mitigate these issues, our analysis was limited to datasets from two companies, referred to as Company A and Company B. These datasets provided a more coherent and consistent basis for generating actionable insights.

Data preprocessing was conducted using the *pandas* library in Python, ensuring efficient and reproducible data cleaning and preparation. Several preprocessing steps were implemented to ensure the quality and reliability of the analysis:

1. ***Column Sparsity Assessment and Removal.*** Columns with excessive sparsity were identified and removed to prevent noise and ensure meaningful analysis. Specifically, columns with 75% or more null values were eliminated from the datasets.
2. ***Correlation Analysis.*** To address redundancy, we analyzed correlations between columns using the Pearson correlation coefficient. Features exhibiting perfect one-to-one correlations were deemed redundant and removed, retaining only one representative feature from each correlated pair.
3. ***Handling Null Values.*** Null values were addressed based on the context of the data. For the *PromoterScores*, which served as the decision attribute, rows with missing values were removed entirely to avoid compromising the analysis. For benchmark features, null values were interpreted as a lack of response and categorized under the label *No Response*.
4. ***Binning of Benchmark Features.*** Benchmark features were transformed into categorical bins to standardize their representation and facilitate rule generation. This binning process grouped responses as follows:

- *No Response*: Null values or missing entries.
- *Low*: Scores ranging from 0 to 4.
- *Medium*: Scores ranging from 5 to 6.
- *High*: Scores ranging from 7 to 8.
- *Very High*: Scores ranging from 9 to 10.

These preprocessing steps ensured a consistent, high-quality dataset for rule extraction. By addressing sparsity, redundancy, and null values, and standardizing benchmark attributes, the cleaned datasets from Companies A and B were prepared for action rule discovery.

4.3 Dataset description and experimental parameters

Dataset A consisted of 542 rows, while Dataset B contained 1279 rows. Both datasets underwent preprocessing to ensure consistency and suitability for analysis. Three stable attributes were identified in both datasets: *division*, *survey type*, and *channel type*. These attributes remained unchanged across all action rules.

The flexible attributes varied slightly between the datasets: Dataset A contained 11 flexible attributes, while Dataset B contained 10. These flexible attributes, derived from customer satisfaction surveys, represented benchmarks of interest within the datasets. The outcome attribute for both datasets was the *promoter status*, which could take one of three values: *Detractor*, *Passive*, or *Promoter*. Our experiments focused on generating action rules to facilitate transitions in the promoter status from *Passive* to *Promoter*.

For filtering rules, a confidence threshold of 0.8 and a support threshold of 2 were applied. For our combination method, we also set a max depth of 2. These thresholds ensured the extraction of high-quality and actionable rules.

Flexible Attributes.
Below is the complete list of flexible attributes used in the datasets, along with their abbreviations and descriptions:

- *BenchmarkAllDealerCommunication* (**DC**): Evaluates communication effectiveness within the dealer network.
- *BenchmarkAllLikelihoodtobeRepeatCustomer* (**RC**): Measures the likelihood of customer repeat business.
- *BenchmarkAllOverallSatisfaction* (**OS**): Assesses overall customer satisfaction levels.
- *BenchmarkPartsEaseofCompletingPartsOrder* (**ECP**): Gauges the ease of completing parts orders.
- *BenchmarkPartsPartsAvailability* (**PA**): Examines component availability for order fulfillment.
- *BenchmarkPartsTimeitTooktoPlaceOrder* (**TPO**): Measures the time taken for order processing.
- *BenchmarkPartsExplanationofDeliveryOptionsCosts* (**EDOC**): Explores the elucidation of delivery options and related costs.
- *BenchmarkPartsOrderAccuracy* (**OA**): Scrutinizes the accuracy of order processing.
- *BenchmarkPartsPromptNotificationofBackOrders* (**NBO**): Examines the timeliness of back-order notifications.
- *BenchmarkPartsKnowledgeofPersonnel* (**KP**): Assesses personnel knowledge about parts and their applications.
- *BenchmarkReferralBehavior* (**RB**): Explores referral behaviors.
- *BenchmarkPartsHowOrdersArePlaced* (**HOP**): Examines the methods of placing orders.

This comprehensive attribute list provides a foundation for generating action rules that highlight opportunities to improve customer experiences and satisfaction.

5 Experimental setup

Our objective was to evaluate and compare the effectiveness of three methodologies for action rule generation: vertical correlation partitioning (our proposed approach), random vertical partitioning, and no partitioning. In the no partitioning method, action rules were generated using all available flexible attributes without any prior clustering or segmentation.

The performance of these methodologies was assessed using a set of evaluation metrics, which are described in detail in Section 5.1. These metrics provide insights into computational efficiency, the richness of the generated rule sets, and the quality of the rules in terms of their applicability and overlap.

The experiments were conducted on the UNC Charlotte Orion research cluster, which consists of compute nodes equipped with Intel Xeon CPUs. Each node features 32 cores and 128 GB of RAM. The SLURM workload manager was employed for job scheduling, resource allocation, and monitoring. All experiments were implemented in Python 3.8.5, leveraging the Ray parallel processing framework to accelerate computation. Each experimental run utilized a single compute node with 16 CPUs and 4 GB of memory allocated per CPU. The

computational capabilities of the cluster significantly reduced the time required to process and analyze the datasets.

To ensure reproducibility, we used the following experimental parameters:

- **Software Environment:** Python 3.8.5 with the following key libraries: `pandas` for data processing, `dython` for calculating associations, `scipy` for clustering, and `ray` for parallel computation.
- **Hardware Configuration:** Each compute node was allocated 16 CPUs and 64 GB of RAM for each experimental setup.
- **Execution Settings:** SLURM was used to manage job scheduling and ensure uniform resource allocation across experimental runs.
- **Dataset Details:** All experiments were performed on the processed datasets described in Section 4, using consistent preprocessing steps.
- **Evaluation Settings:** A confidence threshold of 0.8 and a support threshold of 2 were applied across all experiments to filter the generated action rules.

5.1 Evaluation metrics

To compare the effectiveness of the proposed method, several evaluation metrics were employed, specifically adapted to the context of action rule discovery: runtime, the number of generated rules, precision, recall, F1-Score, coverage, and lightness.

Runtime.
Runtime refers to the duration required for a method to complete its execution. Measured in seconds, this metric captures the computational efficiency of the method, particularly important when working with large datasets. Computationally efficient methods not only reduce processing time but also ensure practical applicability in real-world scenarios.

Number of Generated Rules.
The number of generated rules quantifies the distinct action rules produced by a method. A higher number of rules reflects a richer rule set, providing diverse insights and potential actionable items. However, it is important to balance the quantity of rules with their quality to avoid overwhelming or less meaningful results.

Precision.
In the context of action rules, precision evaluates the reliability of a rule by measuring the proportion of tuples satisfying both the antecedent and consequent conditions out of those satisfying the antecedent alone:

$$\text{Precision} = \frac{\text{Number of tuples satisfying both antecedent and consequent conditions}}{\text{Number of tuples satisfying the antecedent conditions}} \tag{20}$$

Higher precision indicates that the suggested attribute changes are more likely to achieve the desired outcomes.

Recall.
Recall, or coverage, measures how well the rules identify tuples capable of achieving the desired outcome. It is defined as:

$$\text{Recall} = \frac{\text{Number of tuples satisfying both antecedent and consequent conditions}}{\text{Number of tuples satisfying the consequent conditions in the dataset}} \tag{21}$$

This metric emphasizes the breadth of the rule set's impact within the dataset.

F1-Score.
The F1-Score balances precision and recall, providing a single measure of rule quality:

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{22}$$

It ensures that both the reliability and coverage of action rules are jointly evaluated.

Coverage.
Coverage quantifies the unique number of tuples impacted by the action rules. For a set of rules, coverage is calculated as:

$$\text{Coverage(Set)} = \left| \bigcup_{i=1}^{n} \text{Support(Rule}_i) \right| \tag{23}$$

where $\text{Support(Rule}_i)$ represents the set of tuples satisfying the antecedent of rule i.

Lightness.
Lightness measures the evenness of coverage across the rule set. It is defined as:

$$\text{Lightness} = \frac{\sum_{i=1}^{n} \text{Coverage(Rule}_i)}{\text{Coverage(Set)}} \tag{24}$$

Lower lightness values indicate more concise and focused rule sets, while higher values may suggest redundancy. In business and healthcare contexts, where actionable insights are crucial, a lightness value between five and ten is considered ideal.

These metrics collectively provide a robust framework for evaluating the efficiency, quality, and applicability of action rules. By employing them, we ensure that the method generates reliable, impactful, and interpretable insights.

6 Results and discussion

The primary objective of our study was to evaluate the performance of the proposed correlation-based vertical partitioning method compared to both random vertical partitioning and a base method without partitioning. Table 3 presents a comparative analysis of these methods across two datasets. The methods under comparison include "None," representing the baseline without partitioning, "Correlation" for the correlation-based partitioning, and "Random" for random vertical partitioning. Processing time is recorded in seconds, with the "Random" method representing average outcomes from five iterations.

For Dataset A, the base method required 2,543.433 seconds to generate 312 rules, while the correlation-based method completed in 393.558 seconds with 129 rules. Although each run of the random partitioning was faster, taking an average of 164.738 seconds, it required multiple runs to achieve satisfactory consistency, leading to a cumulative runtime of 823.689 seconds. For Dataset B, a similar pattern emerged, with random partitioning taking an average of 34.762 seconds per run over five iterations, while the correlation-based method needed only one iteration to produce stable results.

Table 3 Comparison analysis of vertical partitioning methods across datasets

Dataset	Method	Time (seconds)	Rules	Precision	Coverage	Lightness
A	None	2543.433	312.0	0.939	0.672	28.171
	Correlation	393.558	129.0	0.995	0.623	12.816
	Random	164.738	41.0	0.939	0.525	4.923
B	None	49.077	22.0	0.855	0.418	5.099
	Correlation	39.924	19.0	0.859	0.418	4.324
	Random	34.762	4.2	0.681	0.280	1.618

Adapted from Benedict and Ras (2024b)

6.1 Precision, coverage, and lightness

The correlation-based approach consistently achieved higher precision across datasets, with a precision of 0.995 in Dataset A and 0.859 in Dataset B. This high precision indicates that the correlation-based partitioning method enhances the reliability of generated rules, reducing the likelihood of irrelevant or spurious recommendations—a critical feature in high-stakes applications like customer relationship management or healthcare.

In terms of coverage, the correlation-based approach maintains comparable values to the base method, with 0.623 for Dataset A and 0.418 for Dataset B. Coverage here is a measure of rule applicability across the dataset. This balance between high precision and solid coverage demonstrates that the correlation-based method remains generalizable across the dataset, suggesting it could adapt well to data from other domains where broad coverage is necessary.

Lightness is another key performance metric, as it represents the average number of rules applicable per object, impacting interpretability and ease of rule application. The correlation-based method's lightness (12.816 for Dataset A) suggests a good balance between specificity and applicability, enhancing the rule set's interpretability compared to the denser base method.

6.2 Processing time and efficiency

A key finding is the efficiency of the correlation-based partitioning method. Unlike random partitioning, which requires multiple iterations to achieve stable results, the correlation-based method delivers consistent outcomes in a single run. Although the initial clustering step introduces some computational overhead, this is outweighed by the reduced need for repeated executions during rule generation. This combination of efficiency and consistency makes correlation-based partitioning particularly valuable in scenarios where consistent outputs are crucial, and computational resources are limited.

The correlation-based approach consistently outperformed random partitioning across all datasets, demonstrating superior metrics in precision, coverage, and lightness. Its performance closely approaches that of the base method while maintaining a significant advantage in time efficiency. Although the clustering step incurs an initial computational cost, the subsequent reduction in execution times for rule generation outweighs this cost in datasets with medium to large sizes. For smaller datasets, the efficiency gains may be less pronounced, warranting further exploration.

These results emphasize the strength of the correlation-based method, striking a balance between computational efficiency and the quality of generated rules. This consistency rein-

forces its utility in environments where both accuracy and resource efficiency are critical considerations.

6.3 Rule examples and interpretation

Using the correlation-based partitioning method, one of the derived rules from Dataset A is as follows:

$$\begin{aligned} \text{Rule: } & [\text{ChannelType} = \text{Construction Heavy}] \wedge \\ & [\text{RC} = [\text{High} \rightarrow \text{Very High}]] \wedge \\ & [\text{OA} = [\text{High} \rightarrow \text{Very High}]] \\ \Rightarrow & [\text{PromoterStatus} = [\text{Passive} \rightarrow \text{Promoter}]] \end{aligned}$$

This rule has a confidence of 0.932 and a support of seven, and it can be interpreted as follows: *"If the channel type remains 'construction heavy' and there are improvements in the likelihood of repeat customer ratings (RC) from 'high' to 'very high' and in the accuracy of order processing (OA) from 'high' to 'very high,' then the PromoterStatus is likely to transition from 'Passive' to 'Promoter.'"*

This implies that focusing on enhancing customer retention likelihood and ensuring highly accurate order processing can significantly impact converting passive customers into promoters, especially within the construction-heavy channel type. For customers in the "construction heavy" channel, improving operational metrics (e.g., repeat customer likelihood and order accuracy) can significantly increase customer advocacy.

Another example from the dataset is:

$$\begin{aligned} \text{Rule: } & [\text{RC} = [\text{High} \rightarrow \text{Very High}]] \wedge \\ & [\text{OS} = [\text{High} \rightarrow \text{Very High}]] \wedge \\ & [\text{PA} = [\text{Very High} \rightarrow \text{Medium}]] \\ \Rightarrow & [\text{PromoterStatus} = [\text{Passive} \rightarrow \text{Promoter}]] \end{aligned}$$

This rule has a confidence of 1.000 and a support of six. It can be interpreted as: *"If the likelihood to be a repeat customer (RC) increases from 'high' to 'very high,' overall satisfaction (OS) improves from 'high' to 'very high,' and parts availability (PA) decreases from 'very high' to 'medium,' then the PromoterStatus is likely to change from 'Passive' to 'Promoter.'"*

This indicates that while parts availability may decrease slightly, improvements in key customer satisfaction metrics, such as overall satisfaction and repeat customer likelihood, are sufficient to drive positive outcomes in customer promotion status. Certain attributes, like parts availability, may not weigh as heavily in customer promotion when satisfaction metrics and repeat business likelihood are highly favorable. This emphasizes that customer satisfaction can offset minor declines in some operational factors.

7 Discussion and conclusion

This study introduced a correlation-based vertical partitioning method for distributed action rule discovery, designed to address the limitations of random partitioning and unstructured approaches. Through comparative analysis, we demonstrated that our method achieves a

balance between computational efficiency and rule quality. While the random partitioning approach offered shorter individual execution times, its variability required multiple iterations to produce reliable results. In contrast, the correlation-based method consistently generated robust and interpretable action rules in a single iteration, with higher lightness and precision than the random and base methods.

By clustering attributes based on their correlations, our method enhances both the relevance and consistency of the derived rules, offering significant advantages in scenarios where actionable insights are critical. The structured nature of this approach reduces redundancy and variability, making it a scalable and effective solution for large-scale datasets.

7.1 Relationship to counterfactual explanations

A natural connection exists between action rules and counterfactual explanations, which has implications for explainability and decision-making. While action rules provide a prescriptive framework for identifying changes that can improve an outcome, counterfactual explanations are typically used to explain why an outcome occurred and how it might have been different under alternate conditions.

For instance, an action rule may suggest that increasing customer engagement and reducing response time can improve satisfaction levels. A counterfactual explanation, on the other hand, would state that if engagement had been higher and response time shorter, the customer's satisfaction level would have been different.

Both frameworks share the goal of understanding and influencing decisions, with action rules being more aligned with *prescriptive analytics* and counterfactuals serving a role in *explainability and fairness* in machine learning models. Future work could explore how integrating counterfactual reasoning into action rule discovery might lead to improved interpretability and actionable insights.

7.2 Limitations

The correlation-based method, while efficient and consistent, introduces some computational overhead during the initial clustering step. As datasets scale, this may require additional resources. Additionally, the method's effectiveness may diminish if attributes within clusters are weakly correlated, potentially affecting rule interpretability.

7.3 Practical and research implications

The correlation-based vertical partitioning method introduced in this work offers not only computational benefits but also broader practical and research relevance. From a practical standpoint, the ability to generate high-quality, interpretable action rules at scale has direct implications for domains where actionable insights drive critical decisions. In healthcare, for example, action rules could be used to identify which modifiable factors—such as treatment frequency, patient follow-up methods, or dosage levels—most effectively lead to improved health outcomes. In educational contexts, action rules can support data-driven personalization by revealing which instructional adjustments may help students transition from struggling to successful performance. This work also holds relevance for therapeutic settings. In the domain of music therapy, for example, action rules can help identify which combinations of musical features and contextual factors are most associated with positive therapeutic out-

comes for different populations. In marketing and customer service, the method can uncover combinations of service attributes that are most likely to convert passive users into loyal customers, providing operational strategies grounded in empirical data.

From a research perspective, this work contributes to the growing field of interpretable machine learning, offering a structured and reproducible framework for action rule discovery in high-dimensional data settings. The method's use of correlation-based clustering provides a systematic way to reduce attribute complexity while preserving meaningful relationships, supporting the generation of consistent and relevant rule sets. In addition, the prescriptive nature of action rules naturally complements existing work on counterfactual reasoning, and future research may benefit from integrating these two paradigms to enhance both explanatory power and intervention planning. The method may also be extended to related areas such as causal inference, specification mining, and explainable recommendation systems, particularly in domains that require transparent, justifiable decision-making processes. Overall, this work provides a foundation for further exploration of interpretable and scalable approaches to actionable knowledge discovery.

7.4 Future work

Future research can explore adaptive partitioning techniques, integration with non-linear correlation measures, or hybrid strategies combining the strengths of structured and random partitioning. Such advancements have the potential to further enhance the efficiency and versatility of action rule discovery, broadening its applicability across diverse domains.

Hybrid approaches that combine the efficiency of correlation-based partitioning with the exploratory nature of random partitioning also hold promise for improving performance on certain types of datasets. An area to explore for improvement is the process of combining action rules. As the number of rules increases, the current combination method becomes less efficient, introducing bottlenecks that can hinder overall performance. Developing more scalable and efficient methods for the combination of rules would greatly enhance the applicability of this approach, particularly for data sets with a high volume of rules.

Another promising direction is the incorporation of alternative interestingness measures beyond traditional support and confidence. While these two metrics are widely used in action rule discovery for their interpretability, prior work in rule-based specification mining and pattern discovery has highlighted their limitations in capturing all aspects of rule quality or significance (Le & Lo, 2015; Hämäläinen & Webb, 2019). Measures such as lift, leverage, or odds ratio may help uncover more impactful rules, while statistically sound evaluation frameworks could reduce the risk of false discoveries. Integrating these ideas into action rule discovery may enhance both the robustness and informativeness of the derived rules.

While the proposed correlation-based vertical partitioning method has demonstrated improvements in efficiency and rule quality, an important direction for future research is evaluating its scalability on large-scale datasets. A more extensive experimental study could assess how the method performs when varying the number of examples and attributes, particularly in high-dimensional settings.

Another important direction for future research is evaluating the generalizability of the proposed method by applying it to publicly available benchmark datasets. While the current study was constrained to proprietary data, future work could involve testing the method on well-known datasets in the domains of healthcare, marketing, or education to assess how it performs across different domains. Such benchmarking would provide deeper insights

into the robustness, transferability, and practical impact of the approach in a wider range of scenarios.

Future work could involve:

- Conducting experiments with significantly larger datasets to analyze the computational efficiency and scalability of the method.
- Conducting experiments using publicly available benchmark datasets.
- Evaluating the impact of dataset size on rule quality, including metrics such as precision, coverage, and processing time.
- Investigating optimizations, such as adaptive partitioning strategies, to improve scalability in distributed environments.

By performing these experiments, we can further validate the robustness of the proposed method and explore potential enhancements for handling large-scale data more effectively.

Acknowledgements We would like to acknowledge the use of the ChatGPT-4 language model by OpenAI for assistance in composing certain sections of this paper, including help with conciseness, overall organization, and clarity of the text. Additionally, we acknowledge both ChatGPT and Grammarly for support in spellchecking and grammar checking.

Author Contributions A.B.: Conceptualization, Software, Investigation, Formal analysis, Validation, Writing–original draft, Writing–review & editing. Z.W.R.: Conceptualization, Supervision, Data curation, Resources, Writing–review & editing. All authors reviewed the manuscript.

Funding Open access funding provided by the Carolinas Consortium. No funding was received for conducting this study.

Data Availability No datasets were generated or analysed during the current study.

Declarations

Competing Interests The authors declare no competing interests.

Funding No funding was received for conducting this study.

Ethics Approval and Consent to Participate The study did not require ethical approval.

Conflicts of Interest No conflict of interest.

Materials Availability The materials are not publicly available.

Code Availability The code is not publicly available.

References

Agrawal, R., Imieliński, T., Swami, A. (1993). Mining association rules between sets of items in large databases. In: *Proceedings of the 1993 ACM SIGMOD international conference on management of data*, pp. 207–216. ACM, . https://doi.org/10.1145/170035.170072

Akoglu, H. (2018). User's guide to correlation coefficients. *Turkish Journal of Emergency Medicine, 18*(3), 91–93. https://doi.org/10.1016/j.tjem.2018.08.001

Ammons, G., Bodík, R., & Larus, J. R. (2002). Mining specifications. *ACM SIGPLAN Notices, 37*(1), 4–16. https://doi.org/10.1145/565816.50327

Bagavathi, A., Tripathi, A., Tzacheva, A.A., Ras, Z.W. (2018). Actionable pattern mining—a scalable data distribution method based on information granules. In: *2018 17th IEEE international conference on machine learning and applications (ICMLA)*, pp. 32–39. https://doi.org/10.1109/ICMLA.2018.00013 . IEEE

Bazan, J.G., Szczuka, M. (2005). The rough set exploration system. In: Peters, J.F., Skowron, A. (eds.) Transactions on Rough Sets III. Lecture Notes in Computer Science, vol. 3400, pp. 37–56. Springer. https://doi.org/10.1007/11427834_2

Benedict, A., Ras, Z.W. (2024). Action rules discovery: Leveraging attributes correlation based vertical partitioning. In: *Foundations of Intelligent Systems, ISMIS'2024, LNAI, Vol. 14670*, pp. 285–295. https://doi.org/10.1007/978-3-031-62700-2_25 . Springer

Benedict, A. C., & Ras, Z. W. (2024). Distributed action-rule discovery based on attribute correlation and vertical data partitioning. *Applied Sciences, 14*(3), 1270. https://doi.org/10.3390/app14031270

Borgelt, C., Kruse, R. (2002). Induction of association rules: Apriori implementation. In: Härdle, W., Rönz, B. (eds.) Compstat: Proceedings in Computational Statistics, pp. 395–400. Physica-Verlag. https://doi.org/10.1007/978-3-642-57489-4_59

Cecconi, A., De Giacomo, G., Di Ciccio, C., Maggi, F. M., & Mendling, J. (2022). Measuring the interestingness of temporal logic behavioral specifications in process mining. *Information Systems, 107*, Article 101920. https://doi.org/10.1016/j.is.2021.101920

Chatterjee, S., Tzacheva, A.A., Ras, Z. (2024). Scalable action mining hybrid method for enhanced user emotions in education and business domain. *International Journal on Cybernetics & Informatics (IJCI)13*, 31. https://doi.org/10.5121/ijci.2024.130104

Duan, Y., & Ras, Z. W. (2022). Recommendation system for improving churn rate based on action rules and sentiment mining. *International Journal of Data Mining, Modelling and Management, 14*(4), 287–308. https://doi.org/10.1504/IJDMMM.2022.126665

Geng, L., Hamilton, H.J. (2006). Interestingness measures for data mining: A survey. *ACM Computing Surveys38*(3).https://doi.org/10.1145/1132960.1132963

Hämäläinen, W., & Webb, G. I. (2019). A tutorial on statistically sound pattern discovery. *Data Mining and Knowledge Discovery, 33*(2), 325–377. https://doi.org/10.1007/s10618-018-0590-x

He, Z., Xu, X., Deng, S., & Ma, R. (2005). Mining action rules from scratch. *Expert Systems with Applications, 29*(3), 691–699. https://doi.org/10.1016/j.eswa.2005.04.031

Le, T.-H., Lo, D. (2015). Beyond support and confidence: Exploring interestingness measures for rule-based specification mining. In: *Proceedings of the 22nd IEEE international conference on software analysis, evolution, and reengineering (SANER)*, pp. 331–340. IEEE. https://doi.org/10.1109/SANER.2015.7081843

Lemieux, C., Park, D., Beschastnikh, I. (2015). General ltl specification mining (t). In: *2015 30th IEEE/ACM international conference on automated software engineering (ASE)*, pp. 81–92. https://doi.org/10.1109/ASE.2015.71 . IEEE

Leno, V., Dumas, M., Maggi, F. M., La Rosa, M., & Polyvyanyy, A. (2020). Automated discovery of declarative process models with correlated data conditions. *Information Systems, 89*, Article 101482. https://doi.org/10.1016/j.is.2019.101482

Maggi, F. M., Di Ciccio, C., Di Francescomarino, C., & Kala, T. (2018). Parallel algorithms for the automated discovery of declarative process models. *Information Systems, 74*, 136–152. https://doi.org/10.1016/j.is.2017.12.002

Navathe, S., Ceri, S., Wiederhold, G., & Dou, J. (1984). Vertical partitioning algorithms for database design. *ACM Transactions on Database Systems (TODS), 9*(4), 680–710. https://doi.org/10.1145/1994.2209

Powell, L., Gelich, A., & Ras, Z. W. (2021). How to raise artwork prices using action rules, personalization and artwork visual features. *Journal of Intelligent Information Systems, 57*, 2021. https://doi.org/10.1007/s10844-021-00660-x

Ras, Z.W. (2022). Reduction of hospital readmissions. *Advances in Clinical and Experimental Medicine 31*(1), 5–8. https://doi.org/10.17219/acem/144413

Ras, Z.W., Wieczorkowska, A. (2000). Action-rules: How to increase profit of a company. In: *Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD)*. Lecture Notes in Computer Science, vol. 1910, pp. 587–592. Springer. https://doi.org/10.1007/3-540-45372-5_70

Ras, Z., & Dardzinska, A. (2011). From data to classification rules and actions. *International Journal of Intelligent Systems, 26*, 2011. https://doi.org/10.1002/int.20485

Su, P., Mao, W., Zeng, D., & Zhao, H. (2012). Mining actionable behavioral rules. *Decision Support Systems, 54*(1), 142–152. https://doi.org/10.1016/j.dss.2012.04.013

Sýkora, L., Kliegr, T. (2020). Action rules: Counterfactual explanations in python. In: *CEUR workshop proceedings*, vol. 2644, pp. 28–41. https://ceur-ws.org/Vol-2644/paper36.pdf

Tarnowska, K. A., Bagavathi, A., & Ras, Z. W. (2022). High-performance actionable knowledge miner for boosting business revenue. *Applied Sciences, 12*(23), 12393. https://doi.org/10.3390/app122312393

Tarnowska, K. A., & Ras, Z. W. (2021). Nlp-based customer loyalty improvement recommender system (clirs2). *Big Data and Cognitive Computing, 5*(1), 4. https://doi.org/10.3390/bdcc5010004

Tsay, L.-S., Ras, Z.W. (2003). Discovering extended action-rules (system dear). In: Kłopotek, M.A., Wierzchoń, S.T., Trojanowski, K. (eds.) Intelligent Information Processing and Web Mining. Advances in Soft Computing, vol. 22, pp. 293–300. Springer, . https://doi.org/10.1007/978-3-540-36562-4_31

Tsay, L.-S., & Ras, Z. W. (2005). Action rules discovery: System dear2, method and experiments. *Journal of Experimental and Theoretical Artificial Intelligence, 17*(1), 1–25. https://doi.org/10.1080/09528130512331315855

Tzacheva, A. A., Bagavathi, A., & Ganesan, P. D. (2016). Mr-random forest algorithm for distributed action rules discovery. *International Journal of Data Mining & Knowledge Management Process (IJDKP), 6*(5), 15–30. https://doi.org/10.5121/ijdkp.2016.6502

Yang, J., Evans, D., Bhardwaj, D., Bhat, T., Das, M. (2006). Perracotta: Mining temporal api rules from imperfect traces. In: *Proceedings of the 28th international conference on software engineering (ICSE)*, pp. 282–291. https://doi.org/10.1145/1134285.1134325

Zhang, S., & Wu, X. (2011). Fundamentals of association rules in data mining and knowledge discovery. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 1*(2), 97–116. https://doi.org/10.1002/widm.10

Zhao, Q., Bhowmick, S.S. (2003). Association rule mining: A survey. *Nanyang Technological University, Singapore135*, 18. Available at https://personal.ntu.edu.sg/assourav/Unpublished/UP-ARMSurvey.pdf