Explaining Black-Box Clustering Pipelines With Cluster-Explorer

Sariel Ofek
Bar-Ilan University
sariel.tutay@live.biu.ac.il

Amit Somech
Bar-Ilan University
somecha@cs.biu.ac.il

ABSTRACT

Explaining the results of clustering pipelines by unraveling the characteristics of each cluster is a challenging task, often addressed manually through visualizations and queries. Existing solutions from the domain of Explainable Artificial Intelligence (XAI) are largely ineffective for cluster explanations, and interpretable-by-design clustering algorithms may be unsuitable when the clustering algorithm does not fit the data properties.

To bridge this gap, we introduce Cluster-Explorer, a novel explainability tool for black-box clustering pipelines. Our approach formulates the explanation of clusters as the identification of concise conjunctions of predicates that maximize the coverage of the cluster's data points while minimizing separation from other clusters. We achieve this by reducing the problem to generalized frequent-itemsets mining (gFIM), where items correspond to explanation predicates, and itemset frequency indicates coverage. To enhance efficiency, we leverage inherent problem properties and implement attribute selection to further reduce computational costs. Experimental evaluations on a benchmark collection of 98 clustering results demonstrate the superiority of Cluster-Explorer in both explanation quality and execution times compared to XAI baselines.

PVLDB Reference Format:
Sariel Ofek and Amit Somech. Explaining Black-Box Clustering Pipelines With Cluster-Explorer. PVLDB, 18(5): 1495 - 1508, 2025.
doi:10.14778/3718057.3718075

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/analysis-bots/cluster-explorer.

1 INTRODUCTION

Cluster analysis is an important data mining tool widely used to segment data points into meaningful groups (clusters) in an unsupervised manner, without the need for labeled data.

Similar to the development of a machine learning (ML) predictive model, data scientists optimize clustering outcomes by employing *clustering pipelines*—a series of data preprocessing and preparation steps (e.g., scaling, transformations, dimensionality reduction) followed by the application of a clustering algorithm such as K-means, spectral clustering, or affinity propagation. The choice of algorithm often depends on the data properties and application domain [19, 64]. While the results of these algorithms can be easily visualized on a two-dimensional plane (see Figure 1), allowing users to inspect how well the data points are separated, interpreting the *meaning* of the segmentation and understanding the characteristics of each cluster is challenging. This process often requires users to manually perform additional analytical queries and subsequent data visualizations on the clustered data.

Row ID	Age	Edu.num	Relationship	Gender	. . .	Hrs-per-week	Income	Cluster
124	25	7	Unmarried	Male	. . .	40	$\leq 50K$	0
32	41	10	Unmarried	Female	. . .	50	$\geq 50K$	0
53	34	12	Unmarried	Male	. . .	50	$\geq 50K$	0
. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .
342	36	3	Husband	Male	. . .	50	$\geq 50K$	1
521	40	3	Husband	Male	. . .	50	$\leq 50K$	1
5631	45	5	Wife	Female	. . .	60	$\geq 50K$	1
. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .
39	46	12	Wife	Female	. . .	30	$\leq 50K$	2
938	33	15	Husband	Male	. . .	60	$\geq 50K$	2
693	36	14	Husband	Male	. . .	50	$\geq 50K$	2
. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .

Table 1: Adult dataset sample with cluster labels

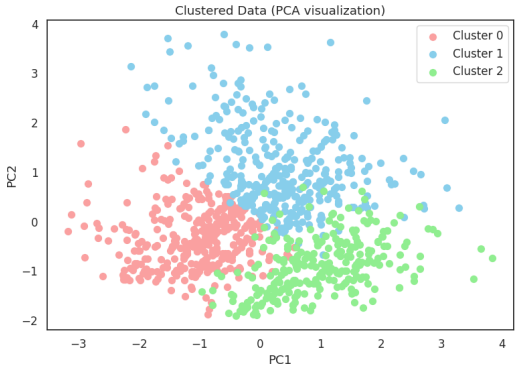


Figure 1: Clustering results visualization (Adult dataset)

Example 1.1. Consider Clarice, a data analyst examining the well-known "Adult" income dataset[1], which contains demographic information on individuals alongside their income (see Table 1 for a sample). Clarice employs a clustering pipeline that includes one-hot encoding of categorical data, Z-scaling of numerical columns, and dimensionality reduction using PCA [16]. She then applies an agglomerative (hierarchical) clustering algorithm [43] to the processed data and visually examines the results, as illustrated in Figure 1. She sees that the data points are fairly segmented into three clusters. However, it is still unclear clear what the characteristics of each cluster are. Specifically, what attributes are shared among points within the same cluster, and what properties differentiate the clusters from one another? □

This work is licensed under the Creative Commons BY-NC-ND 4.0 International License. Visit https://creativecommons.org/licenses/by-nc-nd/4.0/ to view a copy of this license. For any use beyond those covered by this license, obtain permission by emailing info@vldb.org. Copyright is held by the owner/author(s). Publication rights licensed to the VLDB Endowment.
Proceedings of the VLDB Endowment, Vol. 18, No. 5 ISSN 2150-8097.
doi:10.14778/3718057.3718075

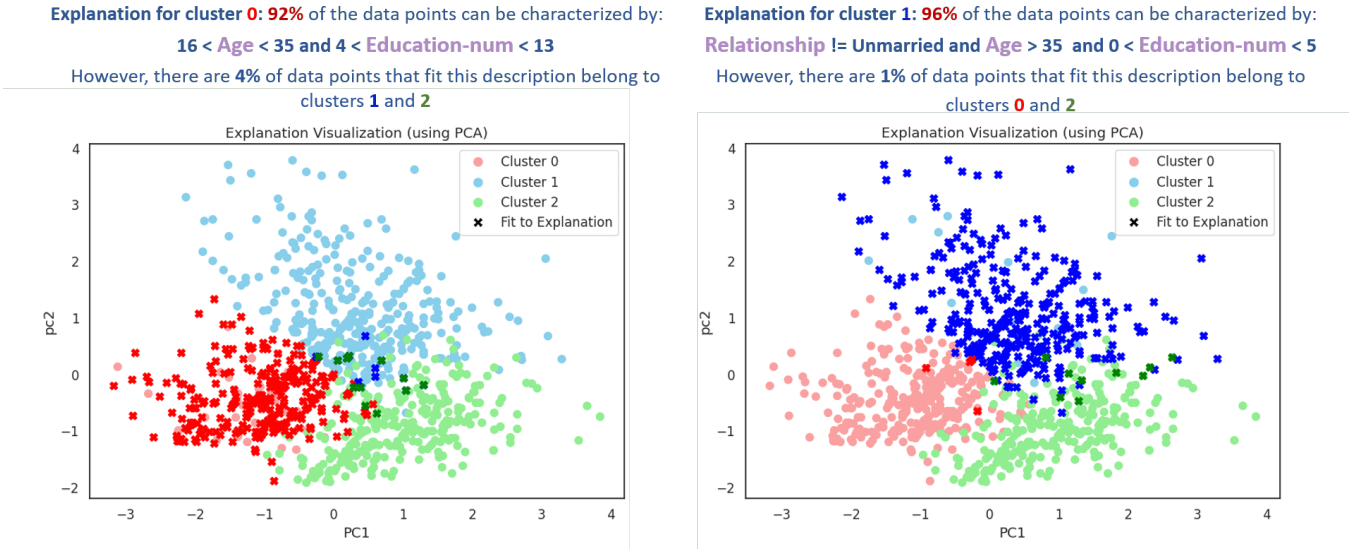


Figure 2: Example cluster explanations generated by Cluster-Explorer

Previous work [37, 40, 47, 48, 53, 56] in the domain of Explainable Artificial Intelligence (XAI) has primarily focused on post-hoc explanations of the outcome of supervised models. This is often achieved by calculating importance scores for features [32, 71] or feature-value combinations [40, 47, 50]. To apply these methods for interpreting cluster results, one must fit an auxiliary supervised model on the clustering labels and aggregate the explanations for each cluster, which is a nontrivial task. Another line of research focuses on interactive, visual tools for cluster analysis [8, 33, 36], enabling the construction and basic evaluation of clustering pipelines without the need for coding or SQL expertise. Closer to our work, systems such as [23, 25, 34, 42] recognizes the importance of explaining clusters and proposes clustering algorithms that are *interpretable by design*. However, it is widely accepted [19, 64] that each clustering algorithm is suitable for a specific data domain and properties. Therefore, there is a significant need for a framework that can produce explanations for any given clustering pipeline.

To this end, we present Cluster-Explorer, a system for post-hoc explanations for black-box clustering pipelines. Given the original dataset and the clustering pipeline results, Cluster-Explorer automatically generates coherent explanations that characterize each cluster. We define a cluster explanation as a conjunction of predicates and use an efficient algorithm to generate a set of explanations for each cluster that optimizes the following criteria, based on XAI Explanation principles [41, 61]: (1) high *cluster coverage* — the explanation should describe as many of the cluster's data points as possible; (2) low *separation error* — the explanation should apply to a minimal number of data points from other clusters; and (3) high *conciseness* — the explanation should be brief, comprising a minimal number of predicates, to ensure coherency and applicability [41].

Example 1.2. Figure 1 depicts two example cluster explanations generated by Cluster-Explorer for the clustering pipeline results described in Example 1.1. Cluster 0 (left hand side) is characterized by individuals with 'Age' between 16 and 35 and 'Education-num' between 4 and 13 (i.e., from middle school education level up to a bachelor's degree). Note that this explanation is not "perfect"—it covers 92% of the cluster points, but 4% of the data points it covers belong to different clusters (as indicated by the X marks on the left-hand side of Figure 1).

The explanation for Cluster 1 is different, primarily characterized by older individuals over 35, with a middle school (or lower) education level, and a relationship status that is not 'unmarried'. This explanations is slightly longer, but covers 96% of the cluster's data points with only a 1% error. □

To efficiently generate such explanations, we use a reduction to the problem of generalized frequent itemsets mining [31, 35, 54, 55] (gFIM). A gFIM algorithm operates on a *transactional* dataset, where each transaction comprises a set of discrete items, and the items are associated with categories in an additionally provided taxonomy. The result is a set of *generalized* frequent itemsets, containing either items or categories that include them. Intuitively, in our problem, the items are equivalent to explanation predicates, and the frequency of itemsets corresponds to the explanations' *coverage*.

Before employing the gFIM algorithm, we transform the raw data into a set of *augmented transactions*, with the goal of increasing and enriching the set of predicates that can be used in the cluster explanations, beyond the raw values. Specifically, we use multiple binning methods (e.g., equal width, 1-D clustering, tree-based binning, etc.) for each numeric attribute and further augment categorical values with negation predicates for the rest of the values not appearing in the row. Once the data is transformed, we organize all numerical bins in a taxonomy of intervals. We then execute the gFIM algorithm separately for each cluster, using the taxonomy to eliminate the possibility of overlapping predicates and efficiently find minimal-size sets of predicates that maximize the coverage of each cluster's data points. To obtain the final set of explanations for each cluster, we further process the gFIM output by filtering out explanations with high separation error, and calculating a set of Pareto Optimal [6, 9] explanations, where each explanation demonstrates an optimal trade-off between the three criteria for explanation quality.

However, gFIM algorithms are known for their lack of scalability, as their cost can be exponential w.r.t. the number of items. In Cluster-Explorer, we tackle this issue in two ways: First, since

we aim for explanations with high coverage (accounting for the majority of a cluster's data points) and small size, we leverage these natural properties to restrict the execution of the gFIM algorithm. By confining the gFIM algorithm to highly frequent itemsets of small size, we significantly accelerate running times, as most infrequent itemsets are pruned in the early stages of execution.

Second, we introduce a simple yet highly effective attribute selection optimization based on feature importance calculation [71] using a set of decision tree models fitted separately for each cluster. By selectively limiting the computation to promising attributes, the gFIM algorithm operates on significantly fewer items, a crucial factor affecting its performance [65].

An extensive set of experiments was conducted to evaluate Cluster-Explorer. We first devised a benchmark dataset containing 98 clustering instances, resulting from the execution of 16 different clustering pipelines with 5 different algorithms on 19 source datasets. We further examined the quality of the clusters by calculating the *silhouette coefficient* [49], filtering out poor clustering pipeline results. We compared the quality of the explanations according to coverage, separation error, and conciseness, as well as the running time of Cluster-Explorer, against four different baseline approaches from the domain of XAI.

Our results show that the explanations generated by Cluster-Explorer are superior to those of the baselines in terms of both quality and running times. Additionally, we demonstrate that our attribute selection optimization improves running times by an average of 14.4X, with a negligible decrease in explanation quality.

A prototype of our solution, wrapped with a user interface, was recently demonstrated in [58]. The accompanying short paper briefly presents the problem and outlines our solution, but it lacks significant algorithmic and optimization details, as well as an experimental analysis which are provided in this paper.

Our main contributions in this paper include:

- We introduce Cluster-Explorer, a framework for post-hoc explanations of black-box clustering pipelines.
- We develop an efficient algorithm based on a careful reduction to generalized frequent itemsets mining [54], combined with a predicate-augmentation process and a dedicated attribute selection method, enabling Cluster-Explorer to efficiently mine concise explanations with good coverage of each cluster's data points.
- We create a benchmark dataset of 98 clustering results, curated from 16 clustering pipelines using 5 different algorithms and 19 datasets (publicly available in [57]).
- We implement a prototype of Cluster-Explorer, also publicly available in [57], and conduct extensive experiments demonstrating the superiority of our approach in explanation quality and running time.

2 RELATED WORK

Cluster Analysis and Algorithms. A plethora of unsupervised clustering algorithms have been proposed [18, 21, 38, 60, 69], each often suitable for different data properties and application domains [19, 64]. As with many data mining and machine learning processes, data preparation and preprocessing steps, such as imputing missing values, scaling, and reducing data dimensionality, are crucial for optimizing the results of clustering algorithms [3, 26, 68].

However, as noted, characterizing and understanding the resulted clusters is a challenge, often requires the user to employ subsequent analytical operations to explore the differences between the data points in each resulting cluster. Cluster-Explorer is specifically designed to address this challenge by explaining the results of black-box clustering pipelines, which may use any clustering algorithm, using a set of coherent and concise rule-like explanations generated for each cluster.

Interactive Visual Interfaces for Cluster Analysis. An adjacent line of work focuses on developing visual, interactive tools for cluster analysis, contributing to broader efforts in designing data exploration interfaces that eliminate the need for coding skills or SQL expertise [67]. For instance, tools like [8, 33, 36] allow users to interactively refine the clustering pipeline by selecting different algorithms and dimensionality reduction techniques, while providing basic descriptive statistics and annotations for the resulting clusters. Other works, such as [10, 63], specifically help users investigate the outcomes of dimensionality reduction methods, offering visual tools to inspect inaccuracies and understand the similarities between points in the low-dimensional projection space.

Unlike these approaches, Cluster-Explorer tackles a complementary task: explaining each resulting cluster by discovering a concise set of rules that tightly characterizes the cluster. Cluster-Explorer can be used side-by-side with visual cluster analysis interfaces, enriching the interactive process by offering robust explanations to help users refine their cluster analysis.

XAI, Explainable ML. Numerous previous works propose solutions for explaining the predictions of supervised ML models. While some approaches focus on developing explainable-by-design ML models [4] (see more below), a prominent line of research emphasizes *post-hoc* analysis of model predictions, where the explainer does not require access to the internal workings of the model [40, 47, 48, 53, 56] (see [37] for a survey). . A key approach in this subfield explains individual predictions by assessing *importance* [40, 47, 50] for each feature-value pair of a data point x regarding the prediction $M(x)$. Such explanations are called *local*, while *global* explanations [32, 71] measure feature importance for the model's overall behavior. Closer to our work [17] introduces a feature importance tool for clustering, akin to [40]. Another line of research focuses on extracting *decision rules* from complex ML models. Works like [27, 48] generate if-then rules for individual predictions, highlighting how specific attribute changes influence outcomes, whereas global explanation approaches [12, 22] mine rules from ensembles [20, 51] to enhance model transparency. A recent study [5] applies Anchor [48] explanations to cluster samples.

However, as demonstrated in our experiments, aggregating *local* explainers [40, 48] and extracting rule-based cluster explanations from ensemble models trained on cluster labels [20, 51] yield suboptimal results due to overfitting [30, 66] and high computational costs. In contrast, Cluster-Explorer does not rely on supervised ML models for explanations and efficiently generates high-quality, rule-like explanations.

Interpretable-by-design Clustering Algorithms. Numerous previous works recognize the difficulty in interpreting clustering results [23, 25, 34, 42] and address this issue by suggesting clustering algorithms that are interpretable by design. For example, [23, 42] propose approximating the K-means algorithm using decision trees; [34] introduces a neural network formulation of K-means, allowing the use of explainability techniques for networks (e.g., class activation maps [70]); and [25] presents a specialized solution for clustering knowledge graph entities, which performs an embedding-based clustering process combined with rule-based explanation mining to cover the majority of entities in each cluster.

However, all of these methods are tailored to a specific, single clustering algorithm, which, as mentioned above, may be ineffective for some datasets and applications [19, 64]. In contrast, Cluster-Explorer can produce explanations for any given clustering pipeline, supporting a variety of clustering algorithms.

3 MODEL & PROBLEM DEFINITION

3.1 Data model, Explanation Candidates

We next describe a data model for clustering pipelines then define candidate explanations for clustering results.

Clustering Pipeline. We assume an input dataset $D = \langle X, A \rangle$, containing n data points $X = x_1, \ldots, x_n$, projected over m attributes $A = a_1, \ldots, a_m$. We denote by x_i the i-th data point, and by $x_{i,a}$ the projection of x_i over attribute a. A clustering pipeline is then defined as a series of data transformations applied to D (e.g., normalization, null-value imputation, one-hot encoding, dimensionality reduction, etc.). These transformations result in a modified dataset $D' = \langle X, A' \rangle$, where the data points $x_1, \ldots, x_n$ are projected onto a transformed attribute space $A' = a_1, \ldots, a_{m'}$. A clustering algorithm is then applied to D', resulting in a clustering mapping function $CL : X \rightarrow C$, which associates each data point x_i with a cluster $c \in C$, where C is a set of cluster labels.

Cluster Explanation Candidates. Following XAI works for producing explanations for ML models [27, 48], which highlight the usefulness of rules-like explanations, we define an explanation, denoted E, as a conjunction of predicates $E = \{P_1 \wedge P_2 \ldots P_l\}$. The predicates are of the form $P \coloneqq \langle a, op, V \rangle$, where $a \in A$, op is an operator (e.g., <, >, 'contains', 'between', 'not', ...), and V is a set of literal values. Given a data point $x \in X$, we say that an explanation E *holds* for x if x satisfies all the predicates in E. Specifically, $E(X) = \text{true} \iff \forall P \in E, P(x) = \text{true}$.

When considering explanations in the context of clustering results, naturally, a subpar explanation for cluster c, denoted E_c, may hold for data points labeled as c as well as for points assigned to different clusters $c' \in C$ with $c' \neq c$. In Section 3.2, we describe three criteria for selecting effective cluster explanations.

Example 3.1. Consider again Table 1, with a sample of the raw Adult dataset, alongside resulted cluster labels, as described in Example 1.1. See that, for example, the first three rows in the table (IDs 124, 32, and 53) are labeled as Cluster 0. Now, three candidate explanations for Cluster 0 are depicted in Table 2 (ignore, for now, the three right-most columns). Explanation E_0^2, for example, comprises of two predicates:

$P_1 \coloneqq \langle$'age', $between$, $(16, 35)\rangle$, and

$P_2 \coloneqq \langle$'education-num', $between$, $(4, 13)\rangle$.

Out of the rows in Table 1 that indeed belong to Cluster 0, see that Explanation E_0^2 holds for Rows 124 and 53, yet is not true for Row 32 (having *age* value of 41). By contrast, Explanation E_0^1 holds for all three rows (IDs 124, 32, and 53) yet unfortunately – it also holds for rows 5631 and 39, which are assigned to different clusters (Cluster 1 and Cluster 2). □

We next define measures for evaluating explanation candidates.

3.2 Quality Measures for Cluster Explanations

The question of what constitutes a 'good' explanation has been investigated in various different domains such as cognitive science, philosophy and psychology. Relying on this vast body of research, works e.g. [41, 61] suggest that in the context of XAI, a *good* explanation is primarily *contrastive* (i.e., why event P happened *instead* of an event Q), but also *simple*, *coherent*, and *truthful*.

In Cluster-Explorer, we adapt these criteria to the use case of explaining clustering results and develop corresponding quality metrics for explanations. Given a cluster $c \in C$, an ideal explanation E_c should have (1) high *coverage* of the points in c, while maintaining a (2) low *separation* error. Specifically, the explanation must be valid for the majority of data points in cluster c, and invalid for data points associated with any other cluster $c' \in C$, where $c' \neq c$. The higher the scores with respect to (1) and (2), the more contrastive and truthful the explanation is. Similar measures to (1) and (2) have been previously proposed in the context of evaluating decision rules [27, 48], drawing parallels to the notions of precision and recall commonly used in supervised learning.

However, as noted previously, a naive explanation achieving perfect scores might consist of the union of all attribute-value pairs for each data point in c. Naturally, such naive explanations are be overly verbose and incoherent. To address this, we introduce a (3) *conciseness* measure, which considers the number of predicates in the explanation E_c. We next provide formal definitions for these three quality measures of explanations.

1. Cluster Coverage. Given a set of data points X, cluster labels C, and a clustering mapping function CL, the *coverage* of an explanation E_c (for cluster c) is defined as the ratio of the points in cluster c for which explanation E_c holds:

$$Coverage(E_c) \coloneqq \frac{|\{x \in X \mid E_c(x) = true \wedge CL(x) = c\}|}{|\{x \in X \mid CL(x) = c\}|}$$

2. Separation Error. This measure is defined as the ratio of points for which the explanation E_c holds, yet these points do not belong to cluster c:

$$SeparationErr(E_c) \coloneqq \frac{|\{x \in X \mid E_c(x) = True \wedge CL(x) \in C \setminus \{c\}\}|}{|\{x \in X \mid E(x) = true|}$$

3. Conciseness. Following [41], the length and simplicity of the explanations are important for its comprehension by the users. We therefore define the conciseness of an explanation to be the inverse number of predicates it contains:

$$Conciseness(E_c) \coloneqq \frac{1}{|\{P \mid P \text{ is a predicate in } E_c\}|}$$

The following example demonstrates the three measures measures for the explanations in Table 2.

Exp.num	Explanation Candidate	Cluster label	coverage	Separation Error	Conciseness
E_0^1	⟨'age',between,(16,48)⟩ ∧ ⟨'education-num',between,(4,13)⟩ ∧⟨'relationship',!=,Husband⟩	0	0.99	0.05	0.33
E_0^2	⟨'age',between,(16,35)⟩∧ ⟨'education-num',between,(4,13)⟩	0	0.95	0.04	0.5
E_0^3	⟨'age',between,(16,53)⟩ ∧ ⟨'hours-per-week',between,(10,72)⟩ ∧⟨'education-num',between,(4,14)⟩	0	0.88	0.04	0.33

Table 2: Example Candidate Explanations

Example 3.2. Consider Explanation E_0^1 for Cluster 0, as in Table 2. Assume that the number of data points belonging to Cluster 0 is 373, out of which 370 satisfy E_0^1. In addition, 20 other data points that belong to Clusters 1 and 2 also satisfy E_0^1. Calculating the scores for E_0^1 we obtain: $Coverage(E_0^1)) = \frac{370}{373} = 0.99$, $SeparationErr(E_0^1)) = \frac{20}{390} = 0.05$, and $Conciseness(E_0^1)) = \frac{1}{3} = 0.33$.

3.3 Problem Definition

There is a natural trade-off between the quality measures. For instance, an explanation obtaining a very high *coverage*, may have a lower *conciseness* score and higher *separation error*, whereas a highly *concise* explanation may fall short on *coverage*.

We therefore define the problem of generating cluster explanations as follows: Given user-defined thresholds for coverage, separation error, and conciseness, we aim to find a set of desired explanations for a cluster c, denoted by $\mathcal{E}_c^*$, such that each $E_c^* \in \mathcal{E}_c^*$ (1) meets the thresholds criteria, and (2) is *Pareto optimal* [9], meaning that there is no other explanation that surpasses E_c^* with respect to all three measures. Formally, the problem is defined as follows.

Definition 3.3 (Cluster Explanations Generation Problem). For data points X, a cluster $c \in C$, we define the set of desired explanations $\mathcal{E}_c^*$ using the following two criteria:

$$\begin{aligned} \forall E_c^* \in \mathcal{E}_c^*, Coverage(E_c^*) \geq \theta_{cov} \\ \wedge\ SeparationErr(E_c^*) \leq \theta_{sep} \\ \wedge\ Conciseness(E_c^*) \geq \theta_{con} \end{aligned} \tag{1}$$

$$\begin{aligned} \forall E_c^* \in \mathcal{E}_c^* \nexists E_c, Coverage(E_c) \geq Coverage(E_c^*) \\ \wedge SeparationErr(E_c) \leq SeparationErr(E_c^*) \\ \wedge Conciseness(E_c) \geq Conciseness(E_c^*) \end{aligned} \tag{2}$$

Example 3.4. Consider again the candidate explanations depicted in Table 2. We aim to generate $\mathcal{E}_0^*$, i.e., the set of optimal explanations for Cluster 0, with the coverage, separation, and conciseness thresholds defined as $\theta_{cov} = 0.8$, $\theta_{sep} = 0.05$, and $\theta_{con} = 0.33$. First, note that all explanations E_0^1, E_0^2, and E_0^3 meet the threshold criteria. Regarding Pareto optimality, Explanation E_0^1 surpasses E_0^2 with respect to coverage (0.99 compared to 0.95) but is inferior with respect to separation error (0.05 compared to 0.04) and conciseness (0.33 compared to 0.5). As for E_0^3, it is *dominated* by E_0^2, having the same separation error (0.04) yet better coverage and conciseness (0.95 compared to 0.88, and 0.5 compared to 0.33). Therefore, $\mathcal{E}_0^* = \{E_0^1, E_0^2\}$.

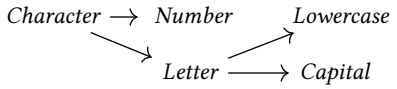
We next describe an efficient algorithm for generating the explanation set $\mathcal{E}_c^*$ for each cluster $c \in C$.

4 ALGORITHM

We next describe the cluster explanation generation process performed by Cluster-Explorer. The effectiveness and efficiency of our solution stem from a careful reduction to the problem of generalized frequent itemset mining [31, 35, 54, 55] (gFIM). In our context, gFIM is used to mine sets of predicates that concisely characterize the majority of a cluster's data points. We first provide a brief background on the gFIM problem and outline our approach, followed by a detailed discussion of each phase.

4.1 Background & Algorithm Outline

Generalized Frequent Itemsets Mining (gFIM). This problem extends the classic data mining problem of finding frequent itemsets (and association rules) in transactional data [2, 28]. Given a set of items $I = \{A, b, 1, 2\}$, let $T = \{[A, 1], [b, 2], [A, 2]\}$ be a set of transactions. Given a *support* (frequency) threshold of 2/3, we see that the itemsets $\{A\}$ and $\{2\}$ are frequent (i.e., occur in two out of three transactions). In the extended problem of *generalized* frequent itemsets mining, a taxonomy of of item's *categories* is also provided as input, then the mined generalized frequent itemsets may contain either an item or one of its associated categories. For example, given the following taxonomy for items I:



We can now obtain generalized frequent itemsets from T such as $\{Letter, 2\}$ and $\{A, Number\}$, both has a frequency of 2/3. Multiple algorithms [31, 35, 54, 55] can be used for efficiently mining such itemsets, given a support threshold and maximal desired itemset size. Naturally, the higher the input support threshold and the lower is the maximal itemset size – the better are the performance.

Cluster Explanations Algorithm Overview. Intuitively, in our problem, the items correspond to explanation predicates, and the frequency of itemsets is equivalent to the explanations' *coverage*. As we explain in Section 4.2, before employing the gFIM algorithm, we first transform the raw data into a set of augmented transactions. Each numeric attribute in a data point is binned using multiple binning strategies, and categorical values are augmented with negation predicates for values not appearing in the row. Once the data is transformed, we organize all numerical bins into an interval taxonomy based on their containment partial order. Our augmentation aims to increase and enrich the set of predicates that can be used in cluster explanations, extending beyond the original value domain.

As detailed in Section 4.3, we then apply the gFIM algorithm to the augmented transactions and interval taxonomy for each cluster. This allows us to efficiently dice the predicates hierarchy and

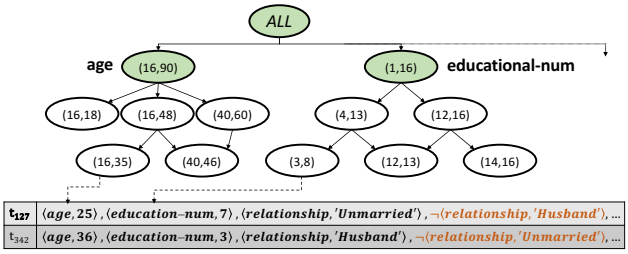


Figure 3: Example augmented transactions

exclude overlapping or suboptimal explanation predicates. Specifically, the mining process is accelerated by using a high support threshold, θ_{cov}, which indicates that explanations should cover the vast majority of the data points, and a relatively small maximal itemset size, as effective explanations should be *concise* [41]. The candidate explanations generated by the gFIM algorithm are further filtered to remove those with high separation error. We then select only the Pareto optimal explanations using the skyline operator [6].

Finally, in Section 4.4, we describe a simple yet highly effective optimization technique that reduces running times by limiting the number of attributes, a known factor that significantly affects the cost of gFIM when applied on relational data [24].

4.2 Transforming the raw data to a set of augmented transactions

Recall that a gFIM algorithm operates on transactional data, where each row contains a set of discrete items associated with a categories taxonomy. A straightforward transformation from relational data to transactional form involves treating each data point x_i as a set of attribute-value pairs, i.e., $t_i = \{\langle a, x_{i,a}\rangle \mid \forall a \in A\}$, where each item $\langle a, x_{i,a}\rangle$ represents an equality predicate $\langle a, =, x_{i,a}\rangle$.

However, this transactional representation is largely ineffective, as the items merely represent the raw data. This limitation restricts the ability to mine generalized patterns and, consequently, to generate effective cluster explanations.

To address this, we transform each data point into an augmented transaction that contains more generalized information. This transformation involves (1) creating an interval taxonomy for numeric values, corresponding with explanations' *range* predicates and (2) injecting negations of categorical values which will represent *inequality* predicates, allowing an explanation to characterize a cluster not only by what it includes but also by what it excludes. Our augmentation process is described in Algorithm 1. We next provide more details on the two segments of this process.

Constructing the intervals taxonomy. The purpose of the interval taxonomy is to generate effective *range* predicates for numeric attributes (e.g., *Age* $\geq$ 35 or *Education-num* between 4-13 years).

To achieve this, we employ multiple binning methods for each numeric attribute a, combining them into an interval taxonomy used as input for the gFIM algorithm. Each unique attribute-value pair $\langle a, x_{i,a}\rangle$ is then augmented with multiple intervals that contain it, allowing these intervals to be used as range predicates in a cluster explanation (in addition to the original values).

Let $A^N \subseteq A$ bet the set of numeric attributes in the dataset D. For each attribute $a \in A^N$ We first employ a predefined set of binning methods, denoted BIN, on X_a (the values of column a in X). Each

Algorithm 1: Generate Augmented Transactions

Input: Dataset $D = \langle X, A\rangle$; numeric and categorical attribute subsets $A^N, A^C \subset A$; a set of binning methods BIN
Output: Augmented transactions set T^D, intervals taxonomy $\mathcal{T}$

1 $\mathcal{T} \leftarrow$ Initialize DAG with root node $\langle ALL\rangle$;
2 **foreach** *numeric attribute* $a \in A^N$ **do**
3 $\quad \mathcal{B}_a \leftarrow \bigcup_{B\in BIN} B(X_a)$
4 $\quad \alpha_M = \arg\min_\alpha \{\alpha \mid [\alpha, \beta] \in \mathcal{B}_a\}$
5 $\quad \beta_M = \arg\max_\beta \{\beta \mid [\alpha, \beta] \in \mathcal{B}_a\}$
6 $\quad \mathcal{B}_a \leftarrow \mathcal{B}_a \cup [\alpha_M, \beta_M]$
7 $\quad \tau_a \leftarrow$ Initialize a DAG with nodes $\mathcal{B}_a$
8 $\quad$ **foreach** *intervals pair* $b, b' \in \mathcal{B}_a$ **do**
9 $\quad\quad$ **if** $b \prec b' \wedge \nexists b'' \in \mathcal{B}_a,\ b \prec b'' \prec b'$ **then**
10 $\quad\quad\quad$ Add edge from b to b' in τ_a
11 $\quad \mathcal{T} \leftarrow \mathcal{T} \cup \tau_a$; add an edge from $\langle ALL\rangle$ to the root of τ_A
12 **foreach** *data point* $x_i \in X$ **do**
13 $\quad t_i \leftarrow \{\langle a, x_{i,a}\rangle \mid \forall a \in A\}$
14 $\quad$ **foreach** *categorical attribute* $a \in A^C$ **do**
15 $\quad\quad$ **foreach** *distinct value* $v \in X_a$, *s.t.* $x_{i,a} \neq v$ **do**
16 $\quad\quad\quad t_i \leftarrow t_i \cup \{\neg\langle a, v\rangle\}$
17 $\quad T^D \leftarrow T^D \cup \{t_i\}$
18 **return** $T^D, \mathcal{T}$

binning method $B \in BIN$ splits X_a into a set of intervals, defined as $B(X_a) = \{[\alpha_1, \beta_1], [\alpha_2, \beta_2], \dots\}$, s.t. $\alpha_i < \beta_i$.

We combine multiple binning methods in our implementation of Cluster-Explorer: Equal-height, Equal-width, 1-D clustering [62], tree-based [39], and Optimal Binning [44], and support additional methods such as domain-specific and semantic binning [52].

Then, we construct an interval taxonomy for each attribute a using the following procedure: First, we unify all binning results $\mathcal{B}_a = B_1(X_a) \cup B_2(X_a) \cup \dots$ and arrange them in an *interval taxonomy*, built on the following strict partial order:

$$[\alpha_i, \beta_i] \prec [\alpha_j, \beta_j] \iff (\alpha_i \neq \alpha_j \vee \beta_i \neq \beta_j) \wedge (\alpha_i \leq \alpha_j \wedge \beta_i \geq \beta_j)$$

The partial order allows as to build a semi-lattice [13] structure for attribute a, denoted τ_a that we will use for building the full taxonomy for all numeric attributes. τ_a is a directed acyclic graph (DAG), in which the nodes are the intervals in $\mathcal{B}_a$, and an edge from b to b', $b, b' \in \mathcal{B}_a$ symbolizes that b is a *parent* of b', namely, that $b \prec b' \wedge \nexists b'' \; b \prec b'' \prec b'$. The root of τ_a is the maximal range composed of the minimal infimum and the maximal supremum of the intervals in $\mathcal{B}_a$, i.e. $[\alpha_M, \beta_M]$, s.t. $\alpha_M = \arg\min_\alpha \{\alpha \mid [\alpha, \beta] \in \mathcal{B}_a\}$ and $\beta_M = \arg\max_\beta \{\beta \mid [\alpha, \beta] \in \mathcal{B}_a\}$. In a similar manner we construct τ_a for each numeric attribute $a \in A_N$. We then artificially combine all individual attribute taxonomies τ_a to a unified taxonomy $\mathcal{T} = \tau_a^1 \cup \tau_a^2 \cup \dots$, by creating an artificial root node labeled *All*, and creating an edge from *All* to the root node of each τ_a^i.

The final taxonomy $\mathcal{T}$ is used in the gFIM algorithm to mine *generalized* frequent itemsets, each containing either a concrete item $\langle a, x_{i,a}\rangle$ or one of its ancestors in $\tau_a \subset \mathcal{T}$ (recall that item $\langle a, x_{i,a}\rangle$ is connected to all leaf nodes in $\tau_a \subseteq \mathcal{T}$ whose interval range contains the value $x_{i,a}$).

Augmented transactions with value negations. We next describe how we transform each data point x_i to an augmented transaction t_i (See Lines 12-17 in Algorithm 1).

Algorithm 2: Explanation Generation

Input: Augmented Transactions set T^D; Taxonomy $\mathcal{T}$; Cluster Mapping $CL : T^D \rightarrow C$; Thresholds $\theta_{cov}, \theta_{sep}, \theta_{con}$
Output: A set $\mathcal{E}_c^*$ of cluster explanations for each cluster

$EX_{\text{all}} \leftarrow$ Initialize results explanations dictionary
foreach $c \in C$ **do**
  $T_c^D \leftarrow \{t | t \in T^D \wedge CL(t) = c\}$
  $\mathcal{IS}_c \leftarrow gFIM\left(T_c^D, \mathcal{T}, \text{minsup} = \theta_{cov}, \text{maxsize} = \frac{1}{\theta_{con}}\right)$
  $\mathcal{E}_c \leftarrow \{\}$
  foreach $IS_c \in \mathcal{IS}_c$ **do**
    $E_c \leftarrow$ Convert IS^c to a conjunction of predicates
    if $SeparationErr(E_c) \leq \theta_{sep}$ **then**
      $\mathcal{E}_c \leftarrow \mathcal{E}_c \cup \{E_c\}$
  $\mathcal{E}_c^* \leftarrow SKYLINE_{E \in EX_\theta}(Coverage, SepError, Conciseness)$
  $EX_{\text{all}}[c] \leftarrow \mathcal{E}_c^*$
return EX_{all}

For each data point x_i, we first convert it to a set of attribute-value pairs $\{\langle a, x_{i,a}\rangle \mid \forall a \in A\}$. By now, items from numeric attributes are associated with their corresponding ancestors in the taxonomy $\mathcal{T}$. We then process items of categorical columns as follows: For each categorical attribute $a \in A^C$, we insert value negation items for all *other* values in the column X^a, in a process similar to *one-hot encoding*. Namely, For each $v \in X_a$, s.t. $v \neq x_{i,a}$, we add to t_i the item $\neg\langle a, v\rangle$. these injected value-negation items will represent *inequality predicates* in the output explanations.

For illustration, consider the following example.

Example 4.1. Figure 3 depicts an example of two augmented transactions with a corresponding interval taxonomy, following our running example. The transactions, t_{124} and t_{342}, correspond to data points x_{124} and x_{342} in the Adult dataset, as shown in Table 1. The augmented transactions include items for the attributes: *age*, *educational-num*, and *relationship* (other attributes are omitted for space constraints). The black-colored items represent a subset of the original attribute-value pairs. Note that the numeric items are connected (using dashed lines) to corresponding leaves in the interval taxonomy, shown in the upper part of Figure 3. The full taxonomy comprises two isolated sub-taxonomies, one for the attribute *age* and one for *educational-num* (others are omitted due to space limitations). The orange items in both t_{124} and t_{342} indicate value negation items for the *Relationship* attribute.

After transforming the remaining data points, the gFIM algorithm is applied to the augmented transactions to generate explanation candidates, as shown in Table 2. These candidates are then further processed to obtain an optimal set of cluster explanations (see Figure 2), as detailed below.

4.3 Generating Explanations with gFIM

We next describe how we mine cluster explanations using a gFIM algorithm, employed on the augmented transactions we generated as described above. Recall that a gFIM algorithm [31, 54], as described above, takes as input a set of transactions $T = t_1, t_2, \ldots$, each containing a set of discrete items, associated with some categories in an additionally provided item-category taxonomy. It also takes as input the maximal itemset size *maxsize*, and a *support* threshold *minsup*. The support of an itemset $IS \subseteq \mathcal{I}$ is defined by: $support(IS) = \frac{|\{t | t \in T \wedge IS \subseteq t\}|}{|T|}$.

The gFIM algorithm mines a set of frequent *generalized* itemsets, $\mathcal{IS}^*$, where each resulted itemset $IS^* \in \mathcal{IS}^*$, has $support(IS^*) \geq minsup$ and $|IS^*| \leq maxsize$.

We next detail our explanations generation process, in which we apply a gFIM algorithm on the augmented transactions, then process the resulted itemsets into effective cluster explanations. Refer to Algorithm 2 for pseudo code.

For each cluster $c \in C$, we first consider the transaction subsets T_c^D, which only contains transactions t_i that are labeled with cluster c. We then apply the gFIM algorithm on T_c^D, together with the intervals taxonomy $\mathcal{T}$, a minimal support of $minsup = \theta_{cov}$, and itemset size limit $maxsize = \frac{1}{\theta_{con}}$. The gFIM algorithm returns a set $\mathcal{IS}_c$ of generalized frequent items with a support value greater than *minsup* and size under *maxsize*.

We then further process the set $\mathcal{IS}_c$ in order to generate the optimal explanations $\mathcal{E}_c^*$ based on the following observation.

Observation 1. *Each $IS \in \mathcal{IS}^c$ is equivalent to an explanation candidate E_c, having $Coverage(E_c) \geq \theta_{cov}$ and $Conciseness(E_c) \geq \theta_{con}$*

Intuitively, in each itemset *IS*, singular items of the form $\langle a, x_{i,a}$ correspond to *equality predicates* of the form $\langle a, =, x_{i,a}\rangle$; generalized items $\langle a, [\alpha, \beta]$ are equivalent *range* predicates of the form $\langle a, \text{between}, [\alpha, \beta]\rangle$; and categorical value negations of the form $\neg\langle a, v\rangle$ are *inequality* predicates, i.e., $\langle a, \neq, v\rangle$.

Naturally, the coverage of E_c is higher than θ_{cov}, since the support of IS^c is guaranteed (by the gFIM algorithm) to be higher than θ_{cov}. Similarly, the conciseness score E_c is higher than θ_{con}, since we limit the maximal size of the itemset using $\frac{1}{\theta_{con}}$.

Let $\mathcal{E}^c$ be the candidate explanations generated from transforming the gFIM output $\mathcal{IS}^c$ to a conjunction of predicates as explained above. Then, in order too obtain the subset $\mathcal{E}_c^* \subseteq \mathcal{E}^c$ of optimal explanations for cluster c, we only need to filter the candidate explanations using the separation error threshold θ_{sep} then find the Pareto optimal explanations , w.r.t. coverage, separation error and conciseness. The latter step is done using the skyline operator [6] on the filtered candidate explanations set, $SKYLINE_{E \in \mathcal{E}^c}(Coverage, SepError, Conciseness)$.

The results of the skyline operator retrieves the final desired set of explanations $\mathcal{E}_c^*$ (see an example explanation for Cluster 0 and Cluster 1 from our running example in Figure 2).

Cost Discussion. The overall computational cost associated with the Cluster-Explorer explanation generation process primarily aligns with the cost of the gFIM algorithm. The preprocessing phase (transaction augmentation) and the postprocessing phase (conversion back to predicates and skyline computation) exhibit linear or small polynomial costs, whereas the cost of *gFIM* can be exponential in the number of items (+ generalized items) [54, 55].

Cluster-Explorer effectively neutralizes the potentially high costs of gFIM by naturally restricting the number of considered items. This restriction arises from the relatively high minimum support (*minsup*) threshold used, which is aligned with the targeted explanation coverage threshold θ_{cov}. Unlike conventional *gFIM* applications, where users might choose low support thresholds

Dataset Name	#Rows	# Attr.
Urban Land Cover	168	148
DARWIN	174	451
Wine	178	13
Flags	194	30
Parkinson Speech	1040	26
Communities and Crime	1994	128
Turkiye Student Evaluation	5820	33
in-vehicle coupon recommendation	12684	23
Human Activity Recognition	10299	561
Quality Assessment of Digital Colposcopies	30000	23
RT-IoT2022	123117	85
Gender by Name	147270	4
Multivariate Gait Data	181800	7
Wave Energy Converters	288000	49
3D Road Network	434874	4
Year Prediction MSD	515345	90
Online Retail	1067371	8
MetroPT-3 Dataset	1516948	15
Taxi Trajectory	1710670	9

Table 3: List of UCI Datasets [59] and their properties

(e.g., 0.01 to 0.05), such settings are impractical in our context due to the need for explanations that adequately cover *the majority* of data points within a cluster.

To further reduce the number of items processed and thereby lower computational costs, we next introduce an *attribute selection* technique used in Cluster-Explorer.

4.4 Attribute Selection Optimization

One of the most crucial dataset properties affecting frequent itemset mining on relational data is the number of attributes, as the number of attributes directly influences the number of items (key-value pairs) [24]. To mitigate this effect, Cluster-Explorer introduces a simple yet effective attribute selection technique based on feature importance calculation [71], where we focus on the most pivotal attributes for each cluster $c \in C$.

Our attribute selection method is applied once for a dataset D and used for generating explanations for all clusters in C.

For each cluster $c \in C$, we train a binary decision tree classifier $M_c(X, \mathrm{CL}(x) = c)$, i.e., predicting if a data point is labeled as c (true) or not (false). We define the maximal depth to be $\frac{1}{\theta_{con}}$. To obtain an importance score for each attribute $a \in A$, we calculate the mean Gini impurity [7] score (known as Gini importance) $G_c(a)$, and select the top n_{attr} attributes. We then average the Gini scores across all models M_c to obtain the final importance score attr-score$(a) = \frac{\sum_{c \in C} G_c(a)}{|C|}$. Finally, we select the top n_{attr} features obtaining the highest attr-score.

Note that our importance measure assigns equal weights to all clusters, thus avoiding bias towards larger clusters that may hinder the explanation quality for smaller ones. In our implementation of Cluster-Explorer, we set n_{attr} to be proportional to the conciseness threshold θ_{con}, using a scaling parameter p: $n_{\text{attr}} = \lfloor \frac{1}{\theta_{con}} \times p \rfloor$.

As detailed in Section 5, we experiment with $1 \leq p \leq 2$ and compare the results to the exact computation, where $n_{\text{attr}} = |A|$. Our results are highly positive, showing that our attribute-selection optimization allows for an average speedup of 14.4X, and may reach up to 26X or higher for larger datasets, while negligibly affecting the quality of the explanations (see Section 5.2).

PID	Scaling	OneHot	PCA	Clustering Algorithm
1	✓	✓	✓	K-Means
2	✓	✓	✗	K-Means
3	✗	✓	✓	K-Means
4	✗	✓	✗	K-Means
5	✓	✓	✓	DBSCAN
6	✓	✓	✗	DBSCAN
7	✗	✓	✓	DBSCAN
8	✗	✓	✗	DBSCAN
9	✓	✓	✓	Birch
10	✓	✓	✗	Birch
11	✗	✓	✓	Birch
12	✗	✓	✗	Birch
13	✓	✓	✓	Spectral
14	✗	✓	✓	Spectral
15	✓	✓	✓	Affinity Propagation
16	✗	✓	✓	Affinity Propagation

Table 4: Clustering pipelines used in our experiments

5 EXPERIMENTS

We evaluated Cluster-Explorer on 98 clustering results, comparing its explanation quality and runtime to several XAI baselines, with an additional user study to validate the quality assessment. Our results shows that Cluster-Explorer produces superior explanations, with a 12X better running times than the closest baseline. Further examining the effectiveness of attribute selection, we show it achieved a 14.4X speedup with minimal impact on quality.

5.1 Experimental Setup

We next describe our benchmark dataset of clustering results, the explanation quality measures we use, the prototype implementation of Cluster-Explorer, and the baselines approaches.

Benchmark Dataset. To evaluate Cluster-Explorer, we developed a benchmark dataset containing 98 clustering results obtained from 16 different clustering pipelines using 5 clustering algorithms, applied to 19 datasets. We used publicly available dataset from the UCI collection [59], chosen to ensure a wide range of data shapes: Lengths between 168 and 1.7M rows, and width between 4 and 561 columns. See Table 3 for exact details.

Table 4 details the exact steps used in each pipeline, each comprises a combination of the following steps: (1) Standard scaling for numeric columns, (2) One-hot encoding for categorical data, (3) Dimensionality reduction using PCA [16] to a number equal to 90% of the original columns, and (4) Applying one of five common clustering algorithms: K-Means [38], DBSCAN [18], Birch [69], Spectral Clustering [60], and Affinity Propagation [21]. All clustering pipelines were executed over all 19 dataset, returning for each dataset $D = \langle X, A \rangle$ a clustering label function $CL(x)$ for the data points X. Finally, we filtered out unsuccessful clustering results where the distance between in-cluster data points is similar to that of different-cluster data points, to avoid explaining close-to-random results. This was implemented by calculating the *silhouette coefficient* [49] $SC(X, CL(x))$ and filtering out clustering results with

low scores, below 0.1. The final benchmark collection includes 98 clustering results instances.

Evaluating Metrics. Based on the clustering explanations quality criteria described in Section 3.2, we use a unified Quality Score for an Explanation (QSE) which balances the coverage, separation error, and conciseness. QSE is defined for a *single* cluster explanation by:

$$\text{QSE}(E_c) = \frac{Coverage(E_c) + (1 - SeparationErr(E_c)) + Conciseness(E_c)}{3}$$

Recall that Cluster-Explorer (as well as the baselines introduced below) may produce multiple explanations for each cluster. Given the results of a clustering pipeline, $CL(x)$, we use each baseline to generate a set of explanations for each cluster. Let $EX_{all} = \{\mathcal{E}_c \mid \forall c \in C\}$ be the set of all explanation sets produced by the baseline for a given clustering results instance. We then aggregate the QSE scores for EX_{all} by calculating the average score of the best explanation generated for each cluster:

$$QSE(EX_{all}) \frac{\sum_{\mathcal{E}_c \in EX_{all}} \max_{E_c \in \mathcal{E}_c} QSE(E_c)}{|C|}$$

We also measure the execution time of each baseline, as the time it takes to generate *all* cluster explanations EX_{ALL}.

Cluster-Explorer Implementation & Configuration. Cluster-Explorer is implemented in Python 3.10. It uses Pandas [45] to store and manipulate the dataset and NumPy [29] and calculate gFIM via the Pythonic implementation for Apriori[1]. We have made the source code available in [57]. The experiments were conducted on a Windows 10 laptop with 32GB RAM and 3.6 GHz CPU.

As for the Cluster-Explorer configuration, we used the attribute selection method (as described in Section 4.4), using $p = 1$, for both quality and running times experiments. The thresholds configuration used is: coverage threshold $\theta_{cov} = 0.8$; separation error threshold $\theta_{sep} = 0.3$; and conciseness threshold $\theta_{con} = 0.2$, allowing for a maximum of 5 predicates in each explanation. These thresholds, when possible, were used for the baseline approaches as described below.

Baseline Approaches. We compared Cluster-Explorer to several XAI baselines [12, 15, 22, 40, 48]. For the first two, we fitted an auxiliary XGBoost [11] model to predict clustering labels, then used the baselines to explain its results. The third baseline followed a similar approach but used a Random Forest internally. The fourth baseline uses a simpler auxiliary model – a standard decision tree.

We next describe the baselines in more detail.

1. SHAP [40]: SHAP is a highly popular game-theoretic approach for explaining individual predictions of machine learning models. SHAP provides an importance score for each attribute-value pair of a given data point x and the model prediction $M(x)$.

To utilize SHAP for deriving cluster explanations, we first employ XGBoost [11] to fit the clustering labels $CL(X)$. Then, to produce explanations for each individual cluster, we calculate the SHAP scores of all correct predictions for a given cluster, i.e., where $M(x) = c$ (we limited the number of samples to 2000 due to expensive execution costs). Since SHAP scores are computed for each attribute-value pair, we applied equal-width binning to each numeric attribute and calculated the average SHAP value for each bin. We returned as explanations the conjunction of the top-i bins (or categorical key-value pairs), for $1 \leq i \leq \frac{1}{\theta_{con}}$.

2. Anchors [48]: The Anchors framework is a more recent solution for local explanations, aiming to provide more concise and meaningful explanations than previous solutions such as SHAP [40] and LIME [47], which provide an importance score for each feature-value combination. In Anchors, the explanations are provided as decision rules that "anchor" the prediction, i.e., changes made to other attributes or ranges that do not appear in those rules do not affect the model outcome. For this baseline, we also utilize an XGBoost model to fit the cluster labels and then produce the anchors (decision rules) for the data points of each cluster. Since rules are produced for individual data points, we return a sample of 20 such explanations for each cluster.

3. SkopeRules[2]: SkopeRules is an open-source library for generating decision rules that characterize a target class, specifically identifying class instances with high precision. The SkopeRules code is based on a line of previous research for extracting decision rules from ensemble learning models [12, 22]. According to their documentation, SkopeRules utilizes a Random Forest model to fit the class labels and then mines a diverse set of rules from the individual decision trees generated by the model. The program allows setting the trees' depth, thereby limiting the explanations' size, as well as thresholds for precision and recall (we used $\frac{1}{\theta_{con}}$, θ_{cov}, and $1 - \theta_{sep}$ for those thresholds).

4. Decision-Tree Explanations: Last, inspired by the notion of ML *interpretability* [15], we devise an additional baseline based on fitting a simple decision tree model to the cluster labels. Unlike the previous baselines, which utilize a complex tree ensemble model (XGBoost) coupled with an external explanation framework, the decision tree model is simple and easy to interpret.

We derive cluster explanations by first fitting a binary decision tree with a maximum depth of $\frac{1}{\theta_{con}}$, then returning all tree paths from root to leaf as cluster explanations.

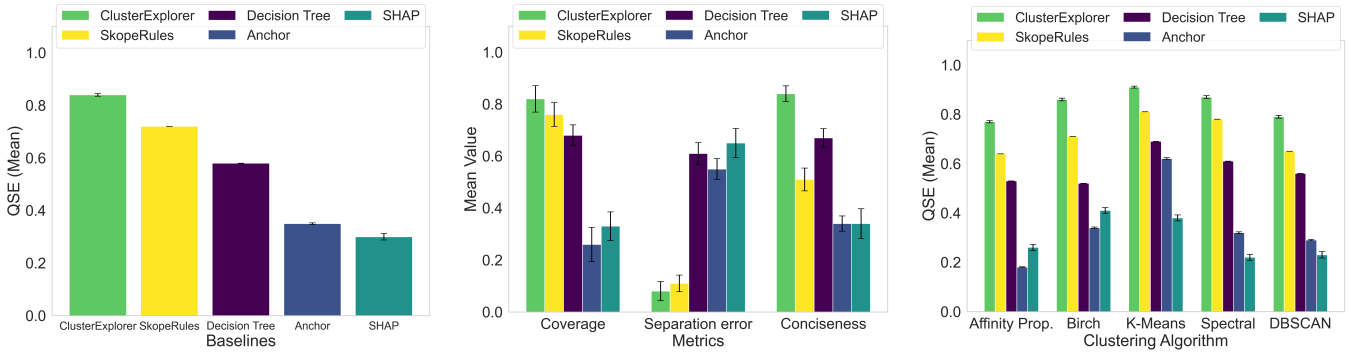
5.2 Experiment Results

We next detail our results, examining first the quality of generated explanations, then the running times comparison, and finally, the effect of our attribute-selection optimization on both the explanation quality and running times.

5.2.1 Explanation Quality Evaluation (Automatic). Figure 4a depicts the explanation set QSE score for each baseline, averaged across all 98 clustering result instances along with a vertical error bar reporting the .95 confidence interval. First, we note that the two baselines based on local explanations for supervised models—SHAP and Anchors—both demonstrate inferior results, with scores of 0.32 and 0.35, respectively. Decision tree explanations, based on the interpretability-by-design approach, achieve a higher average QSE of 0.58, surpassing SHAP and Anchors but still significantly lower than SkopeRules and Cluster-Explorer. SkopeRules, which mines rules from a more complex Random Forest model, achieves higher QSE scores (0.72) than the simple decision tree but is still 13 points

[1] https://efficient-apriori.readthedocs.io/en/latest/

[2] https://github.com/scikit-learn-contrib/skope-rules



(a) Average QSE **(b) Average Score for each metric** **(c) Average QSE per Clustering Algorithm**

Figure 4: Combined figures illustrating QSE and metrics analysis.

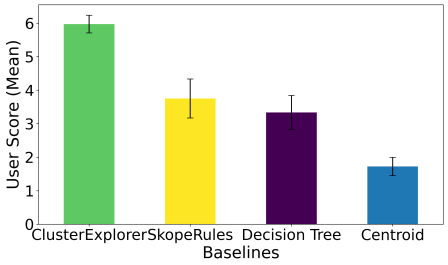


Figure 5: User Study Results

lower than Cluster-Explorer, which attains the highest average QSE of 0.84.

In Figure 4b we further report the average maximal score obtain for each metric individually. Cluster-Explorer achieves the highest scores across all individual metrics. Comparing baselines, see, for example, that SkopeRules shows a comparable separation error (0.11, only 0.03 higher than Cluster-Explorer) – but falls behind in coverage (0.76 vs. 0.82) and conciseness (0.51 vs. 0.84), whereas Decision Tree explanations are rather concise (0.67) with fair coverage (0.68) but suffer from a much higher separation error (0.61). Next, we analyze the distribution of QSE scores across different clustering algorithms, as shown in Figure 4c. Cluster-Explorer again outperforms the other baselines, but, interestingly, the results slightly vary: The highest performance is achieved for K-Means clusters (0.91), while the lowest is observed for Affinity Propagation (0.77). This variation can be intuitively attributed to the differences in cluster shapes: K-Means generates spherical clusters centered around distinct points, whereas Affinity Propagation, which employs a message-passing approach [21], produces clusters with more complex and irregular shapes. Similar differences in QSE scores are observed for the additional baselines as well.

5.2.2 User Study. To further validate the quality of the explanations, we conducted a small-scale user study. We recruited 12 participants by public calls targeting computer science students or graduates familiar with data analysis and/or data science. For the datasets *Wine*, *Turkiye student evaluation*, *Communities and Crime*, we randomly selected 12 clustering pipeline results for evaluation. We chose these datasets due to their moderate complexity, ensuring participants could understand the data and its attributes without requiring additional time or effort.

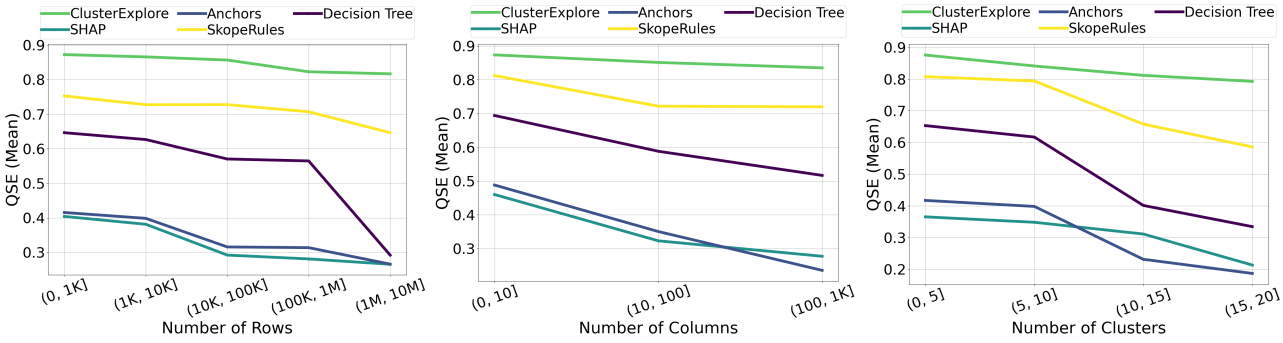
Each participant was first introduced to the data and its attributes, followed by a visualization of the clustering pipeline results in a two-dimensional plot (as in Figure 1). Subsequently, we presented a series of cluster explanations generated by the following methods: Cluster-Explorer, SkopeRules, Decision-Tree, and an alternative non-rule-based approach using *cluster centroids* (i.e., the data point closest to the cluster mean). We excluded Anchor and SHAP as their explanations consistently yielded significantly lower QSE scores as described above. For all rule-based approaches (Cluster-Explorer, SkopeRules, and Decision-Tree), we additionally described the explanation *coverage* and *separation error* rates (phrased as in Figure 2) to provide further context for evaluation. Participants were then asked to rate the quality of each explanation on a scale from 1 (low) to 7 (high) without knowing the associated method used to generate the explanations. They were also required to justify their ratings through free-text responses. Each participant evaluated three cluster explanations per dataset, resulting in a total of 36 user evaluations, equally distributed over the 12 clustering results.

The average user scores are presented in Figure 5 (with vertical lines depicting .95 CI). Participants unanimously preferred rule-based explanations, with cluster centroids receiving the lowest average score of 1.72. Notably, the user scores ranking is strongly correlated with the average QSE scores: Cluster-Explorer achieved the highest average score of 5.97, compared to 3.7 for SkopeRules and 3.33 for Decision Tree.

Analyzing the participants' justification for their ratings, Cluster-Explorer was consistently rated as the best approach due to its combination of high coverage, accuracy, and diversity of explanations. SkopeRules was appreciated for its accuracy (i.e., low separation error) but was criticized for limited coverage and lack of explanation variety. Decision Tree was noted for its concise explanations but was penalized for significantly higher error rates. Explanations based on centroids were generally considered overly complex and not user-friendly, compared to the rule-based approaches.

5.2.3 parameters effect on explanations quality. We further investigate the effect of data and problem complexity on the QSE score.

Figure 6 shows average QSE scores as a function of the number of rows, columns, and clusters. The decision tree baseline is most affected, with QSE scores dropping by 54.8%, 25.5%, and 48.7% as rows, columns, and clusters increase, respectively, reflecting its



(a) QSE vs. Number of Rows **(b) QSE vs. Number of Columns** **(c) QSE vs. Number of Clusters**

Figure 6: Average QSE scores as a factor of the number of rows, columns, and clusters

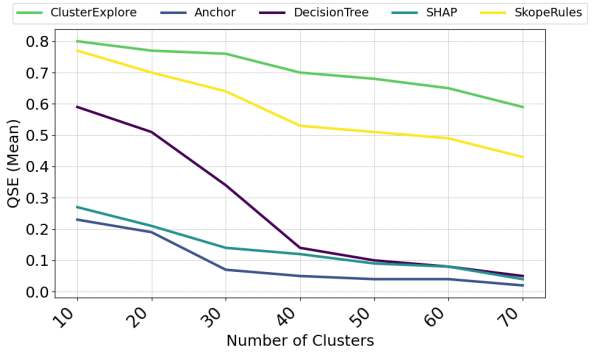


Figure 7: QSE vs. Num. of Clusters(Simulated)

limitations for large, complex datasets [4, 14]. Other baselines also degrade with data complexity: SkopeRules drops by 14.1%, 11.4%, and 27.4%, Anchors by 35.9%, 51.7%, and 55.2%, and SHAP by 34.2%, 39.7%, and 41.6% for datasets with over 1M rows, 100 columns, and 15 clusters, respectively. In contrast, Cluster-Explorer remains stable, with minor decreases of 6.3%, 4.4%, and 9.4%.

Finally, to take a closer look at the performance as a function of the number of clusters, we conducted additional experiments on simulated datasets. In our benchmark of real datasets, all pipelines producing more than 20 clusters yielded Silhouette scores below the acceptable threshold ($\leq$ 0.1). To address this, we generated artificial datasets using the method from [46], each with 10K rows, 10 columns, and between 10 to 70 centroids. We then executed all clustering pipelines (see Table 4), resulting in 5490 clustering results with Silhouette scores $\geq$ 0.1. Figure 7 illustrates the average QSE of the baselines as a function of the number of clusters. While all baselines show a decline in quality as the number of clusters increases, Cluster-Explorer exhibits the most moderate decline, achieving a mean QSE of 0.59 for the cases of 70 clusters, compared to 0.43 for SkopeRules and less than 0.05 for the remaining baselines.

5.2.4 Running Times Evaluation. We further compare the running times of Cluster-Explorer against the four additional baselines. Figure 8 depicts the average time (in minutes) took for each baseline to return all cluster explanations (i.e., $EX_{ALL} = \{\mathcal{E}_c \mid \forall c \in C\}$), as a factor of the number of rows, columns and resulted clusters.

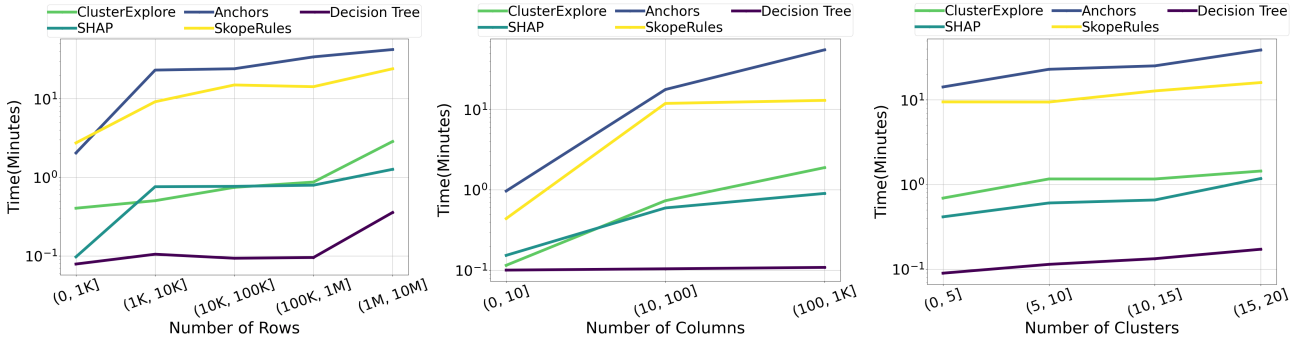
Naturally, we observe a running time increase for all baselines as the number of rows increases (Figure 8a). Similar trends are observed for increasing the umber of columns (Figure 8b), and the number of clusters (Figure 8c). As expect, the most simple decision tree is indeed the fastest in generating explanations, with an average running time of 6.3 seconds (however, recall that the quality of explanations is significantly subpar compared to SkopeRules and Cluster-Explorer). Also see that the more complex frameworks of SkopeRules and Anchors demonstrate the slowest running times: SkopeRules runs in more than 10 minutes on most settings (except for datasets with less than 1K row and 10 columns), and Anchors often exceeds 20 minutes.

In contrast, the running times of Cluster-Explorer are significantly better (roughly on par with SHAP), with an average running time of 55.8 seconds. In particular, it took Cluster-Explorer an average of 2.86 minutes to compute all explanations for for datasets larger than 1M rows (compared to, e.g., 24 minutes by SkopeRules). This is due to the effectiveness of the attribute selection optimization (See Section 4.4) which dramatically decreases running time compared to the exact calculation of Cluster-Explorer (see below) – obtaining 12.6X faster times, on average, than the closest baseline in terms of quality.

5.2.5 Attribute Selection Effect on Explanations Quality and Generation Times. We analyzed the effectiveness of our attribute selection optimization compared to the exact calculation, varying the parameter p (which controls the number of attributes used relative to the conciseness threshold θ_{con}).

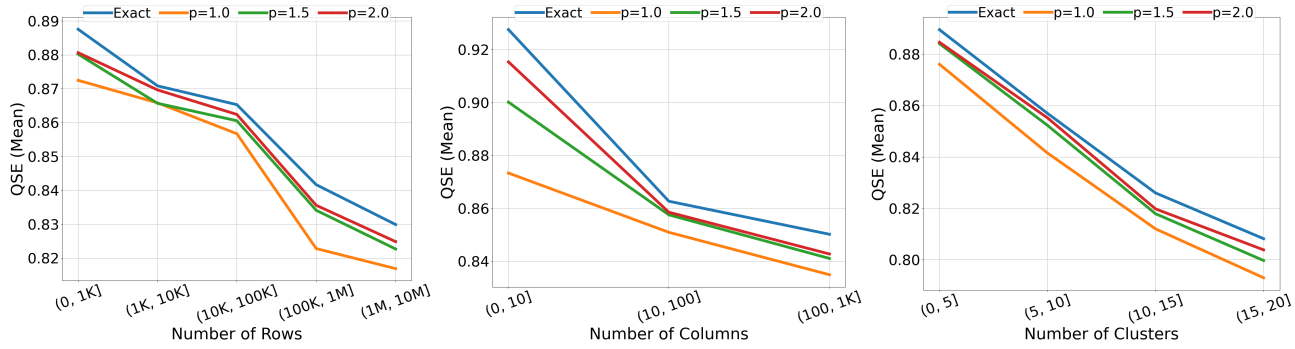
Figure 9 shows that while the exact Cluster-Explorer achieves the highest average QSE score (0.86), the optimized versions with $1 \leq p \leq 2$ perform nearly as well, with $p = 1$ achieving 0.85 on average. The largest gap occurs for datasets with $\leq$ 10 columns, where the exact method scores 0.93 compared to 0.87 for $p = 1$, but this difference is less critical due to the shorter runtime of the exact method for small datasets.

In terms of running time, as depicted in Figure 10, the optimization provides substantial speedups for larger datasets. For datasets with over 1M rows, the optimized Cluster-Explorer achieves a 26.9X improvement, and for datasets with more than 100 columns, the average improvement is 19.7X. Overall, the optimization reduces runtime by 14.4X on average, cutting the exact calculation



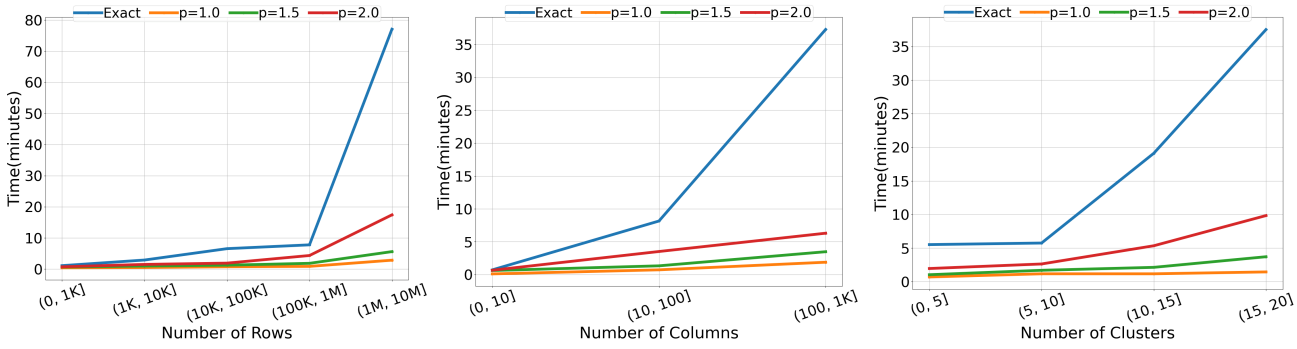
(a) Runtime vs. Number of Rows **(b) Runtime vs. Number of Columns** **(c) Runtime vs. Number of Clusters**

Figure 8: Explanations generation times (for all clusters) as a factor of the number of rows, columns, and clusters.



(a) QSE vs. Number of Rows **(b) QSE vs. Number of Columns** **(c) QSE vs. Number of Clusters**

Figure 9: The effect of p on the QSE as a factor of the number of rows, columns, and clusters



(a) Runtime vs. Number of Rows **(b) Runtime vs. Number of Columns** **(c) Runtime vs. Number of Clusters**

Figure 10: The effect of p on running times as a factor of the number of rows, columns, and clusters

time from 13.42 minutes to just 55.8 seconds, with minimal impact on QSE scores (<0.1 difference).

6 CONCLUSION

Cluster-Explorer is a novel framework for post-hoc, rule-based explanations of clustering results. It is independent of specific clustering algorithms and avoids using auxiliary ML models for cluster labels. To produce explanations, Cluster-Explorer first augments the original data with predicates representing numeric intervals and categorical negations, then effectively mines generalized frequent itemsets from the data. An attribute selection optimization further reduces computational cost by limiting items considered. Experiments on 98 clustering results and a user study demonstrate its effectiveness over existing solutions.

In future work, we plan to incorporate disjunctions, predict which thresholds to use for coverage and separation error, investigate semantic-based evaluation for explanations, and develop targeted solutions for local explanations as well as for more complex cluster structures such as hierarchical and density based clustering.

REFERENCES

[1] Adult Income Dataset (UCI). 2024. https://archive.ics.uci.edu/ml/datasets/Adult/.

[2] Rakesh Agrawal, Ramakrishnan Srikant, et al. 1994. Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB*, Vol. 1215. Citeseer, 487–499.

[3] Suad A Alasadi and Wesam S Bhaya. 2017. Review of data preprocessing techniques in data mining. *Journal of Engineering and Applied Sciences* 12, 16 (2017), 4102–4107.

[4] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, et al. 2020. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information fusion* 58 (2020), 82–115.

[5] Szymon Bobek, Michal Kuk, Maciej Szelążek, and Grzegorz J Nalepa. 2022. Enhancing cluster analysis with explainable AI and multidimensional cluster prototypes. *IEEE Access* 10 (2022), 101556–101574.

[6] Stephan Borzsony, Donald Kossmann, and Konrad Stocker. 2001. The skyline operator. In *Proceedings 17th international conference on data engineering*. IEEE, 421–430.

[7] L. Breiman, J. Friedman, C.J. Stone, and R.A. Olshen. 1984. *Classification and Regression Trees*. Taylor & Francis.

[8] Marco Cavallo and Çağatay Demiralp. 2018. Clustrophile 2: Guided visual clustering analysis. *IEEE transactions on visualization and computer graphics* 25, 1 (2018), 267–276.

[9] Yair Censor. 1977. Pareto optimality in multiobjective problems. *Applied Mathematics and Optimization* 4, 1 (1977), 41–59.

[10] Angelos Chatzimparmpas, Rafael M Martins, and Andreas Kerren. 2020. t-visne: Interactive assessment and interpretation of t-sne projections. *IEEE transactions on visualization and computer graphics* 26, 8 (2020), 2696–2714.

[11] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 785–794.

[12] William W Cohen and Yoram Singer. 1999. A simple, fast, and effective rule learner. *AAAI/IAAI* 99, 335-342 (1999), 3.

[13] Brian A Davey and Hilary A Priestley. 2002. *Introduction to lattices and order*. Cambridge university press.

[14] Finale Doshi-Velez and Been Kim. 2017. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608* (2017).

[15] Mengnan Du, Ninghao Liu, and Xia Hu. 2019. Techniques for interpretable machine learning. *Commun. ACM* 63, 1 (2019), 68–77.

[16] George H Dunteman. 1989. *Principal components analysis*. Number 69. Sage.

[17] Charles A Ellis, Mohammad SE Sendi, Eloy Geenjaar, Sergey M Plis, Robyn L Miller, and Vince D Calhoun. 2021. Algorithm-Agnostic Explainability for Unsupervised Clustering. *arXiv preprint arXiv:2105.08053* (2021).

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

[19] Absalom E Ezugwu, Abiodun M Ikotun, Olaide O Oyelade, Laith Abualigah, Jeffery O Agushaka, Christopher I Eke, and Andronicus A Akinyelu. 2022. A comprehensive survey of clustering algorithms: State-of-the-art machine learning applications, taxonomy, challenges, and future research prospects. *Engineering Applications of Artificial Intelligence* 110 (2022), 104743.

[20] Yoav Freund and Robert E Schapire. 1997. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences* 55, 1 (1997), 119–139.

[21] Brendan J Frey and Delbert Dueck. 2007. Clustering by passing messages between data points. *science* 315, 5814 (2007), 972–976.

[22] Jerome H FRIEDMAN and Bogdan E POPESCU. 2008. PREDICTIVE LEARNING VIA RULE ENSEMBLES. *The Annals of applied statistics* 2, 3 (2008), 916–954.

[23] Nave Frost, Michal Moshkovitz, and Cyrus Rashtchian. 2020. ExKMC: Expanding Explainable k-Means Clustering. *arXiv preprint arXiv:2006.02399* (2020).

[24] Yongjian Fu and Jiawei Han. 1995. Meta-Rule-Guided Mining of Association Rules in Relational Databases.. In *KDOOD/TDOOD*. Citeseer, 39–46.

[25] Mohamed H Gad-Elrab, Daria Stepanova, Trung-Kien Tran, Heike Adel, and Gerhard Weikum. 2020. Excut: Explainable embedding-based clustering over knowledge graphs. In *International Semantic Web Conference*. Springer, 218–237.

[26] Salvador García, Sergio Ramírez-Gallego, Julián Luengo, José Manuel Benítez, and Francisco Herrera. 2016. Big data preprocessing: methods and prospects. *Big Data Analytics* 1, 1 (2016), 1–22.

[27] Riccardo Guidotti, Anna Monreale, Fosca Giannotti, Dino Pedreschi, Salvatore Ruggieri, and Franco Turini. 2019. Factual and counterfactual explanations for black box decision making. *IEEE Intelligent Systems* 34, 6 (2019), 14–23.

[28] Jiawei Han, Jian Pei, and Yiwen Yin. 2000. Mining frequent patterns without candidate generation. *ACM sigmod record* 29, 2 (2000), 1–12.

[29] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. 2020. Array programming with NumPy. *Nature* 585, 7825 (2020), 357–362.

[30] Douglas M Hawkins. 2004. The problem of overfitting. *Journal of chemical information and computer sciences* 44, 1 (2004), 1–12.

[31] Jochen Hipp, Andreas Myka, Rüdiger Wirth, and Ulrich Güntzer. 1998. A new algorithm for faster mining of generalized association rules. In *Principles of Data Mining and Knowledge Discovery: Second European Symposium, PKDD'98 Nantes, France, September 23–26, 1998 Proceedings 2*. Springer, 74–82.

[32] Mark Ibrahim, Melissa Louie, Ceena Modarres, and John Paisley. 2019. Global explanations of neural networks: Mapping the landscape of predictions. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*. 279–287.

[33] Eser Kandogan. 2012. Just-in-time annotation of clusters, outliers, and trends in point-based data visualizations. In *2012 IEEE Conference on Visual Analytics Science and Technology (VAST)*. IEEE, 73–82.

[34] Jacob Kauffmann, Malte Esders, Lukas Ruff, Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. 2022. From clustering to cluster explanations via neural networks. *IEEE Transactions on Neural Networks and Learning Systems* 35, 2 (2022), 1926–1940.

[35] Daniel Kunkle, Donghui Zhang, and Gene Cooperman. 2008. Mining frequent generalized itemsets and generalized association rules without redundancy. *Journal of Computer Science and Technology* 23, 1 (2008), 77–102.

[36] Bum Chul Kwon, Ben Eysenbach, Janu Verma, Kenney Ng, Christopher De Filippi, Walter F Stewart, and Adam Perer. 2017. Clustervision: Visual supervision of unsupervised clustering. *IEEE transactions on visualization and computer graphics* 24, 1 (2017), 142–151.

[37] Pantelis Linardatos, Vasilis Papastefanopoulos, and Sotiris Kotsiantis. 2021. Explainable ai: A review of machine learning interpretability methods. *Entropy* 23, 1 (2021), 18.

[38] Stuart Lloyd. 1982. Least squares quantization in PCM. *IEEE transactions on information theory* 28, 2 (1982), 129–137.

[39] Wei-Yin Loh. 2011. Classification and regression trees. *Wiley interdisciplinary reviews: data mining and knowledge discovery* 1, 1 (2011), 14–23.

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

[41] Tim Miller. 2019. Explanation in artificial intelligence: Insights from the social sciences. *Artificial intelligence* 267 (2019), 1–38.

[42] Michal Moshkovitz, Sanjoy Dasgupta, Cyrus Rashtchian, and Nave Frost. 2020. Explainable k-means and k-medians clustering. In *International conference on machine learning*. PMLR, 7055–7065.

[43] Daniel Müllner. 2011. Modern hierarchical, agglomerative clustering algorithms. *arXiv preprint arXiv:1109.2378* (2011).

[44] Guillermo Navas-Palencia. 2020. Optimal binning: mathematical programming formulation. *arXiv preprint arXiv:2001.08025* (2020).

[45] The pandas development team. 2020. *pandas-dev/pandas: Pandas*. https://doi.org/10.5281/zenodo.3509134

[46] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *the Journal of machine Learning research* 12 (2011), 2825–2830.

[47] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. " Why should i trust you?" Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 1135–1144.

[48] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Anchors: High-precision model-agnostic explanations. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 32.

[49] Peter J Rousseeuw. 1987. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics* 20 (1987), 53–65.

[50] Mirka Saarela and Susanne Jauhiainen. 2021. Comparison of feature importance measures as explanations for classification models. *SN Applied Sciences* 3, 2 (2021), 272.

[51] Omer Sagi and Lior Rokach. 2018. Ensemble learning: A survey. *Wiley interdisciplinary reviews: data mining and knowledge discovery* 8, 4 (2018), e1249.

[52] Vidya Setlur, Michael Correll, and Sarah Battersby. 2022. Oscar: A semantic-based data binning approach. In *2022 IEEE Visualization and Visual Analytics (VIS)*. IEEE, 100–104.

[53] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. 2017. Learning important features through propagating activation differences. In *International Conference on Machine Learning*. PMLR, 3145–3153.

[54] Ramakrishnan Srikant and Rakesh Agrawal. 1997. Mining generalized association rules. *Future generation computer systems* 13, 2-3 (1997), 161–180.

[55] Kritsada Sriphaew and Thanaruk Theeramunkong. 2002. A new method for finding generalized frequent itemsets in generalized association rule mining. In *Proceedings ISCC 2002 Seventh International Symposium on Computers and Communications*. IEEE, 1040–1045.

[56] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *International Conference on Machine Learning*. PMLR, 3319–3328.

[57] Sariel Tutay and Amit Somech. 2023. https://github.com/analysis-bots/cluster-explorer.

[58] Sariel Tutay and Amit Somech. 2023. Cluster-Explorer: An interactive Framework for Explaining Black-Box Clustering Results. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. 5106–5110.

[59] UC Irvine Machine Learning Repository. 2024. https://archive.ics.uci.edu/.

[60] Ulrike Von Luxburg. 2007. A tutorial on spectral clustering. *Statistics and computing* 17 (2007), 395–416.

[61] Danding Wang, Qian Yang, Ashraf Abdul, and Brian Y Lim. 2019. Designing theory-driven user-centric explainable AI. In *Proceedings of the 2019 CHI conference on human factors in computing systems*. 1–15.

[62] Haizhou Wang and Mingzhou Song. 2011. Ckmeans. 1d. dp: optimal k-means clustering in one dimension by dynamic programming. *The R journal* 3, 2 (2011), 29.

[63] Jiazhi Xia, Linquan Huang, Weixing Lin, Xin Zhao, Jing Wu, Yang Chen, Ying Zhao, and Wei Chen. 2022. Interactive visual cluster analysis by contrastive dimensionality reduction. *IEEE Transactions on Visualization and Computer Graphics* 29, 1 (2022), 734–744.

[64] Dongkuan Xu and Yingjie Tian. 2015. A comprehensive survey of clustering algorithms. *Annals of Data Science* 2, 2 (2015), 165–193.

[65] Guizhen Yang. 2004. The complexity of mining maximal frequent itemsets and maximal frequent patterns. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. 344–353.

[66] Xue Ying. 2019. An overview of overfitting and its solutions. In *Journal of physics: Conference series*, Vol. 1168. IOP Publishing, 022022.

[67] Jun Yuan, Changjian Chen, Weikai Yang, Mengchen Liu, Jiazhi Xia, and Shixia Liu. 2021. A survey of visual analytics techniques for machine learning. *Computational Visual Media* 7 (2021), 3–36.

[68] Shichao Zhang, Chengqi Zhang, and Qiang Yang. 2003. Data preparation for data mining. *Applied artificial intelligence* 17, 5-6 (2003), 375–381.

[69] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. 1996. BIRCH: an efficient data clustering method for very large databases. *ACM sigmod record* 25, 2 (1996), 103–114.

[70] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. 2016. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2921–2929.

[71] Alexander Zien, Nicole Krämer, Sören Sonnenburg, and Gunnar Rätsch. 2009. The feature importance ranking measure. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2009, Bled, Slovenia, September 7-11, 2009, Proceedings, Part II 20*. Springer, 694–709.