Interpretable Highlights for Experiment Tracking

Vassilis Stamatopoulos[1,2,*], Panos Gidarakos[2], Stavros Maroulis[2], George Papastefanatos[2] and Panos Vassiliadis[1]

[1] *University of Ioannina, Ioannina, Greece*

[2] *Athena Research Center, Athens, Greece*

Abstract

Experiment tracking systems log hundreds of runs with large numbers of (a) configuration parameters for the fine-tuning of the experiment and (b) a large number of metrics that assess the behavior of each configuration. However, the large numbers of runs makes it difficult to discover the trade-offs of each configuration to its behavior, towards deciding the right configuration that satisfies an analyst's intent. We address this problem with an automated experiment analytics approach that (i) groups runs into behaviorally consistent clusters based on their observed metrics and (ii) generates compact, interpretable cluster-level summaries that connect characteristic metric outcomes to the configuration choices that tend to produce them. For each discovered behavior, the method produces actionable highlights, including representative metrics, salient metric relationships, and concise configuration rules that describe where the behavior occurs in the configuration space. Visualizations with radar charts and parallel coordinates provide an interactive means to highlight the pros and cons of each behavior with respect to its representative metrics and configuration. Experiments on several ML pipelines, show that the method yields strong cluster structure (Silhouette Score > 0.9) and stable descriptors using only 3–6 representative metrics per cluster, while remaining computationally practical with end-to-end runtimes on the order of seconds. An ablation study further shows that removing key components degrades at least one aspect of performance or interpretability, underscoring the importance of the overall approach.

Keywords

experiment monitoring, explainable clustering, highlight extraction, MLOps

1. Introduction

In MLOps, machine learning pipelines are repeatedly executed under different configuration settings, producing large collections of experiment run data annotated with configuration parameters, metrics, and artifacts [1]. Consider a typical supervised learning pipeline that performs data preprocessing, model training, and evaluation. During model development, practitioners explore a high-dimensional space of data preprocessing options, model variants, and hyperparameters, often guided by automated search strategies such as grid search. Each execution yields an experiment run with associated performance and resource metrics.

DOLAP 2026: 28th International Workshop on Design, Optimization, Languages and Analytical Processing of Big Data, co-located with EDBT/ICDT 2026, March 24, 2026, Tampere, Finland

*Corresponding author.

bstam@athenarc.gr (V. Stamatopoulos); pgidar@athenarc.gr (P. Gidarakos); stavmars@athenarc.gr (S. Maroulis); gpapas@athenarc.gr (G. Papastefanatos); pvassil@cs.uoi.gr (P. Vassiliadis)

0000-0002-9044-796X (V. Stamatopoulos); 0000-0003-2816-4368 (S. Maroulis); 0000-0002-9273-9843 (G. Papastefanatos); 0000-0003-0085-6776 (P. Vassiliadis)

To support this process, practitioners rely on experiment tracking tools to log results, maintain execution history, and compare runs. Systems such as MLflow [2], Weights & Biases [3], and Neptune [4] are widely used to systematically record configurations, metrics, and artifacts and expose them through dashboards and visual analytic capabilities. Despite the availability of such tools and large volumes of experiment data, analysis is often restricted to local inspection of individual executions or simple visual comparisons over a small set of metrics. As a result, distinct behaviors and trade-offs that arise in different regions of the configuration space remain difficult to identify.

In practice, a model is often evaluated under many combinations of configuration parameters. For example, some configurations may achieve high predictive performance but exhibit unfavorable fairness properties, while others offer more balanced trade-offs at the expense of peak accuracy. Figure 1 illustrates this situation: from many runs with configurations and heterogeneous metrics, distinct recurring behaviors emerge with characteristic trade-offs (e.g., detection-oriented vs. fairness-oriented outcomes). Being able to explicitly identify such recurring configuration behaviors, and the regions of the configuration space in which they arise (e.g., simple conditions over `max depth` and `estimators`), together with their characteristic metric profiles in informative visualizations, (e.g., radar charts), would not only support optimization in the current experiment but also provide reusable insights to guide the design of subsequent experiments.

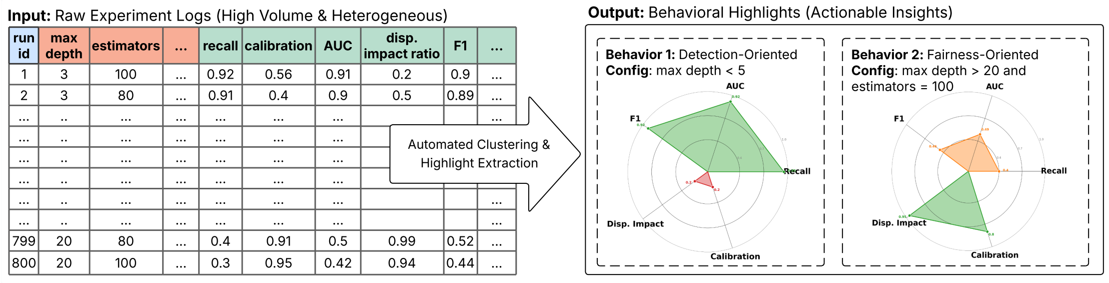
In this work, we introduce a clustering-based experiment analytics approach that derives interpretable summaries of ML experiment results at the level of configuration behavior. By making explicit how different regions of the configuration space are associated with characteristic metric outcomes and trade-offs, the proposed method supports scalable reasoning over large experiment collections. The resulting summaries are compact and directly consumable through textual and visual representations, enabling comparison, exploration, and informed refinement of experiment designs.

This paper makes three main contributions. (1) We formalize the problem of analyzing large collections of ML experiment runs in terms of metric and configuration-level behavior, exposing the limitations of existing run-level analysis practices. (2) We introduce an automated experiment analytics approach that clusters runs in metric space and produces compact, interpretable summaries that relate characteristic metric outcomes to the configuration regions in which they arise, enabling scalable reasoning over complex experiment spaces. (3) We demonstrate, through an experimental evaluation on multiple realistic ML pipelines, that the proposed approach produces stable, discriminative, and interpretable summaries that support effective comparison and refinement of experiment designs.

The remainder of the paper is structured as follows: Section 2 discusses related work. Section 3 defines the problem. Section 4 details the methodology we follow and the overall pipeline. Section 5 presents the experimental evaluation, and section 6 the conclusions.

2. Related Work

Complex scientific and ML experiments are supported by tools for managing workflows, tracking runs, and analyzing performance across many executions. Related work spans automated


Input: Raw Experiment Logs (High Volume & Heterogeneous)
Automated Clustering & Highlight Extraction
Output: Behavioral Highlights (Actionable Insights)
Behavior 1: Detection-Oriented
Config: max depth < 5
AUC
F1
Recall
Disp. Impact
Calibration
0.92
0.92
0.2
0.2
Behavior 2: Fairness-Oriented
Config: max depth > 20 and estimators = 100
AUC
F1
Recall
Disp. Impact
Calibration
0.49
0.46
0.4
0.95
0.8

run id	max depth	estimators	...	recall	calibration	AUC	disp. impact ratio	F1	...
1	3	100	...	0.92	0.56	0.91	0.2	0.9	...
2	3	80	...	0.91	0.4	0.9	0.5	0.89	...
...	..	...	...	...	...	...			...
...	..	...	...	...	...	...			
...	..	...	...	...	...	...			
...	..	...	...	...	...	...			
...	..	...	...	...	...	...			...
...	..	...	...	...	...	...			...
799	20	80	...	0.4	0.91	0.5	0.99	0.52	...
800	20	100	...	0.3	0.95	0.42	0.94	0.44	...

Figure 1: Example of behavior-level summarization over an experiment log: a large collection of runs with configuration parameters and multiple metrics can be condensed into a small set of recurring behaviors, each described by an interpretable configuration condition.

highlight extraction and exploratory data analysis, workflow profiling and analysis, interpretable clustering, and experiment tracking and workflow management systems. We outline the most relevant lines of work and clarify how our approach compares to them.

Automated Highlight Extraction. The current work is related to the field of automated highlight extraction, where data sets are evaluated for 'hidden gems', i.e., the existence of interesting properties (trends, outliers, correlations, etc) hidden in subsets of a data set. Frequently, the term used is *insight*. A general model as well as a survey of related work is found in [5]. Several tools have been proposed in the both the database and the visualization literature for helping analysts understand the hidden patterns in the data as well as assessing the importance of such findings with dedicated scores. The literature can be traced all the way to the distant past (most notably [6, 7, 8]), with a revival in the last 10 years [9, 10, 11, 12]. Most notable tools include Datashot [13], MetaInsight [14], Calliope [15], Erato [16], Notable [17] and InsightPilot [18]. Exploratory Data Analysis (EDA) is a very related field, too, with the particularity that the analyst is interactively guided to the exploration of the data [19, 20, 21, 22, 23, 24, 25].

Workflow Profiling and Analysis. A substantial body of work analyzes workflows to understand performance and resource usage, by monitoring runtime, memory, I/O behavior and provenance across many executions and using these data to detect bottlenecks or anomalies [26, 27, 28, 29, 30, 31, 32]. Clustering has also been used in the context of workflow scheduling and resource management, where runs are grouped to improve throughput, fairness, or cost in distributed environments [33, 34, 35]. These approaches leverage both application-level and system-level metrics, but their analytic goal is typically performance optimization, scheduling, or failure diagnosis rather than the discovery of high-level configuration behaviors. Our methodology uses clustering as an analysis tool over experiment logs, with an explicit focus on metric- and configuration-level highlights rather than scheduling objectives.

Interpretable & Explainable Clustering. Interpretable clustering methods aim to couple unsupervised grouping with human-understandable descriptions of clusters, for example by learning rule sets, decision trees, or prototype-based characterizations that explain why points are assigned to a given cluster [36, 37, 38]. Alvarez-García et al. propose a four-step framework for explainable cluster analysis on high-dimensional mixed-type data, combining data preprocessing, dimensionality reduction, clustering, and a classification module that uses SHapley Additive exPlanations (SHAP) to characterize clusters [36, 39]. More recently, Guilbert et al.

[38], tackle explainable clustering by modeling the data in two spaces: one for clustering and one for explanation; relying on ensemble clustering and constraint programming to produce high-quality interpretable clusters. Complementary work focuses on explaining black-box clustering pipelines by automatically deriving concise conjunctions of predicates that describe each cluster in the original feature space and by supporting interactive exploration of cluster explanations [40, 37]. In this work, we build a clustering-based analysis on top of experiment runs that identifies representative metrics for each cluster and generates multiple complementary highlights based on them. Each run is structurally represented by its hyperparameter configuration, which makes it natural to describe clusters through rules over configuration variables: for every cluster, we train a rule-based classifier that separates the cluster from the remaining runs and extract high-precision rules as cluster descriptors.

Experiment Tracking and ML Workflow Management Systems. ML-native workflow systems (e.g., ZenML, Kubeflow) orchestrate end-to-end ML pipelines and differ from general-purpose engines (e.g., Airflow) by providing ML-specific metadata and experiment management [1, 26]. To manage execution metadata, these systems often integrate specialized experiment tracking systems such as MLflow [2], Weights & Biases [3], and Neptune [4] to log and visualize configuration parameters, metrics, and artifacts across large numbers of runs. While these platforms support basic filtering and comparison, they typically rely on users to manually identify broader configuration behaviors. ExperimentLens [41], offers explainability for individual runs (e.g., via ALE and PDP plots) but lacks mechanisms to automatically identify and explain *group* behaviors. To the best of our knowledge, such capabilities for clustering runs and explaining group behaviors are absent in both experiment trackers and workflow orchestrators. Our method fills this void by offering an explanation layer that can integrate into either environment, organizing the logged metadata into interpretable clusters.

3. Notation & Problem Formulation

3.1. Notation

Data Representation. Let $x_1, \dots, x_{d_X}$ denote the configuration parameters (e.g., `max_depth`, `n_estimators`, algorithm choices), and let $m_1, \dots, m_{d_M}$ denote the logged metrics (e.g., accuracy, recall, fairness scores, CPU usage, memory consumption). Each parameter x_ℓ and metric m_j has an associated domain of admissible values, denoted $\mathrm{dom}(x_\ell)$ and $\mathrm{dom}(m_j)$, respectively. We define the configuration and metric spaces as the Cartesian products

$$\mathcal{X} = \mathrm{dom}(x_1) \times \cdots \times \mathrm{dom}(x_{d_X}), \qquad \mathcal{M} = \mathrm{dom}(m_1) \times \cdots \times \mathrm{dom}(m_{d_M}),$$

so that each configuration vector $\mathbf{x} \in \mathcal{X}$ assigns one value to every configuration parameter, and each metric vector $\mathbf{m} \in \mathcal{M}$ assigns one value to every metric.

We consider a collection of N experiment executions (runs), $\mathcal{E} = \{(\mathbf{x}_i, \mathbf{m}_i)\}_{i=1}^{N} \subseteq \mathcal{X} \times \mathcal{M}$, where $\mathbf{x}_i \in \mathcal{X}$ and $\mathbf{m}_i \in \mathcal{M}$ are the configuration and metric vectors of run i, respectively. Stacking configurations and metrics row-wise yields the data matrices $\mathbf{X}$ and $\mathbf{M}$, which we refer to as the configuration matrix and the metric matrix, respectively, that may contain both continuous and categorical columns.

Clustering. A clustering solution over $\mathcal{E}$ is defined as a complete and disjoint partition $\mathcal{C} = \{C_1, \ldots, C_K\}$ of the run indices $\{1, \ldots, N\}$, with an assignment function $\phi : \{1, \ldots, N\} \rightarrow \{1, \ldots, K\}$.

3.2. Problem Formulation

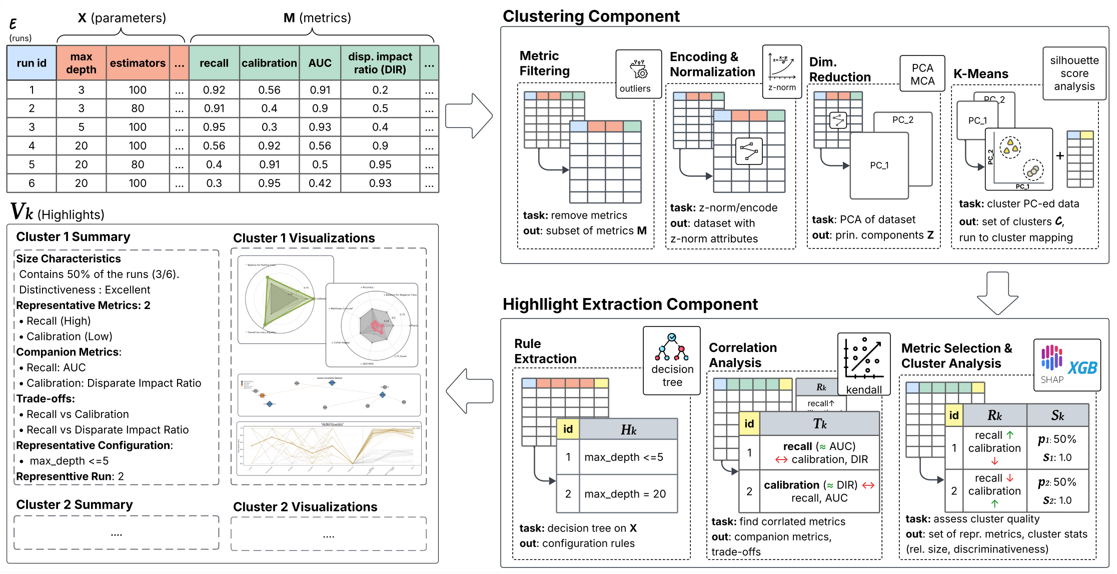
Given the logged experiment runs $\mathcal{E}$, we extract their configuration parameter and metric matrices $\mathbf{X}$ and $\mathbf{M}$ respectively. Our goal is to construct a structured summary of the experiment space in the form of a clustering $\mathcal{C}$ over the runs, together with associated *visual highlights* that can be presented to practitioners within existing experiment tracking tools.

Problem Decomposition. Addressing this objective involves two main tasks:

1. **Experiment Partitioning.** The first task is to compute a clustering solution $\mathcal{C}$. Functionally, each cluster C_k is intended to represent a distinct *operating behavior* of the experiment, grouping runs that exhibit similar metric profiles (e.g., high-accuracy/high-cost vs. low-accuracy/low-cost) to facilitate comparison. In this work, clustering is performed purely on the metrics $\mathbf{M}$, while configuration parameters $\mathbf{X}$ are used downstream to construct interpretable descriptors that explain cluster membership.
2. **Descriptor and Highlight Extraction.** For each cluster C_k, we derive a set of interpretable descriptors and visual artifacts to explain its behavior:
   - **Representative Metrics (R_k):** A subset of metrics $R_k \subseteq \{m_1, \ldots, m_{d_M}\}$ that most strongly distinguish C_k from other clusters.
   - **Metric Relationships (T_k):** A set of salient dependencies (e.g., trade-offs or correlations) between metrics within C_k.
   - **Configuration Rules (H_k):** A small set of human-readable logical rules over the configuration variables $x_1, \ldots, x_{d_X}$ that approximate membership in C_k (e.g., simple conjunctions of threshold and equality predicates).
   - **Auxiliary Statistics (S_k):** Quantitative properties of the cluster, such as relative size, stability, and discriminativeness scores.
   - **Visual Highlights (V_k):** Concrete visual artifacts (e.g., radar charts, parallel coordinate plots) that instantiate the above descriptors for user inspection.

Desiderata. A satisfactory solution should meet three complementary criteria:

1. **Cluster Validity:** The partition $\mathcal{C}$ should be structurally sound, maximizing internal cohesion and external separation (e.g., as measured by the Silhouette Coefficient) to ensure operating behaviors are distinct [42].
2. **Descriptor Faithfulness:** The derived descriptors must provide a faithful view of the clusters: runs assigned to the same cluster should tend to share similar values on the representative metrics R_k and satisfy similar configuration patterns captured by H_k.
3. **Interpretability:** The highlights must be cognitively manageable. We seek to minimize the complexity of the rule sets (e.g., no. of predicates) and the number of representative metrics $|R_k|$.



run id	max depth	estimators	...	recall	calibration	AUC	disp. impact ratio (DIR)	...
1	3	100	...	0.92	0.56	0.91	0.2	...
2	3	80	...	0.91	0.4	0.9	0.5	...
3	5	100	...	0.95	0.3	0.93	0.4	...
4	20	100	...	0.56	0.92	0.56	0.9	...
5	20	80	...	0.4	0.91	0.5	0.95	...
6	20	100	...	0.3	0.95	0.42	0.93	...

Cluster 1 Summary

Size Characteristics
Contains 50% of the runs (3/6).
Distinctiveness : Excellent
Representative Metrics: 2
- Recall (High)
- Calibration (Low)

Companion Metrics:
- Recall: AUC
- Calibration: Disparate Impact Ratio

Trade-offs:
- Recall vs Calibration
- Recall vs Disparate Impact Ratio

Representative Configuration:
- max_depth <=5

Representtive Run: 2

Cluster 2 Summary

....

id	H_k
1	max_depth <=5
2	max_depth = 20

id	T_k
1	**recall** (≈ AUC) ↔ calibration, DIR
2	**calibration** (≈ DIR) ↔ recall, AUC

id	R_k	S_k
1	recall ↑ calibration ↓	p_1: 50% s_1: 1.0
2	recall ↓ calibration ↑	p_2: 50% s_2: 1.0

Figure 2: Overview of the proposed algorithmic pipeline. The method takes as input logged experiment runs with configurations and metrics, performs metric-space clustering, and then derives per-cluster highlights.

4. Experiment Run Clustering and Highlight Extraction

This section describes our method for organizing experiment runs into clusters and deriving cluster-level highlights over configuration parameters and metrics. Following [36], we first embed the logged runs into a low-dimensional metric space, which has been shown to yield more stable and interpretable clusters than operating directly on the raw metrics. We then apply metric-space clustering in this embedded space to identify distinct behaviors of the experiment, followed by a highlight extraction to summarize and interpret each cluster.

Figure 2 provides an overview of the analysis pipeline and an illustrative example with two resulting clusters. The top-left panel shows the input runs $\mathcal{E}$, including their logged configurations $\mathbf{X}$ (light red) and outcome metrics $\mathbf{M}$ (green), while the bottom-left panel presents the resulting cluster summaries as textual and visual highlights. The right side outlines the two main components of our method: a clustering component, which performs metric filtering, normalization, dimensionality reduction, and k-means clustering in metric space, and a highlight-extraction component, which derives representative metrics, relationships, and configuration-level rules for each cluster. In this toy example, runs 1–3 form a detection-oriented cluster, characterized by higher recall and Area Under the Curve (AUC) but worse fairness metrics, whereas runs 4–6 form a fairness-oriented cluster exhibiting the opposite pattern.

4.1. Metric-Space Clustering

The clustering component transforms the logged configuration and metric data into a numerical representation suitable for clustering and explanation, and then applies k-means clustering to obtain a partition $\mathcal{C}$ of the runs.

Variance-based metric filtering. To focus the analysis on informative metrics, we apply two filters based on the coefficient of variation (CV). For every metric m_j, we compute its sample mean and standard deviation over $\{1, \ldots, N\}$ and derive $\mathrm{CV}(m_j)$ as their ratio. We remove metrics with $\mathrm{CV}(m_j) < 0.05$ to eliminate effectively constant features that lack discriminative power [43, 44]. Conversely, we discard metrics with $\mathrm{CV}(m_j) > 1.5$ that indicate measurements where noise variance overwhelms the signal mean (implying a signal-to-noise ratio <0.67) [45]. The remaining metrics form a reduced matrix still denoted $\mathbf{M}$.

Encoding and normalization. Both $\mathbf{X}$ and $\mathbf{M}$ may contain continuous and categorical columns. We decompose $\mathbf{M} = [\mathbf{M}_{\text{cont}} \mid \mathbf{M}_{\text{cat}}]$ and $\mathbf{X} = [\mathbf{X}_{\text{cont}} \mid \mathbf{X}_{\text{cat}}]$, where the subscripts denote continuous and categorical parts, respectively. The continuous matrices $\mathbf{X}_{\text{cont}}$ and $\mathbf{M}_{\text{cont}}$ are standardized column-wise to zero mean and unit variance, while the categorical matrices $\mathbf{X}_{\text{cat}}$ and $\mathbf{M}_{\text{cat}}$ remain nominal and are handled via correspondence analysis in the subsequent metric-space embedding. For the model-based analyses in Section 4.2.1 and Section 4.2.2, we additionally construct encoded matrices $\mathbf{M}'$ and $\mathbf{X}'$ by one-hot encoding all categorical columns in $\mathbf{M}_{\text{cat}}$ and $\mathbf{X}_{\text{cat}}$ and concatenating them with their standardized continuous counterparts.

Metric-space embedding. We perform dimensionality reduction separately on the distinct metric types. Following [36], we apply Principal Component Analysis (PCA) to $\mathbf{M}_{\text{cont}}$, retaining components explaining at least 80% of the variance, and Multiple Correspondence Analysis (MCA) to $\mathbf{M}_{\text{cat}}$, retaining dimensions that describe 80% of the inertia. Let $\mathbf{Z}_{\text{cont}}$ and $\mathbf{Z}_{\text{cat}}$ denote the resulting PCA and MCA embeddings, respectively. We concatenate these embeddings to obtain a joint metric-space representation $\mathbf{Z} = [\mathbf{Z}_{\text{cont}} \mid \mathbf{Z}_{\text{cat}}] \in \mathbb{R}^{N \times d}$, where each row $\mathbf{z}_i$ summarizes the metric profile of run i in a common continuous space.

Clustering algorithm and selection of cluster number. We cluster the embedded runs $\mathbf{Z}$ using k-means, exploring a range of cluster counts $k \in [k_{\min}, k_{\max}]$ (in our experiments, $k_{\min} = 2$ and $k_{\max} = 9$) and set the number of clusters K to the k that maximizes the silhouette score [42]. For each cluster C_k, we record its size $|C_k|$ and relative size $p_k = |C_k|/N$. We use k-means (due to its efficiency and scalability) and silhouette (widely used) for extracting a small set of coarse regions, while the rest of the pipeline only assumes a run-to-cluster assignment and can use any clustering backend. In the running example of Figure 2, this procedure yields $K = 2$, separating runs 1–3 from runs 4–6 (detection-oriented vs. fairness-oriented behaviors).

4.2. Highlight Extraction

For each cluster in the fixed partition $\mathcal{C}$, we derive descriptors and assemble experiment highlights by selecting representative metrics R_k and their relationships T_k, extracting configuration rules H_k that approximate cluster membership, and compiling textual and visual summaries V_k.

4.2.1. Representative Metrics and Relationships

For each cluster C_k, our goal is to select a small, non-redundant set of metrics R_k that best distinguish runs in C_k from the rest of $\mathcal{E}$. We operate on the encoded metric matrix $\mathbf{M}'$ and treat the problem as a series of binary classification tasks.

Selection of representative metrics. For each cluster C_k, we iteratively select representative metrics from the encoded matrix $\mathbf{M}'$. In each of $\lfloor d_M/2 \rfloor$ iterations (with d_M the number of logged metrics), we set up a binary classification task (C_k vs. $\mathcal{E} \setminus C_k$), labeling runs in C_k as positives and all others as negatives, and train a random forest classifier to separate the two classes. We then compute the SHAP scores [39] for this classifier and identify the highest-ranked metric according to its global importance.

To avoid selecting redundant metrics, we group together strongly correlated candidates and keep only one representative per group. For metrics a and b within C_k, we define a type-aware dependence measure $r_k(a, b) \in [-1, 1]$ using Kendall correlation for continuous–continuous pairs, the square root of partial η^2 for categorical–continuous pairs, and normalized mutual information for categorical–categorical pairs [36]. If $r_k(a, b) \geq \tau_{\text{high}}$ (with $\tau_{\text{high}} = 0.75$), we treat b as a companion of a and remove all companions from the candidate pool before the next iteration, ensuring that subsequent iterations capture genuinely new aspects of C_k.

For each metric that is ultimately selected as representative, we additionally record how its typical value in C_k compares to the corresponding typical values in the other clusters by computing a cluster-level z-score: metrics with a z-score ≤ -1 are labeled *low*, those with a z-score ≥ 1 are labeled *high*, and those in between are labeled *mid*. In Figure 2, this procedure identifies *recall* and *calibration* as representative metrics for the two clusters: for Cluster 1, recall is labeled *high* and calibration *low*, while Cluster 2 exhibits the opposite pattern.

Cluster discriminativeness. To quantify how well the selected metrics separate C_k from the remaining runs, we train an XGBoost classifier on the reduced encoded metric matrix $\mathbf{M}'$ for the same binary task (C_k vs. $\mathcal{E} \setminus C_k$). We use an 80/20 stratified train–test split, perform 5-fold cross-validation on the training set, and evaluate on the held-out test set. The discriminativeness of cluster C_k is summarized by a scalar score $s_k = 0.6 \cdot \text{AUC}_k + 0.4 \cdot \text{F1}_k$, defined by the classifier's area under the ROC curve (AUC) and F_1-score. This scalar score $s_k \in [0, 1]$ is then used to categorize clusters by discriminativeness (very well distinguished for $s_k > 0.9$, well distinguished for $s_k \geq 0.7$, moderate distinction for $s_k > 0.5$, and poor discriminative power for $s_k \leq 0.5$) and is stored as part of the auxiliary statistics S_k.

Metric relationships. We summarize relationships between the metrics in R_k using the dependence measure $r_k(\cdot, \cdot)$. The groups of previously computed correlated companions of each representative metric in R_k, are used directly as part of the cluster summary, providing alternative but closely related views on the same underlying behavior. In addition, for each cluster C_k we compute pairwise relationships between the metrics in R_k to detect strong negative correlations that indicate potential trade-offs. Metric pairs with $r_k(a, b) \leq -\tau_{\text{high}}$ (e.g., $\tau_{\text{high}} = 0.75$) are recorded as cluster-specific trade-offs between competing objectives such as performance and resource usage. The resulting collection of correlated companion groups and trade-offs constitutes the metric-relationship summary T_k for cluster C_k. In the illustrative example, Cluster 1 exhibits a strong positive correlation between recall and AUC, at the expense of disparate impact ratio and calibration, whereas Cluster 2 exhibits the opposite correlation profile.

4.2.2. Configuration-Level Rule Extraction

While metric-level summaries describe how runs in a cluster behave, practitioners also need to understand which configuration patterns give rise to that behavior. To this end, we derive human-readable rules over the set of parameters, approximating membership in each cluster.

Rule extraction and scoring. Similar to our method in Section 4.2.1, we define a binary classification problem on the configuration space for each cluster C_k (runs in C_k vs. $\mathcal{E} \setminus C_k$) using the encoded configuration matrix $\mathbf{X'}$. We then learn an interpretable rule-based classifier by training an ensemble of shallow decision trees for this task and extracting those decision paths that attain sufficiently high precision and recall for the positive class.

Each selected path is converted into a human-readable logical rule, that is, a conjunction of simple predicates on x, that dictate membership in C_k. For each candidate rule r, we compute its F_1-score $\mathrm{F1}_k(r)$ for predicting membership in C_k together with its relative coverage $c_k(r) = n_k(r) \ / \ |C_k|$, where $n_k(r)$ is the number of runs in C_k that satisfy r. We then define the rule-quality score as $s_k(r) = c_k(r) \cdot \mathrm{F1}_k(r)$, and retain as configuration-level descriptors only those rules with $s_k(r) \geq 0.75$. The configuration-level descriptor for cluster C_k is the resulting small set of top-scoring rules, denoted H_k, which serves as an interpretable summary of the configuration behaviors associated with the cluster's metrics. In the illustrative example, the top-scoring rule for Cluster 1 is `max_depth` ≤ 5, while `max_depth` $= 20$ defines Cluster 2.

4.2.3. Visual Highlight Generation

Given the metric- and configuration-level descriptors for each cluster C_k, we assemble a compact set of visual highlights V_k. Each highlight card pairs a textual summary with coordinated visualizations, providing both an overview and detailed views of the cluster's behavior and configuration patterns. An example for Cluster 1 is shown in Figure 2, where the card highlights high recall and low calibration, the rule on `max_depth`, and the representative run 2.

The textual summary reports the cluster size $|C_k|$, proportion p_k, and score s_k, which is discretized into a quality label (e.g., *excellent*, *good*, *moderate*, *poor*). It then summarizes the representative metrics in R_k , enumerates key relationships in T_k, and lists the top-scoring configuration rule in H_k. Finally, the card highlights a single representative run from C_k (e.g., the cluster medoid in $\mathbf{Z}$) as a concrete example of a typical configuration and outcome profile. To complement the summary, we generate three coordinated views per cluster. We present R_k (and companions T_k) as an interactive radar chart, allowing the user to toggle between views that distinctively show metrics where the cluster attains notably *high*, *mid*, or *low* values. A graph view visualizes correlations in T_k, scaling edges by $r_k(a, b)$, while a parallel-coordinates plot maps parameters x to metrics R_k, highlighting runs matching the top rule in H_k. In all views, non-selected clusters (or runs) are rendered as a desaturated, low-opacity gray backdrop, while the focused cluster is overlaid in color with higher opacity and thicker strokes, preserving global context while emphasizing cluster-specific patterns.

5. Experimental Evaluation

5.1. Experimental Setup

The goal of our evaluation is to assess the proposed method's performance across three key dimensions: the structural quality of discovered clusters, the stability of representative metrics, and the interpretability of the generated rule-based explanations.

5.1.1. ML Pipelines

Three representative ML pipelines were executed to generate experiment runs for evaluating our method. We first tested an income classification pipeline on the Adult dataset [46]. Following [31], we also ran a wine classification on the UCI wine dataset [47], and taxi fare prediction on New York City trip data from the Taxi and Limousine Commission [48]. Each pipeline follows a standard pattern of data loading, train–test splitting, hyperparameter tuning, and metric logging [31], with specific dataset properties and execution counts summarized in Table 1. Throughout this section, we refer to these pipelines as Inc, Wine and Taxi respectively.

Dataset	Task	Target	Model	Params.	Metrics	Runs
Adult [46]	Classification	Income (>$50k)	Fairness-aware RF	8	32	108
Wine [47]	Classification	Wine Type	Random Forest	7	45	242
Taxi [48]	Regression	Fare Amount	Random Forest	5	31	360

Table 1
Summary of experimental pipelines and execution volume.

5.1.2. Evaluation Metrics

We evaluate our pipeline across four dimensions: (i) **Cluster Quality**, using Silhouette Score (SiS) and Davies–Bouldin Index (DBI)—the most commonly adopted internal validation metrics throughout the literature [42, 49, 50, 51], where higher SiS and lower DBI indicate better-defined clusters; (ii) **Representative Metric Quality**, using the ratio of the representatives' Coefficient of Variation (CV) to the CV of non-selected metrics within the same cluster [44], where lower ratios indicate more stable representatives; (iii) **Rule Interpretability**, using Coverage, Separation Error, and Conciseness, aggregated as $\mathrm{QSE}(E_c) = \big(\mathrm{Coverage}(E_c) + (1 - \mathrm{SeparationErr}(E_c)) + \mathrm{Conciseness}(E_c)\big)/3$ [37]; and (iv) **Runtime**, reporting total end-to-end wall-clock time (sec) per configuration.

5.1.3. Implementation Details.

This work was implemented in Python 3.11. The experiments were executed on a MacBook Pro M3 [1]. Source code and reproducibility instructions can be found on Github [2].

[1]https://support.apple.com/en-us/117735
[2]https://github.com/billstam12/workflow-insight-extraction

Table 2
Clustering, stability, predictive, and explanation aggregate metrics across pipelines and ablations.

Pipeline	Ablation	$\|\mathcal{C}\|$	$\overline{\|R_k\|}$	$\overline{\text{SiS}}$	SiS_{min}	SiS_{max}	$\overline{\text{DBI}}$	$\overline{\text{CV}}$	CV_{min}	CV_{max}	$\overline{\text{QSE}}$	Time (sec)
Inc	FULL	4	5	**0.9216**	**0.8153**	**0.9656**	**0.1395**	**0.0271**	**0.0273**	**0.0153**	**0.9265**	10.17
Inc	NDR	2	N/A	0.0106	-0.7027	0.3027	0.9382	N/A	N/A	N/A	0.8867	**6.16**
Inc	NIS	4	9	0.9216	0.8153	0.9656	0.1395	0.2283	0.0899	0.3055	0.9265	9.00
Inc	NVF	9	5	0.6356	0.0000	0.8039	0.4704	0.0652	0.0304	0.1089	0.6990	17.30
Taxi	FULL	4	3	**0.8044**	**0.5723**	**0.9742**	**0.2305**	0.2655	0.0099	0.2975	**0.8613**	10.18
Taxi	NDR	4	2	0.3974	-0.2092	0.8618	1.0542	**0.0903**	0.0084	**0.1069**	0.8465	10.45
Taxi	NIS	4	9	0.8044	0.5723	0.9742	0.2305	0.3622	0.1652	0.3284	0.8613	**8.70**
Taxi	NVF	17	3	0.6436	-0.2423	0.9645	0.3998	0.1521	**0.0000**	0.1252	0.7679	30.30
Wine	FULL	3	6	0.9078	0.7700	**0.9487**	**0.1319**	0.4587	0.3577	0.5293	**0.9512**	9.47
Wine	NDR	2	N/A	0.0298	-0.2279	0.2256	8.0719	N/A	N/A	N/A	0.6407	10.86
Wine	NIS	3	10	0.9078	0.7700	0.9487	0.1319	1.8436	2.3536	1.4172	0.9512	**8.40**
Wine	NVF	3	9	**0.9128**	**0.7971**	0.9478	0.1405	**0.3222**	**0.1013**	**0.4392**	0.9512	13.45

5.2. Ablation Study

To evaluate our algorithmic pipeline and quantify the contribution of each component, we perform an ablation study. We consider four configurations per pipeline: the full method (FULL); *No Dim. Reduction* (NDR), which omits PCA/MCA and clusters directly in the original metric space; *No Variance Filter* (NVF), which keeps all metrics before clustering regardless of variance; and *No Iterative Selection* (NIS), which uses the same clustering steps as FULL but replaces iterative representative selection with a single-pass SHAP ranking. Table 2 reports, for each pipeline and ablation setting, the number of clusters $|\mathcal{C}|$, the average number of representative metrics $\overline{|R_k|}$, internal clustering indices (SiS and DBI), representative-metric variability (CV), rule-based explanation quality (QSE), and total execution time; rows for FULL are shaded in blue, and best values within each pipeline are typeset in bold.

5.2.1. Cluster Quality

Across all pipelines, FULL yields the strongest cluster structure: Inc and Wine achieve average SiS above 0.90 with the lowest DBI (around 0.14), while Taxi attains SiS $\approx$ 0.80 with DBI $\approx$ 0.23. In contrast, NDR consistently collapses cluster quality (SiS close to 0 and DBI up to 8.07 for Wine), and NVF either inflates the number of weak clusters (e.g., Taxi with $K = 17$) or worsens SiS/DBI. The NIS variant operates on the same cluster assignments as FULL and therefore matches its SiS and DBI, but—as discussed next—differs substantially in representative quality.

5.2.2. Representative Metric Quality

FULL provides the most stable and compact set of representatives, especially on Inc, where it uses 5 representative metrics and attains the smallest CV-ratio values (average ratio $\approx$ 0.03 with a very narrow range), indicating that representatives vary far less than non-selected metrics within each cluster. For the rest of the pipelines, FULL also balances the number of representatives (5 on average) with low or moderate CV-ratios, whereas ablations either fail to define meaningful representatives (NDR), substantially increase the CV-ratio (NIS), or require more representatives to describe the same clusters making the process cognitively unmanageable.

5.2.3. Rule Interpretability

Rule-based explanations are consistently strongest under FULL, which achieves the highest QSE across pipelines. Ablations either sharply reduce QSE (especially NDR and NVF) or, in the case of NIS, match FULL's QSE as they use the same cluster assignments; proving that dimensionality reduction and variance filtering are crucial in producing interpretable clusters.

5.2.4. Runtime

FULL remains computationally practical, completing in around 10sec. across pipelines. In contrast, NVF is consistently the slowest variant (Inc: 17.30s, Taxi: 30.30s, Wine: 13.45s), indicating that retaining all metrics substantially increases end-to-end cost. NIS and NDR are the fastest, but FULL is within roughly 2 seconds while delivering substantially better representative stability (lower CV ratios) and strong explanation quality.

5.3. Summary

Overall, the full configuration is the only one that consistently balances (i) well-formed clusters, (ii) stable and compact representative metrics, and (iii) high-quality rule explanations across all three pipelines. The ablations confirm that dimensionality reduction and variance filtering are important for maintaining meaningful structure and interpretability, while replacing iterative selection degrades the faithfulness/compactness of representatives even when cluster assignments are unchanged. End-to-end runtime remains practical for FULL, and retaining all metrics (NVF) is consistently the most expensive setting.

6. Conclusions

This paper presented an approach that organizes large collections of experiment runs into metric-space clusters and, for each cluster, derives compact textual and visual highlights over representative metrics, metric relationships, and configuration-level rules. The pipeline first filters and embeds metrics and applies k-means clustering to obtain compact, well-separated operating regimes. It then identifies representative metrics using iterative SHAP-based feature extraction, learns configuration rules from shallow decision trees, and assembles chart- and text-based highlights. Experiments on three ML pipelines showed that the proposed method attains high clustering quality (Silhouette Score > 0.9) while using only a handful (3–6) of representative metrics per cluster and short rules with few predicates in reasonable time; and that removing dimensionality reduction, variance filtering, or iterative selection degrades at least one of these properties. Future work includes extending our method with more clustering algorithms, and cluster quality metrics like Dunn Index and WCSS [50, 51].

Acknowledgments

This research is implemented in the framework of H.F.R.I call "3rd Call for H.F.R.I.'s Research Projects to Support Faculty Members & Researchers" (H.F.R.I. Project Number: 23640).

Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

References

[1] F. Suter, T. Coleman, I. Altintas, R. M. Badia, B. Balis, K. Chard, I. Colonnelli, E. Deelman, P. D. Tommaso, T. Fahringer, C. A. Goble, S. Jha, D. S. Katz, J. Köster, U. Leser, K. Mehta, H. Oliver, J. L. Peterson, G. Pizzi, L. Pottier, R. Sirvent, E. Suchyta, D. Thain, S. R. Wilkinson, J. M. Wozniak, R. F. da Silva, A terminology for scientific workflow systems, Future Gener. Comput. Syst. 174 (2026) 107974.

[2] M. Zaharia, A. Chen, A. Davidson, A. Ghodsi, S. A. Hong, A. Konwinski, S. Murching, T. Nykodym, P. Ogilvie, M. Parkhe, F. Xie, C. Zumar, Accelerating the machine learning lifecycle with mlflow, IEEE Data Eng. Bull. 41 (2018) 39–45.

[3] I. Weights & Biases, Weights & biases documentation, https://wandb.ai/, 2025. Accessed: 2025-15-12.

[4] N. Labs, Neptune: Experiment tracking tool for ml teams, https://neptune.ai/, 2025. Accessed: 2025-15-12.

[5] P. Vassiliadis, V. Peralta, P. Marcel, D. Gkitsakis, A. Dougia, F. E. Outa, A conceptual model for data analysis highlights, in: Companion Proceedings of the 44th International Conference on Conceptual Modeling: Industrial Track, ER Forum, 8th SCME, Doctoral Consortium, Tutorials, Project Exhibitions, Posters and Demos. Co-located with ER 2025 Poitiers, France, October 20-23, 2025., 2025.

[6] S. Sarawagi, Explaining differences in multidimensional aggregates, in: VLDB'99, Proceedings of 25th International Conference on Very Large Data Bases, September 7-10, 1999, Edinburgh, Scotland, UK, 1999, pp. 42–53.

[7] S. Sarawagi, User-adaptive exploration of multidimensional data, in: VLDB 2000, Proceedings of 26th International Conference on Very Large Data Bases, September 10-14, 2000, Cairo, Egypt, 2000, pp. 307–316.

[8] G. Sathe, S. Sarawagi, Intelligent rollups in multidimensional OLAP data, in: VLDB 2001, Proceedings of 27th International Conference on Very Large Data Bases, September 11-14, 2001, Roma, Italy, 2001, pp. 531–540.

[9] D. Gkesoulis, P. Vassiliadis, P. Manousis, Cinecubes: Aiding data workers gain insights from OLAP queries, Inf. Syst. 53 (2015) 60–86.

[10] B. Tang, S. Han, M. L. Yiu, R. Ding, D. Zhang, Extracting top-k insights from multi-dimensional data, in: Proceedings of the 2017 ACM International Conference on Management of Data, SIGMOD Conference 2017, Chicago, IL, USA, May 14-19, 2017, 2017, pp. 1509–1524.

[11] R. Ding, S. Han, Y. Xu, H. Zhang, D. Zhang, Quickinsights: Quick and automatic discovery of insights from multi-dimensional data, in: P. Boncz, S. Manegold, A. Ailamaki, A. Deshpande, T. Kraska (Eds.), Proceedings of the 2019 International Conference on Management of Data, SIGMOD Conference 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019, ACM, 2019, pp. 317–332.

[12] J. Xing, X. Wang, H. V. Jagadish, Data-driven insight synthesis for multi-dimensional data, Proc. VLDB Endow. 17 (2024) 1007–1019.

[13] Y. Wang, Z. Sun, H. Zhang, W. Cui, K. Xu, X. Ma, D. Zhang, Datashot: Automatic generation of fact sheets from tabular data, IEEE Trans. Vis. Comput. Graph. 26 (2020) 895–905.

[14] P. Ma, R. Ding, S. Han, D. Zhang, Metainsight: Automatic discovery of structured knowledge for exploratory data analysis, in: SIGMOD '21: International Conference on Management of Data, Virtual Event, China, June 20-25, 2021, ACM, 2021, pp. 1262–1274.

[15] D. Shi, X. Xu, F. Sun, Y. Shi, N. Cao, Calliope: Automatic visual data story generation from a spreadsheet, IEEE Trans. Vis. Comput. Graph. 27 (2021) 453–463.

[16] M. Sun, L. Cai, W. Cui, Y. Wu, Y. Shi, N. Cao, Erato: Cooperative data story editing via fact interpolation, IEEE Trans. Vis. Comput. Graph. 29 (2023) 983–993.

[17] H. Li, L. Ying, H. Zhang, Y. Wu, H. Qu, Y. Wang, Notable: On-the-fly assistant for data storytelling in computational notebooks, in: CHI, 2023.

[18] P. Ma, R. Ding, S. Wang, S. Han, D. Zhang, Insightpilot: An llm-empowered automated data exploration system, in: EMNLP'2023, 2023.

[19] S. Idreos, O. Papaemmanouil, S. Chaudhuri, Overview of data exploration techniques, in: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, 2015, pp. 277–281.

[20] O. B. El, T. Milo, A. Somech, ATENA: an autonomous system for data exploration based on deep reinforcement learning, in: Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM 2019, Beijing, China, November 3-7, 2019, 2019, pp. 2873–2876.

[21] A. Personnaz, S. Amer-Yahia, L. Berti-Équille, M. Fabricius, S. Subramanian, DORA THE EXPLORER: exploring very large data with interactive deep reinforcement learning, in: CIKM '21: The 30th ACM International Conference on Information and Knowledge Management, Virtual Event, Queensland, Australia, November 1 - 5, 2021, 2021, pp. 4769–4773.

[22] T. D. Bie, L. D. Raedt, J. Hernández-Orallo, H. H. Hoos, P. Smyth, C. K. I. Williams, Automating data science, Commun. ACM 65 (2022) 76–87.

[23] S. Amer-Yahia, P. Marcel, V. Peralta, Data narration for the people: Challenges and opportunities, in: Proceedings 26th International Conference on Extending Database Technology, EDBT 2023, Ioannina, Greece, March 28-31, 2023, OpenProceedings.org, 2023, pp. 855–858.

[24] S. Amer-Yahia, Intelligent agents for data exploration, Proc. VLDB Endow. 17 (2024) 4521–4530.

[25] T. Lipman, T. Milo, A. Somech, T. Wolfson, O. Zafar, LINX: A language driven generative system for goal-oriented automated data exploration, in: Proceedings 28th International Conference on Extending Database Technology, EDBT 2025, Barcelona, Spain, March 25-28, 2025, OpenProceedings.org, 2025, pp. 270–283.

[26] E. Deelman, A. Mandal, M. Jiang, R. Sakellariou, The role of machine learning in scientific workflows, Int. J. High Perform. Comput. Appl. 33 (2019).

[27] V. Silva, D. de Oliveira, P. Valduriez, M. Mattoso, Dfanalyzer: runtime dataflow analysis of scientific applications using provenance, Proc. VLDB Endow. 11 (2018) 2082–2085.

[28] L. Sacco, C. Sopranzetti, S. Fiore, Enabling provenance tracking in workflow management systems, in: W. Ding, C. Lu, F. Wang, L. Di, K. Wu, J. Huan, R. Nambiar, J. Li, F. Ilievski,

R. Baeza-Yates, X. Hu (Eds.), IEEE International Conference on Big Data, BigData 2024, Washington, DC, USA, December 15-18, 2024, IEEE, 2024, pp. 4402–4409.

[29] S. Köhler, S. Riddle, D. Zinn, T. M. McPhillips, B. Ludäscher, Improving workflow fault tolerance through provenance-based recovery, in: J. B. Cushing, J. C. French, S. Bowers (Eds.), Scientific and Statistical Database Management - 23rd International Conference, SSDBM 2011, Portland, OR, USA, July 20-22, 2011. Proceedings, volume 6809 of *Lecture Notes in Computer Science*, Springer, 2011, pp. 207–224.

[30] J. Starlinger, B. Brancotte, S. Cohen-Boulakia, U. Leser, Similarity search for scientific workflows, Proceedings of the VLDB Endowment 7 (2014) 1143–1154.

[31] M. Schlegel, K.-U. Sattler, Capturing end-to-end provenance for machine learning pipelines, Information Systems 132 (2025) 102495.

[32] S. Redyuk, Z. Kaoudi, S. Schelter, V. Markl, DORIAN in action: Assisted design of data science pipelines, Proc. VLDB Endow. 15 (2022) 3714–3717.

[33] V. Michalakopoulos, I. Papias, E. Sarantinopoulos, E. Sarmas, V. Marinakis, D. Askounis, A hyperparameter-space clustering methodology of residential electricity loads, Appl. Soft Comput. 181 (2025) 113497.

[34] J. Bader, L. Thamsen, S. Kulagina, J. Will, H. Meyerhenke, O. Kao, Tarema: Adaptive resource allocation for scalable scientific workflows in heterogeneous clusters, in: 2021 IEEE International Conference on Big Data (Big Data), 2021, pp. 65–75.

[35] Y. Kumar, S. Kaul, Y.-C. Hu, Machine learning for energy-resource allocation, workflow scheduling and live migration in cloud computing: State-of-the-art survey, Sustainable Computing: Informatics and Systems 36 (2022) 100780.

[36] M. Alvarez-García, R. Ibar-Alonso, M. A. Parra, A comprehensive framework for explainable cluster analysis, Inf. Sci. 663 (2024) 120282.

[37] S. Ofek, A. Somech, Explaining black-box clustering pipelines with cluster-explorer, Proc. VLDB Endow. 18 (2025) 1495–1508.

[38] M. Guilbert, C. Vrain, T.-B.-H. Dao, A Constrained Declarative Based Approach for Explainable Clustering, in: Symposium on Intelligent Data Analysis (IDA 2025), Konstanz, Germany, 2025. URL: https://hal.science/hal-04995593.

[39] S. M. Lundberg, S.-I. Lee, A unified approach to interpreting model predictions, in: Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS'17, Curran Associates Inc., Red Hook, NY, USA, 2017, p. 4768–4777.

[40] Y. Deng, Y. Wang, L. Cao, L. Qiao, Y. Wang, X. Jingzhe, Y. Yan, S. Madden, Outlier summarization via human interpretable rules, Proc. VLDB Endow. 17 (2024) 1591–1604.

[41] S. Maroulis, V. Stamatopoulos, P. Gidarakos, K. Tsopelas, N. Masouras, K. Kozanis, N. Theologitis, G. Papastefanatos, G. Giannopoulos, E. G. Nilsson, Experimentlens: Interactive visual analytics and explainability for ml experiment management, Proceedings of the VLDB Endowment. ISSN 2150 (2025) 8097.

[42] P. J. Rousseeuw, Silhouettes: A graphical aid to the interpretation and validation of cluster analysis, Journal of Computational and Applied Mathematics 20 (1987) 53–65.

[43] M. Kuhn, K. Johnson, Applied Predictive Modeling, Springer, 2013. Discusses removal of near-zero variance predictors.

[44] C. Stepniak, Coefficient of Variation, Springer Berlin Heidelberg, Berlin, Heidelberg, 2025, pp. 487–488.

[45] D. A. Skoog, F. J. Holler, S. R. Crouch, Principles of Instrumental Analysis, Cengage Learning, 2017. Discusses Signal-to-Noise Ratio and precision limits.
[46] B. Becker, R. Kohavi, Adult, UCI Machine Learning Repository, 1996. DOI: https://doi.org/10.24432/C5XW20.
[47] S. Aeberhard, M. Forina, Wine, UCI Machine Learning Repository, 1992. DOI: https://doi.org/10.24432/C5PC7J.
[48] New York City Taxi and Limousine Commission, TLC Trip Record Data, https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page, 2024. Accessed: 2025-15-12.
[49] D. L. Davies, D. W. Bouldin, A cluster separation measure, IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-1 (1979) 224–227.
[50] J. C. Dunn, Well-separated clusters and optimal fuzzy partitions, Journal of Cybernetics 4 (1974) 95–104.
[51] B. S. Everitt, S. Landau, M. Leese, D. Stahl, Cluster Analysis, 5th ed., Wiley, 2011.