Interpretable Attribute Discretization

Abstract

Data discretization, the conversion of numeric attributes into categorical bins, underpins many data-driven tasks, from visualization and causal inference to symbolic learning. *Interpretable partitions* make results easier to communicate and act upon. Yet, a fundamental tension underlies this discretization process: partitions that are interpretable to humans often differ from those that maximize utility performance (e.g., causal inference). Despite the abundance of discretization techniques, reconciling this utility-semantics trade-off remains an open challenge.

In this work, we formally define the interpretable attribute discretization problem and introduce a reinforcement-learning-based framework that jointly optimizes utility and semantic fidelity. Our approach *learns a partition selection policy* that can navigate a vast space of candidates and accurately estimate the optimal partitions by strategically sampling only 200 candidates out of millions.

We measure success by the average Hausdorff distance between the estimated and ground-truth *Pareto front* of the utility–semantics trade-off. On seven datasets across four tasks (visualization, modeling, imputation, causal inference) and multiple semantic measures, our method achieves the lowest overall error, typically **2–4× smaller** than strong alternatives, yielding a more accurate Pareto front that reveals the complete utility–semantics trade-off.

CCS Concepts

• **Information systems** → **Information integration**; **Wrappers (data mining)**; *Data cleaning*.

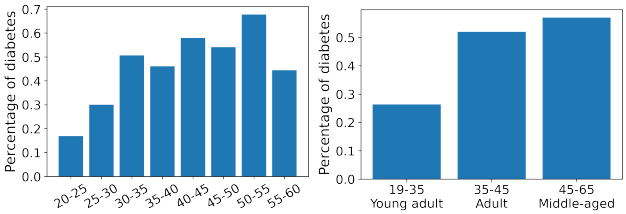
Keywords

Attribute Discretization, Semantic Interpretability, Reinforcement Learning, Multi-Objective Optimization

1 Introduction

Data discretization maps numeric attributes to categorical bins, turning raw values into intervals. For example, a continuous Age attribute can be grouped by decades (e.g., "0-10", "10-20") or life stages, such as "Young adult", "Adult", or "Middle-aged". This transformation underpins many data-driven tasks, such as data visualization, causal inference, and symbolic learning [16], as it reduces noise and highlights structural patterns in the data.

Crucially, discretization serves two complementary yet often competing goals. (i) Utility: the discretized data support strong quality in downstream statistical or learning tasks; and (ii) Interpretability: the resulting bins match human intuition and are semantically meaningful. For example, an automated splitter (e.g., ChiMerge) picks Age cuts at $(20, 24, 30, 33, 42, 43, 47, 48, 54, 61)$, yielding top validation accuracy for diabetes risk. Yet these jagged intervals resist interpretable justification, and small cut shifts change categories—hard to explain or defend to clinicians. Whereas, using guideline-aligned bins—e.g., Age $[19, 35)$, $[35, 45)$, $[45, 65)$—slightly trails the best accuracy, but produces stable, policy-relevant groups [31] that support clear triage rules, reporting, and interventions. *Interpretable discretization* thus acts as **a bridge between raw numerical values and human-readable patterns**.



(a) Age **binned by equal width.** **(b)** Age **binned by life stages.**

Figure 1: (a) user's attempt; (b) our output in Ex.3.

Data-sharing efforts in science and governance further underscore the need for meaningful discretization. Standard databases often reflect *untargeted* design choices and must be adapted to specific analytical goals [16]. **Poor discretization can lead to low utility or limited interpretability**: bins that miss key patterns degrade utility, while semantically unclear bins hinder communication of insights. The following examples illustrate these challenges.

Example 1. Alex, an analyst in a research team, examines a diabetes dataset to understand the percentage of diabetes diagnoses w.r.t age for people between 21 and 60. She hopes to find an interpretable partition that reveals interesting trends in the data. Here, the utility of the visualization task is measured by the correlation between age and the probability of having diabetes. Alex starts with a standard equal-width method to bin Age as: $(20, 25, 30, \ldots, 60)$ (Fig. 1a). She sees no clear trend and finds it difficult to meaningfully label the bins. Alex knows there are countless ways to bin Age and wonders if at least one is satisfactory. Verifying her work without extensive trial and error is impossible. □

Example 2. In a separate task, Alex is now training a tree-based model for diabetes diagnosis, with the partitioned Age, BMI, and Glucose attributes, in the hope of finding interpretable and useful patterns in the data to guide diabetes diagnosis. She starts with an automated machine learning package and finds that the partition $(20, 24, 30, 33, 42, 43, 47, 48, 54, 61)$ for Age by ChiMerge [23] achieves the highest accuracy of 86%. After consulting with a diabetes expert, Alex realizes that this high-accuracy partition does not make much sense in medical contexts. She wonders if she can quickly find other partitions that maintain accuracy while improving the interpretability of the model and results. □

Together, these examples highlight a core dilemma: statistical discretization methods often fall short when both utility and interpretability matter. In Ex. 1, equal-width binning produces semantically uninformative bins that obscure trends. In Ex. 2, a high-accuracy ChiMerge partition yields clinically meaningless intervals.

Conference acronym 'XX, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/2018/06
https://doi.org/XXXXXXX.XXXXXXX

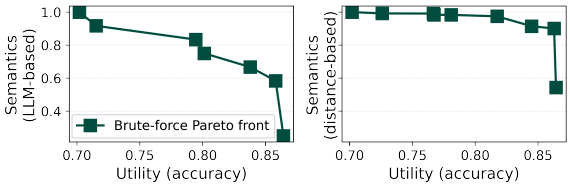


Figure 2: Utility–semantics trade-off for Example 4. Each marker represents an optimal partition set from a brute-force search over 313,820 candidates for `Age`, `BMI`, and `Glucose`. Trade-offs vary slightly across interpretability measures (left: LLM-based; right: distance-based), highlighting the need for a framework robust to different semantic metrics.

Despite the abundance of discretization techniques [16], existing methods offer limited help in situations like Alex's, where both goals are critical for downstream analysis and decision-making.

Broadly, existing approaches fall into two categories. (i) *Statistical methods*, such as equal-width or ChiMerge, optimize task metrics but often ignore domain knowledge, leading to bins that may be high-utility but uninterpretable. (ii) *Domain-specific methods* rely on handcrafted cutoffs, such as life-stage for age, which match human intuition but can degrade downstream utility. Reconciling this utility–semantics trade-off remains open.

Our Approach. We present an *interpretable attribute discretization approach* that bridges the gap by offering an automatic framework to discover *useful and interpretable* discretization strategies (referred to as partitions). In our framework, the user **inputs**: (1) a dataset with the numerical attributes to be discretized, (2) a downstream task with a corresponding utility measure (e.g., model accuracy for tree-based modeling, or interestingness measure for data visualization). Our framework **outputs** a set of attribute-partition pairs (defining how to bin each attribute) that jointly optimize utility and interpretability. We revisit Ex. 1 and 2 to illustrate. Consider what happens when Alex uses our *interpretable-discretization* system:

Example 3. Alex configures the system to measure utility using the Spearman correlation [18] between the percentage of diabetes diagnoses and `Age`. By default, the interpretable-discretization system constructs a *candidate space* using a mix of statistical discretizers (e.g., equal-width, ChiMerge) and domain-informed proposals (e.g., life-stage bins surfaced via curated heuristics/LLM). From this pool, the system selects a life-stage partition $(19, 35, 45, 65)$ with labels ("Young adult," "Adult," "Middle-aged") (Fig. 1b), commonly used in medical literature [31]. Alex quickly notices a sharp jump from young adult to adult, with a milder increase to middle-aged, yielding a visualization that is both interesting and interpretable. □

Example 4. Alex again trains a tree-based model with partitioned `Age`, `BMI`, `Glucose`. She configures the system to measure utility using model accuracy. The system then returns a set of attribute-partition pairs that balances both goals: `Age` $(19, 35, 45, 60)$ ("Young adult," "Adult," "Middle-aged"); `BMI` $(18.5, 25, 30, 68)$ ("Healthy," "Overweight," "Obese"); and `Glucose` $(0, 140, 200)$ ("Normal," "Impaired"). The resulting model attains 70% accuracy, which meets Alex's accuracy tolerance while substantially improving interpretability. Here, BMI has only one medical-aware partition, and the system finds it even though Alex is unaware of this domain-specific. Fig. 2 illustrates the accuracy-semantics trade-offs for this scenario. □

The scenarios above suggest a naïve solution: exhaustively enumerate partitions per attribute and score every candidate by utility and interpretability. This brute-force approach is computationally infeasible even at moderate scales, as the number of candidates grows exponentially with the attribute domains and with the number of attributes. Moreover, jointly maximizing utility *and* interpretability is often ill-posed—improving one can degrade the other—so a single optimum may not exist (as illustrated in Fig. 2). Instead, our goal is to efficiently approximate the *Pareto frontier* [34] of partition sets that best trade off utility and interpretability. To obtain such a system, we identify several challenges: **(C1)** being *task-agnostic* via a general, modular framework (applicable across data-driven tasks including modeling, causal inference, etc.); **(C2)** defining a robust, pluggable semantic interpretability measure for partitions; and **(C3)** navigating the exponentially large search space to discover partitions that balance both objectives.

To address **(C1)**, we provide a modular framework that treats utility as a pluggable component. Users instantiate it per task (e.g., accuracy or AUC for predictive modeling, correlation strength for exploration, or variance reduction for causal inference), without changing the overall architecture.

To address **(C2)**, we design a pluggable semantic module that can use *any* semantic similarity function; in this paper, we demonstrate two instantiations: (i) distance-based metrics (e.g., KL divergence) and (ii) LLM-based judgments that capture domain knowledge (e.g., common age brackets in healthcare). While our implementation focuses on these two, our framework is extensible to other choices (e.g., ontology- or feedback-driven scores).

To address **(C3)**, we propose a reinforcement-learning-based (RL-based) search algorithm for navigating the vast search space. Evaluating the utility of a partition w.r.t a downstream task can be expensive. For example, predictive modeling requires training a model. Therefore, our goal is to assess the utility of as few partitions as possible. To this end, our framework first curates a diverse candidate pool by combining statistical discretization techniques [23] with semantically grounded suggestions from LLMs [38]. These candidates are then represented as distributions and grouped using density-based clustering with Earth Mover's Distance, to reduce the search space. Finally, we cast partition search as a Markov Decision Process (MDP) over clusters. The key observation is that evaluating a partition from one cluster provides evidence about others in that cluster. We adopt the Monte Carlo Control (MCC) algorithm and provide conditions under which the approach exhibits a provable quality guarantee. Our framework balances exploration of new clusters with exploitation of promising ones, allowing us to efficiently discover Pareto-optimal partition sets.

A demonstration of our system usability and its suitability for end-to-end employment was recently published[1], which provides only a high-level description of the system, whereas the present paper provides the theoretical foundations and algorithms underlying the demonstrated system, as well as the experimental study.

Contributions. Our main contributions are as follows:

[1]Details are omitted due to double-blind review restrictions.

- We formalize *interpretable attribute discretization* as a multi-objective optimization over utility and semantic fidelity, and evaluate success by the average Hausdorff distance [47] between the estimated and ground-truth Pareto fronts.
- We present an RL-based framework that *learns a selection policy* over clustered candidate partitions. By clustering partitions and sharing evidence within clusters, Quill explores the space efficiently while preserving interpretability. We provide conditions under which the approach exhibits a provable quality guarantee.
- We instantiate scalable components, including distribution- and value-based clustering with auto-tuned hyperparameters, and budgeted exploration to control evaluation cost.
- On seven real-world datasets across four tasks and multiple semantic measures, our system achieves the lowest overall error, typically **2–4× smaller** than strong baselines [38, 58], by strategically sampling only 200 candidates out of millions. The resulting Pareto fronts expose the full utility–semantics trade-off, enabling users to select partitions that match task-specific preferences.

2 Related Work

Data discretization. There are numerous statistical data discretization methods. [16] evaluates the effect of 30 discretizers across 40 datasets on the accuracy of machine learning models. These methods range from the widely used ones (e.g., Equal Width, ChiMerge [23]) to more sophisticated ones (e.g., distance-based discretizers [11], dependency-preserving approaches [35], correlation-preserving discretizers [32]). In this work, we consider existing discretization methods as a *search space*, from which we identify partitions that are interpretable and useful for downstream tasks. We experiment with the commonly used 11 discretizers [16] along with partitions suggested by LLMs (Section 6.1).

Interpretable AI. A large body of work studies interpretable models and when interpretability matters. Survey [13] articulates formal definitions of interpretability, evaluation taxonomies, and when interpretability matters in practice. Approaches such as TreeFARMS [58], which explores sets of sparse decision trees within the "Rashomon set" of near-optimal models, provide transparency by enumerating multiple high-performing models rather than relying on a single opaque one. Other works like CORELS [53] and Bayesian Rule Lists [27] offer sparse, human-readable logic; post hoc explanation methods like LIME [41] and SHAP [29] provide human-understandable rationales.

However, even when the model class is interpretable, the input features (e.g., discretized attributes) can lack semantic meaning. Many existing methods (including TreeFARMS) require *post-processing* of continuous features or handcrafted discretization to improve interpretability. Our framework differs by producing semantically meaningful partitions *before* modeling, enabling end-to-end interpretability of both features and models (Sec 6).

Automated data visualization (AutoViz) and automated machine learning (AutoML). AutoViz systems assist users in generating effective visualizations with minimal manual intervention. Systems like SeeDB [54] automatically recommend visualizations that maximize deviation between subsets, while others like Voyager [57] and Draco [36] guide chart generation using formal grammars and perceptual principles. A persistent challenge in AutoViz is the transformation of numerical attributes into human-understandable categories. Oscar [50] addresses this by learning semantic binning strategies from past visualizations, but its reliance on historical patterns limits applicability to unseen attributes.

AutoML frameworks automate the design of machine learning pipelines, including feature preprocessing and model selection [15]. While many pipelines include discretization as a preprocessing step, these methods—such as ChiMerge [23] or MDL-based binning [14]—optimize purely for predictive accuracy. Recent systems like DeepLine [19] and GAMA [17] offer end-to-end automation but treat discretization as a low-level operation with limited interpretability. Some efforts in interpretable ML attempt to recover meaningful bins post hoc [25].

In this work, we demonstrate that our generic framework can efficiently identify interpretable and utility-preserving partitions for various downstream tasks, while letting users adjust the trade-off between utility and interpretability.

Discretization for causal inference. In causal inference, discretization plays a crucial role in defining treatment groups, confounder strata, or covariate balance [20, 22]. Many causal estimators—such as matching or stratification—rely on grouped versions of continuous covariates to approximate conditional independence or reduce variance. However, inappropriate binning can introduce residual confounding, increase bias, or distort identification assumptions [21, 42]. Our framework identifies partitions that are both useful for causality tasks (e.g., estimating treatment effects) and meaningful to domain experts, enabling transparent causal analysis and communication.

3 Problem Formulation

The goal of the Interpretable Attribute Discretization Optimal Partition Sets problem is to identify meaningful discretizations of numerical attributes that balance two objectives: maximizing the utility of a given downstream task and ensuring semantic interpretability of the resulting partitions. We begin by introducing the notations and formal problem definition.

3.1 Partitions

Let $\mathbb{A}$ denote the schema of a dataset (i.e., table) D, where each attribute $A \in \mathbb{A}$ has a domain $\mathrm{dom}(A)$. A dataset instance D consists of a set of tuples of the form $t = (a_1, \ldots, a_n)$ where $a_i \in \mathrm{dom}(A_i)$. Let $\mathbb{A}_{\text{num}} \subseteq \mathbb{A}$ denote the set of numerical attributes to be discretized. Namely, each attribute $A \in \mathbb{A}_{\text{num}}$ must be transformed into a categorical attribute via a valid *partition*.

The system allows flexibility in that a user may choose to exclude certain numerical attributes from partitioning if discretization is deemed unnecessary or undesirable for their analysis. However, for simplicity of presentation, we assume that all numerical attributes $\mathbb{A}_{\text{num}}$ are to be partitioned.

We define a **partition** B for a numerical attribute A as an ordered list of cut points $B = (b_1, \ldots, b_{m-1})$, where each $b_i \in \mathrm{dom}(A)$ and $b_1 < b_2 < \cdots < b_{m-1}$. These cut points induce m contiguous bins[2]:

$$(-\infty, b_1),\ [b_1, b_2),\ \ldots,\ [b_{m-2}, b_{m-1}),\ [b_{m-1}, \infty).$$

[2]For simplicity, we adopt left-closed, right-open bins $[b_i, b_{i+1})$. An equivalent right-closed convention $(b_i, b_{i+1}]$ could be used without affecting our algorithms or metrics.

Applying a partition B to the attribute A yields the discretized version $B(A)$. We denote by $\mathbf{B} = \{B_i \mid A_i \in \mathbb{A}_{\text{num}}\}$ a set of partitions applied across numerical attributes $\mathbb{A}_{\text{num}}$.

Example 5. Consider the Age attribute from Example 3, where

$$\text{dom}(A_{\text{Age}}) = \{27, 33, 50, \ldots, 50, 35, 21, 33, 46, 27\}$$

Let us examine two candidate partitions: $B_1 = (20, 40, 60)$ defines four bins $(-\infty, 20)$, $[20, 40)$, $[40, 60)$, $[60, \infty)$; $B_2 = (19,\ 35,\ 45,\ 60)$ defines five bins $(-\infty, 19)$, $[19, 35)$, $[35, 45)$, $[45, 60)$, $[60, \infty)$. □

In this work, we adopt the commonly used 11 discretizers [16], ranging from the widely used (e.g., Equal Width, ChiMerge [23]) to more sophisticated (e.g., distance-based [11]). We demonstrate that our framework is *agnostic* to discretization methods, allowing any (subsets) of them to be considered.

3.2 Problem Statement

We formalize the Interpretable Attribute Discretization (IAD) problem with two quality measures: a task-specific *utility* over discretized data (e.g., accuracy for modeling, correlation for data visualization) and a *semantic* score that captures interpretability. The goal is to balance these two objectives by finding a set of partitions that maximizes a weighted combination of both. We treat both functions abstractly here and instantiate them per task in Section 5.

The user provides (1) a dataset D with input attributes $\mathbb{A}_{\text{num}}$ and (2) a downstream task M_{util} together with a utility function $M_{\text{util}}(\mathbf{B}, D)$ that evaluates performance on a discretized view of D. When an outcome/label attribute is present (e.g., for prediction), M_{util} incorporates it internally; in settings without an outcome, M_{util} is defined without the outcome attribute.

Note that the semantic score $M_{\text{sem}}(\mathbf{B})$ is a *pluggable* component that can be defined by the user, to quantify the interpretability of a candidate partition set $\mathbf{B}$. Higher values indicate more human-meaningful bins (e.g., alignment with domain conventions), independent of task utility. In our implementation (Sec. 4), we assume an LLM-based semantic measure (Sec. 5) and do not require the user to specify a semantic measure. We now state the problem.

Definition 1. [Interpretable Attribute Discretization] Given a dataset D over schema $\mathbb{A}$, a set of numerical attributes $\mathbb{A}_{\text{num}} \subseteq \mathbb{A}$, and a downstream task M_{util}, find valid partitions $\mathbf{B} = \{B_A \mid A \in \mathbb{A}_{\text{num}}\}$ that maximize:

$$\mathbf{B}^* = \arg\max_{\mathbf{B}} \ \mathcal{J}(\mathbf{B}, D) \tag{1}$$

with trade-off parameter $\alpha \in [0, 1]$ and objective

$$\mathcal{J}(\mathbf{B}, D) \ = \ \alpha \cdot M_{\text{sem}}(\mathbf{B}) \ + \ (1-\alpha) \cdot M_{\text{util}}\big(\mathbf{B}, D\big), \tag{2}$$

where M_{util} evaluates the utility of $\mathbf{B}$ after discretizing the input dataset D using partition set $\mathbf{B}$. □

The scalar α determines the relative importance of semantic interpretability and downstream task utility in the objective. Rather than fix a single α, we seek to *estimate the entire frontier* [34] of non-dominated solutions—partition sets where no other configuration achieves both a higher semantic score and a higher task utility (illustrated Figure 2). This provides a faithful picture of the utility–semantics trade-off [3].

Definition 2. [Interpretable Attribute Discretization Optimal Partition Sets (OPS)] Let $\mathcal{B}$ denote the set of all partition sets (one partition per numerical attribute). Each $\mathbf{B} \in \mathcal{B}$ induces a point

$$p(\mathbf{B}) \ = \ \big(M_{\text{sem}}(\mathbf{B}),\ M_{\text{util}}(\mathbf{B}, D)\big) \in [0, 1]^2, \tag{3}$$

where the first coordinate is the semantic interpretability score and the second is the task utility. For points $p, q \in [0, 1]^2$, we say p *dominates* q (write $p \succ q$) if p is at least as large in both coordinates and strictly larger in one.

The *Pareto frontier* [34] is the set of non-dominated points:

$$\mathcal{P} \ = \ \big\{\, p(\mathbf{B}) \ \big|\ \mathbf{B} \in \mathcal{B},\ \nexists\, \mathbf{B}' \in \mathcal{B} \text{ s.t. } p(\mathbf{B}') \succ p(\mathbf{B}) \,\big\}. \tag{4}$$

Optimal partition sets. We call $\mathbf{B} \in \mathcal{B}$ *optimal for IAD* iff its image lies on the frontier, i.e., $p(\mathbf{B}) \in \mathcal{P}$. Equivalently, the set of optimal partition sets is

$$\mathcal{B}^* \ = \ \big\{\, \mathbf{B} \in \mathcal{B} \ \big|\ \nexists\, \mathbf{B}' \in \mathcal{B} \text{ s.t. } p(\mathbf{B}') \succ p(\mathbf{B}) \,\big\}$$

Moreover, for any scalarized objective $\mathcal{J}(\mathbf{B}, D)$ with $\alpha \in [0, 1]$, every maximizer $\mathbf{B}^*$ lies in $\mathcal{B}^*$. □

Example 6. We revisit Example 5. Assume $\mathbb{A}_{\text{num}} = \{\texttt{Age}\}$ so we have $\mathbf{B_1} = \{(20, 40, 60)\}$ and $\mathbf{B_2} = \{(19,\ 35,\ 45,\ 60)\}$. Suppose the downstream task M_{util} is to train an interpretable model, say a decision tree classifier, predicting diabetes outcome. Let the measured utility scores (model accuracy) for these partitions be: $M_{\text{util}}(D_{\mathbf{B_1}}) = 0.72$ and $M_{\text{util}}(D_{\mathbf{B_2}}) = 0.70$. Meanwhile, let the semantic interpretability scores be: $M_{\text{sem}}(\mathbf{B_1}) = 3.0$ and $M_{\text{sem}}(\mathbf{B_2}) = 4.0$.

Given a trade-off parameter $\alpha = 0.5$, the combined objective scores for the two partitions are: $\mathcal{J}(\mathbf{B_1}, D) = 0.5 \cdot 3.0 + 0.5 \cdot 0.72 = 1.86$, while $\mathcal{J}(\mathbf{B_2}, D) = 0.5 \cdot 4.0 + 0.5 \cdot 0.70 = 2.35$. Although $\mathbf{B_2}$ yields slightly lower predictive performance, it achieves significantly higher semantic clarity. As a result, $\mathbf{B_2}$ is included on the Pareto frontier while $\mathbf{B_1}$ is discarded. □

The main challenge in solving the optimal partition sets for IAD problem lies in navigating the vast combinatorial search space of partition sets. If each $A \in \mathbb{A}_{\text{num}}$ admits up to k cuts from v candidates, the per-attribute count is $\sum_{i=1}^{k} \binom{v}{i}$ and the joint space over d attributes is $\left(\sum_{i=1}^{k} \binom{v}{i}\right)^d$, exponential in d. We address this by combining structure-aware clustering of partitions with MDP-based search (Section 4).

4 Quill

In this section, we present our solution to the Interpretable Attribute Discretization Optimal Partition Sets (OPS) problem (Definition 2).

4.1 Overview

Our framework architecture is illustrated in Figure 3, which consists of three main components: (1) a candidate generation step, (2) a partition clustering step to group similar partitions, and (3) a search step to estimate optimal partition sets.

Candidate generation. We begin by curating a diverse set of candidate discretization strategies for each numerical attribute.

[3] When desired, Quill can also solve the simpler single-point variant by optimizing (2) for a fixed α.

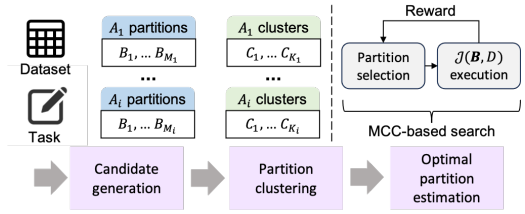


Figure 3: Quill overview.

Our curated pool includes 11 widely-used statistical discretization techniques (e.g., Equal Frequency, ChiMerge [23]) as well as interpretable discretization suggestions generated using GPT-4o [38], which reflects domain practices (e.g., age brackets in healthcare). This hybrid approach ensures that both utility-optimized and interpretable partitions are included in the search space, the details of which can be found in Appendix A.

Partition clustering. For each attribute we enumerate candidate partitions, yielding a combinatorially large space of partition sets. To shrink this space, we (i) represent each partition as a distribution vector over the attribute's domain, and (ii) cluster partitions by similarity in that representation. This groups near-equivalent candidates and lets the search operate over clusters rather than raw partitions. The intuition and details explained in Sec. 4.2.

Optimal partition estimation. We holistically consider partition candidates across all attributes, defined in the Optimal Partition Sets (OPS) problem (Def. 2). Specifically, after clustering similar partitions for each attribute, we cast our OPS problem as a Markov Decision Process over clusters. The high-level intuition is that evaluating a partition from one cluster provides information about the others in the same cluster (details in Sec. 4.3). We adopt the Monte Carlo Control (MCC) algorithm to balance exploration across clusters and exploitation of promising ones (Sec. 4.4).

Our framework is general and modular. Candidate generation, partition clustering, semantic scoring, and utility evaluation are independent components that can be swapped or extended, while still utilizing the same clustering-and-search backbone, which is a salient feature of Quill. We note that these components introduce hyperparameters; to improve robustness, Quill includes an automatic hyperparameter tuner (Sec. 4.6).

4.2 Partition Representations

In this section, we will introduce how we represent the entire search space, including candidate partition sets across all attributes.

Partition distribution vectors. We will start by each candidate partition B as a distribution vector that encodes the distribution of data values of an attribute A across its bins, denoted by $\mathbf{v}(B)$.

Definition 3. [Partition distribution vector] Let A be a numerical attribute with observed values $\mathrm{dom}(A) = \{x_1, \dots, x_N\}$, and let $B = \{b_1, \dots, b_{m-1}\}$ be a partition of A that induces m contiguous bins: $(-\infty, b_1)$, $[b_1, b_2)$, $\dots$, $[b_{m-2}, b_{m-1})$, $[b_{m-1}, \infty)$. We define $n_j(B)$ as the number of values in $\mathrm{dom}(A)$ that fall into the jth bin:

$$n_j(B) = |\{x \in \mathrm{dom}(A) \mid x \in \text{bin } j\}|, \quad j = 1, \dots, m.$$

The *distribution vector* of B is then:

$$\mathbf{v}(B) = \left(\frac{n_1(B)}{N}, \dots, \frac{n_m(B)}{N}\right), \quad \text{where } N = |\mathrm{dom}(A)|.$$

This vector $\mathbf{v}(B)$ represents the empirical distribution of values across bins, enabling comparisons between partitions B using distance metrics over vectors. □

Example 7. Let A = Age with $|\mathrm{dom}(A)| = 114$ in this example for illustration. Consider three candidate partitions:

$$B_1: \{[19, 35),\ [35, 45),\ [45, 60]\}, \quad (n_1, n_2, n_3) = (33, 57, 24),$$
$$B_2: \{[19, 30),\ [30, 45),\ [45, 60]\}, \quad (n_1, n_2, n_3) = (25, 65, 24),$$
$$B_3: \{[19, 60]\}, \quad (n_1) = (114).$$

By Definition 3, the distribution vectors $\mathbf{v}(B_1), \mathbf{v}(B_2), \mathbf{v}(B_3)$ are

$$\mathbf{v}(B_1) = \left(\tfrac{33}{114},\ \tfrac{57}{114},\ \tfrac{24}{114}\right) \approx (0.289,\ 0.500,\ 0.211),$$
$$\mathbf{v}(B_2) = \left(\tfrac{25}{114},\ \tfrac{65}{114},\ \tfrac{24}{114}\right) \approx (0.219,\ 0.570,\ 0.211),\ \mathbf{v}(B_3) = (1.0).$$

□

Partition clusters. We group partitions with similar distribution vectors, using DBSCAN [46] with Earth Mover's Distance (EMD) [44], to reduce the search space.

Definition 4. [Partition clusters] For attribute A_i, let $\mathbf{B}_i$ be its candidate partitions, and $\mathbf{V}_i = \{\mathbf{v}(B) \mid B \in \mathbf{B}_i\}$ the corresponding distribution vectors of the partitions $\mathbf{B}_i$ (Def. 3). Let $\mathrm{EMD}(\mathbf{v}, \mathbf{v}')$ denote the Earth Mover's Distance between two vectors.

We cluster $\mathbf{V}_i$ using DBSCAN with parameters ($radius$, minPts), where: (i) a vector $\mathbf{v}(\mathbf{B})$ is a *core point* if it has at least minPts neighbors within $radius$; (ii) a *cluster* C is any maximal group of core points and their density-connected neighbors; (iii) vectors not assigned to any cluster are treated as singleton clusters. We denote the resulting clusters by $\mathcal{C}_i = \{C_1, \dots, C_{K_i}\} \subseteq 2^{\mathbf{V}_i}$. □

The clustering is task-agnostic: it depends only on $\mathbf{v}(B)$ and the distance measure, not on downstream utility. We use DBSCAN [46] with EMD [44] because it: (i) makes no parametric assumptions on cluster shape, (ii) works directly with a precomputed, non-Euclidean distance (EMD), (iii) automatically detects outliers as *noise* so they can be identified as their individual clusters (singletons), instead of forcing them into a cluster with other partitions, (iv) does not require the number of clusters K to be specified, and (v) has a small, interpretable parameter set ($radius$, minPts) that we autotune (Sec. 4.6). Ablations in Sec. 6 validate this choice of clustering method and distance metric.

Example 8. We continue with Example 7. Using EMD on the distribution vectors $\mathbf{v}(B)$, we obtain:

$$\mathrm{EMD}(\mathbf{v}(B_1), \mathbf{v}(B_2)) \approx \mathbf{0.026} \quad \text{and} \quad \mathrm{EMD}(\mathbf{v}(B_1), \mathbf{v}(B_3)) \approx \mathbf{0.121},$$

with the pairs involving B_3 substantially larger than (B_1, B_2). With DBSCAN (radius between 0.03 and 0.10, minPts = 2), B_1 and B_2 form a cluster, while B_3 is identified as noise (or a singleton), illustrating how clustering groups semantically/statistically similar partitions and isolates outliers. □

We posit that if two partitions B, B' of an attribute A have similar empirical distributions, formalized as a small EMD between their distribution vectors $\mathbf{v}(B)$ and $\mathbf{v}(B')$, then their objective values are close, i.e., $|\mathcal{J}(B, D) - \mathcal{J}(B', D)|$ is small. This is well motivated for the *utility* term: many task scores (e.g., impurity- or likelihood-based criteria, correlations, simple-model accuracies on discretized inputs) depend smoothly on bin frequencies, so a small distance

induces small utility changes. For the *semantic* term, the relationship can be less smooth (e.g., domains with conventionally "hard" boundaries like BMI), and small boundary shifts may cause larger semantic deltas; we acknowledge this limitation.

In practice, we adopt this assumption to make search tractable and interactive, and we find it holds well enough to guide exploration: clustering by EMD lets us evaluate a few representatives and generalize to nearby partitions, dramatically reducing evaluations while still locating high-utility, high-semantic solutions. Empirically (Section 6), this design yields strong quality and consistent gains across datasets and metrics.

4.3 OPS as a Markov Decision Process Problem

Using the partition vectors and partition clusters, we now describe how the OPS problem in Definition 2 can be solved using a reinforcement learning algorithm we design in this work.

To solve OPS, we must select one partition per input attribute from a combinatorially large space to maximize the overall objective $\mathcal{J}(\mathbf{B}, D)$. This is inherently a multi-step process: the objective can only be evaluated once all attribute–partition choices are made, and the goal is not a single optimum but the entire Pareto frontier capturing the utility–semantics trade-off.

To make this search tractable, we first cluster partitions using their distribution vectors (Section 4.2), allowing decisions to operate at the cluster level rather than over individual partitions. Evaluating one representative within a cluster provides informative feedback about others, enabling efficient generalization and exploration. These properties make OPS naturally suited to a reinforcement learning (RL) formulation, where a policy iteratively selects one cluster per attribute and is rewarded based on the joint quality of the resulting partition set.

We are now ready to cast OPS as a finite-horizon Markov Decision Process (MDP), which cleanly exposes the choices, constraints, and objectives of our search.

Definition 5. [Optimal Partition Sets as a Markov Decision Process (OPS-MDP)] Let the numerical attributes be $\mathbb{A}_{\text{num}} = \{A_1, \ldots, A_d\}$, and let each attribute A_i have a set of candidate partitions $\mathbf{B}_i = \{B_1, \ldots, B_{j_i}, \ldots, B_{M_i}\}$, where B_{j_i} is a candidate partition for A_i. From these candidates, we form clusters of similar partitions $C_i = \{C_1, \ldots, C_{k_i}, \ldots C_{K_i}\}$, where each cluster $C_{k_i} \subseteq \mathbf{B}_i$ groups partitions of A_i that are semantically and statistically similar (as defined in Section 4.2). The union of all such clusters across attributes $\mathbb{A}_{\text{num}}$ constitutes the reduced search space for OPS.

OPS-MDP is defined as follows:

- **States:** A state s_t represents a partial assignment of partitions to attributes, i.e., a set $\mathbf{B}_t = \{B_{j_i} \mid i \in \{1, \ldots, d\}\}$ where $d = |\mathbb{A}_{\text{num}}|$.
- **Actions:** An action $a = (A_i, C_{k_i})$ selects one of the clusters $C_{k_i} \in C_i$ for an *undiscretized* attribute A_i. A concrete partition (representative) $B_{j_i} \sim C_{k_i}$ is then sampled from this cluster and applied to discretize A_i.
- **Transitions:** Deterministic, updating the state by adding A_i and its chosen partition B_{j_i} to the current partial assignment.
- **Episodes:** Run for $d = |\mathbb{A}_{\text{num}}|$ steps, terminating when all d attributes have been assigned partitions, resulting in a complete partition set $\mathbf{B}_d = \{(A_1, B_{j_1}), \ldots, (A_d, B_{j_d})\}$, where each $B_{j_i} \in \mathbf{B}_i$ is the chosen partition for attribute A_i.
- **Reward:** The agent receives reward $r = 0$ for $t < d$, and $r = \mathcal{J}(\mathbf{B}_d, D)$ once all attributes have been discretized.
- **Policy:** A mapping $\pi(a \mid s)$ from partial states s to distributions over feasible actions a, where each action $a = (A_i, C_{k_i})$ selects a cluster C_{k_i} for an undiscretized attribute A_i.

This formulation allows us to learn a policy that constructs a high-quality partition set by leveraging generalization within clusters and delayed reward feedback from the final objective. □

Valid solutions to OPS-MDP. We first show how a valid solution to OPS would look like in the Markov Decision Process setup, before describing our RL solution.

We discretize a set of input attributes $\mathbb{A}_{\text{num}}$. Since we are applying exactly one partition to each attribute A_i, then out of all clusters C_i for A_i, only exactly one partition should be chosen (as the representative) $B_{j_i} \sim C_{k_i}$ from exactly one cluster $C_{k_i} \in C_i$. The union of such partitions across all attributes $A_i \in \mathbb{A}_{\text{num}}$ is then the overall selected partition set $\mathbf{B}_d = \{(A_1, B_{j_1}), \ldots, (A_d, B_{j_d})\}$.

Definition 6. [Valid solutions to OPS-MDP] Given the set of attributes $\mathbb{A}_{\text{num}} = \{A_1, \ldots, A_d\}$, each attribute A_i has an associated set of partition clusters $C_i = \{C_1, \ldots, C_{K_i}\}$, where each cluster C_{k_i} contains candidate partitions $B_{j_i} \in \mathbf{B}_i$. A valid OPS solution must select exactly one partition for each attribute. Each valid $\mathbf{B}_d$ thus represents a complete discretization—one partition per attribute—corresponding to an episode in OPS-MDP. □

Given that there exist many valid solutions (exponential in the number of attributes), we resort to the objective function of OPS to evaluate the relative merits of these solutions as $\mathcal{J}(\mathbf{B_d}, D)$.

4.4 Solving OPS-MDP via Monte Carlo Control

We now describe our RL algorithm for OPS-MDP, that can solve OPS-MDP efficiently, with conditional provable quality guarantees.

Monte Carlo Control (MCC) is a classic reinforcement learning algorithm for learning optimal policies in episodic, model-free settings [52]. It is particularly well-suited to our OPS-MDP for three reasons: (i) OPS-MDP requires a delayed reward available only after all attribute–partition selections are complete, which matches MCC's episodic, return-based updates. (ii) The problem has a finite horizon and discrete action space—one decision per attribute—making sample-based value estimation both stable and efficient. (iii) OPS-MDP has deterministic transitions and requires no transition model, aligning naturally with MCC's model-free formulation. These properties make MCC an ideal and principled choice for optimizing policies in OPS-MDP while maintaining computational tractability.

Q-table setup for OPS-MDP. We model the OPS environment as a finite-horizon Markov Decision Process (MDP) in which each state represents a *partially filled partition set* $\mathbf{B}_t$, and each action corresponds to selecting a *partition cluster* C_{k_i} for attribute A_i.

Let K_i denote the number of clusters available for attribute A_i, and define the total number of clusters across all attributes as $\sum_{i=1}^{d} K_i$. We construct a Q-table with $n = \sum_{i=1}^{d} K_i + 2$ total states and actions, where: one dummy *start state* s_0 marks the beginning of each episode; $\sum_{i=1}^{d} K_i$ *intermediate states* correspond to selecting each of the $\sum_{i=1}^{d} K_i$ attribute–cluster pairs; one *terminal state* s_n signifies completion of the partitioning process. For example, with

Algorithm 1: Solve OPS-MDP via Monte Carlo Control

Input: Attributes $\{A_1, \ldots, A_d\}$, clusters $\{C_1, \ldots, C_d\}$, number of episodes T
Output: Estimated Pareto-optimal set of partitions $\mathcal{B}^*$

1 Initialize Q-table Q and returns dictionary $\texttt{returns}[(s,a)] \leftarrow [\,]$;
2 Initialize policy π_θ **for** $n = 1$ *to* T **do**
3 Initialize state s_0, empty partition set $\mathbf{B_t}$, and episode history $H \leftarrow [\,]$;
4 **while** *not all attributes covered* **do**
5 Select valid cluster C_{k_i} from state s using π_θ;
6 Sample $B_{j_i} \sim C_{k_i}$, update $\mathbf{B_t}$ and transition to next state s';
7 Append (s, a) to H;
8 Set $s \leftarrow s'$;
9 Compute final reward $r = \mathcal{J}(\mathbf{B_d})$;
10 **foreach** $(s_t, a_t) \in H$ **do**
11 Append r to $\texttt{returns}[(s_t, a_t)]$;
12 Update $Q(s_t, a_t) \leftarrow \frac{1}{|\texttt{returns}[(s_t,a_t)]|} \sum r' \in \texttt{returns}[(s_t, a_t)] r'$;
13 Update policy π_θ using updated Q;
14 Compute the Pareto front $\hat{\mathcal{B}} = \texttt{compute\_pareto}(\{\mathbf{B}_1, \ldots, \mathbf{B}_T\})$;
15 **return** $\hat{\mathcal{B}}$;

3 attributes—`Age`, `BMI`, and `Glucose`—each having 3 clusters, we have $\sum_{i=1}^{d} K_i = 3+3+3 = 9$ and $n = 11$, yielding an 11×11 Q-table.

Thus, the Q-table is an $n \times n$ matrix, where each entry $Q(s, a)$ estimates the expected return of taking action a in state s. In OPS-MDP, where the reward is only observed at the end of each episode, these values are updated retrospectively using the final reward, which is shared by all state–action pairs visited within that trajectory.

Valid actions and Q-table masking. To enforce the one-partition-per-attribute constraint (Def. 6), we apply a masking strategy that prunes invalid transitions from the Q-table. Specifically, we initialize invalid actions with a value of $-\infty$, preventing the agent from: (i) selecting more than one cluster for the same attribute (ii) transitioning back to the start state s_0, (iii) or taking actions from the terminal state s_n, after a complete partition set has been formed.

This *Q-table masking*, detailed in Appendix B, is a critical design choice in our MCC formulation. It ensures that all episodes represent valid solutions, reduces the effective action space, and accelerates convergence by guiding exploration. Notably, the masking procedure is *generalizes* to any number of attributes and clusters, preserving the modularity and correctness of Quill framework.

Policy for OPS-MDP. At each step in an episode, the agent selects a valid cluster C_{k_i}, samples a partition $B_{j_i} \sim C_{k_i}$, adds B_{j_i} to the current partition set $\mathbf{B}_t$, and computes the reward as the change in $\mathcal{J}$. Let $Q(s, a)$ denote the state-action value for state s and action a, and let $\mathcal{A}_s$ denote the set of valid actions in state s, after masking out. Then the ϵ-greedy policy is defined as:

$$\pi_\epsilon(s) = \begin{cases} \arg\max_{a \in \mathcal{A}_s} Q(s, a), & \text{with probability } 1 - \epsilon \\ \text{Uniform}(\mathcal{A}_s), & \text{with probability } \epsilon \end{cases} \tag{5}$$

This formulation ensures that the agent selects among only valid actions at each state, while balancing exploration and exploitation.

Solving OPS-MDP. With all components described above, we are now ready to introduce our Algorithm 1, which builds on top of Monte Carlo Control (MCC) [52] to solve OPS-MDP.

Algorithm 1 begins by initializing the Q-table Q and the returns dictionary for tracking observed returns, along with the policy π_θ using an ϵ-greedy exploration schedule (lines 1–2). For each of the T episodes, the agent starts from an initial state s_0 with an empty partition set $\mathbf{B}_t$ and initializes an episode history H to store encountered (state, action) pairs (line 3). The agent then iteratively selects a valid cluster C_{k_i} from the current state using the policy π_θ, samples a partition B_{j_i} from C_{k_i}, adds it to $\mathbf{B}_t$, transitions to the next state, and records the (s, a) pair in the episode history (lines 4–8). Once all attributes are covered and the episode ends, the final reward r is computed as the overall objective score $\mathcal{J}(\mathbf{B_d}, D)$ (line 9). This final reward is then propagated to all (state, action) pairs in the episode history: for each pair, the reward is appended to the list of observed returns, and the corresponding Q-value is updated as the mean of these returns (lines 10–12). The policy π_θ is then updated using the newly estimated Q-values (line 13). After all episodes are completed, the algorithm computes the final Pareto-optimal set $\hat{\mathcal{B}}$ over all evaluated partition sets and returns the final estimation (lines 14–15).

Our framework offers several compelling advantages. First, it enables coordinated decision-making across attributes, learning joint discretization strategies rather than treating attributes independently. This coordination allows the framework to naturally capture cross-attribute interactions, which are critical in many real-world datasets where utility quality depend on attribute combinations (e.g., causal inference, modeling). Second, it is metric-agnostic: it requires no assumptions about the form of the objective $\mathcal{J}$, and can flexibly accommodate any combination of utility and semantic measures. These properties make it a flexible, general-purpose approach for OPS. We additionally include an automatic hyperparameter tuner (Sec. 4.6). Empirical results are presented in Section 6.

4.5 Special Case: Single-Attribute OPS-MDP

Attribute discretization spans many use cases (Sec. 5). Some require joint binning across attributes (e.g., causal analysis), while others focus on a single attribute (e.g., visualization, exploratory analysis). The OPS-MDP algorithm covers both; here we zoom in on the d=1 case, where we can establish formal guarantees.

With d=1, the MCC formulation collapses to a single decision: choose one partition for the attribute. Each episode has exactly one action, after which we observe the final reward via the joint objective $\mathcal{J}$. This is precisely a *multi-armed bandit*: each cluster is an arm, and sampling a partition corresponds to pulling that arm.

Since *multi-armed bandits are degenerate MDPs* with a single state [52], classical algorithms such as Upper Confidence Bound (UCB) solve this setting and provide regret guarantees. When d=1, MCC reduces to UCB and *inherits UCB's regret bounds*, giving theoretical grounding for our framework, only in this case.

Theorem 1. If (i) d=1 (single attribute, one action per episode) and (ii) MCC uses the UCB index for action selection and updates, then our Algorithm 1 becomes a UCB-guided MCC variant and reduces exactly to a stochastic multi-armed bandit with K arms (one per cluster) and reward in $[0, 1]$ given by $\mathcal{J}$. Running Algorithm 1 for T episodes yields the regret guarantees, inherited from UCB [10]—gap-free regret $O(\sqrt{KT \log T})$ and gap-dependent regret

$O(\sum_i \frac{\log T}{\Delta_i})$, where Δ_i is how far cluster C_i's expected reward is below the optimal expected reward $\mathcal{J} = 1$.

Proof by induction on the number of episodes t. We show that for all episodes $t \geq 1$, the behavior of Algorithm 1 under $d = 1$ is equivalent to that of UCB. Specifically, their selected arms, maintained statistics, and update rules are the same across both algorithms.

Base case ($t = 1$): At the beginning, all cluster arms are untried. UCB initializes by pulling each arm once to get an initial reward estimate. In MCC, when $d = 1$, there is only one decision per episode: selecting one cluster, sampling a partition from it, and observing the reward $\mathcal{J}(B, D)$. We simplify the partition set notation from $\mathbf{B}$ to B here, since there is only one partition B in the set when $d = 1$.

Suppose there are K clusters (arms) $\{C_1, \ldots, C_K\}$. Then in the first k episodes, MCC samples each cluster once. These samples populate the initial reward statistics, just like UCB. So after $t = k$, both UCB and MCC have exactly one reward value for each arm, and their empirical means match:

$$\hat{r_i} = \mathcal{J}(B_i, D) \quad \text{for each arm } i$$

Inductive step: Assume that after $(t-1)$ episodes, the following invariants hold: (1) Each cluster C_i has been sampled n_i times in both MCC and UCB. (2) The empirical mean reward $\hat{r_i}$ for each cluster is the same in both algorithms. (3) The cluster selected in episode $t-1$ by MCC is the same as that selected by UCB.

Now consider episode t.

- **Action selection:** Since both use the UCB index for action selection and updates.

$$a_t = \arg\max_i \left(\hat{r_i} + \sqrt{\frac{2 \log t}{n_i}} \right)$$

  Thus, both algorithms choose the same cluster arm in episode t.
- **Reward observation and update:** MCC samples a partition $B \sim C_{a_t}$, observes its reward $\mathcal{J}(B, D)$, and updates the empirical mean via the incremental update rule:

$$\hat{r}_{a_t} \leftarrow \hat{r}_{a_t} + \frac{\mathcal{J}(B, D) - \hat{r}_{a_t}}{n_{a_t} + 1}$$

  This is equivalent to recomputing the average over all samples for a_t, matching UCB's update step.
- **Statistics consistency:** After the update, both algorithms have: $n_{a_t} \leftarrow n_{a_t} + 1$, and updated $\hat{r}_{a_t}$ with the same formula. Hence, all maintained statistics remain identical between the two.

Conclusion: By induction, we have shown that for all t, the arm selected and the statistics maintained by MCC match exactly those of UCB. Thus, MCC reduces to UCB when $d = 1$. □

This UCB-guided MCC variant is effective in the single-attribute ($d = 1$) setting because choices are independent and the reward is one-step. for $d > 1$, the reward $\mathcal{J}(\mathbf{B}, D)$ is observed only after selecting a *full* partition set, so actions are interdependent: the reward of a partition for A_i depends on the partitions chosen for A_j ($j \neq i$). Thus, UCB's core assumptions (i.e., independent arms and immediate observable rewards) break, and its regret guarantees do not apply in the multi-attribute setting.

Moreover, naïve UCB extensions are suboptimal. *Independent UCB* runs UCB per attribute, implicitly assuming partition choices can be made separately and ignoring cross-attribute interactions. *Sequential UCB* imposes an attribute order and applies UCB step by step, but early choices cannot be revised, inviting cascading errors. We evaluate both in Sec. 6.

4.6 Tuning Hyperparameters

Our end-to-end framework includes three key hyperparameters: (*radius*, minPts) required to form a dense region in DBSCAN [46], and the exploration rate ϵ in the ϵ-greedy policy in MCC. Exhaustive grid search over these parameters would be computationally expensive. Instead, we employ lightweight, data-driven tuning strategies that adapt hyperparameter values to the specific input dataset D.

The silhouette score [43] measures how well a data point fits into its assigned cluster compared to other clusters. Higher silhouette scores indicate better cluster separation and cohesion. Our auto-tuner selects DBSCAN's *radius* and minPts from data, using Wasserstein distances between candidate distributions. For every (*radius*, minPts) pair, it runs DBSCAN with the precomputed distance matrix and scores the result by silhouette. Our framework uses the pair with the highest score. This way, clustering quality is ensured without using the objective score $\mathcal{J}$, which requires running the framework end-to-end.

To tune the exploration rate in MCC, we use an exponential decay schedule rather than fixed ϵ values, which shrinks the tuning space and adapts exploration during training. The exploration rate at episode n is defined as:

$$\epsilon_n = \max\left(\epsilon_{\min}, \epsilon_0 \cdot e^{-kn}\right), \quad \text{where} \quad k = \frac{\log(\epsilon_0/\epsilon_{\min})}{T},$$

and T is the total number of training episodes. This schedule enables a smooth exploration-to-exploitation transition and requires only two parameters (ϵ_0, $\epsilon_{\min}$), while still covering a broad spectrum of ϵ behaviors in a single run. In our experiments we set $\epsilon_0 = 0.7$, $\epsilon_{\min} = 0.5$ as default. This choice performs strongly across the majority of our 7 datasets and 14 (utility × semantic) scenarios. We include an ablation study against the set of hand-tuned hyperparameters (fit on 2 datasets), to show the effectiveness of the auto-tuner (Section 6.2.3).

5 Instantiations of Utility and Interpretability

To demonstrate our framework's general, modular design, we instantiate it with representative semantic measures and utilities. These cases serve as concrete testbeds to showcase effectiveness and efficiency, and to empirically verify that the framework is agnostic to both task choice and semantic metric (Sec. 6).

5.1 Measuring Interpretability

Our first objective is to quantify the interpretability of a given partition with respect to a numerical attribute. As no off-the-shelf metric directly addresses this, we propose a few intuitive measures tailored to this objective.

LLM-based semantic measure. We use corpus attestations as a proxy for semantic meaningfulness: partitions that appear frequently in literature or code likely reflect domain expertise and practical utility. Direct counting references is not trivial (e.g., attributes with different synonym names, partitions only in source code). We therefore query LLMs to aggregate this knowledge, due

to their comprehensive knowledge base and ability to capture nuances. We investigated two models: GPT-4o [38] for broad domain knowledge, and Perplexity AI [8] for source-grounded responses.

We use a 1-4 scale for the semantic value of a given partition for an attribute, where 1 represents poor semantic value (not used at all) and 4 represents excellent semantic value (very commonly used). The specific prompts used are given in [9].

To validate our approach, we ran a user study with 54 Prolific participants [7]. Each rated 26 items—an attribute description paired with a candidate partition—on a 1–4 semantic scale (matching our LLM prompts; form in repo). Inter-rater agreement (Krippendorff's α [24]) was 0.20, reflecting task difficulty yet indicating modest consensus. Human ratings correlated strongly with LLM scores: Spearman $\rho = 0.90$ for GPT-4o and $\rho = 0.83$ for Perplexity (both $p < .05$). We therefore use GPT-4o for subsequent experiments.

Distance-based semantic measures. When interpretable partitions are known (e.g., binning `Age` into $(19, 35, 45, 65)$ with labels ("Young adult", "Adult", "Middle-aged") in life sciences [31]), users can supply a *gold* partition B_{gold}. Our semantic module then scores any candidate B by its distance to the gold. In this work, we consider the following distance-based semantic measures.

L2 norm. L2 norm is a symmetric and intuitive metric, ideal for situations where partitions differ structurally (e.g., different numbers of cut points), and where bin-wise deviation should be penalized uniformly. Given a candidate B (defined with cut points in Sec. 3.1) and the gold, we pad the shorter partition with zeros to equalize their lengths and compute the L2 norm.

Kullback–Leibler (KL) divergence [12]. Viewing $\mathbf{v}(B)$ and $\mathbf{v}(B_{\text{gold}})$ as discrete probability distributions, $\text{KL}\Big(\mathbf{v}(B) \,\|\, \mathbf{v}(B_{\text{gold}})\Big)$ measures how well B_{gold} is approximated by B; it is 0 iff the distributions match. KL divergence measures the information loss (coding inefficiency) when the candidate partition is used in place of the gold. It penalizes moving probability mass in ways that contradict the gold, thereby aligning with domain priorities.

When to use which measure. If domain guidelines exist (e.g., BMI or age cutoffs), distance-based scores (Euclidean or KL) give transparent alignment to a reference. When such priors are unavailable or hard to formalize, LLM-based measures offer flexible, knowledge-grounded semantics. Our framework is agnostic to the choice and performs consistently across these measures (Sec. 6).

5.2 Measuring Usefulness

Next, we instantiate utilities for common tasks—visualization, interpretable modeling, and causal analysis. Table 1 summarizes the measures used in our experiments.

Data visualization. Data visualizations (e.g., bar plots) help users spot patterns that raw tables can obscure. Visualization recommendation systems [49, 54, 56] often score *interestingness* via simple statistics (e.g., correlations). In this work, we quantify interestingness (utility) using the Spearman correlation coefficient [18]. We choose a partition of a *single* numerical attribute to maximize the correlation between the partitioned A_i and the outcome attribute.

Interpretable modeling. Many applied domains (e.g., economics, social science, agriculture) emphasize transparency and favor inherently interpretable models over post-hoc explanations [45]. Discretizing numeric attributes is especially useful for inherently interpretable models—decision trees, rule-based classifiers, and scorecards—yielding simpler splits, reduced noise sensitivity, and closer alignment with domain knowledge. In this use case, we discretize all numerical attributes in $\mathbb{A}_{\text{num}}$ prior to training. We assume a tree-based model, and we measure utility by predictive accuracy of the outcome model.

Data imputation. Data imputation replaces missing values with estimates (e.g., mean/median, most-frequent value, or nearest-neighbor) so that analysis and modeling need not discard incomplete rows. Discretizing numeric attributes *before* imputation simplifies this step: a smaller, structured set of bin labels makes plausible replacements easier and more consistent (e.g., imputing the most frequent bin within a subgroup). For this task, we select a partition for each $A_i \in \mathbb{A}_{\text{num}}$, discretize the data, and apply KNN-imputation [48] on the discretized data. We then train a tree-based model; utility is the model's accuracy on the outcome attribute.

Causal analysis. Causal analysis estimates the effect of a treatment on an outcome while adjusting for confounders. We focus on the Average Treatment Effect (ATE), the expected treated–vs–untreated outcome difference under proper adjustment [39]. Discretizing numeric confounders simplifies conditioning on continuous variables, can improve robustness, and reveals heterogeneous effects across subgroups. Because counterfactuals are unobserved, evaluation requires specialized metrics. We therefore propose two utility measures for ATE estimation, reflecting different levels of available knowledge; details appear in Appendix 8.

Known ATE. When the true ATE is known, utility is the exponential of the negative absolute error between the true ATE (on D) and the ATE computed after discretization (on $\mathbf{B}(D)$): $M_{\text{util}}(\mathbf{B}, D) = \exp^{-|\text{ATE}(D) - \text{ATE}(\mathbf{B}(D))|}$, where $\text{ATE}(D)$ denotes the causal effect evaluated on dataset D. Thus, smaller ATE deviations yield values closer to 1 (better), and larger deviations shrink toward 0. Confounder adjustment is implicit in $\text{ATE}(\cdot)$.

Unknown ATE. When the true ATE is unavailable (typical in observational data), we evaluate how well discretization preserves dependency structure. We compute pairwise Spearman correlations [51] before and after discretization, label each as *high* or *low* using quartile thresholds ($\tau_{\text{low}} = 25^{\text{th}}$, $\tau_{\text{high}} = 75^{\text{th}}$), and score agreement via macro-averaged F1 [30]. This correlation-order preservation serves as a proxy for causal stability; on synthetic data it aligns with the known-ATE utility ($R^2 = 0.7989$, $p < 0.05$).

6 Experimental Evaluation

We embody our framework in a prototype system, Quill, which was written in 3,000 lines of Python source code. Our default implementation supports the 5 utility (from 4 downstream tasks) and 3 semantic measures in Sec. 5. Quill allows users to define any measures of their interest. Our data and code are in [9]. We empirically demonstrate the following claims about our method:

(1) Quill discovers binning partitions that are *semantically meaningful* while achieving *high downstream utility*, tracing the Pareto trade-off explicitly.
(2) Quill treats discretizers, utilities, and semantic metrics as plugins (11+ off-the-shelf discretizers and LLM suggestions; multiple task utilities). Across **14 dataset × utility cases** and **three**

Table 1: Downstream tasks and utility definitions used in our benchmark. $\mathbf{B}(D)$ denotes the discretized (binned) data.

Task	Attribute setting	Description (utility measure)
Visualization (viz)	Single-attribute	Spearman correlation between the partitioned attribute and the outcome on $\mathbf{B}(D)$ (higher is better).
Interpretable modeling	Multi-attribute	Predictive accuracy of a tree-based model trained on $\mathbf{B}(D)$.
Data imputation	Multi-attribute	Discretize, apply KNN imputation on $\mathbf{B}(D)$, then train a tree-based model; utility is predictive accuracy.
Causal analysis (known ATE)	Multi-attribute	Preservation of causal effects measured by exponential ATE deviation decay (closer to 1 is better).
Causal analysis (unknown ATE)	Multi-attribute	Macro-averaged F1 of preserving high/low Spearman correlations before vs. after discretization.

Table 2: Dataset statistics.

Dataset	Description	#Tuples	$\lvert\mathbb{A}\rvert$	$\lvert\mathbb{A}_{\text{num}}\rvert$	Size (MB)	#Candidates
TITANIC [6]	Passenger data	0.9K	10	4	< 1	339,300
CROP [3]	Agriculture variables	2.2K	8	7	< 1	27,625,536
BANK [2]	Customer loan status	100K	19	10	18.8	28,392,912
LALONDE [26]	Causal effect of training	0.4k	12	3	< 1	870
PIMA [4]	Diabetes predictors	0.8K	9	8	< 1	313,820
SONGS [1]	Properties of songs	19K	15	13	2.2	20,170,080
TAXI [5]	Details of taxi trips	8.5M	5	4	265	217,728

semantic measures, Quill consistently returns partitions on or near the Pareto front.

(3) Each component of Quill contributes meaningfully.
(4) Quill is efficient and can scale to large datasets.

6.1 Setup

6.1.1 Evaluation Metrics. We assess (i) accuracy of the estimated Pareto front and (ii) end-to-end efficiency.

Accuracy. Given the true front $\mathcal{P}^*$ (via brute force) and the estimated front $\hat{\mathcal{P}}$, we use a normalized **averaged Hausdorff distance** [47]:

$$\Delta(\hat{\mathcal{P}}, \mathcal{P}^*) = \frac{1}{Z} \max\{\text{GD}(\hat{\mathcal{P}}, \mathcal{P}^*),\ \text{IGD}(\hat{\mathcal{P}}, \mathcal{P}^*)\}, \tag{6}$$

where $Z = \sqrt{2}$ since utility and semantics are in $[0, 1]$. GD (Generational Distance) is the mean distance from each $\hat{p} \in \hat{\mathcal{P}}$ to its nearest neighbor in $\mathcal{P}^*$; IGD (Inverted Generational Distance) is the mean distance from each $p \in \mathcal{P}^*$ to its nearest neighbor in $\hat{\mathcal{P}}$. Taking max captures both *missing regions* and *spurious predictions*. Smaller Δ is better; $\Delta = 0$ is perfect.

Efficiency metrics. We report (i) wall-clock runtime and (ii) budget, which is the number of the candidates evaluated. Runtime captures end-to-end cost but depends on external components (e.g., model training), whereas percentage explored is a model-agnostic proxy for intrinsic search efficiency. We present both for fair comparison.

6.1.2 Benchmark dataset. We instantiate tasks with the following datasets: *Visualization (single-attr., utility = Spearman):* BANK, PIMA, TITANIC, SONGS; *Interpretable modeling (multi-attr., utility = tree accuracy):* CROP, PIMA, TAXI, TITANIC, SONGS; *Imputation (multi-attr., utility = post–KNN-impute tree accuracy):* CROP, PIMA, TITANIC; *Causal analysis:* LALONDE (known ATE utility) and TITANIC (unknown ATE utility). For each case, we evaluate both LLM-based and distance-based (L2 and KL) semantic measures (cf. Table 1).

We assess accuracy by comparing each method's estimated Pareto front to a brute-force ground truth. Because fronts with fewer than three Pareto points provide poor resolution for trade-off evaluation and make distance metrics unstable, we exclude any scenario with *<3 Pareto points.* After filtering, the benchmark comprises 4 visualization, 5 modeling, 3 imputation, and 2 causal cases—ensuring that comparisons reflect each method's ability to recover meaningful, diverse trade-offs between utility and semantic quality. Table 2 reports statistics; #Candidates is the total number of candidate partition sets in the search space.

6.1.3 Methods Compared. As the OPS problem lacks established baselines, we compare Quill against naive heuristics, ablations of our framework, related methods, and LLM-based approaches.

- **Quill**: Our framework with a hyperparameter auto-tuner; for single-attribute cases, we use the UCB-guided MCC variant (Sec. 4.5).
- **UCB-I (Independent UCB)** [10]: Applies UCB independently to each attribute to obtain high-quality per-attribute candidate sets; evaluates joint objective $\mathcal{J}$ over the Cartesian product.
- **UCB-S (Sequential UCB)** [10]: Orders attributes and applies UCB sequentially. After fixing B_i for A_i, evaluation for A_{i+1} conditions on the current partial selection; $\mathcal{J}$ is recomputed as partitions are added, capturing inter-attribute interactions. We enumerate all attribute orderings and report the average performance.
- **Random**: Estimates the Pareto front via uniform random sampling from the space of candidate partition sets.
- **TreeFARMS (TF)** [58]: Enumerates near-optimal decision trees; we apply it to modeling tasks and post-process the resulting tree sets to estimate the Pareto front. We cap max runtime to 2 hours.
- **LLM**. We use GPT-4o [38] with in-context prompting as a transformation baseline, testing prompt variants (few-shot, chain-of-thought [55], and RAG [28]). For each downstream task, the prompt specifies the utility and semantic measures; the LLM then proposes a set of partition sets $\hat{\mathcal{B}}$ it estimates to lie on the Pareto front. This is done *without* enforcing a global evaluation budget.

Budget-time trade-off. UCB-I inherently explores a limited subset of candidates, while Quill, UCB-S, and Random are budget-controlled. To ensure comparability, we evaluate under a *matched budget* equal to the average number of candidates explored by UCB-I (222 candidates) for multi-attribute and 30 for single-attribute, unless stated otherwise. We note a trade-off between budget and runtime: higher budgets can improve accuracy at the cost of longer runtime. We study this trade-off in Section 6.2.2.

6.2 Experimental Results

6.2.1 Quality Comparison.
We evaluate quality by the average Hausdorff distance to the ground-truth Pareto front, $\Delta(\hat{\mathcal{P}}, \mathcal{P}^*)$ (lower is better).

Varying semantic scoring. Table 3 shows that Quill is consistently best across semantic measures.

For Visualization (single-attribute), the UCB methods match Quill exactly, which empirically aligns with our single-attribute proof in Section 4.5—when there is only one attribute, UCB and Quill learn the same policy to select clusters to explore and exploit. In multi-attribute settings, however, the methods diverge: Quill jointly optimizes partitions for all attributes, whereas UCB variants explore only a limited subset of combinations.

With the *LLM-based* semantic measure, Quill achieves the lowest overall error (0.11), reducing error by 61% vs. UCB-I (0.28), 71%

Table 3: Average error $\Delta(\hat{\mathcal{P}}, \mathcal{P}^*)$ across datasets (lower is better). "–" means the method is not applicable.

Method	Single-attr.	Multi-attr.			
	Visual	Modeling	Imputation	Causal	**Overall**
LLM-based semantic measure					
Quill	**0.04**	**0.10**	**0.08**	**0.16**	**0.11**
UCB-I	**0.04**	0.28	0.17	0.42	0.28
UCB-S	**0.04**	0.46	0.23	0.39	0.38
Random	0.17	0.44	0.47	0.45	0.45
TF	–	0.29	–	–	n/a
LLM	0.23	0.30	0.32	0.48	0.34
Distance-based semantic measure (L2 norm)					
Quill	**0.04**	**0.04**	0.17	**0.03**	**0.08**
UCB-I	**0.04**	0.07	0.15	0.09	0.10
UCB-S	**0.04**	0.09	0.15	0.18	0.13
Random	0.40	0.09	**0.07**	0.38	0.15
TF	–	0.18	–	–	n/a
LLM	0.40	0.38	0.24	0.45	0.35
Distance-based semantic measure (KL-divergence)					
Quill	**0.03**	**0.03**	**0.02**	**0.05**	**0.03**
UCB-I	**0.03**	0.06	0.07	0.08	0.07
UCB-S	**0.03**	0.07	0.07	0.19	0.09
Random	0.50	0.05	0.06	0.39	0.12
TF	–	0.23	–	–	n/a
LLM	0.27	0.26	0.38	0.35	0.31

vs. UCB-S (0.38), 76% vs. Random (0.45), and 68% vs. LLM (0.34). With *distance-based* measures, since we need to supply a gold partition for each attribute, we take the top-ranked partitions under the LLM-based semantic measure as the reference ("gold") and compute distances to these partitions, to quantify the interpretability of the candidates. We use this proxy because those high-ranked partitions are most frequently attested in literature or code and most align with human intuition (Sec. 5.1), and thus plausibly reflect domain expertise and practical utility. The pattern still holds: under L2, Quill attains 0.08 overall (20% vs. UCB-I at 0.10; 39% vs. UCB-S at 0.13; 47% vs. Random at 0.15; 77% vs. LLM at 0.35). Under KL, Quill reaches 0.03 overall (57% vs. UCB-I at 0.07; 67% vs. UCB-S at 0.09; 75% vs. Random at 0.12; 90% vs. LLM at 0.31).

Overall, while UCB matches Quill on single-attribute visualization (as expected), Quill dominates in the multi-attribute regime across metrics at a matched budget, reflecting its ability to more accurately estimate the Pareto front when attribute interactions matter. Note that for LLM, we extensively experimented with 8 prompting strategies to optimize the prompt for GPT-4o, including using few-shot vs. zero-shot, using COT [55] vs. no-COT, and the use of RAG [28] vs. no-RAG. We report the best prompt for LLM, which uses few-shot with COT, where few-shot examples provide demonstrations, with COT further enhancing structured reasoning. Despite extensive optimizations, LLM still lags behind. Details of our results in testing different prompt strategies can be found in Table 4.

Varying datasets. Figure 4 breaks results down by task–dataset pairs (LLM-based semantics). In nearly every panel, Quill is the lowest bar, showing small, stable errors across datasets. In Visualization, Quill is uniformly best across bank, pima, titanic, and songs; in Causal, it leads on both lalonde and titanic. Overall, the per-dataset breakdown indicates that Quill's gains are systematic (not driven by a few datasets) and that its performance is less sensitive to dataset idiosyncrasies than the baselines, yielding more reliable Pareto-front estimates across tasks.

Table 4: LLM $\Delta(\hat{\mathcal{P}}, \mathcal{P}^*)$ results using 8 prompting variations.

Example selection	Chain of thought	Visual	Modeling	Imputation	Causal
Zero-shot	Without CoT	0.32	0.35	0.37	0.59
	With CoT	0.27	0.36	0.37	0.60
Few-shot	Without CoT	0.29	0.34	0.41	0.55
	With CoT	0.26	0.38	0.44	0.55
RAG	Without CoT	0.30	0.41	0.35	0.57
	With CoT	0.27	0.43	0.34	0.58
Few-shot + RAG	Without CoT	0.28	0.34	0.38	0.56
	With CoT	0.26	0.35	0.39	0.58

Table 5: Ablation studies of Quill (budget = 2000).

	Full	**No auto-tuning**	**No DBSCAN**
Visual	**0.04**	0.10	0.08
Modeling	**0.06**	0.08	0.21
Imputation	**0.05**	0.07	0.13
Causal	**0.12**	0.08	0.09
Overall	**0.06**	0.08	0.14

6.2.2 *Scalability Analysis.*

We perform a sensitivity analysis to understand how the performance of Quill changes with different parameters (Fig. 5).

Varying numbers of total candidates. Fig. 5a shows that, in the *single-attribute* setting, Quill remains low and nearly flat as the candidate pool grows, whereas LLM and Random are consistently higher. In the *multi-attribute* setting, scaling the total candidates from 10^3 to 10^7 affects baselines much more than Quill. This indicates that Quill's cluster-based policy generalizes across larger search spaces without loss in estimation accuracy.

Varying data sizes. With more tuples (Fig. 5b), both single- and multi-attribute panels show Quill achieving the lowest error across the entire range. UCB-I/S are especially sensitive in the multi-attribute case as the data scale increases. Overall, Quill is the least sensitive to dataset size, suggesting robust Pareto-front estimation as dataset sizes grow.

Varying budgets. Increasing the evaluation budget monotonically improves quality ((Fig. 5c)), with Quill clearly maintaining the best performance across the budget range. Baselines either lack a budget knob (LLM, UCB-I/S) or improve far less with budget (Random), leaving a consistent gap to Quill. Returns for Quill are diminishing but steady—higher budgets further reduce error, reflecting effective use of additional exploration.

6.2.3 *Ablation Studies.*

We ablate two design choices of Quill (Table 5).

No hyperparameter auto-tuner. Replacing the auto-tuned configuration with a fixed, hand-tuned setting (fit on 2 datasets, detail in Section 4.6) degrades the *overall* error from **0.06** to **0.08** (+33%). The drop is visible across *Visual*, *Modeling*, and *Imputation*. The only exception is *Causal* (0.12→0.08), suggesting that a single global setting can occasionally align with a specific task, but the auto-tuner is more reliable across tasks and datasets.

No DBSCAN (hierarchical instead). Substituting DBSCAN with hierarchical clustering [37] substantially hurts performance (overall **0.14**, more than ×2 worse than full Quill). The largest regression appears in *Modeling* (0.06→0.21). This indicates that density-based

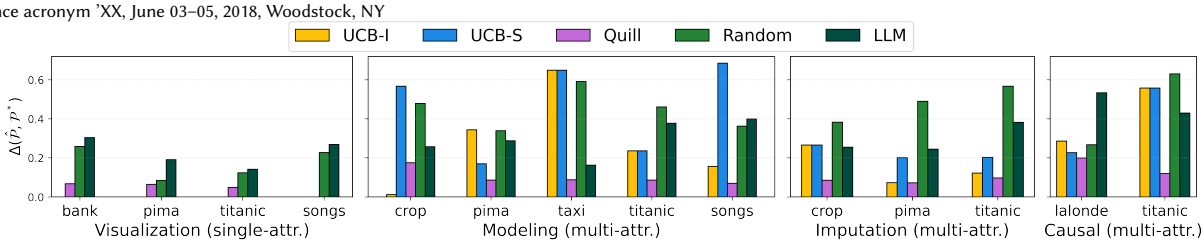
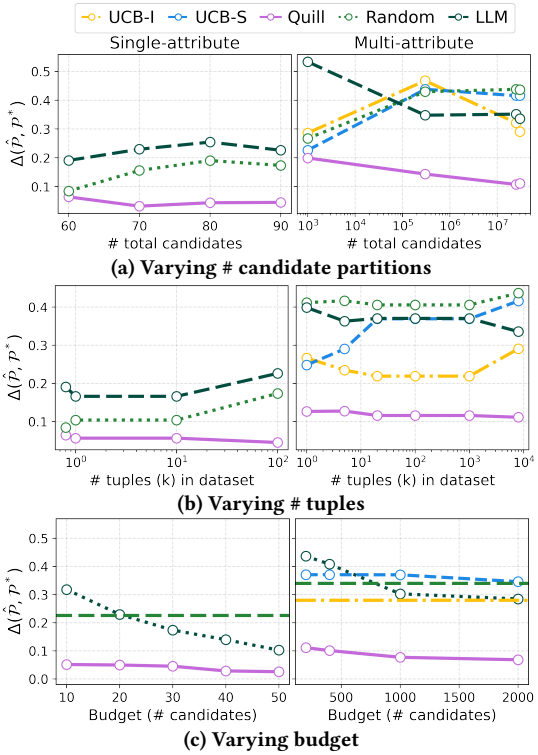


Figure 4: Average error by downstream tasks and datasets (LLM-based semantics); UCB variants omitted for single-attribute.



(a) Varying # candidate partitions

(b) Varying # tuples

(c) Varying budget

Figure 5: Sensitivity analysis (LLM-based semantics).

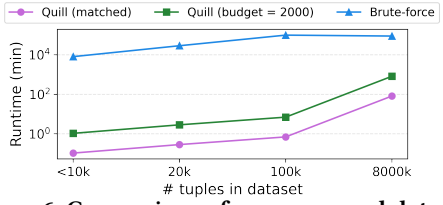


Figure 6: Comparison of average search latency.

clustering better captures the structure of the candidate space than purely linkage-based grouping, which is crucial for forming semantically coherent, utility-relevant clusters—together enabling the lowest overall error.

6.2.4 Efficiency Comparison.
We benchmark end-to-end latency by bucketizing datasets by size and averaging runtimes per bin (Fig. 6). As input size grows, the wall-clock time *per* candidate rises (e.g., fitting models on larger data), amplifying brute-force costs. In contrast, Quill (matched) strategically evaluates only ~200 candidates, versus millions in the full space (Table 2)—a 10^3–$10^5\times$ reduction (median $\approx 1.7 \times 10^3$). Consequently, for datasets under 100k tuples, Quill (matched) completes in under a minute on average, whereas brute force exceeds 16 hours even for < 10k tuples, underscoring the efficiency gains of our RL-based search.

Table 6: Representative partition sets illustrating the utility–semantics trade-off.

Set	Util.	Sem.	Partitions (BMI | Glucose | Age)	Notes
A (high sem.)	0.70	1.00	**BMI:** [0, 18.5, 25, 30, 68] **Glucose:** [0, 140, 200] **Age:** [0, 18, 35, 45, 65, 100]	WHO cutoffs Clinical thresholds Life stages
B (balanced)	0.71	0.83	**BMI:** [0, 17, 34, 51, 68] **Glucose:** [0, 140, 200] **Age:** [0, 25, 50, 75, 100]	Coarse equal steps Clinical thresholds Broad quartiles/decades
C (balanced, coarse)	0.73	0.58	**BMI:** [0, 35, 68] **Glucose:** [0, 140, 200] **Age:** [0, 20, 40, 60, 80, 100]	Coarse, WHO-aligned Clinical thresholds Broad decades
D (max utility)	0.81	0.06	**BMI:** [0, 7.46, 14.91, 22.37, 29.82, 37.28, 44.73, 52.19, 59.64, 67.10] **Glucose:** [0, 99, 157, 199] **Age:** [0, 29, 60]	Near-uniform slices Nonstandard Very coarse

6.3 Deeper Dive: Case Analysis

We analyze a representative run on Pima where the goal is to bin `Age`, `BMI`, and `Glucose` for a tree-based diabetes predictor (recall Ex. 2). Our framework returns multiple partition sets on the Pareto front, each trading semantic interpretability for predictive utility. For brevity, we highlight four illustrative partition sets from this run (the full output contains six).

Observations. Table 6 traces a clear Pareto curve: (A) *high semantics*—clinically meaningful and readable with strong utility (0.70/1.00); (B) *balanced*—coarse but still intelligible with improved utility (0.71/0.83); (C) *balanced, coarse*—further utility gain with reduced semantic alignment (0.73/0.58); and (D) *max utility*—highest accuracy (0.81) at minimal interpretability (0.06). Our framework (i) recovers domain-relevant partitions when they exist and (ii) surfaces nearby, higher-utility options so practitioners can select the point that best matches their interpretability–utility preference.

7 Conclusions and Future Work

Observing the need to identify partitions that are both useful and interpretable, we propose and formalize a new Optimal Partition Sets problem that has been overlooked thus far, design a robust, modular framework, with RL-based algorithms that jointly optimize both objectives.

Our framework has several limitations that suggest future work. (1) *Data cleanliness.* We assume clean inputs; outliers and missingness can skew counts and splits. Robust statistics and integrated imputation (with uncertainty) would improve reliability. (2) *Utility specification.* Utility is user-defined and optimized one metric per run. Many settings are multi-objective (accuracy, fairness, calibration, cost); extending to vector utilities and constraints is needed. (3) *Semantic subjectivity.* Interpretability depends on human and domain norms. We expose a Pareto front, but learned, domain-adapted semantic models (with uncertainty) could raise fidelity. (4)

Human-in-the-loop selection. Users still pick among Pareto candidates. Added decision support (provenance, counterfactuals, stability analyses) would further ease selection.

Our future work includes (i) *LLM-as-agent exploration*, where we replace exploration policies with an LLM-driven agent that consumes utility/semantic feedback, reasons over candidates, and adaptively decides what to sample next (active, budget-aware exploration); and (ii) *Parallelization and result sharing to cut latency*, especially on large datasets and expensive downstream tasks.

References

[1] [n. d.]. 19000 Spotify Songs Dataset. https://www.kaggle.com/datasets/edalrami/19000-spotify-songs. Accessed: 2025-06-29.

[2] [n. d.]. Credit Train Dataset. https://www.kaggle.com/datasets/zaurbegiev/my-dataset/data?select=credit_train.csv. Accessed: 2025-06-29.

[3] [n. d.]. Crop Recommendation Dataset. https://www.kaggle.com/datasets/atharvaingle/crop-recommendation-dataset/data. Accessed: 2025-06-29.

[4] [n. d.]. Pima Indians Diabetes Database. https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database. Accessed: 2025-06-29.

[5] [n. d.]. Taxi Trip Fare Data 2023. https://www.kaggle.com/datasets/hrish4/taxi-trip-fare-data-2023. Accessed: 2025-06-29.

[6] [n. d.]. Titanic Dataset. https://www.kaggle.com/datasets/yasserh/titanic-dataset. Accessed: 2025-06-29.

[7] 2025. Prolific — Online research participant recruitment platform. https://www.prolific.com/.

[8] Perplexity AI. 2021. https://www.perplexity.ai/. Accessed: January 12, 2025.

[9] Anonymous. 2025. Quill. https://anonymous.4open.science/r/quill-574E/full_technical_report.pdf

[10] P Auer. 2002. Finite-time Analysis of the Multiarmed Bandit Problem.

[11] Jesús Cerquides and Ramon López De Màntaras. 1997. Proposal and Empirical Comparison of a Parallelizable Distance-Based Discretization Method.. In *KDD*. 139–142.

[12] I. Csiszar. 1975. I-Divergence Geometry of Probability Distributions and Minimization Problems. *The Annals of Probability* 3, 1 (1975), 146 – 158. doi:10.1214/aop/1176996454

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

[14] Usama M Fayyad and Keki B Irani. 1993. Multi-interval discretization of continuous-valued attributes for classification learning. In *Ijcai*, Vol. 93. Citeseer, 1022–1029.

[15] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. 2015. Efficient and robust automated machine learning. In *Advances in Neural Information Processing Systems*. 2962–2970.

[16] Salvador Garcia, Julian Luengo, José Antonio Sáez, Victoria Lopez, and Francisco Herrera. 2012. A survey of discretization techniques: Taxonomy and empirical analysis in supervised learning. *TKDE* (2012), 734–750.

[17] Pascal Gijsbers, Erin LeDell, Janek Thomas, Stephane Poirier, Bernd Bischl, and Joaquin Vanschoren. 2019. An open source AutoML benchmark. In *Automated Machine Learning*. Springer, 219–239.

[18] Jan Hauke and Tomasz Kossowski. 2011. Comparison of values of Pearson's and Spearman's correlation coefficients on the same sets of data. *Quaestiones geographicae* (2011), 87–93.

[19] Yuval Heffetz, Roman Vainshtein, Gilad Katz, and Lior Rokach. 2020. DeepLine: AutoML Tool for Pipelines Generation Using Deep Reinforcement Learning and Hierarchical Actions Filtering. In *KDD '20*.

[20] Miguel A Hernán and James M Robins. 2020. *Causal Inference: What If*. Chapman & Hall/CRC. Available at https://www.hsph.harvard.edu/miguel-hernan/causal-inference-book/.

[21] Kosuke Imai, Gary King, and Elizabeth A Stuart. 2008. Misunderstandings between experimentalists and observationalists about causal inference. *Journal of the Royal Statistical Society: Series A (Statistics in Society)* 171, 2 (2008), 481–502.

[22] Guido W Imbens and Donald B Rubin. 2015. *Causal Inference for Statistics, Social, and Biomedical Sciences: An Introduction*. Cambridge University Press.

[23] Randy Kerber. 1992. Chimerge: Discretization of numeric attributes. In *AAAI*. 123–128.

[24] Klaus Krippendorff. 2011. *Computing Krippendorff's Alpha-Reliability*. Technical Report. University of Pennsylvania, Annenberg School for Communication. https://www.asc.upenn.edu/sites/default/files/2021-03/Computing%20Krippendorff%27s%20Alpha-Reliability.pdf Literature updated 2013.9.13.

[25] Himabindu Lakkaraju, Stephen H Bach, and Jure Leskovec. 2016. Interpretable decision sets: A joint framework for description and prediction. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 1675–1684.

[26] Robert J. LaLonde. 1986. Evaluating the econometric evaluations of training programs with experimental data. *The American Economic Review* 76, 4 (1986), 604–620.

[27] Benjamin Letham, Cynthia Rudin, Tyler H McCormick, and David Madigan. 2015. Interpretable classifiers using rules and Bayesian analysis: Building a better stroke prediction model. *The Annals of Applied Statistics* 9, 3 (2015), 1350–1371.

[28] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems* 33 (2020), 9459–9474.

[29] Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems* 30 (2017).

[30] Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

[31] Morris L Medley. 1980. Life satisfaction across four stages of adult life. *The International Journal of Aging and Human Development* (1980), 193–209.

[32] S. Mehta, S. Parthasarathy, and Hui Yang. 2005. Toward unsupervised correlation preserving discretization. *IEEE Transactions on Knowledge and Data Engineering* 17, 9 (2005), 1174–1185. doi:10.1109/TKDE.2005.153

[33] S. Mehta, S. Parthasarathy, and Hui Yang. 2005. Toward unsupervised correlation preserving discretization. *IEEE Transactions on Knowledge and Data Engineering* 17, 9 (2005), 1174–1285.

[34] Kaisa Miettinen. 1999. *Nonlinear multiobjective optimization*. Vol. 12. Springer Science & Business Media.

[35] Stefano Monti and Gregory F Cooper. 1999. A latent variable model for multivariate discretization.. In *AISTATS*. Citeseer.

[36] Dominik Moritz, Kanit Wongsuphasawat, Alper Sarikaya Chang, and Jeffrey Heer. 2019. Formalizing visualization design knowledge as constraints: Actionable and extensible models in Draco. In *IEEE Transactions on Visualization and Computer Graphics*, Vol. 25. IEEE, 438–448.

[37] Daniel Müllner. 2011. Modern hierarchical, agglomerative clustering algorithms. arXiv:1109.2378 [stat.ML] https://arxiv.org/abs/1109.2378

[38] OpenAI. 2024. GPT-4o. arXiv:2410.21276 [cs.CL] https://arxiv.org/abs/2410.21276

[39] Judea Pearl. 2009. Causal inference in statistics: An overview. (2009).

[40] Judea Pearl. 2009. *Causality*. Cambridge university press.

[41] 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*. ACM, 1135–1144.

[42] Paul R Rosenbaum and Donald B Rubin. 1984. Reducing bias in observational studies using subclassification on the propensity score. *J. Amer. Statist. Assoc.* 79, 387 (1984), 516–524.

[43] Peter J Rousseeuw. 1987. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math.* 20 (1987), 53–65.

[44] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. 2000. The earth mover's distance as a metric for image retrieval. *International journal of computer vision* 40 (2000), 99–121.

[45] Cynthia Rudin. 2019. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence* 1, 5 (2019), 206–215. doi:10.1038/s42256-019-0048-x

[46] Erich Schubert, Jörg Sander, Martin Ester, Hans Peter Kriegel, and Xiaowei Xu. 2017. DBSCAN revisited, revisited: why and how you should (still) use DBSCAN. *ACM Transactions on Database Systems (TODS)* 42, 3 (2017), 1–21.

[47] Oliver Schütze, Xavier Esquivel, Adriana Lara, and Carlos A. Coello Coello. 2012. Using the Averaged Hausdorff Distance as a Performance Measure in Evolutionary Multiobjective Optimization. *IEEE Trans. Evol. Comput.* 16, 4 (2012), 504–522. https://doi.org/10.1109/TEVC.2011.2161872

[48] scikit-learn developers. 2025. *sklearn.impute.KNNImputer*. scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.impute.KNNImputer.html Version 1.7.1.

[49] Jinwook Seo and Ben Shneiderman. 2005. A rank-by-feature framework for interactive exploration of multidimensional data. *Information visualization* 4, 2 (2005), 96–113.

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

[51] Charles Spearman. 1904. The proof and measurement of association between two things. *The American Journal of Psychology* 15, 1 (1904), 72–101.

[52] Richard S Sutton, Andrew G Barto, et al. 1998. *Reinforcement learning: An introduction*. Vol. 1. MIT press Cambridge.

[53] Berk Ustun and Cynthia Rudin. 2018. Learning optimized risk scores. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 1125–1134.

[54] Manasi Vartak, Sajjadur Rahman, Samuel Madden, Aditya Parameswaran, and Neoklis Polyzotis. 2015. Seedb: Efficient data-driven visualization recommendations to support visual analytics. In *PVLDB*. NIH Public Access.

[55] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems* 35 (2022), 24824–24837.

[56] Graham Wills and Leland Wilkinson. 2010. Autovis: automatic visualization. *Information Visualization* 9, 1 (2010), 47–69.

[57] Kanit Wongsuphasawat, Dominik Moritz, Anushka Anand, Jock Mackinlay, Bill Howe, and Jeffrey Heer. 2016. Voyager: Exploratory analysis via faceted browsing of visualization recommendations. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM, 5416–5427.

[58] Rui Xin, Chudi Zhong, Zhi Chen, Takuya Takagi, Margo Seltzer, and Cynthia Rudin. 2022. Exploring the Whole Rashomon Set of Sparse Decision Trees. arXiv:2209.08040 [cs.LG] https://arxiv.org/abs/2209.08040

Algorithm 2: Q-table masking (for one-partition-per-attribute constraint)

Input: Cluster assignments $\{C_1, \ldots, C_d\}$ with $k_i = |C_i|$ clusters for attribute A_i
Output: Masked Q-table $Q \in \mathbb{R}^{n \times n}$

Let $K \leftarrow \sum_{i=1}^{d} k_i$; // total non-terminal states
Let $n \leftarrow K + 2$; // include start s_0 (index 0) and terminal s_n (index $n-1$)
Initialize $Q \leftarrow -\infty \cdot \mathbf{1}^{n \times n}$; // everything forbidden by default
// Lay out consecutive index ranges for each attribute
$s \leftarrow 1$; // first cluster index (since 0 is s_0)
for $i \leftarrow 1$ **to** d **do**
  // Range of cluster-state indices for attribute A_i
  range$(A_i) \leftarrow [\, s,\ s + k_i - 1 \,]$;
  $s \leftarrow s + k_i$;
// Enable legal transitions across different attributes
foreach *pair of attributes* (A_i, A_j) *with* $i \neq j$ **do**
  for $r \in$ *range*(A_i) **do**
    for $c \in$ *range*(A_j) **do**
      $Q[r, c] \leftarrow 0$; // allow moving between attributes
// Start/terminal semantics
for $c \in \{1, \ldots, n-2\}$ **do**
  $Q[0, c] \leftarrow 0$; // from s_0 you may enter any first attribute state
$Q[0, n-1] \leftarrow -\infty$; // forbid jump $s_0 \to s_n$
for $j \in \{0, \ldots, n-2\}$ **do**
  $Q[n-1, j] \leftarrow -\infty$; // no outgoing actions from terminal s_n
for $i \in \{1, \ldots, n-2\}$ **do**
  $Q[i, n-1] \leftarrow 0$; // allow transition to terminal s_n from any non-terminal
return Q ;

Appendix A Candidate generation: discretizers

We instantiate candidate partitions for each numeric attribute using a mix of supervised and unsupervised discretizers. Unless noted, n_{bins} controls the granularity; default max $n_{\text{bins}} = 20$.

- **Equal-width (global)** (unsupervised; `equal-width`): Compute a constant width $\Delta =$ `(self.max_val − self.min_val)`$/n_{\text{bins}}$ over a fixed global range and form edges `self.min_val` $+ i\Delta$.
- **Equal-width (data-driven)** (unsupervised; `equal-width-data`): Call a helper (`equal_width`) that computes equal spans from the *observed* column range.
- **Equal-frequency / Quantile** (unsupervised; `equal-frequency`): Choose cut points so each bin contains (approximately) the same number of rows.
- **ChiMerge** (supervised; `chi-merge`): Bottom-up merging of adjacent intervals using a χ^2 statistic on class distributions (uses `target`) until n_{bins} (or a stopping rule) is reached.
- **KBinsDiscretizer (uniform)** (unsupervised; `kbins`): scikit-learn–style uniform-width binning wrapper for n_{bins}.
- **KBinsDiscretizer (quantile)** (unsupervised; `kbins-quantile`): scikit-learn–style quantile binning (approx. equal-frequency) for n_{bins}.
- **Decision-tree thresholds** (supervised; `decision-tree`): Fit a (univariate) tree using `target`; use learned split thresholds as bin edges (capped to n_{bins}).
- **KMeans cut points** (unsupervised; `kmeans`): Cluster the 1D values into $k = n_{\text{bins}}$ centroids; convert cluster boundaries to ordered cut points.
- **Random-forest thresholds** (supervised; `random-forest`): Aggregate split thresholds across trees trained with `target`; derive a consolidated, ordered set of edges (up to n_{bins}).
- **Bayesian Blocks** (unsupervised; `bayesian-blocks`): Adaptive histogram binning that places edges where changes in rate are supported by the data (no n_{bins} required).
- **MDLP** (supervised; `mdlp`): Minimum Description Length Principle discretization using class labels (`target`); recursively splits until MDL stopping criterion (exceptions are caught; method may yield no bins).

Appendix B Q-table masking

In Algorithm 2, the block-diagonal "same-attribute" regions remain at $-\infty$, so the agent can never transition between two clusters of the *same* attribute; combined with a single terminal state, this enforces selecting at most one partition per attribute while allowing any cross-attribute order.

8 Utility measures for causal analysis

The broad goal of causal analysis is to estimate the effect of an intervention or treatment on an outcome of interest, while accounting for factors that may bias this estimation. A common measure in this context is the **Average Treatment Effect (ATE)** [40], which quantifies the expected change in the outcome if every unit in the population were to receive the treatment compared to if none received it. Let $T \in \mathbb{A}$ be a binary treatment, Y the outcome variable, and $\mathbf{X} \subseteq \mathbb{A}$ a set of observed confounding variables that influence both T and Y. The ATE of T on Y can be expressed as:

$$\text{ATE}(T, Y) = \mathbb{E}_{\mathbf{X}}\left[\mathbb{E}[Y \mid T = 1, \mathbf{X}] - \mathbb{E}[Y \mid T = 0, \mathbf{X}]\right] \tag{7}$$

This formulation adjusts for confounders $\mathbf{X}$, ensuring that the estimated effect reflects causal influence rather than spurious associations.

Discretization of numerical confounders can simplify causal analysis by reducing the complexity of conditioning on high-dimensional continuous variables, enabling more reliable estimation of conditional expectations and improving the robustness of causal effect estimates. Furthermore, discretization facilitates easier implementation of estimation methods and aids in the identification of heterogeneous treatment effects by grouping units into meaningful subpopulations.

Unlike prediction tasks, where ground truth outcomes are observable, causal inference deals with counterfactual scenarios where there is no absolute truth to compare against. This unique characteristic necessitates specialized evaluation approaches to evaluate the utility of discritization. Therefore, we split our utility evaluation into two scenarios based on the availability of knowledge.

Known ATE. When the ground truth ATE is available, typically when dealing with synthetic or semi-synthetic data, we can directly evaluate the discretization effect. We measure the utility of a set of partitions $\mathbf{B}$ as the preservation of causal effect estimates:

$$\texttt{Util}_{\text{known}}(\mathcal{D}_{\mathbf{B}}) = \exp^{-|\text{ATE}_{\mathcal{D}}(T,Y) - \text{ATE}_{\mathcal{D}_{\mathbf{P}}}(T,Y)|} \tag{8}$$

Unknown ATE. When dealing with real-world observational data, the true ATE is unknown, leaving no direct comparison target. Given the critical importance of preserving relationships in causal analysis tasks, we developed a measurement that maintains data stability and simulates the discritization selection that would have been chosen if the ATE were known. Our approach relies on preserving inner correlations, specifically using Spearman correlation [51] due to its robustness to non-linear monotonic relationships common in real data.

We aim to maintain the stability of dependence and independence relationships in the data, preserving genuine causal connections without introducing biases through discretization. To achieve this, we compare correlation order preservation before and after partitioning the data, ensuring that high correlations remain high and low correlations remain low. To determine appropriate thresholds for classifying high and low correlations, we conducted experiments on synthetic data with known ATE, comparing

different thresholds by examining the relationship between partition selection under known ATE utility versus our correlation-based measure. We sought strategies that achieve high utility for known ATE to also receive high utility under our unknown measure, and vice versa. The quartile-based threshold ($\tau_{\text{low}} = 25^{\text{th}}$ percentile, $\tau_{\text{high}} = 75^{\text{th}}$ percentile) achieved the highest correlation with known ATE utility ($R^2 = 0.7989$, $p < 0.05$) and was therefore selected.

While we acknowledge that correlation does not imply causation, our goal is to estimate stability and select discretization strategies similar to those that would be chosen if the ATE were known. This approach aligns with principles established by [33], who demonstrated the importance of preserving correlation structure during discretization, though we specifically adapt this concept for causal inference applications. The utility is computed using F1-score where class 1 represents pairs classified as high correlation and class 0 represents low correlation:

$$\texttt{Util}_{\text{unknown}}(\mathcal{D}_{\mathbf{B}}) = \text{F1}(\text{corr}_{\text{original}}, \text{corr}_{\text{binned}}) \tag{9}$$

Since we aim to preserve both high and low correlations equally (total order preservation), we employ macro-averaging [30] in the F1-score comparison between binned and unbinned data.

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009