Identifying Valuable Data Subsets: Three Methods for Storage Optimization

Nina Fang
Utrecht University
n.fang1@students.uu.nl

Yi An Liu
Utrecht University
y.a.liu@students.uu.nl

Punya K
Utrecht University
p.krishnamurthy@students.uu.nl

ABSTRACT

This paper addresses the critical challenge of selective data retention in storage-constrained environments. We propose and evaluate three methods for identifying the most valuable subset of data: popularity-based selection, diversity-aware optimization, and Shapley-value computation. Due to technical challenges and time constraints limited our ability to collect comprehensive experimental results for comparative analysis.

1 INTRODUCTION

The exponential growth of data has made efficient data management a critical challenge. Space is running out, forcing companies to continuously seek strategies for optimizing storage utilization while preserving data value. This project addresses the fundamental problem of identifying the most valuable subsets of data when storage constraints necessitate selective retention.

We investigate three distinct methodologies for selecting a subset R' from a relational table R under a strict cardinality constraint T. The core challenge lies in defining and maximizing the importance of the selected subset, balancing multiple criteria including query utility, data diversity, and inherent tuple value. Each method approaches this problem from a different theoretical perspective: popularity-based selection, diversity-aware optimization, and cooperative game-theoretic allocation using Shapley values.

The remainder of this report is organized as follows: Section 2 details our solution approaches, Section 3 describes our experimental methodology and Section 4 presents and discusses results.

2 SOLUTION

Three different methods were implemented in Python using the Spark framework.

2.1 Setup

We employed the "Marketing Campaign" dataset from Kaggle (2,240 records, 29 attributes) as our primary data source. To ensure numerical consistency required by our similarity metrics, we transformed categorical attributes (*Education*, *Marital_Status*) using appropriate encoding schemes and converted temporal data (*Dt_Customer*) into numerical features. A configurable sampling function was implemented to generate subsets of varying sizes for scalability experiments.

The query generation process was implemented using a weighted function that creates synthetic query workloads based on the dataset's value distribution. Key features of this approach include:

Column Importance Tiers: Attributes were classified into three subjective importance categories (high, medium, low) with corresponding selection weights.

Query Complexity: Each query contains a maximum of $max\_conditions$ constraints, with no duplicate columns per query.

Value Range Awareness: Predicate values are sampled from the actual minimum and maximum ranges of each column.

Operator Distribution: The selection of both logical operators (AND/OR) and comparison operators was controlled through configurable weight parameters.

2.2 Method 1

Method 1 implements a straightforward popularity-based approach for selecting the most valuable tuples from dataset R. The core assumption is that a tuple's importance is directly proportional to its frequency of appearance in query results, formally defined as:

$$\text{imp}(R') = \frac{\sum_{i=1}^{|R'|} \text{imp}(t_i)}{\sum_{i=1}^{|R|} \text{imp}(t_i)} = \frac{\sum_{i=1}^{|R'|} \text{pop}(t_i)}{\sum_{i=1}^{|R|} \text{pop}(t_i)} \tag{1}$$

The popularity scores are calculated by evaluating all queries in a single columnar operation. For each query q_i in Q, a Spark SQL expression is constructed that evaluates to 1 for satisfying tuples and 0 otherwise. These expressions are summed directly without storing intermediate results, creating a single $total\_hits$ column where each value represents $pop(t)$ for the corresponding tuple. This approach is efficient because it avoids materializing temporary columns and leverages Spark's optimized expression evaluation. After computing all popularity scores, the dataset is sorted in descending order of $pop(t)$ and the top-T tuples are selected to form R'.

2.3 Method 2

Method 2 extends Method 1 by incorporating diversity-aware selection. While Method 1 only considers popularity, Method 2 penalizes tuples that are similar to other selected tuples. The key insight is that two very similar tuples that are both popular don't provide much additional value if they're redundant. Therefore, the importance of a tuple t is:

$$\text{imp}(t) = \text{pop}(t) * \frac{\sum_{i=1}^{|R'\setminus\{t\}|}(1 - \text{sim}(t, t_i))}{|R'|} \tag{2}$$

where $\text{sim}(t, t_i)$ represents the cosine similarity between tuples t and t_i. The set importance is then normalized as:

$$\text{imp}(R') = \frac{\sum_{i=1}^{|R'|} \text{imp}(t_i)}{\sum_{i=1}^{|R|} \text{imp}(t_i)} = \frac{\sum_{i=1}^{|R'|} \text{pop}(t_i) \times \frac{\sum_{k=1}^{|R'\setminus\{t_i\}|}(1-\text{sim}(t_i,t_k))}{|R'|}}{\sum_{i=1}^{|R|} \text{pop}(t_i) \times \frac{\sum_{k=1}^{|R'\setminus\{t_i\}|}(1-\text{sim}(t_i,t_k))}{|R|}} \tag{3}$$

,
.

Building upon Method 1's popularity scores, Method 2 implements a greedy selection algorithm. The process starts by identifying the tuple with the highest popularity score and adds it to R'. In subsequent iterations, the method selects tuples that maximize both popularity and diversity relative to the already selected tuples in R'.

2.4 Method 3

Method 3 employs an exact Shapley-value computation to select tuples based on their marginal contributions across all possible subsets. The approach defines subset importance using centroid-based dissimilarity:

$$\text{imp}(R') = \frac{\sum_{t \in R'} (1 - \text{cosine\_similarity}(t, \text{centroid}(R')))}{|R'|} \quad (4)$$

Each tuple's Shapley value is calculated by enumerating all 2^n subsets, with computational feasibility ensured by limiting dataset size to $n \leq 20$. The top-T tuples by Shapley score form the final selection R', with evaluation metrics including importance, diversity, runtime, and query coverage.

3 RESEARCH METHOD

To evaluate the performance and effectiveness of the three proposed methods for identifying the most valuable subset R', we designed a series of experiments. The goal of this experimental evaluation is to compare the methods across multiple dimensions, including runtime efficiency, the achieved importance score of the selected subset, the diversity of the selected tuples, and their utility in answering a given workload of queries.

Given the significant computational complexity differences between methods, we employed a two-phase experimental approach to ensure feasible execution while maintaining comprehensive evaluation.

We systematically vary key parameters to understand their impact on the methods' performance. For each unique configuration, we perform 5 runs with different random seeds to ensure the robustness and statistical significance of our results. All methods were evaluated using four consistent metrics per run:
Runtime: Processing time excluding I/O operations
Importance score: Method-specific measure of subset quality
Diversity: Quantified using average pairwise cosine similarity
Query coverage: Proportion of queries returning non-empty results

3.1 Phase 1

Due to Method 3's exponential time complexity $O(n \cdot 2^n)$, we conducted initial experiments with smaller datasets that remain computationally tractable for all methods. This enables direct comparison across all three approaches under identical conditions. While these dataset sizes are modest by conventional data management standards, they represent the computationally feasible upper bound for Method 3's computation given the project timeframe. The exponential complexity made larger instances impossible to complete within the available time constraints. The specific configurations for Phase 1 are summarized in Table 1.

Parameter	Value
$(\|R\|, T)$	(4,1) ; (8,2) ; (12,3)
$\|Q\|$	50, 100
Max cond./query	1, 3
Runs/config.	5

Table 1: Experimental Configurations for Phase 1

3.2 Phase 2

Methods 1 and 2, being computationally more efficient, were evaluated against larger datasets to assess their scalability and performance in practical scenarios. The specific configurations are summarized in Table 2.

Parameter	Value
$(\|R\|, T)$	(100,20); (500,100); (2000,400)
$\|Q\|$	100, 200
Max cond./query	1, 3
Runs/config.	5

Table 2: Experimental Configurations for Phase 2

This two-phase approach ensures both fair comparison across all methods and meaningful evaluation of scalability for practical applications.

4 RESULTS

Due to significant time constraints and computational limitations, we were unable to complete the comprehensive experimental evaluation as originally planned. The implementation of Method 2 encountered technical challenges that affected the accuracy of its diversity calculations, while Method 3's exponential complexity made extensive testing infeasible within the project timeframe.

These implementation issues, combined with the prevalence of zero-valued metrics in our preliminary results, prevented us from obtaining reliable empirical data for comparative analysis. Consequently, we are unable to draw meaningful conclusions about the relative performance of the three methods based on the available data.

The results of the tests can be found here:

- results 1
- results 2

It is expected that Method 1 provides an efficient baseline for small to medium datasets, while Method 2 offers improved diversity at moderate computational cost. Method 3 delivers theoretically optimal selections but remains practical only for small-scale problems due to its exponential complexity. These findings provide valuable guidance for organizations facing data retention decisions under storage constraints.