The Accuracy of Cardinality Estimators: Unraveling the Evaluation Result Conundrum

Nazanin Rashedi
University of Mannheim, Germany
nazanin.rashedi@uni-mannheim.de

Guido Moerkotte
University of Mannheim, Germany
moerkotte@uni-mannheim.de

ABSTRACT

Existing research on the accuracy of cardinality estimators generally suffers from a lack of diversity and sufficient quantity of their experimental datasets, particularly in relation to the claimed scope of the study and the generality of its conclusions. We argue that a sufficiently large number of varied datasets are essential for comprehensive evaluations. However, the prevailing per-dataset evaluation method (PDE), producing one result table per dataset, has so far hindered this necessary expansion of the experiments. Moreover, as we demonstrate, this evaluation method often leaves the reader with contradictory results, where one estimator excels on certain datasets or queries, while the other performs better elsewhere. To address these and similar limitations, we propose a multidimensional evaluation framework. This framework unravels the conundrum of analyzing the evaluation results across multiple datasets through the use of discretization. It establishes a robust foundation for aggregating the evaluation results and conducting pairwise comparisons between estimators. Furthermore, it facilitates informed decision making in the presence of conflicting results through a customizable ranking mechanism.

To empirically highlight the shortcomings of the aforementioned per-dataset evaluation and demonstrate the advantages of our proposed framework, we conduct a benchmarking study of cardinality estimators, incorporating both learned and traditional approaches. We focus on a fundamental challenge: estimating the cardinality of range queries on a single 2-D geographical relation in a static environment. Despite the apparent simplicity of this task, our findings reveal that many estimators struggle to handle this challenge effectively. To further enhance the quality of our study, we provide valuable insights by addressing some critical aspects that were overlooked in previous benchmarking studies.

PVLDB Reference Format:
Nazanin Rashedi and Guido Moerkotte. The Accuracy of Cardinality Estimators: Unraveling the Evaluation Result Conundrum. PVLDB, 18(11): 3744 - 3756, 2025.
doi:10.14778/3749646.3749651

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/Nazanin-Rashedi/CE_Accuracy.

Proceedings of the VLDB Endowment, Vol. 18, No. 11 ISSN 2150-8097.
doi:10.14778/3749646.3749651

1 INTRODUCTION

Motivation. During the last decade, an increasing number of cardinality estimators, including learned methods, have been proposed and numerous studies compared the accuracy of these estimators. However, their experiments fall short of the comprehensiveness required by the broad scope of their conclusions. In this study, we set the stage for large-scale experimentation across diverse datasets by proposing a multidimensional evaluation framework. Moreover, to showcase the analytical capabilities of this framework and address common shortcomings of prior work, we conduct a benchmarking study within a well-defined yet practical scope for the experimental datasets and workload. Importantly, we maintain this scope while drawing conclusions about the accuracy comparison of the estimators. We examine a general claim made by previous studies about the superiority of learned over traditional estimators for single relation range predicates in static environments (for more details, see Section 2). We use a wide range of traditional algorithms, including heuristic approaches and those tailored for 2-D numeric datasets, and select four top-performing learned estimators that according to recent studies [9, 12, 31, 34–36], exhibit superior accuracy compared to traditional algorithms in static environments. The focus on 2-D geographical datasets is motivated by their widespread availability and the broad range of applications. The crucial role of these datasets in industries like urban planning, transportation, and disaster management has led to the development of tailored cardinality estimators and database extensions, some of which are included in this study. Moreover, to the best of our knowledge, the specialized cardinality estimators proposed for these datasets have not been investigated in any prior benchmarking studies. It is noteworthy that although our experiments and the comparison conclusions focus on 2-D geographical datasets, the modifications we propose to the evaluation method are applicable to any benchmarking study on cardinality estimators. We believe these enhancements are crucial for conducting thorough comparisons between different methods.

Contributions. **1)** Our major contribution is proposing a novel evaluation framework that leverages discretization to establish a solid foundation for aggregating the evaluation results across diverse datasets. We discuss the limitations of current evaluation methods and address them in our framework. Most importantly, the aggregation capability of our framework resolves the key deficiency of the existing method (PDE) by eliminating the need to produce one result table per dataset, thereby facilitating comprehensive experiments across any number of datasets. Moreover, our approach enhances insights into the relationships between the major dimensions of the cardinality estimation problem, namely the input relation cardinality, the query selectivity, and the frequency and magnitude of the errors produced by the estimator. It enables flexible pairwise comparisons and a precision-based ranking of the

estimators. These aggregated results aid informed decision-making when selecting a cardinality estimator for a database system.

2) We address the existing bias in current benchmarking studies by highlighting that many previous comparisons between learned and traditional estimators fail to adequately consider the potential tuning knobs of traditional methods. For example, PostgreSQL has been criticized for significant errors in several studies. However, the absence of detailed configuration information suggests the potential use of default settings, without adjusting the number of statistic slots, using extended statistics, or incorporating available extensions. In contrast, we address the configurations of each estimator and, where feasible, evaluate variants with alternative settings to ensure a fair comparison.

3) In our efforts to identify the tuning knobs of the candidate estimators, we found opportunities for significant improvement in two of them, prompting us to introduce enhanced variants. First, we introduce EXGB, an improved version of the lightweight XGBoost estimator LW-XGB [5]. EXGB maintains the rapid training and inference capabilities of LW-XGB, while significantly boosting the estimation accuracy. Finally, we present QTS-2, a new variant of the traditional QTS method [2]. QTS-2 features a more efficient compression technique and an optimized splitting criterion, resulting in substantial improvements in the estimation accuracy.

Outline. The remainder of the paper is organized as follows. Section 2 reviews previous work. Section 3 and Section 4 describe the datasets and the workload we use to showcase the application of our proposed method, while Section 5 presents the cardinality estimators incorporated for this purpose. Section 6 describes our evaluation metric and its continued validity against the challenges discussed in a recent paper. In Section 7, we discuss the current per-dataset evaluation method and its limitations. Section 8 explains our proposed evaluation framework, its capabilities, and implementation details. Finally, Section 9 presents our conclusions.

2 RELATED WORK

Previous studies have conducted comparisons between cardinality estimators, and a recent trend is comparing learned vs. traditional estimators [5, 8, 11, 12, 17–19, 34, 36]. However, their experiments have always been restricted to a small number of datasets. This constraint is a consequence of the per-dataset evaluation method, which has been the standard approach for assessing cardinality estimators to date. Section 7 discusses this issue in more detail. We argue that a limited diversity of the experimental datasets coupled with generalized conclusions is a recipe for biased research and may result in the surveys' inability to reveal the deficiencies of candidate estimators. Wang et al. [34] conducted experiments on single-table cardinality estimators; however, the maximum number of distinct values per-column in their datasets is on the order of only a few thousand. Consequently, challenges such as those encountered by NARU [36] in encoding columns with a larger number of distinct values are not adequately reflected in their experiments. A remarkable number of these studies [5, 17, 18, 34, 36] evaluated the estimators using single relation, range predicate queries in a static environment and concluded that learned cardinality estimators are more accurate than their traditional counterparts in this setting. Notably, some of them [17, 18] incorporated 2-D geographical datasets in their handful of experimental datasets. However, our extensive experiments do not confirm their conclusion for 2-D geographical datasets. Several studies [9, 12, 18, 35] introduced new learned estimators and compared the accuracy of their proposed methods to that of various learned and traditional estimators. However, their evaluations neither describe nor optimize the configurations of the traditional competitors. We show that this neglect can lead to biased conclusions about the estimators' performance. Moreover, all previous studies on the accuracy of cardinality estimators were confined to producing one result table per dataset, reporting q-error percentiles for each estimator. This approach not only limited the number of experimental datasets but also impeded a deeper analysis of the evaluation results. Han et al. [8] mentioned a key shortcoming in the information provided by the typical evaluation results and, as a remedy, proposed a new metric for evaluation and optimization of the estimators. In contrast, we argue that the q-error [24] remains the appropriate metric and address both the limitation mentioned by Han et al. [8] and some other shortcomings of the prevailing evaluation method by proposing a novel evaluation framework.

3 DATASETS

We utilize a repository of 18,020 datasets, each containing the longitude and latitude of datapoints collected from the snapshot of three sets of datasets namely *Earthquake* [33], *OpenStreetMap* [27], and *Tiger* [32] in the year 2013, when we first conducted research on methods for estimating the number of points in two-dimensional datasets [23]. We consider two subsets from these datasets:

S_1: The subset with 5 datasets, explained in Subsection 3.1.

S_2: The subset of 276 datasets with details in Subsection 3.2.

Notably, for the top five estimators in our ranking results, Subsection 8.3, we use the whole set of 18,020 datasets.

3.1 Selection Process for Subset S_1

Although comprehensive experiments across a large number of datasets are preferable, this may not always be feasible. For such cases, we propose an informed approach to dataset selection. We collect statistical information on our datasets, each of which comprises two attributes: longitude (X) and latitude (Y). More precisely, we collect the number of distinct datapoints, total cardinality, variance (VX and VY), skewness (SX and SY), and the correlation between columns for each dataset in our dataset repository. For skewness and correlation, we consider the *Fisher-Pearson coefficient of skewness* and the *Pearson correlation coefficient* [13]. Table 1 includes these measures for the finally selected datasets. Furthermore, the table also presents two other properties of these datasets that will be needed later in the paper. For dataset selection, we employ *K-Means* clustering preceded by a *principal component analysis (PCA)*, a beneficial technique for enhancing clustering performance [4]. We initially apply PCA to the statistical data collected for our datasets and set the number of principal components to be 2. According to the cumulative sum of explained variance ratio, this already covers about 45 % of the data variance. We cluster the resulting datapoints and determine the number of clusters for the K-Means algorithm to be five by analyzing the inertia metric for up to 20 clusters. We then select the dataset corresponding to the datapoint nearest to each cluster centroid as the representative for that cluster. This

results in the selection of three datasets from *OpenStreetMap* (*india*, *netherlands*, and *us-south*), along with two datasets from *Tiger* (*tl_2013_02016_areawater* and *tl_2013_20041_linearwater*), referred to as *areawater* and *linearwater*, respectively. Figure 1 illustrates scatter plots of these selected datasets, showing diverse distributions. For example, *linearwater* exhibits a relatively uniform distribution across the two attributes, whereas *areawater* is sparse with high-density regions concentrated in specific areas. This diversity is further evidenced by the varying statistical measures in Table 1. Taking into account the impact of dataset characteristics on the estimators' performance, as studied by Wang et al. [34], this approach helps identify the limitations of each estimator.

3.2 Description of Datasets in Subset S_2

To conduct more extensive experiments on the estimators and demonstrate the aggregation benefits of our proposed framework, we build a larger subset of datasets by adding the remaining datasets in *OpenStreetMap* to S_1, resulting in S_2 with a total of 276 datasets.

3.3 Per-Dataset Budget Assignment

We apply budget constraints to our estimators, where applicable, to ensure an effective resource management and facilitate fair comparisons. However, a strict budget control is not feasible for some estimators. Specifically, for learned estimators, although the size of the trained model can be influenced by architectural and hyperparameter choices, enforcing the tight controls typical of traditional algorithms is less practical. This limitation extends to our PostgreSQL experiments as well. While PostgreSQL's memory consumption can be influenced by configuring the statistics collection process, actual memory usage is governed by the internal algorithms and processes of the DBMS, which we did not modify in our experiments. Wang et al. [34] and Yang et al. [36] suggest the budget limits of 1.5% and 1% of the dataset size, respectively. Nevertheless, for a real-world dataset that can easily reach 1 TB in size, allocating 10 to 15 GB to a cardinality estimator is excessive. Instead, we suggest a budget constraint of s statistic slots, each accommodating two 32-bit floating-point values, where s is determined by the dataset's cardinality $|R|$:

$$s := \left\lceil \sqrt{|R| \ln |R|} \right\rceil \tag{1}$$

The sublinearity of this function prevents huge storage assignments to large datasets, while allowing smaller datasets to gain a proper budget. Table 1 illustrates the budget (in Bytes) and the number of statistic slots s for the datasets in subset S_1.

4 WORKLOAD

In our experiments, we estimate the number of tuples that satisfy queries with half-open rectangles and simple, single-column predicates on both data dimensions using the following query template: select * from R where $x \geq x_l$ and $x < x_u$ and $y \geq y_l$ and $y < y_u$

For each dataset R, the workload consists of 1 M test queries, along with a set of 100 K queries allocated for training (90 K) and validation (10 K) of learned estimators. This set serves as the training and validation repository, with the exact number of queries used for each estimator detailed in the respective specification section. Queries are generated by randomly selecting a datapoint and applying four random real-valued offsets to determine the lower and upper bounds for each dimension, $(x_l,\ x_u,\ y_l, y_u)$, ensuring that each query returns at least one datapoint.

5 CARDINALITY ESTIMATORS

Here, we introduce the cardinality estimators used in this study.

5.1 Traditional Cardinality Estimators

Traditional cardinality estimators primarily rely on histograms and sampling techniques [8], though many variants of these methods exist. Here, we briefly introduce the traditional cardinality estimators employed in our experiments. It is noteworthy that the budget constraint mentioned in Subsection 3.3 is applied to all these traditional estimators with the exception of Est-Area, which is so cheap that it does not need any storage consumption considerations.

Sampling is a uniform sampler where the number of samples s is determined by Equation 1. For a query with m qualifying samples, the cardinality estimate, CE, is calculated as $CE = m|R|/s$, where $|R|$ denotes the size of the relation.

Est-Area is an estimator that assumes a uniform data distribution and independence among attributes. These assumptions often lead to significant misestimations on real-world datasets [22]. However, Est-Area can still serve as a fast and cost-effective baseline. To estimate the query selectivity, Est-Area calculates the overlap of each attribute domain with the query predicate and multiplies these proportions. To achieve this, it only needs to store the cardinality of the dataset and four floating-point values representing the dataset's bounding rectangle, thereby eliminating the need for budget considerations.

Regular Partitioning (REGP) [20] is a multidimensional histogram that employs the classic equi-width heuristic to determine the bucket boundaries.

Equi-depth histogram (EQD) [25] is a multidimensional histogram. It uses a multidimensional index structure called H-tree [7].

MHist [29] is a multidimensional histogram constructed by partitioning the joint data distribution into mutually disjoint buckets. In this study, we consider the variant that employs the *MHist-2* algorithm and utilizes *Maxdiff* and *Area* to identify the buckets which need to be partitioned.

QTS [2] is a hierarchical summarization technique proposed by Buccafurri et al. [2] for two-dimensional datasets. QTS relies on Quad-Tree partitioning of the data. The original version, referred to as QTS-0 in our experiments, partitions the nodes based on the *sum of squared errors (SSE)*, which measures the "goodness" of the compressed representation for a data array. We propose a new variant, QTS-2, which utilizes the *q-compression* technique [22] and employs cardinalities as the splitting criterion for the tree nodes. Additionally, Buccafurri et al. [2] proposed an Indexed Quad-Tree Summary (IQTS), a variant of QTS that allows indexing of the leaf nodes. In this study, we consider a variant of IQTS that uses maximum variance as the priority queue [2] criterion.

GXTree [23] is a hierarchical histogram that provides a compressed version of the frequency matrix through structured partitioning of the data distribution. Similar to QTS, the current implementation of GXTree is tailored for 2-D datasets, however, it can be extended to handle higher dimensions. Each node in a GXTree

Table 1: Properties of datasets in subset S_1

Dataset	Distinct Points	Cardinality	VX	VY	SX	SY	Correlation	Statistic Slots	Budget (Bytes)
areawater	46,798	48,314	11,500.00	0.99	2.52	0.78	-0.30	722	5,776
linearwater	189,541	193,623	0.02	0.02	-0.04	0.13	-0.04	1,536	12,288
india	18,592,034	18,610,840	42.02	24.07	-0.12	1.17	0.11	17,651	141,208
netherlands	86,177,043	86,192,127	0.33	0.68	0.16	-0.03	0.37	39,686	317,488
us-south	171,059,705	172,194,627	9.36	50.99	-0.54	-0.71	0.43	57,145	457,160

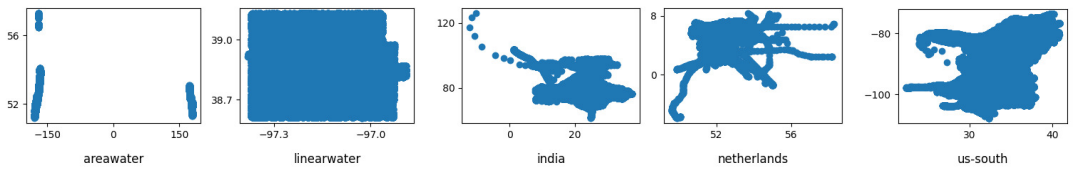

Figure 1: Scatter plot of datasets in subset S_1

stores an encoding of a regular partitioning, referred to as a grid. To minimize the number of bits required for grid representation, GXTree utilizes the q-compression technique.

PostgreSQL is frequently used as a baseline for evaluating learned cardinality estimators in comparison to traditional methods (e.g., [8, 9, 12, 18, 34–36]), often demonstrating significant errors. This poor performance is primarily attributed to its reliance on single-column statistics and the assumption of independence between columns. However, in many cases (e.g., [8, 9, 12, 18, 35]), the configurations of this estimator are neither customized nor mentioned in the description of the experiments. Similar to the learned methods, which require a careful tuning of hyperparameters, PostgreSQL can also be tuned to avoid the huge errors generated by the default settings. This can be achieved by setting a proper number of statistic slots, collecting extended statistics, or using specific extensions that are relevant to the task at hand [30]. In this study, in order to illustrate the role of such configurations, we consider the following three variants of PostgreSQL:
1) *PGDef* is the default, out of the box estimator in PostgreSQL version 15.6. with the default statistics target of 100 for all tables.
2) *PGB* is the variant with our budget limits considered in setting the number of per-column statistic slots. This controls the maximum number of entries collected as the most common values and the histogram buckets [30]. We set the number of statistic slots per column equal to s, where s is calculated by Equation 1.
3) *POSTGIS* is the variant with the PostGIS extension enabled. This extension adds support for geospatial data, which makes it a potential enhancement for the case of our experimental datasets. We set the number of statistic slots for the single column of type GEOMETRY equal to $2s$, where s is calculated by Equation 1, according to the budget limit for each table.

Two key points are noteworthy: First, we execute the ANALYZE command on each table to update the statistics prior to evaluating each variant. Second, PostgreSQL imposes an upper limit of 10 K for the number of per-column statistic slots.

5.2 Learned Cardinality Estimators

We select four learned estimators that have actively been used in the published benchmarking studies (e.g., [8, 18, 19, 34, 36]) demonstrating a promising accuracy. For each, we follow the implementation and configuration guidelines from the original paper unless stated otherwise. During hyperparameter tuning, we carefully monitor the training and validation loss to avoid overfitting.

DeepDB [9] is a data-driven cardinality estimator that builds probabilistic interpretations of the given queries and estimates the cardinality by evaluating the corresponding expectations and probabilities on some learned representations of data. These learned representations, called RSPNs, are a customized version of sum-product networks [28]. As a result of our experimental environment being restricted to single relation queries, for each dataset, we learn a base ensemble using the naive ensemble creation procedure provided in the published implementation [14]. According to the original paper, the storage consumption of DeepDB can be controlled by the sample size used for constructing RSPNs. With the aim of applying our budget limits mentioned in Subsection 3.3, we set the sample size using Equation 1. Notably, although the size of the generated ensemble is correlated with the sample size, it does not comply with our budget constraints. For instance, in the case of the *us-south* dataset, this configuration yields an ensemble size of 4.2 MB, which significantly exceeds our budget limit of 457 KB.

NARU [36] is a deep unsupervised cardinality estimator for single-relation queries. It estimates the joint distribution of input data using autoregressive models and performs progressive sampling to approximate the query selectivity. We use the implementation released in the corresponding GitHub repository [26]. NARU has been reported as a highly accurate learned estimator [19, 34, 36]. According to Yang et al. [36], it achieves a single-digit maximum q-error after about 15 epochs and with a storage consumption around 1 % of the dataset size. Wang et al. [34] found that NARU performs particularly well when the attribute domain is small but the dataset size, and therefore the model budget, is large [34]. Nevertheless, the maximum number of distinct values per attribute in their experiments [19, 34, 36] remains limited to a few thousand.

NARU employs column-specific encoders, starting with dictionary-based encoding to map attribute values to integer IDs. For domains with fewer than 64 distinct values, it applies one-hot encoding, while for larger domains, it switches to embedding encoding. The embedding encoding, however, becomes inefficient with a large number of distinct values, as it requires learning tensors of size $|A_i| \times h$, where $|A_i|$ represents the number of distinct values and h, the embedding vector size, is typically set to 64. The threshold at which this issue becomes problematic varies depending on the available GPU resources. Using our NVIDIA RTX A6000 GPU with 48 GB memory, training becomes impractical when $|A_i|$ exceeds 2.6×10^6, even after reducing the batch size from its default value of 1024 to 64. Utilizing mixed precision [21] only increases this threshold to 3.8×10^6. Consequently, NARU could not be trained on three of the five datasets in S_1 and many datasets in S_2. Therefore, we exclude it from the corresponding evaluation results.

Regarding the budget constraint, the original paper controls the storage consumption by the choice of model architecture. We follow the same approach; however, complying with the budget defined in Subsection 3.3, and the 1 % or 1.5 % of dataset size limits, are all infeasible for NARU. To clarify, we consider training NARU on *areawater*, a 1.2 MB dataset with two attributes, each containing 4.6×10^4 distinct values. With the settings recommended in the original paper, a *ResMade* model with four hidden layers of size 256, the model has over 5.8×10^6 parameters and a size of 25 MB on disk. While alternative architectures from the original paper perform significantly worse with respect to precision, they can only reduce the model size to 11 MB. The reason is that most parameters are associated to the embedding layer. Notably, the budget suggested in Subsection 3.3 and the limits of 1 %, and 1.5 % correspond to 5.8 KB, 12.3 KB, and 18.5 KB, respectively, indicating that NARU cannot meet these constraints. Hence, we pick the original paper's recommended architecture and lift the budget limit for NARU.

For *areawater* and *linearwater*, we train NARU with 15, 20, 50, 80 and 100 epochs. Since the training loss and the accuracy on the validation set show negligible variation for models trained with 50 or more epochs, we report the evaluation results for 50 epochs in Table 2 and Table 3. We evaluate the trained models using two variants of the evaluation step. In the first variant, denoted by NARU, we use 1 K progressive samples. According to Yang et al. [36], this number of progressive samples is sufficient for datasets with up to 100 columns and should therefore be adequate for our datasets, which contain only two columns. In the second variant, denoted by NARU2K, we increase the number of progressive samples to 2 K to assess whether this improves the estimates.

MSCN [12] is a query-driven cardinality estimator that employs multi-set convolutional neural networks and minimizes the mean q-error as its training objective. The preprocessing module converts the queries from the training set, consisting of 90 K queries for training and 10 K for validation, along with the test queries into feature vectors. These feature vectors incorporate the normalized predicate boundaries and a bitmap representing the materialized sample tuples that qualify the corresponding predicate. We follow the implementation in the GitHub repository introduced by the authors [1]. Given that the original paper does not offer explicit guidance on storage control in presence of the bitmap of samples, we do not impose any budget constraints on this estimator. Consequently, the resulting model is 2.6 MB, independent of the dataset size, which exceeds the budget defined in Subsection 3.3 for many datasets. MSCN normalizes labels using the minimum and maximum of log-transformed true cardinalities from the training and validation set and reverses this transformation during inference to estimate the cardinalities. This approach may reduce the model accuracy for test queries with true cardinalities outside the range experienced during the training [12].

LW-XGB [5] is a query-driven cardinality estimator that employs tree-based ensembles using XGBoost [3]. It uses 16 K queries for training and 4 K for validation. In addition to the predicate boundaries, LW-XGB uses three enriching features to provide the model with some information about the data distribution. Regarding the optimization metric, LW-XGB uses mean squared error (MSE) on log-transformed labels and predictions to minimize the geometric mean of q-errors. The storage consumption of the trained model primarily depends on the hyperparameters set for the maximum depth of trees and the number of trees. The design decisions of this estimator aim at minimizing the training and inference latency, as well as the storage footprint to achieve a level of performance comparable to the traditional competitors in these aspects. However, this gain comes at the cost of losing accuracy, since they restrict the enriching features to those derived from 1-D histograms with 100 buckets and keep the model structure as compact as a maximum of 16 trees, each with a depth of at most 4 (16 leaves). This structure results in a model size around 16 KB. For enriching features, the model utilizes predictions from three heuristic estimators: AVI, EBO, and MinSel, derived from 1-D histograms, as described in the original paper. These settings result in an estimator that, according to their experiments, needs less than one minute to produce 2 K training queries, compute the enriching features, and train the model on a dataset of 100 K tuples. However, since all these features rely on 1-D histograms, the accuracy degrades when the selectivity of each single predicate is high, but the conjunction of the predicates returns few tuples [34]. We construct the 1-D histograms in accordance with our budgetary constraints.

EXGB is our proposed variant of LW-XGB. Motivated by the work of Grinsztajn et al. [6], which highlights the superiority of tree-based models over deep learning for tabular data, we explore enhancements to LW-XGB. Our experiments demonstrate that the accuracy is highly dependent on the availability of rich feature vectors, beyond those restricted to 1-D histograms, and a sufficiently large number of trees. However, it is crucial to preserve the competitive advantage of low latency. In our proposed variant, EXGB, the enriching features consist of the estimates generated by two traditional estimators, namely Est-Area and Sampling, which are detailed in Subsection 5.1. We implement EXGB in C++ and leverage the rapid training and inference capabilities to bring the advantage of sample-based estimation to the enriching features. To rule out overfitting while increasing the maximum number of trees, we perform hyperparameter tuning on 20 randomly selected datasets from our repository and monitor the training and validation loss to ensure their consistent decrease and the absence of a significant gap between them. We conduct a grid search varying the maximum number of trees {10, 20, 50, 100, 300, 500, 1000}, the maximum tree depth {3, 5, 10}, and the regularization parameter gamma {0, 0.01, 0.1,

0.5}. Our findings confirm that the XGBoost built-in regularization parameter, gamma, effectively mitigates overfitting. We select the following hyperparameters: maximum number of trees = 300, maximum depth = 3 and regularization parameter gamma = 0.1. These settings produce models approximately 203 KB in size, yielding significantly more accurate estimates than LW-XGB, as shown in Tables 2 to 4. Additionally, we consider training/validation set sizes of 16 K/4 K and 90 K /10 K to assess the impact of using a training set larger than the one used for LWXGB[5]. Since no significant improvement in the validation loss was observed with the larger set, we retain the 16 K/4 K training/validation size. One might assume that the enhancements we propose would diminish the advantage of short latencies. However, in our C++ implementation, generating 2 K training and validation queries, calculating Sampling and Est-Area estimates for each query, and training the XGBoost model with the mentioned hyperparameters takes less than 6 seconds for the *tl_2013_45067_edges* dataset in the *Tiger* repository. We select this dataset for the latency experiments because its cardinality of 100,009 tuples aligns with the 100 K tuples in the measurements of Dutt et al. [5]. Regarding the inference latency, EXGB requires only 11 seconds to compute enriching features and make predictions for 1 M test queries. Our time measurements were obtained through single-threaded execution on a Xeon(R) E5-2690 v3 CPU. Notably, the authors have not publicly released the implementation of LW-XGB, preventing us from reproducing their latency-related experiments in our environment.

6 EVALUATION METRIC: Q-ERROR

The evaluation metric in our experiments is the q-error. The q-error of an estimate e for the true value t is defined [24] as:

$$q\text{-}error := \max(e/t, t/e). \tag{2}$$

This symmetric measure treats overestimations and underestimations equally. There is wide agreement in the literature that the q-error is the appropriate metric for evaluating cardinality estimators. However, Han et al. [8] challenge the ability of the q-error in distinguishing the errors that occurred for queries with large true cardinalities, thereby significantly impacting the quality of the generated plans. They further argue that a lower q-error does not necessarily translate into better execution plans. On the contrary, we contend that the q-error remains the appropriate metric for evaluating and optimizing the accuracy of cardinality estimators for the following reasons:

1) The q-error is directly related to the quality of the generated plans, since it provides a theoretical upper bound for the plan quality if the q-errors of a query are bounded [10, 24].

2) While we agree with Han et al. [8] that the true cardinality of a query affects the impact of a specific q-error on the plan quality, we contend that the lack of this critical information in the evaluation results arises from flaws in the evaluation method rather than from the metric. Our multidimensional evaluation framework introduced in Section 8 addresses this issue.

3) The query optimizer consists of multiple modules, each with their own characteristics, limitations, and errors. Therefore, the possible phenomenon of more accurate cardinality estimates resulting in worse execution plans [8, 15] should be traced and solved in the evaluations of the appropriate modules, namely the cost function or the plan generator, instead of encouraging the cardinality estimator to make less accurate estimates and compensate for errors caused by other modules. This issue is clearly exemplified in the experiments of Leis et al. [16] and Lee et al. [15] where, for some queries, injecting the true cardinalities in the query optimizer resulted in plans with increased execution times. With a similar argument, any metric that uses the plan costs for evaluating cardinality estimators, as is the case for the *p-error* proposed by Han et al. [8], suffers from conflating the errors produced by two distinct and ideally independent modules of the query optimizer.

7 PER-DATASET EVALUATION (PDE)

The predominant approach for evaluating and comparing the accuracy of cardinality estimators involves reporting the 50^{th}, 95^{th}, 99^{th}, and 100^{th} percentiles of the q-errors generated by each estimator on a per-dataset basis. This approach neglects the selectivities of the queries associated with these q-errors (e.g., [5, 8, 9, 11, 12, 19, 31, 34]). Some studies [18, 36] have advanced their analysis one step further by reporting the q-error percentiles in three selectivity categories. More precisely, they set fixed intervals to distinguish between high, medium, and low selectivity queries and, for each dataset, report the q-error percentiles per selectivity category. We refer to this approach, which is thus far the most comprehensive, as per-dataset evaluation (PDE) and apply it with the selectivity boundaries defined by Yang et al. [36]. By designating this approach as per-dataset, we highlight the unavoidable one-to-one relationship between datasets and their corresponding result tables in this evaluation method. Furthermore, we refer to the entries in PDE result tables as *PDE segments*. Consequently, *similar PDE segments* are those sharing the same percentile levels and selectivity categories.

7.1 PDE Results on Datasets for Subset S_1

Tables 2 to 4 display the q-error percentiles for three datasets of subset S_1. Due to space constraints and the need to allocate one result table per dataset in PDE, the results of the other two datasets are excluded and will be released in an extended version of this paper. In these tables, the upper sections list traditional methods, while the lower sections show learned estimators. Focusing on the learned estimators, both variants of NARU exhibit high q-errors for the two datasets on which we could train it. Even the increased number of progressive samples in NARU2K fails to yield better estimates. Similarly, MSCN produces significant errors. LW-XGB outperforms MSCN, NARU, and NARU2K, while EXGB surpasses LW-XGB in accuracy. If we ignore the few segments where DeepDB shows lower q-error percentile values than EXGB, we can conclude that EXGB is the most accurate learned estimator followed by DeepDB. Incorporating the traditional estimators into our analysis yields the following findings: Est-Area and PGDef make huge errors. In some cases, PGDef is even worse than our cheap baseline, Est-Area. However, the two configured variants of PostgreSQL, namely PGB and POSTGIS, have significantly better results. This underscores that conducting comparisons without an appropriate configuration can lead to misleading conclusions. POSTGIS is the winner estimator for *linearwater* in all selectivity categories; however, its accuracy degrades for other datasets. For the other two datasets, QTS-2 and GXTree can be considered the most accurate estimators. Notably,

QTS-2 provides significantly more accurate estimates than QTS-0. As these tables show, most estimators exhibit varying levels of accuracy across different datasets and selectivity categories. These contradictory results make it difficult to establish a clear pairwise superiority or ranking. We address this issue in Section 8.

7.2 Limitations of PDE

PDE has several notable shortcomings, such as:

1) The primary drawback of PDE is its lack of scalability with respect to the number of datasets. While comprehensive evaluations require testing the estimators across numerous datasets, the one-to-one correspondence between datasets and PDE result tables severely hinders this necessary extension. Our experiments with PDE were confined to five datasets, yet they already generated a substantial amount of numbers across five tables, three of which are presented in Tables 2 to 4. How overwhelming would it be to be left with hundreds of such tables and try to draw meaningful conclusions from them?

2) The data entries in similar PDE segments are neither comparable nor aggregable across different datasets. This stems from using fixed boundaries for categorizing query selectivities on datasets of varying cardinalities. To be more precise, when we define low selectivity queries as those with a selectivity below 0.5%, as it is defined by Yang et al. [36], it may correspond to queries with fewer than 10 result tuples in datasets of cardinality below 2 K or up to 1 M tuples in datasets of cardinality 200 M. Notably, the impact of a specific q-error is heavily influenced by the true cardinality of the corresponding query. Therefore, the q-error values and consequently the percentiles derived from them are neither comparable nor aggregable across datasets with various cardinalities, despite ending up in similar PDE segments. This prevents PDE from offering a solid basis for meaningful aggregation and confirms that this approach becomes impractical for a larger number of datasets.

3) Analyzing each estimator's worst-case errors is crucial for a thorough assessment of the estimators' accuracy. However, PDE falls short by only reporting the 99^{th} and 100^{th} percentiles, neglecting the frequency of such extreme errors.

4) PDE leaves the reader with the conundrum of interpreting per-dataset evaluation results that often present conflicting outcomes, where one estimator outperforms another on certain datasets, but not on other datasets. Take POSTGIS and Sampling, for example. Assume we plan to use the evaluation results in Tables 2 to 4 to choose one of these two estimators for the problem of estimating the cardinality of single relation queries on 2-D geographical datasets. Even with the evaluation results of only three datasets, arriving at a conclusion remains challenging. In particular, the winner between these two estimators is Sampling for *areawater* as well as for the majority of segments for *us-south*; however, the winner is POSTGIS for *linearwater*. So what can we conclude? Such contradictory results are often inevitable. Even within the results for a single dataset, it can be challenging to determine which method yields more accurate estimates and to surpass the typical non-conclusive comparison outcomes, such as:

Regarding the accuracy comparison between EQD and REGP on areawater *(Table 2), EQD outperforms REGP in median q-errors, the 95^{th} percentile for high selectivity queries, and maximum q-error for low selectivity queries, while REGP shows a better accuracy elsewhere.*

Overcoming these challenges requires information that is not provided by PDE result tables. In the following, we pose three example questions that are important in analyzing the evaluation results but remain unanswered by PDE. According to the evaluation results on *us-south* (Table 4), we observe that EXGB, POSTGIS, and Sampling demonstrate quite acceptable q-error percentiles except for the PDE segment corresponding to the maximum q-error on low selectivity queries, where the q-error rapidly jumps to 5.2×10^3, 1.5×10^4, and 1.8×10^4 respectively. It is interesting to know:

Q_1) How often do such high-end q-errors, for instance q-errors larger than 512, occur per estimator?

Q_2) How many low-selectivity queries did our query generator produce for this dataset?

Q_3) What are the true cardinalities of these misestimated queries, given that they can range from 1 to 860,973 tuples?

8 MULTIDIMENSIONAL EVALUATION (MDE)

In this section, we propose the *multidimensional evaluation framework (MDE)* that addresses the limitations of PDE. Most importantly, this framework supports experiments on any number of datasets. It is important to note that experimentation on numerous datasets has been hindered up to this point due to PDE's inability to aggregate evaluation results, limiting it to generating one result table per dataset. To illustrate the ability of MDE in handling this challenge, we pick the task of evaluating each of our 16 estimators on 276 datasets in subset S_2. While percentiles provide a compact summary of the q-error distribution, overcoming the limitations of PDE requires evaluation data that is not only more granular but also suitable for meaningful aggregation and effective presentation. In the following, we introduce our multidimensional profile, which forms the foundation for meeting these requirements.

8.1 Multidimensional Profile (MDP)

We collect the raw evaluation results in $(ds, est, card, sel, qe)$ tuples, where ds represents the dataset, est the estimator, $card$ the cardinality of the relation, sel the selectivity of the query, and qe the q-error. Notably, the true cardinality of each query, which is a critical missing piece in PDE, can be derived from the selectivity of the query and the dataset cardinality. For our experiments on the datasets in subset S_2, this produces $16 \times 276 \times 10^6$ tuples that provide fine-grained, yet non-aggregable evaluation results which are difficult to present or conclude from effectively. In order to provide the necessary foundation for aggregation, we apply discretization. For this purpose, we propose the logarithmic scale where the base of the logarithm can be set according to the desired precision of the discretization. While using identical bases for discretizing the cardinality and the selectivity is beneficial for further analysis, the base for the q-error can differ from them. For a relation R with cardinality $|R|$, we define the cardinality class of the relation (cc), the selectivity class of the query (sc), and the q-error class of the

Table 2: Evaluation results on areawater dataset

Method	High Selectivity (2%, 100%]				Medium Selectivity (0.5%, 2%)				Low Selectivity (≤ 0.5%)			
	50^{th}	95^{th}	99^{th}	100^{th}	50^{th}	95^{th}	99^{th}	100^{th}	50^{th}	95^{th}	99^{th}	100^{th}
EQD	1.04	1.18	10.71	529.50	1.24	15.76	68.49	725.00	1.64	15.72	58.00	390.00
Est-Area	65.67	491.00	$1.3 \cdot 10^3$	$5.8 \cdot 10^3$	7.92	109.00	358.00	966.00	4.18	73.50	182.00	241.00
GXTree	1.02	1.08	1.12	**1.34**	**1.06**	**1.20**	**1.32**	**2.01**	**1.14**	**1.81**	**3.40**	**28.00**
IQTS	1.03	1.18	1.42	6.55	1.11	2.33	3.42	50.00	1.24	4.00	11.75	264.80
MHist	3.05	5.15	6.38	15.15	1.39	5.91	9.15	27.82	1.56	7.80	20.03	932.00
PGDef	69.30	184.18	$1.5 \cdot 10^3$	$6.5 \cdot 10^3$	5.17	850.00	945.00	966.00	8.42	220.00	237.00	241.00
PGB	1.77	3.86	11.12	19.54	2.33	15.09	20.07	95.67	3.04	24.75	70.00	582.50
POSTGIS	15.64	105.12	231.66	$1.1 \cdot 10^3$	8.24	397.00	608.00	902.00	6.17	177.00	225.02	241.00
QTS-0	6.07	46.92	114.40	542.25	1.04	5.81	23.20	763.00	1.07	6.50	26.46	322.50
QTS-2	**1.01**	**1.05**	**1.11**	1.69	1.06	1.26	1.46	4.39	1.17	2.44	6.00	49.00
REGP	1.67	2.28	3.31	12.18	1.31	4.56	7.31	125.00	1.72	8.22	24.84	558.60
Sampling	1.03	1.26	1.73	3.92	1.25	2.61	4.11	548.00	1.77	117.00	187.00	241.00
DeepDB	1.06	1.31	1.54	3.40	1.26	2.08	3.65	379.00	1.74	88.00	195.00	241.00
EXGB	1.02	1.11	1.20	2.88	1.10	1.44	1.86	52.00	1.34	3.35	8.29	53.50
LW-XGB	2.06	2.59	3.02	5.31	1.35	2.35	3.19	13.77	2.14	12.41	40.67	715.00
MSCN	3.34	46.12	177.54	$1.9 \cdot 10^4$	3.35	46.66	180.36	$1.2 \cdot 10^4$	3.34	45.50	171.79	$9.9 \cdot 10^3$
NARU2K	3.26	11.83	22.81	43.86	16.76	108.70	147.06	179.41	60.45	794.28	$3.2 \cdot 10^3$	$4.4 \cdot 10^4$
NARU	3.53	13.31	23.28	43.31	20.71	106.50	144.89	176.29	74.85	839.59	$3.3 \cdot 10^3$	$4.3 \cdot 10^4$

Table 3: Evaluation results on linearwater dataset

Method	High Selectivity (2%, 100%]				Medium Selectivity (0.5%, 2%)				Low Selectivity (≤ 0.5%)			
	50^{th}	95^{th}	99^{th}	100^{th}	50^{th}	95^{th}	99^{th}	100^{th}	50^{th}	95^{th}	99^{th}	100^{th}
EQD	1.05	1.12	1.13	1.13	1.05	1.17	1.26	1.88	1.12	1.75	2.75	68.50
Est-Area	2.30	2.37	2.37	2.37	1.40	2.03	2.48	4.09	1.37	2.63	4.00	262.50
GXTree	1.02	1.04	1.04	1.04	1.04	1.11	1.14	1.26	1.07	1.35	1.84	54.00
IQTS	1.04	1.05	1.06	1.06	1.04	1.14	1.21	1.49	1.09	1.56	2.35	59.00
MHist	1.73	1.78	1.78	1.78	1.32	1.87	2.16	3.23	1.32	2.48	3.77	95.50
PGDef	4.24	4.40	4.40	4.40	2.67	17.01	$1.1 \cdot 10^3$	$2.9 \cdot 10^3$	1.97	12.32	83.00	968.00
PGB	1.83	1.96	1.96	1.96	1.24	1.90	2.70	8.87	1.32	2.50	4.60	158.50
POSTGIS	**1.00**	**1.01**	**1.02**	**1.02**	**1.01**	**1.05**	**1.07**	**1.20**	**1.03**	**1.23**	**1.59**	**31.00**
QTS-0	2.52	2.60	2.60	2.60	1.27	2.02	2.68	4.48	1.28	2.48	3.80	239.50
QTS-2	1.01	1.03	1.03	1.03	1.02	1.07	1.10	1.23	1.05	1.34	1.89	58.00
REGP	1.01	1.02	1.02	1.02	1.02	1.06	1.09	1.31	1.04	1.29	1.74	41.00
Sampling	1.38	1.54	1.57	1.58	1.23	1.96	2.84	$1.2 \cdot 10^3$	1.50	152.00	342.00	967.00
DeepDB	1.37	1.53	1.53	1.54	1.25	1.94	3.00	8.84	1.38	2.93	5.91	271.00
EXGB	1.18	1.28	1.30	1.31	1.10	1.34	1.48	2.26	1.19	1.93	2.86	104.00
LW-XGB	3.18	3.23	3.24	3.24	1.41	1.99	2.36	4.00	1.27	2.47	4.56	178.00
MSCN	2.10	5.04	5.10	5.12	2.29	12.55	29.86	$1.9 \cdot 10^3$	2.28	12.76	30.79	$2.3 \cdot 10^3$
NARU2K	6.52	7.64	7.71	7.72	34.14	107.82	133.39	180.99	111.74	797.98	$2.3 \cdot 10^3$	$1.7 \cdot 10^5$
NARU	6.40	7.62	7.69	7.71	34.43	106.17	131.80	180.66	112.05	790.63	$2.3 \cdot 10^3$	$1.7 \cdot 10^5$

estimate (qc) as follows:

$$\begin{aligned} cc &:= \lfloor \log_2 |R| \rfloor \\ sc &:= \lfloor -\log_2 sel \rfloor \\ qc &:= \lfloor \log_{\sqrt{2}} qe \rfloor. \end{aligned} \tag{3}$$

Although discretization introduces some precision loss, this loss can be controlled by the choice of the logarithmic base. With our chosen bases, the loss is bounded by a factor of two for cardinality and selectivity, and a factor of $\sqrt{2}$ for the q-error. Moreover, for each cc and sc combination, we can approximate the true cardinality of the query, true_card, in the following interval:

$$\text{true\_card} \in [2^{cc-sc}, 2^{cc-sc+1}). \tag{4}$$

Given a (cc, sc) pair, this approximation is sufficient to address the issues of aggregability and comparability of the evaluation segments that we had in PDE. By definition, cardinality class values are determined solely by the range of dataset cardinalities, which

Table 4: Evaluation results on us-south dataset

	High Selectivity (2%, 100%]				Medium Selectivity (0.5%, 2%)				Low Selectivity (≤ 0.5%)			
Method	50^{th}	95^{th}	99^{th}	100^{th}	50^{th}	95^{th}	99^{th}	100^{th}	50^{th}	95^{th}	99^{th}	100^{th}
EQD	1.00	1.01	1.01	1.12	1.00	1.02	1.05	1.96	1.01	1.10	1.56	859.00
Est-Area	8.35	17.59	24.08	51.36	5.02	11.88	25.83	166.39	1.94	8.60	12.82	$6.2 \cdot 10^4$
GXTree	1.01	1.02	1.02	1.04	1.00	1.01	1.02	1.04	1.01	1.03	1.05	24.32
IQTS	1.00	1.01	1.02	1.05	1.00	1.01	1.03	1.21	1.01	1.03	1.08	4.00
MHist	6.15	12.96	17.75	37.85	3.70	8.75	19.03	122.60	1.78	6.42	9.68	$3.5 \cdot 10^3$
PGDef	15.65	48.65	65.78	111.95	3.91	18.62	64.18	$3.3 \cdot 10^6$	1.86	20.01	$4.1 \cdot 10^5$	$8.6 \cdot 10^5$
PGB	1.73	5.04	6.62	11.19	1.66	5.69	7.97	27.44	1.69	5.34	10.16	$3.0 \cdot 10^3$
POSTGIS	1.03	1.12	1.20	1.44	1.05	1.19	1.31	3.62	1.09	1.38	1.77	$1.5 \cdot 10^4$
QTS-0	3.79	11.88	16.93	27.41	2.80	12.82	20.29	92.85	1.94	6.73	14.46	$3.7 \cdot 10^4$
QTS-2	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.01**	**1.00**	**1.01**	**1.03**	20.00
REGP	1.00	1.00	1.01	1.03	1.00	1.01	1.01	1.14	1.00	1.02	1.04	**2.81**
Sampling	1.01	1.05	1.08	1.12	1.03	1.10	1.14	1.28	1.06	1.26	1.53	$1.8 \cdot 10^4$
DeepDB	1.02	1.07	1.09	1.15	1.04	1.16	1.24	1.64	1.12	1.61	2.33	$3.0 \cdot 10^4$
EXGB	1.01	1.04	1.06	1.31	1.02	1.08	1.11	1.37	1.05	1.24	1.54	$5.2 \cdot 10^3$
LW-XGB	4.10	6.93	8.17	12.66	2.68	5.17	7.53	14.31	1.60	3.42	5.14	$9.4 \cdot 10^3$
MSCN	3.92	40.31	133.14	$8.5 \cdot 10^4$	2.33	13.26	41.20	$1.1 \cdot 10^5$	4.33	33.23	75.72	$3.1 \cdot 10^4$

for subset S_2 spans from 13 to 27. The selectivity class values, on the other hand, range from 1, indicating queries that return all tuples of the relation, to a maximum value equal to the cardinality class of the relation, which is observed in cases where the query retrieves a single tuple. We cap q-errors larger than 2^{16} by assigning them to the *qc* value of 32 for visualization purposes. Such large q-errors indicate a level of inaccuracy that is highly undesirable, irrespective of the specific q-error value. Notably, this cap is not a mandatory part of our profile structure and can be adjusted or removed entirely based on the analyst's preferences. Having introduced the building blocks, we can now proceed to construct our multidimensional profile, which serves as the solid foundation for aggregating the evaluation results across diverse datasets. For each dataset *ds* and estimator *est*, we collect the *cc* of the relation along with the *qc* and *sc* of each test query. Duplicated tuples are aggregated, with their frequency stored as *cnt*. This results in tuples that can be organized in a database relation with the following relational schema:

$$MDP(ds, est, cc, sc, qc, cnt). \tag{5}$$

We refer to this structure as the *multidimensional profile (MDP)*. For our experiments, this yields 666,124 tuples, representing a substantial reduction from the number of result tuples prior to discretization. In order to provide some numeric examples of the insights we gain from this profile, we address the three questions posed in Subsection 7.2 and concentrate on the specified dataset and estimators. The answer to Q_1 is obtained by

$$\Gamma_{\text{est};sum(\text{cnt})}\left(\sigma_{(ds=\text{“us-south”}\wedge qc>18)}(MDP)\right),$$

where Γ denotes the group-by operator. This yields 1, 26, and 162 queries for EXGB, POSTGIS, and Sampling, respectively. We can approximate the answer to Q_2 through

$$sum(\text{cnt})\left(\sigma_{(ds=\text{“us-south”}\wedge sc>7)}(MDP)\right).$$

In this expression, $sc > 7$ corresponds to the definition of low selectivity category in PDE. The result indicates that 2.7×10^5 out of 10^6 queries generated for *us-south* fall into this category. This means the PDE segment corresponding to the maximum q-error on low selectivity queries consolidates the evaluation results of 2.7×10^3 queries in reporting only the maximum q-error for these queries, regardless of the frequency of such high end errors, which may in this case range from 1 to 2.7×10^3. To answer Q_3 we obtain the selectivity class of the queries with high *qc* values through

$$\Pi_{est,sc,cc}\left(\sigma_{(ds=\text{“us-south”}\wedge qc>18)}(MDP)\right).$$

For instance, the query for which EXGB produced a q-error greater than 512, $qc > 18$, falls in the selectivity class 25. Since the cardinality class for *us-south* is 27, using Equation 4, we can approximate the true cardinality of the query in the range $[4, 8)$.

The multidimensional profile structure introduced in (5) embodies the core concept of MDE. While these profile tuples offer a wealth of information, drawing definitive conclusions from them remains challenging. However, this structure supports various forms of aggregation through slice-and-dice operations. As an example, we can replicate PDE-like evaluation results by grouping *sc* values into selectivity categories, which we can now define with desired granularity, and aggregating *qc* values into specific percentiles across *ds*, *est*, and each selectivity category. The remainder of this section presents examples of analytical investigations on MDP.

8.2 Accuracy Heatmaps

We aggregate the MDP tuples by summing up the frequencies per *(est, cc, sc, qc)* combination. Having grouped our datasets into cardinality classes, we made our first move towards surpassing the main deficiency of PDE. This yields per-estimator evaluation tuples in the form of *(cc, sc, qc, cnt)*, which provide an overview on the estimators' accuracy. While informative, these tuples remain challenging to present or draw a conclusion from. To address this challenge, we

apply dimensionality reduction. We use the log-transformed upper bound of the approximation interval presented for true cardinalities in Equation 4 and define *log_card* as a new dimension to combine *cc* and *sc* as follows:

$$\text{log\_card} := cc - sc + 1. \tag{6}$$

Additionally, for better visualization, we define:

$$\text{log\_cnt} := \log_2(cnt). \tag{7}$$

For each estimator, we use the aggregated *(log_card, qc, log_cnt)* tuples to plot the accuracy heatmaps, Figure 2, where each cell corresponds to a *(log_card, qc)* pair, with its color indicating *log_cnt* value. Notably, in each plot, large *q-errors* for queries with high true cardinalities are identifiable in the top-right corner, where yellow or light green cells indicate a high frequency of these undesirable outcomes. Observably, QTS-2 and GXTree exhibit the least expansion in this area. Although these heatmaps are highly informative, they pose challenges for pairwise comparisons of estimators. For instance, it remains challenging to determine whether DeepDB outperforms Sampling. In Subsection 8.3, we propose a systematic way to rank estimators based on their overall accuracy results.

8.3 From MDP to Ranking the Estimators

We argue that a properly defined weighted sum of *qc* values can serve as a meaningful metric for comparing the accuracy of estimators. Here, we outline the intuition for this approach. Our experiments evaluate each estimator on datasets of diverse cardinality classes. Therefore, queries with identical *sc* values can have very different true cardinalities, which influences the impact of a specific *qc*. To account for these variations, we slice the evaluation results per (cc, sc) pairs. This ensures that the *qc* values within each slice correspond to true cardinalities in a range defined by Equation 4 and are comparable across different datasets. Consequently, the accuracy of estimators can be compared by analyzing the frequency distribution of *qc* values per slice. Since the number of queries per (cc, sc) pair is consistent across estimators, an estimator with a higher frequency mass at larger *qc* values can be recognized as the less accurate one with respect to the corresponding slice. To quantify this undesirable mass, we use a weighted sum of *qc* frequencies, $\Sigma(f(qc) \cdot cnt)$, where $f(qc)$ is a strictly increasing function penalizing larger *qc* vales and *cnt* is the frequency of the corresponding *qc*. A relevant question is which function to choose as f. This depends on the analyst's preference: a polynomial or exponential function imposes stricter penalties on high *qc* values, favoring the search for an estimator with the lowest worst-case errors, while a linear function applies a milder penalty, preventing extreme errors from dominating the evaluation. Having selected the penalty function f, we can determine the superior estimator for each (cc, sc) pair. However, the known problem of contradictory results may arise where an estimator is superior for one slice, while the other is preferable elsewhere. In contrast to PDE, we can resolve this contradiction by imposing higher penalties on the slices with larger true cardinalities while aggregating the weighted sum over slices. In the following, we present an implementation of this approach with three recommended weighting mechanisms, w_i, that account for both penalizing higher *qc* values and emphasizing slices corresponding to larger true cardinalities.

We use the MDP tuples, with the relational schema introduced in (5), and a weighted sum to derive a *penalty score* per estimator. For an estimator *Est*, a tuple t in MDP, and a weight function $w(t)$,

$$\text{penalty\_score}(Est) := \sum_{t \in \sigma_{\text{est}=Est}(\text{MDP})} w(t) \cdot t.cnt. \tag{8}$$

Some recommendations for the weighting mechanism are:

$$w_1(t) := (t.cc - t.sc + 1) \cdot \sqrt{2}^{t.qc}$$

$$w_2(t) := w_{sc}(t) \cdot \sqrt{2}^{t.qc} \text{ where } w_{sc}(t) := \begin{cases} 0.4 & t.cc - t.sc < 2 \\ 0.8 & 1 < t.cc - t.sc < 4 \\ 1 & \text{otherwise} \end{cases}$$

$$w_3(t) := (t.cc - t.sc + 1) \cdot t.qc,$$

where w_1 and w_3 employ the log-transformed upper bound of the true cardinalities, defined in Equation 4, as a strictly positive coefficient to emphasize the slices with larger true cardinalities, whereas w_2 uses a piecewise constant function for this purpose. Regarding the penalty function $f(qc)$, w_1 and w_2 demonstrate the choice of an exponential penalty in contrast to the identity function in w_3. An intuitive choice for the base of the exponent, in w_1 and w_2, is to match the logarithm base used in our discretization of q-errors, allowing the magnitude of large q-errors to be reflected in the penalty score. The resulting penalty scores can be used for establishing a precision-based ranking of the estimators.

8.3.1 Case Study. To further elaborate on the data processing in MDE and our proposed ranking mechanism, Figure 3 presents the data processing pipeline for four example tuples. These tuples are selected from the evaluation results of PGDef and PGB on two OpenStreetMap datasets, namely *alps* and *us-south*. Notably, we deliberately selected these tuples as our examples to highlight the varying accuracy of PostgreSQL's cardinality estimator under different configurations. We hope this further encourages the community to avoid evaluating this estimator without accounting for its configuration potential. While the data structure introduced for the raw evaluation results in Subsection 8.1 is sufficient for the pipeline to work, here we also include *TrueCard* and *EstCard* for better illustration. By following the process on the second tuple, identifiable by its red color, we observe that in the first step, the *cc*, *sc*, and *qc* values are computed, determining the MDP tuple in which the red tuple should be aggregated. Consequently, it contributes a count of one to the frequency of 204 for the tuples sharing the same *cc*, *sc*, and *qc* values for the dataset *us-south* and the PGB estimator. In the next step, the MDP tuples are aggregated and grouped by (est, cc, sc, qc) combinations irrespective of the name of the dataset. As a result, our red tuple is aggregated into the red-highlighted tuple, contributing to the total frequency of 20,930 for this group. The aggregated evaluation results can be used for further analysis, one such example being ranking the estimators according to the penalty score accumulated for each. By comparing the first and third tuples in the raw evaluation results, identifiable by their blue and purple colors, we see that they have similar q-errors both falling into the *qc* of one, however, they have significantly different true cardinalities. As a result, the blue tuple exhibits a misestimate exceeding 1.45×10^7, whereas the purple tuple's misestimate is a few thousand. Following the corresponding colors in the contribution

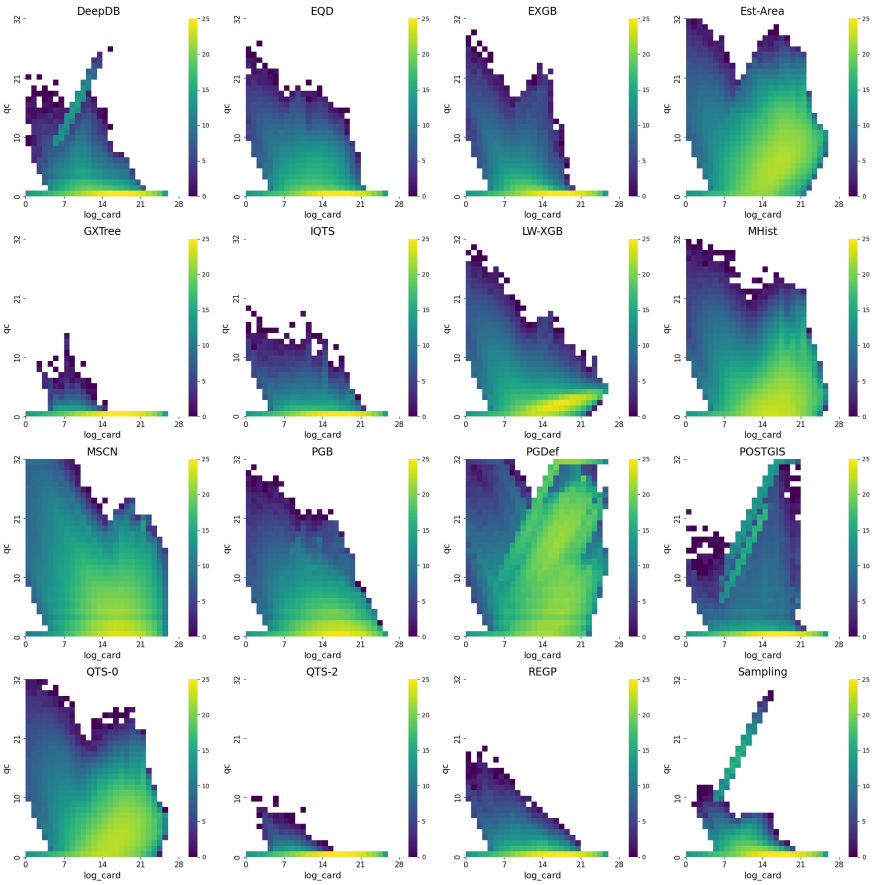

Figure 2: Per-estimator accuracy heatmaps

of each tuple in the penalty score, we observe that w_1 and w_3 take this difference well into consideration. By further comparing the contributions of these tuples in the penalty score under the three weighting mechanisms, we see that w_1 is the strictest in penalizing high errors and emphasizing larger true cardinalities.

8.3.2 Estimators' Ranking. Table 5 presents the rankings according to the penalty scores derived from the evaluation results for the two subsets introduced in Section 3, along with the rankings of the top five estimators derived from experiments on the full set of datasets in our repository. The primary objective of expanding experiments to 18,020 datasets is to assess the robustness of our ranking results across a broader set of datasets. Focusing on the top five estimators is motivated by their greater relevance to decision makers, who prioritize leading performance. Regarding the ranking results, we observe that QTS-2 and GXTree consistently rank as the most accurate estimators. EXGB, followed by DeepDB, achieve better positions compared to other learned methods. The three PostgreSQL variants exhibit notable results: PGDef ranks at the bottom, PGB, however, is mid-range and surprisingly outperforms both MSCN and LW-XGB. POSTGIS does not outperform PGB, except when ranked on the larger dataset subset and with w_3. Particularly, when greater penalties are applied to larger q-errors, POSTGIS ranks lower than PGB, which is unexpected given its tailored features for geospatial data. By extending the experimental datasets from the five datasets in S_1 to the 276 datasets in S_2, we observe the swap between the first and second positions for GXTree and QTS-2, along with quite notable shifts in the positions of REGP and DeepDB for w_3. The overall ranking remains mostly stable, with the top estimators largely remaining in the top tier and those in the middle or bottom exhibiting similar standing. This suggests that the K-Means selection was quite effective in choosing a representative subset of only five datasets, capturing the diversity of our repository. However, conducting comprehensive experiments on a larger number of datasets is preferable whenever feasible. The advantage of exploiting a sufficiently large set of experimental datasets is further confirmed by the consistent ranking of the top estimators across different weighting mechanisms for the subset with 276 datasets, and the robustness of the ranking results when utilizing an even larger set comprising 18,020 datasets.

8.3.3 Robustness Against the Discretization Granularity. To examine the robustness of the ranking against the discretization granularity, we materialize the raw evaluation results of the top five estimators on the 276 datasets in S_2 and apply discretization using different logarithm bases. For cardinality and selectivity, we use bases from

$\{\sqrt{2}, 2, 4, 10\}$, and for q-error, bases from $\{2^{\frac{1}{8}}, 2^{\frac{1}{4}}, \sqrt{2}, 2, 4\}$. We remove the cap for *qc* values, apply our three weighting mechanisms to derive the penalty scores, and rank the estimators accordingly. The key finding is that the ranking is insensitive to the discretization of cardinality and selectivity. For q-error discretization, there is a swap between the first and second places for w_3 with bases larger than $\sqrt{2}$, while the rest remains stable.

8.3.4 Sufficiency of Our Experimental Scope. MDE requires as input the relation cardinalities, the estimated selectivities, and the resulting q-errors. Therefore, its evaluation requires a diverse range of these inputs, regardless of the complexity of the workload used to generate them. Our experiments incorporate 18,020 datasets exhibiting varied statistical properties, alongside 16 estimators ranging from naïve to sophisticated methods. This produces a broad spectrum of q-error distributions, as illustrated in Figure 2, thereby ensuring sufficient input diversity for a robust evaluation of MDE.

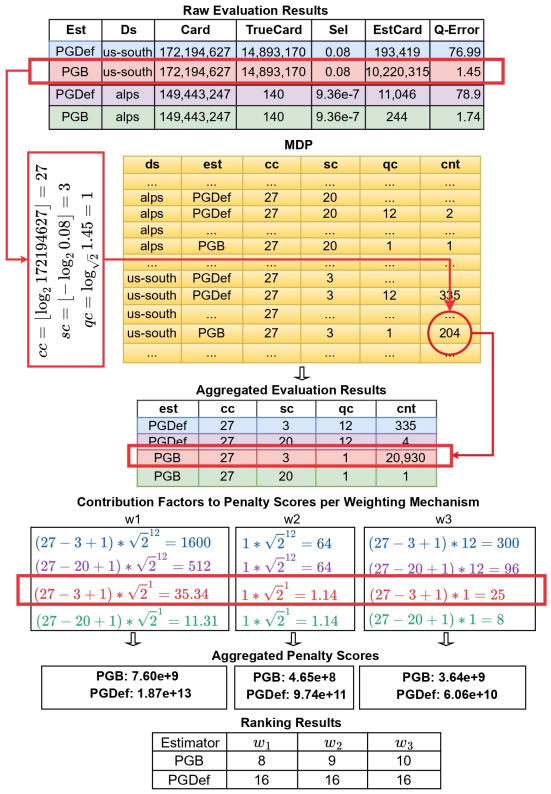

Figure 3: MDE – Data processing pipeline

9 CONCLUSION AND FUTURE WORK

We developed a multidimensional evaluation framework (MDE) that facilitates the aggregation of evaluation results across any number of datasets. This capability eliminates the need for reliance on per-dataset evaluations, which have so far hindered conducting experiments involving a sufficient number of datasets. MDE facilitates pairwise comparisons between estimators and drawing conclusions in the presence of seemingly contradictory results. The customizable ranking mechanism of MDE enables the imposition of stricter penalties on more undesirable misestimation cases, thereby providing rankings that align with the analyst's primary objectives.

We performed experiments on the accuracy of cardinality estimators from learned and traditional methods on 2-D geographical datasets. The narrow yet purpose-driven scope of this study in contrast to the comprehensiveness of the experiments challenges the prevalent but flawed practice of drawing sweeping conclusions about the accuracy of cardinality estimators from a limited set of experiments. Regarding the comparison of the estimators, our experiments do not confirm the general conclusion of previous studies about the superiority of learned over traditional cardinality estimators for single relation range queries in static environments. According to our evaluation results, QTS-2 and GXTree are the most accurate cardinality estimators for 2-D geographical datasets in this setup. Among learned estimators, EXGB demonstrates a superior accuracy, followed by DeepDB. Remember that the storage consumption of DeepDB is influenced by the sample size, which in turn scales with the dataset size and can reach megabytes beyond our budget limits. In contrast, EXGB exhibits a much more manageable storage footprint. Moreover, we highlighted that conducting comparisons with traditional estimators without considering their tuning knobs can lead to misleading results. One surprising finding in this regard is that although PostgreSQL is a common baseline heavily criticized in benchmarking studies, a configured version of it, PGB, outperforms some of the learned estimators. Another interesting finding is that POSTGIS, the extension of PostgreSQL for geospatial data, does not always outperform PGB, although it is expected to be tailored to this experimental environment. Finally, we believe MDE can pave the way for future benchmarking studies to conduct extensive experiments with well-defined scopes across various scenarios, including higher-dimensional datasets, more complex workloads involving joins, and a broad range of estimators suitable for each scenario.

Table 5: Ranking results with three different weightings

	5 datasets			276 datasets			18,020 datasets		
Estimator	w_1	w_2	w_3	w_1	w_2	w_3	w_1	w_2	w_3
QTS-2	2	2	2	1	1	1	1	1	1
GXTree	1	1	1	2	2	2	2	2	2
IQTS	3	3	3	3	3	3	3	3	3
REGP	5	5	7	4	4	4	4	4	4
EXGB	4	4	5	5	5	5	5	5	5
EQD	7	7	6	6	6	7			
DeepDB	6	6	4	7	7	6			
PGB	8	8	9	8	9	10			
Sampling	10	11	8	9	10	8			
LW-XGB	9	9	10	10	8	11			
MHist	11	10	11	11	11	12			
QTS-0	12	12	14	12	12	14			
MSCN	13	13	12	13	13	13			
Est-Area	14	14	15	14	14	15			
POSTGIS	15	15	13	15	15	9			
PGDef	16	16	16	16	16	16			

REFERENCES

[1] Andreas Kipf. 2019. andreaskipf/learnedcardinalities. https://github.com/andreaskipf/learnedcardinalities Retrieved September 3, 2024.

[2] F. Buccafurri, F. Furfaro, G. M. Mazzeo, and D. Sacca. 2003. A quad-tree based multiresolution approach for two-dimensional summary data. In *Proceedings of the International Conference on Scientific and Statistical Database Management (SSDBM)*.

[3] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (San Francisco, California, USA). ACM, 785–794. https://doi.org/10.1145/2939672.2939785

[4] Chris Ding and Xiaofeng He. 2004. K-Means Clustering via Principal Component Analysis. In *Proceedings of the Twenty-First International Conference on Machine Learning* (Banff, Alberta, Canada) *(ICML '04)*. Association for Computing Machinery, New York, NY, USA, 29. https://doi.org/10.1145/1015330.1015408

[5] Anshuman Dutt, Chi Wang, Azade Nazi, Srikanth Kandula, Vivek Narasayya, and Surajit Chaudhuri. 2019. Selectivity estimation for range predicates using lightweight models. *Proceedings of the VLDB Endowment* 12, 9 (May 2019), 1044–1057. https://doi.org/10.14778/3329772.3329780

[6] Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. 2022. Why do tree-based models still outperform deep learning on tabular data? arXiv:2207.08815 [cs.LG]

[7] Antonin Guttman. 1984. R-trees: A Dynamic Index Structure for Spatial Searching. In *Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data*. ACM, 47–57. https://doi.org/10.1145/971697.602266

[8] Yuxing Han, Ziniu Wu, Peizhi Wu, Rong Zhu, Jingyi Yang, Liang Wei Tan, Kai Zeng, Gao Cong, Yanzhao Qin, Andreas Pfadler, Zhengping Qian, Jingren Zhou, Jiangneng Li, and Bin Cui. 2021. Cardinality Estimation in DBMS: A Comprehensive Benchmark Evaluation. *Proc. VLDB Endow.* 15, 4 (Dec. 2021), 752–765. https://doi.org/10.14778/3503585.3503586

[9] B. Hilprecht, A. Schmidt, M. Kulessa, A. Molina, K. Kersting, and C. Binnig. 2020. DeepDB: Learn from Data, Not from Queries! *Proc. VLDB Endow.* 13, 7 (March 2020), 992–1005. https://doi.org/10.14778/3384345.3384349

[10] Fisnik Kastrati and Guido Moerkotte. 2016. Optimization of Conjunctive Predicates for Main Memory Column Stores. *Proc. VLDB Endow.* 9, 1 (Aug. 2016), 1125–1136. https://doi.org/10.14778/2994509.2994529

[11] Kyoungmin Kim, Sangoh Lee, Injung Kim, and Wook-Shin Han. 2024. ASM: Harmonizing Autoregressive Model, Sampling, and Multi-dimensional Statistics Merging for Cardinality Estimation. *Proceedings of the ACM on Management of Data* 2, 1 (March 2024), 1–27. https://doi.org/10.1145/3639300

[12] Andreas Kipf, Thomas Kipf, Bernhard Radke, Viktor Leis, Peter Boncz, and Alfons Kemper. 2018. Learned Cardinalities: Estimating Correlated Joins with Deep Learning. arXiv:1809.00677 [cs]. http://arxiv.org/abs/1809.00677

[13] Stephen Kokoska and Daniel Zwillinger. 2014. *CRC Standard Probability and Statistics Tables and Formulae, Student Edition*. CRC Press, Boca Raton. https://doi.org/10.1201/b16923

[14] Data Management Lab. 2020. DeepDB: Deep Learning for Query Processing. https://github.com/DataManagementLab/deepdb-public. Accessed: 2024-09-03.

[15] Kukjin Lee, Anshuman Dutt, Vivek Narasayya, and Surajit Chaudhuri. 2023. Analyzing the Impact of Cardinality Estimation on Execution Plans in Microsoft SQL Server. *Proceedings of the VLDB Endowment* 16, 11 (2023), 2871–2883. https://doi.org/10.14778/3611479.3611494

[16] Viktor Leis, Andrey Gubichev, Atanas Mirchev, Peter Boncz, Alfons Kemper, and Thomas Neumann. 2015. How Good Are Query Optimizers, Really? *Proceedings of the VLDB Endowment* 9, 3 (Nov. 2015), 204–215. https://doi.org/10.14778/2850583.2850594

[17] Yingze Li, Xianglong Liu, Hongzhi Wang, Kaixin Zhang, and Zixuan Wang. 2024. Updateable Data-Driven Cardinality Estimator with Bounded Q-error. *arXiv preprint* arXiv:2408.17209 (2024). https://doi.org/10.48550/arXiv.2408.17209

[18] Yingze Li, Hongzhi Wang, and Xianglong Liu. 2024. One Seed, Two Birds: A Unified Learned Structure for Exact and Approximate Counting. *Proc. ACM Manag. Data* 2, 1 (March 2024), 1–26. https://doi.org/10.1145/3639270

[19] Yuming Lin, Zejun Xu, Yinghao Zhang, You Li, and Jingwei Zhang. 2023. Cardinality estimation with smoothing autoregressive models. *World Wide Web* 26, 5 (Sept. 2023), 3441–3461. https://doi.org/10.1007/s11280-023-01195-7

[20] T. H. Merrett and Ekow J. Otoo. 1979. Distribution models of relations. In *Proceedings of the fifth international conference on Very Large Data Bases - Volume 5* (Rio de Janeiro, Brazil) *(VLDB '79)*. VLDB Endowment, 418–425. https://doi.org/10.5555/602876.602909

[21] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. 2018. Mixed Precision Training. arXiv:1710.03740 [cs.AI].

[22] Guido Moerkotte. 2024. Query Compiler. https://pi3.informatik.uni-mannheim.de/~moer/querycompiler.pdf. Accessed: 2024-07-16.

[23] Guido Moerkotte, Norman May, and Anja Boehm. 2017. Methods and Systems for Estimating the Number of Points in Two-Dimensional Data. https://patents.google.com/patent/US20170177663A1/en US20170177663A1, June 22, 2017. (accessed 2024-07-16).

[24] Guido Moerkotte, Thomas Neumann, and Gabriele Steidl. 2009. Preventing bad plans by bounding the impact of cardinality estimation errors. *Proc. VLDB Endow.* 2, 1 (2009), 982–993. https://doi.org/10.14778/1687627.1687738

[25] M. Muralikrishna and David J. DeWitt. 1988. Equi-depth multidimensional histograms. In *ACM SIGMOD Record*, Vol. 17. 28–36. https://doi.org/10.1145/971701.50205

[26] naru-project. 2019. naru-project/naru. https://github.com/naru-project/naru Retrieved September 3, 2024.

[27] OpenStreetMap Foundation. n.d.. *OpenStreetMap*. https://www.openstreetmap.org Accessed: 2024-08-31.

[28] Hoifung Poon and Pedro Domingos. 2011. Sum-Product Networks for Deep Learning. In *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence* (Barcelona, Spain) *(UAI'11)*. AUAI Press, Arlington, Virginia, USA, 337–346. https://dl.acm.org/doi/10.5555/3020548.3020596

[29] Viswanath Poosala and Yannis E Ioannidis. 1988. Selectivity Estimation without the Attribute Value Independence Assumption. *ACM SIGMOD Record* 17, 3 (June 1988), 28–36. https://doi.org/10.1145/971701.50205

[30] PostgreSQL Global Development Group. 2023. *Statistics Used by the Planner*. https://www.postgresql.org/docs/15/planner-stats.html Accessed: 2024-06-11.

[31] Ji Sun, Jintao Zhang, Zhaoyan Sun, Guoliang Li, and Nan Tang. 2021. Learned Cardinality Estimation: A Design Space Exploration and a Comparative Evaluation. *Proceedings of the VLDB Endowment* 15, 1 (2021), 85–97. https://doi.org/10.14778/3485450.3485459

[32] U.S. Census Bureau. 2023. *TIGER/Line Shapefiles and TIGER/Line Geodatabase*. https://www.census.gov/geographies/mapping-files/time-series/geo/tiger-geodatabase-file.html Accessed: 2024-08-31.

[33] U.S. Geological Survey. 2013. *Earthquake Data Feed*. https://earthquake.usgs.gov/earthquakes/feed/v1.0/csv.php Accessed: 2024-08-31.

[34] Xiaoying Wang, Changbo Qu, Weiyuan Wu, Jiannan Wang, and Qingqing Zhou. 2021. Are We Ready For Learned Cardinality Estimation? *Proceedings of the VLDB Endowment* 14, 9 (May 2021), 1640–1654. https://doi.org/10.14778/3461535.3461552 arXiv:2012.06743 [cs].

[35] L. Woltmann, C. Hartmann, M. Thiele, D. Habich, and W. Lehner. 2019. Cardinality Estimation with Local Deep Learning Models. In *Proceedings of the Second International Workshop on Exploiting Artificial Intelligence Techniques for Data Management*. ACM, Amsterdam, Netherlands, 1–8. https://doi.org/10.1145/3329859.3329875

[36] Zongheng Yang, Eric Liang, Amog Kamsetty, Chenggang Wu, Yan Duan, Xi Chen, Pieter Abbeel, Joseph M. Hellerstein, Sanjay Krishnan, and Ion Stoica. 2019. Deep Unsupervised Cardinality Estimation. *Proc. VLDB Endow.* 13, 3 (Nov. 2019), 279–292. https://doi.org/10.14778/3368289.3368294