Efficient Selectivity Estimation for Range Predicates using Machine Learning Lightweight Models

Achala Shashidhara Pandit
Department of Computer Science and Engineering
University of California, Riverside
Riverside, CA
apand048@ucr.edu

Farnaz Mozhgani
Department of Computer Science and Engineering
University of California, Riverside
Riverside, CA
fmozh001@ucr.edu

Anjana Venkatesha Murthy
Department of Computer Science and Engineering
University of California, Riverside
Riverside, CA
avenk029@ucr.edu

Radhika Khandelwal
Department of Computer Science and Engineering
University of California, Riverside
Riverside, CA
rkhan053@ucr.edu

***Index Terms*—Query Optimization, Selectivity Estimation, XGBoost, Neural Networks, Random Forest**

I. Introduction

In the evolving landscape of database management systems (DBMS), query optimization stands as a critical challenge, directly influencing the efficiency and performance of data retrieval processes. Traditional query optimization techniques rely on statistical models to estimate query cardinality, an approach that, while useful, often falls short in accuracy due to its inherent assumptions of data uniformity and independence. These limitations can lead to suboptimal query execution plans, significantly impacting system performance.

The advent of machine learning (ML) technologies presents a novel avenue to enhance query optimization. Machine learning models, with their ability to learn complex patterns and dependencies from data, offer a promising alternative to traditional statistical approaches for query cardinality and selectivity estimation. Specifically, deep learning and ensemble learning methods have shown significant potential in capturing the nuanced correlations and multi-dimensional data distributions that traditional methods struggle with.

This project delves into the application of machine learning techniques for database query optimization, with a focus on cardinality and selectivity estimation. It builds upon the methodologies presented in two seminal papers: "Cardinality Estimation with Local Deep Learning Models" and "Selectivity Estimation for Range Predicates using Lightweight Models" . The former introduces a local deep learning model approach for cardinality estimation, proposing a shift from global models that consider the entire database schema to local models that focus on specific sub-parts of the schema. The latter explores the use of neural networks and tree-based ensembles for selectivity estimation, presenting design choices that enhance model performance without increasing complexity.

Our project aims to implement the technique presented in the MSR paper using the Forest evaluation dataset mentioned in the paper, subsequently integrating and comparing it with the technique from the LM paper to assess performance differences. Through a rigorous experimental evaluation, following the parameters and measures used in the MSR paper, this study seeks to offer insights into the efficacy of machine learning tools in improving database query optimization processes.

By bridging the gap between traditional statistical estimation methods and modern machine learning approaches, this project aspires to contribute to the development of more efficient, accurate query optimization techniques, paving the way for advanced DBMS capabilities that can meet the demands of complex data environments.

II. Problem Description

A. Problem Definition

Selectivity estimation is a critical task in query optimization, aiming to predict the fraction of tuples in a relation that satisfy a given predicate. It's essential for the query optimizer to make informed decisions about the most efficient query execution plan. In this context, selectivity estimation is formulated as a regression problem, where the goal is to predict the selectivity (a continuous value) based on various features of the query and dataset.

Our work mainly focuses on:

- Selectivity estimation for query optimization as a regression problem using three lightweight models XGBoost, Random Forest and Neural Network on 'Forest' dataset as outlined in the MSR paper
- Implementation of data generation and vectorization technique presented in the LM paper on the evaluation dataset of MSR paper using Neural Network.
- Comparing the results of the above two methods on the same evaluation dataset. The results obtained from

both methods are compared and analyzed to gain insights into the effectiveness and efficiency of each approach in selectivity estimation tasks.

B. Data Tables

In this project, we have employed the "Forest" dataset, which was originally used in the MSR paper for the implementation and evaluation of the cardinality estimation technique based on local deep learning models. The Forest dataset is sourced from the UCI Machine Learning Repository and is officially named the "Covertype" dataset. It contains detailed information on forest cover types determined from cartographic variables, making it highly relevant for testing the efficacy of machine learning models in query optimization, specifically in the realm of cardinality and selectivity estimation.

The dataset comprises 581,012 instances with a total of 54 attributes. These attributes include various geographical features such as elevation, aspect, slope, horizontal distance to hydrology, vertical distance to hydrology, horizontal distance to roadways, and more.

For the purposes of this project, the Forest dataset was utilized in two significant aspects:

- Model Training and Evaluation: Consistent with the methodology outlined in the MSR paper, the dataset served as the basis for training the local deep learning model.
- Technique Comparison: The same dataset was also used to implement vectorization technique presented in LM paper. This consistency in dataset usage allowed for a direct, fair comparison between the different models' performance, specifically focusing on their accuracy in selectivity estimation.

Based on the Forest dataset, query predicate values are generated for first ten attributes. These values are utilized to create a training dataset, which serves as input for machine learning models tasked with predicting selectivity values. Further elaboration on this process is provided in the following section.

C. Vectorization and Dataset Generation

We have developed two distinct approaches to generate query data involving range predicates on multiple attributes, resulting in a database comprising 20,000 queries split into 80% for training and 20% for testing in each scenario. The first method aligns with the approach described in the MSR paper. Each query sample consists of 2d elements, where 'd' denotes the number of numerical attributes in the dataset. These elements are crafted with upper and lower limits for each numerical attribute. To establish these boundaries, we start by identifying random centers using a uniform distribution and data distribution across the attribute's domain. Subsequently, the lower and upper bounds are determined by subtracting and adding a width, respectively, from the center. For uniformly distributed centers, the widths are determined using a uniform distribution, whereas data-distributed centers employ an exponential distribution. Furthermore, we compute such range values for all combinations of column subsets.

The second method is based on the approach outlined in the LM paper. For generating query predicates, we incorporate two types of encoding: operator encoding and value encoding. The operator encoding vectorizes the selection of operators for each query predicate using one-hot encoding, requiring a vector of length 3 to represent the different operators $(>, =, <)$. Next, to generate query predicate values, we employ a uniform random distribution over the data tuples. This numeric value, situated on the right-hand side of the operator, represents the query predicate's value. The value encoding is a single floating-point number that signifies the value itself. Finally our predicate values are normalized with min max scaler to get values in the range 0 to 1 for training with neural networks.

Once the dataset is prepared, it serves as input for the SQL queries, i.e., utilizing the dataset to generate conditions for each query. This process involves translating each row's range conditions into WHERE clauses, which are then used to calculate the proportion of records in the table that satisfy given conditions. The outcome of this procedure is a series of proportions, each representing the true selectivity measure for a given set of query predicates derived from the dataset. These selectivity measures serve as data for the training of our lightweight machine learning models.

D. Regression with Lightweight Models

Conventional methods for selectivity estimation of predicates often rely on heuristic assumptions, resulting in notable estimation errors. The study introduces regression techniques employing neural networks and tree-based ensembles, alongside novel design strategies such as regression label transformation and feature engineering, aimed at achieving precise and rapid selectivity estimates.
The research investigates the use of regression techniques for accurate selectivity estimation of multi-dimensional range predicates, considering practical constraints like estimation time and memory footprint. The study focuses on neural networks and tree-based ensembles due to their ability to learn complex non-linear functions and the availability of highly optimized libraries and resources to perform such complex tasks efficiently. Furthermore, we have applied a logarithmic transformation (base 2) to the true selectivity labels to evaluate its efficacy in reducing neural network and Tree ensembles error loss and achieving overall higher performance. Note that log-transformed predictions are generated through a log-linear function where each feature value contributes as a multiplicative factor, contrasting with the additive form seen in linear functions. This characteristic allows even simple-form functions in each local region to accommodate larger selectivity variations.

We aim to train a regression model M using a set S of labeled queries to estimate selectivity for conjunctive range

queries on table T. The goal is for model M to produce estimated selectivity est(q) close to the actual selectivity act(q) for any query q on T. It's important to note that our problem definition doesn't assume any specific source or distribution of queries in set S, but we expect M to provide accurate estimates for queries well represented in S.

Our approach involves treating the dataset as input table T, which contains 20 numerical attributes used as range features during the training of regression models. Additionally, we use a separate table containing the actual selectivity labels of the queries as a source of regression labels. Our research showcases various types of regression models discussed in the paper that effectively perform this selectivity estimation task.

Neural Network

The paper discusses experimenting with different neural network models by varying the number of hidden layers and neurons to improve estimation accuracy. However, since the paper evaluates these models across four distinct datasets with varying feature characteristics, it's crucial to explore and conduct sufficient experiments to identify the optimal configuration for each dataset. For the Forest dataset specifically, two types of experiments were conducted. We have developed a deep feedforward neural network with fully connected layers. Its primary objective is to train itself using a set of provided feature inputs. The dataset is typically divided into two parts: the training set, which the model uses to learn the underlying patterns and relationships in the data, and the test set, which remains unseen during training and is used solely for evaluating the model's performance.

During the training process, the neural network adjusts its internal parameters (weights and biases) based on the training set's input features and corresponding output labels (selectivity values in this case). The goal is to minimize the difference between the predicted selectivity values and the actual selectivity values in the training data.

Once the training is complete, the model is evaluated using the test set. This evaluation involves feeding the test set's input features into the trained model and comparing the predicted selectivity values with the actual selectivity values for these unseen data points. The performance metrics obtained from this evaluation, such as accuracy or error rates, provide insights into how well the neural network generalizes to new, unseen data and its effectiveness in predicting selectivity values for such data.

Tree-based ensembles

The experiment done using Tree-based ensembles regression model with the significant focus on XGBoost, a popular implementation of gradient boosting with decision trees. In tree-based ensembles like XGBoost, each learned tree in the ensemble produces a regression value using a subset of features from the input feature vector. The final prediction of the ensemble is typically obtained by taking a weighted sum of these individual tree predictions.

However, a challenge arises when the summed values from different trees are of significantly different magnitudes. In such cases, the predictions with larger magnitudes can heavily dominate the final prediction, leading to insensitivity to predictions with smaller magnitudes. This phenomenon occurs because many predictions end up being determined by only a small number of trees, resulting in nearly identical values for a large portion of the predictions.

To address this issue and ensure a more efficient use of the model capacity, the experiment applies a log transformation to the selectivity labels. This transformation effectively aggregates the predictions from all the trees using a log-linear model, reducing the dominance of predictions with large magnitudes. By doing so, each individual tree can actively contribute to the final prediction across various combinations of leaf nodes, leading to a more balanced and effective model with a higher estimation rate and reduced test loss. In this project, two types of tree ensemble models were used: XGBoost and Random Forest. The evaluation metrics used include R^2, Root Mean Squared Error (RMSE), Mean Squared Error (MSE), and Mean Absolute Error (MAE).

XGBoost VS Random Forest

Random Forest:

Each decision tree in the Random Forest is trained independently on a random subset of the training data using a technique called bootstrap aggregation (bagging). The predictions from individual trees are then combined through an unweighted sum or a majority voting scheme to make the final prediction.

XGBoost:

Each weak learner (tree) in XGBoost is trained to address the errors made by the preceding trees. The predictions from each tree are weighted according to their performance during training. The model's complexity is controlled by parameters such as the number of trees (t) and the number of leaves (v). By optimizing the number of trees and leaves, XGBoost can explore a broader range of ways to partition the query space, leading to improved predictive selections.

III. Experiments

The experiment involved using a log transform with 2 hidden layers in the neural network, while the second experiment used a neural network without the log transform but still with 2 hidden layers. These experiments were aimed at understanding how different model configurations perform on this particular dataset.

The model that exhibited a low error loss, signifying a high accuracy in estimating the actual selectivity labels, consisted of 2 hidden layers, 2 types of activation functions, MSE, and default model parameters which is the number

of input features and number of nuerons in each of the hidden layers. The first hidden layer was composed of 128 neurons or perceptrons with a Rectified Linear Unit (ReLU) activation function, while the output layer utilized a linear activation function. Activation functions play a critical role in introducing non-linearities into neural networks, enabling them to learn intricate patterns and relationships within the data. These functions are fundamental in deep neural network $(l > 1)$ as they facilitate the capture and representation of complex features present in the input data.

During the training phase, the ADAM optimizer was utilized with a learning rate of 0.01, enabling efficient gradient-based optimization. The training process spanned 500 epochs, with a batch size of 50 chosen to partition the training data into smaller subsets. This batch-wise approach facilitates more effective learning as the model iterates through the epochs. The primary objective was to minimize the Mean Squared Error (MSE) loss, which serves as a measure of the model's predictive accuracy. Additionally, the same pipeline was applied to log-transformed labels.

The XGBoost training experiment involved two variations: one using log-transformed models and another using regular models. For each variation, two different learning rates were experimented with $lr = 0.5$ and $lr = 0.1$ In the first set of experiments, default parameters were used for both log-transformed and regular models to establish a baseline performance. In the second set of experiments, the number of trees and leaves were adjusted to evaluate their impact on model performance. Specifically, the parameters for the XGBoost models were tuned by changing the number of trees (n-estimators) and the maximum number of leaves (max-leaves). For the Random Forest model, default parameters were used, and the selectivity labels were log-transformed to assess the model's performance under this transformation. Overall, these experiments aimed to compare the performance of XGBoost and Random Forest models under different configurations, including the use of log-transformed data, varying learning rates, and tuning tree-related hyperparameters to optimize predictive accuracy. Experiments were conducted utilizing various neural network architectures on the training dataset generated through the vectorization method proposed in the LM paper. The objective was to compare the effectiveness of different network depths and widths in minimizing loss during testing. Initially, the model selected mirrored the architecture that demonstrated superior performance in the MSR paper, omitting log transformation, featuring two hidden layers comprising 128 and 64 neurons, respectively. Notably, this configuration yielded a notably low test loss, a detailed analysis of which is provided in the evaluations section. In an attempt to mitigate training loss, further exploration involved increasing the number of layers, which unfortunately resulted in a sharp escalation of test loss across configurations with varying neuron counts per layer. Consequently, this

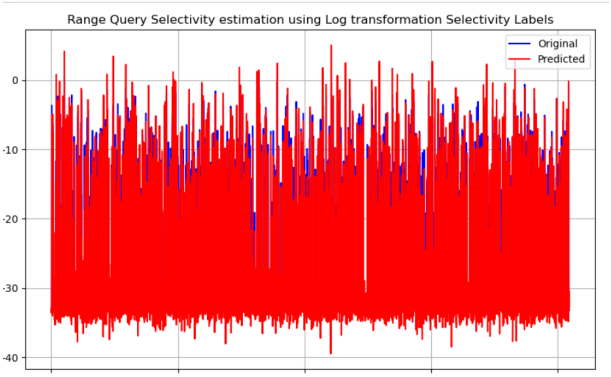

Fig. 1. XGBoost using log transformation selectivity labels

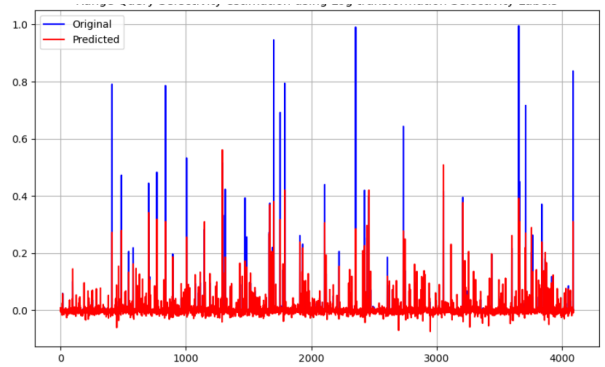

Fig. 2. XGBoost using original selectivity labels

avenue was not pursued extensively. Conversely, narrowing the width of the network proved beneficial in reducing test loss and identifying more effective models. Notably, two additional models with two hidden layers, featuring 64 and 32 neurons, and 32 and 16 neurons, respectively, displayed promising outcomes. Furthermore, noteworthy performance was observed with a single hidden layer comprising 64 neurons. An important observation from these experiments is the efficiency of training, with all configurations completing training in less than 50 epochs as seen in Figure 8. This efficiency underscores the potential utility of employing compact neural networks for query optimization, as their lightweight nature coupled with rapid training makes them viable candidates for deployment.

IV. Evaluations

After implementing neural network and Tree ensembles models in the MSR paper with varying configurations such as different numbers of hidden layers and neurons, applying log transform to selectivity labels, and experimenting with different parameters for tree ensembles (such as increasing the number of trees and leaves), we achieved high accuracy results as expected. The best-performing model overall was the XGBoost model with log-transformed labels. Interestingly, Random Forest performed similarly to XGBoost without

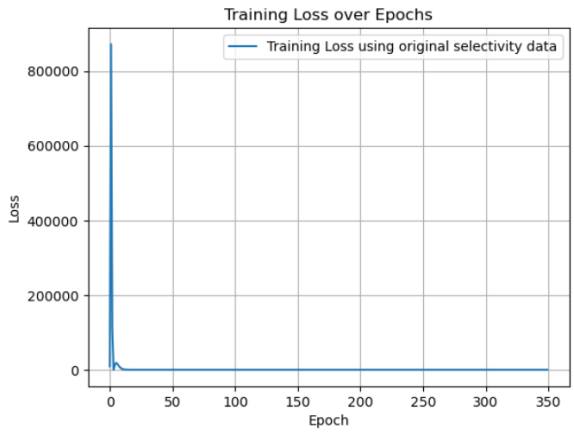

Fig. 3. Neural Network part 1

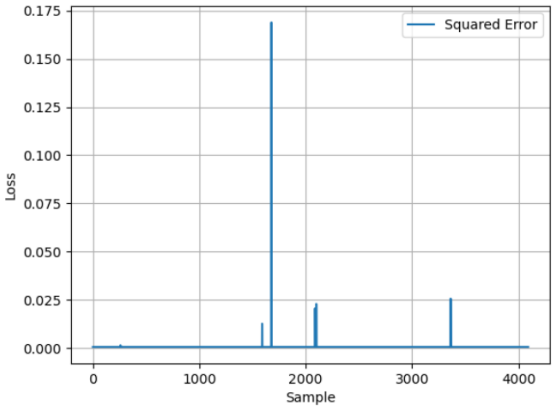

Fig. 4. Neural Network part 2

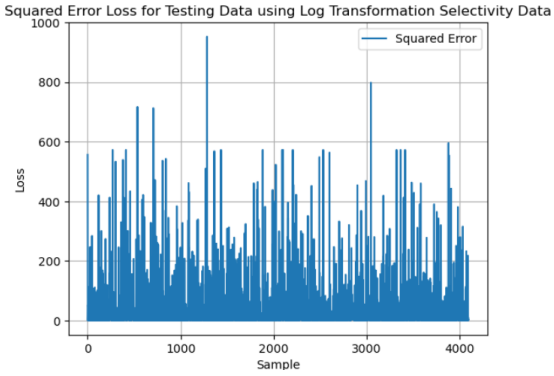

Fig. 5. Neural Network part 3

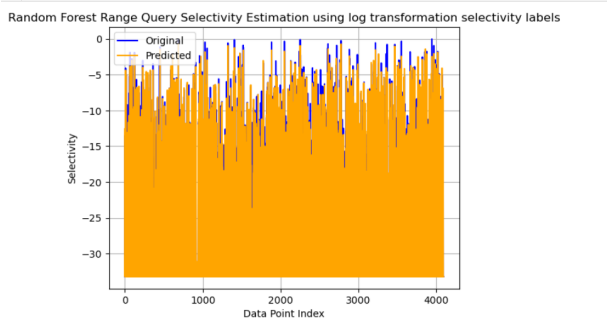

Fig. 6. Random Forest using Log transform labels

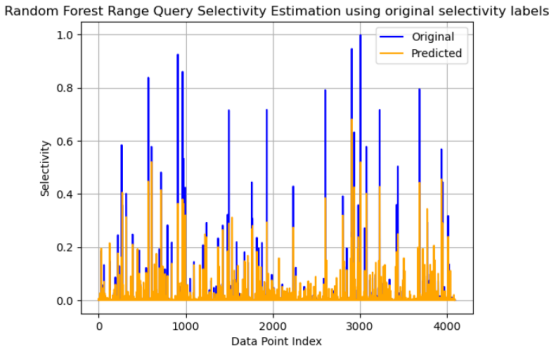

Fig. 7. Random Forest using original selectivity labels

any parameter tuning. The neural network, especially with an adequate number of epochs and two hidden layers, also performed well. However, it's worth mentioning that our neural network model with log-transformed labels did not outperform the neural network without log transform, highlighting an area for future improvement and research.

Upon evaluating the final four models constructed using the LM vectorization technique, it becomes apparent that while

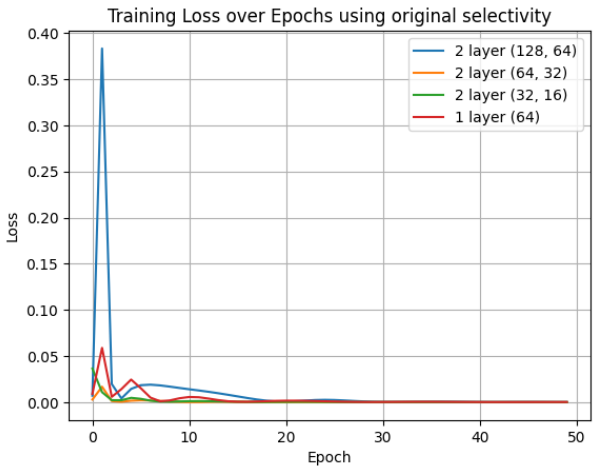

Fig. 8. Training Epochs of four neural networks

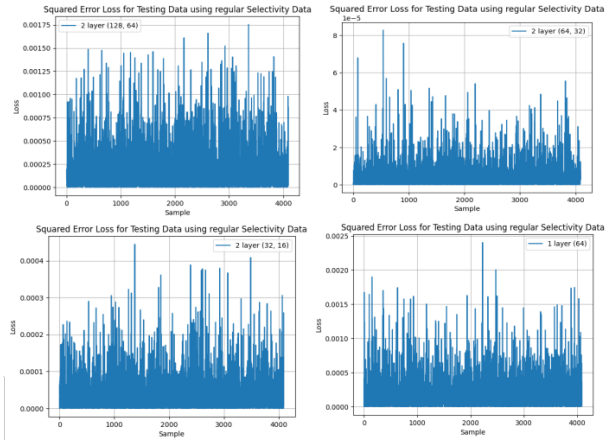

Fig. 9. Test loss comparison across four neural networks

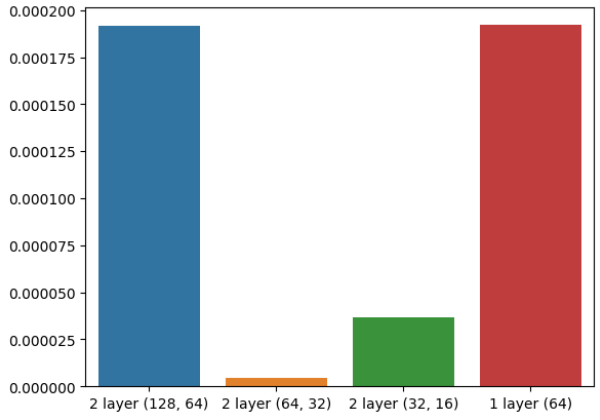

Fig. 10. Average Test loss of 4 neural networks

their performance remains consistent, significant disparities exist in their architectural compositions. This observation is substantiated by the examination of squared error loss graphs for each model in Figure 9 and the relative average error differences in Figure 10. Interestingly, it emerges that it is feasible to streamline the complexity of the model to attain comparable performance levels without compromising the quality of results.

V. Conclusion

Our study assessed the effectiveness of lightweight machine learning models in selectivity estimation, employing two distinct vectorization techniques. We found that applying log transformation and min-max scaling significantly enhances model performance. Interestingly, both vectorization methods yielded comparable performance metrics. Our comprehensive analysis of the results strongly advocates for the adoption of lightweight machine learning models in selectivity estimation tasks.

VI. Future Work

The scope of the conducted experiments has been confined to a singular data table, namely the Forest dataset. Future endeavors should aim to broaden this scope by incorporating other individual tables as well as temporary joined data tables, thereby enabling a comprehensive evaluation of the proposed techniques across diverse datasets. Moreover, further exploration into various network architectures and hyperparameters is warranted to refine the model and optimize its performance to better suit the given scenario.

References

[1] Code repository. https://github.com/farnazi2000/DBMSProject.git
[2] Anshuman Dutt, Chi Wang, Azade Nazi, Srikanth Kandula, Vivek Narasayya, and Surajit Chaudhuri. Selectivity Estimation for Range Predicates using Lightweight Models. PVLDB, 12(9): 1044-1057, 2019. DOI: https://doi.org/10.14778/3329772.3329780
[3] Lucas Woltmann, Claudio Hartmann, Maik Thiele, Dirk Habich,and Wolfgang Lehner. 2019. Cardinality Estimation with Local Deep Learning Models. In International Workshop on Exploiting Arti!cial Intelligence Techniques for Data Management (aiDM'19), July 5, 2019, Amsterdam, Netherlands. ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3329859.3329875
[4] UCI machine learning repository. https://archive.ics.uci.edu/ml/index.php
[5] J. Boulos and M. Bretonneux. "Selectivity Estimation Using Neural Networks"
[6] Shohedul Hasan, Saravanan Thirumuruganathan, Jees Augustine, Nick Koudas, Gautam Das "Multi-Attribute Selectivity Estimation Using Deep Learning" arXiv:1903.09999v2 [cs.DB] 17 Jun 2019
[7] C. M. Chen and N. Roussopoulos. Adaptive selectivity estimation using query feedback. In Proc. of the 199 A CM-SIGMOD Conf., pages 161-172, Washington, DC, May 1994.
[8] Yannis E. Ioannidis, Viswanath Poosala "Balancing Histogram Optimality and Practicality for Query Result Size Estimation" University of Wisconsin Madison, WI 53706
[9] Lise Getoor, Ben Taskar, Daphne Koller "Selectivity Estimation using Probabilistic Models" ACM SIGMOD 2001 May 21-24, Santa Barbara, California, USA
[10] Chieh-Huang Chen, Jung-Pin Lai, Yu-Ming Chang, Chi-Ju Lai, Ping-Feng Pai "A Study of Optimization in Deep Neural Networks for Regression" July 2023

IEEE conference templates contain guidance text for composing and formatting conference papers. Please ensure that all template text is removed from your conference paper prior to submission to the conference. Failure to remove the template text from your paper may result in your paper not being published.