Query-Centric Regression for In-DBMS Analytics

Qingzhi Ma
University of Warwick
Q.Ma.2@warwick.ac.uk

Peter Triantafillou
University of Warwick
P.Triantafillou@warwick.ac.uk

ABSTRACT

Research in enriching DBs with Machine Learning (ML) models is receiving increasingly greater attention. This paper experimentally analyzes the problem of empowering data systems with (and its users with access to) regression models (RMs). The paper offers a data system's perspective, which unveils an interesting '*impedance mismatch*' problem: ML models aim to offer a high expected overall prediction accuracy, which essentially assumes that queries will target data using the same distributions of the data on which the models are trained. However, in data management it is widely recognized that query distributions do not necessarily follow data distributions. Queries using selection operators target specific data subspaces on which, even an overall highly-accurate model, may be weak. If such queried subspaces are popular, large numbers of queries will suffer. The paper will reveal, shed light, and quantify this '*impedance mismatch*' phenomenon. It will study in detail 8 real-life data sets and data from TPC-DS and experiment with various dimensionalities therein. It will employ new appropriate metrics, substantiating the problem across a wide variety of popular RMs, ranging from simple linear models to advanced, state-of-the-art, ensembles (which enjoy excellent generalization performance). It will put forth and study a new, *query-centric*, model that addresses this problem, improving per-query accuracy, while also offering excellent overall accuracy. Finally, it will study the effects of scale on the problem and its solutions.

1 INTRODUCTION

A new dominating trend has emerged for the next-generation data management and analytics systems based on integrating ML models and data management platforms [10, 15, 24, 25, 35, 36, 42, 45]. Additional efforts pertain to connectors to back-end databases, which allow for statistical analyses and related queries on DB data, like MonetDB.R [37], SciDB-Py [23], and Psycopg [48]. Another class of efforts concerns learning from past answers to predict the answers to future analytical queries, e.g. for approximate query processing engines, which provide approximate answers to aggregate queries, using ML techniques [3–5, 41], or for tuning database systems [2], and for forecasting workloads [32]. Yet another class of efforts concerns model and query-prediction serving, like the Velox/Clipper systems [16, 17] managing ML models for predictive analytics. Finally, vision papers suggest the move towards *model selection management systems* [29], where a primary task is *model selection* whereby the system is able to select the best model to use for the task at hand.

In this realm, regression models, being a principal means for predictive analytics, are of particular interest to both analysts and data analytics platforms. RMs are playing an increasingly important role within data systems. Examples of its extended use and significance include many modern DBs which provide support for regression, such as XLeratorDB [6] for Microsoft SQL Server, Oracle UTL_NLA [1, 51], IBM Intelligent Miner [47], which provide SQL interfaces for analysts to specify regression tasks. Academic efforts include MADLib (over PostgreSQL) [24], MAD [15], and MauveDB [18], which integrates regression models into a RDBMS. [39, 40] uses User-Defined Functions (UDFs) to compute statistical machine learning models and data summarization. DBEst [33], an approximate query processing engine, uses ML models (regressions and density estimators) to provide answers for popular queries. Similarly, FunctionDB [46] builds regression models so that future queries can be evaluated using the models. Furthermore, [43] integrates and supports least squares regression models over training data sets defined by join queries on database tables. Thus, in general, RMs are useful for query processing and data analytics tasks. In addition, RMs are helpful for many other key tasks: imputing missing values, testing hypotheses, data generation, fast visualization, etc.

Motivations

Given the above increasing interest in bridging ML and RMs with DBs, we focus on how seamless this process can be. ML models (and RMs in particular) are trained to optimize a loss function (invariably concerning *overall expected error*). We refer to this as a **workload-centric** view, as the aim is to minimize expected error among all queries in an *expected* workload. In essence, this assumes query distributions are (expected workload is) similar to that of the training data. In contrast, data systems research has long recognized that query workloads follow patterns generally different to data distributions. Hence, queries on data subspaces (e.g., using range or equality selection predicates), where an ML model is weak, will suffer from high errors. And, if such queries are popular, many queries will suffer. This gives rise to the need for a "query-centric perspective ": We define **query-centric regression** as a model which strives to ensure both high average accuracy (across all queries in a workload) as well as high per-query accuracy, whereby each query is ensured to enjoy accuracy close to that of the best possible model.

The ML community's general answer to such problems is to turn to ensemble methods, in order to lower variance and generalize better (e.g., to different distributions). We wish to shed light into this possible impedance mismatch problem and see if it holds for simpler and even for state-of-the-art ensemble RMs. We further wish to (i) quantify the phenomenon: We shall use several real data sets (from the UCI ML repository and TPC-DS data) and a wide variety of popular RMs and new metrics that reveal workload-centric and query-centric performance differences, and (ii) see if the problem can be addressed by adopting a query-centric perspective, (using a new ensemble method) whereby the error observed by *each query* is as low as possible (which will also indirectly ensure high workload-centric performance).

The above bear strong practical consequences. Consider a data analyst using python or R, linked with an ML library (like Apache Spark MLLib, Scikit-Learn, etc.), or using a DB connector like MonetDB.R or SciDB-Py, or a prediction serving system like Clipper, etc– and the following use cases.

Scenario 1: The analyst uses a predicate to define a data subspace of interest and calls a specific RM method: It would be great if she knew which RMs to use for which data subspaces.

Scenario 2: Alternatively, the system could select the best RM automatically for the analyst's query at hand.

With this paper we wish to inform the community of this DB-RM impedance mismatch problem, study and quantify it, and lay the foundation for seamless use of RMs for in-DBMS analytics, offering this query-centric perspective.

2 BACKGROUND

Our study employs a set of representative and popular RMs, grouped into two categories: Simple and ensemble RMs.

2.1 Simple Regression Models

Simple RMs include linear regression (LR), polynomial regression (PR), decision tree regression (DTR), SVM Regression (SVR), Nearest Neighbours Regression (NNR), etc. An introduction to these simple regression models is omitted for space reasons.

Table 1 summarizes the known asymptotic time complexity for training for key regression models. And more detailed comparisons are made and discussed in §3.5.

Table 1: Complexity of typical regression models

LR	PR	DTR
$O(d^2n)$	$O(d^4n)$	$[O(dnlog(n)), O(n^2d)]$
NNR	SVR	Gaussian Process
$O(n(d+k))$ or $O(ndk)$	$O(vn^2)$	$O(n^3)$

Note: d is the dimension of the data points, n is the number of points in the training data, k is the number of neighbors for KNN regression, v is the number of support vectors for SVM regression.

2.2 Ensemble Methods

Ensemble methods are powerful methods that combine the predictions and advantages from base models. It is often observed that prediction accuracy is improved by combining the prediction results in some way (e.g., using weighted averaging of predictions from various base models) [8]. Ensemble learning is also useful for scaling-up data mining and model prediction [44]. There have been many well-developed ensemble methods, including averaging based ensemble methods, bootstrap aggregating (bagging) [9], boosting [20, 21], stacking [50], mixture of experts [26], etc.

Averaging-based ensemble methods calculate the weighted average of predictions from all models. This incurs higher computational costs and higher response time.

Boosting refers to a family of algorithms that could potentially convert "weak models"to "strong models". AdaBoost [20], short for "adaptive boosting", is a popular boosting algorithm. Unlike bootstrap aggregating whose models are trained in parallel, the prediction models in AdaBoost are trained in sequence. AdaBoost was firstly proposed to solve classification problems, and was applied to solve regression problems later on. Randomization may be incorporated into boosting, so that its response time is reduced [22].

The objective of gradient boosting is to minimize the loss function:

$$L(y_i, f(x_i)) = MSE = \sum (y_i - f(x_i))^2 \quad (1)$$

And the predictions are updated in the direction of gradient descent, which is

$$f(x_i)^{r+1} = f(x_i)^r + \alpha * \frac{\partial L(y_i, f(x_i)^r)}{\partial f(x_i)^r} \quad (2)$$

where r is the iteration number. Gradient boosting (GBoost) usually uses only the first-order information; Chen et al. incorporate the second-order information in gradient boosting for conditional random fields, and improve its accuracy [12]. However, the base models are usually limited to a set of classification and regression trees (CART). Other regression models are not supported.

XGBoost [11] is a state-of-art boosting method, and is widely used for competitions due to its fast training time and high accuracy. The objective of XGBoost is

$$obj(\Theta) = L(\Theta) + \Omega(\Theta) \quad (3)$$

where $L(\Theta)$ is the loss function, and controls how close predictions are to the targets. $\Omega(\Theta)$ is the regularization term, which controls the complexity of the model. Over-fitting is avoided if the proper $\Omega(\Theta)$ is selected. The base models (booster) can be gbtree, gblinear or dart [49]. Gbtree and dart are tree models while gblinear is linear.

3 EXPERIMENTAL SETUP

All experiments run on an Ubuntu system, with Intel Core i5-7500 CPU @ 3.40GHz × 4 processors and 32GB memory.

3.1 Hypotheses

The study rests on testing, validating, and quantifying two key hypotheses:

Hypothesis 1. *Different RMs, exhibit higher accuracy for different regions of the queried data spaces. Likewise for different data sets. Such differences can be large and may occur even though said RMs may enjoy similar overall accuracy.*

The corollary of Hypothesis 1 is that, even if our analysts in Scenario 1 knew of highly-accurate RMs, many of their analytical queries would be susceptible to large errors. Hypothesis 1 aims to test whether the loss function used by even top-performing RMs, minimizing overall expected accuracy errors, 'hides' this issue. En route, the analysis will quantify this problem across many different RMs, data sets, and dimensionalities.

Hypothesis 2. *Given Hypothesis 1, a model equipped with knowledge of the accuracy distribution of RMs in the query space, can near-optimally serve each query.*

Such a model, coined *QReg*, which is a classifier-based ensemble method that bears a query-centric perspective, will be studied here to validate Hypothesis 2. Hypothesis 2 aims to show that integrating an RM model within a DB can be done in a query-centric manner, avoiding the aforementioned problems. Thus, offering a solution for Scenario 2.

3.2 Data Sets and Dimensionality

To test the hypotheses, eight real-world data sets with different characteristics from the UCI machine learning repository [31] are used, varying the dimensionality from 2 to 5, as well as a large fact table from the TPC-DS benchmark [38].

Data set 1 is a collection of YouTube videos showing input and output video characteristics along with the transcoding time and memory requirements. Data set 2 contains Physicochemical properties of Protein Tertiary Structure. Tasks include predicting

the Size of the residue (RMSD) based on nine properties. There are 45730 decoys and size varies from 0 to 21 armstrong. Data set 3 is an hourly data set containing the PM2.5 gas concentration data in Beijing. The task is to predict PM2.5 concentration (ug/m^3), and the independent variables include pressure (PRES), Cumulated wind speed (Iws), etc. Data set 4 is an online news popularity data set and tasks include predicting the number of shares in social networks (popularity). There are totally 39797 records in this data set. Data set 5 contains 9568 data points collected from a Combined Cycle Power Plant over 6 years (2006-2011), and the task is to predict the net hourly electrical energy output (EP) of the plant. Data set 6 is the YearPredictionMSD data set used to predict the release year of a song from audio features. Most of the songs are commercial tracks from 1922 to 2011. Data set 7 contains the recordings of 16 chemical sensors exposed to two dynamic gas mixtures at varying concentrations. The goal with this data set is to predict the recording of one specific chemical sensor based on other sensors and the gas mixtures. This is a time-series data set containing more than 4 million records in total. Data set 8 records the individual household electric power consumption in one household for more than four years, and there are two million records.

We further employ table `store_ sales` from the popular TPC-DS benchmark [38]. Typical columns used for the experiments include `ss_wholesale_cost`, `ss_list_price`, `ss_ sales_price`, `ss_ext_sales_price` and `ss_ext_wholesale_cost`.

3.3 Evaluation Metrics

Accuracy is measured using the Normalized Root Mean Square Error (**NRMSE**) metric, defined as:

$$NRMSE = \frac{\sqrt{\frac{\sum_{t=1}^{n}(\hat{y}_t - y_t)^2}{n}}}{y_{max} - y_{min}} \tag{4}$$

NRMSE shows overall deviations between predicted and measured values; it is built upon root mean square error (RMSE), and is scaled to the range of the measured values. It provides a universal measure of prediction accuracy between different regression models.

The NRMSE ratio r, compares the prediction accuracy of one RM_i against that of any other RM_j, and is defined as: $r = \frac{NRMSE_i}{NRMSE_j}$. If $NRMSE_j \leq NRMSE_i$, this ratio shows how worse RM_i is compared to RM_j.

The above are standard metrics used for comparing accuracy. However, our study calls for additional metrics. Inherent in our study is the need to reflect the differences in accuracy observed by a query as they depend on the model used. For this we define the concept of *Opportunity Loss* as a natural way to reflect how much the query loses in accuracy by using a sub-optimal model.

Assuming RM_{opt} is the RM with the lowest NRMSE, we define *Opportunity Loss* OL_i as

$$OL_i = \frac{NRMSE_i}{NRMSE_{opt}} - 1, \tag{5}$$

which quantifies as a % the error (the opportunity loss) due to not using the best model RM_{opt} and using RM_i instead.

Furthermore, we define

$$ROL_{i,j} = \frac{OL_i}{OL_k}, \tag{6}$$

as *Relative Opportunity Loss*, which quantifies how much better RM_k does vs RM_i in improving on the opportunity loss.

Intuitively, our aim (with testing Hypothesis 1) is to show that, despite which single model is used, some queries will always be processed by sub-optimal models. So we wish to quantify this opportunity loss. Furthermore, our aim (with testing Hypothesis 2) is to show that a new ensemble model can help significantly alleviate this problem. The $ROL_{i,j}$ metric will help quantify how much a model (*QReg*) improves on this opportunity loss for queries.

3.4 Architecture

Assume that the data system maintains m regression models. When a query arrives, the system needs to identify the model with the least prediction error for this query. We treat this model selection problem as a classification problem. Fig. 1 shows the architecture of this classified regression *QReg* [1].

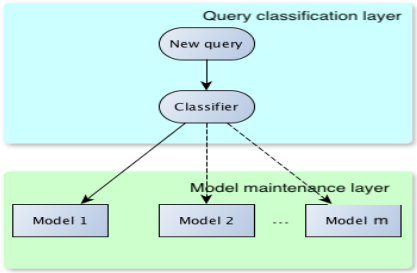


Figure 1: Architecture of the classified prediction method.

There are two layers in the system. (i) The **model maintenance layer**, deploys and maintains the base regression models. (ii) The **query classification layer** implements the core of *QReg*. A query is first passed to the pre-trained classifier. Because the classifier "knows"the prediction ability of each model in the various queried data spaces, the query will be assigned to the model that performs best locally for *this* query's data space. Unlike typical ensemble methods, only one model is invoked for each query (hence *QReg* is less computationally intensive).

Two configurations are studied for *QReg*: *Simple QReg* uses LR, PR, and DTR. *Advanced QReg* uses GBoost and XGBoost as its base models.

3.5 Candidate Base Models

When deciding which models to include the following key criteria are considered.

(a.) Model training time. This should be as low as possible and should exhibit good scalability as the number of the training data points increases.

Given the asymptotic complexity, as summarised in §2.1, a number of experiments were conducted to quantify the training times for various regression models. Fig. 2 is a representative result for data set 4 using 4 dimensions. It shows how model training time (for six regression models) is impacted as the number of training instances increases.

Model training time is shown to behave acceptably with respect to the number of instances in the training data set for LR, PR, DTR, and kNN regression. The training time of Support Vector Regression – Radial Basis Function tends to increase much more aggressively as the number of training points increases. The experiment was repeated for all data sets. The above conclusion holds across all experiments and are omitted for space reasons.

[1]The source code for *QReg* is available at https://github.com/qingzma/qreg

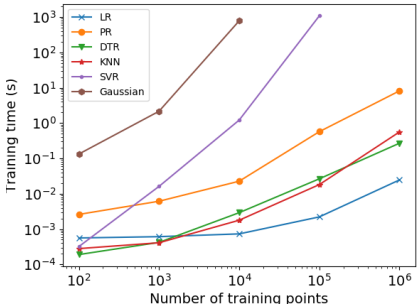


Figure 2: Training time of typical regression models.

(b.) Query response time: In the classifier training process, predictions are made from each base prediction model. To reduce the overall training time as well as the query response time, the models should have as low response time as possible.

(c.) Prediction accuracy: An interesting issue arises from using different regression models together, as in *QReg*. If the base regression models have large differences in accuracy levels, then this may result in *QReg* having poor accuracy. This is a direct result of errors introduced during the separate classification process. Therefore, care must be taken so to ensure that base models enjoy similar and good accuracy levels.

3.6 Model Training Strategy

Fig. 3 shows the model training strategy of *QReg*. The data set is partitioned into three subsets: (i) `Training_Data set_1` is used to train the base models; (ii) `Training_Dataset _2` is used to train the classifier in *QReg*; (iii) `Testing_Dataset` is used to evaluate the accuracy of *QReg*.

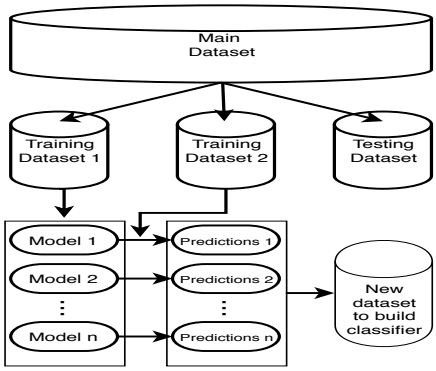


Figure 3: Model training of *QReg*.

After partitioning the data set, m base models are trained upon `Training_Dataset_1`. The selection of base models is in principle open and depends on the users' choice (taking into account the above issues). Each base model $f_i(\mathbf{x})$ makes predictions $\hat{y}_i$ of each data point $\mathbf{x}$ in `Training_Dataset_2`. A comparison is made between the predicted $\hat{y}_i$ and the real label y to find the best prediction model for each query $\mathbf{x}$.

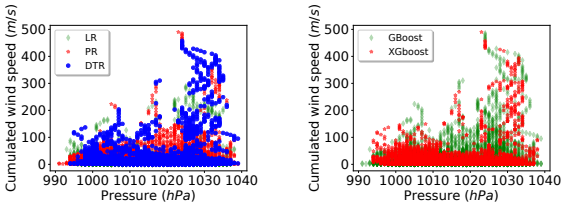
Having the individual predictions and associated errors, a new data set is generated by combining the data point $\mathbf{x}$ and the index i of the best model for this query, depicted $[\mathbf{x}, i]$. This data set is then used to build the classifier reflecting the prediction ability of base models in the query space. The classifier is the core of *QReg*, and a well-designed classifier could potentially grasp the prediction ability of each model in the query space correctly. Thus, the prediction accuracy can be significantly improved compared to individual prediction models.

Note that the original data set is partitioned into 3 subsets instead of 2. This is done in order to ensure that different training data sets are used to train the base models and the classifier, respectively, which avoids potential over-fitting problems. In addition, models are fine-tuned via cross-validation using *GridSearchCV()* in the scikit-learn package.

4 EVALUATING HYPOTHESIS I

Consider data set 3, the Beijing PM2.5 data set [30], using Cumulated Wind Speed (IWS) and Pressure (PRES) as the features, yielding a 3-dimensional regression problem.

Fig. 4.(a) shows the distribution of the model with the least error for all data points. LR, PR, and DTR are used as the base RMs (Simple *QReg*). Fig. 4.(b) shows the distribution of best models when ensemble models GBoost and XGBoost are used (Advanced *QReg*).



(a) Simple models **(b) Ensemble models**

Figure 4: Distribution of best models for Beijing PM2.5.

Take *QReg* using simple models as an example. LR dominates in the upper-central region. PR dominates at the lower central regions. DTR performs best in the rest of the space. the NRMSEs for LR, PR and DTR are 8.48%, 8.84%, and 8.32%, respectively. However, if for each point the best model can be selected to make the prediction (as shown in Fig. 4), the corresponding ("optimal") NRMSE drops to 7.19%. This is a large improvement in accuracy.

Figures like Fig. 4 can help analysts decide on which models to use when querying this space.Similar figures exist for all data sets studied in this work and are omitted due to space reasons.

Table 2: Win-counts of simple RMs.

Data set ID	Count of LR	Count of PR	Count of DTR
1	6468	6919	5366
2	3606	4133	6217
3	1077	1580	1472
4	3618	4826	4155
5	980	885	1307
6	62749	51953	57007
7	9646	4029	4953
8	35702	28800	40311

Table 2 adopts a different perspective. It shows the number of most accurate predictions ("wins") made by each simple RM, per data set. First, note that all models have a good number of wins. Second, no model has the highest number of wins across all data

sets. So, there is no single winner. DTR enjoys the most wins for data sets 2, 5, and 8; PR makes the most accurate predictions for data sets 1, 3, and 4; LR wins for data sets 6 and 7.

Table 3 zooms in, augmenting Table 2 by showing the NRMSEs when different simple RMs win. For example, for data set 1, we know from Table 2 that LR wins 6468 times. For these, the LR's NRMSE was 11.08%, as indicated by Table 3, whereas for PR was enormous and for DTR was 18.28% – see the 3 numbers in cell [1,2] in Table 3. Similarly, for the 6919 queries where PR won, LR's NRMSE was 11.82%, as indicated by Table 3 whereas for PR was 9.35% and for DTR was 11.68% – see the 3 numbers in cell [1,3].

Table 3: NRMSE values when different RMs win.

Data set ID	NRMSE where LR wins	NRMSE where PR wins	NRMSE where DTR wins	Overall NRMSE
1	**11.08%** ≫1000% 18.28%	11.82% **9.35%** 11.68%	14.23% ≫1000% **10.60%**	12.32% ≫1000% 14.06%
2	**27.68%** 30.96% 34.24%	23.78% **20.20%** 26.18%	28.61% 27.85% **21.81%**	27.02% 26.72% 26.79%
3	**8.47%** 9.91% 10.54%	8.00% **6.60%** 7.99%	8.99% 10.04% **6.65%**	8.48% 8.84% 8.32%
4	**2.46%** 3.82% 4.07%	4.96% **2.62%** 3.54%	5.55% 4.49% **2.51%**	4.62% 3.67% 3.41%
5	**16.69%** 18.75% 21.07%	15.97% **13.94%** 16.99%	19.91% 18.79% **15.30%**	17.90% 17.56% 17.72%
6	**11.06%** 12.15% 12.30%	12.43% **11.47%** 12.04%	13.39% 12.77% **12.15%**	12.29% 12.15% 12.17%
7	**80.48%** 210.45% 118.60%	113.75% **83.39%** 111.17%	106.84% 104.22% **76.58%**	95.85% 165.31% 107.31%
8	**8.81%** 10.36% 11.13%	10.02% **8.05%** 9.99%	10.49% 10.04% **8.29%**	9.82% 9.66% 9.80%

Each [i,j] cell shows 3 values for data set i, for the cases where model j wins. $j = 2$ (3, or 4) represents LR, (PR, or DTR) respectively. The top number in each cell shows the NRMSE for LR, the middle shows the NRMSE of PR, and the bottom the NRMSE for DTR.

Consider data set 4. When LR wins, its error is markedly lower (almost half) that of PR and DTR–unlike their overall NRMSEs which show LR to be the worst model.

To further facilitate a query-centric perspective, we delve into the performance of the queries for which each RM reached a top-20% performance. For data set 1, for example, this includes the best 20% of the 6468 queries for which LR wins, the best 20% of the 6919 queries for PR wins, and the best 20% of the 5366 queries for DTR wins. Hence, the NRMSE of interest does not come from all the queries, but from the top 20% queries for which the least error was achieved by a simple RM. Table 4 shows these results, along with the overall NRMSE of each simple RM for the whole set of queries. Again, note that the overall NRMSEs are quite close. However, individual differences are very large. For data set 1, for instance, the top 20% of queries when LR wins enjoy an NRMSE that is about half of the NRMSE of the others. Interestingly, the same holds for PR and DTR! Similar conclusions hold for the other data sets.

Table 4: NRMSEs for the top 20% queries per simple RM.

Data Set ID	NRMSE where LR wins	NRMSE where PR wins	NRMSE where DTR wins	Overall NRMSE
1	**3.61%** 7.06% 7.96%	5.36% **2.92%** 5.78%	6.67% 6.33% **3.42%**	5.36% 5.72% 6.01%
2	**18.35%** 21.26% 24.14%	16.14% **12.39%** 17.09%	16.10% 12.88% **7.97%**	16.89% 16.04% 17.69%
3	**3.03%** 4.89% 5.35%	3.64% **1.44%** 2.78%	4.83% 4.07% **2.12%**	3.902% 3.76% 3.69%
4	**0.96%** 2.33% 2.54%	2.33% **0.63%** 1.40%	2.74% 1.56% **0.78%**	2.15% 1.66% 1.74%
5	**7.56%** 10.00% 12.11%	9.77% **7.75%** 10.55%	9.44% 8.11% **4.85%**	8.98% 8.68% 9.69%
6	**4.11%** 5.22% 5.42%	5.26% **4.12%** 4.76%	5.500% 4.82% **4.13%**	5.00% 4.74% 4.80%
7	**42.14%** 106.72% 79.49%	109.35% **80.88%** 106.48%	95.42% 82.48% **65.26%**	87.25% 90.80% 85.47%
8	**3.07%** 4.37% 5.17%	5.57% **3.77%** 5.92%	3.65% 3.67% **1.60%**	4.23% 3.95% 4.63%

The corresponding data for the advanced ensemble RMs is highly similar and we omit it for space reasons.

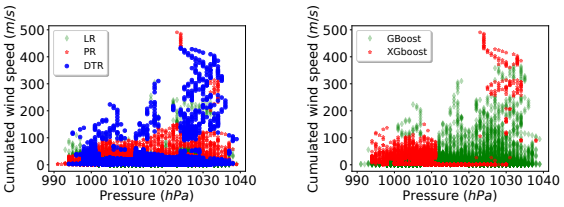
Table 5: Win counts of ensemble RMs.

Data set ID	Count of tree boosting	Count of XGBoost
1	10334	8419
2	7037	6919
3	2653	1476
4	2142	10457
5	1596	1576
6	92916	78793
7	6991	11637
8	52949	51864

5 EVALUATING HYPOTHESIS II

This hypothesis aims to substantiate whether it is possible to develop a method that can learn from the key findings of the previous section and leverage them in order to address Scenario 2, automating the decision as to which RM to use, relieving the DB user/analyst of the conundrum, towards a query-centric RM. Specifically, we study if and at what costs a method can: (i) near-optimally select the best regression model for any query at hand and (ii) achieve better overall accuracy than any single (simple or ensemble) method.

It is natural to treat this problem as a model selection problem, using a classifier for the method selection. We show a new ensemble method, *QReg*, which materializes a query-centric perspective achieving the above two aims.[2] We have considered various classifiers for *QReg*, including SVM-linear classifiers, SVM classifiers using the RBF kernel,and the XGBoost classifier, etc. A comprehensive comparison between various classifiers is made. Unless explicitly stated otherwise, results for the XGBoost classifier are shown, due to its overall prediction accuracy and scalability performance.



(a) Simple *QReg* **(b) Advanced *QReg***

Figure 5: *QReg* distribution of base models.

Fig. 5 shows the RMs chosen by *Simple and Advanced QReg* for each corresponding query, giving a feeling of the overall RM distribution suggested by *QReg*. The model distribution shown resembles the distribution of the truly optimal models across the queried data space, as shown in Fig. 4. Thus, *QReg* does a good job in selecting (near-) optimal RMs per query. As per Scenario 1, presenting such visualisations can be of real value to analysts.

Workload-centric Perspective: Simple QReg

A workload-centric perspective assumes that the query distribution is identical to the data distribution, as described in §1. Simple *QReg* uses simple regression models, including LR, PR, and DTR. Fig. 6 shows the NRMSE ratio r as defined in Equation (4) for all data sets in 3-d space. An NRMSE ratio r larger than 1 means *QReg* has less prediction error than the other base model. *QReg* is shown to outperform or be as good as any of its base models. Specifically, for data sets 2, 3, 5, 6, and 8, *QReg* performs slightly better than other regression models, whereas for data sets 1, 4, and 7, we can see *QReg* being significantly superior versus LR, or PR, or DTR.

Fig. 7 compares the prediction error between *Simple QReg* against the more sophisticated ensemble methods, including AdaBoost, GBoost, and XGBoost. Fig. 7 shows that the even *Simple QReg* often achieves better prediction accuracy than any of the sophisticated ensemble methods. For example, up to 25% reduction in NRMSE is achieved by *Simple QReg* for data set 1 in 3-d space.

Workload-centric Perspective: Advanced QReg

For the majority of the cases, *Simple QReg* is shown to outperform simpler RMs and occasionally more complex ensemble models. For the remainder we concentrate on *Advanced QReg* constructed using GBoost and XGBoost.

Delving deeper, we now show the NRMSE ratios between GBoost (or XGBoost) and *QReg*, broken down to sub-collections

[2]NB: the aim here is ***not*** to find the best method to achieve this, but to show that this is achievable and that significant gains can be achieved using easy to deploy methods.

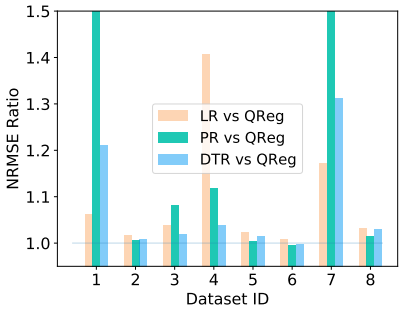
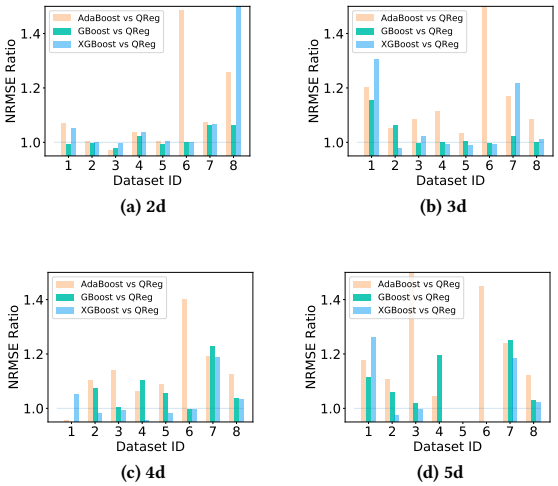


Figure 6: Accuracy of *Simple QReg* vs LR, PR, DTR.



(a) 2d **(b) 3d**

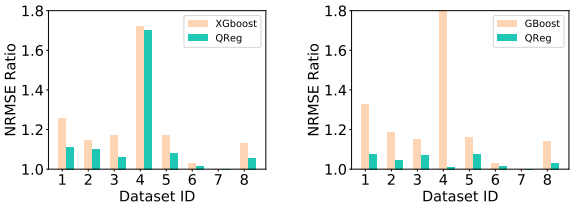
(c) 4d **(d) 5d**

Figure 7: Accuracy of *QReg* vs ensemble RMs

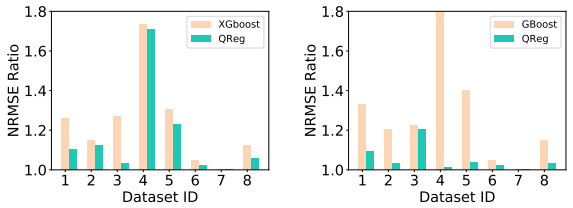
of points in the data space, specifically, for the sub-collection of points where XGBoost regression (or GBoost regression) has the best prediction accuracy.

Fig. 8 shows bands of 2 bars each. Each band of bars shows the NRMSE ratio between other RM and the best RM for the collection of points. Take the 4-d data set 1 as an example, for the collection of points where XGBoost has the best prediction accuracy. The NRMSE ratio between GBoost (second best RM) and XGBoost (collection-best RM) is 1.3288 (orange bar in the figure), while the corresponding NRMSE ratio between *QReg* and XGBoost is 1.0780 (green bar in the figure). This shows that for this collection of points where XGBoost has the best prediction accuracy, GBoost suffers from a 32.88% error relative to the optimal, while using *QReg* reduces this to 7.80%.

As another example, consider the collection of points where XGBoost has the best prediction accuracy in the 5-d data set 8. The NRMSE ratio between GBoost (second best RM) and XGBoost is 1.1458, while the NRMSE ratio between *QReg* and XGBoost is 1.0316. Thus, the relative opportunity loss is 0.1458/0.0316 = 4.61, which means the error caused by using GBoost (relative to the best model) is 4.61 times the error caused by *QReg* for the collection of points where XGBoost has the best accuracy. The relative opportunity loss is much larger for data sets 4 and 5.



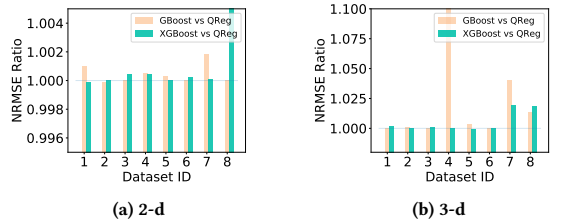
(a) 4-d GBoost win collection (b) 4-d XGBoost win collection



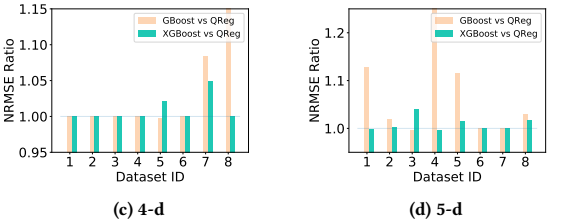
(c) 5-d GBoost win collection (d) 5-d XGBoost win collection

Figure 8: Workload-centric collection-level NRMSE ratio

Fig. 9 shows the ratio r of NRMSE between the base (ensemble) methods and *Advanced QReg* for the whole data sets. Improvement in 2-d space is typically small, but from d=4, we start seeing larger improvements brought about by *QReg*. For 2-d case, the accuracy of GBoost and XGBoost regressor are high and almost equal, which explains why *QReg* cannot improve things further. For the 3-, 4-, and 5-d cases, almost all ratios are above the horizontal line $r = 1$. It shows that models do very well for most data sets. However, there are some cases where *QReg* is significantly better than other ensemble methods, for example against GBoost for the 5-d data set 4 and against XGBoost for the 4-d data set 8. We see that *QReg* can improve accuracy across data sets and dimensionalities.



(a) 2-d (b) 3-d



(c) 4-d (d) 5-d

Figure 9: r between *QReg* and base ensemble models

Comparing Fig. 9 with Fig. 8, we see that even though the overall NRMSE of various RMs is similar, different RMs give different accuracy in different subspaces of the data. Interestingly, this figure also shows that for different data sets different ensemble methods win (as we have seen previously), showcasing the need for a method like *QReg*.

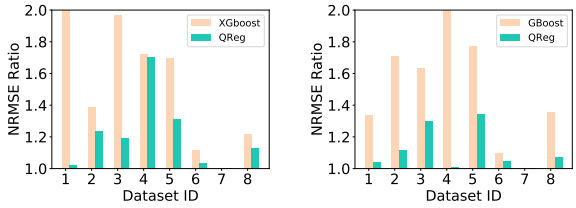
Query-centric Perspective: Advanced QReg

As discussed before, we calculate the NRMSE error for the full collection of points where a single ensemble model wins. To zoom into the context of query-centric prediction serving, we now focus only in the top 20% of queries with the least error, as done previously. To summarize relative performance, the relative opportunity loss between RMs and *QReg* is shown in Table 6.

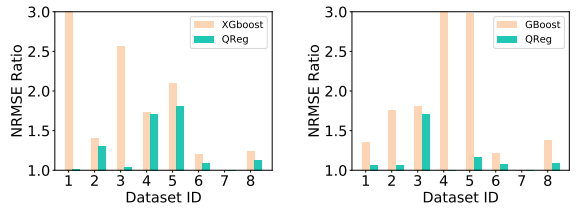
Table 6: ROL w.r.t. *QReg* when different ensemble RMs win for their top 20% queries.

Data set ID	where GBoost wins	where XGBoost wins
1	3.00	2.39
2	2.77	2.68
3	75.50	2.00
4	0.99	>1000
5	2.77	1.87
6	5.67	1.77
7	7.13	3.44
8	2.64	5.29

The values shown exactly are the ROL of using as a second-best RM GBoost (XGBoost) vs *QReg* when XGBoost (GBoost) wins. For example, cell [1,2]=3.00, says that if XGBoost was used (instead of the optimal in this case GBoost) it would result in an error that is 3 times higher than if *QReg* was used. In other words, previous results have shown that, regardless of which RM is chosen, this RM will be suboptimal for certain queries. So these ROL values show how *QReg* can minimize this cost when being suboptimal.



(a) 4-d GBoost win collection (b) 4-d XGboost win collection



(c) 5-d GBoost win collection (d) 5-d XGboost win collection

Figure 10: Query-centric collection-level NRMSE ratio

Similar to Fig. 8, Fig. 10 shows the NRMSE ratio for the 4-5 dimensional space, but in the query-centric (the top 20% of queries) perspective. Focus on the collection of points where XGBoost has the best prediction accuracy in the 5-d data set 8 as an example. The NRMSE ratio between GBoost (second best RM) and XGBoost is 1.3842, while the NRMSE ratio between *QReg* and XGBoost is 1.0872. Thus, the relative opportunity loss is $0.3842/0.0872 = 4.4$, which means the error caused by using GBoost is 4.4 times as the error caused by *QReg* for the collection of points where XGBoost has the best prediction accuracy.

It is noticeable that the NRMSE ratio from *QReg* is always less than that from the second best model, and is very close to 1. Thus, for the most vulnerable queried spaces (where a single ensemble model wins by far), *QReg* can near-optimally achieve the same accuracy, reconciling the otherwise unavoidable loss.

QReg Training Time

Fig. 11 shows the results of our study focusing on the scalability of *QReg*, seeing the performance overheads that need be paid for *QReg*'s accuracy improvement. There exists an approximately

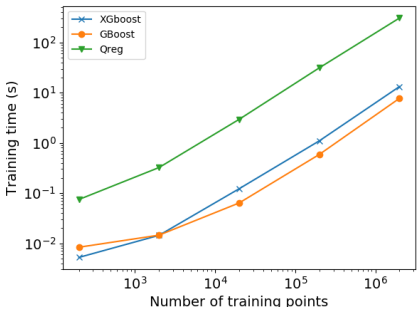


Figure 11: Comparison of model training time

linear relationship between the model training time and number of training points. Even for a relatively high number of training points, (e.g., hundreds of thousands), the training time for *QReg* is shown to be a few dozen of seconds. Although this is an order of magnitude worse than XGBoost in absolute value it is acceptable for medium-sized data sets. Also, about 90% of the training time is spent for getting predictions from the individual base models. In the current version of the code, predictions are received sequentially from base models; doing this in parallel, would reduce the total training time.

6 QREG SCALABILITY

As discussed in §5, the total training time of *QReg* increases approximately linearly as the data size increases. This limits its application to very large data sets. An approach for addressing this issue is to build samples from the data and train *QReg* on the samples. We study the implications of this approach on *QReg*'s performance and observe also whether our Hypotheses hold for this case as well.

6.1 Sample Size Planning

One major question is how big the sample size should be? A smaller sample requires less training time, but might lead to poor accuracy. According to the tasks, various strategies could be used to determine the sample size. For general purposes, Cochran's formula [13] is usually used to determine the sample size for a population.

$$n_0 = \frac{z^2 p(1-p)}{e^2} \tag{7}$$

where n_0 is the sample size, z is the selected critical value of desired confidence level, p is the degree of variability and e is the desired level of precision. For instance, we need to determine the sample size of a large population whose degree of variability is unknown. $p = 0.5$ indicates maximum variability, and produces a conservative sample size. Assume we need 95% confidence interval with 1% precision, the corresponding sample size $n_0 = 9604$. For datasets with a finite size, the sample size is slightly smaller than the value obtained in eq. (7).

For regression-specific tasks, sample size planning techniques include power analysis (PA) [14], accuracy in parameter estimation (AIPE) [28], etc. The sample sizes obtained from both methods are different, and the magnitude is usually hundreds or thousands. [27] proposes a method to combine these methods with a specified probability, while [34] recommends that the largest sample size should be used.

For classification-specific tasks, [19] finds that many prediction problems do not require a large training set for classifier training. [7] uses learning curves to analyze the classification performance as a function of the training sample size, and concludes that 5-25 independent samples per class are enough to train classification models with acceptable performance. Also, 75-100 samples will be adequate for testing purposes.

In this study, the sample size varies from 10k, 100k to 1m, which are conservative compared to the values obtained by the PA and AIPE methods for regression tasks, or the size for classification tasks.

6.2 Workload-centric Perspective

We show results for data sets 6, 7, 8 and Table `store_sales` from the TPC-DS data set. Data sets 6, 7, 8 contain 2-4 million records, and Table `store_sales` is scaled-up to 2.6 billion records. We use reservoir sampling to generate uniform random samples for these data sets. Experiments are done using *Advanced QReg*.

Table 7: Win counts of ensemble RMs.

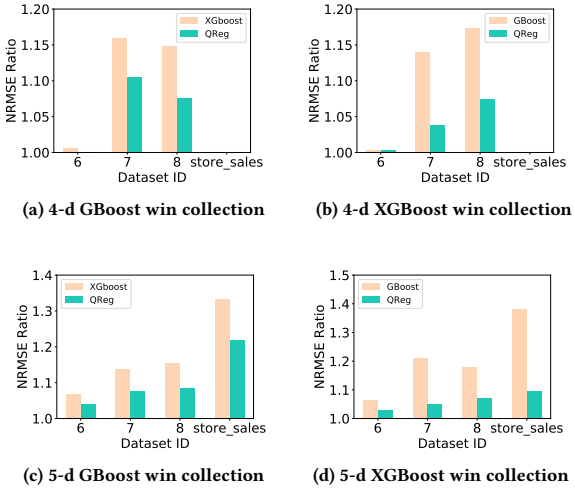
Data set ID	Count of GBoost	Count of XGBoost
6	16209	17124
7	15854	17479
8	13757	19576
store_sales	13415	17003

Table 7 shows the occurrences of best predictions (wins) made by each model, for the samples of size 100k. Similarly to Table 2 in §4, each base model is shown to win for a substantial percentage of queries (or, equivalently for a considerable part of the data set). This supports Hypothesis I that there is not a single regression model capable of dealing with various data sets, and each regression model is only good at sub-spaces of the data sets.

Similar to §5, this section focuses on the workload-centric evaluation but for sample-based *QReg*. We show the NRMSE ratio r between XGBoost (or GBoost) and *QReg*, broken down to subcollections of points in the data space, specifically, for the subcollection of points where GBoost (or XGBoost regression) has the best accuracy.

Consider the collection of points where XGBoost has the best prediction accuracy in the 5-d data set 7. The NRMSE ratio r between GBoost regression (second best RM) and XGBoost regression (best RM) is 1.2107, while the NRMSE ratio between *QReg* and XGBoost regression is 1.0499. Thus, the corresponding ROL between GBoost and *QReg* is 0.2107 /0.0499 = 4.22, which means for this collection of points, GBoost induces 4.22 times higher error than *QReg*.

The same conclusion holds for the query-centric perspective, and is omitted for space reasons.



(a) 4-d GBoost win collection (b) 4-d XGBoost win collection

(c) 5-d GBoost win collection (d) 5-d XGBoost win collection

Figure 12: Workload-centric collection-level NRMSE ratio

6.3 Model Training Time

The training time of sample-based *QReg* consists of two parts: (a) **Sampling time** to generate samples from the base tables; (b) **Training time** to train *QReg* over the samples. Fig. 13 shows

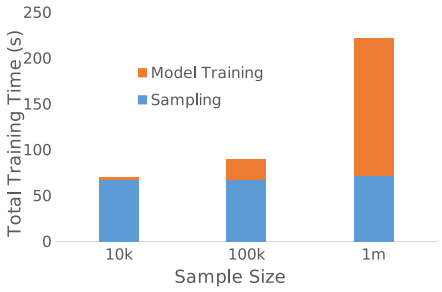


Figure 13: Sample Size vs Training Time for store_sales

the training time of *QReg* for the 100m `store_sales` table, while sample sizes vary {10k, 100k, 1m}. It takes ca. 68-72s to generate the samples. For 10k (100k, 1m) samples, it takes less than 3s (22s, 150s) to train *QReg*. With 100k samples, *QReg* performs excellently. So, in conclusion, sample-based *QReg* is scalable and both hypotheses hold even when models are trained from samples.

6.4 Application to AQP engines.

Previous experiments demonstrate the strength of *QReg*. In this section, *QReg* is applied to DBEst, a newly model-based approximate query processing (AQP) engine [33]. DBEst adopts classical machine learning models (regressors and density estimators) to provide approximate answers to SQL queries. We replace the default regression model in DBEst (XGBoost) with *Advanced QReg*, and compare the accuracy with DBEst using other ensemble methods, including XGBoost and GBoost. The well-known TPC-DS dataset is scaled up with scaling factor of 1000, which contains ~ 2.6 billion tuples (1TB). 96 synthetic SQL queries covering 13 column pairs are randomly generated for `SUM` and `AVG` aggregate functions. DBEst sample size is set to 100k.

Fig. 14 shows the relative error achieved by DBEst using various regression models. For `SUM`, the relative errors using XGboost or GBoost are 8.35% and 8.10%. However, if *Advanced QReg* is

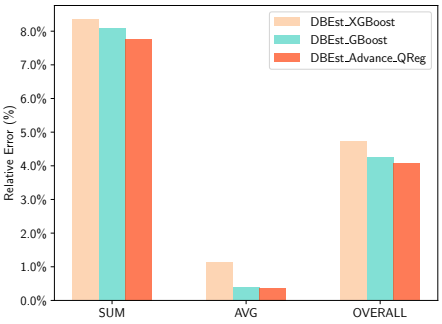


Figure 14: Application of QReg to DBEst

used, the relative error drops to 7.77%. Although *Advanced QReg* is build upon XGBoost and GBoost, the relative error of DBEst using *Advanced QReg* is better than DBEst using XGBoost or GBoost only. For further comparison, if the linear regression is used in DBEst, the relative error becomes 21.20%, which is much higher than DBEst using *Advanced QReg*. Thus, a query-centric regressor, like *QReg*, improves the prediction accuracy and is very important in-DBMS analytics.

7 MAJOR LESSONS LEARNED

The key lessons learned by this study are:

- Different RMs are better-performing for different data sets and, more interestingly, for different data subspaces within them. This holds for simpler models and, perhaps surprisingly, for advanced ensemble RMs, which are designed to generalize better.
- Each examined RM is best-performing (a winner) for a significant percentage of all queries. Necessarily, this implies that, for a significant percentage of queries, regardless of which (simple or ensemble) RM is chosen by a DB user/analyst, a suboptimal RM will be used.
- When said suboptimal RMs are used, significant additional errors emerge for a large percentage of queries.
- Best practice, which suggests using a top-performing ensemble, is misleading and leads to significant errors for large numbers of queries. In several cases, despite the fact that different RMs had a very similar overall error (NRMSE), a significant fraction of queries face very large differences in error when using seemingly-similarly-performing RMs. Thus, sophisticated and simpler RMs cannot cope well, in order to appease query-sensitive scenarios, where query distributions may target specific data subspaces.
- A query-centric perspective, as manifested with *QReg*, can offer higher accuracy across data sets and dimensionalities. This applies to overall NRMSEs. More importantly, it applies to query-centric evaluations. The study revealed that when *QReg* is used, there are significant accuracy gains, compared to using any other non-optimal RM (which as mentioned is unavoidable).
- Accuracy improvements are achieved with small overheads, even with very large data sizes, using sampling.

8 CONCLUSIONS

The paper studied issues pertaining to the seamless integration of DBMSs and regression models. The analysis revealed the complexity of the problem of choosing an appropriate regression

model: Different models, despite having overall very similar accuracy, are shown to offer largely-varying accuracy for different data sets and for different subsets of the same data set. Given this, the analysis sheds light on solutions to the problem. It showed and studied in detail the performance of *QReg*, which can achieve both high accuracy over the whole data set and near-optimal accuracy, per query targeting specific data subsets. The analysis also showed the impact of key decisions en route to *QReg*, such as selecting different constituent base regression models. In addition, it studied issues pertaining to scalability, showing that even with large data sets, the same issues hold and the same model solution can be used to achieve per-query and overall high accuracy. In general, the proposed *QReg* offers a promising approach for taming the generalization-overfit dilemma when employing ML models within DBMSs.

ACKNOWLEDGMENTS

This work was supported in part by the 'Tools, Practices and Systems'theme of the UKRI Strategic Priorities Fund (EPSRC Grant EP/T001569/1) & The Alan Turing Institute (EPSRC grant EP/N510129/1).

REFERENCES

[1] 2005. Database PL/SQL Packages and Types Reference. https://docs.oracle.com/cd/B28359_01/appdev.111/b28419/u_nla.htm#CIABEFIJ
[2] Dana Van Aken, Andrew Pavlo, Geoffrey J. Gordon, and Bohan Zhang. 2017. Automatic Database Management System Tuning Through Large-scale Machine Learning. In *Proceeding of ACM SIGMOD*.
[3] C. Anagnostopoulos and P. Triantafillou. 2015. Learning Set Cardinality in Distance Nearest Neighbours. In *Proceeding of IEEE International Conference on Data Mining, (ICDM15)*.
[4] C. Anagnostopoulos and P. Triantafillou. 2015. Learning to Accurately COUNT with Query-Driven Predictive Analytics. In *Proceeding of IEEE International Conference on Big Data*.
[5] C. Anagnostopoulos and P. Triantafillou. 2017. Query-Driven Learning for Predictive Analytics of Data Subspace Cardinality. *ACM Trans. on Knowledge Discovery from Data, (ACM TKDD)* (2017).
[6] Anon. 2018. XLeratorDB. http://westclintech.com/
[7] Claudia Beleites, Ute Neugebauer, Thomas Bocklitz, Christoph Krafft, and Jürgen Popp. 2013. Sample size planning for classification models. *Analytica chimica acta* 760 (2013), 25–33.
[8] Christopher M Bishop. 2006. *Pattern recognition and machine learning*. springer.
[9] Leo Breiman. 1996. Bagging predictors. *Machine learning* 24, 2 (1996), 123–140.
[10] Zhuhua Cai, Zekai J Gao, Shangyu Luo, Luis L Perez, Zografoula Vagena, and Christopher Jermaine. 2014. A comparison of platforms for implementing and running very large scale machine learning algorithms. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. ACM, 1371–1382.
[11] 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*. ACM, 785–794.
[12] Tianqi Chen, Sameer Singh, Ben Taskar, and Carlos Guestrin. 2015. Efficient second-order gradient boosting for conditional random fields. In *Artificial Intelligence and Statistics*. 147–155.
[13] William G Cochran. 2007. *Sampling techniques*. John Wiley & Sons.
[14] Jacob Cohen. 2013. *Statistical power analysis for the behavioral sciences*. Routledge.
[15] J. Cohen, B. Dolan, M. Dunlap, J. M. Hellerstein, and C. Welton. 2009. MAD skills: new analysis practices for big data. *Proc. VLDB Endow.* (2009).
[16] Dan Crankshaw, Peter Bailis, Joseph Gonzalez, Haoyuan Li, Zhao Zhang, Michael Franklin, Ali Ghodsi, and Michael Jordan. 2015. The missing piece in complex analytics: Low latency, scalable model management and serving with Velox. In *Conference on Innovative Data Systems Research (CIDR)*.
[17] Daniel Crankshaw, Xin Wang, Giulio Zhou, Michael J Franklin, Joseph E Gonzalez, and Ion Stoica. 2016. Clipper: A Low-Latency Online Prediction Serving System. *arXiv preprint arXiv:1612.03079* (2016).
[18] A. Deshpande and S. Madden. 2006. MauveDB: supporting model-based user views in database systems. In *ACM SIGMOD*.
[19] Kevin K Dobbin and Richard M Simon. 2006. Sample size planning for developing classifiers using high-dimensional DNA microarray data. *Biostatistics* 8, 1 (2006), 101–117.
[20] Yoav Freund and Robert E Schapire. 1995. A desicion-theoretic generalization of on-line learning and an application to boosting. In *European conference on computational learning theory*. Springer, 23–37.
[21] Jerome H Friedman. 2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics* (2001), 1189–1232.
[22] Jerome H Friedman. 2002. Stochastic gradient boosting. *Computational Statistics & Data Analysis* 38, 4 (2002), 367–378.
[23] L Gerhardt, CH Faham, and Y Yao. 2018. SciDB-Py. http://scidb-py.readthedocs.io/en/stable/.
[24] Joseph M Hellerstein, Christoper Ré, Florian Schoppmann, Daisy Zhe Wang, Eugene Fratkin, Aleksander Gorajek, Kee Siong Ng, Caleb Welton, Xixuan Feng, Kun Li, et al. 2012. The MADlib analytics library: or MAD skills, the SQL. *Proceedings of the VLDB Endowment* 5, 12 (2012), 1700–1711.
[25] Botong Huang, Matthias Boehm, Yuanyuan Tian, Berthold Reinwald, Shirish Tatikonda, and Frederick R Reiss. 2015. Resource elasticity for large-scale machine learning. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. ACM, 137–152.
[26] Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. 1991. Adaptive mixtures of local experts. *Neural computation* 3, 1 (1991), 79–87.
[27] Michael R Jiroutek, Keith E Muller, Lawrence L Kupper, and Paul W Stewart. 2003. A new method for choosing sample size for confidence interval–based inferences. *Biometrics* 59, 3 (2003), 580–590.
[28] Ken Kelley and Scott E Maxwell. 2003. Sample size for multiple regression: obtaining regression coefficients that are accurate, not simply significant. *Psychological methods* 8, 3 (2003), 305.
[29] Arun Kumar, Robert McCann, Jeffrey Naughton, and Jignesh M. Patel. [n. d.]. Model Selection Management Systems: The Next Frontier of Advanced Analytics. *SIGMOD Rec.* 44 ([n. d.]).
[30] Xuan Liang, Tao Zou, Bin Guo, Shuo Li, Haozhe Zhang, Shuyi Zhang, Hui Huang, and Song Xi Chen. 2015. Assessing Beijing's PM2. 5 pollution: severity, weather impact, APEC and winter heating. In *Proc. R. Soc. A*, Vol. 471. The Royal Society, 20150257.
[31] M. Lichman. 2013. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml
[32] Lin Ma, Dana Van Aken, Ahmed Hefny, Gustavo Mezerhane, Andrew Pavlo, and Geoffrey J Gordon. 2018. Query-based workload forecasting for self-driving database management systems. In *Proceedings of the 2018 International Conference on Management of Data*. ACM, 631–645.
[33] Qingzhi Ma and Peter Triantafillou. 2019. Dbest: Revisiting approximate query processing engines with machine learning models. In *Proceedings of the 2019 International Conference on Management of Data*. ACM, 1553–1570.
[34] Scott E Maxwell, Ken Kelley, and Joseph R Rausch. 2008. Sample size planning for statistical power and accuracy in parameter estimation. *Annu. Rev. Psychol.* 59 (2008), 537–563.
[35] Xiangrui Meng, Joseph Bradley, Burak Yavuz, Evan Sparks, Shivaram Venkataraman, Davies Liu, Jeremy Freeman, DB Tsai, Manish Amde, Sean Owen, et al. 2016. Mllib: Machine learning in apache spark. *The Journal of Machine Learning Research* 17, 1 (2016), 1235–1241.
[36] Xiangrui Meng, Joseph Bradley, Burak Yavuz, Evan Sparks, Shivaram Venkataraman, Davies Liu, Jeremy Freeman, DB Tsai, Manish Amde, Sean Owen, et al. 2016. Mllib: Machine learning in apache spark. *Journal of Machine Learning Research* 17, 34 (2016), 1–7.
[37] Hannes Muehleisen, Anthony Damico, and Thomas Lumley. 2018. MonetDB.R. http://monetr.r-forge.r-project.org/.
[38] Raghunath Othayoth Nambiar and Meikel Poess. 2006. The making of TPC-DS. In *Proceedings of the 32nd international conference on Very large data bases*. VLDB Endowment, 1049–1058.
[39] Carlos Ordonez. 2010. Statistical model computation with UDFs. *IEEE Transactions on Knowledge and Data Engineering* 22, 12 (2010), 1752–1765.
[40] Carlos Ordonez, Carlos Garcia-Alvarado, and Veerabhadaran Baladandayuthapani. 2014. Bayesian variable selection in linear regression in one pass for large datasets. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 9, 1 (2014), 3.
[41] Y. Park, A. S. Tajik, M. Cafarella, and B. Mozafari. 2017. Database Learning: Toward a Database that Becomes Smarter Every Time. In *Proceeding of ACM SIGMOD*.
[42] Christopher Ré, Divy Agrawal, Magdalena Balazinska, Michael Cafarella, Michael Jordan, Tim Kraska, and Raghu Ramakrishnan. 2015. Machine learning and databases: The sound of things to come or a cacophony of hype?. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. ACM, 283–284.
[43] M. Schleich, D. Olteanu, and R. Ciucanu. 2016. Learning Linear Regression Models over Factorized Joins. In *ACM SIGMOD*.
[44] Riyaz Sikora et al. 2015. A modified stacking ensemble machine learning algorithm using genetic algorithms. In *Handbook of Research on Organizational Transformations through Big Data Analytics*. IGI Global, 43–53.
[45] N. Polyzotis T. Condie, P. Mineiro and M. Weimer. 2013. Machine learning for big data. In *ACM SIGMOD*. 939–942.
[46] A. Thiagarajan and S. Madden. 2008. Querying continuous functions in a database system. In *ACM SIGMOD*.
[47] D. S. TKach. 1998. Information Mining with the IBM Intelligent Miner. IBM White Paper.
[48] Daniele Varrazzo. 2014. Psycopg. http://initd.org/psycopg/.
[49] Rashmi Korlakai Vinayak and Ran Gilad-Bachrach. 2015. DART: Dropouts meet multiple additive regression trees. In *Artificial Intelligence and Statistics*. 489–497.
[50] David H Wolpert. 1992. Stacked generalization. *Neural networks* 5, 2 (1992), 241–259.
[51] D. Wong. 2013. Oracle Data Miner. Oracle White Paper.