Large-scale Data Exploration using Explanatory Regression Functions

FOTIS SAVVA, University of Glasgow, UK
CHRISTOS ANAGNOSTOPOULOS, University of Glasgow, UK
PETER TRIANTAFILLOU, University of Warwick, UK
KOSTAS KOLOMVATSOS, University of Thessaly, GR

Analysts wishing to explore multivariate data spaces, typically issue queries involving selection operators, i.e., range or equality predicates, which define data subspaces of potential interest. Then, they use aggregation functions, the results of which determine a subspace's interestingness for further exploration and deeper analysis. However, Aggregate Query (AQ) results are scalars and convey limited information and explainability about the queried subspaces for enhanced exploratory analysis. Analysts have no way of identifying how these results are derived or how they change w.r.t query (input) parameter values. We address this shortcoming by aiding analysts to explore and understand data subspaces by contributing a novel explanation mechanism based on Machine Learning. We explain AQ results using functions obtained by a three-fold joint optimization problem which assume the form of explainable piecewise-linear regression functions. A key feature of the proposed solution is that the explanation functions are estimated using past executed queries. These queries provide a coarse grained overview of the underlying aggregate function (generating the AQ results) to be learned. Explanations for future, previously unseen AQs can be computed without accessing the underlying data and can be used to further explore the queried data subspaces, without issuing more queries to the backend analytics engine. We evaluate the explanation accuracy and efficiency through theoretically grounded metrics over real-world and synthetic datasets and query workloads.

CCS Concepts: • **Mathematics of computing → Exploratory data analysis**; • **Information systems →** *Data analytics*; • **Computing methodologies →** *Supervised learning by regression.*

Additional Key Words and Phrases: Explainability, data exploration, aggregate query explanation, range query explanation.

ACM Reference Format:
Fotis Savva, Christos Anagnostopoulos, Peter Triantafillou, and Kostas Kolomvatsos. 2020. Large-scale Data Exploration using Explanatory Regression Functions. 1, 1 (July 2020), 34 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

In the era of big data, analysts wish to explore, understand and explain processed data in an efficient and effective manner. The typical exploration procedure followed by analysts is rather ad-hoc and domain specific, but invariantly includes the fundamental step of *exploratory analysis* [27]. Exploring data spaces is central for testing hypotheses, building predictive models, modeling data trends, etc. Once this step is complete, the possibilities are endless: from models that predict markets and provide product recommendations to models that decide whether a tumor is benign or malignant.

To this end, Aggregate Queries (AQs), e.g., `COUNT`, `SUM`, `AVG`, play a key role in exploratory analysis, as they *summarize* data regions of interest. The regions are often defined using range operators. A

Authors' addresses: Fotis Savva, f.savva.1@research.gla.ac.uk, University of Glasgow, Glasgow, UK; Christos Anagnostopoulos, christos.anagnostopoulos@glasgow.ac.uk, University of Glasgow, Glasgow, UK; Peter Triantafillou, p.triantafillou@warwick.ac.uk, University of Warwick, Warwick, UK; Kostas Kolomvatsos, kostasks@uth.gr, University of Thessaly, Lamia, GR.

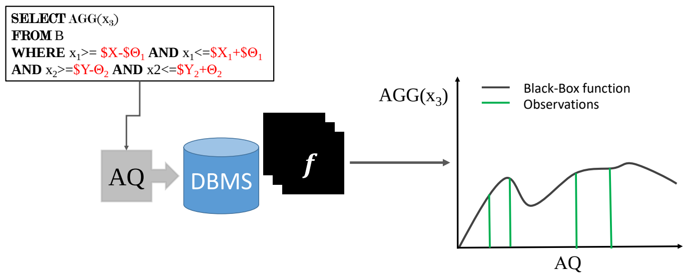
2020. XXXX-XXXX/2020/7-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

range operator limits the number of returned rows (tuples/data items) by restricting the result set to rows within a given region. Using such aggregates, analysts decide whether a data region is of high importance, depending on the task at hand. However, AQs return scalars (single statistics/values) conveying little information for further explaining the underlying data subspace defined by the retrieved rows. For instance, imagine checking whether a particular range of Zip Postal Codes is of interest, where the latter depends on the number of persons having relatively ‘high’ income: if this number of a particular subspace is e.g., 273, then *what does this value mean and/or what other information one could extract by this scalar statistic?* If the selection (range) predicate was less or more selective, how would this statistic change? In addition, if the analyst wanted to approximate the values of input parameters that would generate the maximum/minimum output statistic for a particular region then, a number of extra AQs would have to be executed. To answer such exploratory questions and enhance analysts’ understanding of the queried subspace, one needs to issue more queries to further explore nearby regions. In the beginning of the process, the analyst has no holistic understanding of the space that would steer them in the right direction w.r.t. which and how many queries to issue next; as such, further exploration becomes ad-hoc, wandering, unsystematic, and uninformed. This might, reflect additional cost (e.g., DB access, computations, query processing cost) due to the execution of possibly redundant queries since no systematic guidance is provided to the analysts.

In addition, the importance of AQs in data exploration and analysis is eminent as almost any operation can be described as a collection and/or combination of AQs. For instance, histograms construction can be achieved by executing a number of AQs using range queries. Extracting descriptive statistics for sub-spaces in a dataset, such as various moments (mean, variance, skewness, kurtosis) can also be described by such AQs. To this end, the main objective of this work is to find ways to assist analysts in understanding and analyzing subspaces by explaining AQs as efficiently and effectively as possible.

Our goal is to explain *how* an AQ result over a data subspace is derived. This is required so one can infer the potential impact of a change in the query parameters that defines the queried subspace. The parameters defining a subspace are essentially the predicates in an AQ. A convenient way to represent how a statistic is derived (in terms of compactly and succinctly conveying rich information) is by adopting a statistical regression function. A regression function describes how an output depends upon independent variables (input) and shows the contribution of each one of those variables to the output. For example, we can derive a regression function that explains how the average (mean) household income for each region is generated, based on the size of the region. Thus, the resulting function can inform the data scientists as to how influential the size of a region is and how different results are generated across its different subregions. This functionality is expected to allow the discovery of interesting patterns during exploratory analytics. In addition, such functions can be utilized to interpolate or extrapolate values (approximate AQ results with varying parameters) and generate more predicted data items that can later be used for further analysis. Therefore, the generated regression functions can be used for both predicting the results of AQs and estimating the structural form of the aggregate function generating the results.

To elaborate on our objective, we provide a running example: consider that every result of an AQ is obtained by a black-box aggregate function as shown at Figure 1. An initial SQL query (AQ) is issued and is processed by the DBMS. This particular AQ query defines a region bounding dimensions/attributes (x_1, x_2). An aggregate/statistic is then computed on dimension/attribute x_3. Each aggregate function in an AQ can be represented by a different black-box function that accepts a set of input parameters and maps to a result. The structural form of this black-box function is *unknown*. We can only observe the result given a set of inputs as shown at Figure 1. In this case, the input are the query parameters defining an AQ. Given a number of such observations, we can



```
SELECT AGG(x_3)
FROM B
WHERE x_1>= $X-$Θ_1 AND x_1<=$X_1+$Θ_1
AND x_2>=$Y-Θ_2 AND x2<=$Y_2+Θ_2
```

Fig. 1. The range predicates in an Aggregate Query (AQ) represent the input parameters of a black-box function that generates the corresponding aggregation results.

adopt Machine Learning (ML) methods to approximate its structural form using ML models that are highly interpretable. In doing this, the analysts can infer the potential impact different parameters have on the result/statistic and guide them efficiently and effectively to explore other sub-spaces inexpensively.

Unfortunately, state of the art DB Management Systems (DBMS) and Big Data engines do not provide explanations for AQs, even though they can be of great assistance to analysts. The gap to be filled, is non-trivial as one has to find efficient, scalable and accurate methods of providing such functionality (coined here as *explainability*). Even though explanation methods have recently been gaining attention in the relevant literature, these are not oriented to aid exploratory analytics and they typically require physical access to potentially large-scale datasets, which makes them time- and resource-costly. In this paper, we propose a novel framework which provides explanations as to *how* AQ results are derived while being efficient to compute.

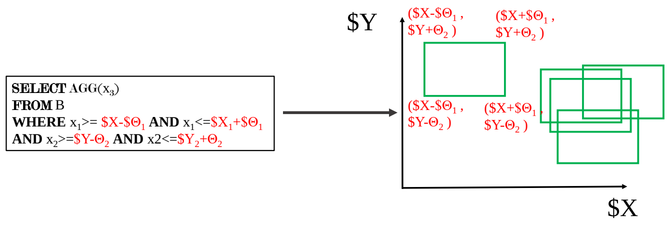
1.1 Motivation and Scenarios

We focus on AQs with a *range* selection operator because of their wide-applicability in analytics tasks. A range operator is defined by a hyper-rectangle in multi-dimensional space. An example of a series of AQs is shown at Figure 2. The range predicates of an AQ are used to represent rectangles. Such operator is evident in many applications including: location-based search, e.g searching for spatially-close objects, such as astronomical objects, objects within a geographical region etc.

Scenario 1: Crimes Data. Consider analyzing a dataset containing recorded incidents along with their location and the type of crime (homicide, burglary, etc.), like the Chicago Crimes Dataset [1]. A typical exploration is to issue the AQ with a range operator:

```
SELECT COUNT(*) AS y FROM Crimes AS C
WHERE C.X > $X - $Theta_1 AND C.X < $X + $Theta_1 AND
C.Y > $Y -  $Theta_2 AND C.Y <  $Y +  $Theta_2;
```

This query can be represented by a rectangle with a 2D center and two variables representing the length of the first and second dimension. The 2D center of the rectangle is defined by the point (\$X, \$Y) and length by $\$\Theta_1$, $\$\Theta_2$ in both dimensions. Such AQ returns the number (count) of crimes in a specific area of interest defined as above corresponding to an arbitrary neighborhood; see Figure



```
SELECT AGG(x3)
FROM B
WHERE x1>= $X-$Θ1 AND x1<=$X1+$Θ1
AND x2>=$Y-Θ2 AND x2<=$Y2+Θ2
```

Fig. 2. A number of 4-dimensional aggregate queries (AQs) represented as rectangles in the 2D data space defined by X and Y features. Each coordinate is constructed by the sum of the corresponding dimension and its length over the data space.

2. Using several such AQs, we build regression functions with input variables the AQ parameters: ($X, $Y, $Θ$_1$, $Θ$_2$) and output y, i.e., the count statistic. The estimated regression coefficients would then provide a way for the analysts to infer the potential impact query parameters (defined region) will have on the outcome, e.g., count of crimes in that region in our example. Therefore, the analysts acquire knowledge of which AQs will provide answers tailored to their interests. For instance, if they are interested in high crime index (increased output y), then the regression coefficients can inform them as to which query to execute next by changing the corresponding query parameters.

Scenario 2: Telecommunication Calls Data. Consider a data scientist tasked with identifying time-frames having high average call times. They need to issue AQs of varying-sized ranges over time, such as the following range-AQ:

```
SELECT AVG(Call_Time) AS y FROM TCD AS C
WHERE C.Time BETWEEN
$X-$Theta AND $X+$Theta
```

Discovering the aforementioned time-frames, without our proposed explanations, can be a daunting task as multiple queries have to be issued, overflowing the system with a number of AQs. These could take minutes or hours to execute depending on the data size and throughput of the system. By using our proposed explanation function, the analyst could carry out their task with highly accurate answers. This is achieved by plugging in different query parameters to the given explanation function. Beyond that, analysts could formulate an optimization problem that could be solved using our methodology. Given a differentiable aggregation function, the maxima and minima points can also estimated, thus, the analysts can easily discover the query parameters at which the AQ result is maximized or minimized. Again, such functionality is very much lacking and is crucial for exploratory analytics.

Another important point to make here is that we do not strive to explain individual SQL queries. We merely adopt the use of SQL due to its prevalence as the language of choice in data analytic systems. Instead what we are aiming at is a way to explain how the aggregate result of a particular SQL query would change given different query parameters. Hence, our proposed solution is not constricted to explaining SQL queries but instead provides a way of identifying which spaces to

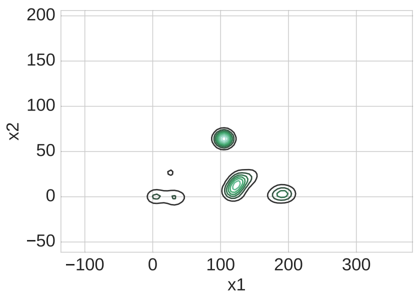


Fig. 3. Cluster analysis and formation of a real query workload (source: SDSS [47]); x_1 and x_2 are parameters of a range query.

explore next. This would in turn help the analyst limit the number of SQL queries that they would execute against an expensive (in terms of computation) data analytic system.

1.2 Problem Definition

We consider the statistic output (result) of an AQ stemming from an unknown function, hereinafter referred to as the *true* function; with that true function's output fluctuating as the query parameter values of an AQ change. Thus, we are faced with the problem of approximating the true function with high accuracy. Using the aforementioned facts, we define an optimization problem to find an optimal approximation to the *true* function. In turn, this problem is further deconstructed into three optimization (sub)problems. The first one seeks the optimal query prototype parameters representing a query space. The described AQs often form clusters as a number of AQs are issued with similar parameters. This is evident from Figure 3 displaying AQs (from real-world analytical query workload [47]) forming clusters in the query space. AQs clustered together are expected to have similar results, thus, similar true functions. Consequently, we are interested in finding a set of optimal query prototype parameters to represent queries in each one of those clusters, as this will be more accurate than one set of global prototype parameters[1]. The second part of the deconstructed optimization problem is approximating the true function(s) using a known class/family of functions over the clustered query space. To this end, we use piecewise-linear functions that are fitted over the clustered query space. In addition, based on historically executed queries and incoming queries, we define a third joint optimization problem, which is solved in an on-line manner by adapting both the obtained optimal parameters and the approximate explanation function from the first two sub problems.

2 RELATED WORK & CONTRIBUTIONS

2.1 Related Work

Our overarching aim is to provide explanations for AQ, whose efficient computation has been a major research interest [4, 7, 8, 15, 17, 25, 26, 39, 52, 56], with methods applying sampling, synopses and Machine Learning (ML) models to compute such queries. Compared to our work, the above works are largely complementary. One distinguishing feature is that our primary task is to *explain the AQ results* and do so efficiently, accurately, and scalably w.r.t. increasing data sizes. Others also focus on the task of assisting analysts by building systems for efficient visual analytics systems [18, 49]. However, we note that our work is different in that it offers explanations for aggregates

[1]This basically represents all queries in a cluster by one query.

(as functions) that can be visualised (as the aforementioned works) but can also be adopted for assisting exploration and for answering subsequent queries.

Explanation techniques have emerged in multiple contexts within the data management and ML communities. Their origin stems from data provenance [16], but has since departed from this notion, and focuses on explanations in varying contexts. One of such contexts is to provide explanations, represented as predicates, for simple query answers as in [19, 35, 42]. Similarly, other authors have extended this in probabilistic and scientific databases [29], [55]. Explanations have been also used for interpreting outliers in both *in-situ* data [54] and in streaming data [9]. The authors first detect outliers, either manually or automatically, and then generate predicates or attribute-value combinations that explain the outliers set. In [38], the authors build a system to provide interpretations for service errors and present the explanations visually to assist debugging large scale systems. In addition, the authors of PerfXPlain [30] created a tool to assist users while debugging performance issues in Map-Reduce jobs. Other frameworks provide explanations used by users to locate any discrepancies found in their data [50], [13], [51]. A recent trend is in explaining ML models for debugging purposes [31] or to understand how a model makes predictions [41] and conveys trust to users using these predictions.

Given the above, one can detect the central themes emerging around the concept of explanations. When working with explanations, one has to determine the (i) domain, (ii) the scalability and efficiency of generating explanations, and (iii) the knowledge representation. For instance, in [9] the domain is in outliers analysis, the explanations are represented using attribute-value combinations, and the approach is to use statistical structures that allow the analysis of streaming values.

In this work, we focus on explanations for AQs because of the AQs' wide use in exploratory analytics [52]. Hence the **domain** is AQ explanation within the context of exploratory data analysis. Both [54] and [5] focused on explaining aggregate queries. However, the former focused on explaining the existence of outliers in aggregate queries and the latter on tracking the 'how?' provenance of AQs using semi-ring formalism. Our major difference is that we explain these AQs solely based on the query input parameters and query results of previously issued and incoming queries, thus not having to rely on directly data access, which makes generating explanations slow and inefficient for large-scale datasets. We wish to find ways to explain AQs and provide further insights to data analysts for establishing informed exploratory and descriptive statistics tasks, in an efficient manner. Hence, using the formalisms from [5] is not possible as they would be incomprehensible and in need of provenance query language, as explicitly stated in [5].

Furthermore, scalability and efficiency are particularly important: Computing explanations is proved to be an NP-Hard problem [50] and generating them can take a long time [19, 42, 54] even with modest datasets. An exponential increase in data size implies a dramatic increase in the time to generate explanations. Our framework does not suffer from such limitations and is able to construct explanations in a matter of milliseconds even with massive data volumes. This is achieved due to two principles: First, on workload characteristics: workloads contain a large number of overlapping queried data subspaces, which has been acknowledged and exploited by recent research e.g., [39, 52], STRAT[14], and SciBORQ [32] and has been found to hold in real-world workloads involving exploratory/statistical analysis, such as in the Sloan Digital Sky Survey and in the workload of SQLShare [28]. Second, we rely on a novel ML model that exploits the workload characteristics to perform efficient and accurate AQ explanations by taking into consideration a stream of query-answer pairs on-line.

2.2 Contributions

The adopted knowledge representation of explanation is in the form of regression functions. Recent works [36, 43] use this representation as it is highly parsimonious and can pinpoint the source of anomalies in aggregate results. In [36] this is used to explain data records and not AQs. Past AQs are not considered for building regression functions. Therefore our work is applied within a different domain. This is an extension of our prior work [43]: in this paper, we extend our approach to cover a wider variety of AQs and also extend the representation so as to be explainable by the user/analyst. In our previous work, the given explanation included a parameterized regression function. A function with a single parameter (the radius) parameterized by the multi-dimensional center was given. This made it harder to interpret as the potential impact of the multi-dimensional center was not easily inferred. Concretely, the **extensions** of this work are:

- Comprehensive extension of the supported AQs to include a wider variety of commonly used aggregate functions and operators;
- Alternative knowledge representation, which includes query parameters to the generated explanation function;
- Fast and accurate approximation of the explanation function by leveraging past and incoming stream of query-answer pairs;
- Formulation of the explainability as a novel hetero-associative statistical learning methodology (combination of supervised and unsupervised learning methods) along with the corresponding optimization algorithm;
- A variety of visualisations constructed using a new form of explanation functions;
- Comprehensive performance evaluation of the accuracy of AQ explanations in terms of efficiency, scalability and sensitivity analysis over real multidimensional data.

In addition to the novel extensions made by this work, we would also like to highlight that although the underlying (base) statistical learning methods are off-the-shelf (e.g., clustering algorithm, Piece-wise Linear regression models), the context and the way in which they are used is novel in the way we combine supervised and unsupervised statistical learning. We chose not to develop our own algorithms for regression/clustering tasks as using tested and highly regarded methods for specialized tasks reduced the complexity of our solution. Instead, in this context we also contribute to:

- A novel constrained-driven statistical learning methodology that combines supervised learning (regression) with unsupervised learning (clustering / vector quantization) to derive online learning algorithms for *explainability* of aggregate results. Note that the underlying (base) algorithms are merely a piece of the complete solution. The complete formulation and solutions to the proposed multi-optimization problem provided in this work are novel.
- A novel *hetero-associative online learning methodology*, which manages to incrementally adjust the models and their representatives as new queries are processed in an on-line manner.

These contribution points go far beyond the basic functionality that is provided by the underlying methods used. Overall, even though the clustering and regression base algorithms are off-the-shelf, the entire optimization framework along with the corresponding hetero-associative statistical learning methods, where the algorithms are trained and the models are adjusted in both modes: offline and online, are novel. To the best of our knowledge, this constrained-driven statistical learning methodology is new in the area of data exploration and explainability of results.

3 EXPLANATION REPRESENTATION

A comprehensive table of notation is in Table 3 in the appendix.

3.1 Query Vectorial Representation

Let $\mathbf{a} = [a_1, \ldots, a_d] \in \mathbb{R}^d$ denote a random row vector (data point) in the d-dimensional data space $\mathbb{D} \subset \mathbb{R}^d$. A dataset $\mathcal{B}$ contains N random row data vectors $B = \{\mathbf{a}\}_{i=1}^{N}$; $|\mathcal{B}| = N$ indicates the cardinality of the set $\mathcal{B}$.

Definition 1. *(Range Query) A range query is defined as the vector:* $\mathbf{q} = (\mathbf{x}, \boldsymbol{\theta})$, $\mathbf{x} \in \mathbb{R}^d, \boldsymbol{\theta} \in \mathbb{R}^d_+$. *This vector* $\mathbf{q}$ *defines a hyper-rectangle in multi-dimensional space as a series of conjunctions of predicates, i.e* $\bigwedge_{i=1}^{d}(x_i - \theta_i \leq a_i \leq x_i + \theta_i)$.

Definition 2. *(Data Subspace) Given a range query, a data subspace* $\mathbb{D}(\mathbf{x}, \boldsymbol{\theta})$ *is the convex subspace of* $\mathbb{R}^d$, *which includes data vectors* $\mathbb{D}(\mathbf{x}, \boldsymbol{\theta}) = \{\mathbf{a} \in \mathbb{R}^d | \bigwedge_{i=1}^{d}(x_i - \theta_i \leq a_i \leq x_i + \theta_i)\}$.

Definition 3. *(Query Similarity) The p-norm* (L_p) *distance between two query vectors* $\mathbf{q}$ *and* $\mathbf{q}'$ *from* $\mathbb{R}^{2d}$ *for* $1 \leq p < \infty$, *is* $\|\mathbf{q} - \mathbf{q}'\|_p = (\sum_{i=1}^{d} |q_i - q_i'|^p)^{\frac{1}{p}}$ *and for* $p = \infty$, *is* $\|\mathbf{q} - \mathbf{q}'\|_\infty = \max_{i=1,\ldots,d}\{|q_i - q_i'|\}$.

Definition 4. *(Aggregate Query) Given a data subspace* $\mathbb{D}(\mathbf{x}, \boldsymbol{\theta})$ *an AQ* $\mathbf{q} = (\mathbf{x}, \boldsymbol{\theta})$ *with input center* $\mathbf{x}$ *and vectorial length* $\boldsymbol{\theta}$, *an aggregate function, is represented via a regression function* $f : \mathbb{R}^d \times \mathbb{R}^d_+ \to \mathbb{R}$ *over* $\mathbb{D}(\mathbf{x}, \boldsymbol{\theta})$, *that produces a query response variable* $y = f(\mathbf{x}, \boldsymbol{\theta})$ *which is the result/answer of the AQ. We notate with* $\mathbb{Q} \subset \mathbb{R}^{2d}$ *the query vectorial space.*

Note that range queries, in general, are over a continuous (real-valued) domain and restrict the data (sub)space to be within a hyper-rectangle. In a categorical data space, we deal with equality predicates in the involved queries, which is beyond the scope of this work for explanation serving. Various encoding schemes could be utilized to encode categorical values into a continuous domain [37], [53]. However, this requires that distances (similarities) between various categorical elements should be given specific meaning; which is beyond the scope of this work. We focus on multi-dimensional range queries in a continuous domain (real-valued query parameters in a vector space) as they are largely used in the domain of exploratory analytics.

3.2 Functional Representation of Explanations

The defined AQ returns a single scalar value $y = f(\mathbf{x}, \boldsymbol{\theta})$ to the analysts. We seek to explain how such values are generated by finding a *function* $f : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ that can describe how y, is produced given an ad-hoc query $\mathbf{q}$. We first discuss possible forms of the said function. An approximate explanation function can be linear or high-order polynomial to approximate non-linear functions. When using high-order polynomial functions for our explanations, the analyst can no longer interpret the structural form of the model because of all the added terms. In addition, by using high-order polynomials we make the assumption that the order of the polynomial is known. On the other hand, adopting a single linear function will still have trouble representing explanations accurately. For instance, if the result y derives from an average operator AVG or a Pearson correlation operator CORR, then the output of the function might increase/decrease abruptly. A linear function increases/decreases infinitely as the input increases/decreases.

Therefore, choosing the right family of functions is non-trivial. We choose to employ multiple local linear functions to capture the possible inherent non-linearity of f as we vary the input query parameters, i.e., $\mathbf{x}$ and $\boldsymbol{\theta}$ vectors. Our idea for the knowledge representation of the explanation function is that the locally linear functions are fitted through a hierarchical clustering scheme in order to decrease the variance among the input parameters as much as possible. Linear functions are then fitted in an iterative manner until fitting is optimal.

A Piecewise-Linear Regression (PLR) approximation of f addresses the above shortcomings by finding the *best* multiple local linear regression functions. Using PLR functions, we can approximate true functions which are linear or non-linear in a coarse grained manner that is more interpretable than a high-order polynomial. The analyst is simply exposed to the different locally-linear functions under a given segment.

Definition 5. *(Explanation Function) Given an AQ* $\mathbf{q} = (\mathbf{x}, \boldsymbol{\theta})$*, an explanation function* $f(\mathbf{x}, \boldsymbol{\theta})$ *is defined as the fusion of local piecewise linear regression functions* $f \approx \sum \hat{f}(\mathbf{x}, \boldsymbol{\theta})$ *derived by the fitting of the local functions* $\hat{f}$ *over similar presviously executed AQ queries* $\mathbf{q}' = (\mathbf{x}', \boldsymbol{\theta}')$ *to the AQ* $\mathbf{q}$.

Although we make the choice of using PLR functions because of their high interpretability and flexibility to fit both linear and non-linear functions, our framework can also use other ML models that can provide a method of estimating the importance of the input parameters. The optimization problems to be defined later on can be adapted to work with such models. However, as we will elaborate later, the importance of the query parameters only show the relative importance at a particular subspace and not necessarily how the output will change w.r.t a change in the input query parameters.

4 EXPLANATION APPROXIMATION FUNDAMENTALS

The challenge in approximating the underlying function f over the data subspaces defined by an AQ, lies in seeking local regression functions. These functions should explain the way query result y varies as query vector $\mathbf{q}$ changes without access to the underlying data, as this would harm efficiency. We build explanation functions by *only* leveraging previously executed and incoming AQs. In this section we provide the methodology followed for accurately identifying such PLR functions.

4.1 Explanation Approximation

We utilize AQs to train statistical learning models that are able to accurately approximate the *true* explanation function f for any possible query. Given these models, we no longer need access to the underlying data, as initially we leveraged queries executed a-priori.

Formally, given a well defined explanation loss or discrepancy metric $\mathcal{L}(f, \hat{f})$ between the *true function* $f(\mathbf{x}, \boldsymbol{\theta})$ and an *approximated* function $\hat{f}(\mathbf{x}, \boldsymbol{\theta})$, we seek the optimal approximation function $\hat{f}^*$ that minimizes the Expected Explanation Loss (EEL) for *all possible* queries:

$$\hat{f}^* = \arg\min_{\hat{f} \in \mathcal{F}} \int_{\mathbf{x} \in \mathbb{R}^d} \int_{\boldsymbol{\theta} \in \mathbb{R}^d_+} \mathcal{L}(f(\mathbf{x}, \boldsymbol{\theta}), \hat{f}(\mathbf{x}, \boldsymbol{\theta})) p(\boldsymbol{\theta}, \mathbf{x}) d\boldsymbol{\theta} d\mathbf{x}, \quad (1)$$

where $\boldsymbol{\theta}$ is strictly positive as it defines the data subspace covered by the query's hyper-rectangle and $p(\mathbf{x}, \boldsymbol{\theta})$ is the probability density function of the query vectors $\mathbf{q} \in \mathbb{Q}$. Eq(1) defines an optimization problem that its solution gives us the optimal approximation of the true underlying function.

However, as stated earlier, accuracy will be problematic as it seems intuitively wrong that one such function can explain *all* the queries at any location $\mathbf{x} \in \mathbb{R}^d$ with any length $\boldsymbol{\theta} \in \mathbb{R}^d$. Such an explanation is not accurate because analysts issue queries over different sub-spaces of interest. For instance, data analysts might issue AQs with a different length and a fixed center to compare if a statistic over a small neighbourhood at a given location increase/decrease as the neighbourhood size increases/decreases. In addition, having a fixed length and a varying center $\mathbf{x}$ will almost surely produce different output as the analysts are essentially querying different fixed size areas. Even with the use of PLRs, we found that variance in the output was still large enough that the produced explanations were not accurate enough.

Therefore, we introduce *local* approximation functions $\hat{f}_1, \ldots, \hat{f}_K$ that collectively minimize the objective in (1). Thus, we no longer wish to find one global approximation to the true function for

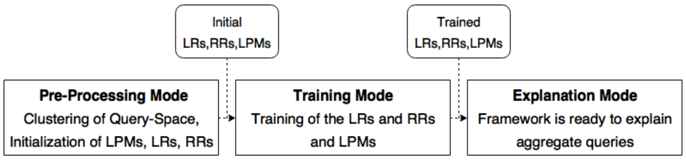

Initial
LRs,RRs,LPMs
Trained
LRs,RRs,LPMs
Pre-Processing Mode
Clustering of Query-Space, Initialization of LPMs, LRs, RRs
Training Mode
Training of the LRs and RRs and LPMs
Explanation Mode
Framework is ready to explain aggregate queries

Fig. 4. The three different modes of the proposed framework. Each resulting output from each mode is pipelined into the next.

all possible queries. Instead, we fit a number of locally optimal functions, where each of them can explain a *subset* of possible queries. We refer to those models as Local PLR Models (LPM).

The LPMs are fitted using AQs forming a cluster of similar query parameters. For each uncovered cluster of AQs, we fit (at least) an LPM. Therefore, if K clusters are obtained from clustering the query vectorial space $\mathbb{Q}$, then K LPMs are fitted. Intuitively, this approach exploits queries that are *similar* to each other within the same cluster. It relies on the hypothesis that these queries will tend to have *similar* results/answers. So in turn, the underlying true function generating their outputs will have *similar* statistical structure. It will be shown that the resulting fused explanation is more accurate as the reduced variance in the input query parameters effectively reduces the variance in the output/answer y. This was empirically shown from our experimental workload.

Formally, to minimize the EEL, we seek K local approximation functions $\hat{f}_k \in \mathcal{F}, k \in [K]$ from a family of linear regression functions $\mathcal{F}$, such that for each query $\mathbf{q}$ belonging to the partition/cluster k of the query space, notated as $\mathbb{Q}_k$, the summation of the local EELs is minimized:

$$\mathcal{J}_0(\{\hat{f}_k\}) = \sum_{\hat{f}_k \in \mathcal{F}} \int_{\mathbf{q} \in \mathbb{Q}_k \subset \mathbb{R}^{2d}} \mathcal{L}(f(\mathbf{q}), \hat{f}_k(\mathbf{q})) p_k(\mathbf{q}) d\mathbf{q} \tag{2}$$

where $p_k(\mathbf{q})$ is the probability density function of the query vectors belonging to a query sub-space $\mathbb{Q}_k$. Thus, $\mathcal{J}_0$ forms our *generic* optimization problem mentioned in Section 1.2. Note: the *explanation loss* $\mathcal{L}(f, \hat{f})$ represents the discrepancy of the actual explanation function f due to the approximation of explanation $\hat{f}$. For evaluating the loss $\mathcal{L}$, we propose two different aspects: (1) *the statistical aspect*, where the goodness of fit of our explanation function is measured, and (2) the *predictive accuracy* denoting how well the results from the *true* explanation function can be approximated using our explanation function; refer to Section 8 for these metrics.

4.2 Solution Overview

The proposed methodology for computing explanations is split into three *modes* (Figure 4). The *Pre-Processing Mode* aims to identify the *optimal* number of LPMs and an initial approximation of their parameters using previously executed queries. The overarching aim of this mode is to jump-start our framework. In the *Training Mode*, the LPMs' parameters are incrementally optimized to minimize the objective function (2) as incoming queries are processed in an on-line manner. In the *Explanation Mode*, the framework is ready to explain AQ results via the obtained LPMs.

4.2.1 Pre-Processing Mode. A training set $\mathcal{T}$ of m previously executed queries $\mathbf{q}$ and their corresponding aggregate results is used as input to the *Pre-Processing Mode*. The central task is to partition (quantize) the query space $\mathbb{Q}$ based on the observed previous queries $\mathbf{q} \in \mathcal{T}$ into K clusters, sub-spaces $\mathbb{Q}_k$, in which queries with similar centers $\mathbf{x}$ are grouped together. Each cluster is then further quantized into L sub-clusters, as queries with similar centers $\mathbf{x}$ are separated with regards to their $\boldsymbol{\theta}$ parameter values. Therefore, this is an approach of a hierarchical query space

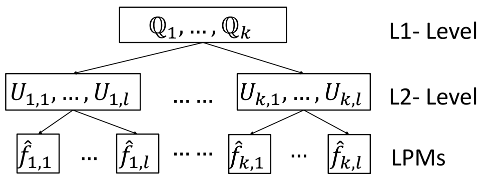

$\mathbb{Q}_1, \dots, \mathbb{Q}_k$
L1- Level
$U_{1,1}, \dots, U_{1,l}$
$U_{k,1}, \dots, U_{k,l}$
L2- Level
$\hat{f}_{1,1}$
$\hat{f}_{1,l}$
$\hat{f}_{k,1}$
$\hat{f}_{k,l}$
LPMs

Fig. 5. The hierarchical hetero-associative learning and quantization scheme provides levels of partitioning for the multi-dimensional query center **x** and region sizes $\boldsymbol{\theta}$. Each LPM is associated and trained with the data points included in a data cluster in L2.

quantization, with the first level partition w.r.t. center **x** and second level partition w.r.t. parameter $\boldsymbol{\theta}$, where each Level-1 (L1) cluster $\mathbb{Q}_k, k = 1, \dots, K$ is associated with a number of Level-2 (L2) sub-clusters $\mathbb{U}_{kl}, l = 1, \dots, L$. For each L1 cluster $\mathbb{Q}_k$ and L2 sub-cluster $\mathbb{U}_{kl}$, we assign an L1 representative, hereinafter referred to as Location Representative (LR) and an L2 representative, hereinafter referred to as Region Representative (RR). The LR converges to the mean vector of the centers of all queries belonging to L1 cluster $\mathbb{Q}_k$, while the associated RR converges to the mean length value of the lengths of all queries, whose lengths values belong to $\mathbb{U}_{kl}$. After the hierarchical quantization of the query space, the task is to *associate an LPM* $\hat{f}_{kl}(\mathbf{q})$ with *each* L2 sub-cluster $\mathbb{U}_{kl}$. This process is nicely summarized in Figure 5. At L1 we have the partitioned Query Space $\mathbb{Q} = \mathbb{Q}_1 \cup \mathbb{Q}_2, \dots, \mathbb{Q}_{k-1} \cup \mathbb{Q}_k$. Each one of the partitions is associated with a sub-cluster at L2 which are also associated with an LPM.

4.2.2 Training Mode. This mode optimally adapts/tunes the clustering parameters (LR and RR representatives) of the pre-processing mode in order to minimize the objective function in (2). This optimization process is achieved incrementally by processing each new pair $(\mathbf{q}_i, y_i)$ in an on-line manner. Consulting Figure 6, in *Training* mode, each incoming query $\mathbf{q}_i$ is projected/mapped to the closest LR corresponding to a L1 cluster. Since, the closest LR is associated with a number of L2 RRs, the query is then assigned to one of those RRs, and then the associated representatives are adapted/fine-tuned. After a pre-specified number of processed queries, the corresponding LPM $\hat{f}_{kl}(\mathbf{q})$ is re-adjusted to account for the newly associated queries.

4.2.3 Explanation Mode. In this mode no more modifications to LRs, RRs, and the associated LPMs are made. Based on the L1/2 representatives and their associated approximation explanation functions LPMs, the model is now ready to provide explanations for AQs. Figure 6 sums up the result of all three modes and how an explanation is given to the user. For a given query **q**, the closest LR;$\mathbf{w}_k$ is initially obtained and then, based on a combination of the RRs;$(\mathbf{u}_{k,1}, \dots, \mathbf{u}_{k,3})$ and their associated LPMs;$(\hat{f}_{k,1} \dots, \hat{f}_{k,3})$, returns an explanation as a fusion of diverse LPMs functions derived by the L2 level. We elaborate on this fusion of L2 LPMs in Section 7.

5 OPTIMIZATION PROBLEMS DECONSTRUCTION

5.1 Optimization Problem 1: Query Space Clustering

The first part of the deconstructed generic problem identifies the need to find *optimal* LRs and RRs, as such optimal parameters guarantee better grouping of queries thus better approximation of *true* function, during the *Pre-Processing* phase. The LRs are initially random location vectors $\mathbf{w}_k \in \mathbb{R}^d, k = 1, \ldots, K$, and are iteratively refined by a clustering algorithm until they converge to the mean vectors of the associated query space $\mathbb{Q}_k$. Formally, this phase finds the optimal mean vectors $\mathcal{W} = \{\mathbf{w}_k\}_{k=1}^{K}$, which minimize the L1 Expected Quantization Error (L1-EQE):

$$\mathcal{J}_1(\{\mathbf{w}_k\}) = \mathbb{E}[\|\mathbf{x} - \mathbf{w}^*\|^2; \mathbf{w}^* = \arg\min_{k=1,\ldots,K} \|\mathbf{x} - \mathbf{w}_k\|^2] \tag{3}$$

where $\mathbf{x}$ is the location of query $\mathbf{q} \in \mathcal{T}$ and $\mathbf{w}_k$ is the mean center vector of all queries $\mathbf{q} \in \mathbb{Q}_k$ associated with $\mathbf{w}_k$. We adopt the K-Means [23] clustering algorithm to identify the L1 LRs based on the queries' centers $\mathbf{x}$ although other clustering algorithms will work as well. The choice of K-Means is mainly due to its scalability to many training examples and its simplicity. A limitation of the K-Means algorithm is that we need to specify the K representatives in advance. Therefore, we devised a simple strategy to find a near-optimal number K. By running the clustering algorithm iteratively, each time increasing the input parameter K for the K-Means algorithm, we are able to find a K that is near-optimal. In this case, an optimal K would *sufficiently* minimize the Sum of Squared Quantization Errors (SSQE), which is equal to the summation of distances, of all queries from their respective LRs. The strategy follows Algorithm 1.

$$\text{SSQE} = \sum_{i}^{n} \min_{\mathbf{w}_k \in \mathcal{W}} (\|\mathbf{x}_i - \mathbf{w}_k\|_2^2) \tag{4}$$

The algorithm is fairly straight-forward and is in-line with a method called the "Elbow Method" in approximating an optimal K for K-means. It essentially performs several passes over the data, applying the algorithm and obtaining an SSQE. It stops when a pre-defined threshold $\epsilon > 0$ has been reached. We note that there are multiple such algorithms available in the literature [22]. However, this is not part of our focused work thus a simple solution was preferred to alleviate such problem.

Algorithm 1 Estimating a near-optimal K

```
Input: $\epsilon$; $K$                                          ▷ initial $K$; predefined improvement threshold.
$\mathcal{W} = \emptyset$; $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^m : \mathbf{x} \in \mathcal{T}$                 ▷ set of LRs; query centers $\mathcal{T}$
while TRUE do
    $\mathcal{W} \leftarrow KMeans(K, \mathcal{X})$                     ▷ call K-Means algorithm with $K$ LRs
    SSQE $\leftarrow \sum_{i=1}^m \min_{\mathbf{w}_k \in \mathcal{W}} (\|\mathbf{x}_i - \mathbf{w}_k\|^2)$            ▷ Calculate SSQE
    if $\Delta|SSQE| > \epsilon$ then                                      ▷ improvement
        $K \leftarrow K + 1$                                         ▷ increase $K$
    else
        break                                      ▷ no more improvement; exit
    end if
end while
Return: $\mathcal{W}$                                                 ▷ set of $K$ LRs
```

We also utilize K-Means over each L2 cluster of queries created by the L1 query quantization phase. Formally, we minimize the conditional L2 Expected Quantization Error (L2-EQE):

$$\mathcal{J}_{1.1}(\{\mathbf{u}_{k,l}\}) = \mathbb{E}[\|\boldsymbol{\theta} - \boldsymbol{u}^*\|^2; \mathbf{u}^* = \arg\min_{l=1,\ldots,L} \|\boldsymbol{\theta} - \mathbf{u}_l\|^2], \tag{5}$$

Therefore, for each LR $\mathbf{w}_1, \ldots, \mathbf{w}_K$, we *locally* run the K-Means algorithm with L number of RRs, where a near optimal value for L is obtained following the same near-optimal strategy. With SSQE being computed using $\mathbf{u}$ and $\boldsymbol{\theta}$ instead of $(\mathbf{w}, \mathbf{x})$. Specifically, we identify the L2 RRs over the lengths of those queries from $\mathcal{T}$ whose closest LR is $\mathbf{w}_i$. Then, by executing the L-Means over the length values from those queries we derive the corresponding set of region representatives $\mathcal{U}_i = \{\mathbf{u}_{i1}, \ldots, \mathbf{u}_{iL}\}$, where each $\mathbf{u}_{il}$ is the mean vector of lengths in the l-th L2 sub-cluster of the i-th L1 cluster. Thus the first part of the deconstructed optimization problem can be considered as two-fold, as we wish to find optimal parameters for both LRs and RRs that minimize (3) and (5).

5.2 Optimization Problem 2: Fitting LPMs per Query Cluster

The second part of the deconstructed generic optimization problem has to do with fitting *optimal* approximation functions such that the local EEL is minimized given the optimal parameters obtained from the first part of the deconstructed problem. We fit PLR functions $\hat{f}_{kl}(\mathbf{q})$ for each L2 sub-cluster $\mathbb{U}_{kl}$. The fitted PLR captures the *local* statistical dependency of the input parameters given that $\mathbf{x}$ is a member of the L1 cluster represented by $\mathbf{w}_k$ and $\boldsymbol{\theta}$ of L2 represented by $\mathbf{u}_{kl}$. Given the objective in (2), for each local L2 sub-cluster, the approximate function $\hat{f}_{kl}$ minimizes the conditional Local EEL:

$$\begin{aligned} \mathcal{J}_2(\{\beta_{kl}, \lambda_{kl}\}) &= \mathbb{E}_{\theta,\mathbf{x}}[\mathcal{L}(f_{kl}(\mathbf{q}), \hat{f}_{kl}(\mathbf{q}))] \\ \text{s.t.} \quad & \mathbf{w}_k = \arg\min_{j\in[K]} ||\mathbf{x} - \mathbf{w}_j||_2^2, \\ & \mathbf{u}_{kl} = \arg\min_{j\in[L]} ||\boldsymbol{\theta} - \mathbf{u}_{kl}||_2^2 \end{aligned} \tag{6}$$

conditioned on the closeness of the query's $\mathbf{x}$ and $\boldsymbol{\theta}$ to the L1 and L2 quantized query space $\mathbb{Q}_k$ and $\mathbb{U}_{kl}$, respectively. Where $\{\beta_{kl}, \lambda_{kl}\}$ are the parameters for the PLR local approximation explanation functions $\hat{f}_{kl}$ defined as follows in (7).

Remark 1. *Minimizing objective $\mathcal{J}_2$ in (6) is not trivial due to the double conditional expectation over each query center and length. To initially minimize this local objective, we adopt the Multivariate Adaptive Regression Splines (MARS) [21] as the approximate model explanation function $\hat{f}_{kl}$. MARS is capable of capturing non-linearities in the parameter $\boldsymbol{\theta}$ space conditioned around a given location vector $\mathbf{w}$ by introducing hinge point functions $h_c(x) = \max(0, x - c)$; c is constant.*

Based on this adoption, thus, our approximate $\hat{f}_{kl}$ has the following form:

$$\hat{f}_{kl}(\mathbf{q}) = \beta_0 + \sum_{i=1}^{M} \beta_i h_i(\mathbf{q}), \tag{7}$$

where $h_i(\mathbf{q})$ are basis hinge functions identified by a forward stepwise procedure. Essentially, this creates M regression functions. The number M of linear regression functions is automatically derived by MARS using a threshold for convergence w.r.t R^2(*coefficient-of-determination*, later defined); which optimize fitting. Thus, guaranteeing an optimal number of M linear regression functions. For each L2 sub-cluster, we fit L MARS functions $\hat{f}_{kl}, l = 1, \ldots, L$, each one associated with an LR and an RR, thus, in this phase we initially fit $K \times L$ MARS functions for providing explanations for the whole query space. Figure 6 illustrates the two levels L1 and L2 of our explanation methodology, where each LR and RR are associated with a MARS model.

5.2.1 Using alternatives to LPMs. As described the used model is MARS, which empirically performs really well and has also desirable properties. We can derive the importance of features by their use in basis functions and their coefficients. Because of its building procedure it can also eliminate terms that do not increase predictive power. Thus the analyst can infer which parameters are not crucial in producing the output of aggregate functions. However, the model choice should not be restrictive. Alternative models that provide similar functionality can be used as well. Regression trees [33] can also provide similar functionality as the importance of each parameter can be inferred by its use in branches. In addition, simple Linear Regression models and their variants Ridge [24], Lasso [20] are also good candidates as their coefficients provide intuitive explanations. Although in this case increased quantization might be needed as the obtained clusters might be non-linear wrt the input parameters and output.

5.2.2 Overfitting of LPMs. There is a possibility of the LPMs overfitting under certain subspaces. In this context, overfitting may be desirable as it essentially means that the functions have learned to accurately mimic the true underlying function. However, it might limit the LPMs potential in generalizing to new (unseen) query parameters. To control overfitting, standard regularization practises of the underlying models should be used in this case. For instance, if we were to use Linear Regression models as an alternative model to MARS, we could regularize its structure by using L_p norm penalization on its coefficients to prevent overfitting, where usually $p = 1, 2$ with its variants labeled as Lasso and Ridge regression [48]. In MARS, overfitting is prevented by a backward procedure in which terms/knots are removed from the function. Hence, by choosing to adopt well known function as our models of choice we can rely on these models to effectively prevent overfitting.

5.3 Optimization Problem 3: Putting it All Together

The optimization objective functions in (3), (5) and (6) should be combined to establish our generic optimization function $\mathcal{J}_0$, which involves the estimation of the *optimal* parameters that minimize the EQE in $\mathcal{J}_1$ (L1) and $\mathcal{J}_{1.1}$ (L2), and then the conditional optimization of the parameters in $\mathcal{J}_2$. In this context, we need to estimate these values of the parameters in $\mathcal{W} = \{\mathbf{w}_k\}$ and $\mathcal{U} = \{u_{kl}\}$ that minimize the EEL given that our explanation comprises a set of regression functions $\hat{f}_{kl}$. Our joint optimization is optimizing *all* the parameters from $\mathcal{J}_1$, $\mathcal{J}_{1.1}$, and $\mathcal{J}_2$ from Problems 1 and 2:

$$\mathcal{J}_3(\mathcal{W}, \mathcal{U}, \mathcal{M}) = \mathcal{J}_1(\mathcal{W}) + \mathcal{J}_{1.1}(\mathcal{U}) + \mathcal{J}_2(\mathcal{M}) \tag{8}$$

with parameters:

$$\mathcal{W} = \{\mathbf{w}_k\}, \mathcal{U} = \{\mathbf{u}_{kl}\}, \mathcal{M} = \{(\beta_i, \lambda_i)_{kl}\} \tag{9}$$

with $k \in [K]$, $l \in [L]$, $i \in [M]$, which are stored for fine tuning and explanation/prediction.

Remark 2. *The optimization function $\mathcal{J}_3$ approximates the generic objective function $\mathcal{J}_0$ in (2) via L1 and L2 query quantization (referring to the integral part of (2)) and via the estimation of the local PLR functions referring to the family of function space $\mathcal{F}$. Hence, we hereinafter contribute to an algorithmic/computing solution to the optimization function $\mathcal{J}_3$ approximating the theoretical objective function $\mathcal{J}_0$.*

6 STATISTICAL LEARNING METHODOLOGY & EXPLAINABILITY

We propose a new statistical learning model that associates the (hierarchically) clustered query space with PLR-based explanation functions. Given the hierarchical query space quantization and local PLR fitting, the training mode fine-tunes the parameters in (9) to optimize both $\mathcal{J}_1$, $\mathcal{J}_{1.1}$ in (3), (5) and $\mathcal{J}_2$ in (6). The three main sets of parameters $\mathcal{W}$, $\mathcal{U}$, and $\mathcal{M}$ of the framework are *incrementally* trained in *parallel* using queries issued against the underlying data management

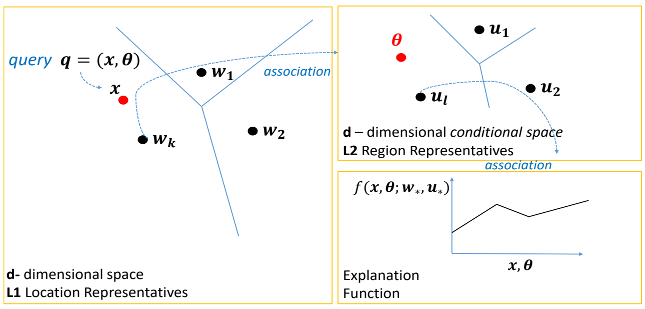


Fig. 6. Illustration of the hetero-association rationale: (i) the query **q** is projected to the closest L1 cluster representative; (ii) then, its parameter $\boldsymbol{\theta}$ is mapped to the closest L2 sub-cluster conditioned on the L1 LR; (iii) then, this is associated with an explanation function f. (iv) The explanation of the result is provided by the associated LPM over the query parameters $\boldsymbol{\theta}$ and $\mathbf{x}$.

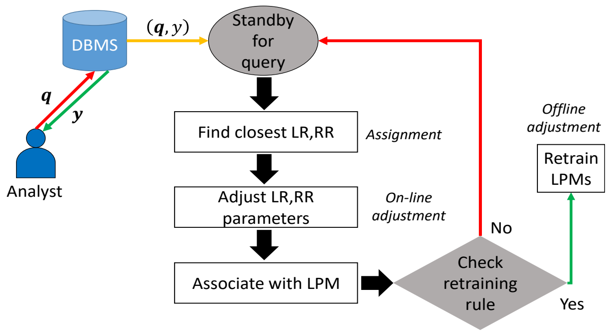


Fig. 7. The training phase: A query-result pair is initially associated with an LR, RR and LPM. The LR and RR parameters are adjusted online as each query is processed. An offline adjustment step is triggered based on a threshold.

system. These queries are issued online and the incremental process is as follows : (i) the analyst issues an AQ $\mathbf{q} = [\mathbf{x}, \boldsymbol{\theta}]$; and (ii) the DBMS answers with result y ; (iii) our framework exploits pairs $(\mathbf{q}, y)$ to train its new statistical learning model.

The Training phase follows three steps and the complete process is shown at Figure 7. First, in the *assignment* step, the incoming query **q** is associated with an L1, L2 and LPM, by finding the closest representatives at each level. Next we perform an *on-line adjustment* step, in which the LR and RR are gradually modified w.r.t. the associated incoming query in the direction of minimizing the said objectives. Finally, the *off-line adjustment* step conditionally fine-tunes any LPM associated with any incoming query so far. This step is triggered when a predefined retraining value is reached or the number of queries exceeds a threshold.

6.1 Query Assignment Step

For each executed query-answer pair $(\mathbf{q}, y)$, we project **q** to its closest LR using only the query center **x** based on (3). Obtaining the projection $\mathbf{w}_k^*$ allows us to *directly* retrieve the associated RRs $\mathcal{U}_k^* = \{\mathbf{u}_{1k}^*, \ldots, \mathbf{u}_{Lk}^*\}$. Finding the best RR in $\mathcal{U}_k^*$ is, however, more complex than locating the best LR. In choosing one of the RRs, we consider both *distance* and the associated *prediction error*. Specifically, the *prediction error* is obtained by the LPM of each RR from the set $\mathcal{U}_k^*$. Hence, in this context, we first need to consider the distance of our query $\mathbf{q}$ to all of the RRs in $\mathcal{U}_k^*$:

$$||\boldsymbol{\theta} - \mathbf{u}_{kl}||_1, \forall \mathbf{u}_{kl} \in \mathcal{U}_k^*, \tag{10}$$

and, also, the *prediction error* given by each RR's associated LPM $\hat{f}_{kl}$. The prediction error is obtained by the squared difference of the actual result y and the predicted outcome of the LPM $\hat{y} = \hat{f}_{kl}(\mathbf{q})$:

$$(y - \hat{f}_{kl}(\mathbf{q}))^2, l = 1, \ldots, L \tag{11}$$

Therefore, for assigning a query $\mathbf{q}$ to a RR, we combine both distances in (10) and (11) to get the assignment distance in (12), which returns the RR in $\mathcal{U}_k^*$ which minimizes:

$$l^* = \arg\min_{l \in [L]} \{z||\boldsymbol{\theta} - \mathbf{u}_{kl}||_1 + (1 - z)(y - \hat{f}_{kl}(\mathbf{q}))^2\} \tag{12}$$

The parameter $z \in (0, 1)$ tilts our decision towards the *distance*-wise metric or the *prediction*-wise metric, depending on which aspect we wish to attach greater significance.

Remark 3. *Why incorporate prediction error? We could associate an incoming query with the closest L2 RR as is being done with L1 LR. However, note that an explanation function may have lower prediction error even though is not the closest (w.r.t to RR). Intuitively, this holds true, as some function might be able to make better generalizations even if their RRs are further apart. Therefore, we introduce the weighted-distance in (12) to account for this and make more sophisticated selections.*

6.2 On-line Representatives Adjustment Step

This step optimally adjusts the positions of the chosen LR and RR so that training is informed by the new query. Their positions are shifted using Stochastic Gradient Descent (SGD) [12] over the $\mathcal{J}_1$ and $\mathcal{J}_2$ w.r.t. $\mathbf{w}$ and $\boldsymbol{\theta}$ variables in the negative direction of their gradients, respectively. This ensures the *optimization of both objective functions.* Theorems 1 and 2 present the update rule for the RR selected in (12) to minimize the EEL given that a query is projected to its L1 LR and its convergence to the median value of the radii of those queries.

Theorem 1. *Given a query $\mathbf{q} = [\mathbf{x}, \boldsymbol{\theta}]$ projected onto the closest L1 $\mathbf{w}_{k^*}$ and L2 $\mathbf{u}_{k^*,l^*}$, the update rule for $\mathbf{u}_{k^*,l^*}$ that minimizes $\mathcal{J}_2$ is:*

$$\Delta\mathbf{u}_{k^*,l^*} \leftarrow \alpha z sgn(\boldsymbol{\theta} - \mathbf{u}_{k^*,l^*}) \tag{13}$$

Proof. We adopt the Robbins-Monro stochastic approximation for minimizing the combining distance-wise and prediction-wise loss given a L2 RR u, that is minimizing $\mathcal{E}(\mathbf{u}) = \mu||\boldsymbol{\theta} - \mathbf{u}||_1 + (1 - \mu)(y - \hat{f}(\boldsymbol{\theta}; \mathbf{w}))^2$, using SGD over $\mathcal{E}(u)$. Given the t-th training pair $(\mathbf{q}(t), y(t))$, the stochastic sample $E(t)$ of $\mathcal{E}(\mathbf{u})$ has to decrease at each new pair at t by descending in the direction of its negative gradient with respect to $\mathbf{u}(t)$. Hence, the update rule for L2 RR u is derived by:

$$\Delta\mathbf{u}(t) = -\alpha(t)\frac{\partial E(t)}{\partial \mathbf{u}(t)},$$

where scalar $\alpha(t)$ satisfies $\sum_{t=0}^{\infty} \alpha(t) = \infty$ and $\sum_{t=0}^{\infty} \alpha(t) < \infty$. From the partial derivative of $E(t)$ we obtain $\Delta\mathbf{u} \leftarrow \alpha\mu\text{sgn}(\boldsymbol{\theta} - \mathbf{u})$. By starting with arbitrary initial training pair $(\mathbf{q}(0), y(0))$, the sequence $\{u(t)\}$ converges to optimal L2 RR u parameters. □

$\alpha \in (0,1)$ is the learning rate defining the shift of θ ensuring convergence to optimal position and $sgn(x) = \frac{d|x|}{dx}, x \neq 0$ is the signum function. Given that query $\mathbf{q}$ is projected on L1 $\mathbf{w}^*_k$ and on L2 $\mathbf{u}_{k^*,l^*}$, the corresponding RR converges to the local median of all radius values of those queries.

Theorem 2. *Given the optimal update rule in (13) for L2 RR $u_{k,l}$, it converges to the median of the θ values of each dimension of those queries projected onto the L1 query subspace $\mathbb{Q}_k$ and the L2 sub-cluster $\mathbb{U}_{kl}$, i.e., for each query $\mathbf{q} = [\mathbf{x}, \boldsymbol{\theta}]$ with $\mathbf{x} \in \mathbb{Q}_k$, it holds true for $u_{kl} \in \mathbb{U}_{kl}$ that: $\int_0^{u_{kl}} p(\theta|\mathbf{w}_k)d\theta = \frac{1}{2}$.*

Proof. Focus on one dimesion, and consider the optimal update rule in (13) for L2 RR u and suppose that u has reached equilibrium, i.e., $\Delta u = 0$ holds with probability 1. By taking the expectations of both sides and replacing Δu with the update rule from Theorem 1:

$$\mathbb{E}[\Delta u] \quad = \quad \int_{\mathbb{R}} \mathrm{sgn}(\theta - u)p(\theta)\mathrm{d}\theta = P(\theta \geq u)\int_{\mathbb{R}} p(\theta)\mathrm{d}\theta - P(\theta < u)\int_{\mathbb{R}} p(\theta)\mathrm{d}\theta = 2P(\theta \geq u) - 1.$$

Since $\Delta u = 0$ thus u is constant, then $P(\theta \geq u) = \frac{1}{2}$, which denotes that u converges to the median of radii for those queries represented by L1 RL and the associated L2 RR. □

Using SGD, $\boldsymbol{\theta}_{k^*,l^*}$ converges to the median of all radius values of all queries in the local L2 sub-cluster in an on-line manner.

6.3 Off-line Adjustment Step

. The mini-batch adjustment step is used to conditionally re-train the LPMs to reflect the changes by (13) in $\mathcal{U}_k$ parameters. As witnessed earlier, representatives are incrementally adjusted based on the projection of the incoming query-answer pair onto L1 and L2 levels. For the LPMs, the adjustment of hinge points and parameters (β_i, λ_i) needs to happen in mini-batch mode taking into consideration the projected incoming queries onto the L2 level. To achieve this, we keep track of the number of projected queries on each L2 sub-cluster $\mathcal{U}_k$ and re-train the corresponding LPMs $\hat{f}_{kl}$ given a conditionally optimal L2 RR $\mathbf{u}_{kl}$. For every processed query we increment a counter. Once we reach a predefined number of projected queries-answers, we re-train every LPM that was affected by projected training pairs.

Remark 4. *Why Pre-Processing and Training Modes ? A concrete explanation as to why these two modes need to co-exist is provided in the remark. One might notice that with just the Pre-Processing step, we could achieve a good accurate estimation by creating good enough LPMs to explain new possible queries. However, with a Pre-Processing step we eliminate the possibility of adjusting as new query-answers pairs are processed. In addition, it does not take into consideration the prediction error, thus, it might lead to inaccurate models, which only consider the similarity of query patterns and not the predictive power of LPMs. Moreover, the Training mode cannot exist on its own as the LPMs need a number of queries to be initialized, thus, the need for a Pre-Processing step. An additional benefit of introducing the Training Mode is that both the representatives and their associated models can be adapted as new queries are executed. This also accounts for cases in which a new stream of queries, that is wildly dissimilar to current representatives, arrives to the data system. This might be an indication of a shift to the underlying distribution of* $\mathbf{q}$. *However, because of the Training Mode, the query parameters can be incrementally adjusted and adapted to the new distribution. A new distribution might signify new user interests and application requirements. Studying the full effects of this has been studied in the context of Approximate Query Processing [44–46], but not in the context of explanation functions, which remains part of our future work.*

7 EXPLANATION SERVING

After *Pre-Processing* and *Training*, explanations can be provided for *unseen* AQs, i.e., AQs which have never been used before in the system's training or executed before over the system's data. The explanations are based on the associated LPM found using the AQ's input parameter values. The process of returning an explanation function to the user is as follows: firstly, the analyst issues a query $\mathbf{q}$, $q \notin \mathcal{T}$, then, the closest LR, RR representatives are located, and finally, the associated LPM is returned to the analyst used for explanation, as will be discussed later.

As with *Training Mode*, the closest L1 representative is identified using Eq. 14 and for the closest L2 representative Eq. 12 is used. As mentioned, the L2 representatives are associated with an LPM and, thus, the related LPM is returned to the analyst.

$$\mathbf{w}^* = \arg\min_{i \in K} ||\mathbf{w}_i - \mathbf{x}||_2^2 \tag{14}$$

This explanation function serves as an approximation of how the result of an unseen AQ varies in that subspace queried by the original AQs. The explanation function coefficients (or feature importance methods of various ML models) gives us a way to infer how the different query parameters contribute to the end result. This is an extremely hard thing to do with only the knowledge provided to us by coarse grained queries executed over the complete data space. We are essentially trying to infer the parameters' importance over much smaller subspaces by utilizing previous query patterns. The main difficulty arises because when the LPMs have finished training, they have limited knowledge over a specific subspace and some might even ignore the knowledge coming from queries executed over that specific subspace. To be more concise, imagine single dimension queries $q \in \mathbb{R}$, where q could be a single input parameter with a true function mapping to results $y \in \mathbb{R}$. Now imagine that q_1, q_2, q_3, are three queries $q_1, q_2, q_3 \in [0, 1]$ such that $q_1 \leq q_2 \leq q_3$. Training our model with those queries leaves our model with high uncertainty as to what happens in the in-between spaces where we have no training examples. Trying to approximate what happens within those spaces requires good generalizability from the trained models as some models might even ignore some of the examples. In our work, this is exactly what we are trying to do. With minimal knowledge given from the coarse grained training examples we are trying to infer the impact of parameters in much smaller unknown spaces. As shown in our experimental section, our models overcome this difficulty and are able to find approximate functions to explain what happens over the smaller subspaces.

7.1 Examples of Explanation Functions and Practical Usage

Using the returned LPM, the analyst can obtain the result of queries not yet executed and infer the importance of query parameters under the given subspace. In this section we provide some examples of how the different LPMs can be utilized to provide insight and guide the analyst during data exploration.

In Figure 8 we have created an interactive plot of the different query parameters listed as features and their importance across different clusters using Bokeh [11]. This overview is constructed by averaging the importance of all LPMs associated with L2 clusters and then shown for each L1 cluster. By hovering over a particular rectangle the analyst can see the importance of a parameter over the specified cluster. As evident, the importance of parameters varies across clusters which can lead the analyst to specific subspaces where the parameter is more important. For instance, if a particular parameter has more weight in a given cluster then the analyst knows that varying this parameter will have more impact to the end result than other parameters. In essence, the analyst has a guide to inform them which parameters to shift in certain subspaces such that they get meaningful results and not waste resources executing queries that will not impact the result.

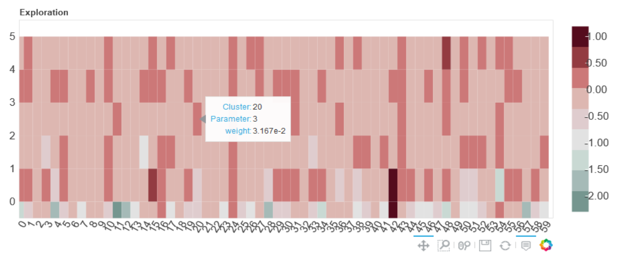


Fig. 8. Analyst's interactive exploration of data (sub)spaces by visualising the importance of parameters in different (sub)spaces.

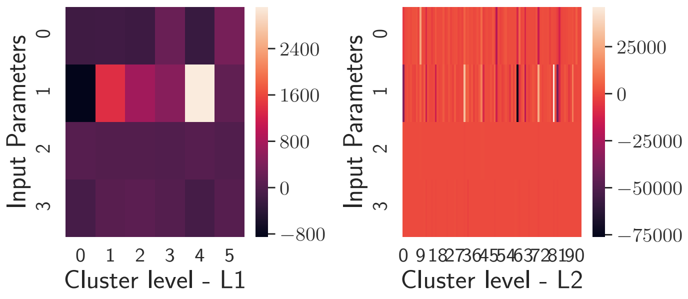


Fig. 9. Hierarchical representation of L1 and L2 clusters and the parameter importance. (Left) Clusters for query centers **x** and the average importance of parameters inferred by the underlying LPMs. (Right) An L2 cluster associated with Cluster 0 from L1.

Moreover, we can construct hierarchical visualizations that show a more fine grained representation of parameter importance. For this example, we used the coefficients of linear regression models for explanation functions. Their main difference with parameter/feature importance is that they also indicate in what way they can influence the result as they can be negative or positive. For instance, a large negative coefficient for a specific parameter as seen for the input parameter 1 at L1=0 in Figure 9 (left) advises us that this particular parameter will have a negative impact on the end query result. In Figure 9(left) we can see the average coefficients for all parameters over clusters at L1. Using such visualisation we obtain an overview of the coefficients associated with each parameter and can also zoom-in to a particular cluster of L1 and see how the coefficients change at L2 clusters associated with the chosen L1 cluster. In this case study, the parameters $0 - 3$ correspond to the 0-*X Coordinate*, 1-*Y Coordinate*, 2-*Size of X Coordinate*, 3-*Size of Y Coordinate*. These parameters define a range query that cover a spatial region on the map. The corresponding response variable of the LPMs built is the *Arrest Rate* within those areas. Therefore, an analyst can begin their analysis by consulting this figure. They can initially focus on cluster 0 at L1 which shows the lowest average coefficient for parameter 1-*Y Coordinate*. The fine grained view shown at Figure 9(right) corresponds to the coefficients for all input parameters across all clusters at L1-0. The analyst might then notice that all parameters have positive coefficients except parameter 1 especially over cluster 47 at L2.

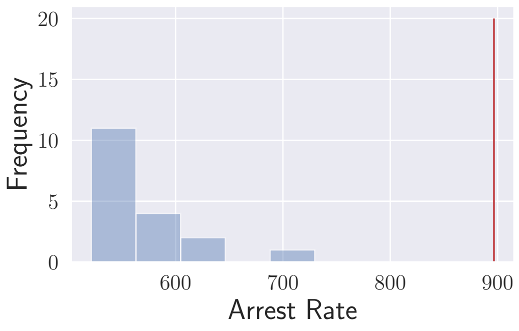


Fig. 10. The average Arrest Rate shown as a red vertical line along with the distribution of arrest rate for queries executed over the region identified by 0-L1 and 47-L2.

They can then obtain the actual LPM model and mean vectors for clusters 0 at L1 and 47 at L2. The mean vectors correspond to the cluster representatives (centroids) and together they denote the mean (average) query executed at those regions. Hence, they can identify a possible region to which the arrest rate is smaller than the average arrest rate over other regions *without* executing anything. The mean arrest rate is visualised in Figure 10 along with the distribution of arrest rates over the pinpointed region. Using our methodology, the analysts have specific knowledge of the area with smaller than the mean arrest rates. By obtaining the coefficients for the corresponding LPM at L1-0 and L2-47, they can observe what drives the arrest rate up or down. The corresponding coefficients in this particular case were $[621.64900659, -455.21564492, 34.93227467, -246.40730294]$ which indicates that as we increase the *Y Coordinate* moving up north on the map, the arrest rate decreases. This is also indicated by the region size covered by the *Y Coordinate*. Using this derived knowledge, the analyst can further investigate as to why that happens. Maybe the number of police officers patrolling the areas has dropped, or maybe the incidents recorded did not warrant an arrest.

As one can observe, without even executing a single query, the analyst has tremendous knowledge which can guide them when they first initiate their exploratory analysis. Surely, the LPMs used were trained using queries which were executed before. However, these queries can be obtained from past analyses made by other analysts. They could have also been obtained from other visualisations previously constructed that have been used for these core operations. Therefore, the analysts can save huge amount of time by focusing on particular areas instead of starting with a coarse grained view with no tool to guide them.

8 EXPERIMENTAL EVALUATION

We run experiments to evaluate the accuracy, performance and scalability of our proposed solution. Accuracy is evaluated on two axes: (1) *predictive capability* of the proposed explanation function by using the function as a surrogate black-box function to compute the result of an unknown query, and (2) *accuracy* of the estimated explanation functions. We conduct experiments to analyse the performance of our system, measuring the time required for training the different LPMs, and the impact of the hyper-parameters in explanation serving.

8.1 Experimental Setup & Metrics

8.1.1 Real Data Sets: We experiment with two real data sets. The first real dataset $\mathcal{B}_1 = \{\mathbf{a}_i\}_{i=1}^N, \mathbf{a} \in \mathbb{R}^2$ with cardinality $|\mathcal{B}_1| = N = 6 \cdot 10^6$ contains crimes for the city of Chicago [1]. We obtain a synthetic workload $\mathcal{T}$ containing $m = 5 \cdot 10^4$ queries and their answers, i.e., $\{(\mathbf{q}, y)_i\}_{i=1}^m = \mathcal{T}$. Each query is a 4-d vector $\mathbf{q} = [\mathbf{x}, \boldsymbol{\theta}]$ with answer y where $\mathbf{x} = [x_1, x_2] \in \mathbb{R}^2$ is the center, $\boldsymbol{\theta} \in \mathbb{R}^2$ is its length and $y \in \mathbb{R}$ the result obtained from executing the query against real dataset $\mathcal{B}_1$. The second real dataset $\mathcal{B}_2$, which is referred to as the Higgs [3, 10] dataset, contains information about kinematic properties measured by particle detectors in accelerators. These data are used in specific signal-versus-background classification problems in High Energy Physics. More information about the Higgs dataset is provided in Section 8.2.4.

8.1.2 Query Workloads: We use workload $\mathcal{T}$ for *Pre-Processing* and *Training* and create a *separate* evaluation set $\mathcal{V}$ containing $|\mathcal{V}| = 0.2 \cdot m$ new query-answer pairs. The synthetic query workloads were obtained from [2] and follow a similar generation process as described in [6, 43] also described at [2].

To implement our algorithms, we used `scikit-learn` [40] , `KMeans` [23], and an implementation of the `MARS` algorithm [21]. We performed our experiments on a desktop machine with a Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz and 16GB RAM. MARS was used as the explanation function algorithm because of its superior accuracy to simple linear regression. We note that any ML algorithm, which is able to provide an estimate on feature importance to characterize the importance of input parameters is a viable alternative to MARS.

8.1.3 Evaluation Procedure. To evaluate the effectiveness of our proposed explanation functions on an unknown set of AQs, we have to generate perturbed versions of each query $\mathbf{q} \in \mathcal{V}$. Assessing the explanation functions just based on set $\mathcal{V}$ is not enough as the metrics will report on the accuracy of performing point estimates using the explanation functions. Instead, by perturbing a query vector, we are able to assess whether the proposed explanation function is an an accurate approximation of the black-box function generating the results of queries. For a query in $\mathcal{V}$, we generate a set of perturbations $\mathcal{P} = \{\mathbf{p}_1, \ldots, \mathbf{p}_n\}$ with $n = 100$, where $\mathbf{p} \in [-1, 1]^d$. Using these perturbations, we obtain a set of *sub*-queries $\mathcal{S}$, where $\mathcal{S} = \{\mathbf{q} + \mathbf{p}_1, \ldots, \mathbf{q} + \mathbf{p}_n\}$ and $|\mathcal{S}| = |\mathcal{P}|$. A number of *sub*-query sets can then be generated which is equal to the number of queries in $\mathcal{V}$, and $\{\mathcal{S}_1, \ldots, \mathcal{S}_{|\mathcal{V}|}\}$. The results for each one of those perturbed versions of query $\mathbf{q}$ are computed using the real dataset. Such that, we have each perturbed version which is now also associated with a result $y \in \mathbb{R}$. We can then evaluate the explanation function over each sub-query set $\mathcal{S}_i$ by measuring the loss (EEL) between our predictions and the actual responses. The evaluation metrics are computed for the recorded responses and the responses obtained from the proposed explanation function. We then report on an average of the evaluation metrics.

8.1.4 Evaluation & Performance Metrics. In this section we report on the statistical metrics that evaluate the quality of explanation from information theoretic and predictive accuracy perspective.

Information Theoretic Metric: The EEL is measured using the Kullback–Leibler (KL) divergence. Concretely, the result is a scalar value denoting the amount of *information loss* when one chooses to use the approximated explanation function for a given unseen query (without executing this query) instead of the actual explanation function (after executing the unseen query). The EEL with KL divergence is then defined as:

$$\mathcal{L}(f, \hat{f}) = KL(p(y)||\hat{p}(y)) = \int p(y) \log \frac{p(y)}{\hat{p}(y)} dy, \tag{15}$$

with $p(y)$ and $\hat{p}(y)$ being the probability density functions of the true and approximated query result, respectively. We require this divergence to be as small as possible to reflect a good approximation of the explanation funtions.

Goodness of Fit: The EEL is measured here using the coefficient of determination R^2. This metric indicates how much of the *variance* generated by f can be explained using the approximated explanation function $\hat{f}$. This represents the goodness-of-fit of an approximated explanation function over the actual one:

$$R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \overline{y})^2}, \tag{16}$$

in which the denominator is proportional to the *variance* of the *true* function and the numerator are the residuals of our approximation. We require that R^2 should get a relatively high value (close to unity) to indicate good explanation fit over the underlying data sub-region. The EEL between f and $\hat{f}$ can then be expressed as:

$$\mathcal{L}(f, \hat{f}) \quad = \quad 1 - R^2 \tag{17}$$

Predictive Accuracy: We also quantify the predictive accuracy associated with explanation (regression) functions. We employ the *Normalized-Root-Mean-Squared-Error(NRMSE)* for this purpose:

$$NRMSE = \frac{1}{y_{max} - y_{min}}(\frac{1}{n}\sum_{i}^{n}(y_i - \hat{y}_i)^2). \tag{18}$$

Essentially, this shows how accurate the results would be if an analyst used the explanation (regression) functions for further data exploration, without executing queries. We require a small NRMSE to reflect high predictive accuracy. This is evaluated on set $\mathcal{V}$ as we want to evaluate the accuracy on point estimates.

8.1.5 Comparative Methods & Baselines. We provide a performance comparative assessment with baseline models and regression methods.

Local Method: The Local model is trained using Linear Regression and the perturbed set of queries and answers $\mathcal{S}$, hence, it is the gold standard that we are trying to achieve. Note that obtaining a Local model is not actually possible as one has to execute a number of perturbed versions of a query to be able to obtain it. We are merely constructing such local functions to provide an estimate of the highest possible accuracy that can be obtained.

Global Method: The Global model is obtained by training a single Linear Regression model on $\mathcal{T}$ and is the *baseline* method that we are trying to compare with.

***p*-order Polynomial Method:** The p-order polynomial (non-linear) regression functions indicated by LR(p) with $p > 0$ are used at Section 8.2.4 and are obtained over the training set defined over the associated data set that is utilized under this experiment for different p order values. As it will be shown later, these non-linear methods do not provide flexibility and interpretability in explanation serving w.r.t. predictive accuracy and goodness of fit.

Local Piecewise linear Models (LPM): LPM is our proposed approach, which utilizes the procedure outlined in the earlier sections.

8.2 Experimental Results: Accuracy

For our experiments we chose to show performance and accuracy results over three representative aggregate functions: `COUNT`, `AVG` and `SUM` due to their extensive use and because of their properties. We compare the accuracy across the different workloads each one with different distributions for

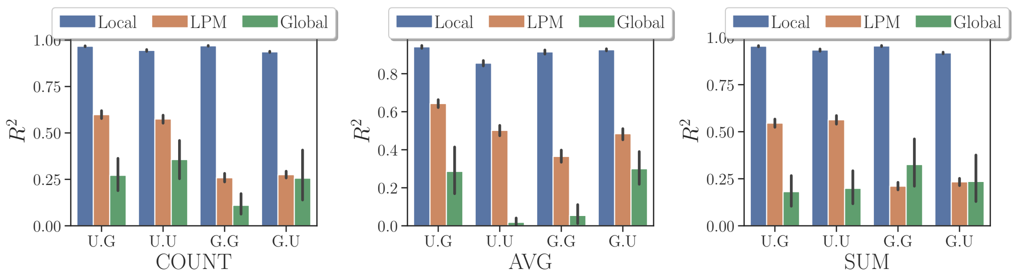


Fig. 11. Goodness of fit results (R^2) for all methods on aggregation operators over $\mathcal{B}_1$; higher is better.

region location and region size. Their abbreviations are shown at Table 1. We compare our LPMs with two baselines **Local** and **Global**.

	Distribution for $\mathbf{x}$	Distribution for $\boldsymbol{\theta}$
G.G	Gaussian	Gaussian
G.U	Gaussian	Uniform
U.G	Uniform	Gaussian
U.U	Uniform	Uniform

Table 1. Abbreviations for the different workloads distribution over query parameters.

8.2.1 Goodness of Fit Results. Figure 11 shows the results for R^2 over all workloads for all three aggregates. We report on the *average* R^2 found by evaluating the models on queries in $\mathcal{V}$ using their pertubed sets $\mathcal{S}$. As expected the Local models perform best as their ttrained using the queries in the pertubed sets. Overall, we see that our proposed explanation functions approaches the accuracy of the **Local** models and is more accurate than the **Global** function, except for one case in which both the LPM and **Global** achieve statistically similar accuracy. It might be the case that the knowledge obtained from the queries was not enough such that our LPMs would be able to approximate the general trend followed by these queries in more specific subsets. We also notice some difference in accuracy levels across different workloads. This leads us to argue that the distribution of queries is a significant factor in determining the accuracy of the proposed models. However, the general outtake of this experiment is that we can fit good approximate models that approach the accuracy of models fitted with the pertubations themselves.

8.2.2 Information Theoretic Results. Figure 12 shows the ratio of bit increase when using the approximated distribution generated by the LPMs rather than the actual values for y of the perturbations (where lower is better). This indicates the inefficiency caused by using the approximated explanations, instead of the *true* function. This information, theoretically, allows us to make a decision on whether using such an approximation would lead to the propagation of errors further down the line of our analysis. We observe that overall the ratio is low for all methods with the Local method approximating the true distribution with minimal loss of information. It is important to note that all results are less than 10% with most of the loss incurred over the AVG function. We notice that sometimes the **Global** function outperforms our LPMs. However, the difference in all of these cases is minimal. We speculate that the Global model obtains a coarse grained view of how the aggregate statistic across a much larger space which could tend to approximate the underlying

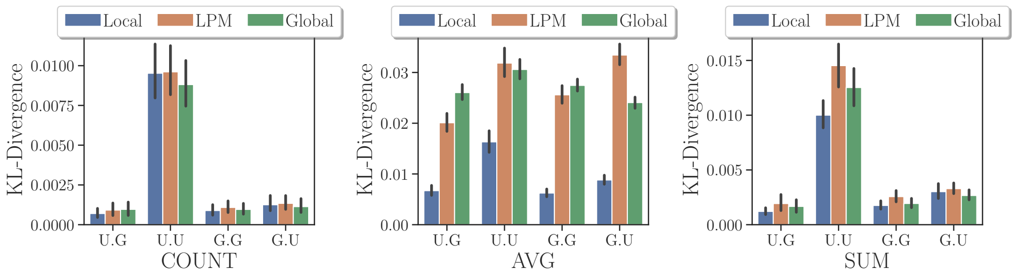


Fig. 12. Information theoretic results (KL-Divergence) for all methods on aggregation operators over $\mathcal{B}_1$; lower is better.

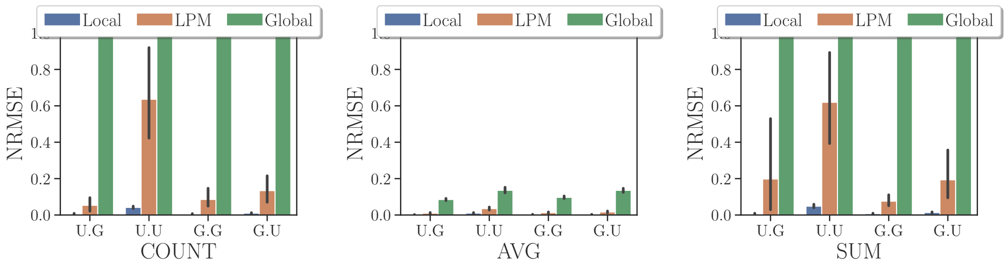


Fig. 13. Predictive accuracy results (NRMSE) for all methods on aggregation operators over $\mathcal{B}_1$; lower is better.

distribution for the pertubed queries. These results serve as evidence that using our approximations is sufficient for conducting further analyses by using these models instead of executing subsequent queries to uncover more information.

8.2.3 Predictive Accuracy Results. We have also measured the predictive accuracy of all methods across all datasets to quantify how good the models would be in predicting the results of future queries. The results are shown at Figure 13. As expected, the error is the lowest for the Local method as the predictions are being made on the set of queries that the model was trained on. The highest error was recorded by the Global method. Apart from AVG, the Global method has extremely large NRMSE which makes it inappropriate to make predictions for the perturbations, thus, it should not be used in this case. We also note that across all three aggregates the U.U distributions appears the hardest to perform predictions. One possible explanation is that models trained on uniform workloads require more queries as they have to be equally good uniformly across the space that they are being evaluated. Their difference with workload distributions having a Gaussian distribution is that queries are executed with a higher probability closer to the mean and thus the patterns might be easier to learn. This experiment also shows that no one model is fit for all problems. Something aligned with the *No Free Lunch* theorem. So in this case it might be more appropriate to investigate different models (with the same properties of interpretability) to be used for different distributions. In conclusion, the accuracy across statistics/aggregates and workloads is not preventive. It shows that our LPMs could in the context of providing approximate answers to queries that analysts might have while conducting their analysis. This could save up computational/monetary/productivity

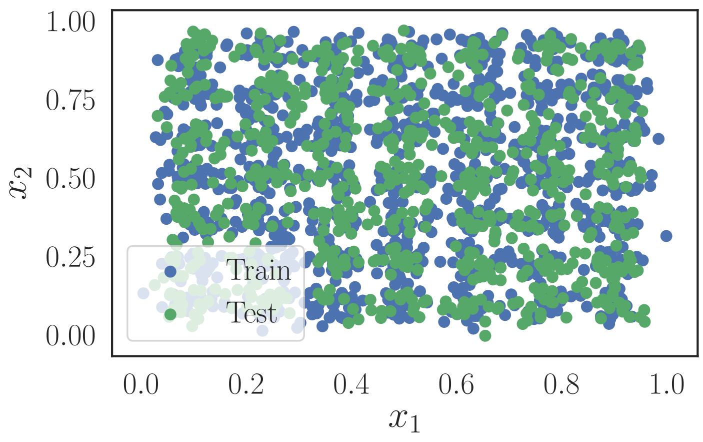


Fig. 14. Query parameter values x_1, x_2 of workload queries executed (training and testing sets) for explanation of the AVG operator for signal classification over the *Higgs* ($\mathcal{B}_2$) data set.

costs, as the analysts would be using the LPMs to get answers efficiently without issuing queries against *any* of the data.

8.2.4 Accuracy Results on Higgs. We also evaluated and compared the methods on the $\mathcal{B}_2$ real data set Higgs [3, 10], which contains $|\mathcal{B}_2| = 11 \cdot 10^6$ rows (data vectors). The individual rows are signals recorded and have a `label`$= \{0, 1\}$ which denotes whether the signal produced Higgs bosons or it was background noise. In this experiment, we constructed queries using the columns (attributes related to energy magnitudes) `m_bb, m_wwbb, label` of the Higgs dataset. We constructed a number of synthetic queries that restricted the rows of the complete data set within the range of:

$$(\texttt{m\_bb}_L \leq \texttt{m\_bb} \leq \texttt{m\_bb}_U) \wedge (\texttt{m\_wwbb}_L \leq \texttt{m\_wwbb} \leq \texttt{m\_wwbb}_U) \tag{19}$$

where the subscripts L, U indicate the lower and upper bounds of the predicate values for the specified columns, respectively [3]. These values were selected randomly to cover the domain space of the selected columns. The aggregate statistic that we used was the AVG over the `label` attribute, such that $y = \frac{1}{|\mathcal{D}_{\mathbf{x},\theta}|} \sum_{i \in \mathcal{D}_{\mathbf{x},\theta}} \texttt{label}$, where the set $\mathcal{D}_{\mathbf{x},\theta}$ holds all the rows returned by the query..

Specifically, a number of clusters were generated (49) across the 2D space spanned by the selected columns. Then, 40 queries were generated for each cluster. So in total 1960 synthetic queries were constructed. For the parameters of the queries, we used a Normal distribution with the center of each cluster as the mean value and the variance is set 10% of the mean value at each dimension. We present the various centers (x_1, x_2) for these synthetic queries at Figure 14. It is evident that a number of clusters is present, which represent the existence of various analysts. In addition, we highlight which queries (illustrating the centers only for legibility) were used for training and for testing. Our target was to use a subset of queries from each cluster to examine whether our LPM method could learn that subspace and provide accurate and effective explanations for this particular area. We used uniform distributions bounded in the interval $[0.2, 0.7]$ for the query lengths (θ_1, θ_2).

In this experiment, we compare the accuracy of our own approach LPM with the Linear Regression (LR) and p-order polynomial regression functions indicated by LR(p) in Figure 15 with $p \in \{3, 5, 7, 9\}$. Specifically, we examine the explanation delivery of the AVG operator over the `label` attribute of the data set, which explains the ratio of correctly classified signals over specific data subspaces.

We obtain similar performance results for the `COUNT` and `SUM` operators. We measured the above mentioned goodness of fit R^2 and the NRMSE to examine how well these methods could provide accurate explanations for the specified subspaces (measured by R^2) and whether the same methods can answer future queries with high accuracy (measured by NRMSE). In Figure 15, it is evident that LPM is able to outperform all other methods in providing accurate explanations over specific data regions. LPM achieves the relatively bigger R^2 value compared to the other regression models reflecting its capability for good explanation fitting over specific data regions (the higher the R^2, the better the fitting of the function over the underlying area). Moreover, LPM achieves the lowest NRMSE compared to other methods indicating an accurate predicted explanation over unseen (yet unexplored) data regions, thus, being appropriate for predictive analytics tasks demanding explainability of results. Finally, it is important to note that even though this comparison was made to highlight the fact that LPM can outperform high-order polynomial regression functions in explanation serving, the use of high-order polynomial functions is not appropriate to this explainability problem. This is because, such models are no longer interpretable as the polynomial features are fictitious and do not quantify the statistical importance of the query parameters.

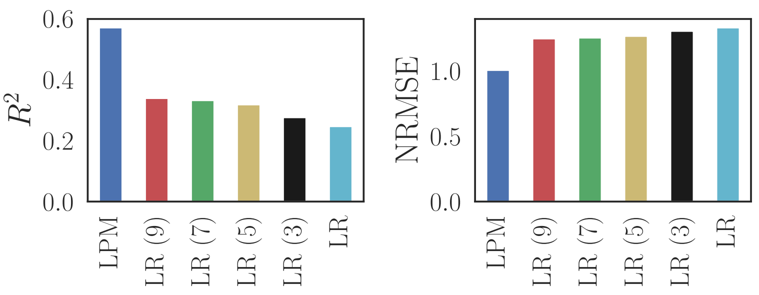


Fig. 15. Predictive accuracy and explanation fit of the methods: LPM, LR and LR(p) over $\mathcal{B}_2$ *Higgs* dataset; (left) R^2-higher is better, (right) NRMSE-lower is better.

8.3 Experimental Results: Efficiency and Scalability

This section presents results on the efficiency and scalability of our proposed approach. The hierarchical structure of our solution means that a number of parameters have to be optimized like the L1/L2 representatives and the parameters of the individual LPMs. Although, the number of parameters is larger than a single linear regression model or a polynomial regression model, we note that the lack of interpretability and low accuracy associated with these methods makes them less favorable candidates for the given problem. In addition, the associated overhead of our approach is relatively small, as it only takes a number of seconds to completely optimize all the associated parameters. This fact will be highlighted in this section. It is also important to mention that this overhead only needs to happen once, as the users/analysts will subsequently use the trained LPMs for data exploration.

8.3.1 Model Training Performance. To effectively measure how long it takes to complete training of LPMs, we restrict the workloads used to "G.G" and set the vigilance thresholds for clustering to the default parameter values of 0.4 for $\boldsymbol{\theta}$ and 0.05 for $\mathbf{x}$. These parameters control the number of clusters created at the L1 and L2 levels and, as it will be shown in our sensitivity analysis section, the number of clusters follows logarithmic scale w.r.t to the vigilance parameters. For this experiment we gradually increase the number of training examples and record the time (in seconds) required for our framework to complete training of the LPMs. We are investigating the potential impact of the size of the training examples fed to our model on the training time required. To account for randomness we repeat the procedure 10 times per number of training examples. Note that this is part of the Pre-processing step which happens offline and helps to initialize the LPMs. Our framework is also able to adjust its parameters online to account for new queries being executed by the back-end analytics engine.

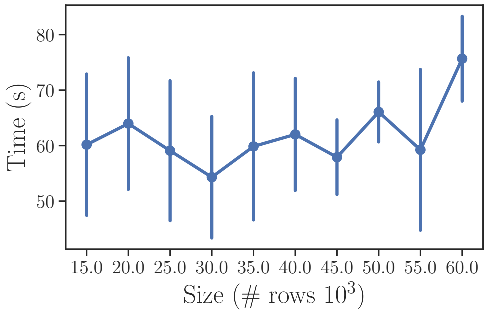


Fig. 16. Training time for an increasing number of model training examples.

The results are shown at Figure 16, as the number of training examples increases we do not observe a significant increase in the time required to train all LPMs. The time ranges from 40(s) to 80(s) with some significant variation. We can then conclude that there is no impact of the size of the training set on the model convergence, which is expected for a given number of clusters. This therefore demonstrates the robustness of our model in its training period. However, for a holistic experimentation on the training time, we further investigated the impact of the number of clusters on the model training time. This investigation reveals that the number of training examples is not highly correlated with the training time. This is clearly exhibited at Table 2, where correlation [2] varies from $[-1, 1]$ with values closer to 1 indicating strong positive correlation and values closer to -1 strong negative correlation. Notwithstanding, the *number* of clusters that formed in L1 and L2 have strong positive correlation with the model training time. And, this is attributed to the fact that an increasing number of clusters yields that the hyper-tuning procedure was running for longer, thus, longer expected training time. The hyper-tuning procedure was outlined at Algorithm 1.

	L1 clusters	L2 clusters	Training Examples	Training Time (s)
L1 clusters	1.000000	0.990559	-0.018008	0.885539
L2 clusters	0.990559	1.000000	-0.004828	0.886221
Training Examples	-0.018008	-0.004828	1.000000	0.218372
Training Time (s)	0.885539	0.886221	0.218372	1.000000

Table 2. Pairwise Pearson's Correlation Coefficient for number of clusters, number of training examples and model training time.

To reinforce this finding we perform the same experiment but this time we vary the number of generated L1/L2 clusters and record the training time. The results are shown at Figure 17. As

[2]Pearson's Correlation Coefficient is used.

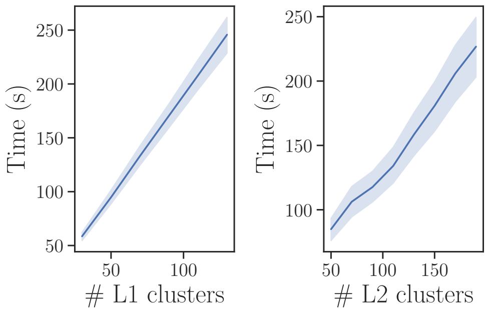


Fig. 17. Training time for an increasing number of training examples with the number of clusters for L1 (left) and L2 (right).

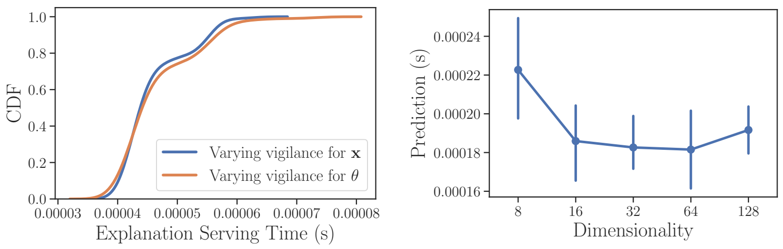


Fig. 18. (Left) A CDF of the explanation serving time by varying hyper-parameters that control number of clusters for L1/L2. (Right) Prediction serving time plotted against an increasing dimensionality of the input vector $\mathbf{q}$

the number of L1/L2 clusters increase, Training Times also increase even with a small number of training examples (we used 15000 training examples in this case). This clearly demonstrates that the main driving factor of the training time is the final number of L1/L2 clusters and not the number of queries. Intuitively, the aforementioned finding is to be expected as more clusters means more models have to be trained.

8.3.2 Explanation & Prediction Serving Performance. We now assess the Explanation and Prediction serving performance of our framework. The explanation functions and any predictions stemming from the explanation functions (LPMs) have to be returned to the user efficiently. Our goal is to do that in less than 500 ms as it has been shown that any answer returned over that limit might hinder productivity [34]. For this experiment we vary the thresholds/vigilance parameters for $\mathbf{x}$ and $\boldsymbol{\theta}$ which control the quantization levels and hence the number of clusters. The threshold values for both parameters $\mathbf{x}$ and $\boldsymbol{\theta}$ were in the interval of $[0.01, 3)$. Figure 18 shows the results of our experiments. Two CDFs are plotted, one where we vary the vigilance parameter for $\mathbf{x}$ and leave the vigilance for $\boldsymbol{\theta}$ constant and vice versa. The creation of more clusters contributes to the total explanation serving time as the closest clusters (according to our formulation) have to be identified for an LPM to be returned to the user. However, the quantization level remains constant after exceeding particular threshold values, as the optimal number of clusters is identified (optimal w.r.t SSQE). Consulting the CDFs at Figure 18 we notice that for both cases the explanation serving time is orders of magnitude less than 500ms. In our experiment with varying the threshold $\mathbf{x}$ we

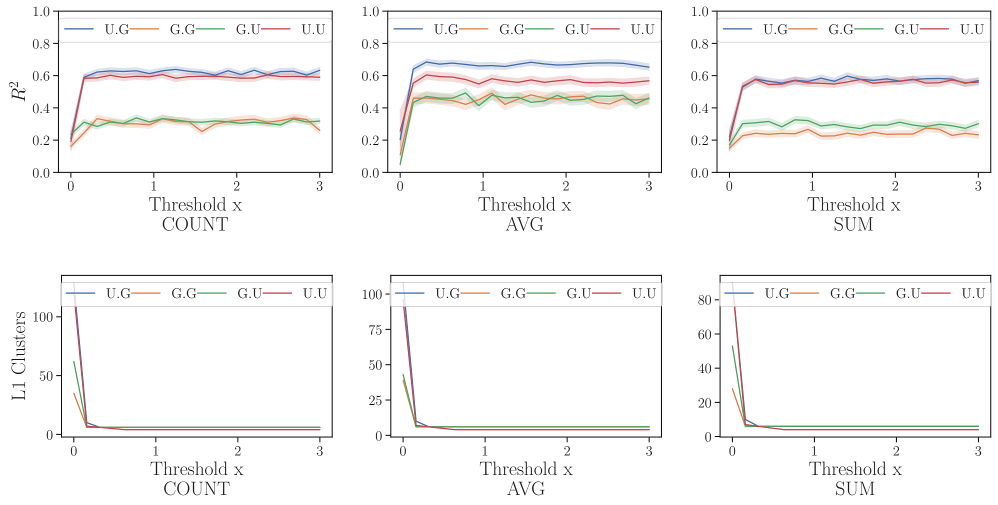


Fig. 19. Measuring the effects of varying threshold/vigilance parameter for **x**. (Top) The effects on accuracy R^2 for different workloads and aggregate functions. (Bottom) The effects on the number of L1 clusters for different workloads and aggregate functions.

saw no significant impact as the quantization level quickly reaches the optimal number of clusters (found to be 6) s before allowing the number of clusters to increase. Once it reaches that optimal value the threshold set for parameter **x** has to be significantly smaller than the SSQE as outlined in Algorithm 1 to allow for the creation of more clusters. Although we would see an impact in performance if the number of clusters at L1 increase, we note that this is no more than linear as a single pass over the LRs of L1 can tell us which one is closer to the query issued by the analyst. Varying the threshold/vigilance parameter for $\boldsymbol{\theta}$ has more effect as we recorded a mean number of total clusters in L2 of 492 with standard deviation of 211. Still, the explanation serving time is no more than 0.08 ms, 6250x faster than the proposed cap for explanation serving time set at 500ms.

To measure the Prediction serving time we vary the dimensionality of the input vector as this would affect the evaluation of the underlying LPM. We used MARS [21] as our ML model and recorded the time required for making a prediction after given an input vector. We varied the dimensionality within the range $[8, 128]$. An input vector with 128 dimensions means that a query is executed with a multi-dimensional center $\mathbf{x} \in \mathbb{R}^{64}$ which translates to filtering based on 64 different attributes. This might be a bit of an overstretch but we wish to stress test our system and see if LPMs can efficiently provide estimations for the answers of unseen queries. The results are shown at Figure 18(right), with the Prediction time fluctuating however never exceeding 1ms. It is important to note that this includes the time of finding the closest LPM.

Overall throughout our performance experiments we conclude that the use of our proposed framework does not hinder productivity. On the contrary, it efficiently informs the user of the unerlying variation in a statistic using our LPMs. The LPMs are located and returned to the user in milliseconds which allows for the analyst to continue ther analysis without bottlenecks. In addition, using LPMs analysts can efficiently get estimates for the answers of any future queries they might have.

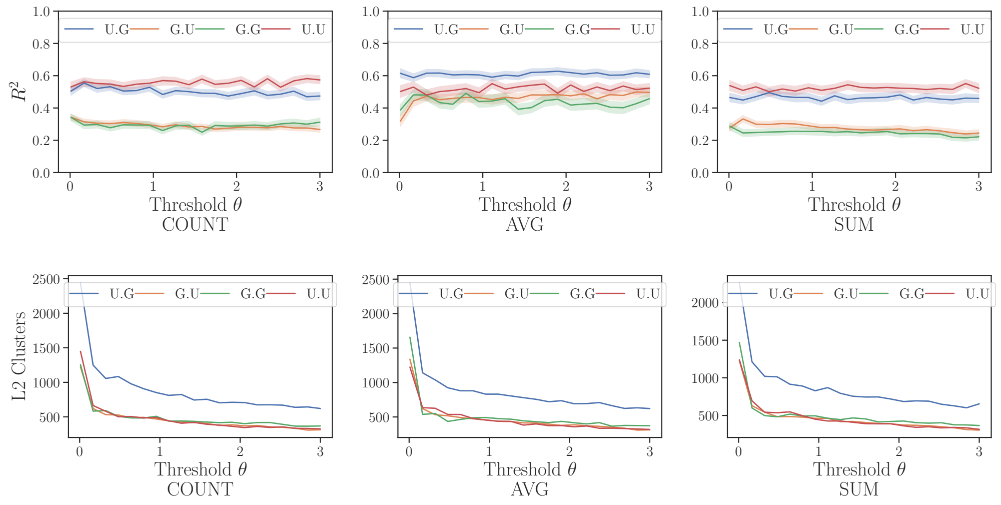


Fig. 20. Measuring the effects of varying threshold/vigilance parameter for $\boldsymbol{\theta}$. (Top) The effects on accuracy R^2 for different workloads and aggregate functions. (Bottom) The effects on the number of L2 clusters for different workloads and aggregate functions.

8.4 Experimental Results: Sensitivity

8.4.1 Threshold/Vigilance Query Center Parameter ($\mathbf{x}$). Several experiments have been conducted to test the sensitivity of our algorithms to the ϵ threshold used in Algorithm 1. As Algorithm 1 is used twice in the Pre-Processing phase we refer to ϵ as the $\mathbf{x}$ Threshold and $\boldsymbol{\theta}$ Threshold to distinguish between the thresholds used in the creation of L1/L2 clustering phases. Effectively these thresholds control the quantization level (the number of clusters). As the thresholds become smaller we continue the clustering procedure with more clusters K for K-Means until the difference of SSQE between a quantization level of K and $K + 1$ is equal to or below the given threshold value.

We varied the threshold for $\mathbf{x}$ in the interval of $[0.0001, 3)$. The results for this experiment are shown at Figure 19. At this figure we see the effects on both the accuracy, Figure 19(top), and the on the number of clusters at L1, Figure 19(bottom). For all workloads and aggregate functions the effects taper off as the number of clusters reaches the optimal number. We can see that accuracy increases and remains constant as the number of clusters converges. So in short, setting the right threshold has an effect on the accuracy of our framework. In our experiments we first normalize the data to be at the same scale which makes setting this threshold parameter easier. In addition, a hyper-tuning procedure could be added for setting this threshold parameter at an optimal value. However this was out of the scope of our work. In general, we can easily determine a near optimal value by randomly testing a number of values along a given range also known as Line-Search.

8.4.2 Threshold/Vigilance Query Parameter ($\boldsymbol{\theta}$). We also vary the threshold for parameter $\boldsymbol{\theta}$ and investigate its effects on the number of clusters for L2 and accuracy measured by R^2. Its effects are recorded for all workloads and datasets at Figure 20. Figure 20(top) shows that the accuracy is constant as the threshold varies meaning that partitioning the region sizes has less effect than optimally partitioning the query locations $\mathbf{x}$. The same effect is observed across aggregates and workloads. On the other hand, the threshold parameter has more effect on the number of L2 clusters and as the threshold increases the quantization level becomes more coarse grained and less clusters

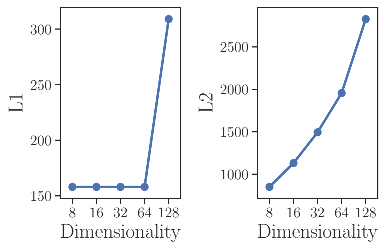


Fig. 21. As the dimensionality of the query vector increases more clusters are needed both in L1 and L2 to sufficiently reduce SSQE.

are created at the L2 level. The number of L2 clusters shown on the y-axis is the total number of clusters associated with all L1 clusters.

8.4.3 Dimensionality of Query Input Vector ($\mathbf{q}$). We experiment with an increasing dimensionality for the query/input vector and investigate its effects on the quantization levels (number of L1/L2 clusters). As evident at Figure 21, the number of L1 and L2 clusters increases w.r.t dimensionality. There is an exponential increase at the total number of L2 clusters as an increase in L1 clusters has a multiplicative effect on the number of L2 clusters. Hence, dimensionality has an effect on the explanation and prediction serving performance as more clusters have to be investigated.

9 CONCLUSIONS & FUTURE WORK

We have defined a novel class of explanations for Aggregate Queries (AQ), which are of particular importance for exploratory analytics. The proposed AQ explanations are succinct, assuming the form of explainable functions. As such, they convey rich information to analysts about the queried data subspaces and as to how the aggregate value depends on key parameters of the queried space. Furthermore, they allow analysts to utilize these explanation functions for data exploration without the need to issue more queries to the DBMS. This is because the proposed explanation functions can be used to estimate the result of an AQ given the parameter values used in an AQ. We have formulated the problem of deriving AQ explanations as a joint optimization problem over a novel hetero-associative statistical learning methodology and have provided novel statistical learning algorithms for its solution. Specifically, the joint optimization problem is deconstructed into three parts which we solve separately. Their solutions serves as an approximation to the solution of the original optimization problem.

Our explainability scheme for computing explanations does not require DBMS data accesses ensuring efficiency and scalability as we use previously executed queries for all phases of our methodology. In addition, we have shown examples of how the explanation functions can be leveraged by analysts when performing data exploration. Data analysts can consult the visualisations constructed using the explanation functions to identify subspaces of interest a-priori without executing any query against the underlying back-end system. We believe this is a significant step forward in expediting the data analytics process using frameworks that leverage past knowledge extracted from previously executed queries.

Our future research agenda includes explainable machine learning models that can learn to interpret the predicted outcomes of incoming analytics queries, while being able to autonomously *explore* and *recommend* to analysts potentially interested (unknown) data sub-spaces in advance. Hence, the system can pre-analyze and provide directives for *deep exploration* to data regions before being visited/targeted by analysts for the analysts.

ACKNOWLEDGEMENT

This research is partially funded by the UK EPSRC Grant EP/R018634/1 (Closed–Loop Data Science for Complex, Computationally- and Data–Intensive Analytics) and the European Union EU/H2020 Marie Sklodowska-Curie Action Individual Fellowship (MSCA-IF) under the project INNOVATE (Grant No.745829).

REFERENCES

[1] 2016. Crimes - 2001 to present. URL: https://data.cityofchicago.org/Public-Safety/Crimes-2001-to-present/ijzp-q8t2. Accessed: 2016-12-01.

[2] 2019. Query Analytics Workloads Dataset Data Set. URL: https://archive.ics.uci.edu/ml/datasets/Query+Analytics+Workloads+Dataset. Accessed: 2019-07-29.

[3] 2020. HIGGS Data Set. URL: http://archive.ics.uci.edu/ml/datasets/HIGGS. Accessed: 2020-02-19.

[4] Sameer Agarwal, Barzan Mozafari, Aurojit Panda, Henry Milner, Samuel Madden, and Ion Stoica. 2013. BlinkDB: queries with bounded errors and bounded response times on very large data. In *Proceedings of the 8th ACM European Conference on Computer Systems*. ACM, 29–42.

[5] Yael Amsterdamer, Daniel Deutch, and Val Tannen. 2011. Provenance for aggregate queries. In *Proceedings of the thirtieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. ACM, 153–164.

[6] Christos Anagnostopoulos, Fotis Savva, and Peter Triantafillou. 2018. Scalable aggregation predictive analytics. *Applied Intelligence* 48, 9 (2018), 2546–2567.

[7] Christos Anagnostopoulos and Peter Triantafillou. 2015. Learning set cardinality in distance nearest neighbours. In *Data Mining (ICDM), 2015 IEEE International Conference on*. IEEE, 691–696.

[8] Christos Anagnostopoulos and Peter Triantafillou. 2017. Efficient Scalable Accurate Regression Queries in In-DBMS Analytics. In *Data Engineering (ICDE), 2017 IEEE 33rd International Conference on*. IEEE, 559–570.

[9] Peter Bailis, Edward Gan, Samuel Madden, Deepak Narayanan, Kexin Rong, and Sahaana Suri. 2016. MacroBase: Analytic Monitoring for the Internet of Things. *arXiv preprint arXiv:1603.00567* (2016).

[10] Pierre Baldi, Peter Sadowski, and Daniel Whiteson. 2014. Searching for exotic particles in high-energy physics with deep learning. *Nature communications* 5 (2014), 4308.

[11] Bokeh Development Team. 2018. *Bokeh: Python library for interactive visualization.* https://bokeh.pydata.org/en/latest/

[12] Léon Bottou. 2012. Stochastic gradient descent tricks. In *Neural networks: Tricks of the trade*. Springer, 421–436.

[13] Anup Chalamalla, Ihab F Ilyas, Mourad Ouzzani, and Paolo Papotti. 2014. Descriptive and prescriptive data cleaning. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. ACM, 445–456.

[14] Surajit Chaudhuri, Gautam Das, and Vivek Narasayya. 2007. Optimized stratified sampling for approximate query processing. *ACM Transactions on Database Systems (TODS)* 32, 2 (2007), 9.

[15] Surajit Chaudhuri, Gautam Das, and Utkarsh Srivastava. 2004. Effective use of block-level sampling in statistics estimation. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*. ACM, 287–298.

[16] James Cheney, Laura Chiticariu, Wang-Chiew Tan, et al. 2009. Provenance in databases: Why, how, and where. *Foundations and Trends® in Databases* 1, 4 (2009), 379–474.

[17] Graham Cormode and Shan Muthukrishnan. 2005. An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms* 55, 1 (2005), 58–75.

[18] Çağatay Demiralp, Peter J Haas, Srinivasan Parthasarathy, and Tejaswini Pedapati. 2017. Foresight: Recommending visual insights. *Proceedings of the VLDB Endowment* 10, 12 (2017), 1937–1940.

[19] Kareem El Gebaly, Parag Agrawal, Lukasz Golab, Flip Korn, and Divesh Srivastava. 2014. Interpretable and informative explanations of outcomes. *Proceedings of the VLDB Endowment* 8, 1 (2014), 61–72.

[20] Jerome Friedman, Trevor Hastie, and Rob Tibshirani. 2010. Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software* 33, 1 (2010), 1.

[21] Jerome H Friedman. 1991. Multivariate adaptive regression splines. *The annals of statistics* (1991), 1–67.

[22] Greg Hamerly and Charles Elkan. 2004. Learning the k in k-means. In *Advances in neural information processing systems*. 281–288.

[23] John A Hartigan and Manchek A Wong. 1979. Algorithm AS 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 28, 1 (1979), 100–108.

[24] Trevor Hastie, Robert Tibshirani, Jerome Friedman, and James Franklin. 2005. The elements of statistical learning: data mining, inference and prediction. *The Mathematical Intelligencer* 27, 2 (2005), 83–85.

[25] Joseph M Hellerstein, Peter J Haas, and Helen J Wang. 1997. Online aggregation. In *ACM SIGMOD Record*, Vol. 26. ACM, 171–182.

[26] Botong Huang, Shivnath Babu, and Jun Yang. 2013. Cumulon: Optimizing statistical data analysis in the cloud. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*. ACM, 1–12.

[27] Stratos Idreos, Olga Papaemmanouil, and Surajit Chaudhuri. 2015. Overview of data exploration techniques. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. ACM, 277–281.

[28] Shrainik Jain, Dominik Moritz, Daniel Halperin, Bill Howe, and Ed Lazowska. 2016. Sqlshare: Results from a multi-year sql-as-a-service experiment. In *Proceedings of the 2016 International Conference on Management of Data*. ACM, 281–293.

[29] Bhargav Kanagal, Jian Li, and Amol Deshpande. 2011. Sensitivity analysis and explanations for robust query evaluation in probabilistic databases. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*. ACM, 841–852.

[30] Nodira Khoussainova, Magdalena Balazinska, and Dan Suciu. 2012. Perfxplain: debugging mapreduce job performance. *Proceedings of the VLDB Endowment* 5, 7 (2012), 598–609.

[31] Sanjay Krishnan and Eugene Wu. 2017. PALM: Machine Learning Explanations For Iterative Debugging. In *Proceedings of the 2nd Workshop on Human-In-the-Loop Data Analytics*. ACM, 4.

[32] M. L. Kersten L. Sidirourgos and P. A. Boncz. 2011. SciBORQ: Scientific data management with Bounds On Runtime and Quality. In *Proceedings of Conference on Innovative Data Systems, (CIDR)*.

[33] Roger J Lewis. 2000. An introduction to classification and regression tree (CART) analysis. In *Annual meeting of the society for academic emergency medicine in San Francisco, California*, Vol. 14.

[34] Zhicheng Liu and Jeffrey Heer. [n.d.]. The effects of interactive latency on exploratory visual analysis. *IEEE Transactions on Visualization & Computer Graphics* 1 ([n. d.]), 1–1.

[35] Alexandra Meliou, Sudeepa Roy, and Dan Suciu. 2014. Causality and explanations in databases. *Proceedings of the VLDB Endowment* 7, 13 (2014), 1715–1716.

[36] Zhengjie Miao, Qitian Zeng, Boris Glavic, and Sudeepa Roy. 2019. Going Beyond Provenance: Explaining Query Answers with Pattern-based Counterbalances. In *Proceedings of the 2019 International Conference on Management of Data*. ACM, 485–502.

[37] John Moody. 1988. âĂIJFast Learning in Multi-Resolution HierarchiesâĂİ. In *Proceedings of the 1st International Conference on Neural Information Processing Systems (NIPSâĂŹ88)*. MIT Press, Cambridge, MA, USA, 29âĂŞ39.

[38] Vinod Nair, Ameya Raul, Shwetabh Khanduja, Vikas Bahirwani, Qihong Shao, Sundararajan Sellamanickam, Sathiya Keerthi, Steve Herbert, and Sudheer Dhulipalla. 2015. Learning a hierarchical monitoring system for detecting and diagnosing service issues. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2029–2038.

[39] Yongjoo Park, Ahmad Shahab Tajik, Michael Cafarella, and Barzan Mozafari. 2017. Database learning: Toward a database that becomes smarter every time. In *Proceedings of the 2017 ACM International Conference on Management of Data*. ACM, 587–602.

[40] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *Journal of machine learning research* 12, Oct (2011), 2825–2830.

[41] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. Why Should I Trust You?: Explaining the Predictions of Any Classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1135–1144.

[42] Sudeepa Roy, Laurel Orr, and Dan Suciu. 2015. Explaining query answers with explanation-ready databases. *Proceedings of the VLDB Endowment* 9, 4 (2015), 348–359.

[43] Fotis Savva, Christos Anagnostopoulos, and Peter Triantafillou. 2018. Explaining Aggregates for Exploratory Analytics. In *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 478–487.

[44] Fotis Savva, Christos Anagnostopoulos, and Peter Triantafillou. 2019. Aggregate Query Prediction under Dynamic Workloads. In *2019 IEEE International Conference on Big Data (Big Data)*. IEEE, 671–676.

[45] Fotis Savva, Christos Anagnostopoulos, and Peter Triantafillou. 2020. Adaptive learning of aggregate analytics under dynamic workloads. *Future Generation Computer Systems* (2020).

[46] Fotis Savva, Christos Anagnostopoulos, and Peter Triantafillou. 2020. ML-AQP: Query-Driven Approximate Query Processing based on Machine Learning. *arXiv preprint arXiv:2003.06613* (2020).

[47] Alexander S Szalay, Jim Gray, Ani R Thakar, Peter Z Kunszt, Tanu Malik, Jordan Raddick, Christopher Stoughton, and Jan vandenBerg. 2002. The SDSS skyserver: public access to the sloan digital sky server data. In *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*. ACM, 570–581.

[48] Jean Claude Utazirubanda, TomÃąs M. LeÃşn, and Papa Ngom. 2019. Variable selection with group LASSO approach: Application to Cox regression with frailty model. *Communications in Statistics - Simulation and Computation* 0, 0 (2019), 1–21.

[49] Manasi Vartak, Sajjadur Rahman, Samuel Madden, Aditya Parameswaran, and Neoklis Polyzotis. 2015. S ee DB: efficient data-driven visualization recommendations to support visual analytics. *Proceedings of the VLDB Endowment* 8,

13 (2015), 2182–2193.

[50] Xiaolan Wang, Xin Luna Dong, and Alexandra Meliou. 2015. Data x-ray: A diagnostic tool for data errors. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. ACM, 1231–1245.

[51] Xiaolan Wang, Alexandra Meliou, and Eugene Wu. 2017. Qfix: Diagnosing errors through query histories. In *Proceedings of the 2017 ACM International Conference on Management of Data*. ACM, 1369–1384.

[52] Abdul Wasay, Xinding Wei, Niv Dayan, and Stratos Idreos. 2017. Data Canopy: Accelerating Exploratory Statistical Analysis. In *Proceedings of the 2017 ACM International Conference on Management of Data*. ACM, 557–572.

[53] Kilian Weinberger, Anirban Dasgupta, John Langford, Alex Smola, and Josh Attenberg. 2009. Feature Hashing for Large Scale Multitask Learning. In *Proceedings of the 26th Annual International Conference on Machine Learning (ICML âĂŹ09)*. Association for Computing Machinery, New York, NY, USA, 1113âĂŞ1120. https://doi.org/10.1145/1553374.1553516

[54] Eugene Wu and Samuel Madden. 2013. Scorpion: Explaining away outliers in aggregate queries. *Proceedings of the VLDB Endowment* 6, 8 (2013), 553–564.

[55] Eugene Wu, Samuel Madden, and Michael Stonebraker. 2013. SubZero: A fine-grained lineage system for scientific databases. In *Data Engineering (ICDE), 2013 IEEE 29th International Conference on*. IEEE, 865–876.

[56] Sai Wu, Beng Chin Ooi, and Kian-Lee Tan. 2010. Continuous sampling for online aggregation over multiple queries. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*. ACM, 651–662.

APPENDIX

Table 3. Table of Notation

Parameter	Description
$\mathbf{a} = [a_1, \dots, a_d]$	d-dimensional data vector
$\mathbf{q} = (\mathbf{x}, \boldsymbol{\theta})$	$2d$-dimensional range query
$\mathbb{D}(\mathbf{x}, \boldsymbol{\theta})$	d-dimensional convex data subspace
$\mathbf{x} \in \mathbb{R}^d$	parameter query center
$\boldsymbol{\theta} \in \mathbb{R}^d$	parameter query length
f	explanation/regression function
$\mathbb{D}, \mathbb{Q}$	data and query vectorial spaces
y	scalar aggregate query output
$\mathcal{L}(f, \hat{f})$	explanation loss metric
$\mathcal{T}, \mathcal{V}$	query (workload) training and testing sets, respectively
$\mathcal{S}$	sub-queries set
$\mathcal{B}$	Data set
$\mathbb{Q}_k$	Level 1 (L1) query cluster of Location Representative (LR)
$\mathbf{w}_k \in \mathbb{R}^d$	location representative
$\mathbb{U}_{kl}$	Level 2 (L2) sub-cluster of Region Representative (RR)
$\mathbf{u}_{kl} \in \mathbb{R}^d$	region representative
(β, λ)	parameters of PLR explanation function
$h(\mathbf{q})$	MARS basis hinge function
$\mathcal{W}, \mathcal{U}$	location and region representative sets, respectively