XAQ: Explaining Aggregate Queries

Fotis Savva
School Of Computing Science
University Of Glasgow, UK
f.savva.1
@research.gla.ac.uk

Christos Anagnostopoulos
School Of Computing Science
University Of Glasgow, UK
christos.anagnostopoulos
@glasgow.ac.uk

Peter Triantafillou
Department of Computer Science
University Of Warwick, UK
p.triantafillou@warwick.ac.uk

ABSTRACT

Aggregation Queries (AQs) have long been recognized as central in data management. Nowadays, in the realm of big data, aggregation queries play a central role in exploratory analytics. Analysts wishing to explore the (multivariate) data space, typically pose selection queries (i.e., range queries) which in essence define data subspaces and then use aggregation functions, the results of which help them decide how interesting such subspaces are. However, these results are single numerical values. As such, they convey limited information. We aim to address this shortcoming of current DBMSs, aiding analysts to explore and better understand data spaces by contributing explanations of AQ results. First, we define the notion of AQ explanations. Second, we present a data-gnostic computation of explanations, which requires accessing the DBMS base data. Third, we contribute a data-agnostic computation of explanations, which does not require base data access and hence goes a long way towards improved efficiency and scalability. Data-agnostic explanations utilize novel Machine Learning models and algorithms which compute explanations with high accuracy by simply monitoring and learning previously executed AQs. The novel AQ explanations are represented as functions that can be used to both generate visualizations and assist analysts in exploring unknown data subspaces. We propose multiple algorithms that produce such explanations with minimal cost, and compare the performance and accuracy of each of the approaches using a set of novel statistical and information-theoretic metrics.

PVLDB Reference Format:
. A Sample Proceedings of the VLDB Endowment Paper in LaTeX Format. *PVLDB*, 11 (5): xxxx-yyyy, 2018.
DOI: https://doi.org/TBD

1. INTRODUCTION

The era of big data has been facilitated by a capacity explosion (in store-compute-communicate resources) and has given birth to the needs of being able to explore data and understand it and derive new knowledge. The procedure followed by analysts tasked with the above is rather ad hoc and domain specific, but invariantly includes the fundamental step of *exploratory analysis* [25]. 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 or give recommendations to models that decide whether a tumor is benign or malignant.

Aggregate Queries (AQs), e.g., `COUNT`, `SUM`, `AVG` (`AVERAGE`), `MAX`, play a key role in exploratory analysis as they summarize regions of data. Using these aggregates, analysts decide whether a region is of importance or not, depending on the task at hand. For example, imagine checking whether a particular range of zip codes is of interest, where the latter depends on the count of persons enjoying a 'high' income. However, such queries return single values: For example, if the count of a particular subspace is 273, what does that mean? If the selection predicate was less or more selective, how would this count change? Or, which of the involved data subspaces contribute more to the total count value? To answer such questions and enhance the analysts understanding of the queried subspace, they need to issue more queries. The analyst has no understanding of the space that would steer them in the right direction with respect to which queries to pose next; as such, further data exploration becomes ad hoc, unsystematic, and uninformed. To this end, the main objective of this paper is to find ways to assist the analysts understanding such subspaces. In turn, this would lead to fewer queries issued against the system saving valuable system resources and drastically reducing the time needed for data exploration.

Our goal is to explain *how* an AQ result is derived. This is required so one can infer the potential impact of a change in a parameter. A convenient way to represent how something is derived (in terms of compactly and succinctly conveying rich information) is by using a function: a function can describe how an output is dependent upon the independent variables and show the contribution of each one of those parameters. For example, we can derive a function that explains how the mean household income for each region is generated, based on the size of the region. Thus, the resulting function can inform the data scientist as to how influential the size of a region is and how the different results are generated across its different subregions. Alas, state of the art 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

Proceedings of the VLDB Endowment, Vol. 11, No. 5

DOI: https://doi.org/TBD

as one has to find efficient, scalable and accurate methods of providing such functionality. Even though, explanation methods have been proposed, the need of accessing and utilizing potentially huge datasets, serves as a drawback which such methods cannot address.

In this paper, we present XAQ (from eXplaining AQ), a framework which tackles this problem by providing explanations as to *how* results obtained from AQs are derived. XAQ makes use of Machine Learning (ML) models which *estimate* functions that can explain AQs. It also provides both data-agnostic and data-gnostic schemes for computing explanations, offering different levels of accuracy, efficiency, and scalability. A *data-gnostic* approach utilizes the underlying data to generate explanations, thus, being inefficient for large-scale datasets. A *data-agnostic* approach does not require knowledge of the underlying data, which improves efficiency and scalability, but at the expense of accuracy, as it *approximates* an explanation.[1]

1.1 Motivation

We provide comprehensive use cases of explanations in AQ results, which will serve as running examples throughout the paper.

Crimes Data: Imagine Police departments employing data scientists to analyze collected data. The law-enforcement datasets contain crimes, recorded in the past, along with their location, the type of crime (homicide, burglary, bank robbery etc.) and other information. One such dataset is the Chicago Crimes Dataset [1]. A common exploration technique is to issue the AQ:

```
SELECT COUNT(*) AS y
FROM Crimes AS C
WHERE $theta>sqrt(power($X-C.X,2)+power($Y-C.Y,2));
```

Such AQ uncovers the number of crimes in a specific area of interest centered around the location coordinates ($X, $Y). An area of interest can be a given neighborhood or a specific district. In this case, an explanation could aid the analyst in identifying whether the crime rate is increasing or decreasing for a changing radius. Also, it can assist in the comparison of crime rates between different regions.

Telecommunication Calls Data: Data scientists working for a telecommunications provider issue AQs to identify areas in need of infrastructure maintenance based on the average call time within an area:

```
SELECT AVG(call_time) AS z
FROM Calls AS C
WHERE $theta>sqrt(power($X-C.X,2)+power($Y-C.Y,2));
```

Explaining the results given from such an AQ helps the telecommunications provider to pin-point the exact locations where better infrastructure is needed.

The aforementioned AQs return scalars/numbers as results conveying no information as to what the influence of the given parameters (in this case location and radius) is w.r.t the final result. Having this information at hand can ease the burden of manually performing Root Cause Analysis. Knowing the influence of each of these parameters can help us pinpoint the exact parameters to which the result of the AQs is at its maximum, minimum or within a given range. Hence, in our first example, identifying areas being in need of more policing because the number of crimes exceeds a threshold of incidents is of high importance.

[1]Gnostic and agnostic terms come from the Greek word gnosis ($\gamma\nu\omega\sigma\iota\varsigma$) which means to know something and agnosis α- (a, without, lacking) which is to not know.

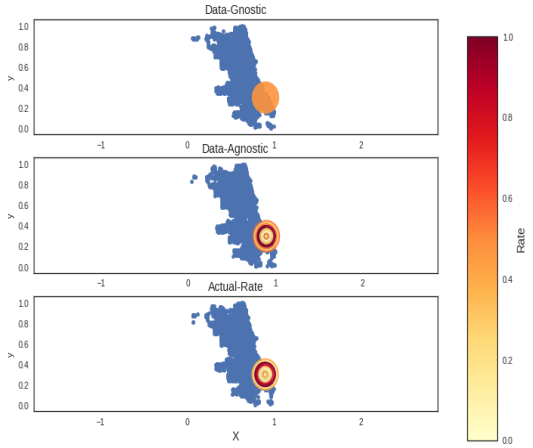


Figure 1: Heatmap Visualisation of an Explanation.

A typical use case for our explanation function would be to visualize the rate in which COUNT y (and AVERAGE z) increases or decreases with respect to *$theta*, as shown in Figure 1. Returning to our example, once the data scientist has received an explanation for their issued query, they can plot the different rates to which the average call time, or the number of crimes increase in a given area. We can easily obtain the rates with the first derivative of our given function. This information is illustrated in Figure 1 in which both a Data-Agnostic and Data-Gnostic explanations are shown. Moreover, we plot the actual rate obtained by issuing multiple AQs at the same location with varying *$theta* noting that such operation affects the performance of a system contrary to our given explanation which constructs this instantaneously.

Note that, in general, the above *radius-selection operators*, in 1-d space are equivalent to range queries and that the input parameters can be any kind of attribute (not limited to location and radius). They are also of high importance in many applications including: location-based search, e.g., searching for spatially-close (within a radius) objects; spatial analytics, e.g., monitoring regions of interest from sensors' acoustic signals, etc., and they are natural to formulate using a user interface, which allows analysts to draw circles or (hyper-)spheres of interest within the visualized data space. Hence, we will be focusing on such radius selection AQs.

2. RELATED WORK & CONTRIBUTION

Our work is inspired by works showing the fundamental role that AQs play for data management, works that provide fast AQ evaluation, by the recent trends in deriving query explanations, and by recent trends in ML for efficiency and scalability. AQ results present key statistics about data (sub)spaces. They are vital for internal DB functioning, e.g., for estimating cardinalities and index utilities [13]), and have been utilized for various more sophisticated purposes, such as data integration [20, 17], time series data understanding [40, 50], and more recently index learning [41]. Recently,

with the added value placed on predictive analytics, in the realm of increasing DBMS functionality with providing elaborate statistics, new specialized DB connectors (e.g., Psycopg [43], SciDB-Py [9] and [34] facilitate the computation of various descriptive statistics about datasets without the expensive transfer of data outside the DB.

The efficient computation of AQs has long been a major research subject [23]. A large body of research has since focused on fast computations of aggregations/statistics, primarily employing sampling techniques (such as [12, 16, 49]), or synopses [15, 16]. Recently, efficient methods for aggregations/statistics computation have been proposed for distributed settings, (e.g., [24]), using ML, e.g., for computing COUNT [4] and AVG [5], and developing semantic caches holding AQ results [46].

XAQ aims to provide explanations for this fundamental set of queries. Compared to XAQ, the above works are largely complementary. For example, the data-gnostic method of XAQ can be used in tandem with any of the above approaches to replace expensive base data accesses and the data-agnostic method of XAQ can utilize the above in order to expedite the training of its ML models. However, the key salient distinguishing feature of XAQ is that the primary task of XAQ is to first *explain the AQ results* and do so efficiently and scalably w.r.t. increasing data sizes.

Explanation techniques have emerged in multiple contexts within the data management and ML communities. These are all linked to Data Provenance [14], which tracks tuples to answer questions such as 'Why? Where? and How?' the user was given a certain query result. But since then, all of the focused work has largely departed from this, resulting in work with varying contexts and different objectives. One of such contexts, is to find explanations for simple query answers as in [18], [39], [38][33]. By utilizing data in a relational setting, the authors represent their explanations as predicates, which give a meaningful interpretation as to what the main characteristics of the resulting set are. Similarly, other authors have extended this in probabilistic and scientific databases [28], [48]. Such approaches, aim to answer 'Why?' a particular answer is given.

Explanations have also been used in a variety of other domains such as interpreting outliers in both *in-situ* data [47] and in streaming data [6]. The authors first detect outliers, either manually or automatically, and then generate predicates or attribute-value combinations that explain the outliers set. In addition, explanations in the context of debugging purposes has been used extensively. For instance, explaining errors in service systems [35] is another domain that people have tried to apply explanations in. In [35], the authors build a system to provide interpretations for service errors and present the explanations visually. In addition, the authors of PerfXPlain [29] created a tool to assist users while debugging performance issues in Map-Reduce jobs. Moreover, multiple frameworks deal with potential errors in a query answer. Such frameworks provide explanations that in turn can be utilized by the users to locate any discrepancies found in their data [44], [10], [45]. A recent trend is in explaining ML models [30], [42], [37] for debugging purposes [30] or to understand how a model makes predictions [42] and conveys trust to users using these predictions [37].

In this work, we particularly focus on AQs because of their wide use. Both [47] and [3] 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 semiring formalisms. Our major difference is that we wish to explain these AQs solely based on the input parameters and results of previous queries, thus not having to rely on the underlying data, which makes generating explanations slow and inefficient for large-scale datasets. That being said are objectives and criteria are vastly different than the aforementioned related work. We wish to find ways where AQs can be explained and provide further insights to data analysts to establish informed exploratory and descriptive statistics tasks. Hence, using the formalisms from [3] is not possible as they would be incomprehensible and in need of provenance query language, as explicitely stated in [3].

Our work is unique in developing and adapting ML models in order to increase efficiency/scalability when generating explanations. Providing accuracy while being efficient and scalable is being increasingly recognized as central within the community [27, 8]. A popular approach rests on algorithmic weakening i.e., given higher volumes of data, faster (but less robust) algorithms are used where the increased data sizes can compensate for any loss in accuracy [27]. We propose a novel approach whereby our models learn from previous query executions and can, thus, ensure high efficiency and scalability as future AQ explanations do not need to access the data. This is in line with recent developments in the DB community as well, such as [36], where answers to past queries are employed to improve future query performance, although our models, like [4, 5], focuses on AQs and require no DB access at all. Further, this work centers on defining AQ explanations and deriving them accurately, scalably, and efficiently.

Scalability/efficiency is particularly important as generating explanations is proved to be an NP-Hard problem [44] and generating explanations can take a long time [47, 38, 18] even when using small-to-medium-sized datasets. An exponential increase in data size will imply a dramatic increase in the time required to generate explanations. XAQ does not suffer from such limitations and is able to construct explanations in a matter of milliseconds even with massive data volumes. To do so, it relies on two pillars: First, on workload characteristics: workloads contain a large number of overlapping queried data subspaces. This characteristic has been acknowledged and exploited also by recent research e.g., [46],[36], STRAT[11], and SciBORQ [31] 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 [26]. Second, XAQ relies on novel, on-line ML models that exploit the above workload characteristics to perform efficient and accurate (approximate) AQ explanations. Our **contributions** are:

1. Define the notion of explanations for AQs. The contributed explanations are succinct and guide the analyst engage in further informed exploratory and descriptive analytics.

2. Generate explanations that are expressed through simple functions and can be used for data visualizations.

3. Contribute the notions and related algorithms for data-agnostic and data-gnostic computations of explanations.

4. Contribute novel statistical and information theoretic metrics for evaluating the accuracy of AQ explanations.
5. Construct novel approaches introducing hierarchical quantized multivariate piece-wise linear regression approximations of explanation functions.
6. Conduct extensive evaluation of the proposed explanations.

3. EXPLANATIONS REPRESENTATION

We first formalize the notion of a data space and AQ. We model both our data and AQs as vectors in a vectorial data space, $\mathbb{D} \subset \mathbb{R}^d$, and vectorial query space, $\mathbb{Q}$, respectively. We formalize the two specified queries in our previous examples, namely `COUNT` and `AVG` noting that this can easily be expanded to any kind of AQ. Then we formally define AQ explanations and formalize the problems and solutions for computing AQ explanations.

3.1 Vectorial Representation of Queries

Let $\mathbf{x} = [x_1, \ldots, x_d] \in \mathbb{R}^d$ denote a multivariate random row vector.

Definition 1. (Vector Norm) The p-norm (L_p) distance between two vectors $\mathbf{x}$ and $\mathbf{x}'$ from $\mathbb{R}^d$ for $1 \leq p < \infty$, is $\|\mathbf{x}-\mathbf{x}'\|_p = (\sum_{i=1}^{d} |x_i - x_i'|^p)^{\frac{1}{p}}$ and for $p = \infty$, is $\|\mathbf{x}-\mathbf{x}'\|_\infty = \max_{i=1,\ldots,d}\{|x_i - x_i'|\}$.

Consider now a scalar $\theta > 0$, hereinafter referred to as *radius*, and a dataset $\mathcal{B}$ consisting of N vectors $\{\mathbf{x}_i\}_{i=1}^{N}$.

Definition 2. (Data Subspace) Given $\mathbf{x} \in \mathbb{R}^d$ and scalar θ, a data subspace $\mathbb{D}(\mathbf{x}, \theta)$ is the convex subspace of $\mathbb{R}^d$, which includes vectors $\mathbf{x}_i : \|\mathbf{x}_i - \mathbf{x}\|_p \leq \theta$ with $\mathbf{x}_i \in \mathcal{B}$.

Definition 3. (Count AQ) Given a vector $\mathbf{x} \in \mathbb{R}^d$ and $\theta > 0$, a count AQ over a dataset $\mathcal{B}$ returns the number of vectors $\|\mathbf{x}_i - \mathbf{x}\|_p \leq \theta$ where $y = n_\theta(\mathbf{x}) \in \{0, 1, \ldots, N\}$ is the cardinality of the set $|\{\mathbf{x}_i : \|\mathbf{x}_i - \mathbf{x}\|_p \leq \theta\}|$ and $\mathbf{x}_i \in \mathcal{B}$. We represent a count AQ as the $(d+1)$-dim. row vector $\mathbf{q} = [\mathbf{x}, \theta] \in \mathbb{Q} \subset \mathbb{R}^{d+1}$. The $(d+1)$-dim. space $\mathbb{Q}$ is referred to as the query vectorial space.

Definition 4. (Average AQ) Given a vector $\mathbf{x} \in \mathbb{R}^d$ and θ, the average AQ over a dataset $\mathcal{B}$ with elements/pairs $(\mathbf{x}_i, z_i) \in \mathcal{B}$ returns the average value: $z = \frac{1}{n_\theta(\mathbf{x})} \sum_{i \in [n_\theta(\mathbf{x})]} z_i$: $\|\mathbf{x}_i - \mathbf{x}\|_p \leq \theta$, where $n_\theta(\mathbf{x})$ is the cardinality of the set $|\{\mathbf{x}_i : \|\mathbf{x}_i - \mathbf{x}\|_p \leq \theta\}|$ and $(\mathbf{x}_i, z_i) \in \mathcal{B}$. We represent an average AQ as the $(d+1)$-dim. row vector $\mathbf{q} = [\mathbf{x}, \theta] \in \mathbb{Q} \subset \mathbb{R}^{d+1}$ and its answer as $z \in \mathbb{R}$. We adopt the compact notation $i \in [n]$ as for $i = 1, \ldots, n$.

Definition 5. (Query Similarity) The L_2^2 distance or similarity measure between AQs $\mathbf{q}, \mathbf{q}' \in \mathbb{Q}$ is $\|\mathbf{q} - \mathbf{q}'\|_2^2 = \|\mathbf{x} - \mathbf{x}'\|_2^2 + (\theta - \theta')^2$.

Figure 2 illustrates count and average queries issued over multiple d-dimensional data-spaces over a data set $\mathcal{B}$; red circles/spheres denote the count/average queries and blue dots are the points/vectors in the dataset. For the 2-dimensional example in Figure 2 (right), we go back to our running example with Data Scientists working for the Police Department. Imagine that circles represent count queries $\mathbf{q} = [x_1, x_2, \theta]$

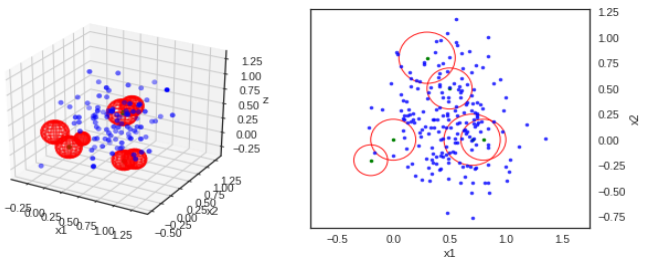


Figure 2: (Left) Average queries visualized in 3D; x-/y-axis denote the center coordinates x of a query and z-axis is the average value of the turnaround time to update a case; (right) Count queries visualized in 2D; x-/y-axis denote the center coordinates x of a query and the circles are defined by the queries radii.

issued by those scientists, exploring a dataset comprised of a number of recorded offenses/crimes (i.e., subset cardinality y) in a particular state/area and where each blue point is the location of one such offense. Concretely, the 3-dimensional example in Figure 2 (left) extends the previous example involving average queries $\mathbf{q} = [x_1, x_2, \theta]$ which return the average call time within an area i.e., $z = \textit{(call\_time)}$. Given a query $\mathbf{q} = [\mathbf{x}, \theta]$, we refer to $\mathbf{x} \in \mathbb{R}^d$ as the *center* of query $\mathbf{q}$, e.g., the location of interest in our examples. To capture a region at the specific location, the query radius θ is provided to define the queried data-subspace $\mathbb{D}(\mathbf{x}, \theta)$.

3.2 Functional Representation of Explanations

The count and average AQs defined in Section 3.1 return a single scalar value y and z, respectively, to the analysts, over the regions/data subspaces of interest. We seek to explain how such values are generated by finding a *function* $f : \mathbb{R} \times \mathbb{R}^d \to \{0, \ldots, N\}$ and $f : \mathbb{R} \times \mathbb{R}^d \to \mathbb{R}$ that can describe how y and z, respectively, are produced given an ad-hoc query $\mathbf{q} = [\mathbf{x}, \theta]$ with $\mathbf{x} \in \mathbb{R}^d$. A thing to note here is that the center of each query we are trying to explain is conditionally constant. This means that, given our interest in this specific parameter $\mathbf{x}$, we desire explaining the evolution/variability of the outputs y and z of the query $\mathbf{q}$ by varying the radius θ. Therefore, the input variable to our parametric function $f(\theta; \mathbf{x})$ is the radius θ given a parameter of interest $\mathbf{x}$. Intuitively, this makes sense as allowing the location $\mathbf{x}$ to change would mean that we are explaining a *different* query w.r.t. location of interest. Such a function can be linear or polynomial w.r.t θ given a fixed center $\mathbf{x}$. However, when using high-order polynomial functions as our explanations, we assume that none of the resulting aggregates will be monotonically increasing. A monotonically increasing function is a function in which the output will not decrease for an increasing input variable like θ. An aggregate belonging to the family of such functions is `COUNT` targeting to count queries. Its output only increases or remains constant for an enlarging radius θ given a fixed center $\mathbf{x}$ in any data-subspace $\mathbb{D}(\mathbf{x}, \theta)$. For aggregates such as `AVG` the previous assumption does not hold, hence representing our explanations with high-order polynomial functions will be problematic. In addition, a high-order polynomial becomes increasingly complex to interpret for an increasing

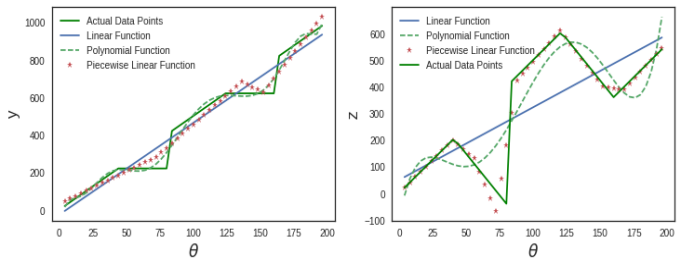


Figure 3: (Left) Explanation actual and approximation functions for `COUNT`, where x-axis is the radius θ of a count query and y-axis is the count y given a fixed center x; (right) Explanation actual and approximation functions for `AVG` z vs radius θ of an average query given a fixed center x.

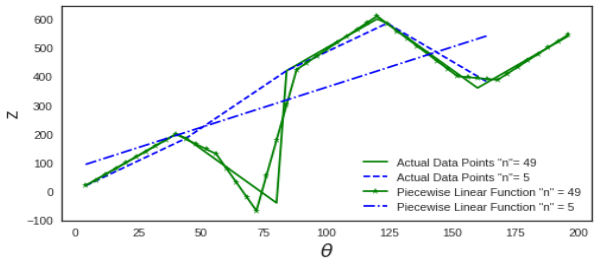


Figure 4: A limiting number of n pairs (θ, y) causes loss of significant information in explanation approximation $y = f(\theta; \mathbf{x})$ given a fixed center x.

order. Therefore, we avoid using polynomials and resort to linear functions. But, linear functions will still have trouble representing our explanations accurately. For instance, the $z =$`AVG(call_time)` given a fixed center $\mathbf{x}$ might decrease for a larger radius θ. Even for `COUNT`, a linear function might be an incorrect representation as the output y might remain constant within an interval θ and then increase abruptly. To summarize, we are faced with the following challenges: **C1**we do not adopt high-order polynomials for approximating function f, as they become increasingly complex to interpret and might fail when result is monotonically increasing; and/or **C2** we adopt use a single linear function for approximating function f as results that oscillate cannot be accurately represented. The above-mentioned challenges C1 and C2 are depicted in Figure 3, where Figure 3 (left) shows that output $y = f(\theta; \mathbf{x}) \in \{0, 1, \ldots, n\}$ is monotonically increasing w.r.t θ given a fixed center $\mathbf{x}$ and the resulting *linear* and *polynomial* fitting/approximation functions fail to resolve the aforementioned issues. The same holds true for the average output $z = f(\theta; \mathbf{x}) \in \mathbb{R}$ in Figure 3 (right). Hence, we need to find an adjustable linear function that is able to capture such irregularities and conditional nonlinearities. The most prominent solution is to adopt and approximate the explanation function f for count and average queries through the Piecewise Linear Function (PLR) a.k.a. *segmented regression*. A PLR approximation of f is able to address the shortcoming of the previously suggested explanations, described in the C1 and C2 by finding and fitting the *best* multiple linear functions, as illustrated in Figure 3. We now provide the definition of an explanation:

Definition 6. (Explanation Function) Given a count / average AQ $\mathbf{q} = [\mathbf{x}, \theta]$ over a center of interest $\mathbf{x}$, an explanation function $f(\theta; \mathbf{x})$ is defined as the fusion of piecewise linear functions derived by the approximation of f over count / average AQs $\mathbf{q}' = [\mathbf{x}, \theta']$ with radii θ' around radius θ and fixed center $\mathbf{x}$.

4. THE XAQ FRAMEWORK

The challenge in approximating the explanation function f defined in Definition 6 is in seeking approximation functions to explain the way count y and average z varies as radius θ changes with and without access to base data and to query workload characteristics. Specifically, we elaborate on two explanation schemes: *Data-Gnostic (DG)* and *Data-Agnostic (DAG)*.

4.1 Data-Gnostic Explanation Approximation

The DG scheme relies on accessing the underlying DB data multiple times to obtain the pairs (θ', y) and (θ', z), respectively, which are in turn used to approximate a PLR function $f(\theta; \mathbf{x})$ given a fixed center $\mathbf{x}$. For instance, let us assume we desire to explain a count result y coming from the aggregate operator `COUNT`. To explain the actual answer y, using DG, we need to issue multiple count queries $\mathbf{q} = [\mathbf{x}, \theta']$ at the same location $\mathbf{x}$ with $\theta' \leq \theta$. We do that for an arbitrary n number of radius values θ' smaller than θ and obtain the n radius-answer pairs $\{(\theta_1, y_1), \ldots (\theta_n, y_n)\}$. For those pairs, we approximate a PLR function $f(u; \mathbf{x})$, $u \in \{\theta_1, \ldots, \theta_n\}$ with $0 < \theta_1 < \cdots < \theta_n \leq \theta$ and obtain the resulting explanation function. One of the core strengths of this approach is that we can get an increasingly accurate PLR function depending on the number of n pairs. Limiting the number of n pairs prohibits the detection of change in the behavior of $y = f(u; \mathbf{x}), u \leq \theta$, as shown in Figure 4. However, a major drawback of such an approach is its lack of efficiency. Generating such an approximation of explanation requires the execution of multiple aggregate queries over a fixed center $\mathbf{x}$. With an increase in data, the response time of getting an explanation will drastically deteriorate leaving the analyst waiting long for a detailed/accurate explanation. Efficiency can be gained at the expense of approximation accuracy by limiting the number of n pairs, however, we risk missing valuable information of the behavior of f, as illustrated in Figure 4. Therefore, we seek an alternative approach, which satisfies the desiderata: **D1 Accuracy**: It can handle an increased number of n pairs (executed queries and returned answers), thus, increasing the explanation approximation accuracy; **D2 Scalability** and **Efficiency**: It has constant high performance, even with an increase in data size.

4.2 Data-AGnostic Explanation Approximation

DAG utilizes previously executed AQs to train statistical learning models that are able to accurately approximate the explanation function f for any center of interest $\mathbf{x}$. We consider such an approach because it can fulfill the desiderata on accuracy, scalability, and efficiency. Access to the underlying data is no longer needed in DAG, thus, efficiency is guaranteed; this is due to the fact that the statistical learning models learn the various query patterns to approximate

explanation functions $f(\theta; \mathbf{x}), \forall \mathbf{x} \in \mathbb{R}^d$. In addition, exploiting insights gained and extracted from past queries assists in *building* accurate approximations of explanation functions. To utilize the DAG approach, previously executed queries are used to train a statistical learning model to approximate an explanation function f only from past issued queries *and* their results.

One could utilize the available past queries and results by training a model that has the form of a PLR function. Hence, for m previously executed queries $\{\mathbf{q}_i\}$, we can obtain m training pairs $\{(\theta_i, y_i)\}, i \in [1 \dots m]$. The difference of this vs the DG scheme is that, now, the center $\mathbf{x}_i$ for each query $\mathbf{q}_i = [\mathbf{x}_i, \theta_i]$ varies, because these queries were issued at different locations/centers with different radii. Hence, we train a PLR function $\hat{f}$ for approximating the actual explanation f using these pairs and for each subsequent and unseen query that we seek to explain, we provide the resulting approximation model function $\hat{f}$ as our explanation. Intuitively, such an approach will yield better performance because we just store the resulting approximation model function $\hat{f}$ and given a new AQ, we can present the function $\hat{f}$ as its explanation.

Formally, given a well defined explanation loss $\mathcal{L}(f, \hat{f})$ (defined later) between the *actual explanation function* $f(\theta; \mathbf{x})$, that is, the explanation provided by accessing the base data, and *approximation* model function $\hat{f}(\theta; \mathbf{x}) \in \mathcal{F}$, we seek the optimal approximation model function $\hat{f}^*$ that minimizes the Expected Explanation Loss (EEL) for *all possible* queries (all locations and radii):

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

where $p(\theta, \mathbf{x})$ is the probability density function of the queries $\mathbf{q} = [\mathbf{x}, \theta]$ over the query workload. Eq(1) is the objective minimization loss function given that we see the optimal model approximation function to explain the relation between y or z against radius θ for any given/fixed center $\mathbf{x}$. However, accuracy will be problematic as it seems inherently wrong that one such function can explain *all* the queries at any location $\mathbf{x} \in \mathbb{R}^d$ with any radius $\theta \in \mathbb{R}$. This is clearly shown in Figure 5(Left), approximating a global PLR with all the previous executed queries. The green line denotes the resulting PLR function which acts as the *global* explanation for all the *possible* queries. Such an explanation is not accurate because: (i) The radius θ can generally be different for different queries. This is shown in Figure 5(Left) as θ is separated into three distinct intervals. For instance, data scientists at the Police department will issue AQs with a different radius to compare if crimes are centered within a small radius at a given location or they increase at a larger radius; (ii) At different locations $\mathbf{x}$, the result y or z will be different. This is clear from Figure 5(Left) as the variance at each one of the three distinct intervals becomes increasingly high. [2]

4.2.1 Local DAG Explanation Approximation

Having described both DG and DAG, we introduce the *local* DAG (LDAG), which follows from the above discussion. In this approach, LDAG addresses the shortcomings of DAG and has the advantages of both DAG and DG. Specifically, it does not incur the cost of accuracy as in DAG, has superior efficiency over DG and high accuracy comparable to that of DG. The way to achieve this is by abolishing the concept of a global model and introducing *local model approximation functions*. What this means is that we no longer train a global optimal model $\hat{f}$ that minimizes (1) and serves as an explanation for all possible queries. Instead, we train a number of local optimal models $\hat{f}_k$ that can explain a *subset* of those queries. We call those models as Local PLR Models (LPM).

We still utilize the m previously executed queries. However, instead of training one PLR model $\hat{f}$ using the m pairs, we cluster the query-space $\mathbb{Q} \subset \mathbb{R}^{d+1}$ to yield a number of K query sub-spaces $\mathbb{Q}_k$. We can then take these local subspaces and train a PLR function $\hat{f}_k$ for each one, $k \in [K]$. Hence, the obtained K LPMs are essentially our multiple possible explanations. Intuitively, this approach makes more sense as queries that are issued at relatively close proximity to each other, location-wise, and that have similar radii will tend to have similar results and in turn will behave more or less the same way. The result of our approach is shown in Figure 5(Right). The whole query-space is now partitioned into K local query sub-spaces $\mathbb{Q}_k$ containing *similar* queries, where the similarity between two queries $\mathbf{q}$ and $\mathbf{q}'$ is defined in Definition 5. The resulting fused explanation for each smaller local sub-space has more accuracy as the clusters have less variance for y (or z) and θ values are similar within such sub-spaces, as shown in Table 1.

Formally, to be aligned with our objective in (1), i.e., minimizing EEL, we seek K local approximation model functions $\hat{f}_k \in \mathcal{F}, k = 1 \in [K]$, such that for each query $\mathbf{q}$ belonging to a partition/cluster k, $\mathbb{Q}_k$, the summation of the local EELs is minimized:

$$\mathcal{J}_0(\{\hat{f}_k\}) = \sum_{\hat{f}_k \in \mathcal{F}} \int_{\mathbf{q} \in \mathbb{Q}_k \subset \mathbb{R}^{d+1}} \mathcal{L}(f(\theta; \mathbf{x}), \hat{f}_k(\theta; \mathbf{x})) p_k(\mathbf{q}) d\mathbf{q}, \quad (2)$$

[2]Figure 5 was generated synthetically to demonstrate that queries can have different radii and can yield different results at different locations.

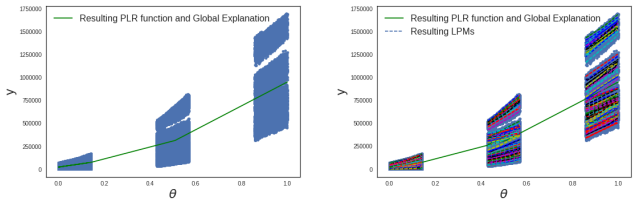


Figure 5: (Left) One global PLR function $y = f(\theta; \mathbf{x})$ for all possible queries yields small accuracy. The three distinct clusters represent multiple queries with a given radius and result. (Right) Local Models focus on smaller local spaces yielding more accurate explanations for related queries.

Table 1: Variance for both θ and result y(or z) is reduced by the use of Local models.(Results from experimental workload described in Experimental Evaluation)

	Global Model $\hat{f}$	**Local Models $\hat{f}_k$**
Variance θ	0.124	0.0017
Variance y	$1.5 \cdot 10^{10}$	$3.5 \cdot 10^{8}$

where $p_k(\mathbf{q})$ is the probability density function of the query vectors belonging to a query sub-space $\mathbb{Q}_k$. We again resort to our example for more clarification for LDAG. As is evident from our example, Data Scientists are tasked with investigating certain regions. Hence, the queries that each one of them will issue is at different locations $\mathbf{x}$, with varying θ. Therefore, the log containing all the queries that each one of them has issued will be diverse. It then makes more sense to find the specific regions / subspaces $\mathbb{Q}_k$ that they targeted, and construct explanations over those regions that contain similar queries and, thus, similar results. This approach is guaranteed to be more accurate.

4.3 Measuring Expected Explanation Loss

We define *explanation loss*, $\mathcal{L}(f, \hat{f})$ the discrepancy of the actual explanation function f due to the approximation of explanation $\hat{f}$. For evaluating $\mathcal{L}$, we propose three different aspects in which the expected *explanation loss* can be measured: *(1) information-theoretic aspect*, where we evaluate the loss as the extra bits of information needed when approximating the explanation function, $f(\theta; \mathbf{x})$ with $\hat{f}(\theta; \mathbf{x})$, *(2) statistical aspect*, where the goodness of fit of our explanation function is measured over the actual explanation and *(3) a model-based divergence* approach in which we compare the model fitting behavior of the approximated explanation function with the actual one. The corresponding metrics for the explanation loss are as follows:

Information-theoretic: Explanation loss from this aspect is measured using the Kullback–Leibler (KL) divergence: Concretely, the result is a scalar value denoting the amount of information lost when one chooses to use the approximated explanation function instead of the actual function. The KL divergence over two explanations is then defined as:

$$\begin{aligned} \mathcal{L}(f(\theta;\mathbf{x}), \hat{f}(\theta;\mathbf{x})) &= KL(p(y;\theta,\mathbf{x})||\hat{p}(y;\theta,\mathbf{x})) \quad (3) \\ &= \int p(y;\theta,\mathbf{x}) \log \frac{p(y;\theta,\mathbf{x})}{\hat{p}(y;\theta,\mathbf{x})} dy, \end{aligned}$$

where $p(y;\theta,\mathbf{x})$ and $\hat{p}(y;\theta,\mathbf{x})$ is the probability density function of the aggregate query result y for explaining a query given the actual and the approximated explanation function, respectively.

Goodness of Fit: The explanation loss is measured using statistical learning metrics for model fitting, namely the coefficient of determination R^2. This metric indicates how much of the variance generated by $f(\theta;\mathbf{x})$ can be explained using the approximation $\hat{f}(\theta;\mathbf{x})$. This represents the goodness-of-fit of an approximated explanation function over the actual explanation function. The explanation loss between f and $\hat{f}$ explanations can then be computed as $\mathcal{L}(f(\theta;\mathbf{x}), \hat{f}(\theta;\mathbf{x})) = 1 - R^2$.

Model-Based Divergence: This aspect measures the *similarity* between two explanation functions f and $\hat{f}$ by calculating the inner product of their parameter vectors $\mathbf{a}$ and $\hat{\mathbf{a}}$ in a given parameter space, respectively. We adopt the *cosine similarity* of the two parameter vectors:

$$\mathcal{L}(f(\theta;\mathbf{x}), \hat{f}(\theta;\mathbf{x})) = -\frac{\mathbf{a}\cdot\hat{\mathbf{a}}}{\|\mathbf{a}\|\|\hat{\mathbf{a}}\|} \quad (4)$$

It can then be interpreted as the two functions being similar if $\mathcal{L}(f(\theta;\mathbf{x}), \hat{f}(\theta;\mathbf{x})) = -1$ and dissimilar if $\mathcal{L}(f(\theta;\mathbf{x}), \hat{f}(\theta;\mathbf{x})) = 1$.

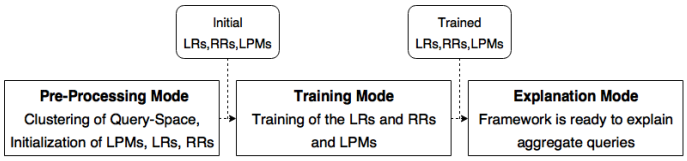


Figure 6: Training and explanation modes of LDAG.

5. XAQ COMPUTING METHODOLOGY

5.1 Methodology Overview

The proposed methodology for computing explanations is split into three *modes* illustrated in Figure 6. The *Pre-Processing Mode* is an initialization step for identifying the optimal number of LPMs and an initial approximation of their parameters. Then, we move to the *Training Mode*, where the LPMs' parameters are incrementally optimized to minimize the objective function (2). The outcome of this mode is the *Explanation Mode*, where the framework is ready to explain aggregate queries.

5.1.1 Pre-Processing Mode

Our available information is a training set $\mathcal{T}$ of m pairs with previously executed queries $\mathbf{q}$ and their corresponding aggregate results, e.g., count y: $\mathcal{T} = \{(\mathbf{q}_1, y_1), \ldots, (\mathbf{q}_m, y_m)\}$, where $\mathbf{q} = (\mathbf{x}, \theta)$. The set $\mathcal{T}$ is the input to the *Pre-Processing Mode*. The fundamental task of this mode is to partition (quantize) the query space $\mathbb{Q}$ defined by the observed/issued queries $\mathbf{q} \in \mathcal{T}$ into K clusters, sub-spaces $\mathbb{Q}_k$, in which queries with similar $\mathbf{x}$ are grouped together. Each one of these clusters, is then further quantized into L sub-clusters, as queries with similar $\mathbf{x}$ are separated with regards to their θ parameter. Therefore, this refers to a hierarchical query space quantization (first level partition w.r.t. $\mathbf{x}$ and second level partition w.r.t. θ) where each Level-1 (L1) cluster $\mathbb{Q}_k, k = 1, \ldots, K$ is associated with a number of Level-2 (L2) sub-clusters $\mathbb{U}_{kl}, l = 1, \ldots, L$ in the θ space. For each L1 cluster $\mathbb{Q}_k$ and L2 sub-clusters $\mathbb{U}_{kl}$, we assign a L1 representative, hereinafter referred to as Location Representative (LR) and a L2 representative, hereinafter referred to as Radius Representative (RR), in the center/location space and in the radius space, respectively. 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 radius value of the radii of all queries, whose radius values belong to $\mathbb{U}_{kl}$. After this hierarchical quantization of the query space, the task is to associate a PLR $f_{kl}(\theta;\mathbf{x})$ with *each* L2 sub-cluster $\mathbb{U}_{kl}$, given that the center parameter $\mathbf{x}$ is a member of the corresponding L1 cluster $\mathbb{Q}_k$.

5.1.2 Training Mode

The *Training Mode* optimizes the clustering parameters (LR and RR representatives) of the pre-processing mode in order to minimize the objective function (2). This optimization process is achieved incrementally by processing each pair $(\mathbf{q}_i, y_i) \in \mathcal{T}$ on-line. The fundamental task is that each query $\mathbf{q}_i$ from $\mathcal{T}$ 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. After a pre-specified number of projected queries to a L1 cluster $\mathbb{Q}_k$ and to a specific L2

sub-cluster $\mathbb{U}_{kl}$, then the corresponding PLR $y = f_{kl}(\theta; \mathbf{x})$ is trained to associate the result y with the θ values in $\mathbb{U}_{kl}$ given that their corresponding centers $\mathbf{x}$ belong to the $\mathbb{Q}_k$.

5.1.3 Prediction Mode

In *Prediction Mode* no more modifications of LRs, RRs, and the derived PLRs is allowed. Based on the structure of the L1/2 representatives and their associated approximation explanation functions PLRs, the model is ready to provide explanations for AQs. This is achieved with either an LDAG approach or a DG approach. As described in Section 3, generating explanations using a DG approach is achieved by fitting a number n of pairs (θ, y) (or (θ, z)) using a global PLR function. The pairs are directly obtained by executing the same number n queries similar to the query issued by the analysts. The only difference of these queries would be θ and their resulting answer y (or z), location $\mathbf{x}$ remains the same. The returned PLR explanation function, given by DG is then an approximate explanation for query $\mathbf{q}$. On the other hand, LDAG utilizes the trained LPMs thus yielding higher efficiency because *no access to data* is needed for delivering explanations. For a given query, the system finds the closest L1 LR and then, based on a combination of the L2 RRs and their associated LPMs, returns an explanation as a fusion of diverse PLR functions derived by the L2 level. We elaborate on this fusion of L2 PLRs in Section 5.3. Figure 7 sums up the result of all three modes and how an explanation is given to the user.

5.2 Solution Fundamentals

5.2.1 Explanation Representatives

This mode is comprised of three phases. The training queries from set $\mathcal{T}$ are first clustered based on their locations $\mathbf{x}$, then each L1 cluster is partitioned into L2 sub-clusters based on parameter θ of the queries. For each L2 sub-cluster, we train a PLR approximation model using queries belonging to that sub-cluster.

First Phase: L1 Query Quantization. In the query quantization phase we adopt the K-Means [22] clustering algorithm to identify the L1 LRs based on the queries' centers $\mathbf{x}$. The outcome of this phase is the K L1 cluster center representatives, LRs. The LRs are initially random location vectors $\mathbf{w}_k \in \mathbb{R}^d, k = 1, \ldots, K$, and are then refined by the clustering algorithm as the mean vectors of the query-centers that are assigned to each LR. Formally, this phase finds the optimal mean vectors $\mathcal{W} = \{\mathbf{w}_k\}_{k=1}^K$, which minimize the L1 Expected Quantization Error (EQE):

$$\mathcal{J}_1(\{\mathbf{w}_k\}) = \mathbb{E}[\min\|\mathbf{x} - \mathbf{w}^*\|^2 | \mathbf{w}^* = \arg\min_{k=1,\ldots,K} \|\mathbf{x} - \mathbf{w}_k\|^2], \quad (5)$$

where $\mathbf{x}$ is the location for a given query $\mathbf{q} = [\mathbf{x}, \theta] \in \mathcal{T}$. The $\mathbf{w}_k$ is the mean center vector of all queries $\mathbf{q} \in \mathbb{Q}_k$ associated with $\mathbf{w}_k$. A limitation of the K-Means algorithm is that we need to specify K number of LRs in advance. Therefore, we devised a simple strategy to find a near-optimal number K. By running the clustering algorithm in a loop, 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 minimize the Sum of Squared Quantization Errors (SSQE), which is equal to the summation of distances, of all queries from their respective LRs, the algorithm is available in the Appendix D.

We start with an initial K value and gradually increment until the SSQE yields an improvement no more than a predefined threshold $\epsilon > 0$. The result of the clustering algorithm is the set of L1 LRs $\mathcal{W} = \{\mathbf{w}_k\}_{k=1}^K$, which minimize (5).

Second Phase: L2 Query Quantization. We utilize K-Means again, but this time over each L1 cluster of queries created by the L1 query quantization phase. 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. Specifically, we identify the L2 RRs over the radii of those queries from $\mathcal{T}$ whose closest LR is $\mathbf{w}_i$. Then, by executing the L-Means over the radius values from those queries we derive the corresponding set of radius representatives $\mathcal{U}_i = \{u_{i1}, \ldots, u_{iL}\}$, where each u_{il} is the mean radius of all radii in the l-th L2 sub-cluster of the i-th L1 cluster.

Third Phase: PLR Initial Fitting. We fit L PLR functions $f_{kl}(\theta; \mathbf{w}_k)$ for each L2 sub-cluster $\mathbb{U}_{kl}$ for those θ values that belong to the L2 RR u_{kl}. The fitted PLR captures the *local* statistical dependency of θ around its mean radius u_{kl} given that $\mathbf{x}$ is a member of the L1 cluster represented by $\mathbf{w}_k$. Given objective (2), for each local L2 subcluster, the approximate function $\hat{f}_{kl}$ minimizes the conditional Local EEL:

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

conditioned on the closeness of query center $\mathbf{x}$ and radius θ to the L1 and L2 quantized query space $\mathbb{Q}_k$ and $\mathbb{U}_{kl}$, respectively. Minimizing objective $\mathcal{J}_2$ in (6) is not trivial due to the double conditional expectation over each query center and radius. To initially minimize this local objective, we use Multivariate Adaptive Regression Splines (MARS) [19] as the approximate model explanation function f_{kl}. In this context, our approximate $\hat{f}_{kl}$ has the following form:

$$\hat{f}_{kl}(\theta; \mathbf{w}_k) = \beta_0 + \sum_{i=1}^{M} \beta_i h_i(\theta), \quad (7)$$

where the basis function $h_i(\theta) = \max\{0, \theta - \lambda_i\}$, with λ_i being the *hinge* points. Essentially, this creates M linear regression functions. The number of M linear regression functions is automatically derived by MARS using a threshold which checks for convergence w.r.t R^2. 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 all over the query space. Figure 7 illustrates the two levels L1 and L2 of our explanation methodology, where each LR and RR are associated with a MARS model. The framework then optimizes the parameters:

$$\mathcal{W} = \{\mathbf{w}_k\}, \mathcal{U} = \{u_{kl}\}, \mathcal{M} = \{(\beta_i, \lambda_i)_{kl}\} \quad (8)$$

with $k = 1, \ldots, K$, $l = 1, \ldots, L$, $i = 1, \ldots, M$, which are stored for fine tuning and explanation/prediction.

5.2.2 Statistical Learning Models

Given the hierarchical query space quantization and local PLR fitting, the training mode fine-tunes the parameters in

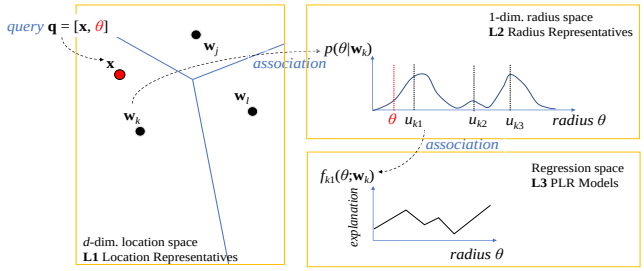


Figure 7: The XAQ Framework Rationale: Query mapping to L1 Location Space, association to L2 Radius Space, and explanation to L3 PLR regression space.

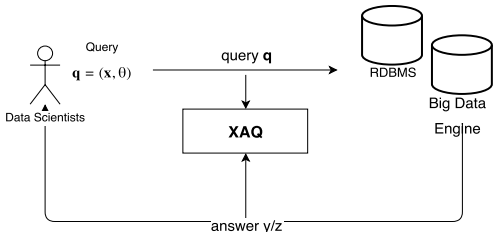


Figure 8: XAQ Framework Training: XAQ observes inputs (issued queries) and associated outputs (answers from executed queries) to train and adjust its local explanation models in an on-line mode.

(8) to optimize both $\mathcal{J}_1$ in (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 a DBMS/Big Data Engine. An example of this on-line training methodology is illustrated in Figure 8, Therefore, a typical use case would be : 1) the analyst issues an AQ $\mathbf{q} = [\mathbf{x}, \theta]$ 2) the system answers with result y (or z) 3) XAQ exploits pairs $(\mathbf{q}, y/z)$ to train its main constructs.

Our Training Mode is comprised of three steps. First is the *assignment* step, where the incoming query $\mathbf{q}$ is associated with an L1, L2 and PLR, by finding the closest representatives at each level. The second step is the *on-line adjustment* step, where the LR, RR and PLR parameters are gradually modified w.r.t. the associated incoming query in the direction of minimizing the objectives $\mathcal{J}_1$ and $\mathcal{J}_2$. Finally, the *off-line adjustment* step conditionally retrains any PLR fitting model associated with any incoming query so far. This step is triggered when a predefined retraining value is reached, or when the number of queries so far exceeds a threshold value.

Assignment Step. For each executed query-answer pair $(\mathbf{q}, y)/(\mathbf{q}, z)$, we project $\mathbf{q}$ to its closest L1 LR using only the query center $\mathbf{x}$ based on (5). Obtaining the projection $\mathbf{w}_k^*$ allows us to *directly* retrieve the associated RRs $\mathcal{U}_k^* = \{u_{1k}^*, \ldots, u_{Lk}^*\}$. Finding the best L2 RR in $\mathcal{U}_k^*$ is, however, more complex than locating the best LR. In choosing one of the L2 RRs, we consider both *distance* and the associated *prediction error*. Specifically, the *prediction error* is obtained by the PLR fitting models of each RR from the set $\mathcal{U}_k^*$. Hence, in this context, we first need to consider the distance of our query $\mathbf{q} = [\mathbf{x}, \theta]$ to all of the RRs in $\mathcal{U}_k^*$ focused on the absolute distance in the radius space:

$$|\theta - u_{kl}|, \forall u_{kl} \in \mathcal{U}_k^*, \tag{9}$$

and, also, the *prediction error* given by each RR's associated PLR fitting model $\hat{f}_{kl}$. The prediction error is obtained by the squared difference of the actual result y of the query $\mathbf{q}$ and the predicted outcome of the local PLR fitting model $\hat{y} = \hat{f}_{kl}(\theta; \mathbf{w}_k^*)$:

$$(y - \hat{f}_{kl}(\theta; \mathbf{w}_k^*))^2, l = 1, \ldots, L \tag{10}$$

Therefore, for assigning a query $\mathbf{q}$ to a L2 RR, we combine both distances in (9) and (10) to get the assignment distance in (11), which returns the RR in $\mathcal{U}_k^*$ which minimizes:

$$l^* = \arg\min_{l \in [L]} (\mu|\theta - u_{kl}| + (1 - \mu)(y - \hat{f}_{kl}(\theta; \mathbf{w}_k^*))^2) \tag{11}$$

The adjustable parameter μ shifts the weight of our decision towards the *distance*-wise metric or the *prediction*-wise metric. For instance, if we prefer to assign queries to RRs that have a smaller prediction error, we decrease the value of μ, and increase μ if we desire our decision to be based only on the distance in radius space.

Remark 1. *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 model may have lower prediction error even though it is not at the shortest distance. Intuitively, this holds true, as some fitting models might be able to make better generalizations even if their RRs are further apart. Therefore, we introduce the weighted-distance in (11) to account for this and make more sophisticated selections.

On-line Adjustment Step. This step optimally adjusts the positions of the chosen LR and RR so that training includes the new query. Their positions are shifted using Stochastic Gradient Descent (SGD) [7] over the $\mathcal{J}_1$ and $\mathcal{J}_2$ w.r.t. the $\mathbf{w}$ and θ 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 (11) 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 $\mathbf{q} = [\mathbf{x}, \theta]$ projected onto the closest L1 $\mathbf{w}_{k^*}$ and L2 u_{k^*,l^*}, the update rule for u_{k^*,l^*} that minimizes $\mathcal{J}_2$ is:*

$$\Delta u_{k^*,l^*} \leftarrow \alpha \mu sgn(\theta - u_{k^*,l^*}) \tag{12}$$

PROOF. See the proof in Appendix C.1. □

The $\alpha \in (0, 1)$ 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 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 (12) for L2 RR $u_{k,l}$, it converges to the median of the θ values 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* $\mathbf{q} = [\mathbf{x}, \theta]$ *with* $\mathbf{x} \in \mathbb{Q}_k$*, it holds true for* $u_{kl} \in \mathbb{U}_{kl}$ *that:*

$$\int_0^{u_{kl}} p(\theta|\mathbf{w}_k)d\theta = \frac{1}{2}. \tag{13}$$

PROOF. See the proof in Appendix C.2. □

Using SGD, the θ_{k^*,l^*} converges to the median of all radius values of all the queries in the local L2 sub-cluster in an on-line manner. The alternative would have been to calculate the median of all training queries associated so far, every time a new query was associated. Using SGD, we drastically cut down our training time for convergence to the optimal positions of L2 RRs.

Mini-batch Adjustment Step. The mini-batch adjustment step is used to conditionally re-train the PLRs to reflect the changes by (12) 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 PLR/MARS functions, 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. In order 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 parameters $(\beta_{kli}, \lambda_{kli})$ PLR/MARS of the fitting $\hat{f}_{kl}$ given a conditionally optimal L2 RR u_{kl} and for every processed query we increment a counter. Once we reach a predefined number of projected queries-answers, we re-train every PLR/MARS model that was affected by projected training pairs. Once the fitting models are retrained, then a new era of on-line adjustment begins until the end of the training pairs or the convergence of the L2 RRs.

5.3 Explanation Provision

After *Pre-Processing* and *Training* explanations can be provided for *unseen* AQs. No modification of the model parameters are performed and explanations do not require executing any AQs. The explanations are *predicted* by the fusion of the trained/fitted PLRs based on the incoming queries. The process is as follows. The analyst issues a query $\mathbf{q}$, $q \notin \mathcal{T}$. To obtain the explanation of the result for the given query $\mathbf{q} = [\mathbf{x}, \theta]$, XAQ finds the closest $\mathbf{w}_k^*$ LR and then directly locates all of the associated RRs of the $\mathcal{U}_k^*$. The resulting explanation uses all *selected* associated L PLRs fitting models $\hat{f}_{k^*,l}$. The selection of the most relevant PLR functions for explaining $\mathbf{q}$ is based on the boolean indicator $I(\theta, u_{k^*,l})$, where $I(\theta, u_{k^*,l}) = 1$ if we select to explain the result/answer y of the query based on the area around θ represented by the L2 RR $u_{k^*,l}$; 0 otherwise. Specifically this indicator is used to denote which is the domain value for the θ radius in order to deliver the dependency of the answer y within this domain as reflected by the corresponding PLR $\hat{f}_{k^*,l}(\theta; \mathbf{w}_{k^*})$. In other words, as the query's radius increases or decreases, XAQ selects different PLR models to represent the answer y, which is associated with the RR that is closer to the query's θ. Using this method, L possible explanations for $\mathbf{q}$ exist and the selection of the most relevant PLR fitting model for the current radius domain is determined when $I(\theta, u_{k^*,l}) = 1$ and, simultaneously, when:

$$\begin{cases} 0 \leq \theta < u_{k^*,1} + \frac{1}{2}|u_{k^*,1} - u_{k^*,2}| \text{ and } l = 1 \\ u_{k^*,l-1} + \frac{1}{2}|u_{k^*,l-1} + u_{k^*,l}| \leq \theta < u_{k^*,l} + \frac{1}{2}|u_{k^*,l} + u_{k^*,l+1}| \\ \qquad \text{and } l < L \\ u_{k^*,l-1} + \frac{1}{2}|u_{k^*,l-1} + u_{k^*,l}| \leq \theta \text{ and } l = L \end{cases} \tag{14}$$

The domain radius in (14) shows the radius interval boundaries for selecting the most relevant PLR fitting model for explanation in the corresponding radius domain. Therefore, when visualizing the explanation, we switch between PLR models according to the indicator function. For instance, from radius 0 up to the point where the first RR would still be the closest representative, indicated as $\leq u_1 + \frac{1}{2}|u_1 - u_2|$, we use the first PLR fitting model. Using this selection method, we can derive an accurate explanation for the AQ. The method's result is shown in Figure 9 where a combination (fusion) of PLR models gives a better explanation than a single model. A single compact equation can then represent the returned explanation when we evaluate (15):

$$\sum_{l=1}^{L} I(\theta, u_{k^*,l})\hat{f}_{k^*,l} \tag{15}$$

The result of this summation is a single PLR function, since the boolean indicator returns 1 only for the selected PLR.

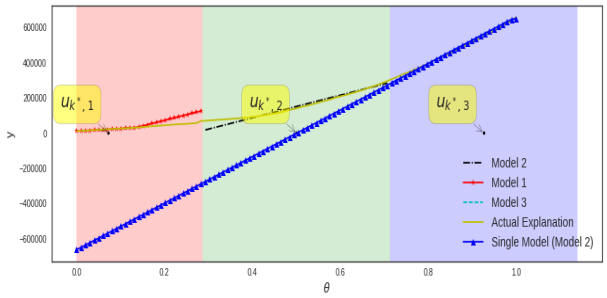


Figure 9: Better explanations with > 1 relevant models as evidenced by the intervals (radius domains), where each model is selected along with their associated L2 RRs.

6. EXPERIMENTAL EVALUATION

6.1 Experimental Setup & Metrics

Data Sets: We use a real dataset $\mathcal{B} = \{\mathbf{x}_i\}_{i=1}^N, \mathbf{x} \in \mathbb{R}^2$ with cardinality $|\mathcal{B}| = N = 6{\cdot}10^6$ containing recorded crimes for the state of Chicago [1]. We create 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 one of those queries is a 3-d vector $\mathbf{q} = [\mathbf{x}, \theta]$ with answer y or $\mathbf{q} = [\mathbf{x}, \theta]$ which answer z where $\mathbf{x} = [x_1, x_2] \in \mathbb{R}^2$ is the location/center of a query, $\theta \in \mathbb{R}$ is its radius and $y \in \mathbb{N}$ or $z \in \mathbb{R}$ the result obtained by executing the query against real dataset $\mathcal{B}$. We scale $\mathcal{B}$ and $\mathcal{T}$ in all their dimensions, restricting their values in $[0, 1]$. We must normalize as θ and y, z may differ by varying orders of magnitude. In cases where y, z are large, the outcome of (11) will only be driven by the prediction error, $(y - \hat{y})^2$ and the value for $|\theta' - u_{k^*l}|$ will be negligible. By normalizing

Table 2: Synthetic Workloads $\mathcal{V}$.

	Location x	Radius θ
Gauss-Gauss	$\sum_{i=1}^{C} \mathcal{N}(\mu_i, \Sigma_i)$	$\sum_{i=1}^{J} \mathcal{N}(\mu_i, \sigma_i^2)$
Gauss-Uni	$\sum_{i=1}^{C} \mathcal{N}(\mu_i, \Sigma_i)$	$\sum_{i=1}^{J} U(v_i, v_i + 0.02)$
Uni-Gauss	$\sum_{i=1}^{C} U(v_i, v_i + 0.04)$	$\sum_{i=1}^{J} \mathcal{N}(\mu_i, \sigma_i^2)$
Uni-Uni	$\sum_{i=1}^{C} U(v_i, v_i + 0.04)$	$\sum_{i=1}^{J} U(v_i, v_i + 0.02)$

and scaling we ensure that the same weight is given to the query's answer and its radius. 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 $(\mathbf{q}, y)$, i.e., $\mathbf{q} \in \mathcal{V} : \mathbf{q} \notin \mathcal{T}$.

AQ workload ($\mathcal{V}$) distributions: In order to be as realistic and comprehensive as possible we generated $\mathcal{V}$ with our running example in mind. We used 2-dim./1-dim. Gaussian and 2-dim./1-dim. Uniform distributions for location $\mathbf{x}$ and radius θ of our queries, respectively. Hence, we obtain four variants of $\mathcal{V}$, each having different multi-modal distributions for the query locations & radii, as shown in Table 2. The parameters C and J signify the number of mixture distributions within each variant. Multiple distributions comprise $\mathcal{V}$ as we desire to analyze our model behavior issuing queries against the real dataset $\mathcal{B}$. For instance, given a number $C \times J$ of analysts, each one of them might be assigned to different geo-locations, hence parameter C, issuing queries with different radii, hence J, given their objectives. For the parameters μ (mean) and v of the location distributions in Table 2, we select points uniformly at random on the map ranging in [0,1]. The covariance Σ_i of each Gaussian distribution, for location $\mathbf{x}$, is set to 0.0001. The number of distributions/query-spaces was set to $C \times J$, where $C \in \mathbb{N}$ and $J \in \mathbb{N}$ with $C = 5$ and $J = 3$, thus, a mixture of 15 query distributions. The radii covered $2\% - 20\%$ of the data space, i.e., μ_i for θ was randomly selected in $[0.02, 0.2]$ for Gaussian distributions, with small σ^2 for the Gaussian distributions, and v_i was in $[0.02, 0.18]$ for Uniform distributions. Further increasing that number would mean that the queries generated would cover a much greater region of the space, thus, making the learning task much easier. The variance σ_i^2 for Gaussian distributions of θ was set to 0.0009 to leave no overlap with the rest of the $J - 1$ distributions. We deliberately leave no overlapping within θ distributions as we assume there is a multi-modal mixture of distributions with different radii used by analysts.

Our implementation was in Python 2.7. Experiments ran single threaded on a Linux Ubuntu 16.04 using an i7 CPU at 2.20GHz with 6GB RAM. The number of n pairs was set to $n = 10$. We evaluate the explanations offered by LDAG and DG. In addition to the MARS model used for DG we also include a *Linear Regression* (LR) model to see if the irregularities captured by MARS are also evident in Linear Regression. For every query in the evaluation set $\mathcal{V}$, we take its radius and generate n evenly spaced radii over the interval $[0.02, \theta']$, where 0.02 is the minimum radius used. For query $q = [\mathbf{x}, \theta]$ or $q = [\mathbf{x}, \theta]$ we generate and find the answer for n sub-queries, $ST = \{([\mathbf{x}, \theta_1], y_1), \cdots ([\mathbf{x}, \theta_n], y_n)\}$. The results of these queries constitute the *Actual Explanation* (AE) which is a collection of n pairs of sub-radii and y.

Evaluation & Performance Metrics: Coefficient of Determination (Model Fitting) is computed as

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

where the numerator is the Sum of Squared Errors (SSE) and denominator is the variance of y given the mean $\overline{y}$ value. R^2 measures the accuracy of approximation (model fitting) of explanations where the closer the value to 1 the better the approximation to the actual explanation curve (AE).

Slope Error (Model Divergence): The model slope indicates the rate with which y changes as θ changes. To measure the slope error, we firstly identify the slopes at the different n sub-radii by the AE and compare against the predicted slopes. The slopes between two points:

$$a_i = \frac{y_i - y_{i-1}}{\theta_i - \theta_{i-1}}, i = 1, \ldots, n. \tag{17}$$

Therefore for n pairs we have $n-1$ slopes. The n slopes form the coefficients for our explanation functions represented by the n-dim. model parameter vector: $\mathbf{a} = [a_1, \ldots a_{n_1}]$. We adopt the cosine similarity for comparing the different $\mathbf{a}$ coefficient vectors as defined in (4).

KL-Divergence (Information Gain): To compute KL-Divergence, we use the n discrete y or z values to construct histograms that approximate the probability distributions for the two explanation functions f and $\hat{f}$, thus:

$$\mathrm{KL}(p||\hat{p}) = \sum_{i=1}^{n} p(y_i|\mathbf{x}, \theta) \log \frac{p(y_i|\mathbf{x}, \theta)}{\hat{p}(\hat{y}_i|\mathbf{x}, \theta)}, \tag{18}$$

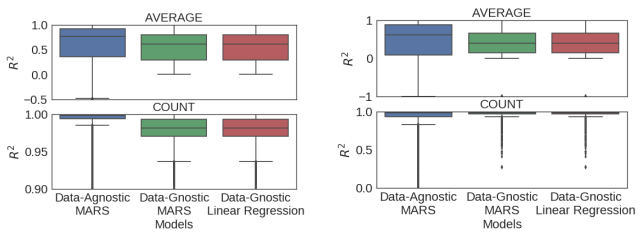
where $p(y_i|\mathbf{x}, \theta)$ and $\hat{p}(\hat{y}_i|, \mathbf{x}, \theta)$ refer to the probability density function for the actual and approximated explanation function, respectively. Then, we calculate the gain-ratio of *additional information* required when adopting our approximated models by $H(p) = -\sum_{i=1}^{n} p(y_i|\mathbf{x}, \theta) \log p(y_i|\mathbf{x}, \theta)$, which is the entropy of the actual distribution p derived from f, and then computing the gain-ratio $r = \frac{KL(p||\hat{p})}{H(p)}$.

Time Performance is measured using wall clock time in *milliseconds*. We compare the time required for LDAG and DG to provide an explanation; **note:** the time for DG and AE would be the same as the n pairs would also be needed.

6.2 Experimental Results

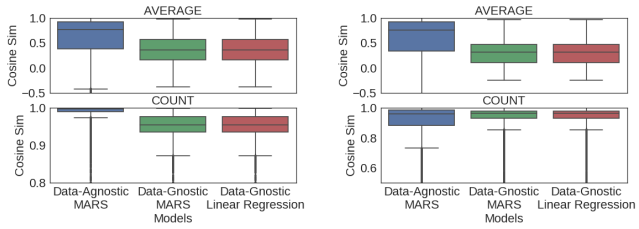
We use box-plot visualizations to show the results for the queries from set $\mathcal{V}$. Thus, we can observe outliers, the maximum and minimum values along with the median and Interquartile Range (IQR). Due to space limitations, we show only representative results – results with additional datasets and query workload types are given in the Appendix of the paper.

Figure 10 shows the results for explanation model fitting R^2 over the two workloads used (the other two workloads provide similar results and are included in the Appendix of the paper) using all three models described in the *Experimental Setup* with both `AVG` and `COUNT`. Overall, we observe high accuracy across all workloads and all models. It is worth noting that the accuracy for `COUNT` is higher than `AVG`, but this is expected as `AVG` fluctuates more than `COUNT`, the latter being monotonically increasing (note that the scales for R^2 are different for `COUNT` and `AVG`). We also conclude



(a) Gaussian $\mathbf{x}$ - Gaussian θ (b) Gaussian $\mathbf{x}$ - Uniform θ

Figure 10: Results for Coefficient of Determination.

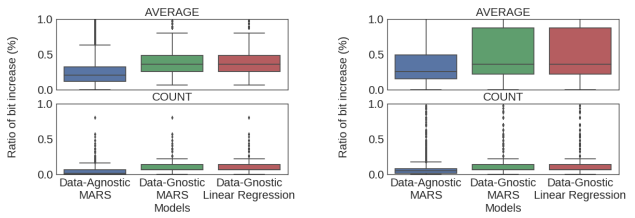


(a) Gaussian $\mathbf{x}$ - Gaussian θ (b) Gaussian $\mathbf{x}$ - Uniform θ

Figure 11: Results for Slope Error.

that is possible for the LDAG approach to have higher approximation accuracy than DG. This should not be surprising, as the DG models might not have all the necessary information to correctly approximate the explanation function as its main purpose is to minimize squared loss given the n pairs. Moreover, the values for R^2 indicate how well the proposed models fit the AE. As such, if the AE is highly non-linear the score for this metric deteriorates as seen in some of the outliers and minimum values. Also, it is important to note that if the AE is constant, meaning the rate of change is 0, then the score return is 0. We have encountered many such AEs especially for small radii in which no further change is detected.

In Figure 11, we observe LDAG behaving remarkably well to approximate the behavior of AE from model divergence perspective. The model is doing better for `COUNT` than for `AVG` as before. This means that LDAG can approximate the slope changes of an AE, thus, informing the analyst for such changes. DG for MARS has the same slope error behavior as for Linear Regression, which indicates a need for better adjustment of the hyper-parameter n. This can serve as evidence that LDAG using the alternating models as described in Section 5.3 can better approximate the behavior of an AE using previously executed queries in the same region. DG has less information for such tasks as this decision can only be made using n pairs. The box-plots in Figure 12 show that for both aggregate functions and all workloads the ratio of bit increase is well below 50%. This result indicates a measure of the inefficiency caused by using the approximated explanations, by DG and LDAG, instead of AE. Which, 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. Reviewing the results of all three evaluation metrics, it is evident that XAQ's LDAG approach competes with the DG approach even though *no* data access is needed and not



(a) Gaussian $\mathbf{x}$ - Gaussian θ (b) Gaussian $\mathbf{x}$ - Uniform θ

Figure 12: Results for KL-Divergence.

a single additional query was executed, thus, reducing the workload needed for such operations. It is able to learn a number of diverse explanations under many different workloads and data spaces, which makes XAQ a viable solution for data analysts in need of data exploratory analytics.

Time was measured in milliseconds. As expected, LDAG outperforms DG by at least two orders of magnitude as queries are not executed against the DBMS. We have also performed experiments showcasing the scalability of our methods measuring both the *Training* and *Pre-Processing* overheads. The results showed, both stages only take at most 2-3 minutes to execute, noting that the *Training* phase happens on-line. Also, we tested our approach with an ever increasing data set, noting that even though the DG approach has an exponentially degrading performance the LDAG approach remained constant. Figures for this section are omitted due to space limitations; therefore, LDAG is a highly accurate and scalable, which with minimal overhead serves as an explanation mechanism.

Additional Real Dataset: We also run the same set of experiments on an additional real dataset $\mathcal{B}_2 = \{\mathbf{x}\}$ where $\mathbf{x} \in \mathbb{R}^2$ and $|\mathcal{B}_2| = 5.6 \cdot 10^6$ containing records of calls issued from different locations for the city of Milano [2]. The obtained results for all metrics and variant workloads are omitted due to space constraints. However, they all follow the same pattern as the results given in our experiments using our first real dataset.

7. CONCLUSIONS

We have contributed a study for providing efficient, scalable, and accurate explanations for AQs. Given the importance of statistical analyses and AQs and the increasing importance of exploratory analytics, the contributed explanations can go a long way towards in-DBMS analytics and helping analysts to understand data subspaces and query results, and guiding them in their data explorations. Our study focused on explaining how the results of AQs depend on data-space characteristics. The explanations are succinct and rich, assuming the form of functions which allow analysts to investigate/experiment with different key parameters and observe their results (therefore appropriate for both exploratory and descriptive analytics). Schemes for computing explanations were presented, including LDAG, which does not require DBMS data accesses and can, thus, ensure efficiency and scalability. Novel ML models and algorithms were provided, which can compute approximate explanation functions, being trained by previous AQs and their associated results, while ensuring high explanation accuracy. Future work includes applying and extending our results for

other AQ types and dealing with on-line learning techniques to accommodate data and query pattern changes.

8. REFERENCES

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

[2] Telecommunications - sms, call, internet - mi. URL: https://dandelion.eu/datagems/SpazioDati/telecom-sms-call-internet-mi/, 2017. Accessed: 2016-12-01.

[3] Y. Amsterdamer, D. Deutch, and V. Tannen. Provenance for aggregate queries. In *Proceedings of the thirtieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 153–164. ACM, 2011.

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

[5] C. Anagnostopoulos and P. Triantafillou. Efficient scalable accurate regression queries in in-dbms analytics. In *Data Engineering (ICDE), 2017 IEEE 33rd International Conference on*, pages 559–570. IEEE, 2017.

[6] P. Bailis, E. Gan, S. Madden, D. Narayanan, K. Rong, and S. Suri. Macrobase: Analytic monitoring for the internet of things. *arXiv preprint arXiv:1603.00567*, 2016.

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

[8] O. Bousquet and L. Bottou. The tradeoffs of large scale learning. In *Advances in neural information processing systems*, pages 161–168, 2008.

[9] P. G. Brown. Overview of scidb: large scale array storage, processing and analysis. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 963–968. ACM, 2010.

[10] A. Chalamalla, I. F. Ilyas, M. Ouzzani, and P. Papotti. Descriptive and prescriptive data cleaning. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 445–456. ACM, 2014.

[11] S. Chaudhuri, G. Das, and V. Narasayya. Optimized stratified sampling for approximate query processing. *ACM Transactions on Database Systems (TODS)*, 32(2):9, 2007.

[12] S. Chaudhuri, G. Das, and U. Srivastava. Effective use of block-level sampling in statistics estimation. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 287–298. ACM, 2004.

[13] S. Chaudhuri and V. Narasayya. Autoadmin what-if index analysis utility. *ACM SIGMOD Record*, 27(2):367–378, 1998.

[14] J. Cheney, L. Chiticariu, W.-C. Tan, et al. Provenance in databases: Why, how, and where. *Foundations and Trends® in Databases*, 1(4):379–474, 2009.

[15] G. Cormode, M. Garofalakis, P. J. Haas, and C. Jermaine. Synopses for massive data: Samples, histograms, wavelets, sketches. *Foundations and Trends in Databases*, 4(1–3):1–294, 2012.

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

[17] N. Dalvi and D. Suciu. Answering queries from statistics and probabilistic views. In *Proceedings of the 31st international conference on Very large data bases*, pages 805–816. VLDB Endowment, 2005.

[18] K. El Gebaly, P. Agrawal, L. Golab, F. Korn, and D. Srivastava. Interpretable and informative explanations of outcomes. *Proceedings of the VLDB Endowment*, 8(1):61–72, 2014.

[19] J. H. Friedman. Multivariate adaptive regression splines. *The annals of statistics*, pages 1–67, 1991.

[20] A. Y. Halevy. Structures, semantics and statistics. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 4–6. VLDB Endowment, 2004.

[21] G. Hamerly and C. Elkan. Learning the k in k-means. In *Advances in neural information processing systems*, pages 281–288, 2004.

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

[23] J. M. Hellerstein, P. J. Haas, and H. J. Wang. Online aggregation. In *ACM SIGMOD Record*, volume 26, pages 171–182. ACM, 1997.

[24] B. Huang, S. Babu, and J. Yang. Cumulon: Optimizing statistical data analysis in the cloud. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pages 1–12. ACM, 2013.

[25] S. Idreos, O. Papaemmanouil, and S. Chaudhuri. Overview of data exploration techniques. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 277–281. ACM, 2015.

[26] S. Jain, D. Moritz, D. Halperin, B. Howe, and E. Lazowska. Sqlshare: Results from a multi-year sql-as-a-service experiment. In *Proceedings of the 2016 International Conference on Management of Data*, pages 281–293. ACM, 2016.

[27] M. I. Jordan et al. On statistics, computation and scalability. *Bernoulli*, 19(4):1378–1390, 2013.

[28] B. Kanagal, J. Li, and A. Deshpande. Sensitivity analysis and explanations for robust query evaluation in probabilistic databases. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, pages 841–852. ACM, 2011.

[29] N. Khoussainova, M. Balazinska, and D. Suciu. Perfxplain: debugging mapreduce job performance. *Proceedings of the VLDB Endowment*, 5(7):598–609, 2012.

[30] S. Krishnan and E. Wu. Palm: Machine learning explanations for iterative debugging. In *Proceedings of the 2nd Workshop on Human-In-the-Loop Data Analytics*, page 4. ACM, 2017.

[31] M. L. K. L. Sidirourgos and P. A. Boncz. Sciborq: Scientific data management with bounds on runtime

and quality. In *Proceedings of Conference on Innovative Data Systems, (CIDR)*, 2011.

[32] D. R. Legates and G. J. McCabe. Evaluating the use of goodness–of–fit measures in hydrologic and hydroclimatic model validation. *Water resources research*, 35(1):233–241, 1999.

[33] A. Meliou, S. Roy, and D. Suciu. Causality and explanations in databases. *Proceedings of the VLDB Endowment*, 7(13):1715–1716, 2014.

[34] H. Mühleisen and T. Lumley. Best of both worlds: Relational databases and statistics. In *Proceedings of the 25th International Conference on Scientific and Statistical Database Management*, page 32. ACM, 2013.

[35] V. Nair, A. Raul, S. Khanduja, V. Bahirwani, Q. Shao, S. Sellamanickam, S. Keerthi, S. Herbert, and S. Dhulipalla. 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*, pages 2029–2038. ACM, 2015.

[36] Y. Park, A. S. Tajik, M. Cafarella, and B. Mozafari. Database learning: Toward a database that becomes smarter every time. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 587–602. ACM, 2017.

[37] M. T. Ribeiro, S. Singh, and C. Guestrin. 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*, pages 1135–1144. ACM, 2016.

[38] S. Roy, L. Orr, and D. Suciu. Explaining query answers with explanation-ready databases. *Proceedings of the VLDB Endowment*, 9(4):348–359, 2015.

[39] S. Roy and D. Suciu. A formal approach to finding explanations for database queries. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 1579–1590. ACM, 2014.

[40] S. Sathe and K. Aberer. Affinity: Efficiently querying statistical measures on time-series data. In *Data Engineering (ICDE), 2013 IEEE 29th International Conference on*, pages 841–852. IEEE, 2013.

[41] M. Schleich, D. Olteanu, and R. Ciucanu. Learning linear regression models over factorized joins. In *Proceedings of the 2016 International Conference on Management of Data*, pages 3–18. ACM, 2016.

[42] M. Sundararajan, A. Taly, and Q. Yan. Axiomatic attribution for deep networks. *arXiv preprint arXiv:1703.01365*, 2017.

[43] D. Varrazzo. Psycopg postgresql adapter for the python programming language. Technical report, Technical report,[Available online at http://initd.org/psycopg/], 2010.

[44] X. Wang, X. L. Dong, and A. Meliou. Data x-ray: A diagnostic tool for data errors. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 1231–1245. ACM, 2015.

[45] X. Wang, A. Meliou, and E. Wu. Qfix: Diagnosing errors through query histories. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 1369–1384. ACM, 2017.

[46] A. Wasay, X. Wei, N. Dayan, and S. Idreos. Data canopy: Accelerating exploratory statistical analysis. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 557–572. ACM, 2017.

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

[48] E. Wu, S. Madden, and M. Stonebraker. Subzero: A fine-grained lineage system for scientific databases. In *Data Engineering (ICDE), 2013 IEEE 29th International Conference on*, pages 865–876. IEEE, 2013.

[49] S. Wu, B. C. Ooi, and K.-L. Tan. Continuous sampling for online aggregation over multiple queries. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 651–662. ACM, 2010.

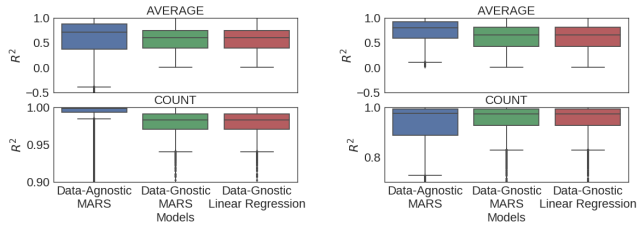
[50] Y. Zhu and D. Shasha. Statstream: Statistical monitoring of thousands of data streams in real time. In *Proceedings of the 28th international conference on Very Large Data Bases*, pages 358–369. VLDB Endowment, 2002.

APPENDIX

A. ADDITIONAL EXPERIMENTS

A.1 Experimental Results for additional synthetic workloads

A.1.1 Coefficient of Determination



(c) Uniform $\mathbf{x}$ - Gaussian θ (d) Uniform $\mathbf{x}$ - Uniform θ

Figure 13: Results for Coefficient of Determination metric

The figures exhibit the same properties and same high accuracy as mentioned in the Evaluation section.

A.1.2 Slope Error

Figure 14 exhibit the same properties and same high accuracy as mentioned in the Evaluation section.

A.1.3 KL-Divergence

Fig 15 exhibit the same properties and same high accuracy as mentioned in the Evaluation section.

A.2 Efficiency and Scalability

A.2.1 Increasing data affects DG approach only

An increase in data will drastically affect the DG approach as evident in figure 16. This is due to the fact that DG requires access to the underlying data to generate an explanation function $f(\theta; \mathbf{x})$. On the other hand, we see the LDAG

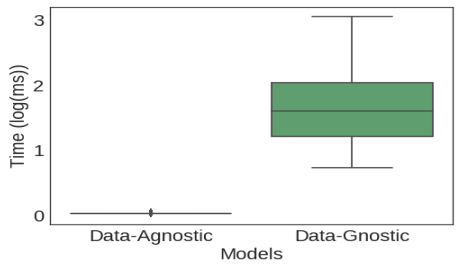
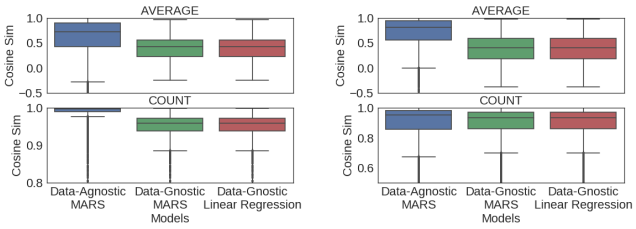


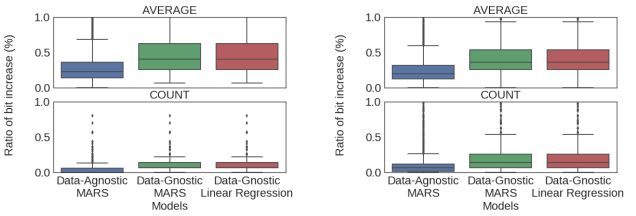
Figure 17: Performance comparison of LDAG and DG. Time in *log-scale* and in *milliseconds*



(c) Uniform $\mathbf{x}$ - Gaussian θ (d) Uniform $\mathbf{x}$ - Uniform θ

Figure 14: Results for Slope Error metric

.



(c) Uniform $\mathbf{x}$ - Gaussian θ (d) Uniform $\mathbf{x}$ - Uniform θ

Figure 15: Results for KL-Divergence

approach remaining invariant to any changes in data. The dataset R was artificially increased by scaling up the number of data points in $\mathcal{B}$ respecting the original distribution of data points.

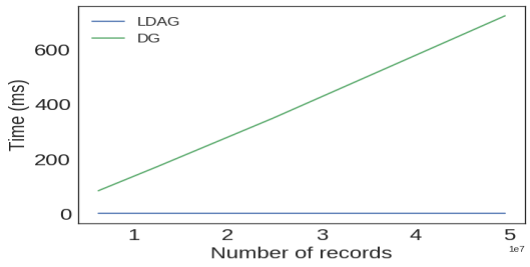


Figure 16: LDAG has stable performance, on the contrary, DG's performance degrades with an increase in data. The x-axis denotes the increase in records/data and the y-axis is the time required to retrieve an explanation measured in milliseconds.

A.2.2 Performance Figure

As discussed in the Experimental Evaluation section a DAG/LDAG approach outperforms a DG approach by orders of magnitude. Thus XAQ is guaranteed to provide the user accurate AQ explanations in lightning fast speed compared to getting an explanation manually or through the issuing multiple new sub-queries. This is evidenced in Figure 17.

A.2.3 Scalability of XAQ

We provide empirical evidence of the scalability of our solution. For an increased workload we see that our framework scales linearly with some minor fluctuations attributed to the inner workings of the system and different libraries used. Evidently, both *Pre-Processing* stage and *Training* stage do not require more than a few minutes to complete. We note that *Training* stage happens on-line, however we test the stage off-line to demonstrate its scalability. The results are shown in Figure 18.

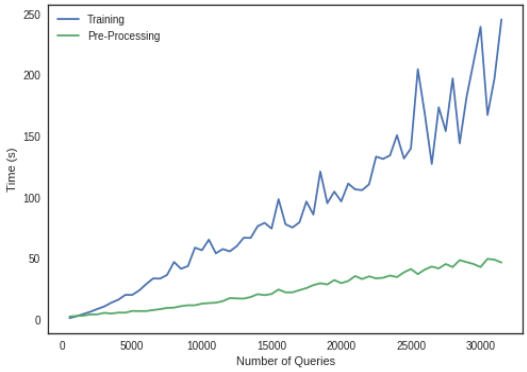


Figure 18: Scalability of *Pre-Processing* and *Training* stages.

A.3 Additional Real Dataset

We also run one set of experiments on an additional real dataset $\mathcal{B}_2 = \{\mathbf{x}\}$ where $\mathbf{x} \in \mathbb{R}^2$ and $|\mathcal{B}_2| = 5.6 \cdot 10^6$ containing records of calls issued from different locations for the city of Milano [2]. We provide the results for workload (*Gauss-Gauss*) on both `Average` and `COUNT` to demonstrate the extensibility of our solution to other datasets and different workloads. The *Data Preparation* and *Experimental Setup* are the same as in our *Experimental Evaluation* section. Lastly, we only report on one metric R^2 due to space restrictions. Figure 19 demonstrates the results of this experiment. As evident from the *Experimental Evaluation* section of our paper, our approach has high accuracy especially for `COUNT`. The obtained results for the rest of the metrics and variant workloads are omitted due to space constraints. However, they all follow the same pattern as the results given in our experiments.

B. ADDITIONAL METRICS

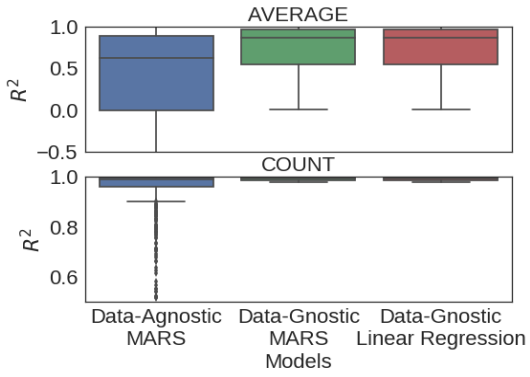


Figure 19: Coefficient of Determination for R_2

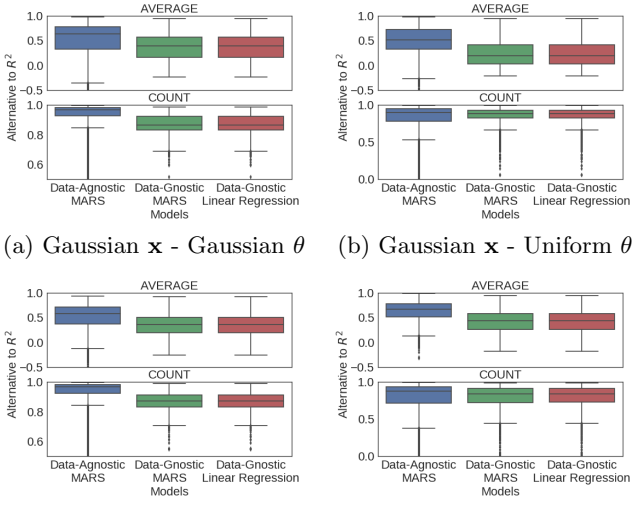
B.1 Alternative to Coefficient of Determination

It has been noted that R^2 is susceptible to outliers [32]. Therefore we also use an alternative to R^2 suggested in Legates *et al.* which uses the absolute value instead of the squared value. It has the following form

$$R^2 = 1 - \frac{|y - \hat{y}|}{|y - \overline{y}|} \tag{19}$$

and it used to get a more accurate estimate for R^2 in cases where outliers have huge impact on its value.

B.1.1 Result for Alternative to R^2



(a) Gaussian $\mathbf{x}$ - Gaussian θ (b) Gaussian $\mathbf{x}$ - Uniform θ

(c) Uniform $\mathbf{x}$ - Gaussian θ (d) Uniform $\mathbf{x}$ - Uniform θ

Figure 20: Results for Alternative to Coefficient of Determination metric

Examining the generated boxplots, figure 20, for the Alternative to R^2 and comparing with the ones for R^2 we see that the results slight decreased across all models, aggregation operators and datasets. This is due to the fact that R^2 tends to inflate the results.

B.1.1.1 Main point.

Results observed are different than R^2 but this is expected and the behavior of the models across both aggregation operators and datasets is the same.

C. PROOFS OF THEOREMS

C.1 Proof of Theorem 1

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 $\mathcal{E}(u) = \mu|\theta - u| + (1-\mu)(y - \hat{f}(\theta; \mathbf{w}))^2$, using SGD over $\mathcal{E}(u)$. Given the t-th training pair $(\mathbf{q}(t), y(t))$, the stochastic sample $E(t)$ of $\mathcal{E}(u)$ has to decrease at each new pair at t by descending in the direction of its negative gradient with respect to $u(t)$. Hence, the update rule for L2 RR u is derived by:

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

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 $\Delta u \leftarrow \alpha\mu\text{sgn}(\theta - u)$. By starting with arbitrary initial training pair $(\mathbf{q}(0), y(0))$, the sequence $\{u(t)\}$ converges to optimal L2 RR u parameters.

C.2 Proof of Theorem 2

Consider the optimal update rule in (12) 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:

$$\begin{aligned}\mathbb{E}[\Delta u] &= \int_{\mathbb{R}} \text{sgn}(\theta - u)p(\theta)\text{d}\theta = P(\theta \geq u)\int_{\mathbb{R}} p(\theta)\text{d}\theta \\ &- P(\theta < u)\int_{\mathbb{R}} p(\theta)\text{d}\theta = 2P(\theta \geq u) - 1.\end{aligned}$$

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.

D. NEAR OPTIMAL K FOR K-MEANS

We provide an algorithm for near-optimal k-means we note that there are multiple such algorithms available in the literature [21]. However, this is not part of our focused work thus a simple solution was preferred to alleviate such problem. In addition, our work can be utilized with any kind of Vector-Quantization algorithm thus k-means was chosen because of its simplicity and wide use.

E. NOMENCLATURE

Algorithm 1 Estimating a near-optimal K

Input: ϵ; 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

Notation	Explanation
d	dimensions
$\mathbf{q}$	query
$\mathbf{x}$	location of query
θ	radius of query
$\mathcal{B}$	dataset consisting of N d-dim. vectors
$\mathcal{T}$	set of $(d+1)$-dim. queries (Workload (WL))
$\mathcal{V}$	set of $(d+1)$-dim. queries for Eval.(WL)
$\mathbb{D}(\mathbf{x}, \theta) \in \mathbb{R}^d$	data subspace
$\mathbb{Q} \in \mathbb{R}^{d+1}$	Query Vectorial Space
y	`COUNT` query result
z	`AVERAGE` query result
$\mathcal{F}$	Function space
u	sub-cluster L2 RR
$\mathbf{w} \in \mathbb{R}^d$	cluster L1 LR
N	number of d-dimensional data points
L	number of sub-clusters
K	number of clusters
λ	hinge point
μ	adjustable parameter for (11)