A Content-Based Approach for Modeling Analytics Operators

Ioannis Giannakopoulos
Computing Systems Laboratory, School of ECE, National Technical University of Athens, Greece
ggian@cslab.ece.ntua.gr

Dimitrios Tsoumakos
Department of Informatics, Ionian University, Corfu, Greece
dtsouma@ionio.gr

Nectarios Koziris
Computing Systems Laboratory, School of ECE, National Technical University of Athens, Greece
nkoziris@cslab.ece.ntua.gr

ABSTRACT

The plethora of publicly available data sources has given birth to a wealth of new needs and opportunities. The ever increasing amount of data has shifted the analysts' attention from optimizing the operators for specific business cases, to focusing on datasets per se, selecting the ones that are most suitable for specific operators, i.e., they make an operator produce a specific output. Yet, predicting the output of a given operator executed for different input datasets is not an easy task: It entails executing the operator for all of them, something that requires excessive computational power and time. To tackle this challenge, we propose a novel dataset profiling methodology that infers an operator's outcome based on examining the similarity of the available input datasets in specific attributes. Our methodology quantifies dataset similarities and projects them into a low-dimensional space. The operator is then executed for a mere subset of the available datasets and its output for the rest of them is approximated using Neural Networks trained using this space as input. Our experimental evaluation thoroughly examines the performance of our scheme using both synthetic and real-world datasets, indicating that the suggested approach is capable of predicting an operator's output with high accuracy. Moreover, it massively accelerates operator profiling in comparison to approaches that require an exhaustive operator execution, rendering our work ideal for cases where a multitude of operators need to be executed to a set of given datasets.

ACM Reference Format:
Ioannis Giannakopoulos, Dimitrios Tsoumakos, and Nectarios Koziris. 2018. A Content-Based Approach for Modeling Analytics Operators. In *The 27th ACM International Conference on Information and Knowledge Management (CIKM '18), October 22–26, 2018, Torino, Italy*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3269206.3271731

1 INTRODUCTION

As big data technologies evolve, emphasis steadily expands to areas not solely related to size. Undeniably, data volume has been the decisive driving force behind big data technologies; in many cases, the effectiveness of an algorithm relies entirely on the amount of data it can access [37], leading to continuous efforts on scalability and optimization. As datasets become increasingly abundant over heterogeneous sources and the requirement to fuse them is pressing, distinct datasets and databases can no longer be viewed as isolated components: Different datasets present inter-dependencies despite their contrast in size, data-type, origin, schema, etc. Combining data sources increases the utility of existing datasets, generating new information and creating services of higher quality [34].

A different type of challenge shifts attention to the actual content: Content-based analytics processes data from social media platforms for sense-making and knowledge generation. Similarly, data content plays a key role in the quality of the insights derived in applications such as recommendation systems, web advertising and marketing, fraud detection, etc. In these cases, analysts increasingly need to focus on "high-impact" data, i.e., intelligence that has the best potential of driving strategic decisions. In the same manner, many data scientists have recently made a case about *medium* data analytics [1, 6], where the utility rather than the size of the data is considered to be the critical factor.

The plethora of available sources and datasets that can be fed to such content-sensitive services now creates an issue: Data scientists need to decide which of the available datasets should be applied for any given workflow independently. Yet, as modern analytics workflows have evolved into increasingly long and complex series of diverse operators, evaluating the utility of immense numbers of inputs can be a daunting task. Even if modern runtime environments (e.g., [4, 45]) and the Cloud paradigm [12] provide the necessary tools to speedup the evaluation, the problem is particularly challenging because of (a) the blowup in the number of available datasets, (b) the ever-increasing type and number of complex operators that need to be executed and (c) the time constraints that latency-sensitive operators pose.

For example, in the case of derivative pricing theory [26], analysts need to consider a multitude of Credit Default Swaps (CDS) time series for different economic entities. These are provided as input to mathematically complex operators so that financial indicators, i.e., Value Adjustments (xVA), are extracted that quantify the credit, funding and financial costs an institution faces during derivative transactions. Selecting the appropriate CDS datasets for extracting the respective xVAs for an entity is of key importance for the indicator's accuracy. Thus, an analyst needs to consider all CDS datasets and execute the aforementioned operators for all of them in order to select the ones that present certain characteristics that make them more suitable for specific entities and maximize accuracy. In a different domain, Security Information and Event Management (SIEM) systems [40], commonly deployed to identify and mitigate cyberattacks, increasingly fail to identify and stop advanced persistent attacks because of their inability to cope with the increasing amount of available datasets utilized to train them [41]. These systems lack the analytics capabilities to process a vast

CIKM '18, October 22–26, 2018, Torino, Italy

ACM ISBN 978-1-4503-6014-2/18/10…$15.00
https://doi.org/10.1145/3269206.3271731

amount of data and, hence, their administrator needs to make a key decision: Which datasets out of the available ones should be used in order to train such a system to stop cyberattacks of a certain type in tight time constraints? In both cases the *data operators* have access to multiple datasets (CDS time series and SIEM training datasets for the two cases respectively), however their outputs entirely rely on the selection of a mere subset of them based on a set of properties that are neither known nor easily identified. For example, one cannot decide a priori which CDS datasets are more suitable for the xVA estimation for a given economic entity, even if experience or prior knowledge is available [20]. Similarly, it is impossible to select the most appropriate SIEM training datasets (especially under stringent time constraints) be used for training systems for intrusion detection without actually training the system in order to evaluate its output.

To facilitate dataset analysis, two complementary directions have been suggested: Data Integration and Data Exploration. Data Integration approaches (e.g., [28, 39]) aim at presenting a unified view of distinct datasets and focus on the systemic problem of fusing data from heterogeneous sources. On the other hand, Data Exploration approaches (e.g., [22, 44]) aim at producing dataset summaries in order to inform the users about properties of the data, such as tuples that encapsulate the most representative data patterns, dependencies between tuple fields, statistical properties, etc. Although both directions can be used to provide valuable insights for unknown datasets, neither of them is designed to model an operator's output when executed for each of the available data sources. Instead, their scope is mostly limited to summarize the datasets and extract representative patterns for them without considering how these datasets affect certain operators.

In order to bridge the gap between input datasets and the output of many popular data operators, in this work we propose an operator-agnostic dataset profiling mechanism. Rather that analyzing or executing over each dataset separately, our work assesses the similarity between the available datasets in light of data characteristics that highly impact operator behavior, i.e., the data *distribution*, the dataset *size* and the *ordering* of the data points. Based on the dataset relationships, we infer knowledge about them. Each dataset is then projected to a point in a low-dimensional metric space that reflects the dynamics between them: Dataset similarity is analogous to the distance among the respective points. Using this metric space, datasets are sampled and a specific operator is applied to each of the samples. Our framework models operator output using Neural Networks [27], allowing it to *predict* its performance over all available datasets. One of the most interesting aspects of the proposed methodology is that the computationally intensive part of our work, i.e., the construction of the dataset metric space, remains entirely *operator-agnostic*. The same constructed space can be utilized for different operators without modifications. In order to consider new operators, one should only repeat the operator execution for a small number of datasets in order to obtain a training set, eschewing the computationally intensive part of reconstructing the space from scratch. Our work shifts its effort towards measuring the dataset inter-relationships and constructing the space that best reflects them (which is an *offline* process), with the intention of minimizing the time and cost of *online* modeling specific operators. The contributions of our work are summarized as follows:

- We present an efficient, operator-agnostic dataset profiling methodology, that estimates the similarity among the available datasets, constructing a *dataset space* that best reflects their properties and modeling the output of the applied operators utilizing Neural Networks trained using this dataset space as input.
- We offer an open source Go prototype [8] of our work, through which a user can execute the dataset profiling and export the generated ML models for integration with other systems.
- We conduct a thorough evaluation of this approach, utilizing a variety of datasets with different characteristics.

Our evaluation indicates that our methodology models the behavior of a wide variety of operators with remarkable accuracy (less than 2% of modeling error in the best case and less than 15% in most cases when a mere 4% of the datasets are examined). Moreover, it presents massive speedups (more than 20× in the best case) in comparison to exhaustively executing the operators for the entirety of available datasets. Finally, it can be customized in order to accelerate data analysis and conduct less detailed dataset examination or increase modeling accuracy when higher execution time is affordable.

2 PRELIMINARIES

2.1 Problem Description

Assume a set of datasets $D = \{D_1, D_2, \cdots, D_N\}$ and an operator F. Each dataset $D_i, 1 \leq i \leq N$, consists of tuples with the same number of columns, containing arithmetic values. F accepts a single dataset as input and produces a scalar output value:

$$F : D \rightarrow \mathbb{R} \tag{1}$$

Each operator can be viewed as a function that projects any dataset D_i to a scalar value $F(D_i)$. The problem that this work addresses, is the following: *We seek for an approximation of the expression $F(D_i)$ without exhaustively executing F for all datasets.* This resembles a typical function approximation problem: One can sample D, execute F for the selected subset of datasets and utilize regression to approximate F for the rest of the datasets. However, this method cannot be applied in this problem because D represents an unordered set of datasets that do not belong to a *metric space* and the relationships between them are unknown. Since all function approximation approaches require D to be a metric space, i.e., the distances between datasets to be known, regression cannot be used.

Albeit constructing a metric space for any given D is possible for a given distance function for each dataset pair in D, the quality of the approximation is heavily affected by the choice of this function. Ideally, the desired distance function must reflect the distance between two datasets D_i, D_j, $1 \leq i, j \leq N$, both in the aforementioned metric space and in the operator's output domain, i.e., if $|D_i - D_j| < \epsilon$ ($|.|$ denoting the Euclidean norm) then $|F(D_i) - F(D_j)| < \epsilon$ as well. If this property applies, the constructed metric space can be accurately used by a model in order to *predict* an operator's output, since the topology of the datasets themselves provide excellent hints on the behavior of the latter. Although this may initially seem an operator-dependent procedure, we argue that only a handful of distance functions that examine specific dataset properties suffice to generate highly informative *dataset spaces*, that facilitate modeling the behavior of diverse real-world operators.

2.2 Operators and Dataset Properties

Associating how a property affects an operator's behavior generally requires extensive knowledge regarding the operator's design. Nevertheless, we argue that there exist some *fundamental* properties that, if examined, they can produce invaluable insight regarding an operator's behavior. Indeed, examining data interrelationships in the light of a handful of fundamental dataset properties can generate a knowledge basis through which the behavior of different operators can not only be explained but also predicted. These properties examined in this work are: (a) the statistic al *distribution* of the datasets, (b) the dataset *size* and (c) the *order* of their tuples. *Distribution* refers to the positioning of a dataset's tuples and it is a fundamental property that is implicitly or explicitly examined during any data analysis task, as it uniquely characterizes the statistical behavior of a dataset. *Size* is an expression of the cardinality of the dataset and it is commonly examined when the behavior of an operator is affected by it. *Order* expresses the ranking of the dataset's tuples and is frequently examined when sequence matters.

Table 1: Operators and Dataset Properties

Operators		Affected by
Class	**Name**	
Aggregate Functions	AVG	Distribution
	SUM COUNT	Distribution + Size
Density	DBSCAN [23] Local Outlier Factor [18]	Distribution
ML	Linear Regression	Distribution
Spectrum	Eigenvalue Estimation	Distribution
Time-Series Forecast	Holt-Winters [19] ARIMA [17]	Distribution + Order

These three fundamental properties highly affect a magnitude of real-world operators. In order to showcase this, in this work, we use popular operators from diverse domains, which are frequently encountered either as isolated components or as part of greater and more complex analytics workflows. The considered operators along with the respective properties they are affected by are summarized in Table 1. Each operator is formulated as per Equation 1. Specifically, the *Aggregate Functions* represent mathematical operations that are applied to one or more dataset columns, producing a scalar value. The *Density* class comprise *DBSCAN* [23] that executes the popular clustering algorithm and return the number of formed clusters and *Local Outlier Factor* (referred to as *LOF*) that executes the algorithm presented in [18] and returns the percentage of tuples that collect a score greater than 2 and are, thus, considered to be outliers. The *ML* class consists of a fundamental Machine Learning operator, i.e., Linear Regression. This operator is trained using a dataset from D and return an estimate of the training error when the model is tested with a given external dataset. The *Spectrum* class contains an operator that returns the i-th eigenvalue (for a given i) of the provided dataset, a procedure executed in various workflows from dimensionality reduction [30] to clustering. Finally, the *Time-Series Forecast* class comprises two operators [17, 19] that forecast the i-th value of a provided Time-Series dataset.

The reason behind the choice of examining these operators is threefold. First, all of these operators are *popular* and extensively utilized in Data Science and Machine Learning applications, either as part of data preprocessing (e.g., outlier detection, statistical analysis, etc.) or core learning workflows (e.g., supervised/unsupervised learning). Second, their *diverse* characteristics enforce us to design a generic solution that makes no assumptions regarding their internals. Third, Machine Learning workflows, parts of which are frequently constructed with the above operators, are an *excellent domain* for the problem this work addresses. As data scientists need to be able to classify an increasing number of datasets [15] without actually executing their complex workflows to them, the identification of the *Right Data* [14], i.e., data of high utility which are essential for driving strategic decisions, is crucial. Finally, it should not be overlooked that Table 1 does not contain a complete list of operators. Although our work uses the mentioned operators for evaluation, our methodology can be used for any operator affected by the three mentioned data properties without modifications.

3 METHODOLOGY

3.1 Methodology Overview

The key observation that datasets with similar specific properties impact certain operators in similar ways and, hence, make them producing similar outputs, highlights a new dimension to the problem under investigation: If one quantifies the similarity between all pairs of datasets and executes an operator for only a handful of them, a first idea of F's domain would become available, as datasets with high similarity would present similar behavior. Let us generalize this idea: Given the relationship between dataset similarity and an operator's output, we seek for a projection of the datasets in D into a metric space D' (also referred to as *dataset space*) that best reflects the resemblance among them. D' can be then utilized by F as the domain space – according to Equation (1) – in order to project the original datasets into the anticipated values. Interestingly so, the relationships between datasets are *independent* of F, allowing different operators to be applied to a unique D'. For each operator, one could sample D, estimate F's values for the selected datasets $D_i \in D_s \subseteq D$ and approximate F for the rest of the datasets utilizing Machine Learning (ML) techniques. Although F is applied to some of the original datasets, i.e., $F(D_i), D_i \in Ds$ is calculated, the ML model is trained using D' as the input space and the approximated operator F' is defined as: $F' : D' \rightarrow \mathbb{R}$. Essentially, D' comprises a set of *features* that best characterize the datasets' interrelationships. Figure 1 depicts an overview of the suggested methodology.

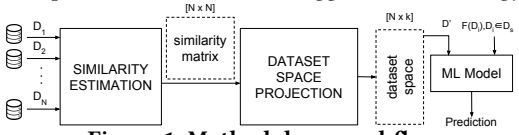


Figure 1: Methodology workflow

The *Similarity Estimation* module quantifies the similarities between datasets $D_1, \cdots, D_N$. The outcome of this process is a symmetrical $N \times N$ similarity matrix whose (i,j) cell represents the similarity between D_i and D_j. The similarity matrix is then accessed by the *Dataset Space Projection* module which transforms the original similarities into a metric space. In this step, Multidimensional Scaling (MDS) [25] transforms the similarity matrix into a set of points in a low-dimensional (k-dimensional) space, with the property that the distances between the points of the space approximate the similarity represented by the original matrix. The final outcome of the process is a $N \times k$ matrix that represents the coordinates of each dataset in the dataset space. Finally, an operator F

can be executed for a small subset of datasets D_s. Using the dataset coordinates and the respective operator values, a Neural Network is trained in order to approximate F for all datasets. Based on the approximated dataset scores, interesting questions can be answered: Which are the dataset(s) with the highest/lowest F values (e.g., with the highest first eigenvalue), how many datasets' output is around a given F value (e.g., dataset with approximately 5 DBSCAN clusters), retrieve the top-k datasets under certain output criteria (e.g., the top-10 datasets with highest percentage of outliers), etc.

Essentially, our approach attempts to shift the computational burden in the first phase of data analysis: The workflow presented in Figure 1 is executed once in an *offline* manner for all datasets. This offline part is entirely *operator-agnostic*: The similarity estimation does not imply the execution of any operator and only the relationships between the raw datasets are evaluated without considering the type of the operator that may be applied to them. Whenever a new operator emerges, it is executed for a mere subset of the available datasets and its behavior is rapidly approximated with minimal computation. In the long run, the overhead introduced by the suggested approach is amortized and the avoided computation linearly increases with the number of operators that need to be executed for the analyzed datasets.

3.2 Similarity Estimation

The notion of similarity adopted in this work focuses on three characteristics, namely the *distributions* of the datasets, their *size* and tuple *ordering*. Based on these primitive properties, one can compose arbitrary similarity expressions that efficiently express multiple dataset aspects. The similarity between two datasets is a real number in the interval $[0, 1]$, 0 indicating total dissimilarity and 1 indicating perfect similarity. We now examine the mechanism that quantifies the similarity for each property in detail.

▹**Distribution:** There exist several methods used to quantify the similarity of the distributions between two datasets. One of the most popular methods used during data analysis, entails the identification of the distribution in a closed form for each dataset separately and the comparison of the probability density functions (pdf) in order to extract the relationships between them. The extraction of a pdf usually occurs in a trial-and-error manner, as different distribution types are tested for each dataset, keeping the one that maximizes a statistical fitness measure such as the p-value. Although this analysis can provide great insight in the behavior of a statistical sample, it also presents certain limitations. First, the size of the sample plays a determinant role, as smaller datasets may not present enough evidence to accurately identify the most suitable pdf. Second, distribution-free datasets may be erroneously represented. Third, the number of tested distributions needs to be high in order to provide accurate data representations, increasing the complexity of the computations. Yet for our problem, we are not interested in the identification of each dataset's pdf per se, but we only want to estimate the extent at which the tuples of two datasets overlap, i.e., their relative positions in the space. If two datasets feature a large percentage of their tuples inside the same regions, then they should have similar distributions.

Quantifying the overlap between two different statistical samples (datasets) is the main idea behind the Bhattacharyya coefficient (BC) [21]. Its estimation relies on partitioning two datasets into l disjoint partitions, identifying the number of tuples that belong to each of them and, finally, summarizing the root of the product of the number of tuples for each region: $BC = \sum_{i=1}^{l} \sqrt{A_i B_i}$, where A_i and B_i denote the number of tuples located in the i-th partition for datasets A and B respectively. For two given datasets A and B, the upper bound of the BC value is obtained when $l = 1$. In order to compare BC values between different pairs of datasets that might enumerate different tuples, we normalize BC with this upper bound, reaching the following Distribution similarity function:

$$Distribution(A, B) = \frac{\sum_{i=1}^{l} \sqrt{A_i B_i}}{\sqrt{|A||B|}} \tag{2}$$

One parameter that highly affects Equation 2 is the partitioning setup. Specifically, both the partitioning algorithm and the number of partitions affect the equation's behavior, in different ways. The choice of the partitioning algorithm is crucial for ensuring fairness among different datasets, since certain partitioning schemes may boost specific distributions and be unfair to others. It also needs to be scalable to multiple dimensions, in order to successfully consider datasets of high dimensionality. Voronoi partitioning [16] is a popular partitioning setup that adheres to these properties. Each partition (or *cell*) is represented by a centroid (seed). Each dataset tuple is assigned to the partition represented by its closest centroid. The partition problem is, thus, transformed to finding a set of seeds. Albeit the estimation of the optimal seeds is NP-Hard, one can generate them using k-means [36]. Since k-means' solutions are largely affected by the initial seed selection, in our work k-means++ [13] was utilized. In order to maximize fairness between datasets and minimize any bias inserted by skewed distributions during the centroid selection, we sample all the available datasets, keeping a small percentage ($1 - 5\%$) of each and generate a new one that consists of all the sampled tuples. k-means is, then, executed for this "merged" dataset, centroids are extracted and then used for partitioning each dataset separately. The number of partitions (l) is determined by the user. More partitions lead to a more fine-grained examination whereas fewer partitions provide greater abstraction.

From a technical perspective, after the execution of k-means, the estimated Voronoi diagram is utilized to partition all the available datasets and, subsequently, the number of the tuples inside each leaf is estimated. For each dataset, an in-memory array is generated: The index of each array element represents the Voronoi cell id and the value of the array element represents the number of tuples that reside in the cell. The estimation of Equation 2 is then reduced to obtaining the dot product of the respective arrays for A and B. Repeating the procedure for every dataset pair provides the final similarity matrix. Note that, the construction times of the arrays, which entails the processing of the raw data, increases linearly with the number of datasets. Essentially, these arrays act as synopses of the datasets: The more the partitions, the more accurate the representation and the more time-consuming is the estimation of the dot product between each pair. Specifically, the complexity of constructing a similarity matrix using Equation 2 equals $O(Nnld' + Nnl + N^2 l)$, where N is the number of datasets, n is the maximum dataset size, l is the number of partitions and d' is the dimensionality of the domain of the datasets. The first factor refers to the Voronoi diagram creation, the second one refers to the synopsis creation for all datasets and the third refers to the matrix.

▹**Order:** The identification of the similarity between the ordering of two datasets entails the estimation of the rank correlation coefficient [33], i.e., the measurement of the ranking between their members. First, a sorted copy of the datasets is created: The columns that are utilized for the sorting step are defined by the user, along with their importance in the comparisons (by default all columns are utilized in increasing column-id order), i.e., if $X[1] > Y[1]$ then $X > Y$. Based on the respective sorted copy for each dataset, the *rank* array is generated. The i-th element of the rank array represents the position of the i-th element in the sorted copy, e.g., if the original array contains the elements $X = [100, 99, 102]$ the respective rank array is $x = [2, 1, 3]$. Given that, we provide the rank similarity measure utilized in this work, that resembles the Kendall rank correlation coefficient (τ) [31]:

$$Order(A, B) = \frac{concord(a, b) - discord(a, b)}{n(n-1)} + \frac{1}{2} \tag{3}$$

in which, a and b represent the rank arrays of A and B respectively, $concord(a, b)$ returns the number of pairs (a_i, b_i) and (a_j, b_j), $i \neq j$ for which the rank of both elements agree, i.e., if $a_i > a_j$ then $b_i > b_j$ or if $a_i < a_j$ then $b_i < b_j$. $discord(a, b)$ returns the respective number of pairs whose rank disagree. When two pairs share the same rank they are considered neither concordant nor discordant. Note that the Kendall τ receives values in the interval $[-1, 1]$, in which 1 means that the two ranks completely match and -1 means that the two datasets are sorted in reverse order. Since, in our case, the similarity metric is expressed in the interval $[0, 1]$, we scale the Kendall τ accordingly, producing the above expression. The complexity of constructing a similarity matrix using Equation 3 equals $O(Nn\log(n) + N^2 n\log(n))$. The first factor relates to the sorting step of all the datasets and the second factor relates to the estimation of $Order$ for each pair of datasets. Note that, although $Order$ seems to require the investigation of every pair of tuples of the two datasets (quadratic time to the dataset size, i.e., $O(n^2)$), a merge-sort based implementation requires $O(n \log n)$ steps [32].

▹**Size:** The similarity between the size of two datasets is evaluated, using the following expression:

$$Size(A, B) = \frac{\min(|A|, |B|)}{\max(|A|, |B|)} \tag{4}$$

Note that, $Size(A, B) \to 1$, if $|A| \to |B|$. The complexity of creating a similarity matrix with Equation (4) is $O(Nn + N^2)$: The first factor represents the cost of counting each dataset's tuples and the second factor represents the evaluation of $Size$ for each pair of datasets.

▹**Combining different metrics:** While certain operators may depend on a single property, there exist cases one would want to construct a similarity matrix that combines several ones. In such cases, one can generate algebraic combinations of simpler matrices that reflect more sophisticated similarity expressions. The form of the algebraic expression (e.g., linear combination, matrix multiplication, etc.) that combines the matrices is defined by the user and expresses the importance of each component.

3.3 Dataset Space Projection

Although the information of a similarity matrix can be directly used by a data engineer, this representation of information is cumbersome for three reasons: (a) The *size* of the matrix increases quadratically with the number of datasets. (b) The matrix provides limited information as it does not represent the relationships at scale, e.g., one can easily identify a dissimilar – to the rest – dataset, but the magnitude of the dissimilarity is not easily comprehensible. (c) Most Machine Learning algorithms require the input metric space to be expressed in a form where the *coordinates* rather than the similarities of the input space points are known. Although a category of nonparametric learning algorithms [10] do support regression based similarities, these algorithms are less sophisticated and are mainly used for simpler learning tasks. In order to address these limitations, our methodology transforms the similarity matrix into a dataset space where the positions of the datasets reflect their similarities: Datasets that are placed closer to each other would be more similar than the more distant ones.

Toward this direction, Multidimensional Scaling (*MDS*) [25] has been used. MDS is a technique used to estimate the coordinates of a set of points given a square matrix that quantifies the dissimilarity between them. According to Classical MDS, the dissimilarity expresses the Euclidean distance between the points, although, in the general case, this dissimilarity can be expressed using any distance function. MDS can also work based on matrices that represent similarity. For clarity and in order to remain aligned to the literature, we will describe MDS based on dissimilarity matrices. Either way, similarity is easily transformed into distance using the following transformation (s being the similarity and d the distance):

$$s = \sqrt{1 - d}, d = 1 - s^2 \tag{5}$$

where both $s, d \in [0, 1]$. Note that when $s \to 1$, $d \to 0$ and when $s \to 0$, $d \to 1$. Any pair of transformation equations that respect this property can be used instead, assuming that d is finite.

There exist several methodologies that execute MDS in order to calculate a coordinates matrix. In this work, we utilize *Classical MDS* [25] which expresses the problem as a matrix eigendecomposition problem. In short, MDS entails the execution of eigenanalysis to the dissimilarity matrix of size $N \times N$, producing a list of eigenvalues $[\lambda_1, \lambda_2, \cdots, \lambda_N]$ in descending order and their respective eigenvectors. Each eigenvector represents a dimension of the metric space and its respective eigenvalue represents the variance it captures. The number of eigenvectors to utilize is decided based on the covered variance, expressed by the *Goodness of Fit* (GoF):

$$GoF = \frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{N} \lambda_i} \tag{6}$$

k being the number of eigenvectors to use. A common rule-of-thumb is to set k to a value where $GoF \geq 0.75$ [30].

Although MDS achieves to identify the dimensionality of the final space and the initial dataset positions to it, the difference between the distances obtained by the space and the matrix can be further reduced through a non-linear projection. To this end, after the execution of MDS, we apply Sammon mapping [42], a non-linear space transformation that aims at slightly "moving" the projected datasets in such a manner that their projected distances best approximate the original dissimilarity matrix. Specifically, all the datasets are initially assigned with the coordinates produced through MDS. The distance between the projected and the target distances is expressed by the *Sammon Stress* E_s:

$$E_s = \frac{1}{\sum_{i<j} d_{ij}} \sum_{i<j} \frac{(d_{ij} - d^*_{ij})^2}{d_{ij}} \tag{7}$$

d_{ij} denoting the distance between the i-th and j-th element from the distance matrix and d^*_{ij} the respective distance as measured by the produced space. The execution of Sammon mapping entails

the use of an iterative optimization methodology such as *Simulated Annealing* (SA) [43], until the E_s value stops declining or a predefined iteration threshold is exceeded. This reduction enables the constructed space to better represent the information of the similarity matrix, without increasing its dimensionality.

3.4 Modeling

After constructing the dataset space, Neural Networks (NNs) are used for predicting a given operator's outputs. A sample $D_s \subseteq D$ of datasets is obtained, the size of which is determined by the user. An operator F is applied to every dataset in D_s and the output values are provided as a training set to a NN. The trained model is, subsequently, tested using a – disjoint from D_s – test set. NNs were preferred against other Machine Learning methodologies due to their efficiency and their ability to model arbitrary distributions. The employed models comprise 1 hidden layer which is configured according to the rule of thumb that "the optimal size of the hidden layer is between the size of the input and size of the output layers" [29]. Note that although model configuration greatly impacts its accuracy, this topic outside the scope of our work.

4 EXPERIMENTAL EVALUATION

Experimental Setup: All experiments are conducted on a server with two Intel Xeon E5645 processors running at 2.40GHz, 96G of main memory and 2TB of hard disk, running Ubuntu 14.04.2 LTS with Linux kernel 3.13.05. Our prototype is implemented in Go (v.1.7.6). R (v.3.3.3) was utilized for MDS and NN training.

Table 2: Datasets and Operators

ID	Description	Datasets	Tuples	Operators
CLU	Google Cluster Monitoring [2]	4797	46 – 2188	**AVG**, **SUM**, COUNT (**CNT**), DBSCAN (**DBS**), Local Outlier F. (**LOF**), Eigenvalue (**EIG**), Regression (**REG**)
HPO	Household Power Consumption [35]	1442	1263 – 1440	
WEA	Weather Station Recordings [3]	552	300 – 8766	
NAS	NASDAQ Tech. Stocks [5]	231	252	Holt-Winters (**HOL**)
WIK	Wikipedia Page Visits [7]	4503	551	ARIMA (**ARI**)

Datasets & Operators: For the evaluation, five real-world sets of datasets are utilized, provided in Table 2. *CLU* provides the monitoring metrics of 4797 physical hosts (each dataset contains metrics from one host) of a data center at Google, running different tasks. Each tuple comprises 14 metrics of one task and different hosts run a varying number of tasks (ranging from 46 – 2188 tasks/host). *HPO* contains 1442 datasets with daily power measurements of a household in Denmark. The measurements usually take place every minute (except for some days where outages were witnessed) and each one comprises 7 features. *WEA* contains weather recordings from 552 different weather stations (w.s.) in 6 countries during 2016. Different w.s. gather measurements in different time intervals, hence, the number of tuples between different datasets ranges from 300 – 8766. All measurements comprise 6 features. For *CLU*, *HPO* and *WEA*, we test the 7 operators presented in the right column of Table 2. The Aggregate Functions are applied for one column of each dataset, whereas Linear Regression is applied for *HPO* and attempts to construct a linear model for predicting the active power of a metering zone based on the rest of the metrics. *NAS* contains various measurements of the NASDAQ Technology Sector stocks, for the interval 2016/05/30 – 2017/05/30 (1 dataset represents 1 stock). Finally, *WIK* contains the number of visits for 4503 different Wikipedia articles for an interval of 551 days. For *NAS* and *WIK* we apply the two Time-Series forecasting algorithms. Throughout the evaluation, we will refer to each operator using the dataset id and the operator id (indicated in bold in Table 2). For example, *HPO-LOF* refers to the Local Outlier Factor operator for the *HPO* dataset.

Methodology: To evaluate our methodology's efficiency, we measure the accuracy of the trained ML model that predicts the value of each of the operators. Specifically, we adopt the Normalized Root Mean Squared Error, $NRMSE = \frac{1}{y_{max}-y_{min}}\sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i-\hat{y_i})^2}$ and the Median Absolute Percentage Error, $MdAPE = \underset{1\le i\le N}{median}\left|\frac{y_i-\hat{y_i}}{y_i}\right|$, where y_i represents the *actual* operator value applied over D_i, $\hat{y_i}$ represents its approximated value and y_{min}, y_{max} represent the minimum and maximum operator values respectively. MdAPE was preferred against the Mean Absolute Percentage Error since it is less susceptible to outliers. All the error metrics are estimated for *all* the datasets, i.e., each operator was exhaustively applied over all the available datasets for testing purposes, in order to avoid cases where the model consistently fails to approximate specific dataset space areas. Neural Network (NN) regression models are utilized for modeling. Each model is trained by executing the operator over a portion of the available datasets (this is referred to as the *sampling ratio*). The construction of similarity matrices that combine more than one dataset properties utilizes Equation 8:

$$Similarity(A,B) = \sum_X (w_X \cdot X(A,B)) \quad (8)$$

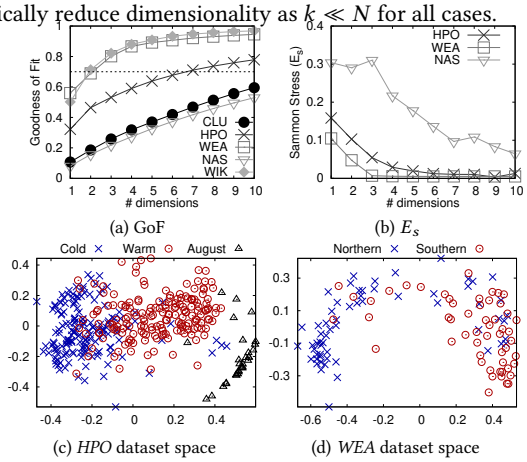
in which $X \in \{Distribution, Order, Size\}$ and $\sum_X w_X = 1$. Finally, in order to decide on the dimensionality of the produced dataset spaces, Equations 6 and 7 are used.

4.1 Dataset Space Construction

We begin our analysis by generating five dataset spaces, one for each set of datasets. For *CLU*, *HPO* and *WEA* the similarity matrix is solely calculated with the *Distribution* property, utilizing 32 partitions for the comparison. For *NAS* and *WIK*, the similarity matrix combines the *Distribution* and *Order* properties with equal weights. Figures 2 (a) and (b) provide the GoF and E_s values, respectively, when obtaining dataset spaces of varying dimensionality whereas Figures 2 (c) and (d) provide 2-d projections of the constructed *HPO* and *WEA* dataset spaces respectively.

Figures 2 (a) and (b) provide an estimate of the dimensionality of the space that should be employed in order to retain the distances of the similarity matrix without information loss. As more dimensions are employed, GoF increases and E_s decreases, something that indicates that the constructed space retains the calculated dataset distances more accurately. Observe that each set of datasets presents different dimensionality requirements for an approximation of high quality: *WEA* and *WIK* require only 3 dimensions in order to be transformed with $GoF > 0.75$, whereas *HPO* requires 7 dimensions and *NAS* and *CLU* require approximately 15. GoF's values are affected by the relationships between the datasets: The more similar the distances between the datasets, the more dimensions are required in order to generate an accurate representation of the matrix. The Sammon mapping was successfully executed only for 3 of the 5 cases, because *CLU* and *WIK* contained some

dataset pairs with distance 0. Since the Sammon mapping requires no points to overlap, it could not be executed for sets that presented this phenomenon. Observe how the Sammon mapping "corrects" the dataset coordinates in order to reduce the number of required dimensions. For example, although *HPO*'s GoF diagram indicates that 7 dimensions are needed, E_s presents a "knee" sooner than that: Since E_s presents similar values for 4 or 5 dimensions, one can use fewer dimensions without losing in accuracy. Finally, the two plots indicate that even for the sets with increased dimensionality requirements, the application of MDS and Sammon mapping can drastically reduce dimensionality as $k \ll N$ for all cases.



(a) GoF (b) E_s

(c) *HPO* dataset space (d) *WEA* dataset space

Figure 2: Dataset spaces, *GoF* and E_s plots

Figures 2 (c) and (d) demonstrate that a visual examination of the dataset spaces provides valuable information to a data analyst, as datasets with similar behavior are projected closer. In the *HPO* case, one can observe that the datasets are positioned according to the day of the year they were collected (only datasets of 2008 are depicted for legibility reasons) and they are organized in three clusters: Datasets from cold and warm days of the year and datasets from August. Note the interesting pattern: The household's energy consumption is much higher during the colder days (due to energy-hungry heating devices), decays during the warmer days and is zero during August. Similarly, this behavior is also exhibited for *WEA* (only depicting Swedish w.s.): The obtained datasets are clustered according to their geographic location and form two groups: The Northern ones (that present lower temperatures) and the Southern ones (with warmer and less humid climate).

4.2 Operator Modeling

For each set of datasets, we construct the similarity matrices based on the properties their operators are affected by, i.e., Distribution and Size for *CLU*, *HPO* and *WEA* (setting $w_{Distribution} = w_{Size} = 0.5$ to Equation 8) and *Distribution* and *Order* for *NAS* and *WIK* ($w_{Distribution} = w_{Order} = 0.5$) and, subsequently, execute MDS to obtain the respective dataset spaces. We, then, train Neural Networks utilizing different sampling ratios that vary from 4% to 32%. In Table 3, we provide the error metrics and the execution speedup.

Modeling Quality: The *NRMSE* and *MdAPE* columns of Table 3 provide the modeling error of our approach (less is better). Generally, we can observe that increasing sampling ratios lead to decreased modeling errors. Examining each operator class in isolation, the Aggregate Functions, present low modeling errors for all datasets. Interestingly, when only a mere 4% of datasets is examined, one can approximate the operators values with an *MdAPE* that varies from a minimal 2 to 15% for most cases (e.g., *HPO-AVG* produced less than 2% for all sampling ratios), with the exception of *CNT* that presents the highest *MdAPE* for *CLU* and *WEA*. In comparison to the other aggregate operators, *CNT* is the one that presents values closest to 0, something that contributes to increased *MdAPE* without actually indicating poor approximation [38]. The Density based operators (*LOF* and *DBS*) also present a robust behavior, measured both in terms of *NRMSE* and *MdAPE*. *LOF* can be modeled with remarkable accuracy (6.5% *MdAPE* when 4% of the datasets are considered) and improves with increasing sampling ratios for all datasets. The least accurate operator of this class is *DBS* that presents a modeling error that quickly reduces with increasing sampling ratios. This operator presents an interesting pattern: Some datasets with different distributions (which are evidently located in distant positions in the dataset space) presented similar outcomes, i.e., number of clusters. This unavoidable situation means that there does not exist a monotonic relationship between an operator's scores and their position to the dataset space. This means that the model needs to obtain more samples in order to increase its accuracy, hence the error degradation with increasing sampling ratios. *REG* and *EIG* also present robust behavior for all the examined datasets. *EIG* in particular, presented *MdAPE* less than 10% for all the examined datasets. This means that one can approximate the value of the most important eigenvalue of each dataset with a minimal error through actually running the operator for a minimal subset of the available datasets. On the contrary, the Time-Series operators exhibit a seemingly abnormal behavior where they present low *NRMSE* and abnormally high *MdAPE* that declines fast with increasing sampling ratios. As in the *CNT* operator case, the fact that both *HOL* and *ARI* produce values close to 0 leads to increased *MdAPE* values. The fact that these values decline fast with the sampling ratio means that while more knowledge is obtained, the approximated scores increase and, hence, *MdAPE* decreases. In a nutshell, the demonstrated errors indicate that our methodology successfully approximates all the considered operators. The fundamental idea behind our approach, that dataset spaces constructed in such a manner so as to reflect the relationships between them, facilitates the training of Machine Learning models that accurately approximate the behavior of the operators.

Speedup: The *Speedup* and *Amortized Speedup* columns of Table 3 provide an estimate of the time needed to approximate each operator in comparison to exhaustively executing them for all datasets (more is better). Specifically, the speedup equals $\frac{T_{op}^{(i)}}{SR \times T_{op}^{(i)} + T_{SM} + T_{MDS} + T_{ML}}$, where $T_{op}^{(i)}$ is the execution time of the i-th operator for all datasets, SR is the sampling ratio, T_{SM} is the time needed to construct the similarity matrix, T_{MDS} the time of executing Multidimensional Scaling and Sammon mapping and T_{ML} is the time needed to train the Neural Network. Considering that T_{SM}, T_{MDS} are only paid once for each set of datasets, we also calculate the Amortized Speedup where $T_{op}^{(i)}$ is replaced with $\sum_i T_{op}^{(i)}$, i.e., the execution time for all operators for each case. The examination of the *Speedup* column indicates that the sampling ratio is inversely proportional to the

Table 3: Modeling Accuracy and Execution Speedup of Operators

Operator	NRMSE				MdAPE				Speedup (×)				Amortized Speedup (×)			
	4%	8%	16%	32%	4%	8%	16%	32%	4%	8%	16%	32%	4%	8%	16%	32%
CLU-AVG	0.086	0.079	0.073	0.066	0.125	0.114	0.100	0.082	3.21	2.84	2.32	1.69	16.34	9.88	5.52	2.93
CLU-SUM	0.085	0.077	0.070	0.063	0.182	0.158	0.136	0.114	3.21	2.84	2.32	1.69				
CLU-CNT	0.115	0.108	0.104	0.097	0.433	0.401	0.377	0.339	3.21	2.84	2.32	1.69				
CLU-DBS	0.098	0.093	0.088	0.083	0.201	0.191	0.173	0.152	5.69	4.63	3.83	2.19				
CLU-LOF	0.082	0.074	0.070	0.066	0.146	0.136	0.125	0.110	12.13	8.17	4.94	2.76				
CLU-EIG	0.069	0.063	0.058	0.053	0.089	0.079	0.071	0.060	4.27	3.65	2.83	1.95				
HPO-AVG	0.104	0.096	0.088	0.084	0.013	0.012	0.011	0.010	3.93	3.4	2.67	1.87	20.27	11.20	5.91	3.04
HPO-SUM	0.070	0.065	0.056	0.051	0.149	0.135	0.122	0.113	3.93	3.4	2.67	1.87				
HPO-CNT	0.098	0.079	0.069	0.061	0.115	0.104	0.092	0.084	3.93	3.4	2.67	1.87				
HPO-DBS	0.124	0.119	0.114	0.111	0.146	0.141	0.133	0.128	8.30	6.23	4.16	2.50				
HPO-LOF	0.064	0.061	0.055	0.052	0.068	0.063	0.061	0.057	16.64	9.99	5.55	2.94				
HPO-EIG	0.071	0.069	0.067	0.065	0.065	0.063	0.059	0.055	7.33	5.67	3.90	2.72				
HPO-REG	0.073	0.071	0.071	0.069	0.162	0.150	0.134	0.124	11.33	7.80	4.80	2.72				
WEA-AVG	0.089	0.074	0.068	0.059	0.035	0.025	0.020	0.018	2.68	2.42	2.03	1.53	18.72	10.71	5.77	3.00
WEA-SUM	0.075	0.068	0.063	0.057	0.114	0.078	0.059	0.047	2.68	2.42	2.03	1.53				
WEA-CNT	0.119	0.106	0.091	0.080	0.324	0.284	0.244	0.214	2.68	2.42	2.03	1.53				
WEA-DBS	0.182	0.180	0.176	0.171	0.323	0.328	0.303	0.288	6.06	4.88	3.51	2.25				
WEA-LOF	0.126	0.123	0.115	0.110	0.118	0.113	0.107	0.093	16.71	10.02	5.56	2.94				
WEA-EIG	0.035	0.032	0.031	0.029	0.024	0.021	0.019	0.018	5.59	4.57	3.35	2.18				
NAS-HOL	0.093	0.090	0.086	0.084	0.700	0.445	0.333	0.283	0.65	0.63	0.60	0.55	3.45	3.03	2.44	1.75
NAS-ARI	0.095	0.090	0.085	0.084	0.773	0.548	0.341	0.262	2.94	2.63	2.17	1.61				
WIK-HOL	0.018	0.018	0.018	0.018	0.812	0.686	0.582	0.353	0.17	0.16	0.16	0.16	1.42	1.34	1.21	1.01
WIK-ARI	0.019	0.019	0.019	0.019	0.595	0.488	0.324	0.237	1.27	1.20	1.10	0.93				

achieved speedup. Lower sampling ratios indicate that an operator is executed to less datasets and, hence, the achieved speedup is greater. The exact value of the speedup is directly related to the complexity of the operator. When the employed operator is complex and its execution time is large, as in the *LOF* cases, the achieved speedup approximates the optimal one, which is $\frac{1}{SR}$. For example, *HPO-LOF* presents a speedup of 16.64× when 4% of the datasets are considered. On the contrary, when the operator is less complex, the achieved speedup is limited (e.g., *CLU-AVG* presents a speedup of 3.21 when 4% of the datasets are considered), because the construction time of the similarity matrix and the MDS execution is not counterbalanced by the avoidance of operator executions. Nevertheless, when the similarity matrix construction and MDS execution is amortized to more than one operators, the achieved speedup largely increases. This is visible for the first 3 sets of datasets where the achieved speedup closely approximates $\frac{1}{SR}$. For example, the amortized speedup of *HPO* is 20.27 for a sampling ratio of 4%. This highlights the power of the suggested methodology. As more operators emerge, they utilize the previously computed dataset space and the amount of avoided computation increases linearly with their number. However, if the cost of constructing a dataset space is high, the achieved speedup may require many more operators in order to counterbalance the offline cost. The *NAS* and *WIK* sets require more computation time for the construction of their similarity matrices since the *Order* property entail the evaluation of Equation 3 which is more expensive than the previous cases. This is the reason behind the lower speedup values encountered to *NAS* and *WIK*. Nevertheless, an increasing number of operators will, eventually, lead them to save an increasing amount of execution time and, hence, accelerate the analysis for these cases as well. Evidently, the observed speedup is strongly affected by two factors: (a) the complexity of the operators and (b) the complexity of the similarity expression. Avoiding the execution of complex operators leads to higher speedups as soon as the similarity expression is efficient. For example, *CLU-LOF* presents a remarkable speedup of 16.64 both because the operator's execution time is increased and due to the low complexity of the similarity expression. Therefore, it is crucial to obtain similarity expressions that are both efficient in terms of expressiveness and complexity.

4.3 Combining Similarity Metrics

We now evaluate the impact of combining multiple similarity metrics for the *WEA* case to our scheme. Each weather station collects measurements in different intervals and, thus, the size of the datasets varies between 300 – 9000 tuples. We construct two similarity matrices based on (a) the Distribution similarity property and (b) the Size similarity property. Based on those, three more similarity matrices are constructed where $(w_{Distr.}, w_{Size})$ equals to $(0.2, 0.8)$, $(0.5, 0.5)$ and $(0.8, 0.2)$. All matrices are then transformed to 5-d dataset spaces and, subsequently, model *WEA-AVG* and *WEA-SUM* for each space, utilizing sampling ratios between 2% – 20%. The modeling part is executed 20 times and Figure 3 depict the results.

We remind here that *AVG* is only affected by the *Distribution* of the datasets whereas *SUM* is affected both by *Distribution* and *Size*. Figure 3 (a), showcases that the most informative space, i.e., the one with the least error, is constructed using only the Distribution property and the sole consideration of Size creates a space with no information as regardless of the increasing sampling ratio, the error is not reduced. However, when constructing spaces that combine both parameters, the size seems to be ignored. Interestingly, when the respective weights are equal or when favoring the important property, e.g., as in the (0.8, 0.2) case, the model is practically unaffected by the consideration of the Size. On the contrary, when an operator is affected by both properties, as in the *WEA-SUM* case, the spaces born from the combination of the primitive matrices are far more informative than the primitive ones and lead the models to increase their accuracy which is, in turn, increasing with the sampling ratio. This practically means that when a

space is constructed using a combination of dataset properties, the simpler operators tend to ignore the non-interesting – to them – properties and do not present degraded modeling accuracy. On the contrary, the operators that do require the property combinations to be reflected to the space, are modeled much more efficiently than considering one property at a time. Finally, in cases where the users are knowledgable about the applied operator, the adjustment of the weights can result in a dataset space that best projects the property of interest, e.g., Distribution in the *WEA-AVG* case.

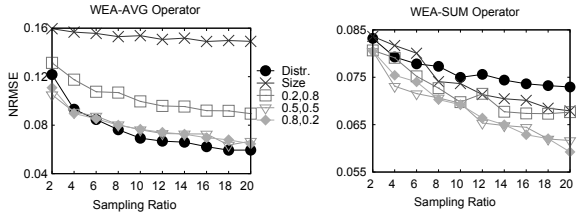


Figure 3: Combining data properties

4.4 Distribution Similarity Granularity

We now evaluate the impact of the number of partitions (l) during the examination of the similarity of *Distribution*. Based on *HPO*, we calculate 8 different similarity matrices, using different numbers of partitions. Each of them is, subsequently, transformed to 5-d dataset spaces and, finally, a NN model for *HPO-AVG* is trained, using three different sampling ratios (2%, 8% and 16%). Recall that *HPO-AVG* is only affected by the *Distribution* of the datasets. The modeling part is repeated 20 times and in Figure 4 (a) we provide the median modeling error. Figure 4 (b) depicts the execution time of the similarity matrix construction and MDS and Figure 4 (c) depicts E_s when different dimensionality is employed for varying l.

Figure 4 (a) presents an interesting finding: Although an increasing l (between 4 – 64) seems to decrease the modeling error, as more partitions imply that the similarity comparison is more detailed, increasing l beyond a point seems to increase the error. In fact, for higher sampling ratios, it is preferable to utilize a smaller l value rather than a higher one, as for the 16%, the error for $l = 4$ is lower than the error in the $l = 256$ case. This finding is explained when examining Figure 4 (c): The utilization of more partitions increases the level of detail for the similarity estimation and, hence, an increasing number of dimensions is necessary in order to accurately transform the similarities to a dataset space. Since all cases from (a) comprise 5 dimensions, the similarity matrices with more partitions are less accurately projected and, hence, less accurately modeled. Therefore, there exists a clear dependency between l and k: More partitions require space of higher dimensionality. Furthermore, the execution time presents another interesting pattern: The time needed to construct the matrix increases linearly with l but MDS seems to require more time when fewer partitions are employed. Fewer partitions leave more room for the "optimal" dataset placement in 5 dimensions and SA needs more time to converge to this point. However, constructing similarity matrices with increased number of partitions and, hence, dimensions, does not always benefit the analysis. To evaluate this, in Figure 4 (d) we provide the median error of NN models trained considering only 8% of the available datasets, using spaces of different dimensionality (horizontal axis), estimated using three different l values. For higher l values, more dimensions need to be utilized to decrease the modeling accuracy. However, the utilization of more dimensions than actually needed (e.g., when $l = 64$ there is no need to use more than 5 dimensions) leads to complex models and the modeling accuracy degrades. There exists an opportune area for selecting the appropriate dimensionality.

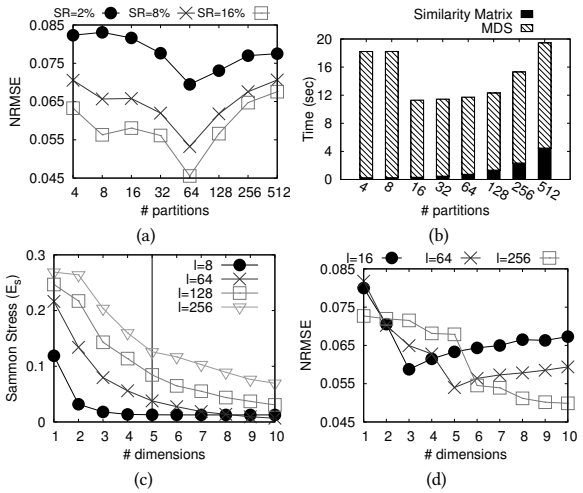


Figure 4: Distribution Similarity Granularity

5 RELATED WORK

The idea of combining data from different sources in order to increase their utility is the key idea behind *Data Integration*. The work in [34] outlines the requirements for the implementation of data integration systems and provides the theoretical models to express the basic operations over distinct datasets. The authors focus on query answering and reasoning using the distinct datasets and attempt to unify them in the most transparent way. On the contrary, our work exploits the differences between the distinct datasets and targets to model them, creating a feature space to model the operators' behavior. Ground [28] is a data context service that focuses on the creation of the appropriate metadata that informs the data scientist on the possible use of each dataset. The *context* of the data retains information regarding the representation of the dataset, details about how data was created and versioning history, in order to keep track of data updates. CrossCat [39] focuses on analyzing high dimensional data. It relies on mixture modeling and Bayesian structure learning. It evaluates each data column separately, constructs different views of the data and uses a separate non-parametric mixture to model each view. Although this work attempts to obtain better knowledge for existing datasets, it does not take into consideration the relationships between different datasets.

The identification of the key properties of a dataset is the target of Data Exploration. Data Canopy [44] demonstrates the power of statistical analysis over unknown datasets in order to infer knowledge. The main idea of this work lies on generating a set of *basic aggregates*, which can be synthesized in order to infer knowledge without constantly accessing the raw data. Data Canopy, similarly to our work, highlights the necessity of statistically analyzing the datasets once and utilizing the analysis for consequent tasks. Yet, in

this work, we are interested in predicting complex analytics operator performance considering more complex statistical properties for the datasets (such as distribution) than simple aggregates. AIDE [22] is a database exploration system using Active Learning. It implements an iterative methodology, in which it requests feedback from the users whether the returned tuple is of interest to them. While the users provide more responses, AIDE isolates the interesting areas of the database and returns them to the user in the form of SQL queries. This work is driven by the assumption that the user is unaware of the database and only provides yes/no responses. On the contrary, our work requires no feedback from the user and the "interesting" datasets are determined by operator performance.

Finally, data profiling and instance selection methodologies tackle a similar problem to this work but from a different angle. [24] presents works that address the challenge of instance selection for active learning. In this setup, a classifier is trained in an *online* fashion and the labels of the training points are either unknown or hard to be obtained. The objective of these approaches is to provide efficient heuristics to select a subset of points that minimize the classifier's uncertainty under given cost constraints. Similarly, Zombie [11] is a system that conducts input selection for feature engineering. The main idea of this work is to focus on the appropriate tuples of a given dataset in order to accelerate feature selection for machine learning tasks. Although the concept of data utility is common between these works and our approach, our work emphasizes not on tuple selection but on highlighting the differences between different datasets. Moreover, the results of our work can be used with different utility functions through applying different operators for the same spaces. Finally, [9] presents different approaches towards data profiling that aim at automatically extracting metadata for given datasets. This metadata can, in turn, be utilized for clustering and categorizing them according to their usage and utility for different operators. Our work is also based on this idea and extends it: Using a unique (i.e., not operator-dependent) knowledge basis, our methodology can infer the outcome of diverse operators when applied to datasets that belong to the same clusters.

6 CONCLUSIONS

In this paper, we tackled the problem of modeling the behavior of an operator given a large set of input datasets. We proposed a novel methodology that relies on estimating the similarity between distinct datasets, based on their distribution, ordering and size. Through these similarities, knowledge inferred about the datasets is transformed to a low-dimensional space and utilized by ML models to approximate the behavior of a given operator for all the available datasets. In practice, this approach proved capable of achieving very accurate performance models, while it can gracefully degrade its efficiency over customizable gains in execution cost.

REFERENCES

[1] 2014. The Big Problem Is Medium Data. http://goo.gl/5nYrrz.
[2] 2015. Google Cluster Monitoring Dataset. https://github.com/google/cluster-data/. Online; accessed Feb 2018.
[3] 2016. National Centers for Environmental Information. https://www1.ncdc.noaa.gov/pub/data/noaa/. Online; accessed May 2017.
[4] 2017. Apache Flink. https://flink.apache.org/.
[5] 2017. Google Finance API. https://www.google.com/finance/historical.
[6] 2017. Medium Data is the New Sweet Spot. https://goo.gl/mnxnEx.
[7] 2017. Web Traffic Time Series Forecasting. https://www.kaggle.com/c/web-traffic-time-series-forecasting/data. Online; accessed Feb 2018.
[8] 2018. Data Profiler source code. https://github.com/giagiannis/data-profiler.
[9] Ziawasch Abedjan, Lukasz Golab, and Felix Naumann. 2015. Profiling relational data: a survey. *The VLDB JournalâĂŤThe International Journal on Very Large Data Bases* 24, 4 (2015), 557–581.
[10] Naomi S Altman. 1992. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician* 46, 3 (1992), 175–185.
[11] Michael R Anderson and Michael Cafarella. 2016. Input selection for fast feature engineering. In *Data Engineering (ICDE), 2016 IEEE 32nd International Conference on*. IEEE, 577–588.
[12] Michael Armbrust, Armando Fox, Rean Griffith, and Joseph others. 2009. *Above the clouds: A berkeley view of cloud computing*. Technical Report. Technical Report UCB/EECS-2009-28, EECS Department, University of California, Berkeley.
[13] David Arthur and Sergei Vassilvitskii. 2007. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, 1027–1035.
[14] Ricardo A Baeza-Yates. 2013. Big Data or Right Data?. In *AMW*.
[15] Peter Bailis, Edward Gan, Kexin Rong, and Sahaana Suri. 2017. Prioritizing attention in fast data: Principles and promise. *CIDR* (2017).
[16] Adrian Bowyer. 1981. Computing dirichlet tessellations. *The computer journal* 24, 2 (1981), 162–166.
[17] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. 2015. *Time series analysis: forecasting and control*. John Wiley & Sons.
[18] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. 2000. LOF: identifying density-based local outliers. In *ACM sigmod record*. ACM.
[19] Chris Chatfield. 1978. The holt-winters forecasting procedure. *Applied Statistics* (1978), 264–279.
[20] Kyriakos Chourdakis, E Epperlin, MARC Jeannin, and James Mcewen. 2013. A cross-section across CVA. *Nomura. Available at Nomura: http://www. nomura. com/resources/europe/pdfs/cva-crosssection. pdf* (2013).
[21] Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer. 2000. Real-time tracking of non-rigid objects using mean shift. In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, Vol. 2. IEEE, 142–149.
[22] Kyriaki Dimitriadou, Olga Papaemmanouil, and Yanlei Diao. 2016. AIDE: An Active Learning-Based Approach for Interactive Data Exploration. *IEEE Transactions on Knowledge and Data Engineering* 28, 11 (2016), 2842–2856.
[23] Martin Ester, Hans-Peter Kriegel, et al. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise.. In *Kdd*.
[24] Yifan Fu, Xingquan Zhu, and Bin Li. 2013. A survey on instance selection for active learning. *Knowledge and information systems* 35, 2 (2013), 249–283.
[25] John C Gower. 1966. Some distance properties of latent root and vector methods used in multivariate analysis. *Biometrika* (1966), 325–338.
[26] Jon Gregory. 2010. *Counterparty credit risk: the new challenge for global financial markets*. Vol. 470. John Wiley & Sons.
[27] Simon Haykin. 1994. *Neural networks: a comprehensive foundation*. Prentice Hall PTR.
[28] Joseph M Hellerstein, Vikram Sreekanti, Joseph E Gonzalez, James Dalton, Akon Dey, Nag, et al. 2017. Ground: A Data Context Service.. In *CIDR*.
[29] TH Jeff. 2005. Introduction to Neural Networks with Java, Heaton Research.
[30] Ian T Jolliffe. 1986. Principal Component Analysis and Factor Analysis. In *Principal component analysis*. Springer, 115–128.
[31] Maurice George Kendall. 1948. Rank correlation methods. (1948).
[32] William R Knight. 1966. A computer method for calculating Kendall's tau with ungrouped data. *J. Amer. Statist. Assoc.* 61, 314 (1966), 436–439.
[33] William H Kruskal. 1958. Ordinal measures of association. *J. Amer. Statist. Assoc.* 53, 284 (1958), 814–861.
[34] Maurizio Lenzerini. 2002. Data integration: A theoretical perspective. In *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. ACM, 233–246.
[35] Lichman. 2013. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml
[36] Yoseph Linde, Andres Buzo, and Robert Gray. 1980. An algorithm for vector quantizer design. *IEEE Transactions on communications* 28, 1 (1980), 84–95.
[37] Sam Madden. 2012. From databases to big data. *IEEE Internet Computing* (2012).
[38] Spyros Makridakis. 1993. Accuracy measures: theoretical and practical concerns. *International Journal of Forecasting* 9, 4 (1993), 527–529.
[39] Vikash Mansinghka, Patrick Shafto, Eric Jonas, Cap Petschulat, Max Gasner, and Joshua B Tenenbaum. 2016. CrossCat: a fully Bayesian nonparametric method for analyzing heterogeneous, high dimensional data. *The Journal of Machine Learning Research* 17, 1 (2016), 4760–4808.
[40] David R Miller, Shon Harris, Allen Harper, Stephen VanDyke, and Chris Blask. 2010. *Security Information and Event Management (SIEM) Implementation (Network Pro Library)*. McGraw Hill.
[41] Peter Schlampp. 2016. Spark takes on the big security threats. http://www.ibmbigdatahub.com/blog/spark-takes-big-security-threats.
[42] John W Sammon. 1969. A nonlinear mapping for data structure analysis. *IEEE Transactions on computers* 100, 5 (1969), 401–409.
[43] Peter JM Van Laarhoven and Emile HL Aarts. 1987. Simulated annealing. In *Simulated Annealing: Theory and Applications*. Springer, 7–15.
[44] Abdul Wasay, Xinding Wei, Niv Dayan, and Stratos Idreos. 2017. Data Canopy: Accelerating Exploratory Statistical Analysis. In *Proceedings of the 2017 ACM International Conference on Management of Data*. ACM, 557–572.
[45] Matei Zaharia, Mosharaf Chowdhury, Michael J Franklin, Scott Shenker, and Ion Stoica. 2010. Spark: Cluster computing with working sets. *HotCloud* (2010).