DiVE: Diversifiying View Recommendation for Visual Data Exploration

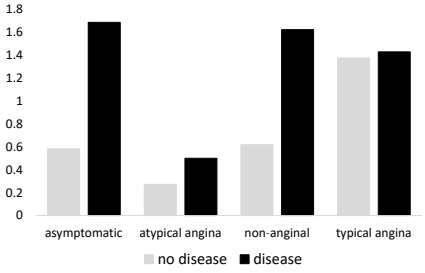
ABSTRACT

To support effective data exploration, there has been a growing interest in developing solutions that can automatically recommend data visualizations that reveal interesting and useful data-driven insights. In such solutions, a large number of possible data visualization views are generated and ranked according to some metric of importance (e.g., a deviation-based metric), then the top-k most important views are recommended. However, one drawback of that approach is that it often recommends similar views, leaving the data analyst with a limited amount of gained insights. To address that limitation, in this work we posit that employing diversification techniques in the process of view recommendation allows eliminating that redundancy and provides a good and concise coverage of the possible insights to be discovered. To that end, we propose a hybrid objective utility function, which captures both the importance, as well as the diversity of the insights revealed by the recommended views. While in principle, traditional diversification methods (e.g., Greedy Construction) provide plausible solutions under our proposed utility function, they suffer from a significantly high query processing cost. In particular, directly applying such methods leads to a "process-first-diversify-next" approach, in which all possible data visualization are generated first via executing a large number of aggregate queries. To address that challenge and minimize the incurred query processing cost, we propose an integrated scheme called *DiVE*, which efficiently selects the top-k recommended view based on our hybrid utility function. Specifically, *DiVE* leverages the properties of both the importance and diversity metrics to prune a large number of query executions without compromising the quality of recommendations. Our experimental evaluation on real datasets shows that DiVE can reduce the query processing cost by up to 40% compared to existing methods.

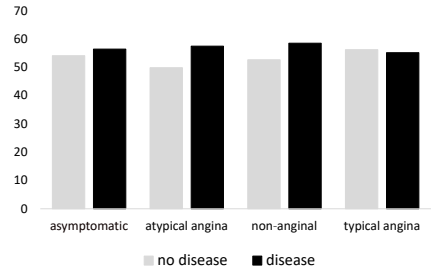
ACM Reference Format:
. 1997. DiVE: Diversifiying View Recommendation for Visual Data Exploration. In *Proceedings of ACMconference (CIKM 2018).* ACM, New York, NY, USA, Article 4, 14 pages. https://doi.org/10.475/123_4

CIKM 2018, October 2018,

ACM ISBN 123-4567-24-567/08/06...$15.00
https://doi.org/10.475/123_4



(a) Visualization of the avg. oldpeak vs. chest pain types



(b) Visualization of the average age vs. chest pain types

Figure 1: Important vs. less important view.

1 INTRODUCTION

In the recent years, visualization recommendation systems have become an integral part of data exploration systems. The users who are interested in finding some meaningful insights in data have neither time nor patience to manually generate all possible data visualizations. In addition to time, the domain knowledge is another key factor when generating visualizations manually. However, with an exponential growth of available data in various domains, there has been an increase in the number of *Data Enthusiasts*, people with little domain knowledge and technical expertise, looking for interesting trends in data.

For instance, consider a Cleveland heart disease dataset [1], which describes patients with and without a heart disease. A data enthusiast might be interested in conducting some comparison between people with heart disease (disease) and people without heart disease (no disease). Without any prior insights about data, she must manually specify different combinations of attributes, measures and aggregate functions

[1] http://archive.ics.uci.edu/ml/datasets/heart+Disease

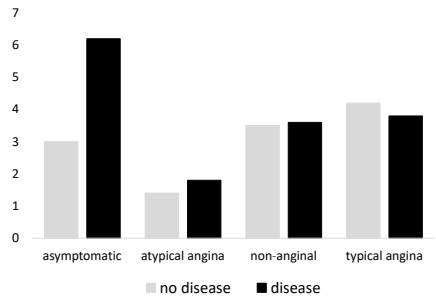


Figure 2: The visualization of maximum oldpeak vs. chest pain types

before finally generating a visualization that reveals some interesting information about the dataset. The user effort and time spent in that process increases exponentially with increase in the number of attributes and measures. Hence, several data-driven visualization recommendation tools have been proposed to reduce the user effort and time during data exploration [5, 17, 18]. The main goal of those recommendation systems is to provide the user with the most important visualizations (top-k views), which are selected from all possible visualizations. The top-k views are selected based on the most important views in the dataset. The importance of a view is defined on the basis of particular criteria. One of the widely used criteria for importance is based on the deviation between the queried subset of data (target view) with the reference subset of data (reference view). The reference subset can be another subset of the dataset, the rest of the dataset, or the whole dataset. The intuition behind deviation based approach is that views that reveal substantially different trends from the reference views are likely to be of higher interest to the user[17, 18].

Consider again the example of the heart disease dataset. Let the target subset be the data of people with heart disease and the reference subset be the data of people without heart disease. As shown in Figure 1a the average oldpeak (pressure of the ST segment) vs. chest pain types is more important view rather than the Figure 1b the average of age vs. chest pain types, due to the large deviation between target view (disease) and reference view (no disease) data. Figure 1a shows people with heart disease, especially who the chest pain types is asymptomatic tend to have much higher oldpeak rather than people without disease. To the contrary, the Figure 1b is potentially less important visualization compared to Figure 1a, even there is a deviation between disease and no disease but the deviation is very small and if it is compared to Figure 1a, it has lower deviation than Figure 1a. Figure 1b shows that there is no significant different in term of the average age of the people with four types of chest pain.

Although the deviation based visualization recommendation systems automatically provide users with the most important visualizations, it is likely that the views in the top-k set might be providing redundant information. For instance, Figure 2 provides the information which close to Figure 1a that the people with heart disease tend to have higher oldpeak values. Figure 2 has same attribute and attribute measure to Figure 1a, the only difference is the aggregate function, where Figure 1a uses AVG and Figure 2 uses MAX. Since both views have a high deviation from the reference subset, both will appear in the top-k set. This leads to an important observation that using only importance as the selection criteria may deliver redundant recommended views, which leads to presents not optimal insights.

However, novelty and diversity are one of the fundamental characteristics of any effective recommendation systems [3, 11, 22, 23]. Specifically, it is highly desirable that a visualization recommendation systems provides users with views that are both importance and also provide novel information that has not been revealed by the other views.

Towards designing an effective visualization recommendation systems that promotes both importance and novelty in recommended views, in this work, we propose an integrated approach called *DiVE*. In particular, *DiVE* aims to generate top-k visualizations that balance the tradeoff between importance and diversity. The main contributions of this paper are summarized as follows:

- We formulate the problem of evaluating recommended views that are both importance and diverse.
- We define a similarity measure to capture the distance between two visualizations.
- We present a hybrid objective function to balance the tradeoff between importance and diversity when ranking the visualizations.
- We propose the novel *DiVE* scheme, that employs various algorithms to evaluate the recommended visualizations based on the hybrid ranking/objective function.
- We present optimization techniques that leverage the hybrid objective function to substantially reduce the computational costs.
- We conduct an extensive experimental evaluation on real datasets, which compare the performance of various algorithms and illustrate the benefits achieved by *DiVE* both in terms of effectiveness and efficiency.

The rest of the paper is organized as follows: in Section II, we formulate the top-k diverse visualization problems and our related work; we present our proposed scheme *DiVE* in Section III; the experimental evaluation is reported in in Section IV and we conclude in Section V.

2 PRELIMINARIES AND RELATED WORK

Several recent research efforts have been directed to the challenging task of recommending aggregate views that reveal interesting data-driven insights (e.g., []). As in previous work, we assume a similar model, in which a visual data exploration session starts with an analyst submitting a query Q on a multi-dimensional database D_B. Essentially, Q selects a subset D_Q from D_B by specifying a query predicate T. Hence, Q is simply defined as:

Q: `SELECT * FROM` D_B `WHERE` T;

Ideally, the analysts would like to generate some aggregate views (e.g., bar charts or scatter plots) that unearth some valuable insights from the selected data subset D_Q. However, achieving that goal is only possible if the analyst knows exactly what to look for! That is, if they know the parameters, which specify aggregate views that lead to those valuable insights (e.g., aggregate functions, grouping attributes, etc.). Meanwhile, such parameters only become clear in "hindsight" after spending long time exploring the underlying database. Hence, the goal of existing work, such as [5, 7, 17–19], is to automatically recommend such aggregate views.

To specify and recommend such views, as in previous work, we consider a multi-dimensional database D_B, which consists of a set of dimensional attributes $\mathbb{A}$ and a set of measure attributes $\mathbb{M}$. Also, let $\mathbb{F}$ be a set of possible aggregate functions over measure attributes, such as `COUNT, AVG, SUM, MIN and MAX`. Hence, specifying different combinations of dimension and measure attributes along with various aggregate functions, generates a set of possible views $\mathbb{V}$ over the selected dataset D_Q. For instance, a possible aggregate view V_i is constituted by a tuple $<A_i, M_i, F_i>$, where $A_i \in \mathbb{A}$, $M_i \in \mathbb{M}$, and $F_i \in \mathbb{F}$, and it can be formally defined as:

V_i: `SELECT` A_i, F_i `(`M_i`) FROM` D_B `WHERE` T `GROUP BY` A_i;

Clearly, an analyst would be interested in those views that reveal interesting insights. However, manually looking for insights in each view $V_i \in \mathbb{V}$ is a labor-intensive and time-consuming process. For instance, consider again our example in the previous section. In that example, let D_B be the Cleveland Heart Disease data table (i.e., `tb_heart_disease`) and the analyst is selecting the subset of patients with heart disease (i.e., D_Q `= disease` subset). Hence, the number of views to explore is equal to: $|\mathbb{V}| = |\mathbb{A}| \times |\mathbb{M}| \times |\mathbb{F}|$, where $|\mathbb{F}|$ is the number of SQL aggregate functions, and $|\mathbb{A}|$ and $|\mathbb{M}|$ are the number of attribute and measures in `tb_heart_disease`, respectively. For that medium-dimensionality dataset, that value of $|\mathbb{V}|$ goes up to 180 views, which is clearly unfeasible for manual exploration. Such challenge motivated multiple research efforts that focused on *automatic* recommendation of views based on some metrics that captures the importance of a recommended view (e.g., [5–7, 14, 17–19]. it needs to be more specific - one sentence for each of those works! The point is to show there is a space of recommendation methods and we are selecting the deviation-based one Some of those works focus on recommending visualizations to facilitate a particular user intent or task [6, 8, 14]. For example: explanations for a certain behavior, finding data anomalies or outliers and correlations among data attributes. Hence, the criteria for ranking the visualizations is driven by the user intent. However, in visual data exploration, often the intent of the user is not clear. Towards that end, data driven metrics are employed to capture the interestingness or importance of a recommended visualization.

Among the data driven metrics, recent case studies have shown that a *deviation-based* metric is effective in providing analysts with interesting visualizations that highlight some of the particular trends of the analyzed datasets [12, 13, 18].

Table 1: Table of Symbols

Symbol	Description
k	no. of top recommended views
S	set of top-k recommended views
$\mathbb{V}$	set of all possible views
X	set of all candidate views
A	a dimensional attribute
M	a measure attribute
F	aggregate function
Q	a user query
D_B	a multi-dimensional database
D_Q	a target subset of D_B
D_R	a reference subset of D_B
V_i	a view query
$I(V_i)$	importance score of V_i
$I(S)$	importance score of views in S
$f(S,D)$	diversity score of views in S
$F(S)$	hybrid objective utility function value of S
$U(V_i)$	the utility score of each candidate view

In particular, the deviation-based metric measures the distance between V_iD_Q and V_iD_B. That is, it measures the deviation between the aggregate view V_i generated from the subset data D_Q vs. that generated from the entire database D_B, where V_iD_Q is denoted as *target* view, whereas V_iD_B is denoted as *reference* view. The premise underlying the deviation-based metric is that a view V_i that results in a higher deviation is expected to reveal some important insights that are very particular to the subset D_Q and distinguish it from the general patterns in D_B.

While recommending views based on their importance has been shown to reveal some interesting insight, it also suffers from the drawback of recommending similar and redundant views, which leaves the data analyst with a limited scope of the possible insights. To address that limitation, in this work we posit that employing diversification techniques in the process of view recommendation allows eliminating that redundancy and provides a good and concise coverage of the possible insights to be discovered. In the next section, we discuss in details the formulation of both importance and diversity, and their impact on the view recommendation process.

check table-text consistency and make shorter - done

3 DIVERSIFYING RECOMMENDED VISUALIZATIONS

short preamble

3.1 Content-Driven Deviation

As briefly described in the previous section, in this work we adopt a deviation-based metric to quantify the importance of an aggregate view [17, 18]. Essentially, the deviation-based

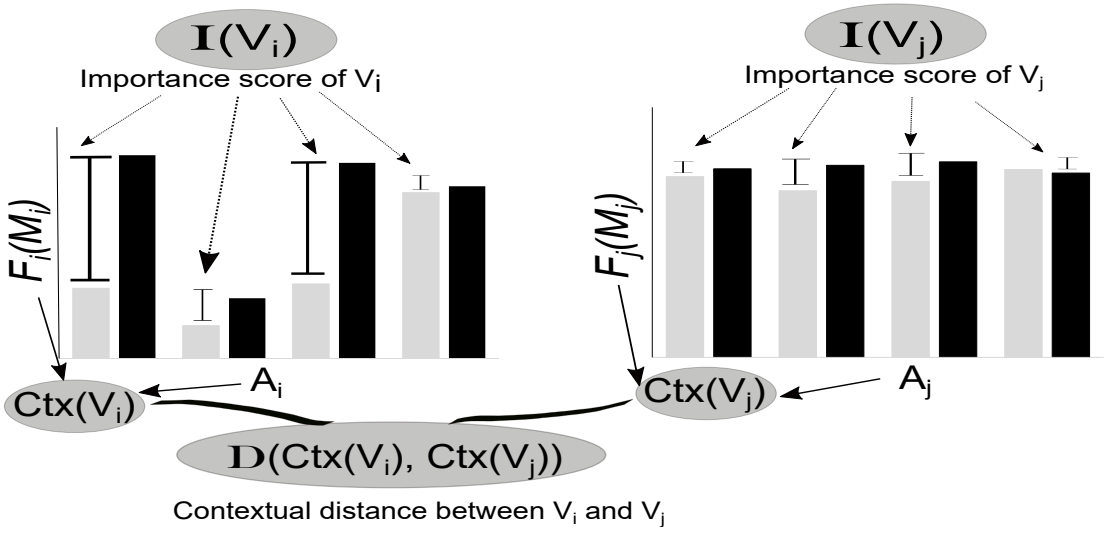


Figure 3: Content vs. Context of views.

metric compares an aggregate view generated from the selected subset dataset D_Q (i.e., target view V_iD_Q) to the same view if generated from a reference dataset D_R (i.e., reference view V_iD_R). That reference dataset could be the whole database (i.e.,$D_R = D_B$) or a selected subset of the database.

Clearly, the deviation between the target and reference views is a *data-driven* metric. That is, it measures the deviation between the aggregate result of V_iD_Q and that of V_iD_R. Consequently, from a visualization point of view, it is a *content-based* metric as it captures the difference between the content of the visualization generated by V_iD_Q vs. the visual content generated from V_iD_R. In the next, we formally describe the standard computation of that data-driven content-based metric, whereas the discussion of its counterpart context-driven metric is deferred to the next section.

To calculate that data-driven content-based deviation, each target view V_iD_Q is normalized into a *probability distribution* PV_iD_Q and similarly, each reference view into PV_iD_R. In particular, consider an aggregate view $V_i = A_i, M_i, F_i$. The result of that view can be represented as the set of tuples: $< a_j, g_j, a_j, g_j, ..., a_t, g_t >$, where t is the number of distinct values (i.e., groups) in attribute A_i, a_j is the j-th group in attribute A_i, and g_j is the aggregated value F_iM_i for the group a_j. Hence, V is normalized by the sum of aggregate values $G = \sum_{j=1}^{t} g_j$, resulting in the probability distribution $PV =< \frac{g_1}{G}, \frac{g_2}{G}, ..., \frac{g_t}{G} >$.

Finally, the importance score of V_i is measured in terms of a score that captures the distance between PV_iD_Q and PV_iD_R, and is simply defined as follows:

$$I\left(V_i\right) = dist\left(\mathcal{P}\left[V_iD_Q\right], \mathcal{P}\left[V_iD_R\right]\right) \quad (1)$$

where $I\left(V_i\right)$ is the importance score of V_i and *dist* is a distance function. In this work, we adopt the Euclidian distance, but other distance measures are also applicable (e.g., Earth Mover's distance, K-L divergence, etc.).

In current approaches for view recommendation, the importance value IV_i of each possible view V_i is computed, and the k views with the highest deviation are recommended (i.e., *top-k*) (e.g., [**?**]). However, in this work, our goal is to ensure that recommended views provide a good coverage of possible insights, which is achieved by considering the context of the recommended views, which is described next.

This metric measures how different a view is in relation to its reference. Hence, in order to compare the views, it needs to be a target subset and a reference subset of the D_B. Given target subset D_Q and D_R as the reference subset of D_B, we use a deviation-based metric as the content evaluation as employed in [17, 18]. The deviation value between target view V_iD_Q and reference view V_iD_R is called as importance score IV_i. The importance score of V_i is measured in terms of a score that captures the distance between probability distribution of target view V_iD_Q and probability distribution of reference view V_iD_R , it can be seen in Figure 3 and it can be formally defined as follow:

$$I\left(V_i\right) = dist\left(\mathcal{P}\left[V_iD_Q\right], \mathcal{P}\left[V_iD_R\right]\right) \quad (2)$$

where $I\left(V_i\right)$ is the importance score of V_i and *dist* is the Euclidian distance or other distance functions. The higher distance between target view V_iD_Q and reference view V_iD_R implies to higher importance score I. When judging the importance score of a view, it is believed that a view with large deviation from a reference view tends to be more interested to the user [17, 18].

To confirm that both views have the same scale, the target view V_iD_Q and reference view V_iD_R are normalized. The results of target and reference views are summaries with two columns, which first column is A and the second column is the aggregate FM. Hence, the normalized probability distribution can be calculated by dividing g_n by G, where g_n is the second column of the result which is the aggregate value of F over M for the group-by A, and $G = \sum_{i=1}^{n} g_n$. G is the sum of aggregate of all groups n in attribute A. The result of normalized probability distribution can be shown as $\frac{g_1}{G}, \frac{g_2}{G}, \frac{g_3}{G}, \frac{g_4}{G}$....., $\frac{g_n}{G}$.

stopped here. adjust the next section to consider the changes above

3.2 Context-Driven Deviation

As mentioned above, recommending top-k views based only on their data content (i.e., content-driven deviation) often leads to a set of similar views. In order to provide good coverage of all possible interesting insights, in this work, we posit that achieving *diversity* within the set of recommended views is an essential quality measure. Diversity has been well known and widely used in recommendation systems for maximizing information gain and minimizing redundancy(e.g., [11, 20, 22, 23]). At a high level, diversity essentially measures how different (i.e., diverse) are the individual data objects within a set.

Before discussing the details of diversity computation in Sec. **??**, it is important to notice that central to that computation is some notion of distance measure between data objects. Existing work provides multiple metrics for measuring that distance between traditional data objects, such as web documents (e.g., []), database tuples (e,g„ []), etc. However, our work in this paper is the first to consider diversity in the context of aggregate data views.

Consequently, in this work, an aggregate data visualization is perceived in terms of its underlying query. That is, the query that has been executed to create the visualization. In turn, the distance between two visualizations is measured based on the distance between their underlying queries. Hence, in addition to the data-driven content-based deviation described above, here we also introduce a query-driven *context-based* deviation metric.

To measure the context-based deviation between two visualizations, we simply measure the distance between their underlying queries. Towards this, we extend on existing work in the area of query recommendation and refinement (e.g., [**? ? ?**]). In that work, the distance between two range queries q_1 and q_2 is mapped to that of measuring the edit distance needed to transform q_1 into q_2, where the set of allowed transformation are: add, delete, or modify a predicate. In the context of our work, however, views are generated from aggregate queries without range predicates. In particular, a view is fully defined in terms of a combination of attribute, measure and an aggregate function. For instance, consider an aggregate view $V_i =< A_i, M_i, F_i >$. Hence, the context C_{tx} of V_i can then be defined as: $C_{tx}V_i = \left[A_i, M_i, F_i\right]$

As mentioned earlier, top-k interesting views based on the content-driven deviation might be very similar to each other. In order to provide good coverage of the possible insights, the diversity within the recommended set of views is an essential quality measure. In fact, diversity has been well known and widely used in recommendation systems for maximizing information gain and minimizing redundancy[11, 20, 22, 23]. The diversity of the set of views essentially measures how different are the individual visualizations in the recommended set.

In this work, we measure the diversity in terms of the difference in the context of the views. Hence, in order to determine the similarity between two views, we need to define the context of each view. The attribute, measure and the aggregate function used to generate a particular view defines the context of that view. For instance, consider an aggregate view $V_i = A_i, M_i, F_i$. The context C_{tx} of V_i can then be defined as:

$$C_{tx}V_i = \left[A_i, M_i, F_i\right]$$

For calculating the distance between contexts of two views, a suitable distance measure need to be applied.

what is this paragraph? It was in black There have been a tons of similarity measurement which proposed in the literatures such as 1) binary similarity [2]; 2) cube similarity measure for multidimensional data [1]. There are also well known similarity measurements such as Euclidien, Jaccard, Manhattan, Hamming [2] that can be used for our work.

Such definition of view context leads to a special case of the existing work on query recommendation (e.g., [**? ? ?**]), in which the normalized distance between two queries is simply measured using the Jaccard similarity measure. Hence, the Jaccard similarity between two aggregate views V_i and V_j is measured as: $J(C_{tx}V_i, C_{tx}V_j) = \frac{|C_{tx}V_i \cap C_{tx}V_j|}{|CtxV_i \cup CtxV_j|}$

We note that the jaccard similarity assigns equal weights to each of the element in a set. Accordingly, when applied to aggregate views, then two views with same attribute and different measure and aggregate function will have same similarity score as any other pair of views with same measure but different attribute and aggregate function. However, an analyst may consider two views with the same attribute A_i more similar than two views with same measure attribute M_i. To allow the analyst to specify such preference, each contextual component of a view is associated with a weight that specifies its impact on determining the (dis)similarity between views. Specifically, let w_i be the weight assigned to i^{th} element of set $C_{tx}V_i$, where $\sum_{i=1}^{3} w_i = 1$. Then, the similarity between views V_i and V_j is measured as: $J(C_{tx}V_i, C_{tx}V_j) = \frac{\sum_{i \in V_i \cap V_j} w_i}{\sum_{i \in V_i \cup V_j} w_i}$

Consequently, the context-based deviation between V_i and V_j is calculated as:

$$D\left(C_{tx}V_i, C_{tx}V_j\right) = 1 - J\left(C_{tx}V_i, C_{tx}V_j\right) \quad (3)$$

Since, the context of each view is defined as a set of three categorical elements, we employ Jaccard similarity measure which is a well-established similarity measure for sets of

categorical data. The Jaccard method measures similarity between finite sets as intersection of sets over union of sets. Hence, the Jaccard similarity between two views V_i and V_j can be measured as: $J(C_{tx}V_i, C_{tx}V_j) = \frac{|C_{tx}V_i \cap C_{tx}V_j|}{|CtxV_i \cup CtxV_j|}$

The jaccard similarity method gives equal weightage to each categorical element of the set. For instance, two views with same attribute and different measure and aggregate function will have same similarity score as any other pair of views with same measure but different attribute and aggregate function. However, user may consider views with same attributes more similar than the views with same measure attributes. Hence, we assign weights to each contextual component of a view. Let w_i be the weight assigned to i^{th} element of set $C_{tx}V_i$ where $\sum_{i=1}^{3} w_i = 1$.(not sure about sum of ratios being 1 as the example rischan used is 3:2:1 which does not sum up to 1).

Considering the weights assigned to each element of the context of a view, we modify the jaccard similarity measure as :

$$J(C_{tx}V_i, C_{tx}V_j) = \frac{\sum_{i \in V_i \cap V_j} w_i}{\sum_{i \in V_i \cup V_j} w_i}$$

The contextual distance between V_i and V_j is then calculated as:

$$D\left(C_{tx}V_i, C_{tx}V_j\right) = 1 - J\left(C_{tx}V_i, C_{tx}V_j\right) \quad (4)$$

I am not sure what is that paragraph either? But something needs to be said about the figure! Maybe one sentence in each of the previous section, then simple discussion here Thus, in the process to recommend users a set of top-k views, we work at two levels. At the first level, we evaluate that how "important" is the content of the view as compared to the reference view using content driven deviation metric. At the second level, we evaluate contextually how different a view is from other views in the recommended set using context driven deviation measure.

3.3 Problem Definition

The proposed recommendation visualization systems should recommend set of views with the high importance score and maximum diversity as well as it should has high efficiency in terms of costs due to all computations are running on the fly. Hence, there will be two variables that can be used for evaluation, which are *effectiveness* and *efficiency*.

3.3.1 Effectiveness. *DiVE* scheme using hybrid approach which designed to recommend a set of k views that considering importance and diversity. In this section, we formally define the problem of the quality of recommended views.

In order to generate the top-k views that considering both importance and diversity, each view is evaluated by its reference to measure the quality of individual view and all views in the set are evaluated in terms of diversity. For instance, given an example view V_i, $V_i \in S$, set of all possible views $\mathbb{V}$, our objective is to select a set of views S $\subseteq \mathbb{V}$ where size of S = i = k, such that the importance score of the views in S and the diversity of all views in the set S is maximized. To achieve this goal, there are two components that need to be considered, the importance score of a set of views S and the diversity score of the set S. Hence, we need to combine those two components as a hybrid objective utility function.

The importance score computation. The importance score of the set S is calculated as the average value of the importance measure of each view in S, it can be defined as:

$$I\left(S\right) = \frac{\frac{1}{k}\sum_{i=1}^{k} IV_i}{I_u}, V_i \in S$$

The average of the importance score of set S needs to be normalized due to the value is not as straight forward, it depends on the real data. The normalization works by dividing the average of importance score of set S by the absolute maximum of importance score or the absolute upper bound of the importance score I_u.

The diversity score computation. There are several diferent diversity functions have been employed in the literature [3, 10, 20], among which previous research has mostly focused on measuring diversity based on either the average or the minimum of the pairwise distances between elements of set [21]. We focus on the first of those variants (i.e., average), as it considers all the views in S. Given a distance matric $D\left(V_i, V_j\right)$ as given in equation 2, the diversity of a set S can be measured by a diversity function $f\left(S, D\right)$ that captures the dissimilarity between the views in S, defined as:

$$f\left(S, D\right) = \frac{1}{k\left(k-1\right)} \sum_{i=1}^{k} \sum_{j>i}^{k} D\left(V_i, V_j\right), V_i, V_j \in S$$

There is no need to normalize the function for the diversity score of set S $f\left(S, D\right)$ due to the maximum diversity score of each pairs of views $div\left(V_i, V_j\right)$ never be more than 1 as in equation 4 and $f\left(S, D\right)$ uses $k\left(k-1\right)$ as the divisor to make sure that there will be no result that more than 1.

Hybrid objective utility function. In order to capture both importance and diversity in the set of recommended views, we define a hybrid objective function. Specifically, for a set of views S $\subseteq V$ an objective function is formulated as the linear weighted combination of the importance score, $I\left(S\right)$ and diversity function $f\left(S, D\right)$ which is defined as:

$$F\left(S\right) = \left(1-\lambda\right).I\left(S\right) + \lambda.f\left(S, D\right) \quad (5)$$

where $0 \leq \lambda \geq 1$ is employed to control the contribution between importance and diversity in the hybrid objective function. The higher values of λ result in a set of more diverse views whereas lower values of λ generate a set of the most important views that might be similar to each other. Given the hybrid objective function, our goal is to find an optimum set of views S^* that maximizes the objective function $F\left(S\right)$:

$$S^* = \underset{\substack{S \subseteq V \\ |S|=k}}{\operatorname{argmax}} F\left(S\right) \quad (6)$$

3.3.2 Efficiency. In the section before, we defined the problem in terms of quality of the results which hybrid utility function was proposed as the solution. This section defines the issue in terms of efficiency, as explained in the preliminaries section that in case of the modest data which has small number of dimensions, it can generate large number of views. Meanwhile, views that will be presented to users only in small number (top-k views) and all computation will be done on the fly. Hence, we need the scheme that robust in terms of the quality of results as well as in terms of efficiency (costs).

In fact, it has been mentioned a lot in the literatures [5, 17, 18] that the main issue of the visualization recommendation systems is the query execution. To deal with the query costs, there are several approaches that proposed, such as query results caching[10], shared computation among views, combine target and reference query, combination of multiple aggregates, combination of multiple GROUP BY, and parallel query and execution, [18, 21].

However, in this work, we propose *DiVE* scheme that leverages the propoerties of importance and diversity metrics to prune a large number of query executions without reducing the quality of the results. The detail is presented in the proposed methods section.

4 PROPOSED METHODS

As discussed in the introduction, the current view recommendations [5, 17, 18] generated solely on the basis of importance score suffer from the redundancy problem. The extreme solution to overcome the redundancy in the top-k views in the set S is to select views such that the diversity score of S is maximized.

This leads to two extreme baseline solutions, one based only on the importance score of the views which is called ***"Linear-Importance"*** and second based on only diversity score of the views which is called ***"Greedy-Diversity"***.

Instead of uses one of those two extrem, *DiVE* scheme employed hybrid function for evaluating top-k views which captures both the importance as well as diversity. Moreover, DiVE also equipped with pruning scheme which can prune a large number of queries without reducing the quality of the result. Our proposed DiVE scheme described below.

4.1 DiVE-Greedy Scheme

In order to capture both importance and diversity in the recommended top-k views, *DiVE* employs a greedy construction algorithm to iteratively select views that maximize the hybrid objective function $F\left(S\right)$ as presented in equation 3, and it is called as ***DiVE-Greedy*** scheme.

The details of the *DiVE-Greedy* scheme are given in Algorithm 1. The key ingredient of any Greedy algorithm for solving an optimization problem is the objective function itself that needs to be maximized. In particular, *DiVE-Greedy* initializes the set S with two most distant views. The distance between all the views is calculated using the distance function as given in equation 2. In each iteration, a new candidate view is selected from remaining views X and added to S. Thus, *DiVE-Greedy* assigns a utility score to each candidate view which is based on the hybrid objective function $F\left(S\right)$ as defined in equation 5. The utility score of each candidate view V_i in X is computed as:

Algorithm 1: *DiVE* Greedy

Input: Set of views V and result set size k
Output: Result set $S \geq V$, |S| = k

```
S ← [V_i, V_j] get two most distant views;
X ← [V\S];
i ← len(S);
while i < k do
    for j in set X do
        max_v ← argmaxF(X[j], S);
    end
    S.add(max_v);
    X.remove(max_v);
    i ← i + 1;
end
return S
```

$$U\left(V_i\right) = \left(1-\lambda\right).I\left(V_i\right) + \lambda.setDist\left(V_i, S\right) \tag{7}$$

$$\text{Where } setDist\left(V_i, S\right) = \frac{1}{|S|}\sum_{\substack{j=1 \\ V_j \in S}}^{|S|} D\left(V_i, V_j\right)$$

Thus, the view with highest utility score in each iteration is selected and added to S.

DiVE-Greedy cost. The costs of Greedy Construction algorithm has two components which are the query execution cost C_Q that computing the importance score of view and the diversity cost C_D that computing set distance of each view from the views already in S. The complexity of query execution cost is $O(n)$ as the content of each view is generated only once. Meanwhile, the diversity cost C_D is $O(kn)$ where k is the size of subset of views S and n is the number of all possible views.

Traditional diversification method applied approach such "process-first-diversity-next" which leads to generating all possible views and computing the importance score in advance, then the diversity is executed next. Although, greedy algorithm is very efficient, however, as the number of attributes $\mathbb{A}$, measures $\mathbb{M}$ and aggregate functions $\mathbb{F}$ increase, the number of views that need to be generated grows exponentially. Hence, without a good strategy, *DiVE-Greedy* suffers from the high costs of query execution.

Therefore, to overcome this issue, *DiVE-Greedy* employs static pruning strategy as elaborated next.

4.2 DiVE-Greedy-Static Pruning

In order to recommend a small subset of views, the importance score I of all possible views need to be computed. However, only few views are eventually recommended to the analyst and rest of the views are discarded. Clearly, this

Views in V\S	minU′	maxU′
V_1	0.23	0.65
V_2	0.19	0.21
V_3	0.25	0.85
V_4	0.15	0.70
V_5	0.20	0.23
V_6	0.21	0.24
V_7	0.22	0.91

Figure 4: Max-Min Pruning: All views which has $maxU'$ less than the maximum of $minU'$ will be pruned

approach would hinder the performance of the view recommendation systems for high dimensional datasets. Thus, motivated by the need to reduce the number of views that need to be generated, *DiVE-Greedy* employs a static pruning technique.

The proposed pruning technique is based on the observation that the utility score of each view UV_i is a weighted sum of two different measures; 1) the importance score of view IV_i and 2) diversity score of a view from S $setDist\left(V_i, S\right)$. The diversity score of a view requires only CPU computations and is faster operation. Whereas, computing the importance score of a view by comparing the target view to the reference view incurs high cost which is dominated by I/O cost of the query executions.

DiVE-Greedy applied Max-Min pruning method which has been presented in [9], by leveraging the diversity score of a view to decide whether a view query should be executed or not, and it is called ***DiVE-Greedy-Static***.

Max-Min utilize the maximum bound of importance score I_u and minimum bound of importance score $I_0 = 0$. In each iteration, the distance $setDist\left(V_i, S\right)$ between all remaining views in X to the current set is calculated. Hence, the utility score of each views is computed using the real value of distance $setDist\left(V_i, S\right)$ and the importance score I_u and $I_0 = 0$. Thus, it produces two version of utility score which are $minU'$ and $maxU'$ of each candidate views, which can be formaly defined as:

$$minU'\left(V_i\right) = \left(1-\lambda\right).I_0\left(V_i\right) + \lambda.setDist\left(V_i, S\right)$$
$$maxU'\left(V_i\right) = \left(1-\lambda\right).I_u\left(V_i\right) + \lambda.setDist\left(V_i, S\right)$$

If $maxU'$ of the candidate view less than the maximum of $minU'$, then this view definitely will be pruned, the detail example can be seen in Figure 4. This approach generates same set of recommended views as generated by the *DiVE-Greedy* without pruning. This is due to the fact that in each iteration *DiVE-Greedy* selects the view with the highest utility score to be added to S.

This Max-Min pruning approach has been mentioned has a good performance in pruning [9]. However, in that literature, the dataset have a large of points which the value of each point is very diverse to others. To the contrary, this work is quite different. The context of view only has three dimentions, while using three dimentions and computing the diversity score of the small set of view S, the utility score may not diverse as in the literature. Perhaps the performance of pruning is not same as well as in the literature.

Moreover, although Greedy algorithms have been shown to be efficient and provide good approximations to the optimal solutions [22], [20], [15]. However, since Greedy is constructive type algorithm, it contructs the set S by adding a new candidate view, there is no guarantee that the new view selected in each iteration is the best view for the objective function FS. It is because the view which has the highest utility score not necessary be the best one that improve the objective function FS (e.g: local optimum).

To overcome this Greedy limitation, in this work, we also proposed another scheme which based on swap technique. The discussion of the swap technique is elaborated next.

Algorithm 2: *DiVE* Swap

Input: Set of views V and result set size k
Output: Result set $S \geq V$, |S| = k

```
S ← Result set of only importance or only diversity;
X ← [V\S];
F_current ← 0;
improve ← True;
while improve = True do
    for i in set X do
        S′ ← S;
        for j in set S do
            if F(S′) < F(S\Sj ∪ Xi) then
                S′ ← S\j ∪ Xi ;
            end
        end
        if F(S′) > F(S) then
            S ← S′
        end
    end
    if F(S) > F_current then
        F_current ← F(S);
        improve ← True;
    else
        improve ← False;
    end
end
return S
```

4.3 DiVE-Swap Scheme

Swap is local search type algorithm and it has been known and used to maximize diversity in the literature [4, 20]. This algorithm starts with a complete initial set S, and try to achieve better result by interchanging the remaining views in X to the current set S. If the views in X is able to give better

objective function value $F(S)$, then this view will be joined to the current set and one view in the current set that has the lowest contribution to the $F(S)$ will be removed. The details of *DiVE-Swap* algorithm can be seen in Algorithm 2.

We proposed two types of Swaps which are: 1) ***DiVE-iSwap*** , the underlying behind this sheme is, it has the initial set from the result of Linear-Importance which is importance score maximized. Thus, there will be exchanging view from X to the current set and view that can improve the $F(S)$ of current set S will be considered as the member of S. 2) ***DiVE-dSwap*** which is quite similar to DiVE-iSwap, however, this scheme is initialized by results of Greedy-Diversity, which is diversity maximized.

DiVE-Swap cost. The costs of Swap algorithm is also depend on the query execution time C_Q of all possible views and the diversity computation C_D. The query cost C_Q is executed only once but the cost is high due to it needs I/O cost. However, the complexity of diversity computation C_D is $O\left(k^2\right)$ and the number of distance computation depends on the number of iterations of the swap and the number of views in X. In the worst case, swap algorithm can perform $O\left(k^n\right)$iterations.

DiVE-iSwap utilize the results of Linear-Importance as the first initialization. Hence, this algorithm cannot escape from executing all queries due to Linear-Importance needs to execute all possible views to get the results. However, the second proposed swap algorithm, *DiVE-dSwap* is initialized by the result of Greedy-Diversity. This algorithm does not execute any query to generate the results. Therefore, leveraging the properties of diversity for pruning in *DiVE-dSwap* scheme is possible. The proposed static pruning of *DiVE-dSwap* is discussed below.

4.4 DiVE-dSwap Static Pruning

As in the Greedy technique, *DiVE-dSwap* leverages the diversity score of a candidate view to decide whether a view query should be executed or not. The details of static pruning technique of *DiVE-dSwap* as following:

- *DiVE-dSwap* using initial set which is the result of Greedy-Diversity. All query views in the initial set need to be executed to get the objective function FS of the current set S.
- Before start an exchanging view from X to the current set S. All views in X is sorted based on $setDist\left(V_i, S\right)$. This is to confirm that exchanging process starts from the view that has highest score of diversity to the current set. We called this technique as *"top-1"* method.
- In order to exhanging the candidate view in X, the query of the candidate view need to be executed to get the importance score. Instead of getting the actual importance score, the maximum bound of importance score is used to compute the utility score of each view as in Greedy technique. Hence, the result is not the actual utility score but $maxU'$, which defined as: $maxU'\left(V_i\right) = \left(1-\lambda\right).I_u\left(V_i\right) + \lambda.setDist\left(V_i, S\right)$.

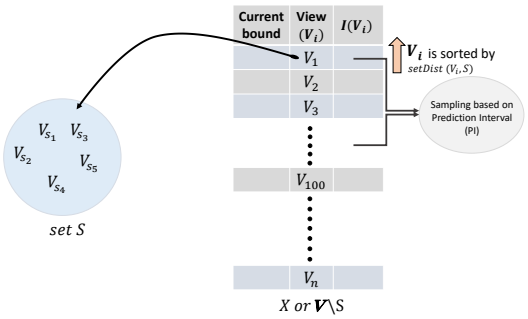


Figure 5: dSwap-Pruning: Each candidate view has importance score and *currentbound*. The importance score of view will be generated only for view which its utility score is able to improve set S while using *currentbound*

- The exchanging process is started by comparing FS of the current set to FS of new set as can be seen in Algorithm 2 lines 9 - 10.

If candidate views in X, while using importance score I_u cannot perform better in terms of improving the objective function FS to the current set S, those views will be pruned. This method is valid due to if the maximum score of importance is used and that view cannot improve FS of the current set, then there is no reason to execute the view query to get the importance score. This pruning scheme is called ***DiVE-dSwap-Static***.

All proposed pruning techniques including DiVE-Greedy-Static and DiVE-dSwap-Static are using static value I_u as the bound. Hence, the pruning performance of this technique will not optimal while the value of I_u is far away from the real maximum value of importance in database.

To overcome this issue, we proposed adaptive pruning scheme as described in the next section.

4.5 Adaptive Pruning Scheme

The idea behind pruning scheme is to minimize the query execution, which is by early prune low quality views. There are two factors that improve pruning performace: 1) weight of λ and 2) the max bound value I_u.

For instance, assume that an analyst wants to get set of views from view recommendation and she uses $\lambda = 0.7$. The λ value equal to 0.7 means that the diversity contribution to the utility score is 70% and the contribution of importance score will be 30 %, as it can be seen in equation 5. Thus, by setting the λ to higher value, the contribution of importance score will be lower and there will be more prune queries. This λ value is determined by analyst, however, in this experiement 0.5 is used as the default.

Moreover,*DiVE schemes* utilize the max bound I_u to check whether the query view need to be executed or not. In case that a view cannot improve the current set while using I_u then this view will not be executed. Hence, the optimal pruning can be achived while the maximum bound value is close to the real maximum value of importance score in

the dataset. In order to overcome this issue, instead of using static bound I_u, we also proposed adaptive pruning scheme that automatically adapts the bound to the real maximum importance score in the dataset.

The idea of adaptive pruning schemes is by setting the bound to the maximum bound I_u as in static pruning, however, this bound is changed to the real value of importance score after its query view executed. However, the problem occurs when the executed query has a small importance score and it is far from the maximum importance score in the dataset. Thus, it brings the pruning out of control. Therefore, DiVE needs the strategy to ensure that the selected importance score is as close as possible to the maximum importance score in the dataset. Hence, it needs some queries to be executed as a sampling. This brings us to the question of how many samples are needed to get a view that has a maximum score from the dataset.

There are some literature related to the sampling [cite]. The importance score of candidate views in X is not in normal distribution. The highest importance score I_u is equal to $\sqrt{2}$ whereas the lowest is 0, and it is long tail distribution. Hence, we adopt the sampling method from this [cite] as our data is not in normal distribution, it is called as prediction interval (PI). PI is similar as a confidence interval in normal distribution. The relation between PI and the number of samples defined as in equation 8.

$$PI = \frac{\left(N-1\right)}{\left(N+1\right)}, where\, N = Number\, of\, samples \quad (8)$$

When analyst uses PI = 80%, it means there are 9 sample views executed, 85 %, 90%, 95%, 97%, and 99% means 12, 20, 40, 60, and 200 samples executed respectively.

The details of our adaptive pruning scheme described as follows:

- As in *DiVE-dSwap-Static* that all query views in the initial set needs to be executed to get the objective function FS of the current set S and all candidate views in X is sorted based $setDist\left(V_i, S\right)$ which called as *"top-1"* technique.
- All query of the candidate views need to be executed to get the importance score, however, Before start an exchanging view from X to the current set S. All views in X will be sorted based on $setDist\left(V_i, S\right)$. This is to ensure that exchanging process starts from the view that has highest score of diversity to the current set. We called this technique as *"top-1"* method.
- In order to exhanging the candidate view in X, the query of the candidate view need to be executed to get the importance score. Instead of getting the actual importance score, the maximum bound of importance score is used to compute the utility score of each view as in Greedy technique. Hence, the result is not the actual utility score but $maxU'$, which defined as: $maxU'\left(V_i\right) = \left(1-\lambda\right).I_u\left(V_i\right) + \lambda.setDist\left(V_i, S\right)$.

Table 2: Parameters testbed in the experiments

Parameter	Range (**default**)
datasets	**Heart disease**, Flights
sample queries	**10**
diversity weight ratio	**3**(A) **: 2**(M) **: 1**(F)
tradeoff weight λ	0.0, 0.2, 0.4, **0.5**, 0.6, 0.8, 1.0
result set (size of k)	**5**, 15, 25, 35
prediction interval %	80 , 85, 90, 95, **97**, 98

- The exchanging process is started by comparing FS of the current set to FS of new set as can be seen in Algorithm 2 lines 9 - 10.

Adaptive pruning in Greedy is called *DiVE-Greedy-Adaptive* wheras in Swap is called *DiVE-dSwap-Adaptive*.

5 EXPERIMENTAL EVALUATION

Experiment Setup. This experiment running on Windows 10 64 bit, Intel Core i7-7700 CPU @ 3.60 GHz, RAM 16 GB. The experiment is developed using Python and run on Python version 3.6.3 with PostgreSQL as the database engine. Two real datasets are used, as follows:

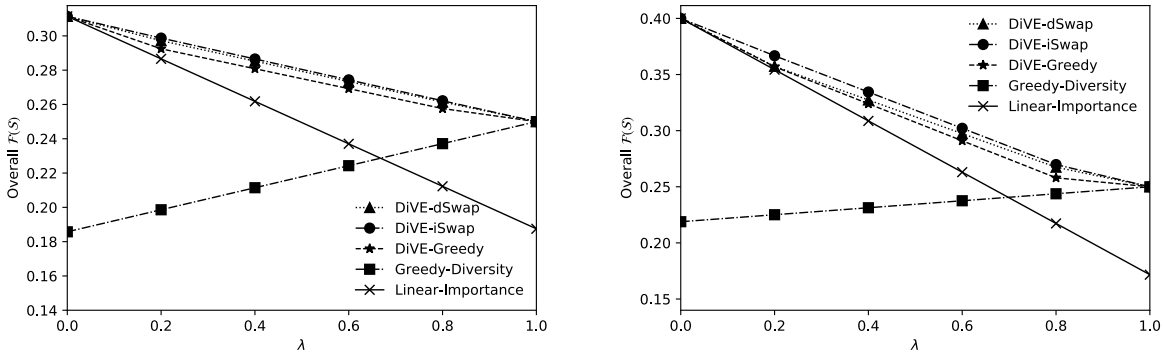
- Heart Disease Dataset [2], it has 14 attributes in total which are 9 attributes $\mathbb{A}$ and 5 measure attributes $\mathbb{M}$, $\mathbb{A}$= sex, cp, fbs, restecg, exang, slope, ca, thal, num; $\mathbb{M}$ = age, trestbps, chol, thalach, oldpeak. This dataset has small number of rows (299 rows).
- Airline (Flights) Dataset [3], it has 7 attributes, 4 measure attributes. The detail as follows: $\mathbb{A}$= year, month, week, day, carrier, origin, destination; $\mathbb{M}$= arrivaldelay, departurdelay, weatherdelay, distance. This dataset has 855,632 number of rows.

5.1 Effectiveness Evaluation

In order to test the performance of our proposed approach, we run experiments using two real datasets: flights dataset and heart disease dataset. For each dataset, we execute ten random queries. Afterwards, the average of all results is calculated as the final result. We examine the performance of our proposed schemes in term of the quality of the result (effectiveness) and the efficiency. All parameters used in this experiments are summarized in Table 2 and the detail results of the experiments is elaborated next.

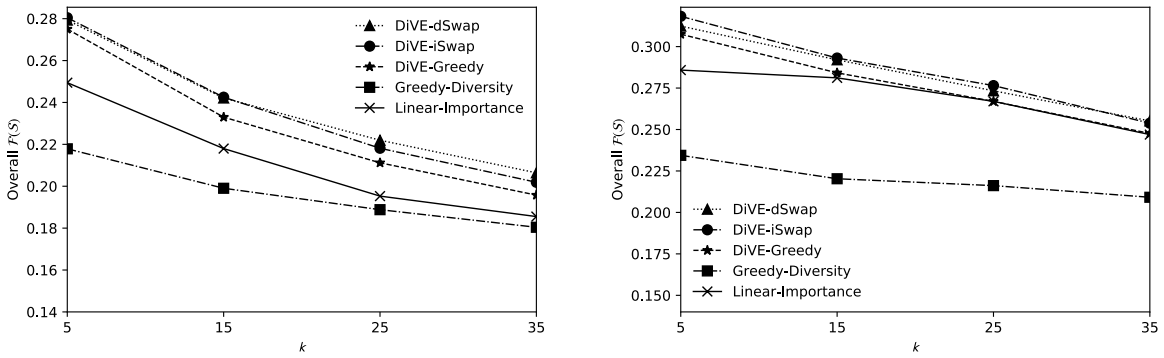
The impact of parameter weight λ to the overall $F\left(S\right)$. One of the advantage *DiVE* scheme is users can decide what kind of the results that they want by changing the value of weight λ. The main function of λ is this weight can be used to tradeoff between importance and diversity, it is explained in equation 5. Figure 6 shows the impact of λ to the overall objective values $F\left(S\right)$ by fixed k equal to 5, where $0 \leq \lambda \geq 1$. The figure shows that in the beginning when the λ is close to 0, means the role is dominated by the importance score,

[2] http://archive.ics.uci.edu/ml/datasets/heart+Disease
[3] http://stat-computing.org/dataexpo/2009/the-data.html



(a) Heart disease dataset **(b) Flights dataset**

Figure 6: Impact of λ to overall objective function value $F\left(S\right)$ while k = 5 and running on three real datasets



(a) Heart disease dataset **(b) Flights datasets**

Figure 7: Overall objective function value $F\left(S\right)$ using different value of k and running on three real datasets

Linear-Importance has high overall objective function value $F\left(S\right)$ but it will decline gradually while increasing of λ. To the contrary, the Greedy-Diversity is the opposite, it has the high overall objective function value when the λ equal to 1, which means the diversity gets the full role. Hence, there is a crossover between Linear-Importance and Greedy-Diversity, this crossover shows the role balancing between importance score and diversity score. Futhermore, our proposed schemes have stable performance in all various λ value and its $F\left(S\right)$ value always better than Linear-Importance and Greedy-Diversity.

The impact of k to the overall objective function value $F\left(S\right)$. It has been explained the impact of λ to the overall objective function value $F\left(S\right)$.This section discuss the overall objective function value $F\left(S\right)$ by increasing k. Figure 7 shows the overall objective values $F\left(S\right)$ by using different of k. As shown in those figures, the value of $F\left(S\right)$ decrease while increasing the value of k. There are two reasons why this moment happens: 1) the importance score of the set $I(S)$ is generated by sum of all importance score of views and it is divided by the number of views=k, increasing the number of k means increasing the divisor. 2) the diversity score of a set of views $f(S, D)$ also depends on the number of k. While k is increasing, the diversity score of the set will be decreased, due to the score of $f(S, D)$ is yielded from the sum of all distance of views divided by $k * k - 1$. Hence, the more views selected, the probability to get similar views will be high and the score of diversity will be declined.

The most important point in this results is our $DiVE$ schemes always have better overall objective function value $F\left(S\right)$ compared to the two extreme baselines in various number of k.

5.2 Efficiency and Pruning Scheme Evaluation

The total time to run DiVE schemes. In order to start analyzing the efficiency, we need to know the main issue in term of costs. Figure 8 shows the example of exactly time that needed

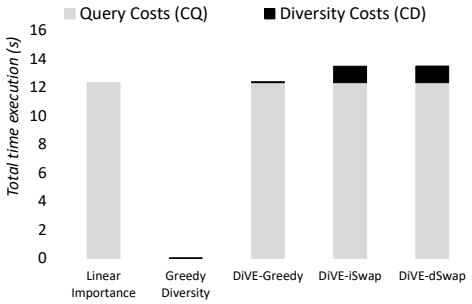


Figure 8: Total time (seconds) to execute schemes on flights dataset using k =5 and $\lambda = 0.5$. It shows that costs are dominated by query costs C_Q

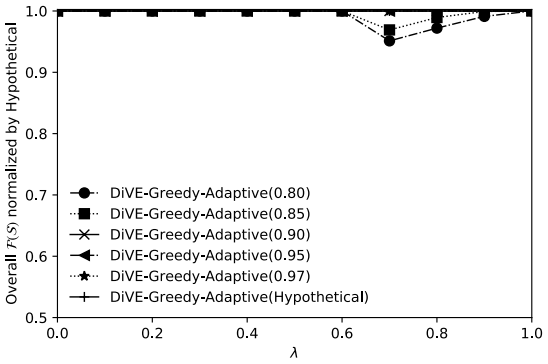


Figure 9: Impact of λ to the performance of *DiVE-Greedy-Adaptive-Pruning* with different of PI in terms of the effectiveness, running on Heart disease dataset using k = 5. Using PI 80 and PI 85 can reduce the quality of the result while the λ value is low

to run schemes on flights dataset. It shows Greedy-Diversity which only considering diversity and no query executions, it has very low costs. Meanwhile, Linear-Importance and *DiVE-Greedy* seems in the same line but that was not exactly same. The total of diversity computations C_D of *DiVE-Greedy* is very low, the total costs of *DiVE-Greedy* is dominated by query costs C_Q. Due to of this reason, the total execution time of *DiVE-Greedy* closes to Linear-Importance. Those are the proof that the total costs of all schemes execpt Greedy-Diversity are dominated by query cost C_Q.

Due to the page limitation, all our experiments in three real dataset cannot be showed. Hence, for the next sections, we use heart disease dataset as our focus observation.

Impact of λ to the pruned queries of DiVE-Greedy scheme. In this work, we proposed two kind of pruning schemes, that are using estimated static value of maximum value of the

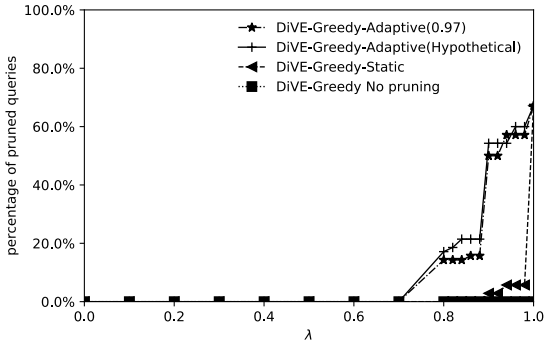


Figure 10: Impact of λ to the pruned queries of *DiVE-Greedy* scheme, running on Heart disease dataset using k = 5. Using PI-97 able to prune queries around 20 percent while λ higher than 0.8

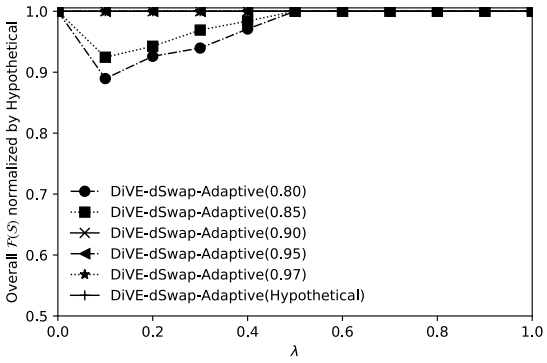


Figure 11: Impact of λ to the performance of *DiVE-dSwap-Adaptive-Pruning* with different of PI in terms of the effectiveness, running on Heart disease dataset using k = 5. Using PI 80 and PI 85 can reduce the quality of the result while the λ value is low

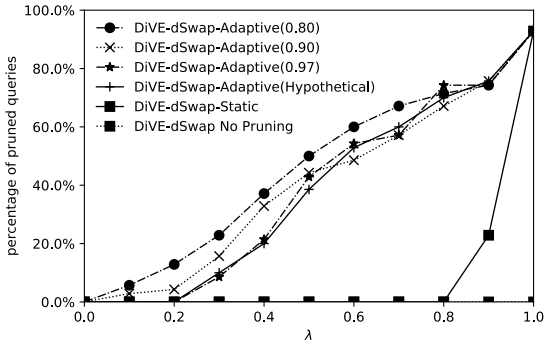


Figure 12: Impact of λ to the pruned queries of *DiVE-dSwap* scheme, running on Heart disease dataset using k = 5. Adaptive pruning schame able to prune queries start from λ 0.1, it has more pruned queries while λ increased

importance score $maxI$ and using adaptive maximum value of importance score.

The first our pruning scheme is using static value $maxI$. To check the performance of our pruning scheme, we run it to all real datasets using different value of λ. We analyze the result by comparing the result of schemes with pruning enable on it and the schemes without pruning enable on it. The example result which running on heart disease dataset is shown in Figure 10 . The static pruning scheme only able to prune queries while the value of λ is high, closes to 0.9. As we expected, it is because the value of $maxI$ that too far from the real value of importance score.

To overcome the flaw in static pruning scheme, we also proposed adaptive pruning scheme which used dynamic value of the maximum importance score $maxI$. In order to get $maxI$ as close as possible to the real value of importance in the dataset, we applied sampling method. By using adaptive pruning, users also able to change the confidence and the margine error to tradeoff between time and precision.

Impact of λ to the pruned queries of DiVE-dSwap scheme. If static pruning scheme only able to prune queries while λ closes to 0.9 and higher, in this section, we shows the adaptive pruning performance. Figure 12 shows the performance of adaptive pruning scheme by using different value of λ. It shows the impact of λ to the percentage of pruned queries. The adaptive pruning schemes especially *DiVE-dSwap-Adaptive-Pruning* is able to prune queries significantly, pruning start while λ closes to 0.2 and by increasing the λ, more queries can be pruned.

The impact of query load to the pruning performance. As shown in Figure 7, that the actual value of importance score affects to the overall value of objective function $F\left(S\right)$ and the shape of the graphs. The next question is do the actual value of importance score also affects the pruning performance while using adaptive pruning scheme? We did experiments using heart disease dataset to answer this question. We lists all subsets in the heart disease dataset and classify those subsets to three categories: low, middle, and high. Low category consist of subsets that have mostly low value of importance score, high category is the opposite, and the middle category is the middle of low and high. We compare the result in terms of pruning performance using input low, middle, and high categories. The result can be seen in Figure 13. It has different result while using low category and high category. Input using low category or low query load has more pruned queries compared to high importance query load (high category).

6 CONCLUSIONS

In this paper, we proposed *DiVE* scheme which the main purposes are to evaluates and optimizes the results of visualization recommendation systems with respect to importance and diversity. The advantage of *DiVE* is that analyst can set their preferences by changing the parameter to tradeoff between importance and diversity to get result set. We also performed an experimental study and present the results which focus on effectiveness and efficiency of our approach

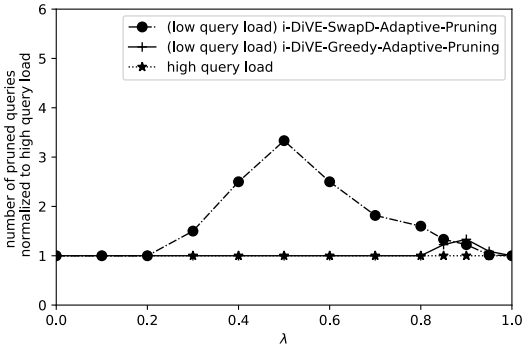


Figure 13: Number of pruned queries of high and low query load normalized by high importance query load using different value of λ, k = 5 and running on Heart disease dataset

on real datasets. We proposed *DiVE* scheme which based on Greedy and Swap approach, *DiVE-iSwap* have the best performance in recommending result views but it has the highest costs due to this scheme executing all possible view from the dataset, this scheme can be used for the analyst who only cares about the results without worrying execution time. However, to the analyst who care about execution time, we proposed *DiVE-dSwap-Adaptive* and *DiVE-Greedy-Adaptive*, those schemes are able to decrease costs significantly without reducing the quality of results.

ACKNOWLEDGMENTS

This research is supported by the Indonesia Endowment Fund for Education (LPDP) as scholarship provider from the Ministry of Finance, Republic of Indonesia.

REFERENCES

[1] E. Baikousi, G. Rogkakos, and P. Vassiliadis. [n. d.]. Similarity measures for multidimensional data. *ICDE 2011* ([n. d.]).
[2] S. Choi, S. Cha, and C. C. Tappert. 2010. A survey of binary similarity and distance measures. *Journal of Systemics, Cybernetics and Informatics* (2010).
[3] Charles L.A et al. Clarke. 2008. Novelty and diversity in information retrieval evaluation. *SIGIR* (2008).
[4] M Drosou and E Pitoura. 2010. Search Result Diversification. *SIGMOD Record* (2010).
[5] H Ehsan, M. Sharaf, and Panos K. Chrysanthis. 2016. MuVE: Efficient Multi-Objective View Recommendation for Visual Data Exploration. *ICDE* (2016).
[6] Sean Kandel, Ravi Parikh, Andreas Paepcke, Joseph M Hellerstein, and Jeffrey Heer. 2012. Profiler: Integrated statistical analysis and visualization for data quality assessment. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*. ACM, 547–554.
[7] Alicia Key, Bill Howe, Daniel Perry, and Cecilia R. Aragon. 2012. VizDeck: self-organizing dashboards for visual analytics. *SIGMOD Conference* (2012).
[8] Alicia Key, Bill Howe, Daniel Perry, and Cecilia R. Aragon. 2012. VizDeck: self-organizing dashboards for visual analytics. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2012, Scottsdale, AZ, USA, May 20-24, 2012*. 681–684. https://doi.org/10.1145/2213836.2213931
[9] Hina A. Khan and Mohamed A. Sharaf. 2015. Progressive diversification for column-based data exploration platforms. *ICDE* (2015).

[10] Hina A Khan, Mohamed A Sharaf, and Abdullah Albarrak. 2014. DivIDE: Efficient Diversification for Interactive Data Exploration. *SSDBM* (2014).

[11] Davood Rafiei, Krishna Bharat, and Anand Shukla. 2010. Diversifying web search results. *WWW* (2010).

[12] Thibault Sellam and Martin L. Kersten. 2016. Fast, Explainable View Detection to Characterize Exploration Queries. In *Proceedings of the 28th International Conference on Scientific and Statistical Database Management, SSDBM 2016, Budapest, Hungary, July 18-20, 2016*. 20:1–20:12. https://doi.org/10.1145/2949689.2949692

[13] Thibault Sellam and Martin L. Kersten. 2016. Ziggy: Characterizing Query Results for Data Explorers. *PVLDB* 9, 13 (2016), 1473–1476. http://www.vldb.org/pvldb/vol9/p1473-sellam.pdf

[14] Jinwook Seo and Ben Shneiderman. 2006. Knowledge Discovery in High-Dimensional Data: Case Studies and a User Survey for the Rank-by-Feature Framework. *IEEE Trans. Vis. Comput. Graph.* 12, 3 (2006), 311–322. https://doi.org/10.1109/TVCG.2006.50

[15] Barry Smyth and Paul Mcclave. 2001. Similarity vs . Diversity. (2001).

[16] Manasi Vartak, Silu Huang, Tarique Siddiqui, Samuel Madden, and Aditya Parameswaran. 2017. Towards Visualization Recommendation Systems. *ACM SIGMOD Record* (2017).

[17] Manasi Vartak and Samuel Madden. 2014. S EE DB : Automatically Generating Query Visualizations. *VLDB* (2014).

[18] M Vartak, S Rahman, S Madden, A Parameswaran, and N Polyzotis. 2015. SEEDB : Efficient Data-Driven Visualization Recommendations to Support Visual Analytics. *VLDB* 8 (2015).

[19] Fernanda B. Viegas, Martin Wattenberg, Frank Van Ham, Jesse Kriss, and Matt McKeon. 2007. Many Eyes: A site for visualization at internet scale. *IEEE Transactions on Visualization and Computer Graphics* (2007), 1121–1128.

[20] Marcos R. et al. Vieira. 2011. On query result diversification. *ICDE* (2011).

[21] Eugene Wu, Leilani Battle, and Samuel R. Madden. 2014. The case for data visualization management systems. *VLDB Endowment* (2014).

[22] Cong Yu, Laks Lakshmanan, and Sihem Amer-Yahia. 2009. It takes variety to make a world: diversification in recommender systems. *EDBT* (2009).

[23] Mi Zhang and Neil Hurley. 2008. Avoiding Monotony: Improving the Diversity of Recommendation Lists. *ACM Conference on Recommender Systems* (2008).