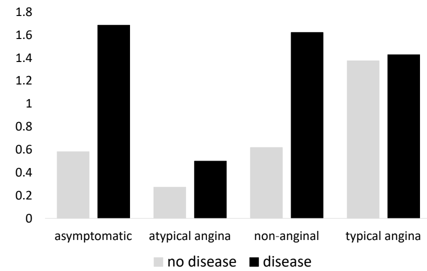
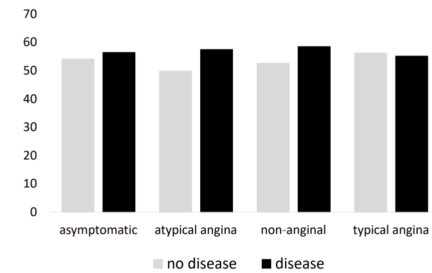
DiVE: Diversifiying View Recommendation for Visual Data Exploration

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.



(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 people with little domain knowledge and technical expertise, looking for interesting trends in data.

Generaly, without any prior knowledge about the data, she must manually specify different combinations of attributes, measures and aggregate functions before finally generating a view that reveals some insights from the dataset. However, manually looking for insights in each view is a labor-intensive and time-consuming process.

Hence, several data-driven visualization recommendation have been proposed that can automatically recommend data visualizations that reveal interesting and useful insights [3, 15, 16]. The main goal of those works is to provide the user with the most important visualizations (*top-k views*). The top-k views are selected according to some metric of importance (a deviation-based metric). This metric determines the deviation between the queried subset of data (*target view*) to the reference subset of data (*reference view*). The intuition behind deviation-based approach is that views that reveal substantially different trends from the reference view is judged as the important view [15, 16].

CIKM 2018, October 2018,

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

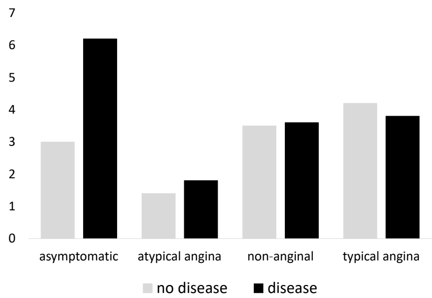


Figure 2: The visualization of maximum oldpeak vs. chest pain types

For instance, consider a Cleveland heart disease dataset [1], which describes patients with and without a heart disease. A data analyst might be interested in conducting some comparison between people with heart disease (**disease**) and people without heart disease (**no disease**). 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 *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). To the contrary, Figure 1b is potentially less important view compared to Figure 1a, even there is a deviation but the deviation is very small and it is lower than Figure 1a.

Although the deviation-based visualization recommendation automatically provide the top-k most important views, however, it often recommends similar views and leaving the analyst with a limited amount of gained insights. For instance, Figure 2 provides similar insights to Figure 1a, both figures show that people with heart disease tend to have higher oldpeak values. Figure 2 is generated using *chest pain types* as the attribute, *oldpeak* as the measure and AVG as the aggregate function, whereas Figure 1a, uses same attribute and measure but uses MAX as the aggregate function. Since both views have a high deviation from the reference view, both views are considered as the important views and those will appear in the top-k set. This leads to an important observation that using "only importance" as the only criterion (e.g. a deviation-based metric) is that often deliver redundant recommended views, which leads to presents limited insights of results.

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 concise coverage of the possible insights to be discovered. In fact, novelty and diversity are one of the fundamental characteristics of any effective recommendation systems [1, 9, 20, 21]. Specifically, it is highly desirable that a view recommendation can recommend the top-k views that are both importance and also provide new insights that has not been revealed by the other views.

Towards designing an effective view recommendation that promotes both importance and diversity in recommended views, in this work, we propose an integrated approach called *DiVE*. In particular, *DiVE* aims to generate top-k views 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 (**Section 3**).
- We define a similarity measure to capture the distance between two visualizations (**Section 3**).
- We present a hybrid objective function to balance the trade-off between importance and diversity when ranking the visualizations (**Section 3**).
- We propose the novel *DiVE* schemes, that employs various algorithms to evaluate the recommended visualizations based on the hybrid ranking/objective function (**Section 4**).
- We present optimization techniques that leverage the hybrid objective function to substantially reduce the computational costs (**Section 4**).
- 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 (**Section 5**).

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., [3, 15, 16]). 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 some 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 [3, 7, 15–17], 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 specified 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.

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

In that example, let D_B be the Cleveland Heart Disease 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 capture the utility of a recommended view (e.g., [3, 4, 7, 10–12, 15–17]. next sentences need 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. Can come from your old related work section or from Humaira's TKDE Some of those works focus on recommending visualizations to facilitate a particular user intent or task. For example: using user feedback as a basis for view recommendation [7], recommending interactive visualizations on the webpage and engage user for collaboration and discussion [17], explanations for a certain behavior, finding data anomalies or outliers and correlations among data attributes [4, 12]. 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 *important* visualizations that highlight some of the particular trends of the analyzed datasets [10, 11, 15, 16].

In particular, the deviation-based metric measures the distance between $V_i(D_Q)$ and $V_i(D_R)$. That is, it measures the deviation between the aggregate view V_i generated from the subset data D_Q vs. that generated from a reference dataset D_R, where $V_i(D_Q)$ is denoted as *target* view, whereas $V_i(D_R)$ is denoted as *reference* view. That reference dataset could be the whole database (i.e., $D_R = D_B$) or a selected subset of the database. The premise underlying the deviation-based metric is that a view V_i that results in a high deviation is expected to reveal some important insights that are very particular to the subset D_Q and distinguish it from the patterns in D_R. In case, $D_R = D_B$, then the patterns extracted from D_Q are fundamentally different from the generals ones manifested in the entire database 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. refer back to the intro example and reiterate that issue in one sentence The example has been shown in Section 1, two views which generated by same attribute and meassure and different aggregate function, both of them are considered as important views, however, those are similar and delivering 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. In the next section, we discuss in details the formulation of both importance and diversity, and their impact on the view recommendation process.

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

3 DIVERSIFYING RECOMMENDED VISUALIZATIONS

short preamble

In order to diversifying recommended visualizations, we work at two levels. At the first level, content driven deviation metric is used to evaluate that how "important" is the content of the view as compared to the reference view. At the second level, we evaluate contextually how different a view is from other views in the recommended set using context driven deviation measure.

3.1 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 [15, 16]. Essentially, the deviation-based metric compares an aggregate view generated from the selected subset dataset D_Q (i.e., target view $V_i(D_Q)$) to the same view if generated from a reference dataset D_R (i.e., reference view $V_i(D_R)$).

Clearly, the deviation between a target and a reference view is a *data-driven* metric. That is, it measures the deviation between the aggregate *result* of $V_i(D_Q)$ and that of $V_i(D_R)$. Consequently, and from a visualization point of view, that deviation is a *content-based* metric that captures the difference between the content of the visualization generated by $V_i(D_Q)$ vs. the visual content generated from $V_i(D_R)$. 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 the content-based deviation, each target view $V_i(D_Q)$ is normalized into a *probability distribution* $P[V_i(D_Q)]$ and similarly, each reference view into $P[V_i(D_R)]$. In particular, consider

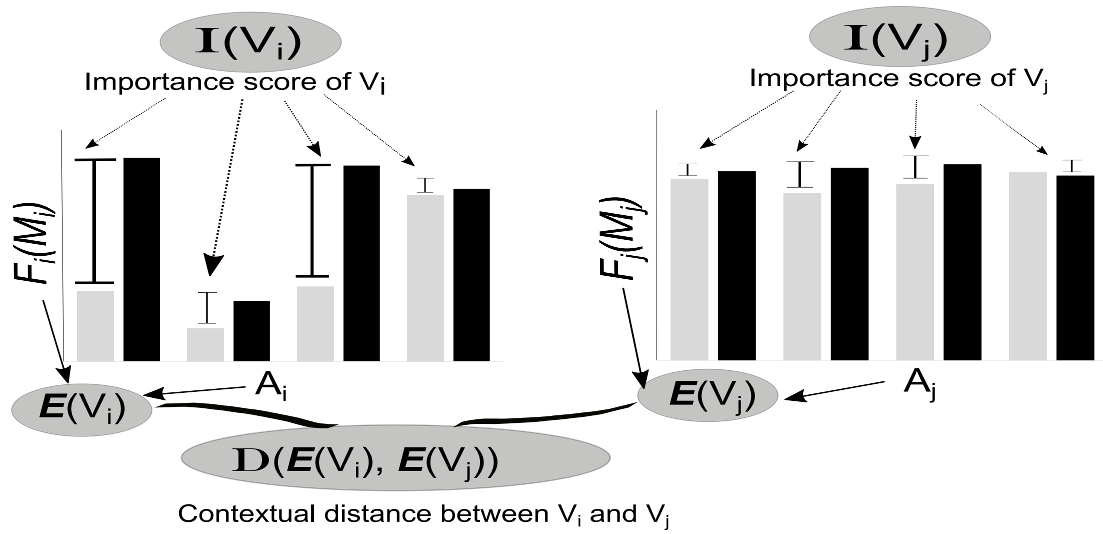


Figure 3: Content vs. Context of views.

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 $F_i(M_i)$ for the group a_j [3, 15]. Hence, V is normalized by the sum of aggregate values $G = \sum_{j=1}^{t} g_j$, resulting in the probability distribution $P[V_i] = < \frac{g_1}{G}, \frac{g_2}{G}, ..., \frac{g_t}{G} >$.

Finally, the importance score of V_i is measured in terms of the distance between $P[V_i(D_Q)]$ and $P[V_i(D_R)]$, and is simply defined as:

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

where $I(V_i)$ is the importance score of V_i and *dist* is a distance function. Similar to existing work (e.g., [15, 16]), we adopt a Euclidian distance function, 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 $I(V_i)$ of each possible view V_i is computed, and the k views with the highest deviation are recommended (i.e., *top-k*) (e.g., [15]). 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.

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 full 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., [9, 18, 20, 21]). 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. 3.3, 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., [1, 9, 21]), database tuples (e,g„ [14]), etc. However, our work in this paper is the first to consider diversity in the context of aggregate data visualizations. As such, a metric is needed to capture and quantify the (dis)similarity between the distinct features of different visualizations. Meanwhile, as each visualization is merely a data view generated by an SQL aggregate query, such metric naturally lends itself to considering the query underlying each view. 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., [5, 6, 14]). 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. Hence, in addition to the content of a view V_i which is described by its probability distribution (i.e., $P(V_i)$) as defined in Sec 3.1), we also consider the context of the view $E(V_i)$, which is defined in terms of the query underlying V_i as: $E(V_i) = [A_i, M_i, F_i]$.

Such definition of view context leads to a special case of the existing work on query recommendation (e.g., [5, 6, 14]), 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:

$J(V_i, V_j) = \frac{|E(V_i) \cap E(V_j)|}{|E(V_i) \cup E(V_j)|}$

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 the same attribute and different measure and aggregate function will have the 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 $E(V_i)$, where $\sum_{i=1}^{3} w_i = 1$. Then, the similarity between views V_i and V_j is measured as: $J(V_i, V_j) = \frac{\sum_{i \in V_i \cap V_j} w_i}{\sum_{i \in V_i \cup V_j} w_i}$

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

$$D\left(V_i, V_j\right) = 1 - J\left(V_i, V_j\right) \quad (2)$$

something needs to be said about the figure! and quick summarized comparison of the two equations/metrics

As a summarize, the difference beteween content and context of view is described in Figure 3. Content is the probability distribution of the aggregated query result, whereas, Context is described as a set containing the name of the attribute, measure and function used to generate the view.

3.3 Problem Definition

In this section, we formally define our problem for recommending diversified interesting aggregate views. Towards this, we first define the metrics to measure the performance of our proposed visualization recommendation system in terms of: 1) the quality of recommended visualizations, and 2) the processing cost incurred in computing those visualizations.

3.3.1 Hybrid Objective Function. Our hybrid objective function is designed to consider both the importance and diversity of the recommended views. Particularly, it integrates two components: 1) the total importance score of set S and 2) the diversity score of S.

The importance score of the a S is calculated as the average value of the importance measure of each view in S, as given in Eq.1. Hence, the total importance score of S is defined as:

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

where I_u is the upper bound on the importance score for an individual view. The value of I_u is used to normalize the average importance score for set S.

In order to measure the diversity of a set of objects, several diversity functions have been employed in the literature [1, 18]. Among those, previous research has mostly focused on measuring diversity based on either the average or the minimum of the pairwise distances between the elements of a set [19]. In this work, we focus on the first of those variants (i.e., average), as it maximizes the coverage of S. Hence, given a distance metric $D\left(V_i, V_j\right)$, as given in equation 2, the diversity of a set S can be simply measured as follows:

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

Since the maximum context-based deviation between any two views in equation 2 is 1.0, then dividing the sum of distances by $k(k-1)$ ensures that the diversity score of set S is normalized and bounded by 1.0.

Next, we define our proposed hybrid objective function that captures both the importance and diversity of the set of recommended views S. Specifically, for a set of views $S \subseteq V$, our hybrid objective function is formulated as the linear weighted combination of the importance score, $I(S)$ and diversity score $f(S, D)$, and is defined as:

$$F(S) = (1 - \lambda) I(S) + \lambda f(S, D) \quad (3)$$

where $0 \leq \lambda \geq 1$ is employed to control the preference given to each of the importance and diversity components. For instance, a higher value of λ results in a set of more diverse views, whereas a lower value of λ generates a set of the most important views, which is likely to exhibit redundancy in the recommended views.

Given the hybrid objective function, our goal is to find an optimum set of views S^* that maximizes the objective function $F(S)$, which is defined as follows:

Definition 1. **Recommending diversified important views :** *Given a target subset D_Q and a reference subset D_R, the goal is to recommend a set $S \subseteq \mathbb{V}$, where $|S| = k$, and $\mathbb{V}$ is the set of all possible target views, such that the overall hybrid objective $F(S)$ is maximized.*

Given the definition above, the quality of the recommended set of views is measured in terms of the value of the hybrid objective function $F(S)$.

$$S^* = \underset{\substack{S \subseteq V \\ |S|=k}}{\operatorname{argmax}} F(S) \quad (4)$$

3.3.2 Cost of Visualization Recommendation. Exisiting research has shown that recommending aggregate data visualizations based on data-driven content-based deviation is a computationally expensive task [3, 15, 16]. Moreover, integrating diversification to the view recommendation problem, as described above, further increases that computational cost. In particular, the incurred processing cost includes the following two components:

(1) Query processing cost C_Q: measured in terms of the time needed to execute and compare all the queries underlying the set of target views as well as their corresponding reference views (i.e., content-based deviation).
(2) View diversification cost C_D: measured in terms of the time needed to compute all the pairwise distances between each pair of target views (i.e., context-based deviation).

Consequently, the total cost C_T for recommending a set of views is simple defined as:

$$Total\,Cost(C_T) = Query\,Cost(C_Q) + Diversity\,Cost(C_D)$$

In principle, traditional data diversification methods that consider both relevance and diversity can be directly applied in the context of our problem to maximize the overall utility function formulated in Eq.3. For instance, in the context of recommending web search, such methods are designed to recommend a set of diversified objects (e.g., web documents) that are relevant to the user needs. However, in that setting, the relevance of an object is either given or simply computed. To the contrary, in our setting for view recommendation, the importance of a view is a computational expensive operation, which requires the execution of a target and reference view. As such, directly applying those 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, in the next section we propose an integrated scheme called *DiVE*, leverages the properties of both the importance and diversity to prune a large number of a large number of low-utility views without compromising the quality of recommendations, as described next.

4 THE DIVE SCHEMES

new preamble As discussed in the introduction, the current view recommendations [3, 15, 16] 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.

4.1 Baseline Solutions

As baseline solutions to compare the performance of our proposed DiVE schemes, we simply incorporate methods from existing work that optimize either for importance or diversity. In terms of diversity, we employ the classical *Greedy Construction* algorithm [13], which has been shown to maximize diversity within reasonable bounds compared to the optimal solution [18, 20]. In this work, we refer to that baseline as *Greedy-Diversity*. Similarly, in terms of importance, we adopt the work on SeeDB for recommending the top-k views with the highest deviation [15, 16]. Particularly, in that method, all possible target and reference views are generated by executing their underlying queries, then the list of views is linearly scanned to recommend the top-k for which the target view shows high deviation from its corresponding reference view (denoted as *Linear-Importance* in this work). Clearly, those two methods are "oblivious" to our hybrid objective function (i.e., Eq.3). In particular, as shown in our experimental evaluation (Sec. 6), each of those two methods performs well under extreme settings of our hybrid function. As expected, Greedy-Diversity provides its best performance when $\lambda = 1.0$ (i.e., all preference is given to diversity), whereas Linear-Importance is the winner when $\lambda = 0.0$ (i.e., all preference is given to importance). In the following, we present our DiVE schemes which are able to provide the best performance (i.e., maximize the overall hybrid objective), irrespective of the value of λ, while minimizing the processing time.

4.2 The DiVE-Greedy Scheme

Algorithm 1: *DiVE* Greedy

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

$S \leftarrow \left[V_i, V_j\right]$ get two most distant views;
$X \leftarrow [V \backslash S]$;
$i \leftarrow len\,(S)$;
while $|S| < k$ **do**
    $v_i \leftarrow argmax\,(1-\lambda)\,I\,(v_i) + \lambda * setDist\,(v_i, S)$;
    $S.add\,(v_i)$;
    $X.remove\,(v_i)$;
end
return S

In this section, we discuss our first DiVE scheme (*DiVE-Greedy*), which simply extends the basic Greedy Construction algorithm to work under our hybrid objective function (i.e., Eq. 3). Such extension is straightforward and is described in Algorithm 1. Similar to the classical Greedy Construction, *DiVE-Greedy* initializes the set S with the two most distant views, where the distance between any two views is calculated using our context-based function, as given in Eq.2. Then, *DiVE-Greedy* iteratively selects new views to be added to S. Particularly, in each iteration a view is selected from the set of remaining views X and is added to S. To make that selection, *DiVE-Greedy* assigns a score to each view in X, which is based on the hybrid objective function $F\,(S)$, as defined in equation 3. Specifically, the utility score assigned to a view $V_i \in X$ is computed as:

$$U\,(V_i) = (1-\lambda)\,I\,(V_i) + \lambda setDist\,(V_i, S) \tag{5}$$

where

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

Thus, in each iteration, the view with highest utility score is selected and added to S, until $|S| = k$, as shown in Algorithm 1. ,

DiVE-Greedy Cost: We note that the only difference between our DiVE-Greedy scheme and our baseline Greedy-Diversity (i.e., the classical Greedy Construction algorithm) is in the utility score assigned to each view (i.e., $U(V_i)$ in Eq.5). In fact, in the special case where $\lambda = 1.0$, Eq. 5 boils down to $U\,(V_i) = setDist\,(V_i, S)$, which is the same score used by Greedy-Diversity for maximizing diversification. However, that simple change in the utility score leads to executing the query underlying each view V_i in order to compute the $(1-\lambda) \times I\,(V_i)$ component of its score. Hence, the overall cost of DiVE-Greedy is $C_T = C_Q + C_D$, as opposed to the cost of Greedy-Diversity, which is only $C_T = C_D$, where C_Q is the query processing cost (i.e., data-driven), and C_D is the cost for computing Jaccard distances (i.e., query-driven), as described in Sec. 3.

Clearly, C_Q is equal to the number of possible views and is $O(n)$, where n is the number of possible views. do we have that symbol n? No, should we add it into table of simbol? or just explaine here like this is enough?, whereas C_D is $O(kn)$, where k is the number of recommended views. Hence, as presented, our extended DiVE-Greedy is an instance of what has been described earlier as a "process-first-diversity-next" approach [], in which diversification only takes place after all possible views are executed to calculate their importance. In the next section, we describe our work to overcome the limitations of that approach.

4.3 DiVE-Greedy-Static Pruning

As mentioned in the previous section, the cost of the DiVE-Greedy algorithm is dominated by the query processing cost C_Q and is proportional to the number of possible views. Although hundreds of views are generated for a given subset of data D_Q, only a small fraction of those views are actually of interest and are candidates to be included in the top-k set. As such, a significant fraction of the query processing cost is incurred in evaluating low-utility views. This observation motivated us to propose a pruning technique to minimize the search space of views and narrow it down to the most promising ones, as follows.

As mentioned in the previous section, the cost of the DiVE-Greedy algorithm is dominated by the cost of executing view queries and is proportional to the number of possible views. Although hundreds of views are possible for a given subset of data, only a small fraction of the views are actually of interest and are included in the top-k set. Consequently, a significant fraction of the query processing cost is incurred on evaluating low-utility views. Hence, in this section we propose a pruning technique to reduce the search space of views.

Our proposed pruning technique is based on the observation that the utility score of each view $U(V_i)$ is a weighted sum of two measures; 1) the importance score of the view (i.e., $I(V_i)$), and 2) the distance of the view from S (i.e., $setDist\,(V_i, S)$). We note that the computation of $setDist\,(V_i, S)$ is a CPU-bound requires fast operation. To the contrary, computing the importance score of a view $I(V_i)$is an expensive operation that requires executing two queries to generate the target and reference data for V_i. is max-min something we proposed or we borrowed from somewhere else? and is max-min a technical term? Thus, we employ a *max-min* pruning technique, that leverages the diversity score to bound the maximum utility score achieved by each view V_i, and in turn allows for pruning low-utility views without incurring the high cost for evaluating their importance.

Our proposed pruning technique is based on the observation that the utility score of each view $U(V_i)$ is a weighted sum of two measures; 1) the importance score of view $I(V_i)$ and 2) distance of a view from S $setDist\,(V_i, S)$. The computation of $setDist\,(V_i, S)$ requires only CPU computations and is a faster operation. Whereas, computing the importance score of a view is an expensive operation that requires executing view query on both the target and reference subset of data. Thus, we employ a max-min pruning technique as presented in [8], that leverages the diversity score to estimate the utility score of a view without computing the importance score.

Particularly, max-min utilizes the maximum and minimum bound on the importance score $I(V_i)$. discuss bound Clearly, the minimum utility score is $I_0 = 0$, whereas the maximum utility value I_u a view can score is computed as $I_u = \sqrt{2}$. Using those bounds, in each iteration the maximum $maxU(V_i)$ and minimum utility $minU(V_i)$ score for each view $V_i \in X$ is calculated as:

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

Accordingly, the maximum value of $minU$ is recorded (i.e., maximum of mimumum). Then, if $maxU$ of any candidate view is less than the maximum of $minU$, then that view is pruned.

As shown in Figure 4, the maximum value of $minU'$ is 0.25. Hence, all the views with $maxU'$ value less than 0.25 are pruned and view queries are generated only for the remaining views. Once the actual importance score is calculated for the remaining queries, the DiVE-Greedy algorithm selects the view with highest utility score to be added in S. Thus, the set of views computed by DiVE-Greedy using pruning is same as the one calculated without pruning.

Views in $\mathbb{V}\backslash$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

The Max-Min pruning technique utilizes the maximum and minimum bound on the importance score. The maximum utility value I_u a view can score is computed as $I_u = \sqrt{2}$, and the minimum utility score is $I_0 = 0$. Using those bounds, in each iteration maximum and minimum utility score is calculated for each view in X as :

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

The maximum value of the $minU$ is recorded. If $maxU'$ of any candidate view is less than the maximum of $minU'$, then that view is pruned. As shown in Figure 4, the maximum value of $minU'$ is 0.25. Hence, all the views with $maxU'$ value less than 0.25 are pruned and view queries are generated only for the remaining views. Once the actual importance score is calculated for the remaining queries, the DiVE-Greedy algorithm selects the view with highest utility score to be added in S. Thus, the set of views computed by DiVE-Greedy using pruning is same as the one calculated without pruning.

4.4 DiVE-Swap Scheme

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 $F(S)$. It is because the view which has the highest utility score not necessary be the best one that improve the objective

Algorithm 2: *DiVE* Swap

```
   Input: Set of views V and result set size k
   Output: Result set S ≥ V, |S| = k
1  S ← Result set of only importance or only diversity;
2  X ← [V\S];
3  F_current ← 0;
4  improve ← True;
5  while improve = True do
6      for i in set X do
7          S′ ← S;
8          for j in set S do
9              if F(S′) < F(S\S[j] ∪ X[i]) then
10                 S′ ← S\j ∪ X[i] ;
11             end
12         end
13         if F(S′) > F(S) then
14             S ← S′
15         end
16     end
17     if F(S) > F_current then
18         F_current ← F(S);
19         improve ← True;
20     else
21         improve ← False;
22     end
23 end
24 return S
```

function $F(S)$ (e.g: local optimum). To overcome that issue, we proposed other schemes which based on swap technique.

Swap is local search type algorithm and it has been known and used to maximize diversity in the literature [2, 18]. 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 a view in X is able to improve objective function value $F(S)$, then this view can 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.

Due to Swap need a complete initial set, 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. 2) ***DiVE-dSwap*** which is quite similar to DiVE-iSwap, however, this scheme is initialized by results of Greedy-Diversity, which is diversity maximized. Those two swaps have different initial set and in each iteration, the candidicate view is exchanged from X to the current set S till the $F(S)$ is maximized as given in Eq 3.

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 which can be seen in Algorithm 2 line 5. In the worst case, swap algorithm can perform $O\left(k^n\right)$ iterations. Without any pruning scheme, the cost of *DiVE-iSwap* is same as *DiVE-dSwap* due to those both schemes are using same technique only different in the initial set.

4.5 DiVE-dSwap Static Pruning

In terms of pruning, two our proposed Swap are quite different. *DiVE-iSwap* utilize the results of Linear-Importance as the initial set. 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, we can employ pruning technique in *DiVE-dSwap* by leveraging the properties of importance and diversity.

While in *DiVE-Greedy-Static*, the maximum and minimum bound of importance score are utilized, in this scheme, only maximum bound I_u is used. This *DiVE-dSwap-Static* also leverage both the importance and diversity score of a candidate view to decide whether a view query should be executed or not. The details *DiVE-dSwap-Static* technique explained as follows:

- Since the initial set of *DiVE-dSwap* is the result of Greedy-Diversity, all query views in the initial set need to be executed in order to get the objective function $F(S)$ of the current set S. The $F(S)$ of current set will be compared to the new $F(S)$ after exchanging a view as shown in Algorithm 2 line 9.
- In order to confirm that exchanging process starts from the candidate view that has highest score of diversity, all views in X is sorted based on $setDist\,(V_i, S)$ before start exchanging view from X to the current set S. This is called as *"top-1"* technique.
- To start exchanging view, the importance score must be known by executing the query of the candidate view. Instead of executing query view, the maximum bound of importance score is used to compute the utility score of each view as in *DiVE-Greedy-Static* technique. Hence, the result is not the actual utility score but $maxU'$, which defined as: $maxU'\,(V_i) = (1 - \lambda) \times I_u\,(V_i) + \lambda \times setDist\,(V_i, S)$.
- The exchanging process is started by comparing $F(S)$ of the current set to $F(S)$ of new set as given in Algorithm 2 lines 9 - 10. The $F(S)$ of new set is computed by Eq 3 while $maxU'$ is used as the utility score of candidate view from X.
- If using importance score I_u candidate views in X cannot improve the objective function $F(S)$ to the current set S, those views will be pruned.

This technique is valid due to if the maximum score of importance I_u is used and that view cannot improve $F(S)$ of the current set, then there is no reason to execute the view query to get the importance score $I\ (I \leq I_u)$.

All proposed pruning techniques including *DiVE-Greedy-Static* and *DiVE-dSwap-Static* are using static value I_u as the bound. However, the pruning performance cannot be optimal while the value of I_u is far away from the actual maximum of importance score in database. To overcome this issue, we proposed adaptive pruning scheme as described in the next section.

4.6 Adaptive Pruning Scheme

Two pruning techniques *DiVE-Greedy-Static* and *DiVE-dSwap-Static* have been presented. Those two static pruning techniques utilized maximum bound I_u to determine whether the query view need to be executed or not. Only view that can improve the $F(S)$ of the current set while using I_u will be executed otherwise those are pruned. However, one drawback using static bound I_u in pruning technique is that if the bound is far away from the maximum score of importance score in the dataset, the pruning cannot work optimal. To overcome this issue, instead of using static bound I_u, we proposed adaptive pruning scheme that automatically adapts the bound to the real maximum importance score in the dataset.

The adaptive pruning technique is utilizing the maximum bound I_u as in static pruning as a first initial bound, however, this bound is changed to the real value of maximum importance score after some query views are executed. The problem occurs when the executed views have a small importance score and it is far below from the most views in the dataset. Thus, it brings the pruning out of control because while the bound is very low and there are many views in dataset have higher importance score compared to the bound, it may result wrong prune. Hence, DiVE needs the strategy to ensure that the bound score is close as possible to the maximum importance score in the dataset while it is changed. One of the approach that can be used is by selecting sample views to be executed then get the maximum importance score of the view from those sample. This brings us to the question of how many samples are needed in order to hit a view that has a maximum score from the dataset.

There are several literatures have been mentioned related to the confidence interval and the number of samples in the normal distribution [cite]. However, the importance score of candidate views in X is not in normal distribution. The highest importance score is the upper bound of maximum importance I_u 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*) which is similar to a confidence interval in normal distribution. The relation between *PI* and the number of samples defined as in equation 6.

$$PI = \frac{(N-1)}{(N+1)}, where\ N = Number\ of\ samples \tag{6}$$

In general, analyst may use *PI* start from 80 to 99. While *PI* = 80% states that there are 9 sample views need to be executed, 85 %, 90%, 95%, 97%, and 99% means 12, 20, 40, 60, and 200 samples need to be executed respectively.

Adaptive pruning flows. We employ adaptive pruning technique to both schemes, *DiVE-Greedy* and *DiVE-dSwap*. In case of Greedy technique, the upper bound I_u is used at the first time, thus the value I_u is changed to maximum importance score from the samples of views which are executed. In order to change the bound value, the number of samples that need to be executed depends on the *PI* value which defined by the analyst. Futhermore, the bound is changed while in the next view execution that there is a view which has importance score higher than the used current bound.

For adaptive pruning technique in *DiVE-dSwap*, the details is described as follows:

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
aggregate functions	Max Min Avg Sum

- Firstly, as in *DiVE-dSwap-Static* that all query view in the initial set are executed in order to get the objective function $F(S)$ of the current set S and all candidate views in X is sorted based on $setDist\ (V_i, S)$.
- $maxU'$ of each view is computed by utilizing the maximum bound of importance score I_u, where $maxU'\ (V_i) = (1-\lambda) \times I_u\ (V_i) + \lambda \times setDist\ (V_i, S)$.
- All views in X is exchanged to the current set one by one and a view that can improve $F(S)$ will be executed in order to get the actual value of importance.
- The bound is changed while the number of views which are executed reaches the number of sample based on the PI which determined by analyst. For instance, analyst may use PI = 97%, hence, bound is changed while the sum of number of candidate views and th number of views in the initial set equal to 60 views. While it reaches to 60 views, the bound is replaced by the maximum importance score of executed views.
- If in the next query view execution, there is a view which has higher importance score than the bound. Thus, the bound is changed to that score.

In this work, adaptive pruning in Greedy is called *DiVE-Greedy-Adaptive* wheras in Swap is called *DiVE-dSwap-Adaptive*.

5 EXPERIMENTAL TESTBED

In this section, we present an evaluation of our proposed *DiVE* scheme both in terms of effectiveness and efficiency when returning diversified interesting visualizations.Table 2 summarizes the different parameters used in our experimental evaluation. All the experiments . The default values are presented in bold. All experiments were run on a single machine with 16 GB RAM and a 64 bit, Intel Core i7-7700 processor. All the performance measures are averaged over ten runs. We have conducted our experiments over following real datasets:

- Heart Disease Dataset [2]: This dataset is comprised of 9 dimensional attributes and 5 measure attributes. There are 299 records in the dataset and number of possible views are 180.
- Airline (Flights) Dataset [3]: This dataset is comprised of 7 dimensional attributes and 4 measure attributes. It has 855633 records and the number of possible views are 112.

In particular, we evaluate the performance of following schemes:

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

[3]http://stat-computing.org/dataexpo/2009/the-data.html

- Linear-Importance: Selects top-k views on the basis of only the importance score.
- Greedy-Diversity: Selects top-k diverse views.
- DiVE-Greedy: Selects top-k views on the basis of hybrid objective function using greedy algorithm.
- DiVE-iSwap: Selects top-k views on the basis of hybrid objective function using swap algorithm initialized by Linear-importance method.
- DiVE-dSwap: Selects top-k views on the basis of hybrid objective function using swap algorithm initialized by Greedy-Diversity method.

6 EXPERIMENTAL EVALUATION

In this section, we evaluate the sensitivity of *DiVE* scheme to the different parameters as given in Table 2.

6.0.1 **The impact of λ on the objective function $F(S)$.** The value of λ balances the trade off between importance and diversity score. Figure 5 shows how the performance of each scheme in terms of $F(S)$ is effected as the value of λ varies from 0 to 1. It is clearly seen in Figure 5, that for the lower values of λ the highest objective function value is achieved by Linear-Importance method. To the contrary, the Greedy-Diversity method achieves highest values of $F(S)$ as the λ approaches 1. Hence, there is a crossover between Linear-Importance and Greedy-Diversity. However, our proposed schemes based on the hybrid objective function have stable performance for all values of λ and are able to achieve $F(S)$ values higher than those achieved by Linear-Importance and Greedy-Diversity. compare the DiVE schemes, why value of F goes down for all with increase in λ

6.0.2 **The impact of k on the objective function $F(S)$.** Figure 6 shows the $F(S)$ values for various schemes as the number of recommended views k varies from 5 to 35. Overall the $F(S)$ value decreases with increasing value of k for all the schemes. This is because both average importance score and diversity of a set S decreases as new views are added to S. The views added earlier to S have higher importance score then the views added later. Similarly, the diversity function exhibits a diminishing marginal gain trend as the size of set S increases. The important observation here is the fact that our *DiVE* schemes always have higher overall objective function values as compared to the two extreme baselines approaches for all values of k. Among the *DiVE* schemes, DiVE-iSwap and DiVE-dSwap perform better than the Greedy versions. This is because, swap algorithm performs number of additional iterations to improve the value of the objective function.

7 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 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] Charles L.A Clarke and et al. 2008. Novelty and diversity in information retrieval evaluation. *SIGIR* (2008).

[2] M Drosou and E Pitoura. 2010. Search Result Diversification. *SIGMOD Record* 39, 1 (2010).

[3] H Ehsan, M. Sharaf, and Panos K. Chrysanthis. 2016. MuVE: Efficient Multi-Objective View Recommendation for Visual Data Exploration. *ICDE* (2016).

[4] Sean Kandel and et al. 2012. Profiler: Integrated statistical analysis and visualization for data quality assessment. In *AFI*. ACM, 547–554.

[5] V. Kantere. 2016. Query Similarity for Approximate Query Answering. In *DEXA*.

[6] Verena Kantere, George Orfanoudakis, Anastasios Kementsietsidis, and Timos K. Sellis. 2015. Query Relaxation across Heterogeneous Data Sources. In *CIKM*.

[7] Alicia Key, Bill Howe, Daniel Perry, and Cecilia R. Aragon. 2012. VizDeck: self-organizing dashboards for visual analytics. *SIGMOD Conference* (2012).

[8] Hina A. Khan and Mohamed A. Sharaf. 2015. Progressive diversification for column-based data exploration platforms. *ICDE* (2015).

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

[10] Thibault Sellam and Martin L. Kersten. 2016. Fast, Explainable View Detection to Characterize Exploration Queries. In *SSDBM*. 20:1–20:12.

[11] Thibault Sellam and Martin L. Kersten. 2016. Ziggy: Characterizing Query Results for Data Explorers. *PVLDB* 9, 13 (2016), 1473–1476.

[12] Jinwook Seo and Ben Shneiderman. 2006. Knowledge Discovery in High-Dimensional Data: Case Studies and a User Survey for the Rank-by-Feature Framework. *TVGC* 12, 3 (2006), 311–322.

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

[14] Quoc Trung Tran and Chee-Yong Chan. 2010. How to ConQueR why-not questions. In *SIGMOD*.

[15] M Vartak and et al. 2015. SEEDB : Efficient Data-Driven Visualization Recommendations to Support Visual Analytics. *VLDB* 8 (2015).

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

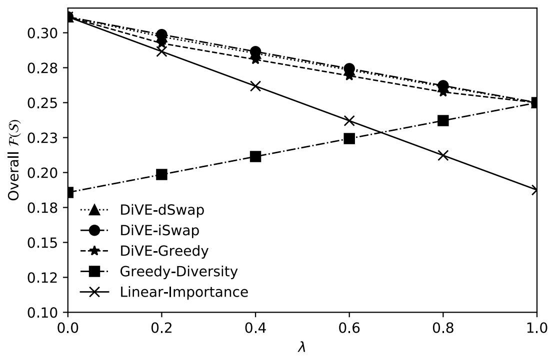
[17] Fernanda B. Viegas and et al. 2007. Many Eyes: A site for visualization at internet scale. *TVGC* (2007), 1121–1128.

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

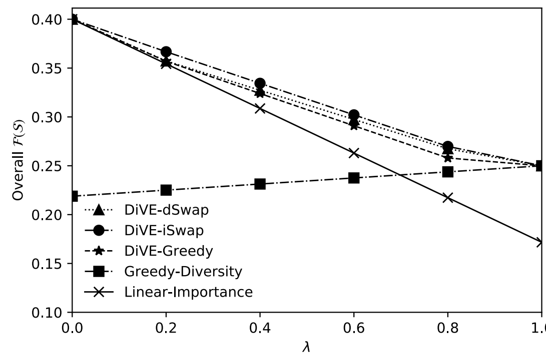
[19] Eugene Wu, Leilani Battle, and Samuel R. Madden. 2014. The case for data visualization management systems. *VLDB Endowment* (2014).

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

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

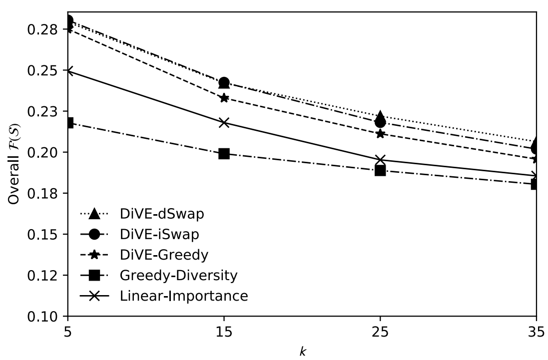


(a) Heart disease dataset

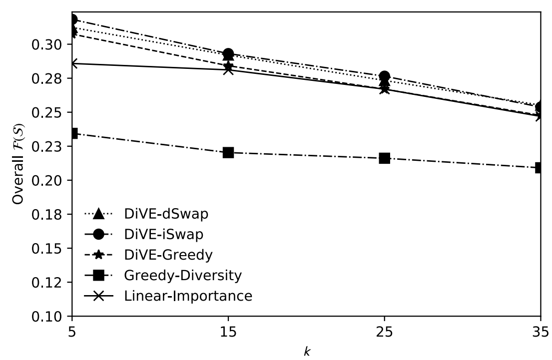


(b) Flights dataset

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



(a) Heart disease dataset



(b) Flights datasets

Figure 6: Overall objective function value $F(S)$ using different value of k and running on three real datasets