Explore-by-Example: An Automatic Query Steering Framework for Interactive Data Exploration

Kyriaki Dimitriadou*, Olga Papaemmanouil* and Yanlei Diao†
* Brandeis University, Waltham, MA, USA, † University of Massachusetts, Amherst, MA, USA
*{kiki,olga}@cs.brandeis.edu, †yanlei@cs.umass.edu

ABSTRACT

Interactive Data Exploration (IDE) is a key ingredient of a diverse set of discovery-oriented applications, including ones from scientific computing and evidence-based medicine. In these applications, data discovery is a highly ad hoc interactive process where users execute numerous exploration queries using varying predicates aiming to balance the trade-off between collecting all relevant information and reducing the size of returned data. Therefore, there is a strong need to support these human-in-the-loop applications by assisting their navigation in the data to find interesting objects.

In this paper, we introduce AIDE, an *Automatic Interactive Data Exploration* framework, that iteratively steers the user towards interesting data areas and "predicts" a query that retrieves his objects of interest. Our approach leverages relevance feedback on database samples to model user interests and strategically collects more samples to refine the model while minimizing the user effort. AIDE integrates machine learning and data management techniques to provide effective data exploration results (matching the user's interests with high accuracy) as well as high interactive performance. It delivers highly accurate query predictions for very common conjunctive queries with very small user effort while, given a reasonable number of samples, it can predict with high accuracy complex conjunctive queries. Furthermore, it provides interactive performance by limiting the user wait time per iteration to less than a few seconds in average. Our user study indicates that AIDE is a practical exploration framework as it significantly reduces the user effort and the total exploration time compared with the current-state-of-the-art approach of manual exploration.

Categories and Subject Descriptors

H.2.4 [**Systems**]: Relational Databases

Keywords

data exploration; database sampling; query formulation

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
SIGMOD'14, June 22–27, 2014, Snowbird, UT, USA.
Copyright 2014 ACM 978-1-4503-2376-5/14/06 ...$15.00.
http://dx.doi.org/10.1145/2588555.2610523.

1. INTRODUCTION

Traditional DBMSs are designed for applications in which the user queries to be asked are already well understood. There is, however, a class of *interactive data exploration* (IDE) applications, in which users are trying to make sense of the underlying data space by experimenting with queries, backtracking on the basis of query results and rewriting their queries aiming to discover interesting data objects. IDE often incorporates "human-in-the-loop' and it is fundamentally a long-running, multi-step process with the user data interests often specified in imprecise terms.

One example of IDE can be found in the domain of evidence-based medicine (EBM). Such applications often involve the generation of systematic reviews, a comprehensive assessment of the totality of evidence that addresses a well-defined question, such as the effect on mortality of giving versus not giving drug A within three hours of a symptom B. While a content expert can judge whether a given clinical trial is of interest or not (e.g., by reviewing parameter values such as disease, patient age, etc.), he often does not have a priori knowledge of the exact attributes that should be used to formulate a query to collect all relevant clinical trials. Specifically, IDE is a highly ad hoc process that includes three steps: 1) processing numerous selection queries with iteratively varying selection predicates, 2) reviewing returned objects (i.e., trials) and classifying them to relevant and irrelevant, and 3) adjusting accordingly the selection query for the next iteration. Users aim to discover a query that balances the trade-off between collecting all relevant information and reducing the size of returned data. Systematic reviews are typically labor-intensive: they may take days to weeks to complete since users need to examine thousands of objects.

Scientific applications, such as ones analyzing astrophysical surveys (e.g., [2, 4]), also suffer from similar situations: scientists may not be able to express their data interests precisely. Instead, they may want to navigate through a subspace of the data set (e.g., a region of the sky) to find objects of interest, or may want to see a few samples, provide yes/no feedback, and expect the system to find more similar objects.

To address the needs of IDE applications, we propose an *automatic interactive data exploration (AIDE)* framework that automatically "steers" the user towards data areas relevant to his interest. Our approach integrates the three IDE steps—query formulation, query processing, and result reviewing—into a single automatic process, significantly reducing the user effort and the overall exploration time. In AIDE, the user engages in a "conversation" with the system indicating his interests, while in the background the system automatically formulates and processes queries that collect data matching the user interest.

In AIDE, the user is prompted to label a set of strategically collected sample objects (e.g., clinical trials) as relevant or irrelevant

to his IDE task. Based on his feedback, AIDE generates the user exploration profile which is used to collect a new set of sample objects. These new samples are presented to the user and his relevance feedback is incorporated into his profile. In the background, AIDE leverages the user profile to *automatically* generate data extraction queries that retrieve more objects relevant to the user's IDE task while minimizing the retrieval of irrelevant ones.

The design of AIDE raises new challenges. First, AIDE operates on the unlabeled space of the whole data space that the user aims to explore. To offer effective data exploration results (i.e., accurately predict the user's interests) it has to decide and retrieve in an *online* fashion the example objects to be extracted and labeled by the user. Second, to achieve desirable interactive experience for the user, AIDE needs not only to provide accurate results, but also to minimize the number of samples presented to the user (which determines the amount of user effort).

These challenges cannot be addressed by existing machine learning techniques. Classification algorithms (e.g., [8]) can be leveraged to build the user interest model and the information retrieval community offers solutions on incrementally incorporating relevance feedback in these models (e.g., [31]). However, these approaches operate under the assumption that the sample set shown to the user is either known a priori or, in the case of online classification, it is provided incrementally by a different party. Therefore, classification algorithms do not deal with *which* data samples to show to the user. Furthermore, the active learning community has proposed solutions that maximize the accuracy of the model while minimizing the number of samples shown to the user. However, these techniques are domain specific (e.g., document ranking [24], image retrieval [22], etc.) and they exhaustively examine *all* objects in the data set in order to identify the best samples to show to the user [23]. Therefore, they implicitly assume negligible sample acquisition costs and hence cannot offer interactive performance on big data sets as expected by IDE applications. In either case, model learning and sample acquisition are decoupled, with the active learning algorithms not addressing the challenge of *how* to minimize the cost of sample acquisition.

To address the above challenges, AIDE closely *integrates* classification model learning (from existing labeled samples) and effective data exploration and sample acquisition (deciding new data areas to sample). Our techniques leverage the classification properties of decision tree learning to discover promising data exploration areas from which new training samples are extracted, as well as to minimize the number of samples required. These techniques aim to predict linear patterns of user interests, i.e., range selection queries.

The specific contributions of this work are the following:

1. We introduce AIDE, a novel, automatic data exploration framework, that navigates the user throughout the data space he wishes to explore. AIDE relies on the user's feedback on example objects to formulate queries that retrieve data relevant to the user. It employs a unique combination of machine learning, data exploration, and sample acquisition techniques to deliver highly accurate predictions of linear patterns of user interests with interactive performance.
2. We propose data exploration techniques that leverage the properties of classification models to identify *single* objects of interest, expand them to more accurate *areas of interests*, and progressively refine the characterization of these areas.
3. We introduce optimizations that reduce the number of samples required by each of the proposed exploration techniques, the number of sample extraction queries, and the user wait time. Our optimizations are designed to address the trade-off between quality of results (i.e., accuracy) and efficiency (i.e., the total *exploration time* which includes the total sample reviewing time and wait time by the user).
4. We evaluated our implementation of AIDE using the SDSS database [4] and a user study. Our results indicate that AIDE can predict common conjunctive queries with a small number of samples, while given an acceptable number of labeled samples it predicts highly complex disjunctive queries with high accuracy. AIDE also offers interactive performance as the user wait time per iteration is less than a few seconds in average. Our user study revealed that AIDE can reduce the user's labeling effort by up 87%, with an average of 66% reduction. When also including the sample reviewing time, it reduced the total exploration time by 47% in average.

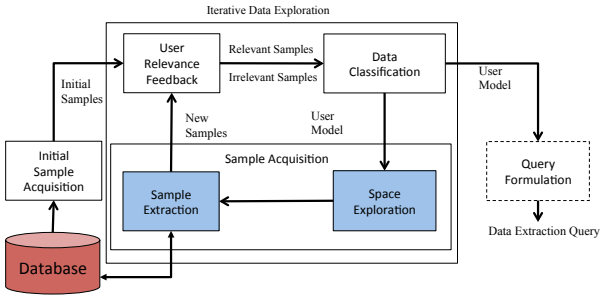


Figure 1: Automated Interactive Data Exploration Framework.

The rest of the paper is organized as follows. Section 2 outlines the AIDE framework and the phases of our data exploration approach. Section 3 discusses the object discovery phase, and Sections 4 and 5 describe the misclassified and boundary exploitation phase, respectively. Section 6 presents our experimental results. Section 7 discusses the related work and we conclude in Section 8.

2. AIDE FRAMEWORK OVERVIEW

In this section we describe the main functionality of our system and the classification algorithms we used. Furthermore, we provide an overview of our exploration techniques.

2.1 System Model

The workflow of our framework is depicted in Figure 1. Initially, the user is presented with sample database objects (*Initial Sample Acquisition*) and asked to characterize them as relevant or not to his exploration task. For example, in the domain of evidence-based medicine, users are shown sample tuples extracted from a table with clinical trials records and they are asked to review their attributes (e.g., year, outcome, patience age, medication dosage, etc) and label each sample trial as interesting or not. In this work we assume a binary, non noisy, relevance system where the user indicates whether a data object is relevant or not to him and this categorization cannot be modified in the following iterations.

The iterative steering process starts when the user provides his feedback. The labeled samples are used to train a classification model that characterizes the user interest, e.g., it predicts which clinical trials are relevant to the user based on the feedback collected so far (*Data Classification*). Classification models may use any subset of the object attributes to characterize user interests. However, domain experts could restrict the attribute set on which the exploration is performed. For instance, one could request an exploration only on the attributes *dosage* and *age*. In this case, relevant clinical trials will be characterized only based on these attributes (e.g., relevant trials have dosage >45mg and age <35 years).

In each iteration, more samples (e.g., records of clinical trials) are extracted and presented to the user for feedback. AIDE lever-

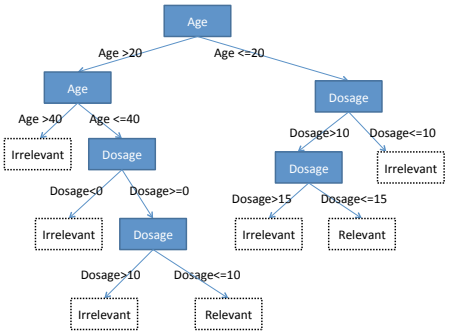


Figure 2: An example decision tree.

ages the current user model to identify promising sampling areas (*Space Exploration*) to retrieve the next sample set from the database (*Sample Extraction*). New labeled objects are incorporated with the already labeled sample set and a new classification model is built. The steering process is completed when the user terminates the process explicitly, e.g., when reaching a satisfactory set of relevant objects or when he does not wish to label more samples. Finally, AIDE "translates" the classification model into a query expression (*Data Extraction Query*). This query will retrieve objects characterized as relevant by the user model (*Query Formulation*).

AIDE strives to converge to a model that captures the user interest, i.e., eliminating irrelevant objects while identifying a large fraction of relevant ones. Each round refines the user model by exploring further the data space. The user decides on the effort he is willing to invest (i.e., number of samples he labels) while AIDE leverages his feedback to maximize the accuracy of the classification model. The more effort invested in this iterative process, the better accuracy achieved by the classification model. However, higher numbers of samples increase the sample extraction and processing time, i.e., the user *wait time* between iterations.

2.2 Data Classification & Query Formulation

AIDE relies on decision tree classifiers to identify linear patterns of user interests. Decision tree learning [8] produces classification models that predict the class of an unclassified object based on labeled training data. The major advantage of decision trees is that they provide easy to interpret prediction models that describe the features characterizing each data class. Furthermore, they perform well with large data and provide a white box model, i.e., the decision conditions of the model can be easily translated to simple boolean expressions. This feature is important since it allows us to map decision trees to queries that retrieve the relevant data objects.

Finally, decision trees can handle both numerical and categorical data. This allows AIDE to operate on both data types assuming a distance function is provided for each domain to calculate the similarity between two data objects. Measuring the similarity between two objects is a requirement of the space exploration step. AIDE treats the similarity computation as an orthogonal step and can make use of any distance measure.

Query Formulation Let us assume a decision tree classifier that predicts relevant and irrelevant clinical trials objects based on the attributes `age` and `dosage` as shown in Figure 2. This tree provides predicates that characterize the relevant class and predicates that describe the irrelevant class. In Figure 2, the relevant class is described by the predicates $(age \leq 20 \wedge 10 < dosage \leq 15)$ and $(20 < age \leq 40 \wedge 0 \leq dosage \leq 10)$, while the irrelevant class is characterized by the predicates $(age \leq 20 \wedge dosage \leq 10)$ and $(20 < age \leq 40 \wedge dosage > 10)$ (here we ignore the predicates that refer to values outside attribute domains, such as $age > 40$, $age < 0$, $dosage < 0$ and $dosage > 15$). Given the decision tree in Figure 2 it is straightforward to formulate the extraction query for the relevant objects (*select * from table where (age* $\leq$ *20 and dosage >10 and dosage* $\leq$ *15) or (age > 20 and age* $\leq$ *40 and dosage* $\geq$ *0 and dosage* $\geq$ *10)*).

2.3 Problem Definition

Given a database schema $\mathcal{D}$, let us assume the user has decided to focus his exploration on d attributes, where these d attributes may include both attributes relevant and those irrelevant to the final query that represents the true user interest. Each exploration task is then performed in a d-dimensional space of T tuples where each tuple represents an object characterized by d attributes. For a given user, our exploration space is divided to the relevant object set T^r and irrelevant set T^{nr}. In each iteration i, a sample tuple set $S_i \subseteq T$ is shown to the user and his relevance feedback assigns them to two data classes, the relevant object class $D^r \subseteq T^r$, and the irrelevant one, $D^{nr} \subseteq T^{nr}$. Based on the samples assigned to these classes up to the i-th iteration, a new decision tree classified C_i is generated. This classifier corresponds to a predicate set $P_i^r \bigcup P_i^{nr}$, where the predicates P_i^r characterize the relevant class and predicates P_i^{nr} describe the irrelevant one.

We measure AIDE's effectiveness (aka accuracy) by evaluating the F-measure, the harmonic mean between precision and recall.[1] Our goal is to maximize the F-measure of the final decision tree C on a total data space T, defined as:

$$F(T) = \frac{2 \times precision(T) \times recall(T)}{precision(T) + recall(T)} \quad (1)$$

The perfect precision value of 1.0 means that every data characterized as relevant by the decision tree is indeed relevant, while a good recall ensures that our final query can retrieve a good percentage of the relevant to the user objects.

2.4 Space Exploration Overview

Our main research focus is on optimizing the effectiveness of the exploration while minimizing the number of samples presented to the user. We assume that user interests are captured by *range queries*, i.e., relevant objects are clustered in one or more areas in the data space. Therefore, our goal is to discover *relevant areas* and formulate user queries that select either a single relevant area (conjunctive queries) or multiple ones (disjunctive queries).

To do so, AIDE incorporates three exploration phases. First, we focus on collecting samples from yet unexplored areas and identifying single relevant objects (*Relevant Object Discovery*). Next, we strive to expand single relevant objects to relevant areas (*Misclassified Exploitation*). Finally, given a set of discovered relevant areas, we gradually refine their boundaries (*Boundary Exploitation*).

These phases are designed to collectively increase the accuracy of the exploration results. Given a set of relevant objects from the object discovery step, the misclassified exploitation increases the number of relevant samples in our training set by mapping relevant objects to relevant areas. Furthermore, as we will discuss in Section 4, it reduces misclassified objects (specifically false positives). Hence, this step improves both the recall and the precision parameters of the F-measure metric. The boundary exploitation further refines the characterization of the already discovered relevant ar-

[1] Here, if tp are the true positives results of the classifier (i.e., correct classifications as relevant), fp are the false positives (i.e., irrelevant data classified as relevant) and fn are the false negatives (i.e., relevant data classified as irrelevant), we define the precision of our classifier as $precision = \frac{tp}{tp+fp}$ and the recall as $recall = \frac{tp}{tp+fn}$.

eas. Therefore, it discovers more relevant objects and eliminates misclassified ones, leading also to higher recall and precision.

In each iteration i, these three phases define the new sample set we will present to the user. Specifically, if $T^i_{discovery}$, $T^i_{misclass}$ and $T^i_{boundary}$ samples will be selected by the object discovery, the misclassified and the boundary exploitation phase, respectively, then the new sample set in the i-th iteration will be:

$$S_i = T^i_{discovery} + T^i_{misclass} + T^i_{boundary} \quad (2)$$

One interesting artifact of using the F-measure is that failing to discover a relevant area reduces our accuracy dramatically more than failing to precisely characterize the boundaries of that relevant area. Therefore, if we assume that the user's relevant objects are clustered in $M > 1$ areas, discovering (with slightly imprecise boundaries) all M relevant areas will lead to higher accuracy that predicting less than M areas but with highly accurate boundaries. Therefore, our exploration techniques place a limit on the samples collected by the boundary exploitation phase, aiming to utilize the user effort more on the other two more effective phases.

3. RELEVANT OBJECT DISCOVERY

This phase aims to discover relevant objects by showing to the user samples from diverse data areas. To maximize the coverage of the exploration space it follows a well-structured approach that allows AIDE to (1) ensure that the exploration space is explored widely, (2) keep track of the already explored sub-areas, and (3) explore different data areas in different granularity.

Our approach operates on a *hierarchical exploration grid*. For an exploration task on d attributes, the overall *exploration space* is the d-dimensional space defined by the *domain* of these attributes. AIDE creates off-line a set of grids where each grid divides the exploration space into d-dimensional grid cells with equal width in each dimension. We refer to each grid as an *exploration level* and each level has a different granularity, i.e., cells of different width. Lower exploration levels include finer-grained grids (i.e., more grid cells of smaller width) and therefore moving between levels allows us to "zoom in/out" into specific areas as needed.

Exploration Level Construction To generate an exploration level we divide each normalized attribute domain[2] into β equal width ranges, effectively creating β^d grid cells. The β parameter defines the granularity of the specific exploration level. A higher number leads to more grid cells of smaller width per dimension and the use of more samples to explore *all* grid cells for fine-grained search for relevant objects. Each cell in our grid covers a certain range of attribute values for each of the d exploration attributes. Therefore, each cell includes a set of unique attribute value combinations. Each combination can be mapped to a set of data objects that match these attribute values. Next we present the general algorithm for this phase which explores uniformly all cells, collects one object from each cell and shows it to the user for feedback.

Object Discovery Phase Our approach starts at a given exploration level and retrieves one data object from each non-empty cell. Our goal is to uniformly spread the samples we collect across the grid cells to ensure the highest coverage of the exploration space. We achieve that by retrieving objects that are on or close to the center of each cell. Since the exploration levels are defined on the normalized domains of d attributes and we split each domain to β equal width ranges, each cell covers a range of $\delta = 100/\beta$ of the domain for each attribute. For each cell, we identify the "virtual" center and we retrieve a single random object within distance $\gamma < \delta/2$ along each dimension from this center. This approach guarantees that at each exploration level the retrieved samples are within $\delta \pm (2 \times \gamma)$ normalized distance from each other. Figure 3 shows a two-level 2-dimensional hierarchical grid (we show the second exploration level only for the top right grid cell) and the center for each cell.

[2]We normalize each domain to be between [0,100]. This allow us to reason about the distance between values uniformly across domains. Operating on actual domains will not affect the design of our framework or our results.

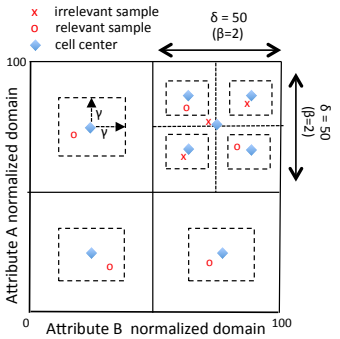


Figure 3: Grid-based object discovery example in 2-D space.

The sampling distance around the center of each cell, i.e., the γ parameter, depends on the skewness of each attribute domain. Specifically, we adjust γ based on the density of the cell. Sparse cells should use a higher γ value than dense ones to increase their sampling areas and hence improve the probability of retrieving one (relevant) object per grid cell. This will reduce ineffective zoom in operations into the next level for that cell.

Our general grid-based exploration proceeds as follows. In the first iteration we show to the user one object for each cell of the initial exploration level and we request his feedback (no other phases are applied in the first iteration). If no relevant object is retrieved from one cell, we can safely infer that the whole grid cell is not included in any relevant area. However, sub-areas of the grid could partially overlap with some relevant areas. Therefore, we further explore this grid by "zooming-in" and using a lower exploration level (only for the specific cell). In Figure 3 we found one relevant object in all but the top right cell and so we use the next exploration level for that cell only: we will search for objects around the center of the smaller four sub-cells inside the higher level cell.

By using this grid-based approach, our framework is able to keep track of the already explored subareas in the data space and spread its exploration across the whole domain of all the exploration attributes, as opposed to wasting resources sampling highly overlapping areas. At each exploration level the system requires β^d samples (equal to the number of grid cells for that level) to fully explore that level. Assuming the user is willing to label more samples the system can continue showing labels from the next level.

3.1 Optimizations

Next we discuss two optimizations for improving the relevant object discovery phase.

Hint-based Object Discovery Our general grid-based exploration assumes that our system does not have any information about the areas that the user is looking for. However, if the user provides some *hints* about his interests, this process can be optimized. One hint may be that the selection ranges in a given attribute d will have a width of at least x_d. For instance, the user may be interested in clinical trials at least one year apart or sky objects within a certain distance from each other. Given the parameter x_d, AIDE can initiate its exploration on the exploration level with cell width $\delta \leq x_d/100$. This will ensure that we will retrieve exactly one relevant object from each area of interest, therefore we will not miss

any relevant areas. Assuming that the user is willing to provide feedback on at least β^d samples (plus more samples for the two other phases) this hint can prevent unnecessary sampling on higher exploration levels, therefore improving the convergence to an accurate result while reducing the user's labeling effort.

Another hint is based on specific attribute ranges on which the user desires to focus (e.g., clinical trials in years [2000, 2010]). These range-based hints allow our framework to operate on specific grid cells and avoid necessary sampling and computation on irrelevant data areas. Obviously the more specific these hints, the more accurate the final query and the less the user labeling effort.

Handling Skewed Attributes Our grid construction approach is skew-agnostic: it creates equal width cells regardless of the cell density, i.e., the number of data objects mapped to each cell. In the presence of skewed exploration domains this approach will construct cells with highly diverse density. For those cells with low density (*sparse areas*), AIDE has a slim chance to discover samples close to its center. The current approach will then zoom into a lower exploration level to collect samples from the smaller subcells. This extra cost, however, often does not result in improved accuracy because the number of samples returned from a sparse area may be too low to improve the F-measure substantially.

To address this issue, we propose a skew-aware clustering-based approach to identifying sampling areas. More specifically, AIDE uses the k-means algorithm [8] to partition the data space into k clusters. Each cluster is characterized by its centroid and database objects are assigned to the cluster with the closest centroid. Thus, each cluster includes similar objects (where similarity is defined by a distance function) and each centroid serves as a good representative of the cluster's objects. Under this approach, AIDE collects samples around the centroid of each cluster. Specifically, we show one object per cluster within distance $\gamma < \delta$ along each dimension from the cluster's centroid, where δ is the radius of the cluster. We create multiple exploration levels, where higher levels include fewer clusters than lower ones, and we initialize our exploration on the highest level. If no interesting objects are discovered, we sample the lower level where finer-grained clusters are available.

In the presence of skewed exploration domains, this approach will be more effective since k-means will create most of the clusters in dense areas. This will allow AIDE to focus its sampling in areas with high density. Assuming that the user interests lie mostly in dense areas, our technique will avoid redundant sampling in sparse areas and the unnecessary subsequent zooming into the next level. Therefore, AIDE will converge to an accurate result with fewer labeled samples for those skewed exploration attributes.

4. MISCLASSIFIED EXPLOITATION

The object discovery phase identifies single points of interest, one for each sampling areas (gird cell or cluster) explored. Our goal though is to discover *relevant areas* as opposed to single objects. To map relevant points to relevant areas, a significant number of relevant objects from within each relevant area need to be fed to the classification algorithm. However, the object discovery phase cannot achieve this solely (except when the relevant areas are much larger than individual grid cells). Therefore, AIDE employs the *misclassified samples exploitation* phase to strategically increase the relevant objects in our training set such that the predicted queries will select relevant *areas*. This phase is designed to increase both the precision and recall of the final query as it increases the relevant samples while reducing the misclassified ones.

In this section we first discuss how misclassified samples are generated. By interpreting the characteristics of these samples we propose a clustering-based exploration technique that increases the accuracy of our approach. Our technique strives to limit the number of extraction queries and hence the time overhead of this phase.

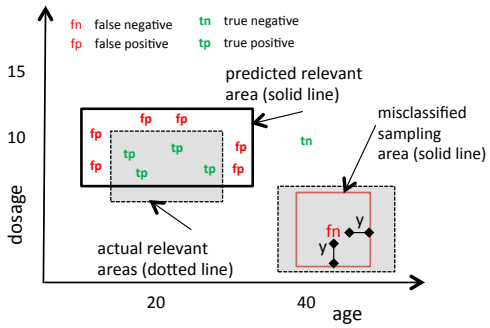


Figure 4: Misclassified objects and sampling around a single object.

4.1 Generation of Misclassified Samples

Let us assume that in iteration i a new decision tree classifier C_i is generated based on the sample set T_i. The decision tree characterizes a set of relevant and irrelevant areas. In the first few iterations, these areas typically cannot classify the training data accurately, leading to: (i) *false positives*, i.e., objects that are categorized as relevant by the classifier but labeled as irrelevant by the user and (ii) *false negatives*, i.e., objects labeled as relevant but categorized as irrelevant by the classifier. AIDE leverages the misclassified samples to identify the next set of sampling areas in order to discover more relevant areas. Since each type of misclassified samples has unique characteristics, we handle them differently.

False negatives are objects of interest that belong in a relevant area. However, there are no sufficient samples within that area to allow the classifier to characterize this area as relevant. Note that the object discovery phase provides a single object from each sampling area (i.e., grid cell or cluster). If the object is labeled as interesting then the training set of the following iteration will include only one relevant object from that sampling area. Depending on the sampling area size and the number of relevant areas, neighboring grid cells or clusters may not provide enough relevant samples to predict the relevant area. AIDE addresses the lack of relevant samples by collecting more objects around false negatives.

False positives on the other hand are less common in decision tree classifiers. This is due to the metrics used by decision tree algorithms for deciding the best classification rules for each class. In our case (CART [8]), this metric measures the homogeneity of the discovered relevant and irrelevant areas and the decision tree algorithm opts for rules that maximize this homogeneity. Practically, this implies that the predicted relevant areas are chosen to include as many of the relevant samples while minimizing the irrelevant ones. Therefore, most false positives are due to wrongly predicted boundaries of *discovered* relevant areas. Figure 4 shows examples of false positives around a not-precisely-predicted relevant area. This problem will be addressed by the boundary exploitation phase (Section 5) which refines the relevant areas.

4.2 Misclassified Samples Exploitation

Our misclassified exploitation phase operates under the assumption that relevant tuples will be clustered close to each other, i.e., they typically form relevant areas. One possible technique to leverage this knowledge is to handle each misclassified sample independently and collect samples around *each* false negative to obtain more relevant samples. Our experimental results show that this technique is very successful in identifying relevant areas. However, it incurs high time cost, mainly because (i) we execute one retrieval query per misclassified object and (ii) we often redundantly sample

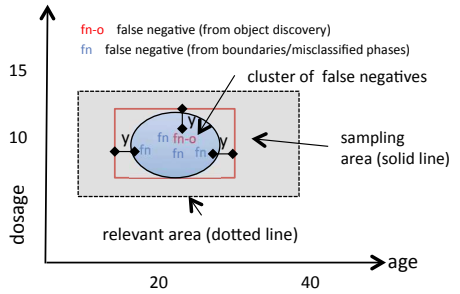


Figure 5: Clustering based misclassified sampling.

highly overlapping areas, spending resources (i.e., user labeling effort) without increasing much AIDE's accuracy. The last problem appears when many misclassified samples are close to each other.

We also noticed that sampling around each misclassified object often requires multiple iterations of the misclassified exploitation phase before the decision tree has enough samples to characterize a relevant area. Let us assume that in the i-th iteration the training set T_i has m_i false negatives based on the classifier C_i. Then, in the next iteration $i+1$ we add $T^{i+1}_{missclass} = m_i * f$ samples, where f is the number of samples we collect around each false negative. If the relevant area is still not discovered, these samples are also misclassified and then f samples are collected around each one of these samples. If AIDE needs k iterations to identify a relevant area, the user might have labeled f^k labeled samples without improving the F-measure (i.e., discovering a relevant area).

Clustering-based Exploitation To address this limitation, AIDE employs a clustering-based approach. The intuition is that misclassified samples that belong in the same relevant area will be located close to each other. Therefore, instead of sampling around each misclassified sample independently, we generate *clusters of misclassified objects* and we sample around each cluster. An example of a cluster is shown in Figure 5. We create clusters using the *k-means* algorithm [8] and have one sampling area per cluster.

The main challenge in this approach is identifying the number of clusters we need to create. Ideally, we would like the number of clusters to match the number of relevant areas we have "hit" so far, i.e., the number of areas from which we have collected at least one relevant object. This will ensure that we will sample within each discovered relevant area. We argue that the number of relevant objects created by the object discovery phase is a strong indicator of the number of relevant areas we have already "hit". The object discovery phase identifies objects of interest that belong to either different areas or the same relevant area. In the former case, our indicator offers correct information. In the latter case, our indicator will lead us to create more clusters than the already "hit" relevant areas. However, since these clusters belong in the same relevant area they are typically close to each other and therefore the decision tree classifier eventually "merges" them and converges to an accurate number of relevant areas.

In each iteration i, the algorithm sets k to be the overall number of relevant objects discovered in the object discovery phase. Since our goal is to reduce the number of sampling areas (and therefore the number of sample extraction queries), we run the clustering-based exploitation only if k is less than the number of false negatives. Otherwise we collect f random sample around *each* false negative. These samples are retrieved randomly from a normalized distance y on each dimension from the false negative, as shown in Figure 4. When clusters are created, we collect samples within a distance y from the farthest cluster member in each dimension. For each cluster we issue a query that retrieves $f \times c$ random samples within a sampling area, where c is the size of the cluster (number of cluster members). Figure 5 shows an example.

Our experimental results showed that f should be set to a small number (10-25 samples) since higher values will increase the user effort without improving the exploration outcome. The closer the value y is to the width of the relevant area we aim to predict, the higher the probability to collect relevant objects than irrelevant ones. An interesting optimization would be to dynamically adapt this value based on the current prediction of the relevant areas. We plan to explore this direction in our future work.

5. BOUNDARY EXPLOITATION

Given a set of relevant areas identified by the decision tree classifier, our next phase aims to refine these areas by incrementally adjusting their boundaries. This leads to better characterization of the user's interests, i.e., higher accuracy of our final results. In this section, we describe our general approach. We also discuss a series of optimizations that allow us to reduce the number of samples and the sample extraction time required to refine the boundaries of already discovered areas.

5.1 General Boundary Exploitation

AIDE represents the decision tree classifier C_i generated at the i-th iteration as a set of hyper-rectangles in a d-dimensional space defined by the predicates in $P^r_i \bigcup P^{nr}_i$, where the predicates P^r_i characterize the relevant areas and predicates P^{nr}_i describe the irrelevant areas. We iteratively refine these predicates by *shrinking* and/or *expanding* the boundaries of the hyper-rectangles. Figure 6 shows the rectangles for the classifier in Figure 2. If our classification is based on d attributes ($d = 2$ in our example) then a d-dimensional area defined by $p \in P^r_i$ will include objects classified as relevant (e.g., areas A and D in Figure 6). Similarly, objects in an area defined by $p \in P^{nr}_i$ are classified as irrelevant (e.g., areas B and C in Figure 6).

Our evaluation showed that this phase has the smallest impact on the effectiveness of our model: not discovering a relevant area can reduce our accuracy more than a partially discovered relevant area with imprecise boundaries. Hence, we constrain the number of samples used during this phase to α_{max}. This allows us to better utilize the user effort as he will provide feedback mostly on samples generated from the previous two, more effective phases.

Our approach aims to distribute an equal amount of user effort to refine each boundary. Let us assume the decision tree has revealed k d-dimensional relevant areas. Each area has 2^d boundaries. Hence we collect $\alpha_{max}/(k \times 2^d)$ random samples within a distance $\pm x$ from the each boundary. This approach is applied across all the boundaries of the relevant hyper-rectangles, allowing us to shrink/expand each dimension of the relevant areas. The new collected samples, once labeled by the user, will increase the recall metric: they will discover more relevant tuples (if they exist) and eventually refine the boundaries of the relevant areas.

The x parameter can affect how fast we converge to the real relevant boundary. If the difference between the predicted and real boundaries is less than x, this phase will retrieve both relevant and irrelevant samples around the boundary and allow the decision tree to more accurately predict the real boundary of the relevant area. Otherwise, we will mostly collect relevant samples. This will still improve our prediction of the boundary by bringing it closer to the actual one, but it will slow down the convergence to the actual relevant area. We follow a conservative approach and set x to 1 (we search for objects with normalized distance ± 1 from the current predicted boundary). This gradually improves our recall metric.

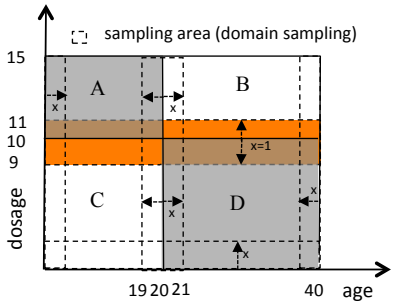


Figure 6: Boundary exploration for the relevant areas A and D.

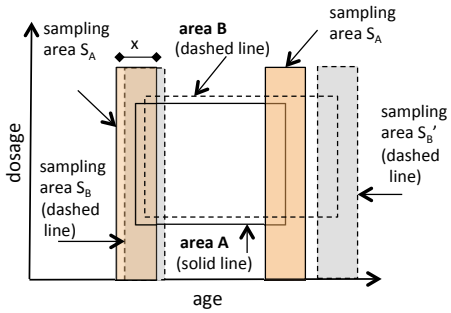


Figure 7: Overlap of sampling areas for similar decision trees.

Figure 6 shows with dotted lines the sampling areas when we shrink/expand the boundaries of the relevant area D ($20 < age \leq 40 \wedge 0 \leq dosage \leq 10$ of our example tree in Figure 2). Here, we collect samples that have distance $x = 1$ from the $dosage = 10$ boundary, i.e., random samples within the $9 \leq dosage \leq 11$ area.

5.2 Optimizations

Next we propose optimizations for boundary exploitation.

Adaptive Sample Size The first optimization dynamically adapts the number of samples collected. The goal is to reduce the sample size around boundaries that are already predicted with acceptable accuracy. The challenge here is that the accuracy of the projected areas cannot be determined in real-time as the actual interest of the user is unknown to our system. To address this, we leverage information from the already built decision tree.

Our decision tree consists of a set of decision nodes (also named split rules). Each rule corresponds to a boundary of a relevant or irrelevant area. AIDE identifies the split rules that characterize a relevant area (e.g., $dosage > 10$, $dosage \leq 15$ in Figure 2) and in each iteration quantifies the change of this rule (i.e., the change of its boundary value). Bigger changes of a split rule will lead to more samples extracted around the corresponding boundary. The intuition is that significant changes in the rule indicate that the corresponding boundary is not yet very accurate, and hence new samples were able to affect its split rule significantly. In the contrary, small changes in the split rule between two iterations indicate that the decision tree already has a good approximation of the boundary and the new samples did not affect the accuracy of the specific rule. In this case we restrict the samples provided to the user to a lower limit. We keep collecting this lower limit from all, even unmodified boundaries, to compensate the cases where lack of change in a boundary was due to randomness of the sample set and not to precise predictions of its coordinates.

Given a set of new training samples T_i at the i-th iteration, our algorithm will identify the decision tree split rules that are modified and translate them to changes on the boundaries of relevant areas. In each iteration i the number of samples collected in the boundary exploitation phase is calculated as:

$$T^i_{boundary} = \sum_{j=0}^{j=2^d} (pc^j_{i-1} * \frac{a_{max}}{k \times 2^d}) + er * (k \times 2^d)$$

where d is the dimensionality of the exploration space, pc^j_{i-1} is the percentage of change of the boundary j between the $(i-1)$-th and i-th iterations, and er is an error variable to cover cases where the boundary is not modified but also not accurately predicted. The percentage of change pc_{i-1} is calculated as the difference of the boundary's normalized values of the specific dimension.

Non-overlapping Sampling Areas Although the boundary exploitation can be effective, it is often the case that new samples lead to only a slightly (or not at all) modified decision tree. In this case, the exploration areas did not evolve significantly between iterations, resulting in redundant sampling and increased exploration cost (e.g., user effort) without improvements on classification accuracy. Figure 7 shows an example of this case, where one iteration indicates that relevant tuples are within area A whereas the following iteration reveals area B as the relevant one. Given the high overlap of A and B, the sampling areas around boundaries, e.g., S_A and S_B, will also highly overlap.

To address this challenge we rely on the decision tree structure. In each iteration, we identify the decision tree nodes that are modified and translate them to changes on the boundaries of relevant areas. Our sampling will then be limited to only areas with small or no overlap. For example, in Figure 7 in the second iteration we will sample area S'_B (but not S_B), since it does not overlap with the previous sampling areas S'_A and S_A. This approach allowes us to more efficiently steer the user towards interesting areas by reducing the number of iterations and the cost of redundant sampling.

Identifying Irrelevant Attributes Our general boundary exploitation is applied only on the attributes appearing in the decision tree and on the range of each attribute domain selected by the decision tree. Inevitably, this approach introduces skewness on the exploration attributes which in some cases may prevent the convergence to a good classification model. The problem is particularly obvious when the decision tree includes split rules with attributes irrelevant to the user interests, often due to lack of sufficient training samples. In our current experimentation we have often seen such cases which resulted into proposing a final query with different selection attributes from our target query.

To handle these cases we rely on *domain sampling* around the boundaries. While we shrink/expand one dimension of a relevant area we collect random samples over the *whole* domain of the remaining dimensions. Figure 6 demonstrates our technique: while the samples we collect are within the range $11 \leq dosage \leq 9$ they are randomly distributed on the domain of the age dimension. Our experimental results are based on this approach and we observed that the quality of our final classifier was noticeably improved compared with an approach that selects samples bounded in all dimensions of the relevant areas (e.g., samples in the range $11 \leq dosage \leq 9 \wedge 20 \leq age \leq 40$).

Exploration on Sampled Datasets The previous optimization forces our sample extraction queries of this phase to execute random sampling across the whole domain of each attribute. Such queries are particularly expensive, since they need to fully scan the whole domain of all attributes. Even when covering indexes are used to prevent access to disk, the whole index needs to be read for every query, increasing the sampling extraction overhead. We observed that although sampling on the whole domain improves our accuracy by an average of 42%, it also has a time overhead of 95% more than the other two phases. This overhead becomes even higher as we increase the size of our database.

To improve our sample extraction time and improve the scalability of AIDE, we apply our techniques on sampled data sets. Specifically, we generate a random sampled database and extract our samples from the smaller sampled dataset. We note that this optimization can be used for both the misclassified and the boundary exploitation phases. This allows our sample extraction time to improve, with the most positive impact coming from the boundary exploitation phase. We observed that operating on a sampled dataset with size equal to 10% of that of the original data set can improve our boundary exploitation time by up to 83.4% and the time for the misclassified exploitation phase by up to 74.5%.

Operating on sampled datasets could potentially reduce the accuracy of our exploration results. However, an interesting artifact of our exploration techniques is that their accuracy does not depend on the frequency of each attribute value, or on the presence of all available tuples of our database. This is because each phase executes *random* selections within data hyper-rectangles and hence these selections do not need to be deterministic. In the contrary, as long as the domain value distribution within these hyper-rectangles is roughly preserved, our techniques are still equally effective on the sample dataset as in the actual one (i.e., their accuracy will not be reduced significantly). Therefore, we generate our sampled data sets using a simple random sampling approach that picks each tuple with the same probability [21]. This preserves the value distribution of the underlying attribute domains and allows us to offer a similar level of accuracy but with significantly less time overhead.

6. EXPERIMENTAL EVALUATION

Next, we present experimental results from a micro-benchmark on the SDSS dataset [4] and from a user study.

6.1 Experimental Setup: SDSS Dataset

We implemented our framework on JVM 1.7. In our experiments we used various Sloan Digital Sky Survey datasets (SDSS) [4] with a size of 10GB-100GB ($3\times10^6-30\times10^6$ tuples). Our exploration was performed on combinations of five numerical attributes (`rowc`, `colc`, `ra`, `field`, `fieldID`, `dec`) of the `PhotoObjAll` table. These are attributes with different value distributions, allowing us to experiment with both skewed and roughly uniform exploration spaces. A covering index on these attributes was always used. We used by default a 10GB dataset and a dense exploration space on `rowc` and `colc`, unless otherwise noted. All our experiments were run on an Intel PowerEdge R320 server with 32GB RAM using MySQL. We used the Weka [30] library for executing the CART [8] decision tree algorithm and the k-means clustering algorithm. All experiments report averages of ten exploration sessions.

Target Queries AIDE characterizes user interests and eventually "predicts" the queries that retrieve his relevant objects. We focus on predicting range queries (*target queries*) and we vary their complexity based on: a) the number of disjunctive predicates they include (*number of relevant areas*) and b) the data space coverage of the relevant areas, i.e., the width of the range for each attribute (*relevant area size*). Specifically, we categorize relevant areas to *small*, *medium* and *large*. Small areas have attribute ranges with average width of 1-3% of their normalized domain, while medium areas have width 4-6% and large ones have 7-9%. We also experimented with queries with a single relevant area (conjunctive queries) as well as complex disjunctive queries that select 3, 5 and 7 relevant areas. The higher the number of relevant areas and the smaller these areas, the more challenging is to predict them.

The diversity of our target query set is driven by the query characteristics we observed in the SDSS sample query set [3]. Specifically, 90% of their queries select a single area, while 10% select only 4 areas. Our experiments cover even more complex cases of 5 and 7 areas. Furthermore, 20% of the predicates used in SDSS queries cover 1-3.5% of their domain, 3% of them have coverage around 13%, and 50% of the predicates have coverage 50% or higher while the median coverage is 3.4%. Our target queries have domain coverage (i.e., the relevant area size) between 1-9% and our results demonstrate that we perform better as the size of the areas increases. Hence, we believe that our query set has a good coverage of queries used in real-world applications while they also cover significantly more complex cases.

User Simulation Given a target query, we simulate the user by executing the query to collect the exact *target set* of relevant tuples. We rely on this set to label the new sample set we extract in each iteration as relevant or irrelevant depending on whether they are included in the target set. We also use this set to evaluate the accuracy (F-measure) of our final predicted extraction queries.

Evaluation Metrics We measure the accuracy of our approach using the F-measure (Equation 1) of our final extraction query and report the number of labeled samples required to reach different accuracy levels. Our efficiency metric is the *system execution time* (equivalent to *user wait time*), which include the time for the space exploration, data classification, and sample extraction. We may also report the total *exploration time*, which includes both the system execution time and the sample reviewing time by the user.

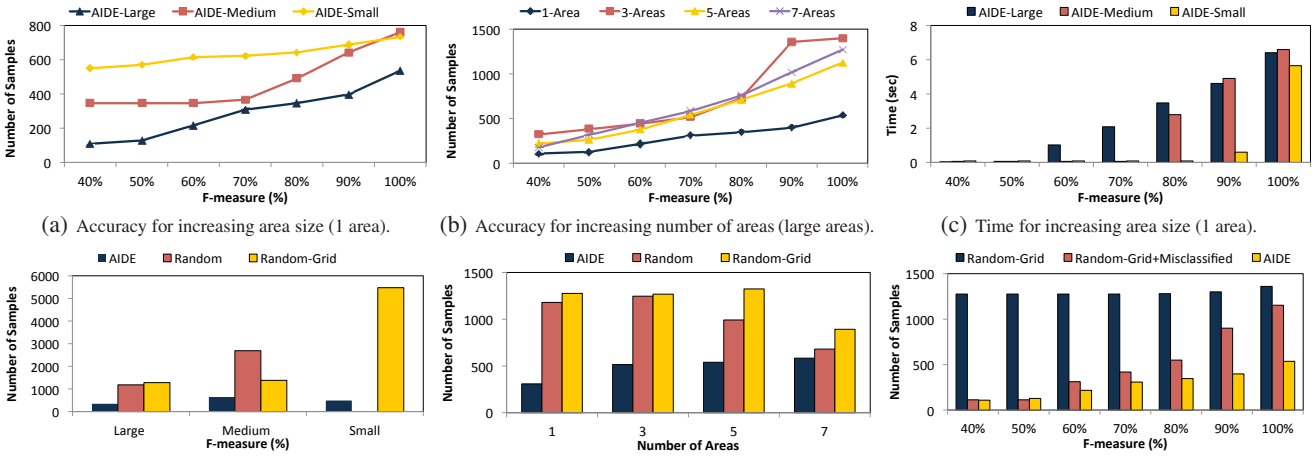
6.2 Effectiveness & Efficiency of AIDE

Figure 8(a) shows AIDE's effectiveness when we increase the query complexity by varying the size of relevant areas from large (*AIDE-Large*) to medium (*AIDE-Medium*) and small (*AIDE-Small*). Our queries have one relevant area which is the most common range query in SSDS. Naturally, labeling more samples improves in all cases the accuracy. As the query complexity increases the user needs to provide more samples to get the same level of accuracy. By requesting feedback on only 215 out of 3×10^6 objects AIDE predicts large relevant areas with accuracy higher than 60% (with 350 samples we have an accuracy higher than 80%). In this case, the user needs to label only 0.4% of the total relevant objects and 0.01% of the irrelevant objects in the database. Furthermore, AIDE needs only 345 samples to predict medium areas and 600 samples for small areas to get an accuracy of at least 60%.

We also increased the query complexity by varying the number of areas from one (1) to seven (7). Figure 8(b) shows our results for the case of large relevant areas. While AIDE can perform very well for common conjunctive queries (i.e., with one (1) relevant area), to accurately predict highly complex disjunctive queries more samples are needed. However, even for highly complex queries of seven (7) areas we get an accuracy of 60% or higher with reasonable number of samples (at least 450 samples).

Figure 8(c) shows the time overhead (seconds in average per iteration). In all cases, high accuracy requires the extraction of more samples which increases the exploration time. The complexity of the query (size of relevant areas) also affects the time overhead. Searching for larger relevant areas leads to more sample extraction queries around the boundaries of these relevant areas. However, our time overhead is acceptable: to get an accuracy of 60% the user wait time per iteration is less than one second for small and medium areas, and 1.02 second for large areas, while to get highly accurate predictions (90%-100%) the user experiences 4.79 second wait time in average. To reach the highest accuracy (> 90%) AIDE executed 23.7 iterations in average for the large areas, 37 iterations for the medium and 33.4 iterations for the small areas.

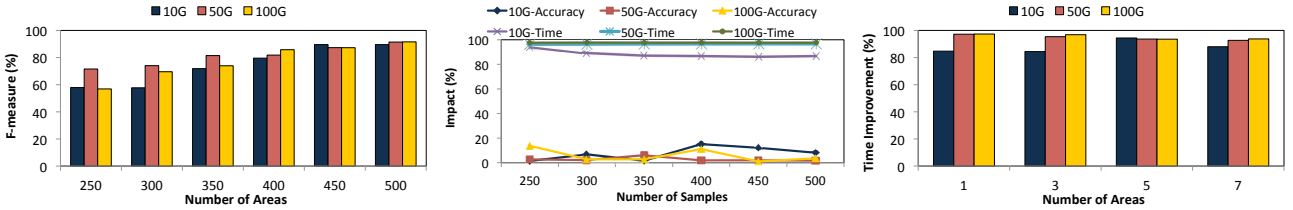
Comparison with Random Exploration Next we compared our approach (AIDE) with two alternative exploration techniques. *Ran-*



(a) Accuracy for increasing area size (1 area). (b) Accuracy for increasing number of areas (large areas). (c) Time for increasing area size (1 area).

(d) Comparison to random exploration for increasing area size (>70% accuracy, 1 area). (e) Comparison to random exploration for increasing number of areas (>70% accuracy, large areas). (f) Impact of exploration phases (1 large area).

Figure 8: Figures (a), (b) show AIDE's effectiveness, i.e., prediction accuracy. Figure (c) shows efficiency results, i.e., time overhead. Figures (d) and (e) compare AIDE with random exploration techniques. Figure (f) demonstrates the effectiveness of AIDE's exploration phases.



(a) Accuracy for increasing data set size (1 large area). (b) Impact of sampled datasets on accuracy and time overhead (1 large area). (c) Time improvement of sampled datasets and increasing number of areas (>70% accuracy, large areas).

Figure 9: AIDE's effectiveness and efficiency on big data sets.

dom randomly selects 20 samples per iteration, presents them to the user for feedback and then builds a classification model. *Random-Grid* is similar to Random but the sample selection is done on an exploration grid, i.e., it selects one random sample around the center of each grid cell. This allows our samples to be better distributed across the exploration space. This approach also collects 20 samples per iteration. AIDE also limits the number of new samples it extracts per iteration: we calculated the number of samples needed for the boundary and the misclassified exploitation and we used the remaining of 20 samples to sample unexplored yet grid cells.

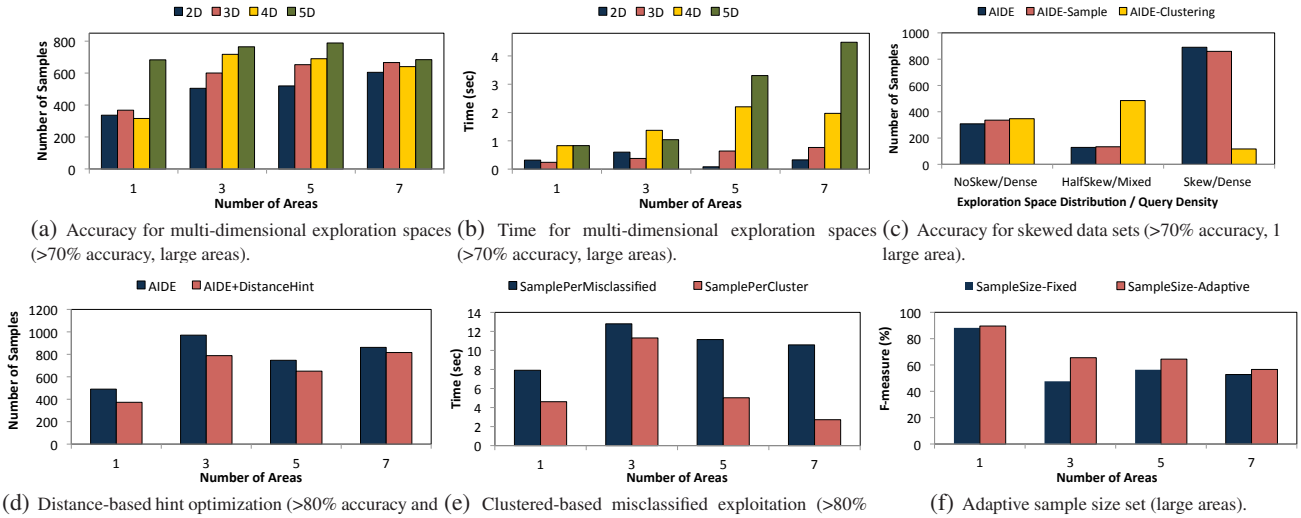
Figure 8(d) shows the number of samples needed to achieve an accuracy of at least 70% when our target queries have one (1) relevant area of different sizes. AIDE is consistently highly effective (requiring only 308 samples for large areas and 365 and 623 samples in average for medium and small samples). Both random exploration approaches cannot discover small and medium areas with that few samples. Random fails to discover small areas of interest even when we increase the labeled samples to 6,400, while Random-Grid needs 5,457 samples in average for these complex queries. Random can identify medium and large relevant areas with 70% accuracy when given at least 2,690 and 1,180 samples respectively. Random-Grid is also highly ineffective, since it needs 1,380 and 1,275 samples in average for medium and large areas. Figure 8(e) shows the number of samples to achieve at least 70% accuracy when varying the number of target relevant areas. AIDE consistently requires less samples (less than 500 samples for all cases) than Random and Random Grid (more than 1000 objects in almost all cases). AIDE outperforms Random and Random-Grid since it optimizes the sample selection process through the misclassified and boundary exploitation phases, leading to highly accurate results with less sampled data.

Impact of Exploration Phases We also studied the impact of each exploration phase independently. Figure 8(f) compares the number of samples we need to reach different accuracy levels for queries with one large relevant area. We compare AIDE with two variants: one where only the object discovery phase is used (*Random-Grid*) and one where we only add the misclassified exploitation phase (*Random-Grid+Misclassified*). The results show that combining all three phases gives the best results. Specifically, using only the object discovery phase requires consistently more than 1,000 samples to get an accuracy greater than 40%. Adding the misclassified exploitation phase reduces the sample requirements by 60% in average while adding the boundary exploitation phase allows us to achieve higher accuracy with 42% less samples in average. Hence, combining all three phases is highly effective in predicting relevant areas while reducing the amount of user effort.

6.3 Scalability

We also evaluated the scalability of our framework.

Database Size Figure 9(a) shows the accuracy we can achieve with a given number of labeled samples for dataset sizes of 10GB, 50GB and 100GB. Our target queries have one large relevant area and the average number of relevant objects increases as we increase the size of the dataset (our target query returns in average 26,817 relevant objects in the 10GB, 120,136 objects in the 50GB and 238,898 objects in the 100GB database). AIDE predicts these objects in all datasets with high accuracy without increasing the user's effort. We conclude that the size of the database does not affect our effectiveness. AIDE consistently achieves high accuracy of more than 80% on big data sets with only a few hundreds of samples (e.g., 400 samples). These results were consistent even for more complex queries with multiple relevant areas.



(a) Accuracy for multi-dimensional exploration spaces (>70% accuracy, large areas).

(b) Time for multi-dimensional exploration spaces (>70% accuracy, large areas).

(c) Accuracy for skewed data sets (>70% accuracy, 1 large area).

(d) Distance-based hint optimization (>80% accuracy and medium areas).

(e) Clustered-based misclassified exploitation (>80% accuracy and large areas).

(f) Adaptive sample size set (large areas).

Figure 10: AIDE's performance on multi-dimensional exploration spaces (Figures (a-b)) and on skewed exploration spaces (Figure (c)). Figures (d-e-f) demonstrate the effectiveness of our optimizations.

Sampled Datasets Applying our techniques to larger datasets increases the time overhead since our sampling queries have higher response times. One optimization is to execute our exploration on a sampled database (Section 5.2). In this experiment, we sampled datasets of 10GB, 50GB, 100GB and generated the 10% sampled datasets of 1GB, 5GB and 10GB, respectively. Figure 9(b) shows the absolute difference of the final accuracy (*10GB-Accuracy, 50GB-Accuracy, 100GB-Accuracy*) when AIDE is applied on the sampled and on the total datasets. The average difference is no more than 7.15% for the 10GB, 2.72% for the 50GB and 5.85% for the 100GB data set. In the same figure we also show the improvement of the system execution time (*10GB-Time, 50GB-Time, 100GB-Time*). For 10GB (and a sampled dataset of 1GB) this time is reduced by 88% in average, while for the larger datasets of 50GB and 100GB it is reduced by 96%-97%.

Figure 9(c) shows the time improvement when AIDE runs over the sampled data sets and we increase the query complexity (i.e., increase the number of relevant areas). Here, we measure the improvement of the system execution time when we reach an accuracy higher than 70% . The average time per iteration is 2.8 seconds for the 10GB, 37.7 for the 50GB and 111 for the 100GB database. By operating on the sampled datasets we improved our time by more than 84% while our average improvement for each query type was more than 91%. Our improved iteration time is 0.37 second for the 10GB, 2.14 seconds for the 50GB and 5.3 seconds for the 100GB dataset, in average. The average number of iterations is 37 and hence AIDE offers a total execution time of 13secs for the 10GB, 1.3mins for the 50GB and 3.2mins for the 50GB dataset while the user wait time is less than 3secs per iteration in average. Hence, AIDE can scale to big datasets by applying its techniques on sampled datasets. This incurs very low impact on the accuracy while it significantly improves the system execution time.

Exploration Space Size Figure 10(a) shows the number of samples to reach an accuracy greater than 70% as we increase the complexity of our queries (the number of relevant areas) and the size of our exploration space from 2-dimensional to 5-dimensional. These results are on large size areas and on the sampled datasets. Our target queries have conjunctions on two attributes and the main challenge for AIDE is to identify in the 3D, 4D and 5D spaces only the two relevant attributes. AIDE correctly identifies the irrelevant attributes and eliminates them from the decision tree classifier and hence from the final output query. Furthermore, although the exploration over more dimensions requires naturally more samples to reach an acceptable accuracy, the number of samples only increases by a small percentage (the 3D space and 4D space require in average 13% more tuples than the 2D space and the 5D space requires 32% more tuples than the 2D space) and they remain within the range of 100's even for the complex cases of 7 areas and 5-dimensional exploration space. Figure 10(b) shows that even for the very complex case of seven (7) relevant areas the time overhead is always less than 4.5 seconds, while for the less complex queries of 1 area the time drops below 1 second. These results reveal a small increase in the user's wait time as we add more dimensions (each new dimension adds in average 0.70 second overhead to the previous one) but always within acceptable bounds.

6.4 Effectiveness of Optimizations

Next we study the impact of our proposed optimizations.

Handling Skewed Attributes We also studied AIDE in the presence of skewed exploration spaces. We experimented with three types of 2-dimensional exploration spaces: (a) *NoSkew* where we use two non-skewed attributes (`rowc`, `colc`), (b) *HalfSkew* that includes one skewed (`dec`) and one non-skewed attribute (`rowc`) and (c) *Skew* that uses two skewed attributes (`dec`, `ra`). Figure 10(c) shows the number of samples needed to achieve accuracy greater than 70% for queries with one large relevant area. For the Skew and NoSkew case this area is located on a dense region, while for the HalfSkew case we experimented with queries that cover both sparse and dense areas. We compare three variants: (a) AIDE that uses the grid-based technique for the relevant object discovery phase, (b) *AIDE-Clustering* that uses the clustering-based optimization for skewed distributions (Section 3.1), and (c) *AIDE-Sample* that uses the grib-based approach on a sampled database.

The results show that the clustering optimization requires 87% less samples than the grid-based approach when predicting dense relevant areas within highly skewed data spaces. This is because both the cluster and our relevant areas are concentrated in dense sub-areas while grid cells are created uniformly across the data space. This allows AIDE-Clustering to sample smaller, finer-grained areas than the grid-based approach, eliminating the need to zoom into the next exploration level. When the distribution is uniform (NoSkew) clusters and grid cells are highly aligned pro-

viding roughly the same results. In the hybrid case (HalfSkew) our relevant areas and our grid-cells cover both dense and sparse areas. The clustering technique though creates most of its clusters on the dense areas and hence fails to discover relevant objects in the sparse ones. It therefore has to zoom in into finer exploration levels and it required 73% more samples to converge to the same accuracy as the grid-based technique. This result indicates that a hybrid approach could be the best strategy for the relevant object discovery phase. AIDE would be initialized with the clustered approach to explore first dense areas. When the users interests are partially revealed the system could switch to the grid-based approach if these interests appear to lie on sparse areas. We plan to explore this in a future extended version of the paper.

Our experiment also revealed AIDE and AIDE-Sample have similar performance regardless of the underlying data distribution. This is because our sampled dataset preserves the distribution of the exploration domains. This result is consistent for queries of different complexity. We conclude that sampled datasets do not affect our accuracy even in the presence of skewed data spaces.

Distance-based Hints The user can optionally specify a lower bound for the sizes of elevant areas (see Section 5.2). In Figure 10(d) we show the results when the user has specified that the area width along each dimension will be at least 4% on the normalized domains (*AIDE+DistanceHint*). We compare it with the regular AIDE with no hints. These results are on medium relevant areas and we vary the number of areas. AIDE+DistanceHint performs better in all cases: to reach an accuracy higher than 80% we need in average 656 samples which is 14% less samples than AIDE. The hint allows us to know the exact exploitation level to use in order to guarantee that from the first iteration the object discovery phase will "hit" all relevant areas. Hence, it eliminates the need to spend samples exploring more fine-grained exploration levels.

Clustered-based Misclassified Exploitation Next we compare the time overhead when the clustering-based misclassified exploitation is used (*SamplePerCluster*) with the approach that defines one sampling area for each misclassified object (*SamplePerMisclassified*). Here, we used queries with a different number of large relevant areas and we show the exploration time for reaching an accuracy of at least 80%. Figure 10(e) demonstrates that the clustering approach can improve the time overhead by 45.6% in average, since it creates one sampling area (i.e., issues one sample extraction query) per cluster. We note that the accuracy was not affected by incorporating this optimization (we needed in average 2% more tuples (15 tuples) to reach the same F-measure).

Adaptive sample size In Figure 10(f) we compare the accuracy when keeping the sample size in the boundary exploitation phase fixed (*SampleSize-Fixed*) with adapting the size based on the changes of the decision tree between iterations (*SampleSize-Adaptive*). Our queries select an increasing number of disjoint large areas and we report the accuracy we achieve when the user labels 500 samples. We can observe that our accuracy improved by an average of 12%. This is due to the fact that our strategy reduces the number of samples we collect through boundary exploitation, therefore requesting feedback on more samples collected by the other two phases. These two phases (relevant object discovery and misclassified exploitation) have a higher impact on the F-measure, therefore our accuracy is increased.

6.5 User Study Evaluation

Our user study used the AuctionMark dataset [1] that includes information on auction items and their bids. We chose this "intuitive" dataset in the user study, as opposed to the SDSS dataset. This is because the user study requires identifying a significant number of users with sufficient understanding of the domain. Thus, AuctionMark meets the requirement: we were able to identify a group of computer science graduate students with SQL experiences and designed their exploration task to be "identifying auction items that are good deals". Note that the exploration task should not be trivial, i.e., users should not have an upfront understanding of the exact selection predicates that would collect all relevant objects.

User	Manual: returned objects	Manual: reviewed objects	AIDE: reviewed objects	Reviewing savings (%)	Manual: time (min)	AIDE: time (min)
1	253,461	312	204.9	**34.3%**	60	39.7
2	656,880	160	82.4	**48.5%**	70	36.3
3	933,500	1240	157	**87.3%**	60	7.9
4	180,907	600	319	**46.8%**	50	28.2
5	2,446,180	650	288.5	**55.6%**	60	27.5
6	1,467,708	750	334.5	**55.3%**	75	33.8
7	567,894	1064	288.4	**72.8%**	90	24.8

Table 1: User study results.

The exploration data set had a size of 1.77GB and it was derived from the ITEM table of AuctionMark benchmark. It included seven attributes: initial price, current price, number of bids, number of comments, number of dates an item is in an auction, the difference between the initial and current item price, and the days until the auction is closed for that item. Each user explored the data set "manually", i.e., iteratively formulating exploratory queries and reviewing their results until they obtained a query, Q, that satisfied their interests. We then took Q as the true interest of a user and used it to simulate user labeling results in AIDE. We measured how well AIDE can predict the user query Q.

The results demonstrated that AIDE was able to reduce the user's reviewing effort by 66% in average (*Reviewing savings* column in Table 1). Furthermore, with the manual exploration users were shown 100s of thousands objects in total (*Manual returned objects*) while AIDE shows them only a few hundred strategically selected samples. Furthermore, with the manual exploration our users needed about an hour to complete their task (*Manual time*). Assuming that the most of this time was spent on tuple reviewing, we calculated the average tuple reviewing for each user. This varied significantly across users (3secs - 26secs). Using this time we estimated the total exploration time needed by AIDE including the reviewing effort (*AIDE time*). AIDE was able to reduce the exploration time 47% in average. We believe these time savings will be even more pronounced for more complex exploration tasks (e.g., in astronomical or medical domains) where examining the relevance of an object requires significant time.

Our user study revealed that five out of the seven users used only two attributes to characterize their interests while the rest needed three, four and five attributes. Similarly to our SDSS workload, the most common type of query was conjunctive queries that selected a single relevant area. Our exploration domain was highly skewed and all our relevant areas were on dense regions. These characteristics indicate that our micro-benchmark on the SDSS dataset was representative of common exploration tasks while it also covered highly more complex cases, i.e., small relevant areas and disjunctive queries selecting multiple areas. More details on our user study can be found in [12].

7. RELATED WORK

Query by Example Related work on "Query-By-Example" (QBE) (e.g., [33]) focused on minimizing the burden to remember the finer details of SQL by translating user actions, such as assigning a value to an attribute, to query statements. In [7] they also propose a graphical visualization of the database that allows users to formulate queries with widgets. These systems provide alternative front-end interfaces and do not attempt to understand

user interests nor retrieve "similar" data objects. In [5] they learn user queries based on given value assignments used in his intended query, which are assumptions that are made in our work.

Data Exploration Numerous recent research efforts focus on data exploration. Our vision for automatic, interactive navigation in databases was first introduced in [9]. AstroShelf [20] allows users to collaboratively annotate and explore sky objects while YMALDB [13] recommends to users data similar to their query results. DICE [15] supports highly efficient exploration of data cubes using faceted search. SciBORQ [29] relies on hierarchical database samples to support scientific exploration queries within strict query execution times. Blink [6] relies on run-time sample selection to provide real-time answers with statistical error guarantees. Idreos et al. [16] envision a system for interactive data processing tasks aiming to reduce the time spent on data analysis. Finally, [27] interactively explores the space based on statistical properties of the data and provides suggestions for further exploration. These systems are different than AIDE: we rely on the user's feedback to provide query suggestions and we focus on collecting samples that improve our understanding of the user's interests.

Query Relaxation Query relaxation techniques have also been proposed for supporting exploration in databases [11]. In [17, 19] they refine SQL queries to satisfy cardinality constraints on the query result. In [14] they rely on multi-dimensional histograms and distance metrics for range queries for accurate query size estimation. These solutions are orthogonal to our problem; they focus on adjusting the query parameters to reach a cardinality goal and therefore cannot characterize user interests.

Active Learning The active learning community has proposed solutions that maximize the learning outcome while minimizing the number of samples labeled by the user [23, 26]. However, these techniques assume either small datasets or negligible sample extraction costs which is not a valid assumption when datasets span 100s of GBs and interactive performance is expected. Relevance feedback have been studied for image retrieval [22], document ranking [24], information extraction and segmentation [28] and word disambiguation [32]. All these solutions are designed for specific data types (images or text) and do not optimize for efficient sample acquisition and data space exploration.

Collaborative and Interactive Systems In [18] a collaborative system is proposed to facilitate formulation of SQL queries based on past queries and in [10] they use collaborative filtering to provide query recommendations. However, both these systems do not predict "similar" data tuples. In [25] they cluster related queries as a means of understanding the intents of a given user query. The focus is on web searches and not structured databases.

8. CONCLUSIONS

In this paper, we present AIDE, an *Automatic Interactive Data Exploration* framework that assists users in discovering new interesting data patterns and eliminate expensive ad-hoc exploratory queries. AIDE relies on a seamless integration of classification algorithms and data management optimization techniques that collectively strive to accurately learn the user interests based on his relevance feedback on strategically collected samples. Our techniques minimize the number of samples presented to the user (which determines the amount of user effort) as well as the cost of sample acquisition (which amounts to the user wait time). AIDE can deliver highly accurate query predictions for very common conjunctive queries with small user effort while, given a reasonable number of samples, it can predict with high accuracy complex disjunctive queries. It provides interactive performance as it limits the user wait time per iteration of exploration to less than a few seconds. Our user study also shows that AIDE improves the current state-of-the-art of manual exploration by significantly reducing the user effort and total exploration time.

In future work, we plan to improve our system by extending the set of target queries beyond conjunctive queries and linear predicates, for instance, to include non-linear predicates, as well as further optimizing the database backend for exploration query workloads using materialized views and multi-query optimization.

9. ACKNOWLEDGMENTS

This work was funded by in part by NSF under Grant IIS-1049974, IIS-1253196 and IIS-1218524.

10. REFERENCES

[1] AuctionMark Benchmark, http://hstore.cs.brown.edu/projects/auctionmark/.
[2] Large Synoptic Survey Telescope, http://http:/www.lsst.org/.
[3] SDSS Samples Queries, http://cas.sdss.org/dr4/en/help/docs/realquery.asp.
[4] Sloan Digital Sky Survey, http://www.sdss.org/.
[5] Abouzied et al. Learning and verifying quantified boolean queries by example. In *PODS*, 2013.
[6] Agarwal et al. Blink and it's done: interactive queries on very large data. In *VLDB*, 2012.
[7] Ahlberg et al. Dynamic queries for information exploration: an implementation and evaluation. In *CHI*, 1992.
[8] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. 1984.
[9] Cetintemel et al. Query Steering for Interactive Data Exploration. In *CIDR*, 2013.
[10] G. Chatzopoulou, M. Eirinaki, and N. Polyzotis. Query Recommendations for Interactive Database Exploration. In *SSDBM*, 2009.
[11] S. Chaudhuri. Generalization and a framework for query modification. In *ICDE*, 1990.
[12] Dimitriadou et al. Explore-by-Example: An Automatic Query Steering Framework for Interactive Data Exploration. Technical Report CS-13-284, Brandeis University, 2013.
[13] M. Drosou and E. Pitoura. YMALDB: exploring relational databases via result-driven recommendations. *VLDB Journal*, 22:849–874, 2013.
[14] A. Kadlag, A. V. Wanjari, J. Freire, and J. R. Haritsa. Supporting Exploratory Queries in Databases. In *DASFAA*, 2004.
[15] Kamat et al. Distributed and Interactive Cube Exploration. In *ICDE*, 2014.
[16] Kersten et al. The Researcher's Guide to the Data Deluge: Querying a Scientific Database in Just a Few Seconds. *PVLDB*, 4(12):1474–1477, 2011.
[17] C. Mishra and N. Koudas. Interactive query refinement. In *EDBT*, 2009.
[18] N. Khoussainova et al. A Case for A Collaborative Query Management System. In *CIDR*, 2009.
[19] N. Koudas et al. Relaxing join and selection queries. In *VLDB*, 2006.
[20] Neophytou et al. AstroShelf: Understanding the Universe through Scalable Navigation of a Galaxy of Annotations. In *SIGMOD*, 2012.
[21] F. Olken and D. Rotem. Random sampling from databases - a survey. *Statistics and Computing*, 5, 1994.
[22] N. Panda, K.-S. Goh, and E. Y. Chang. Active learning in very large databases. *Multimedia Tools Appl.*, 31(3):249–267, 2006.
[23] N. Roy and A. McCallum. Toward optimal active learning through sampling estimation of error reduction. In *ICML*, 2001.
[24] Ruthven et al. A survey on the use of relevance feedback for information access systems. *The Knowledge Engineering Review*, 18(2):95–145, 2003.
[25] E. Sadikov, J. Madhavan, L. Wang, and A. Halevy. Clustering query refinements by user intent. In *WWW*, 2010.
[26] S. Sarawagi and A. Bhamidipaty. Interactive Deduplication Using Active Learning. In *KDD*, 2002.
[27] T. Sellam and M. L. Kersten. Meet Charles, big data query advisor. In *CIDR*, 2013.
[28] B. Settles and M. Craven. An analysis of active learning strategies for sequence labeling tasks. In *EMNLP*, 2008.
[29] L. Sidirourgos, M. Kersten, and P. Boncz. SciBORQ: Scientific data management with Bounds On Runtime and Quality. In *CIDR*, 2011.
[30] Witten et al. Weka: Practical Machine Learning Tools and Techniques with Java Implementations, 1999.
[31] X. S. Zhou and T. Huang. Relevance Feedback in Image retrieval: A comprehensive review. *Multimedia System*, 8(2):95–145, 2003.
[32] J. Zhu. Active Learning for Word Sense Disambiguation with Methods for Addressing the Class Imbalance Problem. In *ACL*, 2007.
[33] M. M. Zloof. Query-by-example: the invocation and definition of tables and forms. In *VDLB*, 1975.