Discovering Interesting Subsets Using Statistical Analysis

Maitreya Natu Girish K. Palshikar

Tata Research Development and Design Centre
Pune, MH, India, 411013
Email: {maitreya.natu, gk.palshikar}@tcs.com

Abstract

In this paper we present algorithms for identifying interesting subsets of a given database of records. In many real life applications, it is important to automatically discover subsets of records which are interesting with respect to a given measure. For example, in the customer support database, it is important to identify subsets of tickets having service time which is too large (or too small) when compared with the service time of the rest of the tickets. We use Student's t-test to check whether the measure values for a subset and its complement differ significantly. We first discuss the brute-force approach and then present heuristic-based state-space search algorithm to discover interesting subsets of the given database. To use the proposed heuristic-based approach on large data sets, we then present a sampling-based algorithm that uses sampling together with the proposed heuristics to efficiently identify interesting sets in large data sets.

We discuss an application of these techniques to customer support data, to discover subsets of tickets that have significantly worse (or better) service times than the rest of the tickets.

1 Introduction

In this paper, we consider the problem of finding *interesting subsets* of a given relational table of records. First, there is a need to formalize a domain-independent notion of interestingness of a given subset of records. In many real-life applications, the given table includes one or more numeric attributes each of which is a quantitative measure for the record. Examples:

1. In a customer-support database of tickets of problems and their resolutions, the service time needed to resolve each ticket.

2. In a database of customer (or employee) satisfaction survey responses, the satisfaction index for each customer (or employee).

3. In a database of sales orders, the value, quantity and the time taken to fulfil each order.

In each of these applications, it is important to automatically discover subsets of records which are interesting with respect to a given measure. For example, in the customer support database, it is important to identify subsets of tickets having service time which is too large (or too small) when compared with the service time of the rest of the tickets. Such subsets of tickets provide insights for improving the business processes involved in resolving tickets: identify bottlenecks, identify areas for improvement, increase per-person productivity, etc.

Unlike anomaly detection, the focus is on finding interesting *subsets*, rather than individual interesting records. Two central questions arise: (i) how to construct subsets of the given records; and (ii) how to decide whether a given subset of records is interesting or not. We present algorithms to automate both these steps. Finding interesting subsets of records in a given table is often an important part of exploratory data analysis in practice. The user typically uses SQL-like `SELECT` command to form a subset of the given records and then checks whether or not this subset is interesting. SQL-like `SELECT` commands provide an intuitive way for the end-user to characterize and understand a subset of records. Further, the user can interactively refine the definition of the subset by adding or removing conditions in the `WHERE` clause. Our algorithms systematically explore subsets of records of a given table by increasingly refining the condition part of the `SELECT` command.

The problem of identifying interesting subsets is quite different from the usual top-k heavy hitters analysis. In the latter approach, the records are sorted with respect to the chosen measure and then k records at the top (or at the bottom) are returned. Thus *each* record in the top-k subset has an unusual (high or low) value for the measure. In contrast, we wish to identify *common characteristics* of the records in the interesting subsets (rather than the individual interesting records) and then use these patterns for purposes such as designing improvements. Since *every* record identified by the top-k approach has an unusual measure value, in general, it is difficult to see if there is any discernible pattern among these records.

International Conference on Management of Data
COMAD 2008, Mumbai, India, December 17–19, 2008

We propose the following approach. Assume that a subset A of the database D is given. Let $\overline{A} = D - A$ denote the complement of the subset A in D i.e., $\overline{A}$ consists of all records in D which are not in A. Let $\Phi(A)$ denote the (multi)set of the measure values for the records in the subset A. In the customer support example, $\Phi(A)$ is the (multi)set of the values of service times of all tickets in A. We say A is an *interesting subset* of D if the statistical characteristics of the subset $\Phi(A)$ are very different from the statistical characteristics of the subset $\Phi(\overline{A})$. In the customer support example, a given subset A of tickets would be interesting if the service times of tickets in A are in general very different from the service times of the rest of the tickets in $\overline{A}$.

More formally, A is an *interesting subset* of D if the probability distribution of the values in the subset $\Phi(A)$ is very different from the probability distribution of the values in the subset $\Phi(\overline{A})$. Essentially, we consider the subsets A and $\overline{A}$ (and hence correspondingly, the subsets $\Phi(A)$ and $\Phi(\overline{A})$ of their measure values) as two samples and use statistical hypothesis testing techniques to decide whether or not the observed differences between them are statistically significant. Many statistical tests are available to test whether or not the two probability distributions (one for $\Phi(A)$ and the other for $\Phi(\overline{A})$) are significantly different. We choose the Student's t-test for this purpose, although other tests (e.g., Kolmogorov-Smirnov test) could be used instead. Note that we compare the *overall* statistical characteristics of the values in subsets $\Phi(A)$ and $\Phi(\overline{A})$. Thus we focus on interesting subsets of tickets, rather than on individual interesting tickets themselves.

The brute-force approach to identify interesting subsets is now clear. Systematically generate subsets A of D and use the t-test to check whether or not the subsets $\Phi(A)$ and $\Phi(\overline{A})$ of their measure values are statistically different. If yes, report A as interesting. Clearly, this approach is not scalable for large datasets, since a subset of N elements has 2^N subsets. We propose a two-pronged strategy to limit the exploration of the state-space of all possible subsets of the given database D.

1. We impose a restriction that the discovered interesting subsets need to be described succinctly for the end-user. Hence, instead of considering all subsets of D, we consider only those subsets that can be described using SQL-like expressions of the form
   `SELECT * FROM` D `WHERE` $A_1 = v_1$ `AND` $A_2 = v_2$ `AND ... AND` $A_k = v_k$
   where attributes A_i are all different and v_i denotes a possible value for attribute A_i. Restriction to subsets of records that have a single value for some attributes is not a severe restriction. We adopt it here to simplify the presentation.

2. We impose additional restrictions (described later) to eliminate certain subsets from consideration; e.g., eliminate subsets that are "too small".

In this paper, we describe a heuristic-based pruning algorithm that examines the state-space of the subsets of the given database D and discovers and reports all interesting subsets. Each state is a subset of the given set of records and is characterized by a `SELECT` command. Children of a state S are formed by refining the condition used to form S. The crux of the algorithm is in the heuristics used to prune the state-space.

To apply the proposed algorithms on large data sets the heuristic-based approach might also require an unacceptably large execution time. To improve the efficiency of the proposed heuristic-based approach, we then propose a sampling-based algorithm, where we run the proposed algorithm on randomly picked samples of the data set and process the results of the algorithm to infer the properties of the entire data set.

To make the discussion more concrete and to provide an illustration of how the results of this algorithm can be used in practice, we focus in the rest of the paper on a database of customer support tickets. Each ticket has attributes like timestamp-begin, timestamp-end, priority, location, resource, problem, solution, solution-provider, etc. and service-time. We focus on the problem of discovering interesting subsets of tickets that have very high (or low) service times, as compared with the rest of the tickets. However, we emphasize that the techniques of discovering interesting subsets are perfectly general and can be applied to any database where a quantitative measure is available for each record.

The rest of the paper is organized as follows. Section 2 describes the algorithm to discover interesting subsets. Section 3 presents experimental results. Section 4 contains related work and section 5 contains conclusions and further work.

2 Interesting Subset Discovery (ISD)

We first explain the process of building subsets of records to check for *interestingness*. We then describe a brute-force algorithm to systematically explore all possible subsets of records and identify the interesting ones. Since the search space of subsets could be very large, we then present different pruning heuristics to reduce the number of subsets examined.

2.1 Building subsets

Each record is associated with a finite set of n attributes $A = \{A_1, \ldots, A_n\}$; e.g., attributes of customer support records may be $\{PR, CT, AC, AI\}$ representing priority, category, affected city and affected item. Let D_i denote the domain of possible values for attribute A_i. We assume each D_i to be a finite set of discrete values; e.g., $D_{PR} = \{L, M, H\}$.

A *descriptor tuple* has the form (A_i, v_i) where $A_i \in A$ is an attribute and $v_i \in D_i$ is a value for it. A *descriptor* is a set of descriptor tuples which contains at most 1 descriptor tuple for any particular attribute. A descriptor

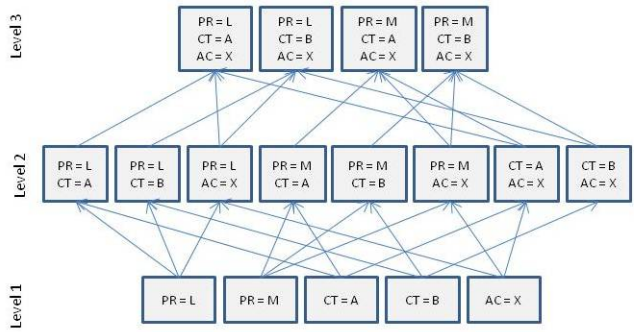

Figure 1: The state space of record attributes

corresponds to a subset of records selected using the corresponding `SELECT` statement; e.g. the descriptor {*(PR, L),(AC, 'New York')*} corresponds to the subset of records selected using

```
SELECT * FROM D WHERE PR = L AND AC =
'New York'
```

Let $\Psi(\theta)$ denote the subset of records corresponding to the descriptor θ. Then the subset relation imposes a natural partial order on the collection of all descriptors, and equivalently, on the subsets corresponding to descriptors. Thus, $\theta_1 \subseteq \theta_2 \Rightarrow \Psi(\theta_2) \subseteq \Psi(\theta_1)$. That is, refining a descriptor leads to refinement of the corresponding subset.

The number of attributes in a descriptor is called its *level*; e.g., level of the descriptor {*(PR,L),(AC= 'New York')*} is 2. We build the subset of records based on the attributes and the attribute values in a hierarchical manner. With increasing *level* in the hierarchy, a larger number of attributes are considered in the subset construction. At the first level, we build subsets based on the value(s) of a single attribute. In the next level, we refine the subsets built in the previous level by building subsets based on the value(s) of two attributes. The subset thus built is smaller than the original subset. We perform this grouping up to a predefined number of levels or until the subset sizes become too small to be significant.

The subsets built at level 1 are those that correspond to the descriptors $\{(A_i = u)\}$ for each attribute $A_i \in A$ and for each value $u \in D_i$. The subsets built at level 2 are those that correspond to the descriptors $\{(A_i = u), (A_j, v)\}$ for each pair of distinct attributes $A_i, A_j \in A$, for each value $u \in D_i$ and for each value $v \in D_j$. Figure 1 shows the subset state space up to level = 3 and built over 3 attributes PR, CT, and AC where $D_{PR} = \{L, M\}$, $D_{CT} = \{a, b\}$, and $D_{AC} = \{x\}$.

2.2 *interestingness* of subsets

We define a subset A of records as *interesting* if the corresponding subset $\Phi(A)$ of measure values is significantly different from the subset $\Phi(\overline{A})$ where $\overline{A} = D - A$ is the subset of all remaining records in the database D. We use the Student's t-test to check whether the two subsets $\Phi(A)$ and $\Phi(\overline{A})$ differ significantly in terms of their statistical characteristics.

The Student's t-test makes a *null hypothesis* that the means of the two sets do not differ significantly from eachother. Let X and Y be the two sets of numbers ($\Phi(A)$ and $\Phi(\overline{A})$ in our case). Let n_1 and n_2 denote the sizes of sets X and Y, $\bar{X}$ and $\bar{Y}$ denote the means of the values in X and Y, and S_X, S_Y denote the unbiased estimators of the standard deviations of the values in X and Y. The t-statistic for two unpaired sets X and Y assumes unequal sizes and unequal variances and tests whether the means of the two sets are *statistically* different and is computed as follows:

$$t = (\bar{X} - \bar{Y})/\sqrt{(S_x^2/n_1 + S_y^2/n_2)}$$

The denominator is a measure of the variability of the data and is called the *standard error of difference*. Another quantity called the p-value is also calculated. The p-value is the probability of obtaining the t-statistic more extreme than the observed test statistic under null hypothesis. If the calculated p-value is less than a threshold chosen for statistical significance (usually 0.05), then the null hypothesis is rejected; otherwise the null hypothesis is accepted. Rejection of null hypothesis means that the means of two sets do differ significantly. Student's t-test is a robust test and works effectively even if the normality assumption of the data is violated.

The computed t-value is positive if the mean of the first subset is larger than that of the second subset and negative if smaller. We use this property to identify subsets of records whose measure values are significantly smaller (or greater) than the rest of records. For customer support records data, *Service Time* is a natural performance metric. Hence subsets of records with positive t-values have service times significantly greater than the rest of the records and are thus performing worse. The subsets of records with negative t-values have service times significantly smaller than the rest of the records and are thus performing better than the rest of the records.

2.3 Heuristic-based ISD algorithm

In the brute force algorithm, the search space for identifying interesting attribute combinations would be very large. This search space becomes formidable for application of the algorithm on large data sets and results in very large execution time. In this section, we present various heuristics to prune the search space without affecting the effectiveness of the algorithm. We later show through simulation results that the proposed heuristics significantly prune down the search space and yet maintain high coverage and accuracy in identifying interesting subsets.

- The size heuristic: The *t-test* results on the subsets with very small size can be noisy leading to incorrect inference of interesting subsets. Small subset sizes are not able to capture the properties of the record attributes represented by the subset. Thus by the size heuristic we apply a threshold M_s and do not explore the subsets with size less than M_s.

- The goodness heuristic: While identifying interesting subsets of records that have performance values greater than the rest of the records the subsets with the performance values lesser than the rest of the records can be pruned. As we are using the case of identifying the records that perform significantly worse than the rest of the records in terms of the *service time*, we refer to this heuristic as the goodness heuristic. By the goodness heuristic, if a subset of records show significantly better performance than the rest of the records then we prune the subset. We define a threshold M_g for the goodness measure. Thus, in the case of the customer support tickets database with *service time* as the performance measure, a subset is pruned if the *t-test* result of the subset has a *t-value* < 0 and a *p-value* $< M_g$.

- The p-prediction heuristic: A level k subset is built from two subsets of level $k-1$ which share a common $k-2$ level subset and the same domain values for each of the $k-2$ attributes. The p-prediction heuristic prevents combination of two subsets that are statistically very different, where the statistical difference is measured by the *p-value* of the *t-test*. We observed that if the two level $k-1$ subsets are statistically different mutually, then the corresponding level k subset built from the two sets is likely to be less different from the rest of the data. Consider two level $k-1$ subsets S_1 and S_2 of the database D. Let $\Phi(S_i)$ represent the performance measure values of the set S_i. Let the *p*-values of the *t-test* ran on performance data of these subsets and that of the rest of data are p_1 and p_2 respectively. Let p_{12} be the mutual p-value of the *t-test* ran on the performance data $\Phi(S_1)$ and $\Phi(S_2)$. Let S_3 be the level k subset built over the subsets S_1 and S_2 and p_3 be the p-value of the *t-test* ran on the performance data $\Phi(S_3)$ and $\Phi(\overline{S_3})$, where $\overline{S_3} = D - S_3$. Then the p-prediction heuristic states that $if(p_{12} < M_p)$ *then* $p_3 > min(p_1, p_2)$, where M_p is the threshold defined for the p-prediction heuristic. We hence do not explore the set S_3 if $p_{12} < M_p$. We verified this property by experimental analysis. We defined accuracy of the p-prediction heuristic as the ratio of the number of subset pairs with mutual p-value less than M_p that hold the p-prediction heuristic property over the total number of subset pairs with mutual p-value less than M_p. We present the experimental evaluation of the accuracy of the p-prediction heuristic in Figure 2. Figure 2 plots the accuracy of the p-prediction heuristic for experiments ran for different levels of attribute sets. Figure 2 shows very high values (95%) of the p-prediction accuracy for all levels.

  Based on the above explained heuristics, we present Algorithm ISD_H for discovery of interesting subsets in an efficient manner. As explained in Figure 1 and Section 2.1, we build a level *k* subset from the subsets at level *k-1*. A level *k* subset is associated with a *k*-tuple descriptor that represents the *k* attribute-value pairs used for selection of the records to build the subset. The algorithm searches for appropriate level $k-1$ descriptors that can be combined to build a level k descriptor. A level *k-1* descriptor can be combined to another level *k-1* descriptor that has exactly one different attribute-value pair. For instance, consider a level 2 descriptor with 2 attribute-value pairs: {*PR=L, CT=A*}. This subset can be combined with a subset defined by the attribute value pairs: {*PR=L, AC=X*}. The combination of the two descriptors gives a level 3 descriptor defined by 3 attribute-value pairs: {*PR=L, CT=A, AC=X*}.

  Before combining two subsets, the algorithm applies the p-prediction heuristic and skips the combination of the subsets if the mutual *p*-value of the two subsets is less than the threshold M_p. The subsets that pass the p-prediction heuristic test are processed further to identify records with the attribute-value pairs represented by the subset descriptor. The interestingness of this subset of records is computed by applying the *t-test*. Similar to Algorithm ISD_BF, the interesting subset descriptors are identified in the result subset R^+ and R^-.

  The algorithm then applies the size and goodness heuristic on the level k subset descriptors to decide if the subset descriptor should be used for building subset descriptors in subsequent levels. The worst case computational complexity of the algorithm is exponential but the heuristics effectively reduce the average computational time.

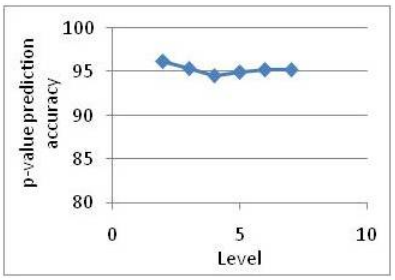

Figure 2: Accuracy of the p-value prediction

2.4 Heuristic-based ISD algorithm using sampling

Algorithm ISD_H reduces the search space as compared to the brute force based algorithm ISD_BF. But for very large data set (in the order of millions of records) the search space for algorithm ISD_H can also be large leading to unacceptable execution time. In this section we propose algorithm ISD_HS that identifies interesting subsets by performing sampling of the data set and using the heuristics proposed in the previous section on the samples.

The algorithm ISD_HS is based on the following observations:

- A large number of interesting sets discovered in a data set might not be very useful for the user. An exam-

Algorithm 1: Algorithm ISD_H

input : A = Set of record attributes, D = Domain of the values of record attributes, T = Set of records, C = Significance level, L = Max. level (number of attributes in a descriptor)
output : I = Set of positive and negative interesting record properties

```
1  n = |A|;
2  R+ = R− = ∅;
3  S0 = φ
4  for l = 1 to L do
5      foreach subset Si ∈ Sl−1 do
6          foreach subset Sj ∈ {Sl−1 - Si} do
7              if combinationValidity(Si, Sj) == FALSE then
8                  continue;
9              end
10             if MutualPValue(Si, Sj) < Mp then
11                 continue;
12             end
13             θ = Combine(Si, Sj);
14             θ consists of l attribute-value pairs
               A1 = a1 , A2 = a2 , . . . , Al = al;
15             T1 = SELECT * FROM T WHERE
               A1 = d1 , A2 = d2 , . . . , Alevel = dlevel;
16             T1 = T - T1;
17             Obtain the p value and t value by running t test
               on the subsets Φ(T1) and Φ(T2);
18             if p value < (C) then
19                 if t value > 0 then
20                     R+ = R+ ∪ θ;
21                 end
22                 else
23                     R− = R− ∪ θ;
24                 end
25             end
26             if |T1| < Ms then
27                 Prune the subset descriptor θ; continue;
28             end
29             if (t value < 0) and (p value < Mg) then
30                 Prune the subset descriptor θ; continue;
31             end
32             Add θ to Sl;
33         end
34     end
35 end
36 Sort the result sets R+ and R− based on the p value;
```

ple application of the interesting subset discovery is on the data set consisting of customer support tickets served for an enterprise system. In this domain the interesting subset discovery algorithm can be used to identify records with very large service times as compared to the rest of the records. In this example a large number of discovered interesting sets will not be very useful. The enterprise manager would be more interested in a small set of interesting subsets that can give major insight into the functioning and improvement of the system.

- Out of all the interesting subsets the subsets that have maximum impact on the overall system performance are of more importance. The properties of such subsets provide insights for system improvement that can provide maximum impact. Continuing the example of customer support tickets for an enterprise system, identification of properties that represent large number of records with poor performance are more interesting for the manager. On identification of these properties, the manager can focus on the inferred properties and improve appropriate subsystems resulting in significant system improvement.

Using the above observations we propose the algorithm ISD_HS. We propose of take samples of original data set of records and execute the algorithm ISD_H on these samples. The interesting subsets computed by running the ISD_H algorithm on these samples are combined. Note that the interestingness is computed with respect to the sample and not the original data set. The subsets thus obtained are ranked based on the number of occurrences of an interesting subset in the results of different samples. The larger the number of occurrences higher is the rank of the subset. If the number of occurrences of a subset is less than some predefined threshold, then that subset is removed from the result.

The rational behind this approach is based on the above explained observations. In case of very large data sets the ISD_H algorithm can have large execution time. In such data sets running the ISD_H algorithm on smaller samples randomly chosen from the data set is much faster. As a randomly chosen sample might not be able to capture all the prominent interesting properties of the data set, we run the ISD_H algorithm on multiple samples of the data set. In order to identify important subsets out of all the discovered interesting subsets we rank the subsets based on the number of occurrences in results of different samples. If the original data set has a large interesting subset then the records of this subset are very likely to be present in the randomly chosen samples. With the presence of larger number of such records, the algorithm is very likely to identify the interesting subset even in the randomly picked sample. A subset that is more prominent in the original data set is more likely to be discovered in larger number of samples. Algorithm ISD_HS thus decreases the number of discovered interesting sets by limiting the results only to the frequently occurring interesting subsets in the random chosen samples.

The pseudocode for the algorithm ISD_HS presents the steps involved. Algorithm ISD_H is run on randomly chosen samples of the data set. For a sample i, the computed interesting subsets with interestingly good and bad performance measures are stored in $Positive\_ISD_i$ and $Negative\_ISD_i$ respectively. The union of the subsets computed from all samples are stored in $Positive\_ISD$ and $Negative\_ISD$. The subsets are ranked based on their number of occurrences in results of different samples. Subsets with occurrences less than the $MAJORITY\_COUNT$ are pruned.

Algorithm 2: Algorithm ISD_HS

input : A = Set of record attributes, D = Domain of the values of record attributes, T = Set of records, C = Significance level, L = Max. level (number of attributes in a descriptor), RUN_COUNT, SAMPLESIZE, MAJORITY_COUNT

output : I = Set of positive and negative interesting record properties

1 **for** *i=1 to RUN_COUNT* **do**
2 T_i = randomly picked sample of size SAMPLESIZE from the set of records T;
3 Run Algorithm ISD_H on T_i to obtain the interesting sets $Positive\_ISD_i$ and $Negative\_ISD_i$;
4 **end**
5 Positive_ISD = $Positive\_ISD_1 \cup Positive\_ISD_2 \cup \ldots \cup Positive\_ISD_{RUN\_COUNT}$;
6 Negative_ISD = $Negative\_ISD_1 \cup Negative\_ISD_2 \cup \ldots \cup Negative\_ISD_{RUN\_COUNT}$;
7 For each set in Positive_ISD compute the number of occurrences of the set in $Positive\_ISD_1$, $Positive\_ISD_2$, ..., $Positive\_ISD_{RUN\_COUNT}$;
8 For each set in Negative_ISD compute the number of occurrences of the set in $Negative\_ISD_1$, $Negative\_ISD_2$, ..., $Negative\_ISD_{RUN\_COUNT}$;
9 Positive_ISD = Sets in Positive_ISD with number of occurrences greater than the MAJORITY_COUNT;
10 Negative_ISD = Sets in Negative_ISD with number of occurrences greater than the MAJORITY_COUNT;
11 return Positive_ISD and Negative_ISD;

3 Experimental evaluation

In this section we present the experimental evaluation of the proposed algorithms.

3.1 Experimental setup

We executed the proposed algorithm on a data set consisting of service request records for the IT division of a major financial institution. Each record in the database consisted of a set of attributes representing various properties associated with the record. The database contained 6000 records. Each record had seven attributes namely *PR, AC, ABU, AI, CS, CT, CD* with the domain sizes of 4, 23, 29, 48, 4, 9, and 7 respectively. We used these attributes for classification of records into different subsets. Each record also contained *service time* as a performance metric which we used to measure the interestingness of the subsets of records. We applied the proposed algorithm on this data set to find interesting subsets. For the given performance metric we classified interesting subsets into the subsets performing significantly better or worse in terms of the *service time*.

We successfully identified the subsets of records with significantly large service time. We ran the algorithms from level 1 to 5. Level 1 results contain large subsets defined by a single attribute-value pair. These results provide the most high impact properties of the customer support tickets. We were able to identify the tickets of a specific day of the week, or tickets from a specific city to have significantly high service times than the rest of the tickets. With higher level results we were able to perform finer analysis of the properties of tickets that have significantly high service times. Such analysis gives interesting insights into the system behavior to identify performance bottlenecks. The algorithm also provides insights into the system improvements that can have highest impact on the improvement of the overall service time of the system.

We compare the ISD_H algorithm with the brute force algorithm ISD_BF. Because of its high execution time, we were not able to run the ISD_BF algorithm on large data sets and higher levels of attribute combinations. We hence compared the correctness of ISD_H algorithm on large data sets and higher levels with the ISD_R algorithm. In algorithm ISD_R, for a level *l*, we randomly pick *N* subsets where each subset consists of *l* attribute-value pairs such that every attribute is different in the subset. For these *N* subsets, we compute the interestingness of each subset. We run this algorithm multiple times and then evaluate the results obtained by the ISD_H algorithm for level *l* to contain the interesting subsets inferred by the ISD_R algorithm.

We first present the evaluation of the ISD_H algorithm. We compare the set space explored by ISD_H algorithm with ISD_BF algorithm to show that ISD_H algorithm explored significantly smaller set space. We then compare the coverage and accuracy of the ISD_H algorithm with the ISD_R algorithm to show that the correctness of ISD_H algorithm is not affected by the set space pruning performed by the algorithm. We then show the effect of varying the thresholds of the three heuristics of the ISD_H algorithm on the coverage, accuracy, and amount of pruning done by the ISD_H algorithm. We then evaluate the ISD_HS algorithm. We compare the results of ISD_HS algorithm with ISD_H algorithm to show the similarity in the results of ISD_H algorithm and the ISD_HS algorithm.

3.2 State space size

We compared the reduction in the state space obtained by the ISD_H algorithm as compared to the entire state space searched by algorithm ISD_BF. We ran the two algorithms over the given data set and measured the number of subsets considered by the two algorithms for the test of interestingness. We ran the ISD_H algorithm setting the heuristic thresholds as M_p = .01, M_s =5, M_g = .01. We ran the brute force algorithm up to level 4. We mathematically computed the state space size of the brute force algorithm for the higher levels. We were not able to run the

brute force algorithm for higher levels because of its computational complexity and long execution time. Figure 3a shows the state space size considered by the two algorithms for the test of interestingness for different levels. This set size is proportional to the execution time of the two algorithms. Figure 3a shows that the state space searched by the ISD_H algorithm is significantly smaller than the state space searched by the brute force algorithm. Furthermore, the state space increases linearly for the ISD_H algorithm with the increase in the level unlike the exponential growth of state space in the ISD_BF algorithm.

3.3 Correctness of the ISD_H algorithm

As we were not able to run the brute force algorithm over the entire state space, we evaluated the correctness of the ISD_H algorithm by comparing the ISD_H algorithm with the ISD_R algorithm. For a level *i*, we ran ISD_R algorithm 10 times and then evaluated the results obtained by the ISD_H algorithm for level *i* to contain the interesting subsets discovered by the ISD_R algorithm.

As the ISD_H algorithm prunes the state space based on different heuristics, the algorithm stops the search at a certain node in a search tree branch if the node is not found to be interesting enough for analysis. For instance, while exploring a particular branch of the state space, the algorithm may stop at a level *i* subset descriptor θ_i defined by the *i*-tuple ($A_1 = v_1, A_2 = v_2, \ldots, A_i = v_i$) because of the size heuristic. Thus for the subset of records $\Psi(\theta_i)$ corresponding to the descriptor θ_i, $|\Psi(\theta_i)| < M_s$. However, on the same branch of the search space the random algorithm might identify a *j*-tuple θ_j, where $j > i$, to be interesting. This *j*-tuple θ_j might not be searched by the ISD_H algorithm because of the above explained pruning. Thus in the above example, the ISD_H algorithm covers $\Psi(\theta_j)$ in $\Psi(\theta_i)$. We define a metric *Coverage* to measure this property of the ISD_H algorithm for a level *l* as follows:

> Coverage = Number of level *l* ISD_R set descriptors covered by ISD_H set descriptors / Total number of level *l* ISD_R set descriptors

It is also important to measure the accuracy of the coverage. In the above case if the subset $\Psi(\theta_j)$ is very small while subset $\Psi(\theta_i)$ covers $\Psi(\theta_j)$ but builds a very large subset, then the subset $\Psi(\theta_i)$ does not cover subset $\Psi(\theta_j)$ with enough accuracy. Note however that the ISD_H algorithm stops at a higher level *i* in the search space at the subset descriptor θ_i only if the records identified by the subset $\Psi(\theta_i)$ are interesting. Thus the extra records present in $\Psi(\theta_i)$ are also interesting. In order to compare how close are the results of the ISD_H algorithm to the ISD_R algorithm, we define the accuracy metric. If $\Psi(\theta_j)$ is the subset of records computed by the ISD_R algorithm as interesting and $\Psi(\theta_i)$ is the subset of records computed by the ISD_H algorithm that covers the records in $\Psi(\theta_j)$, then the *Accuracy* of coverage of θ_j can be computed as:

$$Accuracy_{\theta_j} = 1 - (|\Psi(\theta_i)| - |\Psi(\theta_j)|)/|D|$$

,where $|D|$ is the total number of records in the database. The numerator gives a measure of the number of extra records present in the subset $\Psi(\theta_i)$ and the division by the total number of records gives a measure of how significant is this difference in the given space of records. Accuracy of algorithm ISD_H for a level *l* is thus computed as follows:

> Accuracy = ($\sum$ Accuracy of each level *l* ISD_R set descriptor θ_j that is covered by ISD_H) / Total number of level *l* ISD_R set descriptors covered by ISD_H

Figure 3b presents the coverage and accuracy of the ISD_H algorithm by comparing it with the random algorithm. Each point plotted in Figure 3b is the average of the comparison of the ISD_H algorithm with 10 different runs of the random algorithm for a particular level. Figure 3b shows that the state space pruning shown in Figure 3a does not affect the accuracy of the results. The ISD_H algorithm achieves 100% coverage with 80% to 90% accuracy.

3.4 Effect of heuristics

In this section we analyze the effect of the three heuristics, the p-prediction, size, and goodness heuristics, on the effectiveness and the efficiency of the ISD_H algorithm. We ran the ISD_H algorithm with different threshold values for the three heuristics and measured the effect of the heuristics on the coverage, accuracy, and pruning. We ran the experiments by setting the threshold of two heuristics to fixed values and varying the threshold of the third heuristic. We ran the ISD_H algorithm in this fashion for 7 levels and compared the result with the results of the ISD_R algorithm. Each point plotted in the graphs is an average of the comparison of the ISD_H algorithm with 10 different runs of the ISD_R algorithm.

We first present the effect of the size heuristic on the ISD_H algorithm. M_s is the threshold that defines the minimum size of the subset to be considered for interestingness. Set sizes below this threshold are pruned and are not explored further by the algorithm. We ran ISD_H algorithms with M_p = 0.01, M_g = 0.01, and varying the value of M_s to 5, 10, and 20. Figures 4a, 4b, and 4c present the total number of subsets explored, coverage, and accuracy respectively when running the ISD_H algorithm for different values of M_s. With the increase in M_s more subsets get pruned. We can see that the coverage stays high for all values of M_s. It can also be seen that with the increase in M_s the coverage stays the same but the accuracy decreases. This behavior can be explained from the observation that with the increase in M_s more subsets get pruned at a higher level preventing the ISD_H algorithm from identifying a subset to a finer level. The algorithm still covers all the interesting subsets maintaining a high coverage but results in a decrease in accuracy with higher values of M_s.

We then present the effect of the p-prediction heuristic on the ISD_H algorithm. The p-prediction heuristic prevents combination of two subsets if the two subsets are statistically very different, where the statistical difference is measured by the *p-value* of the *t-test*. Mutual *p-value* below the threshold M_p is considered statistically different

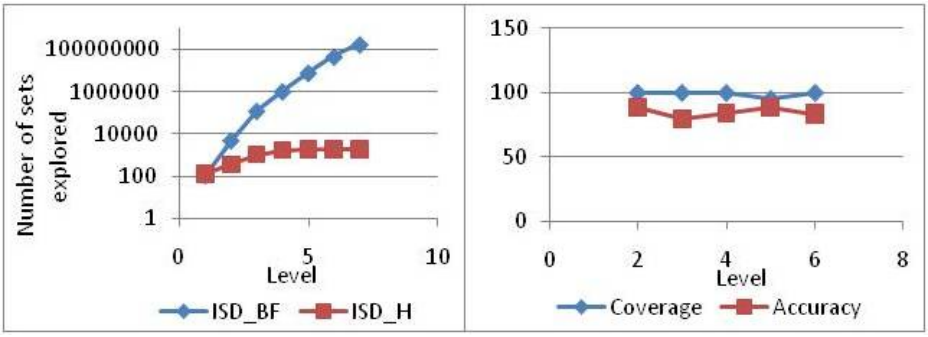

Figure 3: (a) Comparison of attribute state space size searched by the ISD_BF and the ISD_H algorithms. (b) Coverage and accuracy of the ISD_H algorithm.

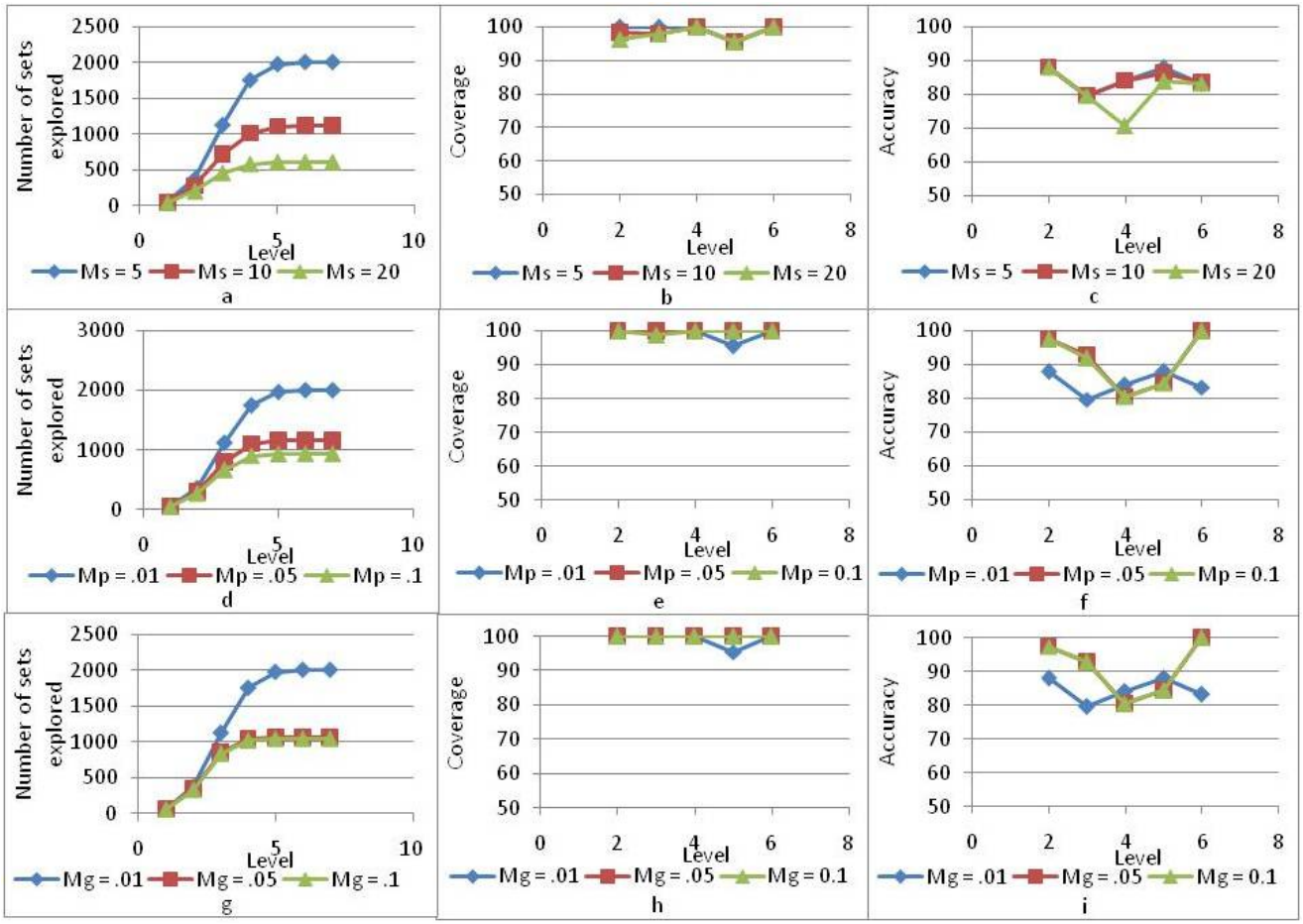

Figure 4: Effect of changing M_s, M_p, and M_g on total state space pruning, coverage, and accuracy

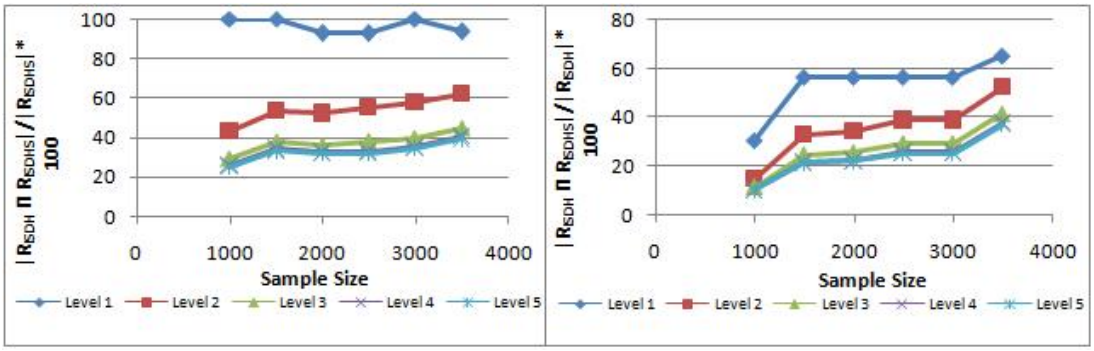

Figure 5: (a) % of algorithm ISD_HS results that match with algorithm ISD_H, (b) % of algorithm ISD_H results covered by algorithm ISD_HS.

and hence the subsets with mutual p-value less than M_p are not combined by the p-prediction heuristic. We present the results of running the ISD_H algorithm by varying the value of M_p to 0.01, 0.05, and 0.1 and setting M_s = 5, M_g =0.01 . Figures 4d, 4e, and 4f present the total number of subsets explored, coverage, and accuracy of the ISD_H algorithm respectively when running the ISD_H algorithm with different values of M_p. As shown in Figure 4d smaller values of M_p drop less subset combinations thus allowing more subsets to pass through. This in turn affects the accuracy of the algorithm. Smaller values of M_p are likely to provide less accuracy. ISD_H algorithm provides high coverage and accuracy for all values of M_p. Coverage stays the same for different values of M_p and accuracy tends to increase with higher values of M_p.

We then analyze the effect of varying the goodness heuristic. We executed the ISD_H algorithm with M_s = 5, M_p = 0.01 and vary M_g to 0.01, 0.05, and 0.1. The goodness heuristic drops the subsets that are significantly better than the rest by dropping the subsets with a positive value of *t* and a *p-value* less than M_g. Figures 4g, 4h, and 4i present the total number of subsets explored, coverage, and accuracy of the ISD_H algorithm respectively when running the ISD_H algorithm with different values of M_g. Smaller value of M_g drops less subsets and thus tend to result in lower accuracy of the ISD_H algorithm. The coverage stays high for all values of M_g.

3.5 Evaluation of algorithm ISD_HS

We now present the experimental evaluation of the sampling-based algorithm ISD_HS. We ran the ISD_HS algorithm on a data set of 11000 records with sample sizes ranging from 1000 to 3500. We set the RUN_COUNT to 10 running algorithm ISD_H on 10 different samples. We used the MAJORITY_COUNT values of 1, 2, and 5.

We compared the performance of the ISD_HS algorithm with the ISD_H algorithm. We ran the ISD_H algorithm on the same data set of 11000 records. Figure 5a presents the number of subsets obtained by algorithm ISD_HS that are also present in the results of ISD_H algorithm. Let R_{ISDH} are the interesting subsets identified by the ISD_H algorithm and R_{ISDHS} are the interesting subsets identified by the ISD_HS algorithm, then Figure 5a plots the values of

$$(|R_{ISDH} \cap R_{ISDHS}|)/|R_{ISDHS}| * 100$$

for different sample sizes.

Figure 5a shows larger number of matches between the results of ISD_H and ISD_HS for lower levels but as the levels increase the number of matches decrease. Higher level sets have smaller sizes because of the increased specification of attributes and values. The smaller sets are not captured by the algorithm ISD_HS because of the reduced number of records in the sample to represent the set. The algorithm ISD_HS thus focusses on identifying the interesting subsets that are large in size. Such results can be useful to identify high impact set properties in the data set. For instance, in a data set of customer support tickets, identification of the attribute-values of a large subset of tickets that have a high service time can be of greater interest to improve the overall system performance as compared to a smaller subset of tickets.

Figure 5b presents the percentage of algorithm ISD_H results captured by the algorithm ISD_HS. Figure 5b thus plots the values of

$$(|R_{ISDH} \cap R_{ISDHS}|)/|R_{ISDH}| * 100$$

for different sample sizes. This percentage decreases for higher levels as higher level sets are smaller in size and are not captured in the smaller samples with enough number of records.

Note that as the sample size increases the match percentage increases. Also, with the increase in sample size algorithm ISD_HS is able to capture more number of interesting sets and thus the match percentage with the algorithm ISD_H increases.

4 Related work

Design of algorithms to automatically discover important subgroups (e.g., a subset of records) in a given data set is an active research area in data mining. Such subgroup discovery algorithms are useful in many practical applications [6], [7], [2], [3].

Assume that the input data includes a class label attribute C, whose domain is a finite set of discrete symbols. Let $A_1, A_2, \ldots, A_n$ denote the set of other data attributes, each having a finite domain (continuous attributes can be suitably discretised). A *subgroup* (i.e., a subset of records) is specified using a formula in propositional logic, with $Attribute = Value$ forming a basic proposition. Conjunctions of such $Attribute = Value$ tuples (e.g., $Outlook = Sunny \wedge Humidity = Normal$) is a commonly used representation of subsets. Subgroup discovery algorithms systematically search the hypothesis space of all possible propositional formulae (each of which represents a subgroup) to identify subgroups that are unusual or interesting in some well-defined sense. Typically, a subgroup is *interesting* if it is sufficiently large and its statistical characteristics are significantly different from those of the data set as a whole. The subgroup algorithms mostly differ in terms of (i) subgroup representation formalism; (ii) notion of what makes a subgroup interesting; and (iii) search and prune algorithm to identify interesting subgroups among the hypothesis space of all possible subgroups representations.

Many quality measures are used to evaluate the interestingness of subgroups and to prune the search space. Well-known examples (when C is a binary class label) include binomial test and relative gain, which measure the relative prevalence of the class labels in the subgroup and the overall population. Other subgroup quality measures include support, accuracy, bias and lift. We use a continuous class attribute (in contrast to discrete in almost all related work). Another new feature of our approach is the use of Students t-test as a measure for subgroup quality.

Initial approaches to subgroup discovery were based on a heuristic search framework [8], [5], [12]. More recently, several subgroup discovery algorithms adapt well-known classification rule learning algorithms to the task of subgroup discovery. For example, CN2-SD [4] adapts the CN2 classification rule induction algorithm to the task of subgroup discovery, by inducing rules of the form $Cond \rightarrow Class$. They use a *weighted relative accuracy (WRA)* measure to prune the search space of possible rules. Roughly, WRA combines the size of the subgroup and its accuracy (difference between true positives and expected true positives under the assumption of independence between $Cond$ and $Class$). They also propose several interestingness measures for evaluating induced rules. Some recent work has adopted well-known unsupervised learning algorithms to the task of subgroup discovery. [11] adapts the a priori association rule mining algorithm to the task of subgroup discovery. The SD-Map algorithm [1] adopts the FP-tree method for association rule mining to the task of minimum-support based subgroup discovery. Some sampling based approaches to subgroup discovery have also been proposed [9], [10].

5 Conclusion and future work

We presented algorithms for the discovery of interesting subsets from a given database of records with respect to a given quantitative measure. We proposed various heuristics to prune the state space of subsets and presented a heuristic-based algorithm ISD_H for interesting subset discovery. We then presented a sampling-based algorithm ISD_HS to efficiently use the proposed heuristic-based approach on large data sets. We presented an experimental evaluation of the proposed algorithms by applying the algorithm on a data set consisting of service request records for the IT division of a major financial institution. We showed that algorithm ISD_H prunes the state space very effectively and yet provides a high coverage and accuracy in the discovery of interesting subsets. We also presented the effect of various heuristics on the behavior of the algorithm. We presented an evaluation of the sampling-based algorithm ISD_HS and showed that the algorithm captures high impact interesting sets with very high correctness. We also presented a comparison of the ISD_H and ISD_HS algorithms. The interesting subsets discovered by the algorithm prove to be very insightful for the given data set. Discovery of such subsets of records can provide insights for improving the involved business processes, e.g. identification of the bottlenecks, identification of the areas for improvement, etc.

As part of the future work, first, we are interested in strengthening the heuristics, so as to further reduce the number of states searched. We are working on extending the approach to work with continuous heuristics and to use the full logical power of SQL commands to systematically form more complex subsets (e.g., an arbitrary mix of conjunctions, disjunctions, and negations). One way to extend the algorithms is to allow expressions of the kind $A_i \in V$ for subset formation, which identifies a subset where attribute A_i takes any value in a subset V of its domain. Lastly, we are using the algorithms to discover interesting subsets in the real-life data-sets from different domains (e.g., employee data).

References

[1] M. Atzmueller and F. Puppe. Sd-map: a fast algorithm for exhaustive subgroup discovery. In *Proc. PKDD 2006*, volume 4213 of *LNAI*, pages 6 – 17. Springer-Verlag, 2006.

[2] M. Atzmueller, F. Puppe, and H.P. Buscher. Profiling examiners using intelligent subgroup mining. In *Proc. 10th Intl. Workshop on Intelligent Data Analysis in Medicine and Pharmacology (IDAMAP-2005)*, pages 46 – 51, 2005.

[3] N. Lavrač, B. Cestnik, D. Gemberger, and P. Flach. Subgroup discovery with cn2-sd. *Machine Learning*, 57:115 – 143, 2004.

[4] N. Lavrač, B. Kavšek, P. Flach, and L. Todorovski. Subgroup discovery with cn2-sd. *Journal of Machine Learning Research*, 5:153 – 188, 2004.

[5] J. Friedman and N. I. Fisher. Bump hunting in high-dimensional data. *Statistics and Computing*, 9:123 – 143, 1999.

[6] D. Gemberger and N. Lavrač. Descriptive induction through subgroup discovery: a case study in a medical domain. In *Proc. of 19th Int. Conf. on Machine Learning (ICML02)*, pages 163 – 170. Morgan-Kaufman, 2002.

[7] D. Gemberger and N. Lavrač. Expert guided subgroup discovery: methodology and application. *Journal of Artificial Intelligence Research*, 17:501 – 527, 2002.

[8] W. Klosgen. Explora: a multipattern and multistrategy discovery assistant. In *Advances in Knowledge Discovery and Data Mining*, pages 249 – 271. MIT Press, 1996.

[9] T. Scheffer and S. Wrobel. Finding the most interesting patterns in a database quickly by using sequential sampling. *Journal of Machine Learning Research*, 3:833 – 862, 2002.

[10] M. Scholtz. Sampling based sequential subgroup mining. In *Proc. 11th SIG KDD*, pages 265 – 274, 2005.

[11] B. Kavšek, N. Lavrač, and V. Jovanoski. Apriori-sd: adapting association rule learning to subgroup discovery. In *Proc. 5th Int. Symp. On Intelligent Data Analysis*, pages 230 – 241. Springer-Verlag, 2003.

[12] S. Wrobel. An algorithm for multi-relational discovery of subgroups. In *Proc. 1st European Symposium on Principles of Data Mining and Knowledge Discovery (PKDD97)*, pages 78 – 87. Springer-Verlag, 1997.