A Hierarchical Approach for Anomalous Subgroup Discovery

Anonymous authors
[Eliana: It is single blind!] Anonymous institution
Anonymous country
anonymous@mail.com

***Abstract*—Understanding peculiar and anomalous behavior of machine learning models for specific data subgroups is a fundamental building block of model performance and fairness evaluation. The analysis of these data subgroups can provide useful insights into model inner working and highlight its potentially discriminatory behavior. Data slices based on attribute values provide an effective and interpretable definition of data subgroups. State-of-the-art data slicing approaches ignore the presence of hierarchies in the data and can only be applied on discretized attribute domains. The discretization process required for continuous attributes may affect significantly the identification of relevant subgroups.**

We propose a hierarchical subgroup exploration technique to identify anomalous subgroup behavior at multiple granularity levels. The exploration pipeline consists of two main steps. We first propose a hierarchical discretization method that exploits a bias metrics to optimize interval definition during the discretization process. The method induces a hierarchy on each continuous attribute. Both derived hierarchies and any pre-defined hierarchy are then leveraged by our subgroup identification algorithm based on generalized itemset mining. The experimental results shows the effectiveness of both the hierarchical discretization and exploration in revealing peculiar behaviors for multiple granularities of attribute values.

***Index Terms*—fairness, model evaluation, subgroup discovery, discretization, generalized itemsets, data slicing**

I. Introduction

The pervasiveness of black-box machine learning models in a variety of applications has brought increased interest in Explainable AI research. Furthermore, in critical domains such as health care and insurance, their application has raised concerns about algorithmic bias and fairness [1]–[3]. Typically, model performance is analyzed at the global level, indicating the model behavior for the overall dataset, or for specific class labels of interest. However, the identification of problematic *data subgroups*, identified by slicing the dataset on the value of specific attributes, is relevant in multiple applications, e.g, the evaluation of model fairness and model validation. Understanding model behavior at the *subgroup level* allows the identification of anomalous, and potentially problematic, behaviors of a model in specific data subgroups, characterized by a different model performance with respect to the overall dataset.

Recently, several works have been proposed to identify and characterize subgroups for which a peculiar behavior is observed. Works as Slice Finder [4], DivExplorer [5], and SliceLine [6] identify possibly-overlapping data subgroups for which a model performs differently. They automatically identify the most relevant attributes and attribute values on which data subgroups are defined. More specifically, the subsets are identified by slicing the data in the attribute domain. Each subgroup is a subset of the data characterized by a set of attribute values. The subgroups are defined in terms of *patterns,* or *itemsets*, which are conjunctions of *attribute=value* pairs. Consequently, the subgroups are directly interpretable.

Data subgroup	FPR	Δ_{FPR}	Support
Entire dataset	0.088	0.000	1
#prior>3	0.219	0.131	0.29
#prior>8	0.384	0.295	0.11
age<27	0.155	0.067	0.31
age<27, #prior>3	0.377	0.288	0.05

TABLE I: Example of impact of discretization of the attribute *#prior* on false positive rate (FPR) and FPR divergence for the corresponding identified data subgroups (*compas* dataset).

Current subgroup identification approaches suffer from two important limitations: (i) ignoring the presence of hierarchies in data, which may allow a wider exploration of data slices, and (ii) requiring the data attributes to be *discrete,* that is, to be able to assume only a finite, small set of values.

A hierarchy induces a nested partitioning of attribute values into subgroups with different levels of detail in data representation. Traditionally, hierarchies have been exploited to represent structural dependencies (e.g., functional dependencies) between attribute values. For example, a geographical hierarchy may be induced on geographic locations by considering the geographic coordinates, city, state, and country attributes, which provide a growing generality (or decreasing detail level) in the data representation.

Hierarchies may be explicitly represented by structural dependencies between attributes, or revealed by analyzing data. For categorical attributes, if hierarchical dependencies between attributes are not explicitly defined, they can be automatically derived, e.g., by considering functional dependencies between attributes [7], [8]. As an alternative, hierarchical information on attributes may be directly provided by means of user-defined dependencies. For continuous attributes, the discretization process may be exploited to induce a hierarchy of data values. Hence, discretization plays an important role in subgroup identification, both in the identification of relevant attribute value ranges, and in enabling the hierarchical exploration of continuous attributes.

For a continuous attribute, subgroups can be obtained by considering value ranges. However, the choice of ranges affects the detection of problematic behaviors. To illustrate this concept, consider the *compas* dataset [9]. It contains demographic information and the criminal history of defendants screened in Broward County, Florida, during 2013 and 2014 and collected by ProPublica. For each defendant, the dataset includes a score of recidivism risk derived by a proprietary algorithm, and the information on whether the defendant did actually recidivate. We consider the predicted high-risks scores as the predicted positive class of recidivism.

Table I shows the false positive rate (FPR) of the subgroups identified by different ranges on the number of prior offenses (attribute *#prior*). Along with the false positive rate, we report the difference in the behavior of the machine learning algorithm on the data subgroup vs. the entire dataset: this behavior difference is the *divergence* of [5]. For the false positive rate, the divergence represents the difference between the false positive rate of the subgroup and the overall one. The entire dataset has a false positive rate of 0.088.

Consider now two different subgroups, (i) defendants with more than 3 prior offenses, and (ii) defendants with more than 7 prior offenses. Both groups show a (high) divergence from the overall behavior. However, defendants in group (ii) show a (much) higher divergence than those in group (i). This illustrates how interval definition for continuous attribute discretization may significantly affect the effectiveness of anomalous subgroups identification.

We propose a novel discretization strategy that aims at supporting the anomalous subgroup identification algorithms. Given a continuous attribute, our discretization method builds a hierarchy of discretization intervals, characterized by a different granularity at each level of the tree, that can all be exploited by slicing techniques to detect the most interesting slices. These intervals can then be fed to subgroup-exploration tools, along with the already-discrete attributes. We base this paper on the subgroup-exploration approach of [5], but other exploration methods, such as those of [4], [6], could also be used.

Our discretization method naturally identifies a hierarchy or taxonomy for the attributes. We integrated our discretization approach in [5] by means of generalized itemset discovery [10], [11], that directly exploits the hierarchical structure derived by the tree. As a result, we are able to identify data subgroups at any level of the taxonomy. Consider again the example for the *compas* dataset. Using the taxonomy in Figure 1, we are able to derive data subgroups characterized both by *#prior*>8 and *#prior*>3 and select the most interesting subgroups among them. Suppose that we want to identify data subgroups with at least a support of 0.05. The hierarchical exploration automatically explores associations at multiple granularity. Therefore, we can identify the subgroups of defendants with age greater than 27 and more than 3 prior offenses as divergent. Using instead only a fixed discretization with *#prior*>8, we cannot explore the association with defendant with age lower than 27 since this subgroup is characterized by an excessively low support (below 0.01%).

Our main contributions are as follows.

a) Hierarchical approach to group detection: Traditional subgroup detection approaches do not consider the presence of hierarchies on attributes. Hence, the attribute space exploration they perform is limited to data slices identified at the most detailed level of the hierarchy. Due to the reduced number of instances in the group, more detailed (and tiny) slices of potentially interesting (divergent) subgroups may be disregarded by the subgroup detection techniques. We show that, by defining hierarchies over data and exploring data at higher levels in the hierarchy, thus considering a coarser granularity, it is possible to detect interesting subgroups characterized by a larger number of instances. If only the smallest granularities at the lowest level of a hierarchy were available, combining items (attribute conditions) for different attributes may yield subgroups that are too small to be statistically significant. Only a hierarchy offers full flexibility to the exploration of subgroups.

b) Hierarchical discretization: Usually, discretizing a dataset means turning its continuous attributes into discrete ones, mapping the continuous values into non-overlapping ranges. We show here that we can obtain a much richer exploration of subgroups by associating the continuous values with a hierarchy of ranges, at multiple levels of granularity. Our experiments indicate how this *hierarchical* discretization leads to more effective subgroup identification, as the subgroup exploration algorithms can then select the appropriate range granularity for continuous attributes, thus leading to the identification of subgroups that are both anomalous and statistically significant.

c) Individual-attribute trees for discretization: Trees such as decision trees are constructed by considering all attributes. In contrast, we propose the use of *individual* trees to separately discretize each attribute. For example, discretizing the *#prior* attribute (number of prior offenses) for *compas* can use a tree such as the one in Figure 1. Individual trees allow both to discretize attributes and induce a hierarchy.

d) Divergence-aware tree construction: The construction of decision trees is driven by attributes of individual data points, such as the class label. In contrast, divergence is an ensemble property. We illustrate how the gain and support criteria used in generating decision trees can be adapted to obtain discretizations that reveal divergent subgroups.

e) Hierarchical-subgroup exploration pipeline: We have implemented a pipeline for discretization and hierarchical subgroup analysis which is capable of handling datasets with predefined hierarchies and continuous attributes. For continuous attributes, the pipeline first uses trees to discretize attributes, using individual per-attribute trees. The outcome of this stage consists in range hierarchies for each continuous attribute. Both predefined hierarchies on categorical attributes and range hierarchies on continuous attributes are then fed, jointly with

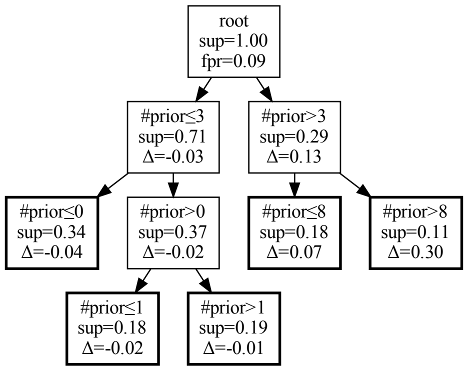


Fig. 1: Item hierarchy for the *#prior* attribute for the FPR of the *compas* dataset obtained via tree discretization.

the dataset, to the proposed extension of DIVEXPLORER [5] to deal with attribute hierarchies based on generalized frequent pattern mining. Finally, the output of this step consists in the subgroups built out of the available ranges and discrete attributes, and that are both statistically significant, and exhibit anomalous behavior. This extends the state of the art by providing a complete pipeline for the analysis of generic datasets with a mix of continuous and discrete attributes. We provide open-source code implementing the complete pipeline at https://anonymous.4open.science/r/treedis-CE56.

The paper is organized as follows. Section II discusses the related work. Section III provides our main definitions and discusses the problem statement. Section IV introduces the generalized subgroup identification algorithm. Section V describes the tree discretization process and outlines the hierarchical-subgroup exploration pipeline. Section VI presents experimental results, showing the ability of the proposed pipeline to reveal peculiar behavior in subgroups, while Section VII draws conclusions.

II. RELATED WORK

We address related work in the areas of (i) anomalous subgroup detection and (ii) discretization.

Anomalous subgroup detection: Several recent works address the automatic identification of subgroups with anomalous behavior [4]–[6], [12], [13]. Among these works, the most effective approaches in subgroup identification exploit the notion of pattern and lattice search to identify data slices (i.e., subgroups) [4]–[6]. These works share their subgroup representation, but characterize and build the subgroups differently. The anomalous behavior of a subgroups is generally characterized with respect to the overall behavior (as in [5] in terms of divergence) or with respect to its counterpart (as in [4] in terms of effect size as a function of the distribution of loss differences).

Tree-based approaches to subgroup identification are explored in [4] and the Error Analysis dashboard of the Responsible AI Toolbox [13]. These approaches use the trees to partition the dataset into non-overlapping subgroups. Lattice search approaches have been proposed in [4]–[6]; these approaches consider attribute lattices.

These approaches ignore the presence of hierarchies in data. Hierarchies of attributes allow to represent of data at different levels of granularity. Attribute hierarchies can explicitly represent structural dependencies among data, specified by users or automatically be derived by analyzing data as in the case of functional dependencies. In our work, we propose a hierarchical exploration of data subgroups that exploit hierarchies to extract subgroup with different granularity. We rely on generalized frequent-pattern mining approach built on the basis of [5] to extract subgroups with anomalous behavior.

Lattice-based approaches, while having the advantage of identifying overlapping subgroups, additionally suffer from requiring the data attributes to be *discrete*. Any continuous attribute needs to be discretized before subgroup identification. The discretization induces a unique and flattened representation. For each continuous value, we have a unique discretization interval. The discretization step is hence crucial: too coarse discretization, or one in which anomalous and normal data coexist in the same "slice", hampers the identification of the anomalous subgroups. In our work, we instead propose a *hierarchical discretization* of continuous attribute. We exploit separate trees, built on each continuous attribute. However, differently from [4], [13], we leverage trees to identify interesting ranges of continuous attributes in a hierarchical representation, rather than anomalous subgroups.

Differently from all the mentioned approaches [4]–[6], [13], our hierarchical-subgroup exploration pipeline leverages predefined hierarchies on categorical attributes and item hierarchies obtained from our hierarchical discretization approach to reveal anomalous subgroups at multiple granularity levels.

Discretization: Discretization approaches can be categorized into supervised and unsupervised [14]. Unsupervised methods discretize the data without considering the target variable, while supervised ones use the target variable information to discretize the data. Our approach falls into the latter category. When discretization is applied as a step towards classification, as in the construction of classification trees, the class label becomes the target. Entropy is typically used to measure the class information entropy by decision tree-induction algorithms (e.g., ID3 [15] and C4.5 [16]). For example, in [17], the authors propose a C4.5-based discretization, in which the C4.5 decision tree algorithm is applied for each continuous feature separately to determine the splitting values. In [18], each attribute is recursively split to minimize the class entropy using the minimum description length as a stopping criterion.

We also leverage trees for discretization, but with two differences. First, we use as supervision target not the class label, but quantities directly connected to the anomaly we seek to detect. In particular, we use either divergence itself as a measure of the anomalous behavior, or entropy with respect to a three-valued (true, false, undefined) outcome function that is used to define the divergence. Second, we do not just use the leaves of the tree to derive the discretization. Rather, we propose the use the entire hierarchy. Thus, our approach generalizes former discretization approaches by deriving a hierarchy of discretizations at different granularity (see, e.g., Figure 1), which can be fed to our subgroup identification algorithms to explore discretization intervals at different detail level. This allows the identification of patterns that would not be frequent, if only the finer discretization was used for each attribute. This flexibility is not available in previous lattice-based approaches [4]–[6].

III. Preliminaries

Our goal is to identify regions of a dataset that differ from the whole dataset with respect to some statistics of interest, such as precision, recall, false-positive or false-negative rate, and similar. In particular, we focus on a hierarchy of dataset regions that allows us to investigate anomalous behaviors at different granularity levels. Furthermore, when the dataset includes continuous (and real-valued) attributes, we propose a discretization technique that allows to identify intervals in a way that best identifies the anomalous regions. The section first provides our main definitions. We then outline the notion of divergence as a measure of peculiar behavior and the algorithm DivExplorer that serves as base for the generalized exploration approach proposed in our paper.

A. Datasets and Itemsets

We consider a dataset D with attributes $\mathcal{A} = \{A_1, ..., A_n\}$ consisting of a set of instances over $\mathcal{A}$. Each attribute $A_i \in \mathcal{A}$ has domain $\mathcal{D}_{A_i}$. An attribute can be *categorical* or *continuous*: the domain of a categorical attribute is a finite set; the domain of a continuous attribute is the set $\mathbb{R}$ of real numbers. We denote by $\mathcal{C} \subseteq \mathcal{A}$ the set of continuous attributes of the dataset. An instance $x \in D$ of the dataset assigns a value $x(A)$ to each attribute $A \in \mathcal{A}$, with $x(A) \in \mathcal{D}_A$. We indicate with D the set of all instances of the dataset. We use $\#S$ to denote the number of elements of a set S; in particular, $\#D$ is the number of instances of the dataset.

An item α is a constraint on an attribute:

- For a categorical attribute $A \in \mathcal{A} \setminus \mathcal{C}$, an item has the form $A = a$ for $a \in \mathcal{D}_A$.
- For a continuous attribute $A \in \mathcal{C}$, an item has the form $A \in J$, for an (open or closed, finite or infinite) interval J of real numbers.

We denote by $\mathcal{D}_\alpha \subseteq \mathcal{D}_A$ the subset of the domain of A that satisfies the item I.

We define subgroups of data instances via sets of items, known as *itemsets* or *patterns*. Given a set of items $\mathcal{I}$, an *itemset* I over $\mathcal{I}$ is a set of items $I \subseteq 2^{\mathcal{I}}$, such that no two items $\alpha, \beta \in I$ refer to the same attribute. The *length* $|I|$ of an itemset I is simply its cardinality. An example of itemset is $\{age \in [25, 45], sex = Female\}$. An instance $x \in D$ satisfies an item α, written $x \models \alpha$, if x satisfies the constraint α, and x satisfies an itemset I, written $x \models I$, if it satisfies all the constraints of the items in I. For example, a data instance with $age = 28$ and $sex = Female$ satisfies the itemset $I = \{age \in [25, 45], sex = Female\}$. We denote by $D_I = \{x \in D \mid x \models I\}$ the set of instances that satisfy an itemset I. The *support* $\sup(I)$ of an itemset I is the fraction of instances that satisfy it, given by $\sup(I) = \frac{\#D_I}{\#D}$. For a given support threshold s, an itemset (or pattern) I is *frequent* if $\sup(I) \geq s$.

B. Divergence

To capture how a subgroup differs from the the entire dataset, we use the notion of divergence [5], which is the difference between a specified statistics as measured on the subgroup, and as measured on the whole dataset. Precisely, a *statistic* $f : 2^D \mapsto \mathbb{R}$ associates a value $f(I)$ to every subset $I \subseteq D$ of dataset instances, and the *divergence* of I with respect to f is given by $\Delta_f(I) = f(I) - f(D)$. For example, the *false-positive divergence* of a subgroup is the difference between the false-positive rate measured on the subgroup, and measured on the whole dataset. In practice, we define statistics via *outcome functions*, which associate an outcome in $\mathbb{R} \cup \{\perp\}$ with each instance. Given an outcome function o, we compute the divergence of an itemset I via

$$o(I) = E\{o(x) \mid x \models I, o(x) \neq \perp\} ,$$

where E is the average operator. By using different outcome functions, we can easily define statistics such as false and true positive rates and the corresponding negative rates [5], rates related to rankings [19], and more. To measure the statistical significance of the divergence, we use the t-value of the divergence, computed according to the Welch's t-test as in [5].

C. DivExplorer

Given a set of items $\mathcal{I}$ and a support threshold s, DivExplorer [5] computes the divergence of all frequent itemsets over $\mathcal{I}$. The lattice-based exploration of the frequent itemsets is efficient, since DivExplorer integrates the computation of divergence into the well-known frequent pattern mining algorithms Apriori [20] and FP-Growth [21]. In particular, DivExplorer accumulates the statistics required for divergence computation within the algorithms that explore all frequent itemsets, so that the divergence can be computed at essentially no additional cost compared with exploration. The support threshold s acts as a stopping criterion, limiting the search to the frequent itemsets that have support at least s. Limiting the exploration to frequent itemsets is justified: the divergence of small subgroups may not be statistically significant, and in any case, anomalies affecting larger subgroups are more interesting, from the point of view of anomaly detection, than anomalies affecting smaller subgroups.

IV. Generalized Subgroups

The exploration implemented in DivExplorer is based on a fixed set of items $\mathcal{I}$, computed before the exploration starts. This carries two drawbacks. The first is that continuous attributes need to be discretized into items *before* the exploration starts, and without the benefit of choosing the discretization that maximizes divergence detection in the resulting itemsets. The second drawback is that the required discretization ignores the presence of hierarchies.

A. *Benefits of Item Hierarchies*

Hierarchies are instrumental in finding highly divergent subgroups with support size above a specified threshold. To understand the interplay between hierarchy and exploration for divergent subgroups, consider a dataset consisting of two attributes, a continuous *income*, and a discrete *zip* code. To study the dataset at a national level, we may wish to consider itemsets based on *income* alone, in which case quite a fine discretization may work well: we could for instance divide *income* in intervals of \$100, with the confidence that each resulting itemset (such as, for instance, $51,300 \leq income \leq 51,400$) has support above the threshold (contains enough people to be statistically significant). But if we wish to study the situation in a particular zip code, divining income in ranges of \$100 may be much too fine, as it would result in most itemsets having support below the threshold. For the analysis of a single zip code, a coarser discretization, perhaps in ranges of \$5,000, might be necessary. With fixed discretizations, there is a trade-off: a fine discretization enables the study of individual attributes, but a coarser discretization on each attribute may be necessary to study itemsets involving multiple attributes. Hierarchies enable us to have both benefits at once. In a hierarchical discretization, an item is discretized into coarse items, which in turn are discretized in finer items, and so on, up to items that are just larger than the support. An itemset can consist of items at different levels of granularity (at different places in the hierarchy) for the various attributes.

A hierarchical discretization allows the selectivity budget to be spent optimally during the exploration for highly divergent subgroups, combining high selectivity (items lower in the hierarchy, and thus with smaller support) for some attributes, with lower selectivity (items higher in the hierarchy) for others. In the following, we introduce hierarchies, and we show that they may be beneficial both for the discretization of continuous attributes and the exploration of categorical ones.

B. *Item Hierarchies*

In a hierarchy of items, each item for an attribute can be refined by two or more items whose supports forms a partition of the support of the refined item. For example, $age > 50$ can be refined into $50 < age \leq 60$ and $age > 60$. We use $\succ_A$ to indicate the refinement relation between items for an attribute A, so that we write $(age > 50) \succ_{age} (50 < age \leq 60)$. The precise definition is as follows.

Definition 4.1 (Item hierarchy.): Let A be an attribute with domain $\mathcal{D}_A$. An *item hierarchy* $(\mathcal{I}_A, \succ_A)$ for A consists of a set $\mathcal{I}_A$ of items for A, along with a hierarchy relation $\succ_A \subseteq \mathcal{I}_A \times \mathcal{I}_A$; we write $\alpha \succ_A \beta$ to mean that β is a (one-step) refinement of α, that is, it includes only a portion of the instances of α. We require that $\mathcal{D}_\alpha$ is partitioned into the set of $\mathcal{D}_\beta$ for all β with $\alpha \succ \beta$. Precisely, we require $\succ_A$ to be acyclic, and:

- If there is β such that $\alpha \succ_A \beta$, then $\mathcal{D}_\alpha = \cup_{\beta:\alpha\succ_A\beta}\mathcal{D}_\beta$;
- If $\alpha \succ_A \beta_1$ and $\alpha \succ \beta_2$ with $\beta_1 \neq \beta_2$, then $\mathcal{D}_{\beta_1} \cap \mathcal{D}_{\beta_2} = \emptyset$.

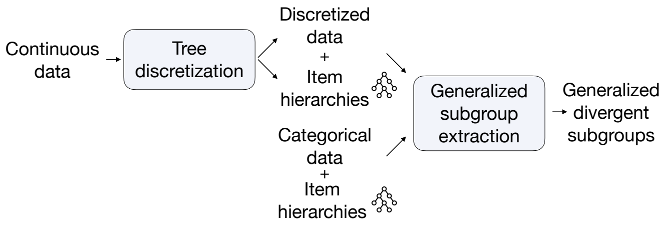

Continuous data
Tree discretization
Discretized data + Item hierarchies
Categorical data + Item hierarchies
Generalized subgroup extraction
Generalized divergent subgroups

Fig. 2: Hierarchical-subgroup exploration pipeline.

A *hierarchical discretization* for a dataset D consists in an item hierarchy for each of the attributes of D.

Item hierarchies can be derived directly by explicit structural dependencies of the attributes themselves, provided through user-defined dependencies or revealed by analyzing data. Examples of explicit hierarchies are geographical hierarchies, IP addresses (where the hierarchy corresponds to their byte sequence), or product taxonomies. For categorical attributes, hierarchical dependencies can also be automatically derived by considering functional dependencies between attributes [7].

For continuous attributes, we propose a hierarchical discretization process to automatically define the data ranges and the item hierarchy in a way that facilitates the exploration of divergent itemsets.

V. Hierarchical Divergence Exploration

A. *Hierarchical-subgroup exploration pipeline*

In the approach of [5], DivExplorer was used directly over datasets that had been already discretized, and the discretizatoin could not include any hierarchy. In practice, users of DivExplorer would first discretize any continuous attributes in the dataset based on their experience and intuition about the dataset, likely aided by data visualization tools.

Here, we introduce an automated analysis process based on the use of hierarchical items, where the discretization is guided by the divergence of the statistics of interest. The proposed analysis pipeline is depicted in Figure 2.

The first step consists in discretizing any continuous attributes into item *hierarchies*. This discretization is driven by the statistic whose divergence we are interested in measuring, as well as by the minimum support size of any divergent itemset we are interested in detecting. For discrete attributes, we consider item hierarchies using any hierarchy intrinsic in the attributes.

In the second step of the pipeline, the generalized subgroup discovery algorithm leverages the item hierarchies derived by the hierarchical discretization and the predefined hierarchies for the categorical attributes, if available. The algorithm outputs the frequent itemsets ranked according to their divergence, and it enables users to explore the lattice of frequent itemsets, identifying data subgroups with anomalous behavior.

We describe both steps in detail below; in the next section, we will provide results that demonstrate the superiority of this divergence-driven hierarchical discretization, compared with the prior non-hierarchical approach.

B. Hierarchical Attribute Discretization

To generate the item hierarchies for a continuous attribute, we inductively generate a binary tree: each node of the tree represents an item, and when a node is split into two children, the corresponding item is refined into the children items. An example tree for the *#prior* attribute of the *compas* dataset is depicted in Figure 1.

The tree is constructed in a way that closely parallels the generation of decision trees, except that the splitting is driven by divergence rather than by label class. The root of the tree for attribute A represents the complete range of A. A node can be split into two children with respect to a value $a \in \mathcal{D}_A$ and the two children are labeled $A \leq a$ and $A > a$. To each node ν we associate the item $I(\nu)$, which has the form $A \in J(\nu)$, where $J(\nu)$ is the interval describing the range of values of A for which ν can be reached from the root.

The trees we use for discretization are constructed much like decision trees [15], [16], by starting from a root node representing all data, and by recursively splitting the leaf nodes. At each iteration, a node ν is chosen and a value $a \in J(\nu)$, such that both the following conditions hold: (i) the support of each of the two nodes ν', ν'' resulting from the split of ν for value a of A is above a prescribed support level, and (ii) a is chosen among values satisfying (i) in a way that maximizes the *gain* of the split, to be defined below.

Thus, the split occurs on node ν for an attribute-value pair that maximizes the split gain under the support constraint. The construction of the tree terminates when either no more nodes can be split under the support constraint, or when the gain due to further splitting falls below a specified threshold. For decision trees, the split gain is defined in terms of instance labels. For our discretization trees, we propose two split criteria, one based on entropy and one on divergence.

Entropy-based gain criterion. Recall that our ultimate goal is to determine items, and thus itemsets, where a measure $f : 2^D \mapsto \mathbb{R}$ assumes divergent values with respect to the dataset as a whole. The entropy-based criterion is applicable when the measure f can be defined on the basis of a *boolean* outcome function $o : D \mapsto \{\mathtt{T}, \mathtt{F}, \perp\}$, as follows:

$$f_o(S) = \frac{\#\{x \in S \mid o(x) = \mathtt{T}\}}{\#\{x \in S \mid o(x) \neq \perp\}} = \frac{k^+}{k^+ + k^-} \ , \qquad (1)$$

where $S \subseteq X$ is the set whose measure we are taking, and

$$k^+ = \#\{x \in S \mid o(x) = \mathtt{T}\} \qquad k^- = \#\{x \in S \mid o(x) = \mathtt{F}\} \ .$$

Intuitively, $f_o(S)$ is the *probability* that the outcome is true, discounting from the computation of the probability the elements with outcome $\perp$. Many measures f of interest in the study of the performance of classifiers can be expressed via boolean outcome functions [5]; for instance, to capture the false-positive rate, we can let $o(x) = \mathtt{T}$ for false-positives, and $o(x) = \mathtt{F}$ for true-positives, and $o(x) = \perp$ for every negative instance x.

In classification trees, entropy-based criteria are used to split nodes so as to increase the statistical purity of the resulting nodes. Similarly, we can use entropy-based criteria to split instances according to the purity of the outcome function. Since $f_o(S) = k^+/(k^+ + k^-)$ is a probability, we can define the entropy $H(S, f_o)$ of the outcome over a set of instances S by

$$H(S, f_o) = -f_o(S)\log(f_o(S)) - (1 - f_o(S))\log(1 - f_o(S)) \ . \qquad (2)$$

As in decision trees, we use the entropy $H(S, f_o)$ to measure the quality of a split: the lower the entropy, the higher the purity, the better the split. To favor balanced splits, we weigh the entropy by the size of the split nodes, as common in classification trees [15], [16]. Thus, we let the gain of the split of S into S_1, S_2 be:

$$\begin{aligned} g(S_1, S_2 \mid S, f_o) = \frac{S}{\#D} H(S, f_o) - \\ - \left[\frac{\#S_1}{\#D} H(S_1, f_o) + \frac{\#S_2}{\#D} H(S_2, f_o)\right] \ . \end{aligned} \qquad (3)$$

Divergence-based gain criterion. The entropy-based splitting criterion can only be applied for measures f that have the form of a probability. For more general measures $f : 2^D \mapsto \mathbb{R}$, we define the gain directly with reference to f itself, via:

$$g(S_1, S_2 \mid S, f) = \frac{\#S_1}{\#D} \cdot |f(S_1) - f(S)| + \frac{\#S_2}{\#D} \cdot |f(S_2) - f(S)| \ . \qquad (4)$$

Building the discretization from the trees. From each attribute tree, we create an item hierarchy for the attribute; together, the item hierarchies for all continues attributes are used as discretization.

For each attribute, the *leaf items*, corresponding to the leaves of the tree, define a discretization consisting of non-overlapping intervals. These leaf items can be used directly by anomalous subgroup identification tools as [4]–[6] that perform a *base* exploration and do not consider the hierarchical relationships among items. In our pipeline, we use all items, including those corresponding to internal nodes: this hierarchical approach, as discussed, gives us greater flexibility in constructing itemsets above the support size, and will be helpful in the detection of divergent data subgroups.

Discussion. An alternative to deriving an individual tree for each continuous attribute is to consider all attributes jointly and build a single combined tree.

The advantage of a combined tree is that it is well suited to capturing dependencies between attributes. However, combined trees have several drawbacks. First, it is difficult to guarantee that each attribute is discretized. The tree is constructed with a single minimum-support constraint, and once nodes reach that minimum support, they are no longer split, regardless of whether all continuous attributes have been split. Second, it is very difficult to control the granularity of the identified splits due to the single minimum-support constraint.

Third, combined trees do not give rise to an item hierarchy for each attribute. Each attribute may be split differently across the nodes of the tree since attribute dependencies are considered. Finally, when there is a large number of discrete attributes, building a combined tree can be less efficient than building individual trees. When splitting a node in the combined tree, all attributes (discrete and continuous) need to be considered, to choose the optimal split. All these four properties are instead enabled by the individual-tree discretization.

Hierarchies for Categorical Attributes. For a categorical attribute A, we simply take as items all items $A = a$ for all $a \in \mathcal{D}_A$; further, we add all items corresponding to the item hierarchy. For example, if the attribute is an IP address, we can consider attributes of the form $A = a$ for each IP address a, and also, $A = a_l$, where a_l is the truncation of a to its first l bytes, for $1 \leq l \leq 3$. Thus, an IP address such as 118.114.119.88 will belong both to the item 118.114.119.88, and to the items 118.114.119, 118.114, and 118.

C. Generalized divergence subgroup extraction

The original DivExplorer exploits well-know frequent pattern mining algorithms Apriori [20] and FP-Growth [21] to extract frequent itemsets. These algorithms, per se, do not cope with generalized itemsets: they assume that all items for the same attribute have disjoint support, and they are unaware of any hierarchical relation between them. We extend the DivExplorer algorithm to deal with item hierarchies and extract generalized itemsets by using generalized frequent pattern (GFP) techniques [10], [11]. In particular, we integrated the divergence computation with the generalized versions of both the *Apriori* [20] and the *FP-growth* [21] algorithms.

The high-level description of the algorithm for the generalized divergence subgroup extraction is reported in Algorithm 1. Let Γ be a hierarchical discretization for a dataset D, consisting in a hierarchy of items for each attribute of dataset D. The hierarchies in Γ include both the predefined hierarchies for categorical attributes, and the hierarchies obtained via our hierarchical discretization process for continuous attributes. For each $step_i$ of a generic GFP technique, generalized itemsets are extracted (Line 3). The general function *extractGeneralizedItemsets* extracts the generalized itemsets; its implementation varies depending on the GFP used (e.g., $step_i$ could be level i iteration in Apriori [20], or the recursive step of FP-growth [21] on the FP-tree). During the computation of the itemsets, we accumulate not only the support count, as done by all mining algorithms, but also the total of the outcome function in each itemset (and the count of how many outcome values are $\perp$). In this way, after the frequent itemset extraction is done, we can compute not only the support size of each itemset, but also its divergence, without requiring any additional pass over the original dataset. Hence, our algorithms inherit the same computational efficiency as the frequent pattern mining algorithms from which they are derived.

Algorithm 1: Generalized divergence subgroup extraction.

```
Input: $D$, $f$, $\Gamma$, $s$
Output: Generalized itemsets divergence $GI_\Delta$
1 $GI$s_divergence=[];
2 for $step_i$ in Generalized Frequent Pattern Mining steps do
3     $I_{step_i}$ = extractGeneralizedItemsets($D, \Gamma, s, step_i$);
4     for $I$ in $I_{step_i}$ do
5         $I$.s, $I.\Delta_f$ = evaluate_itemset($I$, $f(D)$)
6         if $I.s \geq s$ then
7             $GI$s_divergence.append($I$);
8 return $GIs$_divergence
```

VI. Experimental results

We evaluate the proposed hierarchical discretization and the hierarchical-subgroup discovery with a wide set of experimental evaluations. First, we examine the ability of our hierarchical-subgroup exploration to identify highly divergent itemsets (Section VI-A) Next, we evaluate the ability of the hierarchical discovery to reveal anomalous behavior in the data (Section VI-B). We then study the impact of minimum support parameters on the identification of problematic behaviors (Section VI-C). Finally, Section VI-D presents the performance analysis of our hierarchical-subgroup exploration approach.

A. Identification of Highly Divergent Itemsets

In our first series of experiments, we compare the effectiveness of our hierarchical-subgroup exploration pipeline to discover highly divergent subgroups. We apply the tree-based hierarchical discretization technique for each continuous attribute with a uniform support threshold $s_{tree} = 0.1$, so that the generated items will contain at least 10% of the dataset instances. We, therefore, obtain a set of discretization ranges (the leaf nodes) and the set of item hierarchies, one for each continuous attribute. We then compare the two following subgroup discovery strategies: the *base* one with DivExplorer [5] and our *generalized* (or hierarchical) exploration. The base exploration only considers the discretized dataset determined by the leaf nodes. The generalized exploration instead also leverages the entire set of item hierarchies. We refer as *base itemsets* the one extracted by the *base* DivExplorer and with *generalized itemsets* the itemsets derived by our hierarchical exploration based on GFP mining. For each subgroup discovery strategy, we measure the maximum divergence of the extracted itemsets.

The datasets: compas and folkstables. We perform the experiments on two public datasets: *compas* and *folkstables*. The *compas* dataset provides data on criminal defendants, such as age, gender, race, number of prior offenses, and whether the charge under consideration is a felony or a misdemeanor [9]. Each defendant also has a score indicating the probability of recidivism in the two years subsequent the charge, computed according to the proprietary COMPAS algorithm; we consider high-risk scores ($\geq$ 8) as a prediction of recidivism. For each defendant, we also know the true recidivism; we can thus compute the false-positive rate, which is the probability that a defendant is incorrectly assumed to recidivate. We will consider the divergence of the false-positive rate, which indicates which subgroups are more likely to be incorrectly predicted to recidivate. The *compas* dataset includes 6,172 instances, and has continuous attributes *age*, *#prior* (number of prior offenses) and *stay* (number of days spent in jail), and discrete attributes gender, charge degree (misdemeanor or felony) and the ethnicity of the defendant.

The *folkstables* dataset is based on US Census data; we use here the data for the census year 2018 and the state of California [22]. We use the attributes of the income prediction task, and we consider the divergence of the income (the measure f we consider is directly the income) [23]. The dataset includes two continuous attributes: *age* and *number of hours per week* that a person works. The dataset consists of 195,665 instances and a total of 10 attributes. We also consider item hierarchies for two categorical attributes: the place of birth (*POBP*) and occupation (*OCCP*). The item hierarchy for the place of birth is a geographical hierarchy. The occupation item hierarchy is obtained from the data themselves. Each occupation category available in the dataset (e.g., *MGR-Financial Managers*, *MED-Dentists*) is mapped to its supercategory (e.g., *MGR*, *MED*).

HIGHLY DIVERGENT ITEMSETS. We first evaluate the ability of the hierarchical exploration compared to base explorations in revealing divergent subgroup by inspecting the most divergent patterns on the two datasets.

Top divergent itemsets in compas. Table II reports the top FPR-divergent itemsets of the *compas* dataset according to these three setting of divergent subgroup exploration:

- we discretize the continuous attributes using the *manual discretization* (i.e., user-defined) used in [5], [9] and we apply the base subgroup discovery of DIVEXPLORER
- we discretize using our tree discretization using the leaf items and we perform base DIVEXPLORER
- we apply the proposed hierarchical-subgroup exploration pipeline, based on the tree discretization to extract the hierarchy trees and the generalized subgroup discovery.

For the tree discretization strategies, we used divergence as the splitting criterion $g\Delta$ and support threshold limit s_{tree} of 0.1, as in the previous experiments. The divergence subgroup exploration uses support thresholds $s = 0.05, 0.025$, and 0.01.

We see that the tree-generated discretizations are superior to the manual one, allowing to reveal higher divergence for all support thresholds. We also see that the generalized subgroup exploration which leverages the items at all levels of hierarchy in the tree leads to the identification of more divergent itemsets, due to the flexibility afforded by being able to combine items with different granularity into itemsets.

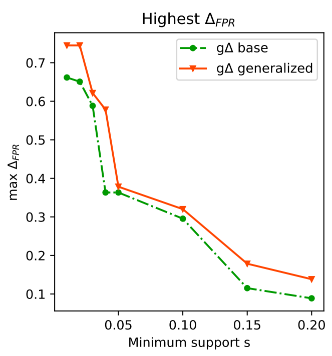
Divergent itemsets in folktable. Table III shows the base and generalized divergent itemsets with highest divergence in absolute terms for the *folktable* dataset. We use the same configuration of the compas experiment: divergence as splitting criterion with $s_{tree} = 0.1$ and support thresholds s of 0.05, 0.025 and 0.01. For all the support thresholds, the hierarchical exploration of divergence reveals the highest divergence. Consider for example the generalized itemset {*AGEP≥35.0, OCCP=MGR, SEX=Male*} which have the highest divergence for $s = 0.05$. Men with age greater or equal than 35 with a managerial occupation have an average income that is 90.2k greater than the average income of the overall data. Both *AGEP≥35.0* and *OCCP=MGR* represents two aggregations. The former is an aggregation of a more fine grained discretization of age ranges, induced by our hierarchical discretization approach. The latter aggregates more fine grained occupation categories as *MGR-Sales Managers* and *MGR-Financial Managers*. Only thanks to the hierarchical exploration we are able to identify these interesting subgroups at different level of the taxonomies.

Depending on our context, we may be interested not only in revealing the most divergent subgroups but also in identifying the ones whose divergent behavior affects more instances. We refer with $wlogr$ to the log-ratio between the statistic f for the itemset and on the overall dataset, weighted for the itemset support. Table IV shows the base and generalized divergent itemsets with highest $wlogr$ in absolute terms for the *folktable* dataset. The generalized exploration is again able to adapt and reveal the highest absolute $wlogr$. Consider the generalized itemset *WKHP≤39*. Compared to the finer discretization *WKHP≤29*, it affects more instance, specifically 1/3 of the datasets, with a divergence that is lower by 7.4k. A practitioner may decide which policy apply: prioritize more divergent itemsets or focus on a larger portion of the data affected by divergence. In both scenarios, generalized itemsets allow revealing the itemsets on which it is best to focus.

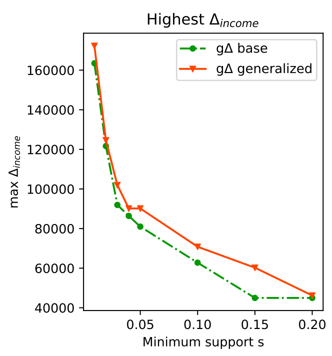
HIERARCHICAL SUBGROUP EXPLORATION. In the following set of experiments, we quantitatively evaluate the impact on divergence of Figures 3a and 3b summarize the maximum divergence of the itemsets that the *base* exploration of DIVEXPLORER and our generalized exploration can identify for *compas* (a) and *folkstables* (c). We see how the hierarchical exploration provides better performance than when only leaf items are used. The flexibility provided by having items at multiple levels of granularity is beneficial in the exploration phase.

Figures 3c and 3d report the number of explored frequent itemsets, again according to the different exploration strategies and gain criteria. The hierarchical exploration extracts more itemsets, since, by definition, base itemsets are a subset of the generalized ones.

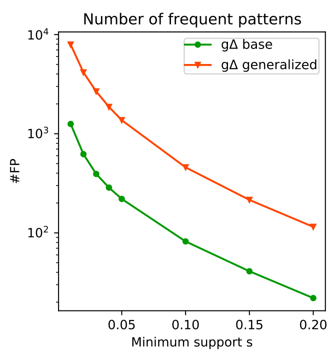
For the *compas* dataset, we also compare the divergence results for both the entropy-based gain criterion and the divergence-based one (*entr* and $g\Delta$ in the figures) since the false-positive rate can be defined by means of a boolean outcome function and therefore the entropy criterion can be adopted. The results are reported in Figure 4. We can



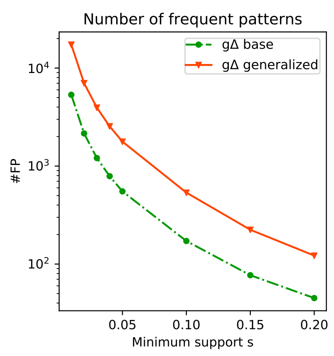
(a) Highest FPR divergence, *compas*



(b) Highest income divergence, *folkstables*



(c) Number of frequent patterns, *compas*



(d) Number of frequent patterns, *folkstables*

Fig. 3: Highest identified divergence and number of frequent itemsets (#FP) resulting from exploring with the *base* DIVEXPLORER using the leaf items and our generalized subgroup discovery using item hierarchies varying the tree discretization strategies (with s_{tree}=0.1) for the *compas* and *folkstables* datasets. We provide the trend for splitting based on the divergence criteria. The horizontal axis reports the support threshold s for the exploration.

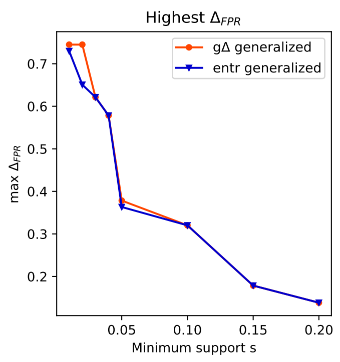


Fig. 4: Highest divergence found with by our hierarchical-subgroup exploration with the divergence ($g\Delta$) and entropy (*entr*) criteria for the *compas* dataset.

observe that the results generally match, with the divergence-based gain performing slightly better for some values of s, corroborating its proposal as an alternative gain criterion.

B. *Identification of anomalies*

We now experimentally demonstrate the superiority of the hierarchical-subgroup exploration over the base one when no hierarchy is considered and its superiority over unsupervised discretizations.

We perform experiments on an *artificial* dataset, in which a divergent behavior is inserted in a controlled fashion. We compare problematic identification results for the hierarchical exploration and the base exploration of subgroup identification approaches, evaluating their ability to identify the injected divergent behavior. We then evaluate the advantages of our hierarchical discretization over flatten unsupervised discretizations for the identification of anomalous behaviors.

The dataset. The dataset consists of 10,000 points randomly chosen in the 3-dimensional space $[-5, 5]^3$; each coordinate corresponds to an attribute. We set the class label to T or F with equal probability. We then inject a gaussian error rate by setting the predicted class label as follows. We define a multivariate normal random variable with a mean of [0, 1, 2] and covariance of 1. We first generate prediction labels that reflect the class labels. We then flip the prediction labels with a probability equal to the normalized multivariate normal distribution. The error rate is therefore a function of the normalized gaussian distribution of the three attributes.

ANOMALY IDENTIFICATION. We first compare the anomaly identification results for the *base* exploration and the *generalized* one. We first derive the hierarchical splits for each attributes, using $s_{tree} = 0.1$ and the divergence-based gain criterion. The leaf items are used for the base exploration with DIVEXPLORER. The entire hierarchy for all the attributes is instead the input of our generalized exploration. Figures 5a and 5c visualize the itemset with highest divergence for the *base* and *generalized* explorations respectively with s=0.05. Each figure shows, for a given attribute, the projection of the normalized multivariate normal distribution used to generate errors and the ranges identified in the itemsets. The base itemset {*b=[-0.19-1.34]*} with divergence $\Delta_{error} = 0.046$ is the one with the highest divergence (Figure 5a). Base itemsets tend to represent specific ranges of the input, with a support that is near the threshold s_{tree}. As a result, even with a minimum support threshold $s = s_{tree}/2$, base itemsets fail to capture the interesting association of attribute values. On the other hand, the generalized itemset with the highest divergence includes correctly all tree attributes with a divergence Δ_{error} of 0.212 (Figure 5c). The hierarchical exploration considers multiple splits for the same attributes. As a result, the exploration adapts more effectively to the most suitable range given the input threshold.

Figures 5b and 5d show the result with s=0.025. By lowering the support, the base exploration identifies 2 out of three terms with a divergence of 0.229 (Figure 5b) . On the other

s	Discretization approach	Itemset	Sup	Δ_{FPR}	t
0.05	Manual discretization + base [5]	age=[25-45], #prior>3, race=Afr-Am, sex=Male	0.13	0.220	7.1
	Tree discretization + base [5]	#prior≥9, race=Afr-Am	0.09	0.363	8.
	Tree discretization + generalized	age<=32, stay≥3.0, #prior≥4, sex=Male	0.06	**0.378**	6.7
0.025	Manual discretization	age=[25-45], stay=1w-3M, #prior>3, race=Afr-Am, sex=Male	0.03	0.292	4.4
	Tree discretization + base [5]	age=[28-32], #prior≥9, sex=Male	0.03	0.590	6.8
	Tree discretization + generalized	age=[25-32], charge=F, #prior≥9, sex=Male	0.03	**0.621**	7.7
0.01	Manual discretization	age<25, charge=F, #prior>3	0.02	0.618	5.7
	Tree discretization + base [5]	age≤24, charge=F, #prior=[4-8]	0.02	0.662	6.2
	Tree discretization + generalized	age=[25-32], stay>=3.0, #prior≥9	0.02	**0.745**	8.1

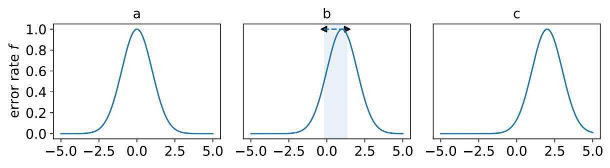
TABLE II: Top divergent itemsets found by *base* DIVEXPLORER using (i) a manual discretization, (ii) the leaf items of the tree discretization and (iii) our hierarchical-subgroup exploration pipeline for different support thresholds s for *compas* dataset (s_{tree}=0.1, $g\Delta$ gain).

s	Itemset type	Itemset	Sup	Δ_{income}	t
0.050	base	MAR=Married, RAC=White, SEX=Male, WKHP ≥44.0	0.07	81.0k	62.3
	generalized	AGEP≥35.0, OCCP=MGR, SEX=Male	0.05	**90.2k**	60.6
0.025	base	SCHL=Prof beyond bachelor	0.03	105.3k	46.7
	generalized	AGEP≥35.0, OCCP=MGR, SEX=Male, WKHP≥44.0	0.03	**119.3k**	50.6
0.010	base	SCHL=Prof beyond bachelor, WKHP≥44.0	0.01	163.5k	40.3
	generalized	AGEP≥35.0, SCHL=Prof beyond bachelor, SEX=Male, WKHP≥40.0	0.01	**172.3k**	39.3

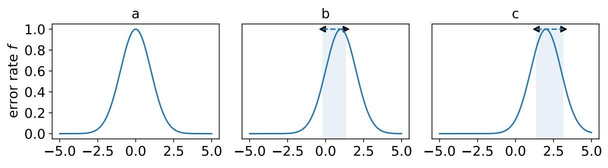
TABLE III: Top divergent itemsets found by *base* DIVEXPLORER using the leaf items of the tree discretization and our hierarchical-subgroup exploration pipeline for different support thresholds s for *folkstable* dataset (s_{tree}=0.1, $g\Delta$ gain).

Itemset type	Itemset	Sup	$\Delta_{outcome}$	t	wlogr
base	AGEP≤26.0	0.17	-43.9k	173.8	-0.191
base	AGEP≤26.0, MAR=Never married/<15yrs	0.15	-44.9k	175.9	-0.179
base	WKHP≤29.0	0.18	-40.4k	122.5	-0.178
generalized	WKHP≤39.0	0.31	-33.0k	108.5	**-0.216**
generalized	AGEP≤34.0, WKHP≤39.0	0.14	-49.9k	203.6	**-0.197**
generalized	AGEP≤34.0	0.34	-28.0k	104.3	**-0.193**

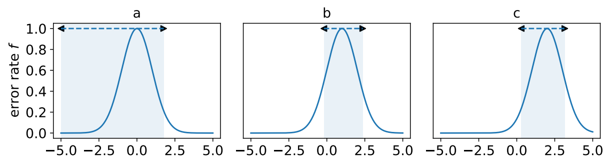
TABLE IV: Top-3 itemsets with highest (absolute) $wlogr$ found by *base* DIVEXPLORER using the leaf items of the tree discretization and our hierarchical-subgroup exploration pipeline for *folkstable* dataset (s_{tree}=0.1, $g\Delta$ gain, s=0.01).



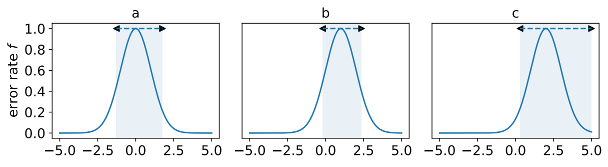
(a) Base itemset, s=0.05, Δ_{error} = 0.045



(b) Base itemset, s=0.025, Δ_{error} = 0.229

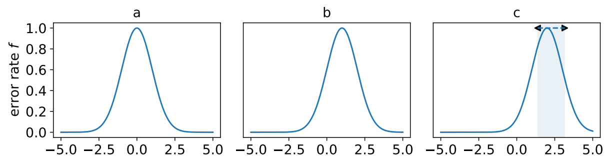


(c) Generalized itemset, s=0.05, Δ_{error} = 0.212

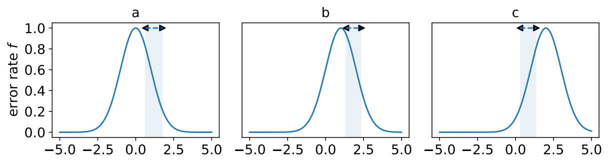


(d) Generalized itemset, s=0.025, Δ_{error} = 0.297

Fig. 5: Ranges of the itemset with highest divergence obtained via the *base* and our *generalized* one with s=0.05 and s=0.025.



(a) Base itemset, default parameters



(b) Base itemset, effect size threshold = 1

Fig. 6: Ranges of the itemset with highest divergence obtained via the *base* with SliceFinder.

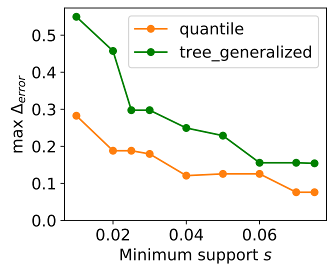


Fig. 7: Highest divergence found by our hierarchical-subgroup exploration pipeline and the best result for the quantile discretization with base DivExplorer for the *artificial* dataset.

hand, the generalized exploration adapts to the lower frequency threshold, revealing smaller split ranges with a divergence of 0.297 (Figure 5d).

This experiment demonstrates the advantage of the hierarchical exploration in adapting to select the most suitable split ranges. While we illustrate this example with three attributes for clarity and visualization purposes, we remark that the shortcomings of base itemsets are exacerbated when the number of attributes increases.

Comparison with subgroup identification approaches. We compare the hierarchical exploration with two other base exploration approaches: Slice Finder [24] and SliceLine [6]. Slice Finder and Slice Line, as the base DivExplorer, identify slices of the data, identified by *base itemsets*, in which the model performs poorly. The three approaches mainly differ on two aspects: the measure of 'problematic' behavior and the exploration approaches. Slice Finder measures the problematic behavior of a slice with respect to the 'remainder' of the dataset while both DivExplorer and SliceLine compare with the overall dataset. Moreover, Slice Finder considers the classifier loss on the data slice and uses the effect size to measure the effect of the difference in the behavior; while SliceLine focuses on model errors and defines a scoring function that considers both the average slice error rate and the size of the slice. The notion of divergence we adopt in our hierarchical exploration is instead general and captures the difference in the behavior for a slice and the overall data for a generic metric (e.g., error rate, false positive and negative rates, average scores). Concerning the exploration aspect, all the approaches perform a lattice search, with different stopping criteria. Slice Finder performs a not exhaustive search of the top-k slices subject to a minimum effect size threshold: the lattice exploration stops when a sufficiently large deviation is found. SliceLine and base DivExplorer both leverage frequent pattern mining techniques. While base DivExplorer only uses a minimum support threshold, SliceLine additionally considers the monotonicity properties of errors and the proposed scores for pruning. Our hierarchical subgroup exploration, differently from all base exploration approaches, leverages instead generalized frequent pattern mining techniques to identify subgroups with different granularity. It is difficult to compare SliceFinder and SliceLine with our hierarchical exploration on a general dataset since the three approaches measure the problematicness and explore data slices differently. We hence leverage the artificial dataset for which the problematic behavior is known and we evaluate the ability of SliceFinder and SliceLine[1] to identify it. We discretize the data using leaf items as in the previous experiments with base DivExplorer.

Figure 6a shows the itemset with highest effect size identified by SliceFinder with default parameters. Since the slice of $\{$*(c=[1.37-3.16])* $\}$ is already problematic (with an effect size of 0.79 which is greater than the input threshold of 0.4) the search stops. Only if we increase the effect size to 1, Slice Finder identifies as the slice with the highest effect size as composed by the three terms (Figure 6b). We note that however, the slice represents only 13 instances (out of all 10000 instances). SliceFinder top-k exploration does not control the slice frequency and prevents the identification of larger data slices.

The base exploration of SliceLine suffers from the same limitation as the base exploration of DivExplorer. We consider the same minimum support thresholds for SliceLine of 0.05 and 0.025 as in the previous set of experiments. We vary the weight parameter α of SliceLine for the importance of the average slice error and we consider the best slice in terms of the highest error rate. The best itemsets for the 0.05 and 0.025 support thresholds of SliceLine match the ones identified by base DivExplorer, reported in Figures 5a and 5b respectively. The results show the limitation of the base explorations of existing approaches to reveal large and frequent slices associated with a problematic behavior when a fixed discretization is used and the benefit of adopting a hierarchical exploration.

Comparison with unsupervised discretization. We compare the quality of our hierarchical-subgroup exploration pipeline with the proposed tree discretization with the quantile discretization. We note that, since the data are generated uniformly, the quantile discretization corresponds to the uniform one. Contrary to our discretization, these are *unsupervised* discretizations, in the sense that the discretization itself does not depend on the measure f whose anomalies we wish to explore. In contrast, our technique relies on decision trees, which are a supervised technique, as the discretization depends on a gain function, defined in our case via entropy or divergence.

We vary the number of bins for the discretization from 2 to 10. We then extract the itemsets and their divergence via the base subgroup discovery of DivExplorer using the discretized ranges. We consider the input bin size which achieved the highest divergence for each evaluated minimum support threshold and we reported the results in Figure 7. The generalized form achieves the highest results for all the input

[1]We adopted the implementation available at https://github.com/DataDome/sliceline.

thresholds. The experiments show the ability of the generalized exploration to adapt to identifying the best granularity.

C. Sensitivity analysis

The experimental results reported in Figures 3a, 3b and 7 also allows us to study the impact of the minimum support s for the exploration on divergence *compas*, *folkstables* and the *artificial* dataset respectively. As expected from Property **??**, finer subdivisions lead to higher divergence.

The item hierarchies for the continuous attributes are obtained by keeping fixed the minimum support of the tree nodes s_{tree} to 0.1 for the item hierarchy generation. As a result, all the generated items in the hierarchy are at least represented with support $s_{tree} = 0.1$ in the dataset. We now evaluate the impact of minimum support of the tree nodes s_{tree} on divergence, keeping fixed the minimum support for the exploration s to 0.025.

Figures 8a and 8b show the maximum divergence of the itemsets with the base and generalized exploration varying s_{tree} for the *compas* and *artificial* datasets. For the *folktables* dataset, the highest divergence Δ_{income} is stable for all the evaluated s_{tree} from 0.005 to 0.2, therefore results are not reported in a plot. Specifically, we have Δ_{income}=105.3k for the base exploration and Δ_{income}=119.3k for the generalized one. The results for the generalized exploration of the other two datasets are stable for a wide range of s_{tree}. Only when the require minimum node support s_{tree} is high we observe a drop in divergence for *compas* and the *artificial* dataset. Specifically, it is observed only when we require interval ranges (i.e., the generated items) to be frequent in the dataset more than 15% for the *artificial* and more than 12.5% for *compas*. The base exploration is instead sensible to the s_{tree} parameter. For $s_{tree} < s$, leaf items generated via tree discretization may not explored since they may be lower than the support thresholds. Therefore, generated itemsets are based on categorical terms (if any) or fewer combination of items above the support.

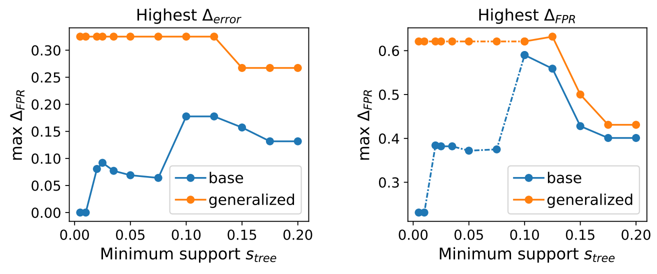
The generalized exploration not only is more stable varying s_{tree} but also always reveals the highest divergence. The results again show the adaptability of the generalized exploration for the identification of anomalous behavior.

D. Performance analysis

The experiments were performed on a PC with Ubuntu 16.04.1 LTS 64 bit, 16 GB RAM, 2.40GHz×4 Intel Core i7. For the *compas* dataset, the discretization process takes in all cases less than 0.2s. The subsequent hierarchical subgroup exploration can take up to 2.75s, when $s = 0.01$ is used. The discretization process for the *folkstable* dataset takes less than 14.0s and the generalized itemset exploration takes up to 22.5s with support $s = 0.01$.

VII. Conclusions and Future Work

We propose a novel hierarchical-subgroup exploration pipeline to identify anomalous subgroups. The exploration pipeline consists of two main steps. The first step induces



(a) *artificial* (b) *compas*

Fig. 8: Highest divergence for base itemsets and for generalized itemsets varying the minimum support of the node s_{tree} for the tree construction for the *artificial* and *compas* datasets. s=0.025.

a discretization hierarchy on each continuous attribute, thus allowing the explicit representation of different granularity levels for the discretization. The simultaneous availability of discretization intervals at different detail level is then leveraged by our hierarchical subgroup identification algorithms which allows the identification of anomalous subgroups at the optimal granularity level. We evaluated the performance of our hierarchical-subgroup exploration pipeline on both real and synthetic datasets, showing its effectiveness in revealing anomalous behaviors for multiple granularities of attribute values.

As future work, we plan to integrate a dynamic and interactive drill-down exploration of divergent subgroups across item hierarchies.

References

[1] S. Barocas and A. D. Selbst, "Big data's disparate impact," *Calif. L. Rev.*, vol. 104, p. 671, 2016.

[2] V. Eubanks, *Automating inequality: How high-tech tools profile, police, and punish the poor*. St. Martin's Press, 2018.

[3] C. O'neil, *Weapons of math destruction: How big data increases inequality and threatens democracy*. Crown, 2016.

[4] Y. Chung, T. Kraska, N. Polyzotis, K. H. Tae, and S. E. Whang, "Automated data slicing for model validation: A big data - ai integration approach," *IEEE TKDE*, vol. 32, no. 12, pp. 2284–2296, 2020.

[5] E. Pastor, L. de Alfaro, and E. Baralis, "Looking for trouble: Analyzing classifier behavior via pattern divergence," in *SIGMOD/PODS '21*, 2021, p. 1400–1412.

[6] S. Sagadeeva and M. Boehm, "SliceLine: Fast, linear-algebra-based slice finding for ML model debugging," in *SIGMOD/PODS '21*, 2021, p. 2290–2299.

[7] Y. Huhtala, J. Kärkkäinen, P. Porkka, and H. Toivonen, "Tane: An efficient algorithm for discovering functional and approximate dependencies," *The computer journal*, vol. 42, no. 2, pp. 100–111, 1999.

[8] S. Kruse and F. Naumann, "Efficient discovery of approximate dependencies," *VLDB '18*, vol. 11, no. 7, pp. 759–772, 2018.

[9] J. Angwin, J. Larson, S. Mattu, and L. Kirchner, "Machine bias," May 2016. [Online]. Available: https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing

[10] J. Han and Y. Fu, "Discovery of multiple-level association rules from large databases," in *VLDB*, vol. 95. Citeseer, 1995, pp. 420–431.

[11] R. Srikant and R. Agrawal, "Mining generalized association rules," in *VLDB'95*, 1995, pp. 407–419.

[12] Á. A. Cabrera, W. Epperson, F. Hohman, M. Kahng, J. Morgenstern, and D. H. Chau, "Fairvis: Visual analytics for discovering intersectional bias in machine learning," in *2019 IEEE VAST*. IEEE, 2019, pp. 46–56.

[13] M. Research, “Error analysis of the responsible ai toolbox.” 2021. [Online]. Available: https://erroranalysis.ai

[14] H. Liu, F. Hussain, C. L. Tan, and M. Dash, “Discretization: An enabling technique,” *Data mining and knowledge discovery*, vol. 6, no. 4, pp. 393–423, 2002.

[15] J. R. Quinlan, “Induction of decision trees,” *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.

[16] ——, *C4.5: programs for machine learning*. Morgan Kaufmann, 1993.

[17] R. Kohavi and M. Sahami, “Error-based and entropy-based discretization of continuous features.” in *KDD*, 1996, pp. 114–119.

[18] U. Fayyad and K. Irani, “Multi-interval discretization of continuous-valued attributes for classification learning,” 1993.

[19] E. Pastor, L. de Alfaro, and E. Baralis, “Identifying biased subgroups in ranking and classification,” 2021.

[20] R. Agrawal and R. Srikant, “Fast algorithms for mining association rules in large databases,” in *VLDB ’94*, 1994, p. 487–499.

[21] J. Han, J. Pei, and Y. Yin, “Mining frequent patterns without candidate generation,” in *ACM SIGMOD*, 2000, pp. 1–12.

[22] F. Ding, M. Hardt, J. Miller, and L. Schmidt, “Retiring adult: New datasets for fair machine learning,” *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[23] S. Flood, M. King, R. Rodgers, S. Ruggles, and J. R. Warren, “Integrated public use microdata series, current population survey: Version 8.0 [dataset]. minneapolis, mn: Ipums,” IEEE, pp. 1719–1723, 2020.

[24] Y. Chung, T. Kraska, N. Polyzotis, K. H. Tae, and S. E. Whang, “Slice finder: Automated data slicing for model validation,” in *IEEE ICDE*, 2019, pp. 1550–1553.