A Hierarchical Approach to Anomalous Subgroup Discovery

A Hierarchical Approach to Anomalous Subgroup Discovery

Eliana Pastor
Politecnico di Torino
Torino, Italy
eliana.pastor@polito.it

Elena Baralis
Politecnico di Torino
Torino, Italy
elena.baralis@polito.it

Luca de Alfaro
UC Santa Cruz
Santa Cruz, USA
luca@ucsc.edu

***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. Current approaches to subgroup exploration ignore the presence of hierarchies in the data, and can only be applied to discretized attributes. The discretization process required for continuous attributes may significantly affect the identification of relevant subgroups.**

We propose a hierarchical subgroup exploration technique to identify anomalous subgroup behavior at multiple granularity levels, along with a technique for the *hierarchical discretization* of data attributes. The hierarchical discretization produces, for each continuous attribute, a hierarchy of intervals. The subsequent hierarchical exploration can exploit data hierarchies, selecting for each attribute the optimal granularity to identify subgroups that are both anomalous, and with enough elements to be statistically and practically significant. Compared to non-hierarchical approaches, we show that our hierarchical approach is more powerful in identifying anomalous subgroups and more stable with respect to discretization and exploration parameters.

***Index Terms*—model evaluation, fairness, subgroup discovery, discretization, generalized itemsets, hierarchies, 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, i.e., 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 [4]–[11]. 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 [12], [13]. 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 [14]. 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 8 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 be used.

Our discretization method naturally identifies a hierarchy or taxonomy for the attributes. We integrated our discretization approach into H-DIVEXPLORER, a novel subgroup identification approach that, by means of generalized itemset discovery [15], [16], 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 (0.01).

Our main contributions are as follows.

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. This limits the flexibility of the exploration: if fine slices are used to analyze some attributes in detail, then these slices cannot be combined with slices for other attributes without obtaining overly small subgroups, which may not be statistically or practically significant. 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 anomalous subgroups in a far more flexible way, selecting an appropriate granularity for each attribute.

Hierarchical discretization. Usually, discretizing a dataset means turning its continuous attributes into discrete ones, mapping the continuous values into non-overlapping ranges. We show 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.

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

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.

Hierarchical subgroup identification. H-DIVEXPLORER is a pipeline for discretization and hierarchical subgroup analysis capable of handling datasets with predefined hierarchies and continuous attributes. H-DIVEXPLORER first uses trees to discretize continuous attributes into interval hierarchies. The available hierarchies are then fed, jointly with the dataset, to our hierarchical exploration algorithm, based on an extension of [5], which detects anomalous subgroups with different granularities.

Polarity pruning. To improve the performance of subgroup exploration, we also propose *polarity pruning,* an heuristic that allows pruning the search space while preserving the quality of identified subgroups.

We provide open-source code implementing H-DIVEXPLORER at https://github.com/elianap/h-divexplorer.

The paper is organized as follows. Section II discusses related works. Section III provides our main definitions and

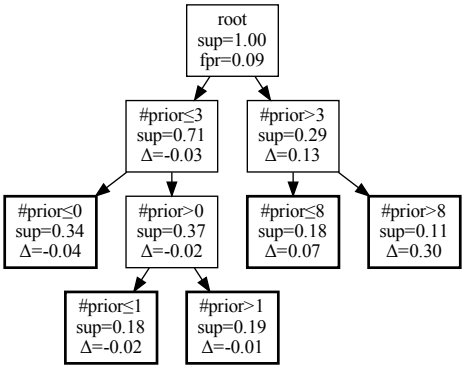


Fig. 1: Item hierarchy for the *#prior* attribute on the FPR of the *compas* dataset obtained via tree discretization.

preliminaries. Section IV introduces the notion of generalized subgroup and item hierarchy. Section V describes the tree discretization process and outlines our hierarchical subgroup exploration approach H-DIVEXPLORER. Section VI presents experimental results, showing the benefits of the hierarchical approach, and offering a detailed comparison with prior non-hierarchical approaches. 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], [17], [18]. 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 (e.g., in [5] in terms of divergence) or with respect to its counterpart (e.g., 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 [18]. 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.

Lattice-based approaches suffer from a significant shortcoming: They require all attributes to be *discrete*. Any continuous attribute needs to be discretized before subgroup identification. Discretization induces a non-overlapping partitioning of attribute values. Hence, the discretization step is crucial. If it is too coarse, or if anomalous and normal data coexist in the same bucket, discretization may hamper the identification of the anomalous subgroups. In our work, we propose a *hierarchical discretization* approach for continuous attributes. The discretization method is driven by the notion of divergence [5] to improve the quality of the identified buckets. Differently from all the mentioned approaches [4]–[6], [18], our hierarchical subgroup exploration approach 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 [19]. 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 [20] and C4.5 [21]). For example, in [22], 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 [23], 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 use all tree nodes, rather than just the leaves, and obtain a hierarchical discretization. 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. In this work, we extend the DIVEXPLORER [5] algorithm to explore all the subgroups that can be formed with discretizations at different hierarchy levels. This flexibility is not available in previous lattice-based approaches [4]–[6]. Differently from [4], [18], we leverage trees to identify interesting ranges of continuous attributes in a hierarchical representation, rather than anomalous subgroups.

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. In this section, we first define datasets, then item and itemsets, which allow us to define data subgroups. We then outline the notion of divergence as a measure of peculiar behavior, Finally, we describe DIVEXPLORER, which implements non-hierarchical subgroup exploration.

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 dataset instances.

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 α.

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 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. Different outcome functions allow us to easily define statistics such as false and true positive rates and the corresponding negative rates [5], rates related to rankings [24], and more. Statistical significance of the divergence is measured by its t-value, 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 [25] and FP-Growth [26]. 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 exploration 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 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 (i.e., contains enough people to be statistically significant). But if we wish to study the situation in a particular zip code, dividing income in ranges of \$100 may be much too fine, as it would result in most itemsets having support below the threshold. A coarser income discretization might be necessary to study zip code and income jointly. 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 attribute 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 levels 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 $\mathcal{D}_{\beta_1} \cap \mathcal{D}_{\beta_2} = \emptyset$.

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 [12].

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

H-DivExplorer is an automated hierarchical-subgroup exploration approach based on the use of generalized items, where the discretization is guided by the divergence of the statistics of interest. It identifies divergent subgroups in two steps. 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, 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 illustrate the superiority of this divergence-driven hierarchical discretization, compared with the prior non-hierarchical approach.

A. Hierarchical Attribute Discretization

To generate the item hierarchy 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. 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 non-hierarchical subgroup identification methods such as [4]–[6]. Our method will use the entire tree.

Our discretization trees are constructed much like decision trees [20], [21], by starting from a root node representing all data, and by recursively splitting the leaf nodes in a way that maximizes the split gain under the support constraint. The root of the tree for attribute A represents the complete range of 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 associated with ν. A node ν can be split into two children ν', ν'' with respect to a value $a \in \mathcal{D}_A$; the two children correspond to the intervals $J(\nu) \cap \{A \mid A \leq a\}$ and $J(\nu) \cap \{A \mid A > a\}$. We choose the splitting point a so that (i) the support of each of the two nodes ν', ν'' 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.

The process greedily selects an interval split according to a local optimality criterion, in order to identify divergent items. We consider two such split optimality criteria: one tied to entropy, and a novel one tied directly to divergence. The construction of the tree terminates when no more nodes can be split under the support constraint. For decision trees, the split gain is defined in terms of instance labels. We describe below the split criteria based on entropy and divergence.

Entropy-based gain criterion. Entropy is commonly used to define splitting criteria for decision tree-induction algorithms [20], [21] and discretization techniques [22], [23].

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 has the form

of a probability, that is, when f is defined on the basis of a *boolean* outcome function $o : D \mapsto \{\mathrm{T}, \mathrm{F}, \bot\}$, as follows:

$$f_o(S) = \frac{\#\{x \in S \mid o(x) = \mathrm{T}\}}{\#\{x \in S \mid o(x) \neq \bot\}} = \frac{k^+}{k^+ + k^-} \ ,$$

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

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

Intuitively, $f_o(S)$ is the *probability* that the outcome is true, discounting from the computation of the probability the elements with outcome $\bot$. Many measures f of interest in the study of classifier performance can be expressed via boolean outcome functions [5]. For instance, to capture the false-positive rate, we let $o(x) = \mathrm{T}$ for false-positives, $o(x) = \mathrm{F}$ for true-positives, and $o(x) = \bot$ 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 [27]. 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

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

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 [20], [21]. Thus, we let the gain of the split of S into S_1, S_2 be:

$$g(S_1, S_2 \mid S, f_o) = \\ = \frac{S}{\#D} H(S, f_o) - \Big[\frac{\#S_1}{\#D} H(S_1, f_o) + \frac{\#S_2}{\#D} H(S_2, f_o)\Big] \ .$$

Divergence-based gain criterion. The entropy-based splitting criterion can only be applied for measures f that have the form of a probability, that is, that are defined using a boolean outcome function. Other node splitting methods, devised for quantitative outcomes, are typically driven by error functions such as mean square error [28], and are not applicable to an ensemble property such as divergence. Only the divergence-based criterion can be applied to the divergence of general outcome functions, such as the income in the *folkstables* dataset [29]. For a general measure $f : 2^D \mapsto \mathbb{R}$, we define the gain directly with reference to f, via:

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

The divergence criterion has a form similar to the entropy one, again balancing the gain (in this case, the divergence of the two intervals) by their size. When both applicable, the two criteria may yield different discretizations; our experiments will show that their effectiveness is similar.

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 control the granularity of the splits, or even 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, 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.

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 items 118.114.119.88, 118.114.119, 118.114, and 118.

B. Generalized divergence subgroup extraction

The original DIVEXPLORER exploits well-know frequent pattern mining algorithms Apriori [25] and FP-Growth [26] 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 [15], [16], [30]. In particular, we integrated the divergence computation into generalized versions of Apriori and FP-growth by inspiring our implementation to the works in [16], [31].

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 [25], or the recursive step of FP-growth [26] 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 $\bot$). In this way, at the end of

frequent itemset extraction, 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.

The hierarchical exploration considers items at all granularity levels, rather than just the finest (leaf) items. Hence, the hierarchical exploration considers a superset of the itemsets considered by non-hierarchical methods. Thus, for the same support threshold, hierarchical exploration is guaranteed to find itemsets (subgroups) that are at least as divergent as those found by non-hierarchical exploration.

C. Polarity pruning

When we seek itemsets with high (absolute-value) divergence, the divergence can be either positive or negative. *Polarity pruning* is a powerful heuristic for the identification of high-divergence itemsets. We apply polarity pruning to the items generated by our discretization trees, as they are obtained via a controlled splitting process that maximizes divergence.

When searching for itemsets with high positive divergence, the heuristics will combine into itemsets only items that, individually, cause positive divergence; and symmetrically for negative divergence. In other words, the heuristic attributes a *polarity* to each item: when considered in isolation, items with positive polarity increase divergence, and items with negative polarity decrease it. The heuristic only combines items with the same polarity to form itemsets: as this prunes the search space, we refer to it as *polarity pruning*.

Polarity pruning leads to a speedup that is typically exponential in the number of continuous attributes. Indeed, if there are n continuous attributes, assuming that for each attribute roughly half of the items have positive and negative polarity, the exploration speedup is of the order of 2^{n-1}. This because itemsets are created selecting at most one item per attribute, and the polarity criterion requires that all polarities of selected items match. The experimental results reported in Sections VI-B and VI-F show that this speedup typically comes for free, in the sense that the maximum divergence of the itemsets found is the same or very close, when polarity pruning is used or not.

Algorithm 1: Generalized divergence subgroup extraction.

Input: D, f, Γ, s
Output: Generalized itemsets divergence GI_Δ

```
GIs_divergence=[];
for step_i in Generalized Frequent Pattern Mining steps do
    I_{step_i} = extractGeneralizedItemsets(D, Γ, s, step_i);
    for I in I_{step_i} do
        I.s, I.Δ_f = evaluate_itemset(I, f(D))
        if I.s ≥ s then
            GIs_divergence.append(I);
return GIs_divergence
```

VI. Experimental results

We provide experimental results that, together, show the effectiveness of the hierarchical approach to discretization and anomalous group discovery implemented in H-DivExplorer. We first compare H-DivExplorer to standard non-hierarchical DivExplorer on a wide range of datasets, focusing the qualitative evaluations on the well known *compas* and *folktables* datasets. Next, we turn our attention to the *synthetic-peak* dataset, an artificial dataset in which we injected anomalous behavior in a controlled way. In Section VI-C we compare the hierarchical and non-hierarchical approaches over *synthetic-peak*, showing how the advantage of the hierarchical approach is due to its ability to spread the fixed "selectivity budget" (the narrowing down of subgroups, while staying over the support size) across attributes. In Section VI-E we study the sensitivity of the approach with respect to the parameter driving the discretization. In Section VI-F we offer some analysis of the performance of H-DivExplorer. Finally, in Section VI-G, we provide a detailed comparison of H-DivExplorer with prior approaches, including Slice Finder [8] and SliceLine [6].

In the following, we denote with *base* exploration the exploration performed without considering hierarchies, as implemented in Slice Finder, SliceLine and DivExplorer and with *base* itemsets the itemsets returned via such non-hierarchical exploration. We denote with *hierarchical* or *generalized* exploration the one performed by H-DivExplorer and with *generalized* the itemsets it derives. Throughout, we denote by s_t the support used in the tree construction for individual attribute discretization, and by s the (smaller) support used for subgroup discovery.

A. Datasets

We perform qualitative and quantitative experiments on two public datasets: *compas* and *folktables*, and on one artificial dataset, *synthetic-peak*, created by us and publicly available at [32]. For quantitative and performance experiments, we also use the *adult*, *bank* in its full version, *german*, online shoppers *intentions*, *wine* datasets [33]. Table II shows the main characteristics of all the adopted datasets, after standard preprocessing steps [33]. Furthermore, in the *bank* and *intentions* datasets, we consider the month as a numerical attribute to derive discretization hierarchies.

Compas. The *compas* dataset [14] 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. 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 (≥ 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 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.

Folktables. The *folktables* dataset [29] is based on US Census data. We use here the data for the census year 2018 and the state of California. We use the attributes of the income prediction task [34], and we consider the divergence of the income (the measure f we consider is directly the income). 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: 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 information available in the data. More specifically, each occupation category available in the dataset (e.g., *MGR-Financial Managers*, *MED-Dentists*) is mapped to its supercategory (e.g., *MGR*, *MED*).

Synthetic-peak. 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. We use this dataset to illustrate the effectiveness of item hierarchies in locating anomalous subgroups defined by multiple attributes (in this case, the three coordinates).

The task of the *adult* dataset is to predict whether the income exceeds \$50k per year or not based on census data. The *bank* dataset describes the subscription outcomes of a bank marketing campaign. We adopt the 'full' version from [33]. The *german* dataset describes loan applications where the task is to predict the individuals' credit risks (good or bad). The online shoppers purchasing *intentions* dataset [35] consists of online sessions and the task is to predict the user's intention to finalize the transaction. The *wine* dataset consists of physicochemical measures for red and white wines; we predict if the quality score is greater than 5 or not.

B. Benefits of Hierarchical Exploration

In our first series of experiments, we measure the benefit of item hierarchies in discovering highly divergent subgroups in the public datasets *compas* and *folktables*. We compare three different discretization methods:

- *Manual:* this is the discretization used in previous work on *compas* [5], [14].
- *Tree discretization, base:* we use our tree discretization process, and we retain only the leaf items. These leaf items can be analyzed via DIVEXPLORER, as no hierarchy is present.

dataset	$\|D\|$	$\|A\|$	$\|A\|_{num}$	$\|A\|_{cat}$
adult	45,222	11	4	7
bank (full)	45,211	15	7	8
compas	6,172	6	3	3
folktables	195,556	10	2	8
german	1,000	21	7	14
intentions	12,330	17	11	6
synthetic-peak	10,000	3	3	0
wine	9,796	11	11	0

TABLE II: Dataset characteristics. $|A|_{num}$ is the number of numerical attributes, $|A|_{cat}$ of categorical ones.

- *Tree discretization, hierarchical:* we use our tree discretization process to generate item hierarchies that are then explored via H-DIVEXPLORER.

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. Of course, combining items for different attributes, itemsets with support below 10% can be produced. For each subgroup discovery strategy, we measure the maximum divergence of the extracted itemsets.

Compas dataset. We focus on the divergence of the false-positive rate (FPR), which is the rate at which defendants are incorrectly predicted to recidivate. Table III reports the top FPR-divergent itemsets of the *compas* dataset according to the three discretization and exploration settings. The tree discretization uses support threshold $s_t = 0.1$, while the divergence subgroup exploration via DIVEXPLORER and H-DIVEXPLORER uses support thresholds $s = 0.05, 0.025$, and 0.01. We see that the tree-generated discretizations enable the identification of subgroups with higher divergence, compared to the manual discretization. This effect is due to the fact that the tree generation process is guided by divergence. Further, we see that the hierarchical approach leads to the identification of subgroups with higher divergence than those found via non-hierarchical techniques, for the same value of the exploration support. Consider for example the subgroup exploration with $s = 0.025$. The generalized itemset obtained via H-DIVEXPLORER is {*age=[25-32], stay≥3.0, #prior≥9*}, and has divergence 0.745. All three items derive from continuous attributes and are derived at different levels of the hierarchies. While *#prior≥9* is a leaf node, both the *stay* and *age* items are internal nodes, which represent generalizations of more refined items. This subgroup can't be identified by non-hierarchical subgroup identification techniques. Without hierarchical exploration, the highest-divergent subgroup is *age≤24, charge=F, #prior=[4-8]* and has divergence 0.662.

The hierarchical exploration is able to automatically adapt to the proper granularity at the *intersection* of multiple attributes. While for the age between 25 and 32 the prior which is frequently associated with and with high divergence is greater than 9, for younger defendants (*age≤*24), the number of priors is lower (equal to [4-8]), probably for indeed their younger age and a lower chance for having committed more prior offenses.

s	Exploration approach	Itemset	Sup	Δ_{FPR}	t
0.05	Manual discretization	age=[25-45], #prior>3, race=Afr-Am, sex=Male	0.13	0.220	7.1
	Tree discretization, base	#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	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	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 III: *compas* dataset: top divergent itemsets found by base DIVEXPLORER using Manual and Tree Discretization (leaf items only) and by the generalized exploration of H-DIVEXPLORER with $s_t = 0.1$; t is the statistical significance of divergence.

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 IV: *folktables* dataset: top divergent itemsets found by DIVEXPLORER and H-DIVEXPLORER using Tree Discretization both base (leaf items only) and hierarchical with $s_t = 0.1$; t is the statistical significance of divergence.

Folktables dataset. Top divergent itemsets are reported in Table IV. The manual discretization approach has been omitted because no common discretization is available in the literature. As for *compas*, we use support 0.1 for the trees used in attribute discretization. The exploration is conducted with support thresholds s equal to 0.05, 0.025 and 0.01. For all the support thresholds, the hierarchical exploration of divergence reveals the highest divergence.

Consider for example support threshold 0.05. The highest divergence is achieved by the generalized itemset {*AGEP≥35.0, OCCP=MGR, SEX=Male*}, corresponding to men with age greater than or equal to 35 and a managerial occupation. This itemset is characterized by an average income that is 90.2k greater than the average of the entire dataset. Both items *AGEP≥35.0* and *OCCP=MGR* are non-leaf items. Age leaf items correspond to finer age intervals. The item *OCCP=MGR* represents a managerial occupation, and in the tree is split into finer categories, e.g., *MGR-Sales Managers* and *MGR-Financial Managers*. Only by combining these non-leaf items in a hierarchical exploration, we are able to find an itemset including the *OCCP* attribute that is above the support threshold 0.05. This illustrates the ability of our approach in detecting divergent behaviors for sensitive attributes such as age, gender, and occupation by selecting the proper granularity level. This itemset, which is relevant for bias analysis, would not have been detected by base exploration approaches, due to the low support of its base items.

Quantitative analysis. Figure 2a summarizes the maximum divergence of the itemsets that the *base* exploration of DIVEXPLORER (in dashed lines) and our generalized exploration identify for the *adult*, *compas*, *german*, *intentions*, *synthetic-peak* and *wine* datasets. For *compas* we adopt the FPR as in the previous analysis. For *synthetic-peak* we use the already provided class label and we consider the error rate as described in Section VI-A. For the other datasets, we use

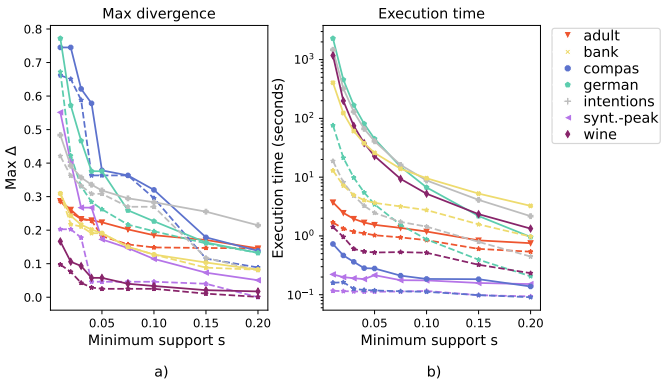
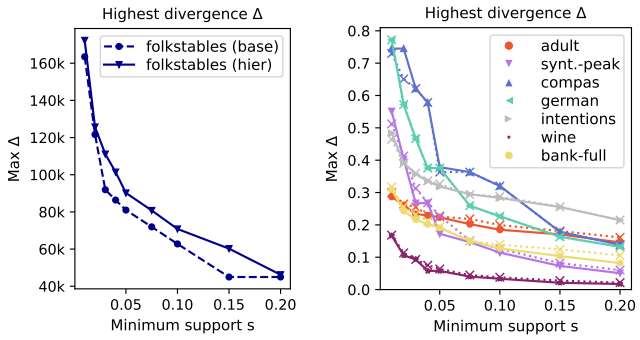


Fig. 2: (a) The highest divergence found, and the (b) execution time for the base exploration in *dashed* line, and our hierarchical subgroup exploration in *solid* line. We use $s_t = 0.1$ and the divergence gain criterion.

a random forest classifier with default parameters to provide the outcome function and we compute divergence for the error rate. Figure 3a shows the maximum income divergence for *folktables*. We report the result separately as the scale (income) is not a probability; hence, only the divergence criterion can be used for the discretization trees. In all cases, the hierarchical exploration enables the identification of subgroups with higher divergence, compared to when only leaf items are used.

Entropy-based vs. divergence-based tree node splitting. We compare the divergence results for both the entropy-based gain criterion and the divergence-based one. We consider all the analyzed datasets except *folktables* because the false-positive rate and the error rate are defined via a boolean outcome function and have the form of a probability. The results are reported in Figure 3b (solid line for the divergence criterion and dotted line for the entropy one). While the two criteria



(a) Divergence criterion (b) Divergence vs Entropy criteria

Fig. 3: (a) Highest identified divergence for *folktables* dataset, divergence criterion, base vs. hierarchical exploration. (b) Highest divergence found by hierarchical exploration with the divergence (solid line) vs. entropy (dotted line) criteria.

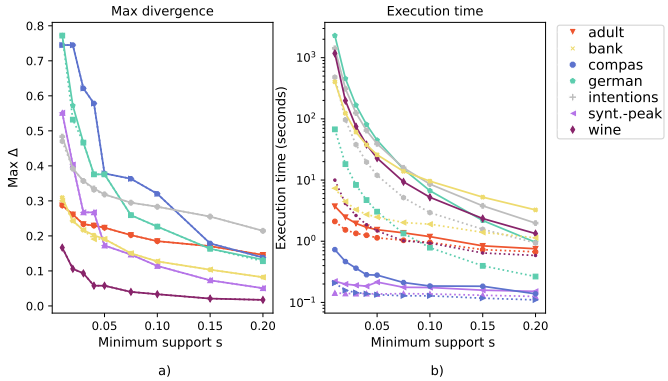


Fig. 4: (a) The highest divergence found and the (b) execution time for our hierarchical subgroup exploration with the complete (*solid* line) and pruned search (pr., *dotted* lines); $s_t = 0.1$ and divergence gain criterion.

in general do not yield the same discretization, they show similar effectiveness in identifying anomalous subgroups. The divergence criterion has the advantage of being applicable to a generic outcome function.

Effectiveness of polarity pruning. We evaluate the effectiveness of polarity pruning (its efficiency is evaluated in Section VI-F). Figure 4a compares the highest divergence found by the complete and the polarity-pruned search. Polarity pruning is very effective. While allowing a reduction of the search space, it does not impact divergence results. More specifically, it generally achieves the same results as the complete exploration. The highest divergence differs by a slight amount in only four cases of all explored datasets and support thresholds.

C. Identification of injected anomalies

The chief advantage of hierarchical discretization and exploration, compared to an exploration that relies only on leaf items, is to enable the "selectivity budget" to be spent on multiple attributes in the search for divergent subgroups. To illustrate this, we turn to our artificial dataset *synthetic-peak*, consisting of a rectangular domain filled with random points. The random points have a true class which is T, F with equal probability; the predicted class has an error rate that depends on the proximity to a point of maximum anomaly. Because the anomaly is centered around a point in three-dimensional space, the divergence in error rate is best detected when the "selectivity budget" can be spent homogeneously on the three coordinates.

As before, we derive the trees with respect to a support threshold of 0.1, which is fine enough to identify, for each attribute, intervals around the anomalous point. Then, we use DivExplorer to explore itemsets composed of leaf items only, and H-DivExplorer to explore generalized itemsets.

Figures 5a and 5c visualize the itemset with highest divergence for an exploration support threshold of $s = 0.05$. Each figure shows, for a given attribute, the projection of the normalized multivariate normal distribution used to generate the errors, and the ranges identified in the itemset. The asymmetry in the results — the reason why the attributes a, b, c are treated differently — is due to two factors. First, the anomaly center is not situated in the center of the rectangle. Second, the coordinates a, b, c of the data points in *synthetic-peak* are chosen uniformly at random, and the random density fluctuations drive the tree-generation algorithms used for discretization to treat a, b, c slightly differently.

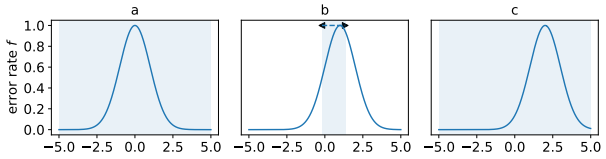
Figure 5a shows that, when base itemsets are used, the exploration returns $\{b = [-0.19, 1.34]\}$ with divergence $\Delta_{error} = 0.045$. This itemset is based on attribute b only; it is not possible to join it with items for a or c without going below the support threshold 0.05. In Figure 5c, we see instead that if we use item hierarchies, we can find a generalized itemset consisting of items for a, b, c with support over the 0.05 threshold, and divergence $\Delta_{error} = 0.229$. Hierarchical exploration enables the identification of subgroups that are over four times as divergent.

Figures 5b and 5d show the results for exploration support 0.025. For this smaller support, the base exploration finds an itemset constraining both b and c. Still, its divergence is inferior to the one of the subgroup found by hierarchical exploration.

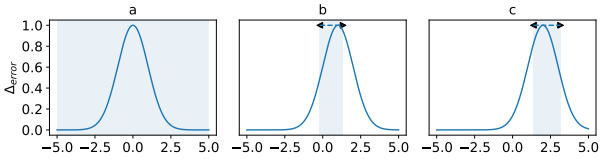
This example used a 3-dimensional dataset for ease of illustration. The performance difference between the hierarchical and base approaches would be larger in higher-dimensional datasets.

D. Comparison with unsupervised discretization

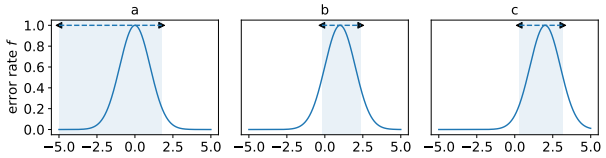
Again on our synthetic dataset *synthetic-peak*, we compare the quality of H-DivExplorer, with quantile discretization followed by base DivExplorer (that is, considering the leaf items generated with quantiles only). We note that in the *synthetic-peak* dataset, the quantile discretization corresponds roughly to the uniform one, as the values for the a, b, c attributes are generated uniformly at random. Contrary to



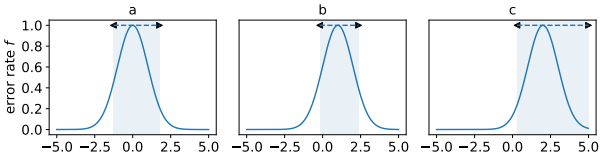
(a) Base itemset, $s = 0.05$, $\Delta_{error} = 0.045$



(b) Base itemset, $s = 0.025$, $\Delta_{error} = 0.212$

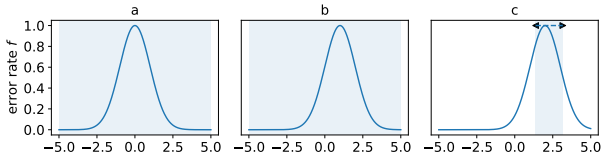


(c) Generalized itemset, $s = 0.05$, $\Delta_{error} = 0.229$

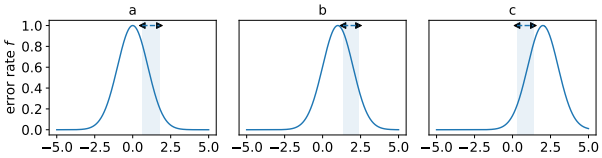


(d) Generalized itemset, $s = 0.025$, $\Delta_{error} = 0.297$

Fig. 5: Ranges of the itemset with highest divergence obtained via the *base* and our *generalized* one with $s = 0.05$ and $s = 0.025$. The shaded areas indicate the attribute ranges included in the itemsets.



(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* exploration with Slice Finder (see Section VI-G).

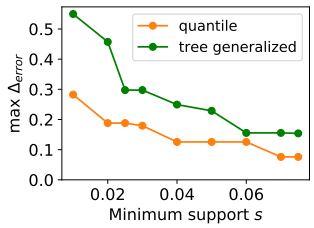
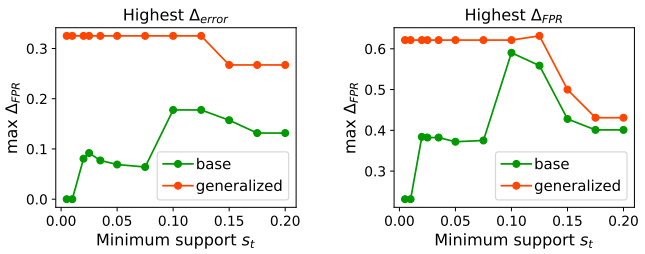


Fig. 7: Highest divergence found by H-DIVEXPLORER and the best result for the quantile discretization with base DIVEXPLORER for the *synthetic-peak* dataset.



(a) *synthetic-peak* (b) *compas*

Fig. 8: Highest divergence for base and generalized itemsets varying s_t for the *synthetic-peak* and *compas* datasets ($s = 0.025$).

our discretization, quantile discretizations are *unsupervised*, as they are not guided by the divergence of interest.

We vary the number of bins for the discretization from 2 to 10, and we extract the itemsets and their divergence via the base subgroup discovery of DIVEXPLORER. We consider the input bin size which achieved the highest divergence for each evaluated minimum support threshold and we report the results in Figure 7. H-DIVEXPLORER achieves the highest results for all the input thresholds: the generalized exploration, relying on hierarchical discretization, is able to adapt and identify the best item granularity.

E. Sensitivity analysis

The s_t parameter represents the minimum support of the tree nodes, which is used in the discretization process. We evaluate the impact of varying s_t on divergence. For these experiments, we set the divergence exploration minimum support s to 0.025.

Figures 8a and 8b plot the maximum divergence of base and generalized itemsets when varying s_t for the *compas* and *synthetic-peak* datasets. We do not report the results for the *folktables* dataset, because the highest divergence Δ_{income} is constant for the considered s_t range (Δ_{income} is 105.3k for the base and 119.3k for the hierarchical explorations).

In Figures 8a and 8b the maximum divergence returned by the hierarchical exploration is stable for a wide range of s_t values. Only when the minimum node support s_t becomes very high (i.e., nodes have to be supported by a large number of instances, for example more than 15% for the *synthetic-*

peak dataset), maximum divergence drops, as the items are too coarse. The base exploration is instead sensitive to the s_t parameter. More specifically, for $s_t < s$, leaf items generated via tree discretization may be characterized by a support lower than the s support threshold used for subgroup identification. Hence, since fewer items will be available for exploration, subgroup identification may return significantly lower divergence itemsets.

This experiment highlights the interplay between discretization and subgroup exploration. When discretization intervals are set a-priori, their (improper) definition may prevent the identification of interesting anomalous subgroups.

As a final remark, the hierarchical exploration, beyond being more stable when varying s_t, always identifies itemsets characterized by the highest divergence.

F. Performance analysis

All the experiments were performed on a PC with Ubuntu 22.04, 128 GB RAM, Intel Core i9.

The time required by the discretization process is always negligible compared to that required by the exploration. For *wine* and *intentions* dataset, characterized by the highest number of continuous attributes, the discretization process takes less than 1s and 7s with $s_t = 0.1$ respectively.

As expected (Figure 2b), hierarchical subgroup exploration takes more time than base exploration, because a larger number of items is generated and more itemsets have to be considered. However, while requiring more time, the hierarchical exploration allows the identification of higher divergence itemsets as shown in Figure 4.

Figure 4b shows the efficiency of the proposed polarity pruning. The average performance gain ranges from x1.4 for the *adult* dataset to x27.6 for the *wine* dataset, which is characterized by a larger number of continuous attributes, reaching a peak of x116.8 for low support ($s = 0.01$). Hence, polarity pruning allows a significant reduction of computation time, while identifying divergent itemsets as effectively as the complete search (see Figure 4).

G. Comparison with prior subgroup identification approaches

We compare H-DivExplorer with two other base exploration approaches: Slice Finder [8] and SliceLine [6].

All the approaches perform a lattice search, with different stopping criteria. Slice Finder performs a non-exhaustive search of the top-k slices which stops when a minimum effect size threshold is overcome. SliceLine and DivExplorer both leverage frequent pattern mining techniques and consider minimum support threshold and, for [6], monotonicity properties of errors for pruning. While H-DivExplorer also exploits the minimum support as stopping criterion, differently from all base exploration approaches, it explores a richer lattice that includes generalized items.

Since the three approaches measure the anomaly and explore data slices differently, it is difficult to compare their exploration results. Hence, we leverage our synthetic dataset *synthetic-peak* for which the problematic behavior is known and we evaluate the ability of Slice Finder and SliceLine[1] in identifying it. Data is discretized using leaf items as in the experiments reported in Section VI-C.

Figure 6a shows the itemset with the highest effect size identified by Slice Finder with default parameters. Since the slice $\{(c = [1.37, 3.16])\}$ is already problematic (with effect size 0.79, larger than the 0.4 input threshold) the search stops. Only if we increase the input threshold to 1, Slice Finder identifies a slice with the highest effect size composed by all the three terms (Figure 6b). However, this slice is not representative, because it includes only 13 instances (i.e., it has support 0.0013). Since Slice Finder does not control the slice support, it can fail to identify larger and more significant 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 ($s = 0.05$ and 0.025) of the experiments in Section VI-C. We vary the weight parameter α of SliceLine (importance of the average slice error) and 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.

These experimental results highlight the limitations of the base exploration performed by previous approaches. Fixed discretization hampers the identification of large and frequent slices associated with a problematic subgroup, while hierarchical exploration allows an adaptive identification of most problematic subgroups.

VII. Conclusions

This paper introduced hierarchical discretization and subgroup exploration. We evaluated the performance of our hierarchical approach H-DivExplorer on both real and synthetic datasets. Our results show that a hierarchical approach enables a flexible extraction of subgroups and is robust to the choice of discretization. We believe it holds promise in other areas related to the study of ML pipeline quality and fairness.

Acknowledgment

This study was carried out within (i) the project "National Centre for HPC, Big Data and Quantum Computing", CN000013 (approved under the M42C Call for Proposals - Investment 1.4 - Notice "Centri Nazionali" - D.D. No. 3138, 16.12.2021, admitted for funding by MUR Decree No. 1031,17.06.2022) and (ii) the FAIR - Future Artificial Intelligence Research and received funding from the European Union Next-GenerationEU (PIANO NAZIONALE DI RIPRESA E RESILIENZA (PNRR) – MISSIONE 4 COMPONENTE 2, INVESTIMENTO 1.3 – D.D. 1555 11/10/2022, PE00000013). This manuscript reflects only the authors' views and opinions, neither the European Union nor the European Commission can be considered responsible for them.

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

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] E. Pastor, A. Gavgavian, E. Baralis, and L. de Alfaro, "How divergent is your data?" *Proc. VLDB Endow.*, vol. 14, no. 12, p. 2835–2838, jul 2021. [Online]. Available: https://doi.org/10.14778/3476311.3476357

[8] Y. Chung, T. Kraska, N. Polyzotis, K. H. Tae, and S. E. Whang, "Slice finder: Automated data slicing for model validation," in *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, 2019, pp. 1550–1553.

[9] A. Asudeh, Z. Jin, and H. Jagadish, "Assessing and remedying coverage for a given dataset," in *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, 2019, pp. 554–565.

[10] N. Shahbazi, Y. Lin, A. Asudeh, and H. V. Jagadish, "A survey on techniques for identifying and resolving representation bias in data," 2022. [Online]. Available: https://arxiv.org/abs/2203.11852

[11] A. Asudeh, N. Shahbazi, Z. Jin, and H. V. Jagadish, "Identifying insufficient data coverage for ordinal continuous-valued attributes," in *Proceedings of the 2021 International Conference on Management of Data*, ser. SIGMOD '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 129–141. [Online]. Available: https://doi.org/10.1145/3448016.3457315

[12] 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.

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

[14] 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

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

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

[17] 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.

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

[19] 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.

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

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

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

[23] U. Fayyad and K. Irani, "Multi-interval discretization of continuous-valued attributes for classification learning," in *Proceedings of the 13th International Joint Conference on Artificial Intelligence.*, 1993, pp. 1022–1029.

[24] E. Pastor, L. de Alfaro, and E. Baralis, "Identifying biased subgroups in ranking and classification," 2021.

[25] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules in large databases," in *VLDB '94*, 1994, p. 487–499.

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

[27] J. Han, M. Kamber, and J. Pei, *Data mining concepts and techniques, third edition*. Waltham, Mass.: Morgan Kaufmann Publishers, 2012.

[28] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, ser. Springer Series in Statistics. New York, NY, USA: Springer New York Inc., 2001.

[29] 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.

[30] J. Han and Y. Fu, "Mining multiple-level association rules in large databases," *IEEE Transactions on Knowledge and Data Engineering*, vol. 11, no. 5, pp. 798–805, 1999.

[31] I. Pramudiono and M. Kitsuregawa, "Fp-tax: Tree structure based generalized association rule mining," in *Proceedings of the 9th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, ser. DMKD '04, 2004, p. 60–63.

[32] E. Pastor, E. Baralis, and L. de Alfaro, "Synthetic peak dataset," Oct 2022. [Online]. Available: https://github.com/elianap/h-divexplorer/blob/main/datasets/synthetic-peak-dataset.csv

[33] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[34] 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.

[35] C. O. Sakar, S. O. Polat, M. Katircioglu, and Y. Kastro, "Real-time prediction of online shoppers' purchasing intention using multilayer perceptron and lstm recurrent neural networks," *Neural Computing and Applications*, vol. 31, no. 10, pp. 6893–6908, 2019.