How Divergent Is Your Data?*

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

Andrew Gavgavian
UC Santa Cruz, USA
agavgavi@ucsc.edu

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

Luca de Alfaro
UC Santa Cruz, USA
dealfaro@acm.org

ABSTRACT

We present DivExplorer, a tool that enables users to explore datasets and find subgroups of data for which a classifier behaves in an anomalous manner. These subgroups, denoted as divergent subgroups, may exhibit, for example, higher-than-normal false positive or negative rates. DivExplorer can be used to analyze and debug classifiers. If the data has ethical or social implications, DivExplorer can be also used to identify bias in classifiers.

PVLDB Reference Format:
Eliana Pastor, Andrew Gavgavian, Elena Baralis, and Luca de Alfaro. How Divergent Is Your Data?[1]. PVLDB, 14(1): XXX-XXX, 2020.
doi:XX.XX/XXX.XX

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://divexplorer.github.io/.

1 INTRODUCTION

Machine learning model performance is commonly evaluated by means of well-known measures like accuracy, error rate, false positive and false negative rates. While these measures are typically estimated over all the data in a dataset, major differences in performance may occur across different data subgroups. Hence, the identification of these subgroups may be exploited for a variety of model evaluation tasks, ranging from model validation and error analysis, to model debugging. Furthermore, difference in performance may occur for sensitive attributes. In this case, machine learning models may inherit societal bias from unfair or underrepresented data.

The notion of *divergence* [8] allows us to estimate the difference in classification performance measures, such as false positive rate, on a subgroup with respect to the entire dataset. In this paper we introduce DivExplorer, an interactive visual analytics tool to automatically identify and inspect data subgroups in which a model performs differently. Divergence exploration may reveal data subgroups in which a model performs poorly, thus helping data scientists both in model quality evaluation, and in (unsupervised) model audit for algorithmic bias. Our approach is model agnostic. Hence, it treats the classification model as a black box, without requiring knowledge of, or access to, its internal working.

We measure divergence both for (a) data subgroups, and (b) individual (attribute, value) pairs. In the paper, we denote a single (attribute, value) pair as *item*, and a subgroup, i.e., a subset of the data characterized by a set of attributes values, as *pattern* or *itemset*. When analyzing divergence for a data subgroup, DivExplorer allows drilling down into the individual contribution of each item to the subgroup divergence, thus measuring its local divergence. The value of global item divergence, instead, is computed by considering the item contribution to all itemsets to which it belongs. This exploration also allows highlighting peculiar phenomena, the most interesting of which is the notion of *corrective item*, which is an item that *decreases* divergence when added to patterns.

Previous approaches to subgroup analysis include both supervised and unsupervised approaches. Several interactive visualization tools [1, 7] audit model performance over data subsets relying on user expertise to identify the subgroups of interest. Differently, DivExplorer leverages on an automatic subgroup detection to identify critical subgroups. Several interactive techniques that automatically identify interesting data subgroups have been recently proposed [3–6]. The visual analytics system FairVIS [4] exploits a clustering techniques to identify subgroups. The generated groups are then described by a few dominant features via feature entropy. Differently from FairVIS, DivExplorer slices the data by attribute domains. Hence, the characterizing features are known and easily interpretable. We further describe subgroups by quantifying the role of each attribute value to subgroup divergence using concepts of coalition game theory. The interactive system MithraCoverage [3, 6] focuses on investigating population bias in intersectional groups. The notion of uncovered pattern is introduced to identify intersectional subgroups with inadequate representation in the dataset. Differently, in our work, only subgroups with adequate representation are selected by a frequency threshold.

Slice Finder [5] automatically identifies data slices in which the model performs poorly. It uses a top-down approach to find the top-k "problematic" slices. However, since its data exploration is stopped by a heuristic criterion, it may miss some relevant problematic subgroups. Slice Finder provides interactive visualizations to inspect identified subgroups and interactively adjust input parameters to refine the exploration. Differently, DivExplorer exhaustively identifies all divergent subgroups that are sufficiently represented in the dataset, selected by a frequency threshold. Our tool allows users to interactively inspect subgroups via multiple views to fully characterize the divergent behavior, which include

[1]This is the author's draft of the paper, to appear in the proceedings of VLDB Demo Track, 2021.

Proceedings of the VLDB Endowment, Vol. 14, No. 1 ISSN 2150-8097.
doi:XX.XX/XXX.XX

the contribution of each attribute value to subgroup divergence and its visual exploration through its lattice graph. An in depth comparison between our approach and Slice Finder is provided in [8]. Finally, DivExplorer estimates the global contribution to divergence of each attribute value over the entire dataset.

2 THE DIVEXPLORER SYSTEM

DivExplorer [8] enables users to analyze the behavior of a classifier over a dataset, and to identify the portions of the dataset on which the classifier behaves in anomalous fashion, for example, by exhibiting particularly high false-positive or false-negative rates. If the underlying data has ethical implications, as in the running example we will consider, DivExplorer can be used to identify subgroups of data for which the behavior of the classifier is problematic, and to analyze the factors that contribute to the problematic performance.

2.1 Background

DivExplorer analyzes datasets in tabular form. Rows represent dataset *instances*, and columns are the *attributes*. DivExplorer assumes that continuous attributes are *discretized*. Hence, attributes that are continuous in nature, such as age, or income, need to be discretized into fixed ranges prior to the dataset being input to DivExplorer. However, the classifier itself can use the original, non-discretized data to build its model. Discretization is used only for data exploration, not for classification. We are considering, as future work, to integrate the discretization step directly in DivExplorer. Finally, DivExplorer requires users to identify two columns: the *true* class of each instance, and the *predicted* class of each instance as output by the classifier.

An *item* is a pair *attribute=value*, such as *age=25-45* (where 25-45 is the discretized range). An *itemset* is a set of items, such as {*age=25-45, race=Afr-Am*}. The items in an itemset always involve different attributes. The *support set* of an itemset consists in the instances that satisfy it, and the *support* is the fraction of the dataset that satisfies it. DivExplorer analyzes the behavior of the classifier on all itemsets whose support is above a specified *support threshold s*. Specifying a support threshold has two related purposes: it excludes from analysis itemsets with few instances, on which the analysis would be affected by statistical fluctuations, and it reduces the number of itemsets under consideration, leading to a tool output of manageable size.

2.2 Implementation

DivExplorer is implemented as a web app, and it can be deployed on any cloud that provides services for running containerized web services. Our hosting relies on Google Appengine. The back-end, which implements the data access layers and analysis algorithms, is written in Python, and relies on the py4web web framework [10]. The dataset analysis operations are implemented on top of the Pandas library for dataset processing [12], and the scikit-learn library for data mining [9]. The front-end is written using the vue.js Javascript framework, which enables dynamic visualizations and explorations of the dataset. Data is stored in a cloud SQL database. In particular, we use Google Cloud Mysql and Google Cloud Storage.

3 A TOUR OF DIVEXPLORER

The demo highlights DivExplorer [2] interactive features that allow users to dynamically explore the behavior of a classifier on an arbitrary dataset. We illustrate the flow of the demonstration by using as a concrete example the *COMPAS* dataset [2]. *COMPAS* contains demographic information and criminal history of defendants. For each defendant, the dataset provides a score (called the *COMPAS* score), produced by a classifier, and intended to reflect the likelyhood that the defendant will commit crimes in the future (recidivate); the dataset also contains ground truth information on which defendants actually recidivated. The attributes of the *COMPAS* dataset include defendant age, gender, and race; number of prior crimes, whether the crimes were felonies or misdemeanors, and the length of prior stays in jail, among others.

In the demo, we show how DivExplorer first provides global information, such as the list of itemsets where the classifier behavior most deviates from the average (see Section 3.1). Next, we show that users can drill into specific regions of data — specific itemsets, and explore the roles of individual attributes in affecting the classifier behavior, both considering specific itemsets (see Section 3.2) and on a global scale over the dataset (see Section 3.3).

Before analyzing a dataset with the DivExplorer system, the dataset is uploaded and the two attributes (columns) corresponding to the predicted class and the ground-truth class are identified. The only parameter to be configured is the support threshold. In our running example, we set it to 0.05. Thus, only itemsets supported by 5% or more of the dataset instances are considered. The algorithms at the core of DivExplorer enable the efficient extraction and analysis of very large collections of itemsets. Hence, much lower supports can be chosen; see [8] for a discussion.

3.1 Divergent Itemsets

DivExplorer analyzes the dataset and computes the *divergence* of itemsets with respect to error metrics such as false-positive, false-negative, and error rates. The *false-positive divergence* of an itemset is the difference between the false-positive rate as measured on the support set of the itemset, and as measured on the whole dataset. False-negative divergence and error divergence are similarly defined. DivExplorer computes the divergences for all itemsets above the specified support size, and displays them in a table that can be sorted according to any characteristics, as depicted in Figure 1. The t-value indicates the Welch statistical significance of the divergence.

In Figure 1, the itemsets are sorted with respect to false-positive rate divergence. The itemset that has greatest divergence is *{age=25-45, #prior>3, race=Afr-Am, sex=Male}*, with support 0.13 (or 13% of the dataset), and false-positive divergence 0.220. This indicates that male African-American defendants between 25 and 45 years of age, with more than 3 prior offenses, are wrongly predicated to recidivate at a rate 0.220 (or 22%) greater than the average defendant. Sorting the itemsets according to their false-negative rate divergence (FNR) would reveal that the most divergent itemset is *{age>45, stay<week, #prior=[1,3]}*. These defendants have a false negative rate 28% lower than the average defendant.

[2]Source code available at http://divexplorer.github.io, demo video at http://bit.ly/divexplorer-demo

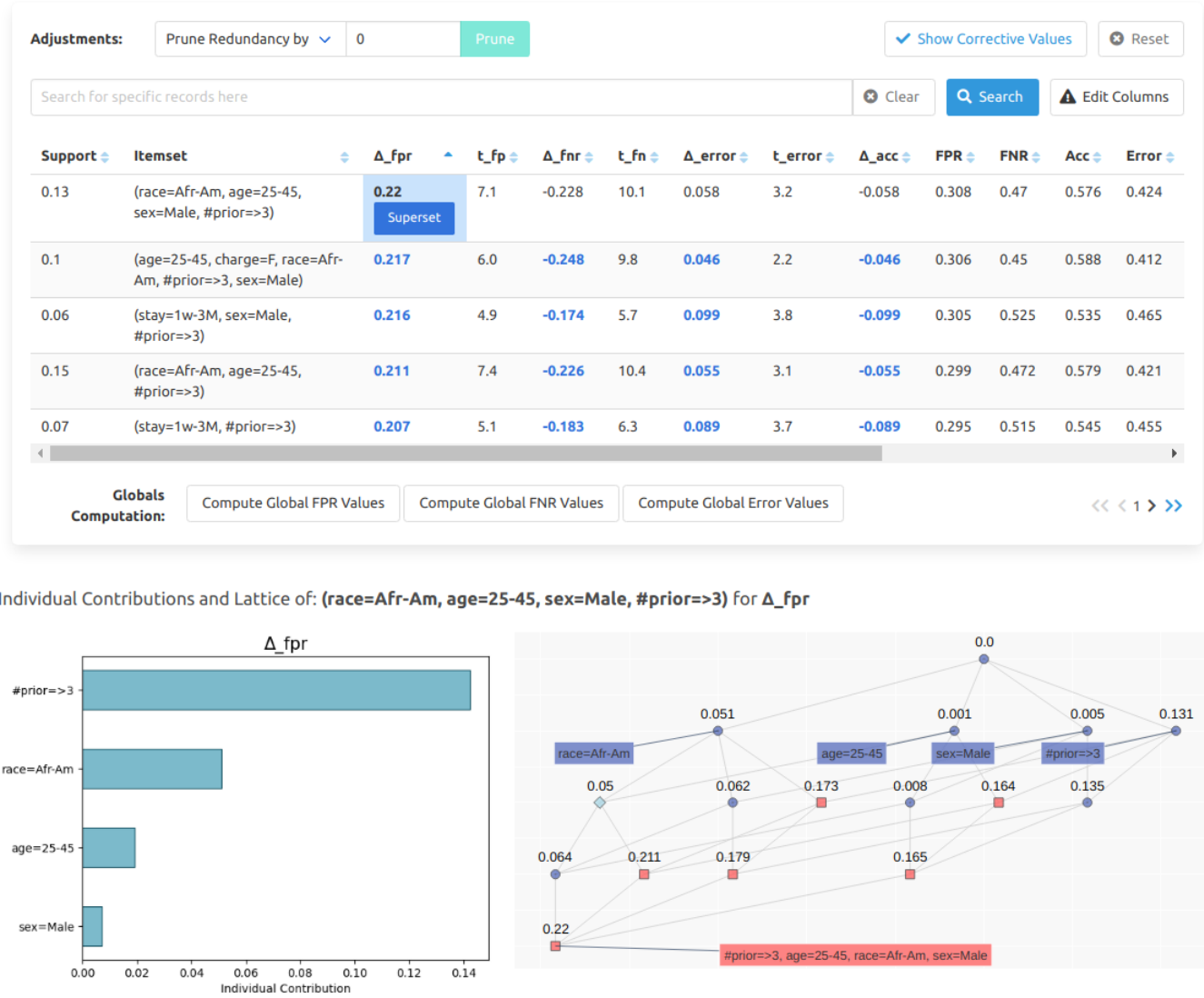

Figure 1: DivExplorer main UI.

Itemset	Sup	Δ_{FPR}	t_{FPR}
age=25-45, #prior>3, race=Afr-Am	0.15	0.211	7.4
stay=1w-3M, #prior>3	0.07	0.207	5.1
age<25, #prior=[1,3], race=Afr-Am	0.06	0.194	4.7
#prior>3, race=Afr-Am	0.20	0.173	7.6
age=25-45, stay<week,#prior>3, race=Afr-Am	0.10	0.171	5.4

Table 1: Top-5 divergent itemsets with the redundancy pruning for FPR. *COMPAS* dataset, ϵ=0.02, s=0.05.

Many of the most divergent itemsets in Figure 1 are quite similar. To obtain a more insightful summarization, DivExplorer enables users to prune (redundant) itemsets by specifying a threshold ϵ. If two itemsets I and $I \cup \{A\}$ have divergences closer than ϵ, only the smaller itemset I is included in the output. The top 5 itemsets for FPR-divergence, summarizing with $\epsilon = 0.02$, are shown in Table 1. The total number of patterns is reduced from 313 to 51.

3.2 Measuring the Role of Items in Itemset Divergence

On DivExplorer divergence table, the users can select the individual divergence values for detailed inspection. In Figure 1, the user has selected Δ_{FPR} for the top itemset *{age=25-45, #prior>3, race=Afr-Am, sex=Male}*. Thus, the details for the FPR-divergence for this itemset are displayed at the bottom.

On the bottom left is a bar graph indicating the extent to which the individual items contributed to the divergence of the itemset. The analysis is based on the notion of Shapley values [11]. The items are considered as players in a game, and the Shapley value is used

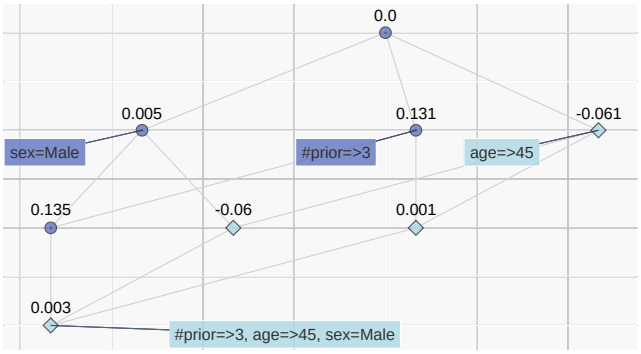

Figure 2: Lattice showing a corrective phenomenon for FNR divergence on the *COMPAS* dataset (rhombus nodes).

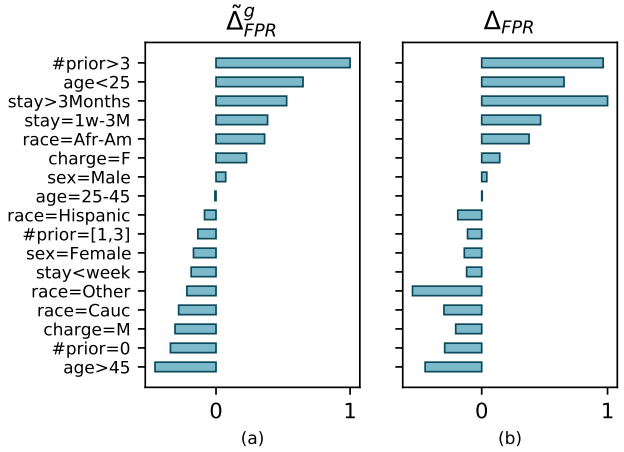

Figure 3: Relative magnitudes of global Shapley value and individual item divergence, for false-positive rate in the *COMPAS* dataset with s=0.05.

to attribute to the items the overall divergence of the itemset (we refer to [8] for the details). Most of the divergence of this itemset (most of the excess false-positive rate) is due to having more than 3 prior offenses, with *race=Afr-Am* as the second most important factor.

On the bottom right, similar information is conveyed via the lattice of subsets of the selected itemset. Each subset is labeled with its own divergence, and by hovering on it, the user can see which of the itemset's items appear in the subset. The lattice view is particularly useful when items are correlated. In this case, the Shapley value splits the contribution among correlated items. The itemset lattice shows the precise effect of each item in each context.

The lattice visualization is also useful for studying the role of *corrective items,* which are items that, when added to an itemset, *reduce* the divergence. For instance, Figure 2 shows that adding the *age>45* item to the *{#prior>3, sex=Male}* itemset *reduces* the false-negative divergence from 0.13 to 0.003. This indicates that the bias corresponding to *{#prior>3, sex=Male}*, upon closer inspection, affects only people below 45.

3.3 Item Influence on the Entire Dataset

The DivExplorer system also enables users to examine the influence of each item on the divergence of the entire dataset. This can be done in two different ways, both illustrated in Figure 3. The simplest measure, named the *individual divergence* of an item, is simply the divergence of an item in isolation. The *global divergence* of an item is a generalization of the Shapley value to the entire set of all items. We provide is [8] a precise mathematical definition. Intuitively, the global divergence summarizes how much an item increases the divergence when added to any itemset with which it is compatible (i.e., with which it does not share any attribute). The individual divergence is a simple and intuitive measure, but considers only items in isolation. The global divergence captures the role of an item in giving rise to divergence jointly with other attributes. From Figure 3, we see that the top item for global false-positive divergence, *#prior>3*, is part of the itemset that has overall greatest divergence. Interestingly, this is not so for the next two items, *age<25* and *stay>3Months*. To examine more closely their role, the user can use the search function in DivExplorer and examine the itemsets in which these items appear.

REFERENCES

[1] TensorFlow Model Analysis. 2018. Introducing TensorFlow Model Analysis: Scaleable, Sliced, and Full-Pass Metrics. https://medium.com/tensorflow/introducing-tensorflow-model-analysis-scaleable-sliced-and-full-pass-metrics-5cde7baf0b7b.

[2] Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. 2016. Machine Bias. https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing

[3] A. Asudeh, Z. Jin, and H. V. Jagadish. 2019. Assessing and Remedying Coverage for a Given Dataset. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. 554–565. https://doi.org/10.1109/ICDE.2019.00056

[4] Ángel Alexander Cabrera, Will Epperson, Fred Hohman, Minsuk Kahng, Jamie Morgenstern, and Duen Horng Chau. 2019. FairVis: Visual analytics for discovering intersectional bias in machine learning. In *2019 IEEE Conference on Visual Analytics Science and Technology (VAST)*. IEEE, 46–56.

[5] Y. Chung, T. Kraska, N. Polyzotis, K. Tae, and S. E. Whang. 2019. Automated Data Slicing for Model Validation: A Big data - AI Integration Approach. *IEEE Transactions on Knowledge and Data Engineering* (2019). https://doi.org/10.1109/TKDE.2019.2916074

[6] Zhongjun Jin, Mengjing Xu, Chenkai Sun, Abolfazl Asudeh, and H. V. Jagadish. 2020. MithraCoverage: A System for Investigating Population Bias for Intersectional Fairness. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data* (Portland, OR, USA) *(SIGMOD '20)*. Association for Computing Machinery, New York, NY, USA, 2721–2724. https://doi.org/10.1145/3318464.3384689

[7] Minsuk Kahng, Dezhi Fang, and Duen Horng (Polo) Chau. 2016. Visual Exploration of Machine Learning Results Using Data Cube Analysis. In *Proceedings of the Workshop on Human-In-the-Loop Data Analytics* (San Francisco, California) *(HILDA '16)*. New York, NY, USA, Article 1, 6 pages. https://doi.org/10.1145/2939502.2939503

[8] Eliana Pastor, Luca de Alfaro, and Elena Baralis. 2021. Looking for Trouble: Analyzing Classifier Behavior via Pattern Divergence. In *Proceedings of the 2021 International Conference on Management of Data (SIGMOD '21)* (to appear).

[9] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[10] Massimo Di Pierro. 2021. Py4Web. https://py4web.com

[11] Lloyd S Shapley. 1953. A value for n-person games. *Contributions to the Theory of Games* 2, 28 (1953), 307–317.

[12] Wes McKinney. 2010. Data Structures for Statistical Computing in Python. In *Proceedings of the 9th Python in Science Conference*, Stéfan van der Walt and Jarrod Millman (Eds.). 56 – 61. https://doi.org/10.25080/Majora-92bf1922-00a