https://github.com/poloclub/visual-auditor/tree/258c59d7052d4fe2a7b3fafc129a2184b4c25f8c/visual-auditor-package/notebook-widget
https://github.com/poloclub/visual-auditor/blob/258c59d7052d4fe2a7b3fafc129a2184b4c25f8c/visual-auditor-package/notebook-widget/visual_auditor/visual_auditor.py#L401
https://github.com/poloclub/visual-auditor/blob/258c59d7052d4fe2a7b3fafc129a2184b4c25f8c/visual-auditor-package/notebook-widget/visual_auditor/visual_auditor.py#L480

- Flit for Python packaging
- The visualization (HTML file/bundle) is embedded in the Jupyter notebook using IPython's display_html()

VISUAL AUDITOR: Interactive Visualization for Detection and Summarization of Model Biases

David Munechika[1] Zijie J. Wang[1] Jack Reidy[2] Josh Rubin[2] Krishna Gade[2]
Krishnaram Kenthapadi[2] Duen Horng Chau[1]

Demo: https://visual-auditor.surge.sh/

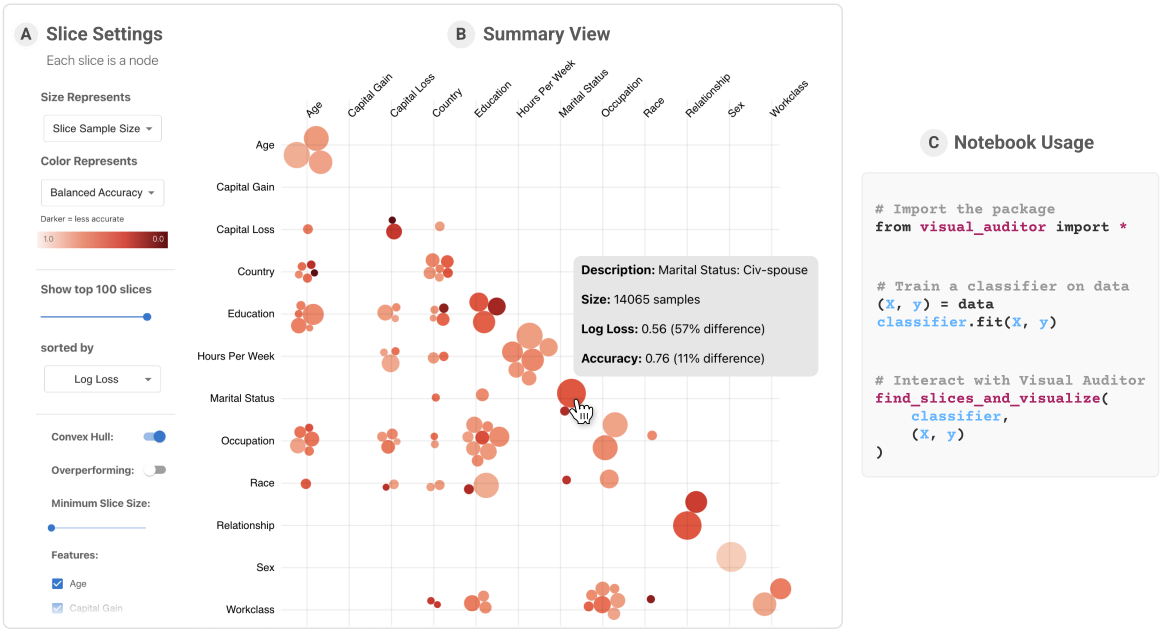

Figure 1: VISUAL AUDITOR provides an overview of underperforming data slices to show where intersectional biases exist. **(A)** The *Slice Settings* sidebar contains options for filtering the data and modifying the visualization, such as customizing the size and color encodings, showing the top *k* slices, or selecting particular features of interest. **(B)** The *Summary View* provides a visual overview of the underperforming data slices. Here currently displays the *Force Layout* which shows underperforming data slices as nodes on a grid. The location of each node is determined by the features that define the data slice. Users can view clusters of similar data slices to better understand where intersectional bias might exist in their model. **(C)** VISUAL AUDITOR is an open-source tool that easily integrates within existing data science workflows and can be accessed directly within computational notebooks.

ABSTRACT

Data subgroups or data slices or data subsets

As machine learning (ML) systems become increasingly widespread, it is necessary to audit these systems for biases prior to their deployment. Recent research has developed algorithms for effectively identifying intersectional bias in the form of interpretable, underperforming subsets (or slices) of the data. However, these solutions and their insights are limited without a tool for visually understanding and interacting with the results of these algorithms. We propose VISUAL AUDITOR, an interactive visualization tool for auditing and summarizing model biases. VISUAL AUDITOR assists model validation by providing an interpretable overview of intersectional bias (bias that is present when examining populations defined by multiple features), details about relationships between problematic data slices, and a comparison between underperforming and overperforming data slices in a model. Our open-source tool runs directly in both computational notebooks and web browsers, making model auditing accessible and easily integrated into current ML development workflows. An observational user study in collaboration with domain experts at Fiddler AI highlights that our tool can help ML practitioners identify and understand model biases.

Index Terms: Human-centered computing—Visualization

1 INTRODUCTION

The growing success and popularity of machine learning (ML) has led to widespread applications in the real world. It is therefore necessary to ensure that deployed systems exhibit fair treatment across all subgroups of people [5]. Without proper model auditing and validation, we risk encoding prejudicial biases into our models, thereby deploying systems in the real world that reflect our social biases and result in user discrimination [5, 25].

An example can be found at the forefront of the ML fairness debate, specifically surrounding the algorithms behind recidivism prediction instruments (RPIs). Studies have found the predictive accuracy of certain RPIs, such as the COMPAS [26] or PCRA [27] instruments, vary significantly between different demographic groups, serving as evidence of algorithmic bias [12, 17, 35]. Similar concerns about perpetuating discrimination through algorithmic unfairness have been raised in other areas including credit card scoring, housing advertisements, and mortgage systems [3]. Often, a high overall model performance can cover up the unsatisfactory performance of individual subgroups (or slices) of data [10, 30, 33].

To mitigate biases, recent researchers often identify slices of data

[1]Georgia Tech. {dmunechika3|jayw|polo}@gatech.edu
[2]Fiddler AI. {jack|josh|krishna|krishnaram}@fiddler.ai

that appear particularly problematic—these are characterized by the ML model exhibiting unusually poor performance on the slice compared to the model's overall performance and the slice being large enough to be significant rather than simply an outlier [30, 33]. While existing algorithms [30, 33] are capable of identifying problematic slices, they fail to provide effective [34], interpretable [24] overviews of the model bias. Furthermore, existing tools [7, 8, 37] that aim to illustrate model performance and examine discrimination in ML models require users to have prior knowledge about which biases exist within the model.

To address these limitations, we design and develop VISUAL AUDITOR, an interactive, visual interface for model bias detection and summarization. We aim to create a visual overview that enables ML models to be audited so that underperforming subgroups can be effectively surfaced and analyzed by human users. Our research makes the following major contributions:

- **VISUAL AUDITOR, an interactive visualization tool** for auditing and summarizing ML model biases. VISUAL AUDITOR fills a critical research gap and practical need for visually summarizing and auditing underperforming data slices. Underperforming subgroups are identified using slice-finding algorithms [30] and are visually displayed along with important accompanying information including the size of the subgroup, how significantly it is underperforming, and what features it is defined by. VISUAL AUDITOR also complements existing research that focuses primarily on algorithmic slice finding [30, 33]. It goes beyond simply providing details about each data slice by identifying relationships between similar problematic data slices and providing methods for filtering these results to match the interest of the user.
- **Design lessons distilled from a user study** with Fiddler AI engineers and data scientists. An observational user study with 4 domain experts highlights how VISUAL AUDITOR may be useful when integrated within existing ML and data science workflows. We discuss design lessons from our iterative design process and the user study results.
- **An open-source**[1] **and web-based implementation** to empower ML practitioners to audit their models for bias. We developed VISUAL AUDITOR with modern web technologies so that anyone can access our tool directly in a web browser or computational notebook. A demo video of VISUAL AUDITOR can be viewed at https://youtu.be/ZGCVtu2fcbc.

It is our goal that VISUAL AUDITOR will encourage and improve the process of model validation while better enabling the analysis and eventual mitigation of intersectional bias in models.

2 BACKGROUND & RELATED WORK

Slice-Finding Algorithms. Auditing models trained on data defined by numerous features is a non-trivial task. One must not only look at subgroups defined by each particular feature but also consider the intersectional bias which may result when defining subgroups by multiple features. The number of potential subgroups of data grows combinatorially, rendering it impractical for a human to perform this type of model validation manually [33].

Existing research aims to address these computational issues of data slicing for model validation. SliceFinder is one existing framework for identifying problematic data slices. It uses statistical techniques to find interpretable slices as opposed to arbitrary subsets which are commonly the result of traditional clustering techniques. SliceFinder considers how significant the difference in loss is between the slice and the model itself as well as how large the slice is. It computes an effect size for each data slice which determines how significant (in other words, problematic) a particular data slice is for model validation—a higher effect size indicates higher significance. This effect size metric is used to rank data slices and determine which are the most problematic [30].

Similarly, SliceLine is an enumeration algorithm which attempts to extend the SliceFinder algorithm by addressing the scalability limitations of traditional methods. It leverages efficient sparse linear algebra to enable slice enumeration even in complex datasets [33].

Visualization for Model Performance and Bias. The objectives of ML fairness and understanding model performance has inspired a wealth of literature [19, 28]. Proposed systems have emerged with the intention of elucidating the behavior and interior of models and improving the fairness of AI systems using various visualization techniques. Uber's Manifold [40] and ModelTracker [4] are two systems that have been used in practice to provide performance insights and comparisons for ML models. However, these tools are solely focused on performance analysis without a specific aim on identifying biases or mitigating algorithmic unfairness.

Other recent work has emerged within the visualization community to specifically address model fairness. Audit AI [1], Google's What-If tool [37], Microsoft's Fairlearn project [8], and IBM's AI Fairness 360 toolkit [7] are all different solutions aiming to mitigate fairness issues within AI and ML models. They provide details about fairness metrics of different subgroups of data and use various statistical methods for comparing different groups. However, these systems are limited by requiring a priori knowledge of the discriminated groups. Users need to manually identify the protected attributes as well as the privileged and unprivileged groups in order for these systems to generate comparisons and evaluate potential biases. It is infeasible for users to manually consider all potential subgroups of data in order to identify biases [33]—therefore, without combining bias detection with these visualization methods, these tools are not easily integrated into current ML workflows.

The SliceFinder GUI attempts to solve this issue by combining a user interface with the SliceFinder algorithm [13]. However, this visual system only presents the textual outputs of the algorithm in a table without offering additional details about the data slices. It lacks capabilities to analyze problematic slices or understand model bias at a higher level. In contrast, our tool synthesizes bias detection with interpretable visualizations to address the limitations of these existing systems. It effectively summarizes data slices, allows filtering by desirable characteristics, displays fairness metrics, illustrates related overlapping slices as well as clusters of similarly-defined slices, and is easily integrated within current workflow environments.

3 DESIGN GOALS

Through close collaboration with engineers and scientists at Fiddler AI since August 2021, we have learned about the need for an interactive visual tool that helps ML practitioners summarize and analyze model bias. Through a literature review, we have identified four design goals of VISUAL AUDITOR.

G1. Visual summary of problematic data slices. Depending on the hyperparameters of the slice-finding algorithm (such as the degree, effect size threshold, or maximum number of slices), the number of problematic slices found by slice-finding algorithms can vary from only a few to over a hundred [30]. Without effective visual techniques, it is challenging for users to understand and explore existing bias in their model [31]. Therefore, we designed scalable visualizations to summarize the problematic data slices to help users better understand where their model is underperforming (Sect. 4.1).

G2. Ability to filter slices by desirable characteristics. To effectively audit their models, users need the ability to narrow their focus down to problematic slices that are of particular interest to them [31]. These might be slices with important and potentially sensitive features [15], slices of a particular size (to eliminate outliers), or slices characterized by an unusually high or low associated fairness metric [32] (Sect. 4.2).

[1] Code: https://github.com/poloclub/visual-auditor

G3. Comparing slices and analyzing slice relationships. Examining similar data slices can be useful to understand feature importance or identify larger, generalized slices (as the result of merging similar slices) [10, 16, 39]. This would be useful for identifying efficient bias mitigation techniques that target clusters of similar problematic slices in order to yield the greatest improvement in model performance (Sect. 4.3). The inclusion of overperforming slices for comparison can also be useful, especially in inbalanced datasets [11]. Comparing the differences in characteristics between underperforming and overperforming slices can yield useful insights regarding the performance of a particular model in order to identify strategies for mitigating existing model bias [6, 18, 23] (Sect. 4.4).

G4. Integrating into common development workflows. Modern data science workflows rely on the use of computational notebooks such as Jupyter Notebook [21]. Data scientists and ML practitioners frequently work within computational notebooks to develop and train ML models [29]. To ensure model auditing is accessible within the current workflow, we designed VISUAL AUDITOR to run in a web browser and as an interactive widget in computational notebooks. Finally, we open-sourced our tool to encourage and support future design, research, and development of model auditing and bias mitigation (Sect. 4.5).

4 SYSTEM DESIGN

Following the design goals, VISUAL AUDITOR (Fig. 1) tightly integrates four components: the *Force Layout* (Sect. 4.1), the *Slice Settings* panel (Sect. 4.2), the *Graph Layout* (Sect. 4.3), and the *Overperforming Slices* (Sect. 4.4).

4.1 Summarizing the Problematic Slices

VISUAL AUDITOR uses the SliceFinder [13] algorithm in the backend for identifying problematic data slices but extends its interpretability and actionability. We limit intersectional biases to be defined by at most two features to ensure interpretable slices with a significant effect size are identified. To help users efficiently understand and analyze the potentially large number of problematic slices produced by the algorithm, we present these in a summary view.

or Summary View (there is a dropdown menu to choose the layout)

Force Layout. The *Force Layout* (Figure 1B) summarizes the problematic slices in a dataset (G1). Each slice is displayed as a node on a grid and mapped to an area defined by the intersection of its features. Slices will be spatially located next to other slices defined by similar features. By default, the color of each node maps to the percent difference of its log loss compared to the overall model (a darker color indicates worse log loss and more severe underperformance). Similarly, the size of each node represents the sample size of the slice standardized on a log scale. However, both the color and size encodings can be customized based on user preference. Users can also hover over a specific node to display a tooltip containing details about that particular slice. Overall, the *Force Layout* is an effective visualization design because it immediately draws attention to the largest and most problematic slices through color and size encodings (G1) while simultaneously conveying information about relationships through clusters of similarly defined slices (G3).

4.2 Slice Filters and Visualization Settings

The *Force Layout* by default displays an overview of all of the slices. To enable users to focus on particular slices of interest (G2), the *Slice Settings* sidebar (Figure 1A) offers a collection of options for modifying the visualization and filtering the visible slices.

The *Color Represents* and *Size Represents* dropdown menus allow the user to customize each of these encodings. For example, the respective default encodings of log loss and sample size could be changed to a fairness metric such as balanced accuracy, if desired. Additionally, the *Show Top k Slices* slider and *Sorted By* dropdown allow users to specify a dimension (e.g. sample size, log loss, balanced accuracy) to sort the slices by and filter down to only the top *k* slices for that particular dimension.

The *Minimum Slice Size* slider sets a threshold for the smallest allowed slice sample size which is useful to filter out outlier slices that are composed of only a few samples and not necessarily representative of an existing bias. Similarly, the *Features* checkboxes enable users to specify features of interest and only display slices that include these features in their definitions. This is particularly helpful when dealing with model fairness because usually some attributes will be considered "protected" or more sensitive than others [38].

These various filtering options allow users to quickly identify the most problematic slices within their data and slices that are of particular importance to them. This effectively speeds up the process of identifying the most significant slices hurting model performance and understanding how to mitigate the existing bias.

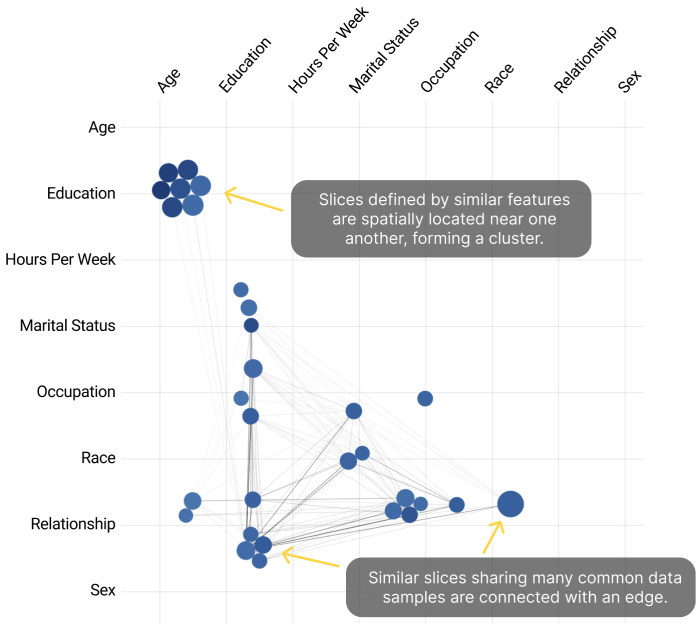

Figure 2: Users can also view overperforming slices to compare and learn from. This figure shows the *Graph Layout* which connects two slices with an edge if they share many data instances. Thicker edges indicate stronger relationships between slices and a darker node color indicates higher performance compared to the overall model.

4.3 Similar Slice Relationships

Graph Layout. While the *Force Layout* displays slices with similar features in distinct clusters on a grid, it does not allow the user to identify similar slices and relationships between slices. We define similar slices to be those that share a high number of data instances (i.e. overlapping samples) [10]. To display the relationships between slices (G3), we design the *Graph Layout* (Figure 2) which is an extension of the *Force Layout* that includes edges connecting nodes which represent similar data slices.

The number of overlapping samples between two slices determines the thickness of that particular edge, and the strength of the force pulling the nodes together. This results in nodes that share many common data samples being clustered closer together. The *Edge Force Strength* slider can be used to control the strength of the force of attraction between two connected nodes (with an edge force strength of zero degenerating into the *Force Layout*). Furthermore, the *Edge Filtering* slider can set the minimum number of overlapping samples necessary for an edge to exist between two nodes.

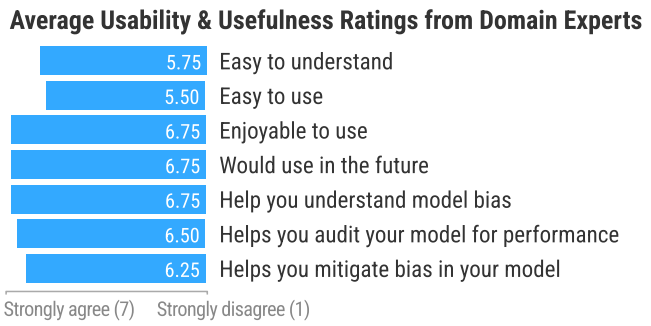

Figure 3: In our user evaluation, participants thought VISUAL AUDITOR was easy and enjoyable to use. The domain experts rated the usefulness of the tool very favorably, providing support that our interactive tool is effective at summarizing and analyzing model bias.

4.4 Overperforming Slices

Viewing which intersections of features yield the highest accuracies presents users with an additional insight into the performance of their model. To support the analysis of overperforming data slices (G3), we automatically compute these slices as part of the algorithm when the problematic slices are found. Users can switch to viewing overperforming slices by toggling on the *Overperforming* switch in the *Slice Settings* sidebar when viewing any visualization (Figure 2).

4.5 Accessible, Open-source Implementation

VISUAL AUDITOR is a web-based, interactive visualization tool built with D3.js [9] and React.js [2]. Users can access the tool using a web browser or directly within a computational notebook (G4). To increase the accessibility of our tool, we have released VISUAL AUDITOR on the Python Package Index (PyPI) [2]. Computing slices and generating the visual interface can be done in only a few lines of code (Figure 1C). We also open-sourced our implementation so future researchers can extend the tool's design and functionality.

and MUI

5 USER STUDY

6 participants in total from Fiddler AI

The VISUAL AUDITOR project started in August 2021. Internal design iterations with two data scientists from Fiddler AI were performed to gather feedback and further refine the tool. Finally, a user study was conducted with 4 additional data scientists and engineers (P1-P4) to investigate the effectiveness and usability of VISUAL AUDITOR. All participants were domain experts who indicated the highest level of familiarity with ML and data science. Participants received no compensation for taking part in the study.

Data. Participants were given the option of working with either the Adult Census Income dataset [22], which predicts whether or not an individual's income exceeds $50k per year, or the Statlog (German Credit) dataset [14], which classifies people as good or bad credit risks according to a set of attributes. Both of these datasets are popularly used in research related to algorithmic fairness [7, 20, 36].

Procedure. The study duration was one hour per participant. In each study, participants was invited to an individual Zoom meeting. First, they were asked to fill out a pre-study background survey and sign a consent form. They were given a brief tutorial that explained the purpose of VISUAL AUDITOR and demonstrated its capabilities. Following the tutorial, the participants selected a dataset to use and were provided with the relevant feature names and textual descriptions. They were also given the link to a Jupyter Notebook hosted on Binder[3] for testing VISUAL AUDITOR with their particular dataset. Within this simulated data science workflow, participants were asked to explore VISUAL AUDITOR and analyze problematic slices in their model while thinking aloud. The exploration of the tool was screen-recorded over Zoom. During this meeting they were informed that they could ask design- or functionality-related questions to the researchers. Each session ended with a usability questionnaire and post-study exit survey to finalize their evaluation.

Results. Average ratings for VISUAL AUDITOR's usability and usefulness are shown in Figure 3. All participants found the tool to be easy to understand and enjoyable to use. The domain experts also agreed that this tool provides new functionality for understanding model bias, auditing models for performance, and finding approaches to mitigate ML bias that did not previously exist.

Through the user study, we learned that the ***Force Layout* was an effective visual design for summarizing model bias**. P2 commented, *"the key feature I found most helpful was being able to visualize sample size and performance of all the slices at once. My eye was immediately drawn to underperforming slices with a large sample size."* This layout was also interpretable and understandable for participants as they found the visual representation of slices to be inherently intuitive. P3 noted that VISUAL AUDITOR provided a *"great visual representation of problematic slices [making] it easier to debug the model. The tool did a great job of representing sizes and relative performances of the slices using colors, so I was able to quickly understand and use the tool."*

Participants appreciated the sample size scaling and slice filtering options. Multiple participants identified the ability to filter the top k slices and set a minimum slice size as one of the most useful features. P1 said these filters helped them *"hone in on the most problematic slices"*. Other participants found the *Graph Layout* to be effective at identifying relationships between slices. P1 was particularly interested in examining the groups of densely connected nodes in the *"Graph Layout to target a bigger subsection."* Their strategy was to identify generalized data slices connecting multiple problematic slices so that targeting these slices would result in the greatest performance improvement in the model.

Lastly, the **notebook support and ease of integration into current workflows was viewed very favorably by participants**. From an initial background survey, all participants rated their notebook usage at either the highest level ("every day") or next highest level ("very frequently"). Every participant expressed an interest in using VISUAL AUDITOR in the future, and one participant (P4) with expertise specifically in mitigating intersectional fairness issues said they would *"absolutely use this tool in [their] visualization work."*

6 CONCLUSION & DISCUSSION

Mitigating bias and maximizing performance should be important considerations during the development of ML models. VISUAL AUDITOR addresses these issues by providing an overview of problematic data slices through the use of interactive visualization techniques. Our user study has shown that VISUAL AUDITOR is effective at summarizing and analyzing model bias and can be easily integrated into existing development workflows. For future work, we plan to extend VISUAL AUDITOR's capabilities to offer more actionable methods for mitigating existing bias. Intersectional bias can exist in a model for various reasons, including insufficient training data, explicitly or implicitly encoded social biases, or an overly-simple model architecture. A tool that identifies which solution would be most effective for addressing model bias would make the model validation process more efficient. We also plan to enhance VISUAL AUDITOR by improving graph view readability and allowing for the summarization of intersectional bias generated from more than two features. By interactively exploring the intersectional bias that exists in a model and providing effective bias mitigation strategies, VISUAL AUDITOR will help to ensure models deployed in the real world exhibit fair treatment across all subgroups of people.

[2]PyPI Package: https://pypi.org/project/visual_auditor/

[3]User Study Binder: https://mybinder.org/v2/gh/davidmunechika/visual-auditor-demo-repo/master

hone in on: to find and go directly toward (someone or something)

https://www.merriam-webster.com/dictionary/hone%20in%20on

REFERENCES

[1] Audit ai. https://github.com/pymetrics/audit-ai. Accessed: 2022-04-25.

[2] React.js. https://github.com/facebook/react. Accessed: 2022-04-25.

[3] J. A. Allen. The color of algorithms: An analysis and proposed research agenda for deterring algorithmic redlining. *Fordham Urb. LJ*, 46:219, 2019.

[4] S. Amershi, M. Chickering, S. M. Drucker, B. Lee, P. Simard, and J. Suh. Modeltracker: Redesigning performance analysis tools for machine learning. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pp. 337–346, 2015.

[5] S. Barocas and A. D. Selbst. Big data's disparate impact. *Calif. L. Rev.*, 104:671, 2016.

[6] G. E. A. P. A. Batista, R. C. Prati, and M. C. Monard. A study of the behavior of several methods for balancing machine learning training data. *SIGKDD Explor. Newsl.*, 6(1):20–29, jun 2004. doi: 10.1145/1007730.1007735

[7] R. K. Bellamy, K. Dey, M. Hind, S. C. Hoffman, S. Houde, K. Kannan, P. Lohia, J. Martino, S. Mehta, A. Mojsilovic, et al. Ai fairness 360: An extensible toolkit for detecting, understanding, and mitigating unwanted algorithmic bias. *arXiv preprint arXiv:1810.01943*, 2018.

[8] S. Bird, M. Dudík, R. Edgar, B. Horn, R. Lutz, V. Milan, M. Sameki, H. Wallach, and K. Walker. Fairlearn: A toolkit for assessing and improving fairness in ai. *Microsoft, Tech. Rep. MSR-TR-2020-32*, 2020.

[9] M. Bostock, V. Ogievetsky, and J. Heer. D^3 data-driven documents. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2301–2309, 2011. doi: 10.1109/TVCG.2011.185

[10] 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. *CoRR*, abs/1904.05419, 2019.

[11] N. V. Chawla. Data mining for imbalanced datasets: An overview. *Data mining and knowledge discovery handbook*, pp. 875–886, 2009.

[12] A. Chouldechova. Fair prediction with disparate impact: A study of bias in recidivism prediction instruments, 2016.

[13] 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)*, pp. 1550–1553. IEEE, 2019.

[14] D. Dheeru and E. Karra Taniskidou. Uci machine learning repository. https://archive.ics.uci.edu/ml/datasets/Statlog+%28German+Credit+Data%29, 2017. Accessed: 2022-04-25.

[15] J. Dong and C. Rudin. Exploring the cloud of variable importance for the set of all good models. *Nature Machine Intelligence*, 2(12), 2020.

[16] C. Dwork, M. Hardt, T. Pitassi, O. Reingold, and R. Zemel. Fairness through awareness, 2011. doi: 10.48550/ARXIV.1104.3913

[17] A. W. Flores, K. A. Bechtel, and C. T. Lowenkamp. False positives, false negatives, and false analyses: A rejoinder to "machine bias: There's software used across the country to predict future criminals. and it's biased against blacks". *Federal Probation*, 80:38, 2016.

[18] H. He and E. A. Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284, 2009. doi: 10.1109/TKDE.2008.239

[19] F. Hohman, M. Kahng, R. Pienta, and D. H. Chau. Visual analytics in deep learning: An interrogative survey for the next frontiers. *IEEE transactions on visualization and computer graphics*, 25(8):2674–2693, 2018.

[20] M. Kearns, S. Neel, A. Roth, and Z. S. Wu. An empirical study of rich subgroup fairness for machine learning. In *Proceedings of the conference on fairness, accountability, and transparency*, pp. 100–109, 2019.

[21] T. Kluyver and others. Jupyter Notebooks - a publishing format for reproducible computational workflows. *ELPUB*, 2016.

[22] R. Kohavi et al. Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid. In *Kdd*, vol. 96, pp. 202–207, 1996.

[23] M. Kubat, S. Matwin, et al. Addressing the curse of imbalanced training sets: one-sided selection. In *Icml*, vol. 97, p. 179. Citeseer, 1997.

[24] M. S. A. Lee and J. Singh. The landscape and gaps in open source fairness toolkits. In *Proceedings of the 2021 CHI conference on human factors in computing systems*, pp. 1–13, 2021.

[25] N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, and A. Galstyan. A survey on bias and fairness in machine learning. *ACM Computing Surveys (CSUR)*, 54(6):1–35, 2021.

[26] Northpoint. Compas risk & need assessment system: Selected questions posed by inquiring agencies., 2012.

[27] A. O. of the United States Courts Probation and P. S. Office. An overview of the federal post conviction risk assessment, 2018.

[28] K. Patel, J. Fogarty, J. A. Landay, and B. Harrison. Investigating statistical machine learning as a tool for software development. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 667–676, 2008.

[29] J. M. Perkel. Reactive, reproducible, collaborative: Computational notebooks evolve. *Nature*, 593(7857), May 2021.

[30] N. Polyzotis, S. Whang, T. K. Kraska, and Y. Chung. Slice finder: Automated data slicing for model validation. In *Proceedings of the IEEE Int' Conf. on Data Engineering (ICDE), 2019*, 2019.

[31] C. Rudin, C. Chen, Z. Chen, H. Huang, L. Semenova, and C. Zhong. Interpretable machine learning: Fundamental principles and 10 grand challenges. *Statistics Surveys*, 16, Jan. 2022.

[32] K. K. S. The impossibility theorem of machine fairness - A causal perspective. *CoRR*, abs/2007.06024, 2020.

[33] S. Sagadeeva and M. Boehm. *SliceLine: Fast, Linear-Algebra-Based Slice Finding for ML Model Debugging*, p. 2290–2299. Association for Computing Machinery, New York, NY, USA, 2021.

[34] D. Saha, C. Schumann, D. Mcelfresh, J. Dickerson, M. Mazurek, and M. Tschantz. Measuring non-expert comprehension of machine learning fairness metrics. In *International Conference on Machine Learning*, pp. 8377–8387. PMLR, 2020.

[35] J. L. Skeem and C. T. Lowenkamp. Risk, race, and recidivism: Predictive bias and disparate impact. *Criminology*, 54(4):680–712, 2016.

[36] S. Verma and J. Rubin. Fairness definitions explained. In *2018 ieee/acm international workshop on software fairness (fairware)*, pp. 1–7. IEEE, 2018.

[37] J. Wexler, M. Pushkarna, T. Bolukbasi, M. Wattenberg, F. Viégas, and J. Wilson. The what-if tool: Interactive probing of machine learning models. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):56–65, 2020. doi: 10.1109/TVCG.2019.2934619

[38] Z. Yu. Fair balance: Mitigating machine learning bias against multiple protected attributes with data balancing. *CoRR*, abs/2107.08310, 2021.

[39] R. Zemel, Y. Wu, K. Swersky, T. Pitassi, and C. Dwork. Learning fair representations. In *International conference on machine learning*, pp. 325–333. PMLR, 2013.

[40] J. Zhang, Y. Wang, P. Molino, L. Li, and D. S. Ebert. Manifold: A model-agnostic framework for interpretation and diagnosis of machine learning models. *IEEE transactions on visualization and computer graphics*, 25(1):364–373, 2018.