FAIRVIS: Visual Analytics for Discovering Intersectional Bias in Machine Learning

Ángel Alexander Cabrera, Will Epperson, Fred Hohman, Minsuk Kahng, Jamie Morgenstern, Duen Horng (Polo) Chau

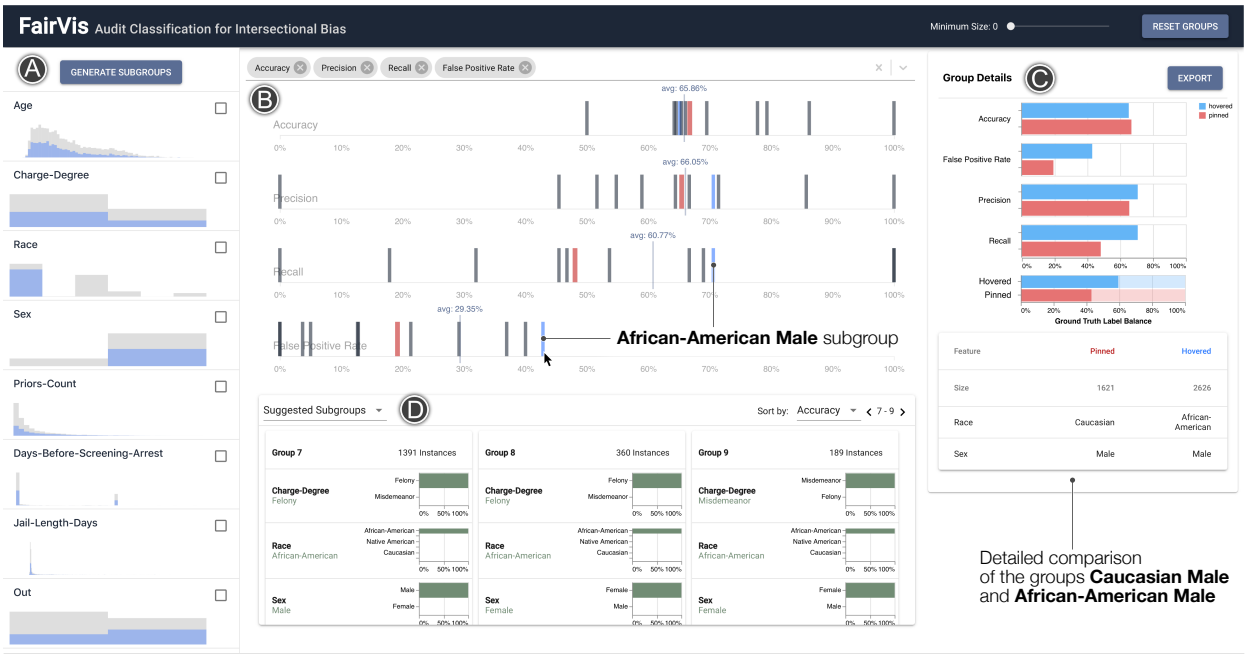


Fig. 1. FAIRVIS integrates multiple coordinated views for discovering intersectional bias. Above, our user investigates the intersectional subgroups of *sex* and *race*. **A.** The *Feature Distribution View* allows users to visualize each feature's distribution and generate subgroups. **B.** The *Subgroup Overview* lets users select various fairness metrics to see the global average per metric and compare subgroups to one another, e.g., pinned **Caucasian Males** versus hovered **African-American Males**. The plots for *Recall* and *False Positive Rate* show that for African-American Males, the model has relatively high recall but also the highest false positive rate out of all subgroups of sex and race. **C.** The *Detailed Comparison View* lets users compare the details of two groups and investigate their class balances. Since the difference in False Positive Rates between Caucasian Males and African-American Males is far larger than their difference in base rates, a user suspects this part of the model merits further inquiry. **D.** The *Suggested and Similar Subgroup View* shows suggested subgroups ranked by the worst performance in a given metric.

Abstract— The growing capability and accessibility of machine learning has led to its application to many real-world domains and data about people. Despite the benefits algorithmic systems may bring, models can reflect, inject, or exacerbate implicit and explicit societal biases into their outputs, disadvantaging certain demographic subgroups. Discovering which biases a machine learning model has introduced is a great challenge, due to the numerous definitions of fairness and the large number of potentially impacted subgroups. We present FAIRVIS, a mixed-initiative visual analytics system that integrates a novel subgroup discovery technique for users to audit the fairness of machine learning models. Through FAIRVIS, users can apply domain knowledge to generate and investigate known subgroups, and explore suggested and similar subgroups. FAIRVIS' coordinated views enable users to explore a high-level overview of subgroup performance and subsequently drill down into detailed investigation of specific subgroups. We show how FAIRVIS helps to discover biases in two real datasets used in predicting income and recidivism. As a visual analytics system devoted to discovering bias in machine learning, FAIRVIS demonstrates how interactive visualization may help data scientists and the general public in understanding and creating more equitable algorithmic systems.

Index Terms—Machine learning fairness, intersectional bias, visual analytics.

1 INTRODUCTION

Over the past few years significant strides have been made in machine learning (ML) research, enabling automated, data-driven systems to tackle ever more challenging and complex tasks. Many of the new domains in which these novel techniques are being applied are human-focused and consequential, including hiring, predictive policing, predicting criminal recidivism, and pedestrian detection. The latter two cases are two of many examples where differing levels of predictive accuracy have been observed for different demographic groups [8, 28].

- *Ángel Alexander Cabrera, Will Epperson, Fred Hohman, Minsuk Kahng, Jamie Morgenstern, Duen Horng Chau are with Georgia Tech. E-mails: {acabrera30, willepp, fredhohman, kahng, jamiemmt.cs, polo}@gatech.edu.*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

When deploying ML to these societally impactful domains, it is vital to understand how models are performing on all different types of people and populations. ML algorithms are usually trained to maximize the overall accuracy and performance of their model, but often do not take into account disparities in performance between populations. The trained models thus provide no guarantees as to how well they will perform on different subgroups of a dataset.

The potential disparity in performance between populations may have many sources; an ML model can naturally encode implicit and explicit societal biases [5], which is often referred to as algorithmic bias. Performance disparity can arise for a variety of reasons: the training data may not be representative, either in terms of its representation of different demographic groups or within a particular demographic group; the training data labels may have errors which reflect societal biases, or be an imperfect proxy for the ultimate learning task; unequal rates of labels across demographic groups; the model class may be overly simple to capture more nuanced relationships between features for certain groups; and more [8]. A stark example of algorithmic bias in deployed systems was discovered by Buolamwini and Gebru's Gender Shades study [7], who showed that many commercially available gender classification systems from facial image data had accuracy gaps of over 30% between darker skinned women and lighter skinned men. While the overall models' accuracies hovered around 90%, darker skinned women were classified with accuracy as low as 65% while the models' accuracies on lighter skinned men were nearly 100%.

In order to discover and address potential issues before ML systems are deployed, it is vital to audit ML models for algorithmic bias. Unfortunately, discovering biases can be a daunting task, often due to the inherent intersectionality of bias as shown by Buolamwini and Gebru [7]. Intersectional bias is bias that is present when looking at populations that are defined by multiple features, for example "Black females" instead of just people who are "Black" or "female". The difficulty in finding intersectional bias is pronounced in the Gender Shades study introduced above — while there were performance differences when looking at sex and skin color individually, the significant gaps in performance were only found when looking at the intersection of the two features. An example of how aggregated measures can hide intersectional bias can be seen in Fig. 2.

In addition to the intersectional nature of bias, addressing bias is challenging since there are numerous proposed definitions of unfairness. These metrics for measuring the fairness of a model include measuring a model's group-specific false positive rates, calibration, and many others. While a user may decide on one or several metrics to focus on, achieving true algorithmic fairness can seem an insurmountable challenge. In Sect. 2, we describe how recent research has shown that it is often impossible to fulfill multiple definitions of fairness at once.

While it can be straightforward to audit for intersectional bias when looking at a small number of features and a single fairness definition, it becomes much more challenging with a large number of potential groups and multiple metrics. When investigating intersectional bias of more than a few features, the number of populations grows combinatorially and quickly becomes unmanageable. Data scientists often have to balance and take into account the tradeoffs between various fairness metrics when making changes to their models.

In order to help data scientists better audit their models for intersectional bias, we introduce **FairVis**, a novel visual analytics system dedicated to helping audit the fairness of ML models. FairVis's major contributions are:

- **Visual analytics system for discovering intersectional bias.** FairVis is a mixed-initiative system that allows user to explore both suggested subgroups and user-specified subgroups that incorporate a user's existing domain knowledge. Users can visualize how these groups rank on various common fairness and performance metrics and contextualize subgroup performance in terms of other groups and overall performance. Additionally, users can compare the feature distributions of groups to make hypotheses about why their performance differs. Lastly, users can explore similar subgroups to compare metrics and feature values.

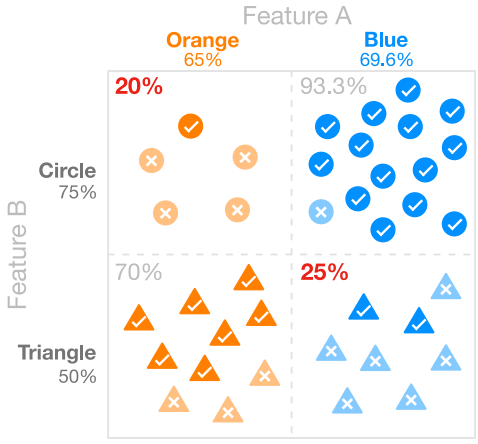


Fig. 2. This illustrative example highlights how inequities in populations can be masked by aggregate metrics. While the classifier in this example has an accuracy of between 50% and 70% when looking at groups defined by a single feature, the accuracy drops to as low as 20% when looking at the intersectional subgroups.

- **Novel subgroup generation technique.** In order to aid users in exploring a combinatorially large number of subgroups, we introduce a new subgroup generation technique to recommend intersectional groups on which a model may be underperforming. We first run clustering on the training dataset to find statistically similar subgroups of instances. Next, we use an entropy technique to find important features and calculate fairness metrics for the clusters. Lastly, we present users with the generated subgroups sorted by important and anomalously low fairness metrics. These automated suggestions can aid users in discovering subgroups on which a model is underperforming.
- **Method for similar subgroup discovery.** Once a subgroup for which a model has poor performance has been identified, it can be useful to look at similar subgroups to compare their values and performance. We use similarity in the form of statistical divergence between feature distributions to find subgroups that are statistically similar. Users can then compare similar groups to discover which value differences impact performance or to form more general subgroups of fewer features.

2 Background in ML Fairness

Significant discoveries and advances have been made in algorithmic bias and machine learning fairness in recent years. Most of the work has come from theoretical computer scientists and sociologists focusing on the mathematical foundations and societal impacts of machine learning.

A major difficulty in machine learning fairness is that it is mathematically impossible to fulfill all definitions of fairness simultaneously when populations have different base rates. This incompatibility between fairness metrics was formalized by the *impossibility theorem* for fair machine learning. Two papers [12, 20] simultaneously proved that if groups have different base rates in their labels, it is statistically impossible to ensure fairness across three base fairness metrics — balance for the positive class, balance for the negative class, and calibration of the model. Data scientists must therefore decide which fairness metrics to prioritize in a model and how to make trade-offs between metric performance.

The implications of this discovery were made apparent in the recidivism prediction tool COMPAS, a system that is used to predict the risk of letting someone go on bail. A ProPublica article [3] showed that COMPAS is more likely to rank a Black defendant as higher risk than a White defendant given that they have equal base rates. A follow-up study showed that while COMPAS is not balanced for the positive class prediction, it is well calibrated, meaning that the model provides similarly accurate scores for both groups relative to their base rates [11].

Due to inherent base rate differences, it is not possible for COMPAS to meet the all three fairness definitions at once.

There have been various solutions proposed for addressing algorithmic bias in machine learning across the entire model training pipeline. These range from techniques for obfuscating sensitive variables in training data [29], to new regularization parameters for training [6] and post-processing outcomes by adding noise to predictions [14]. While these can help balance certain inequities, the impossibility theorem dictates that hard decisions will still have to be made about which fairness metrics are the most important for each problem. Ideally, over time these will become standard processes for ensuring model fairness, and tools like FAIRVIS can be used to ensure their effectiveness and investigate tradeoffs between metrics.

3 RELATED WORK

3.1 Intersectional Bias

Important innovations have come from the machine learning community in relation to intersectional bias.

Kearns et al. [19] proposes a framework for auditing a (possibly very large) number of subgroups for unfair treatment. Their work has the same high-level concerns that motivate this project: that there may be a very large number of intersectional groups over which one wants to satisfy some notion of fairness. However, for their work, they assume the collection of these groups is predefined for the task at hand, and construct an algorithm for creating a distribution over classifiers which (approximately) minimizes a particular fairness metric over all the subgroups simultaneously. Our work differs from theirs in several key ways. First, we aim to operate in a space where a predefined notion of groups is not necessarily well-defined, and so cooperation between an automated system and a domain expert might be necessary to uncover subgroups whose treatment by a particular model is problematic. Second, our goal is to help a user explore their model and dataset for a deeper understanding of *why* the model might be treating particular groups very differently, a far different task compared to picking a distribution over a set of models to satisfy a particular fairness metric.

We hope this will naturally suggest interventions and promote a deeper understanding of a learning task, a dataset's suitability to this task, and whether a model (class) matches the dataset and task. In particular, exploring a model and dataset in this way should help encourage our tools' users to consider whether their dataset is simply inappropriate for high-stakes tasks where fairness is a concern; if the learning task demands more complex models to guarantee fairness for a larger number of groups; whether the datset need to be augmented with additional data from certain subgroups, perhaps with certain labels; whether the model's training need to emphasize its classification performance on some subgroups more heavily; or whether additional features might be necessary for improving accuracy and equitable treatment of numerous subgroups.

Recent techniques have also been proposed for discovering and analyzing intersectional bias. Most similar to our work is Slice Finder [9], a technique for automatically generating subgroups. Slice Finder takes a top-down approach to generating subgroups, adding features to create more granular groups until the training loss is statistically significant. Our technique for automated subgroup discovery is bottom-up, clustering instances without imposing any structure on the features used. In addition to potentially generating more diverse subgroups, our subgroups are not tied to training loss, allowing us to use any performance metric to order and suggest subgroups.

3.2 Visual Analytics for Machine Learning

There is a large body of work on visual analytics techniques for understanding and developing machine learning models [2, 18, 21, 23, 25]. Various systems have been created focused on helping users understand how complex models work and visually debugging their outputs [26]. Additionally, systems have been introduced that enable users to analyze production-level models [17] and the full ML workflow from training to production [2, 21, 23]. These visual systems have been shown to aid users in understanding and developing machine learning models [16].

The research most directly related to the present work are techniques for analyzing both datasets and the results of machine learning models. MLCube [18] is a visualization technique that allows users to compare the accuracy of groups defined by at most two features. Squares [25] introduces a novel encoding for visually understanding the performance of a multi-class classifier. Finally, Facets [1] is a visualization system for interactively exploring and subdividing large datasets. While these systems provide novel and useful methods for exploring data and outcomes, they are limited by the complexity of subgroups and number of performance metrics they support, essential features for auditing for intersectional bias.

While there are various visual systems for analyzing machine learning models, there have been few advancements in visual systems or techniques focused on algorithmic bias and fair machine learning. One notable exception is the *What-If* tool from Google [13]. The *What-If* tool is a more general data exploration tool that combines dataset exploration with counterfactual explanations and fairness modifications. Users can explore a dataset using the Facets interface, and then look for counterfactual [22] explanations for specific instances. There is also a feature that allows users to modify a classifier's threshold to change which fairness principles are being satisfied. While the *What-If* tool is a powerful data exploration tool, it does not allow users to explore intersectional bias nor does it aid users in auditing the performance of specific subgroups.

4 DESIGN CHALLENGES AND GOALS

Our goal is to build an interactive visual interface to help users explore the fairness of their machine learning models and discover potential biases they should address. Many of the challenges present in auditing for bias derive from the combinatorial number of subgroups generated when looking at various features. Additionally, any visual system must convey multiple fairness metrics for a subgroup. A successful visual system should allow users to quickly narrow the large search space of possible subgroups. We formalize these important factors in the design of FAIRVIS with the following key design challenges:

4.1 Design Challenges

C1. **Auditing the performance of known subgroups.** For many datasets and problem definitions, users already know of certain populations for which they want to ensure fair outcomes. It is often cumbersome and slow to manually generate and calculate various performance metrics for subgroups. A system should enable users to generate any type of subgroup they want to investigate, and efficiently generate and calculate metrics for it.

C2. **Contextualizing subgroup performance in relation to multiple metrics and other groups.** To measure the severity of bias against a certain subgroup, it is important to know how the subgroup is performing in relation to the overall model. Any visual encoding of subgroup performance should convey how groups perform for different performance metrics and in relation to other subgroups. Our interface should also allow users to drill down into subgroup details while maintaining the high-level view.

C3. **Discovering significant subgroups in a large search space.** When investigating intersectional bias, there could be hundreds or thousands of subgroups a user may need to look at. It is often not feasible to analyze every group, so deciding how to prioritize subgroups is an important and difficult task. Methods for discovering and suggesting potential groups can aid users in searching this large space and finding potential issues more efficiently.

C4. **Finding similar subgroups to investigate feature importance and more general groups.** When a biased subgroup has been identified, it can be informative to look at the performance of similar subgroups to draw conclusions about feature importance or to create more general groups. This is a difficult task since there are an immense number of potential subgroups that have to be searched to find similar subgroups, and it is not clear how similarity between subgroups should be defined or calculated.

C5. **Emphasizing the inherent trade-offs between fairness metrics.** Classifiers are often not able to fulfill all measures of fairness if the base rates between populations are different, as proven by the *impossibility of fairness* theorem (Sect. 2). This means users often have to keep in mind the tradeoffs between fairness metrics when deciding what modifications to make to their models. It is essential to show the various fairness metrics when displaying subgroup performance and emphasize their tradeoffs.

C6. **Suggesting potential causes of biased behavior.** How to address bias in machine learning models is a difficult and open question, but there are indicators that can help users start to improve their models. Emphasizing information like ground-truth label balance and subgroup entropy can point users in the right direction for addressing biases.

4.2 Design Goals

Using the design challenges we identified for ML bias discovery, we iterated and developed design goals for FAIRVIS. The following goals address the challenges presented in Sect. 4.1, and align with the primary interface components of our system:

G1. **Quick generation and investigation of user-specified subgroups.** Since users often have domain knowledge about important subgroups they want to ensure fairness for (C1), quickly generating these groups to enable investigation is vital. Users should be able to select either entire features (e.g. "race") or specific values (e.g. "white" or "black") to generate groups of any feature combination (C3). Users should then be able to explore the performance of these groups in detail.

G2. **Combined overview relationships with detailed information of subgroup performance.** To understand the magnitude and type of bias a model has encoded for a subgroup, it is important to show the performance of the group in relation to the overall and other subgroups' performance (C2). At the same time, the interface should also display detailed information about the performance of the selected subgroup (C6). We aim to achieve this goal by using multiple, coordinated views that are customizable for different fairness metrics (C5).

G3. **Suggested under-performing subgroups for user investigation.** When more than a couple features are used to define subgroups, the number of generated groups grows combinatorially, often times into the hundreds (C3). We aim to develop both an algorithmic technique for automatically discovering potentially under-performing subgroups and an intuitive visual encoding for suggesting discovered groups to the user. By suggesting these groups automatically, we can make the subgroup discovery process quicker and potentially discover groups the user had not originally thought about (C2).

G4. **Efficient calculation of similar subgroups.** For any given subgroup, there is a combinatorially large space of groups that need to be searched to find similar groups (C3). Since it is often useful to look at similar subgroups to analyze the importance of certain features or to generate more general groups, we aim to develop a technique that efficiently discover these similar groups (C4, C6).

G5. **Effective visual interfaces for subgroup comparison.** Users may want to analyze two subgroups side by side to compare their values or performance (C2). We aim to provide an intuitive interface for highlighting the differences between two groups. Users can compare these groups to help pinpoint which features or values are causing the difference in fairness metrics (C6).

5 FAIRVIS: AUDIT CLASSIFICATION FOR INTERSECTIONAL BIAS

Based off of the design challenges introduced in Section 4, we have developed FAIRVIS, a visual analytics system for discovering intersectional bias in machine learning models. To meet the design goals we set out, we developed two novel techniques to generate underperforming subgroups and find similar subgroups. We combine these techniques in a web-based system that uses multiple, coordinated views to allow users to quickly discover fairness issues in known and unknown subgroups.

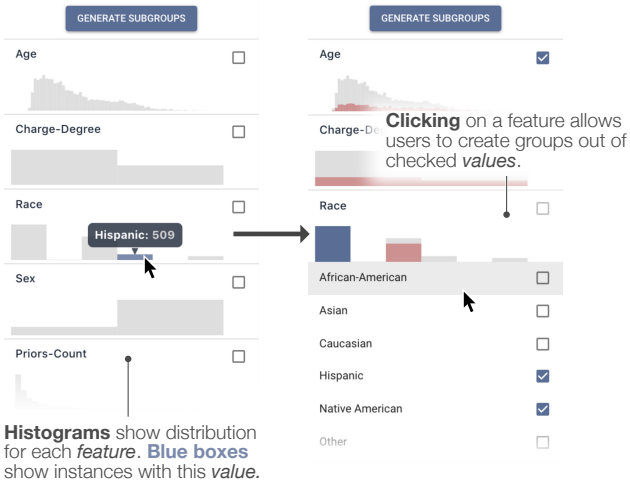


Fig. 3. The *Feature Distribution View* allows users to explore both the distributions of each feature in the entire dataset and also create user-specified groups out of features or specific values. When a user hovers over a bar such as "Hispanic", it shows the number of instances for that value. Red bars show the distribution of the pinned group (in this case "Male Caucasians") from the *Subgroup Overview* .

Our interface consists of four primary views, the *Feature Distribution View* (Sect. 5.1), *Subgroup Overview* (Sect. 5.2), *Suggested and Similar Subgroup View* (Sect. 5.3, Sect. 5.4), and *Detailed Comparison View* (Sect. 5.5). The *Feature Distribution View* gives users an overview of the dataset distribution and allows them to generate groups to visualize in the *Subgroup Overview*. Users can then add additional subgroups provided by the *Suggested and Similar Subgroup View*, and compare and further analyze them in the *Detailed Comparison View*. Each section of our interface aligns with one of the stated design goals, addressing each desired feature.

5.1 Feature Distribution View and Subgroup Creation [G1]

The left sidebar, or *Feature Distribution View*, acts as both a high-level overview of a dataset's distribution and the interface for generating user-specified subgroups. As a starting place for FAIRVIS, the *Feature Distribution View* helps users develop an idea of their dataset's makeup and begin auditing subgroups right away.

Feature distribution. A large part of understanding model performance is understanding how the data used to train a model is distributed (C6). We enable users to investigate feature distributions by providing large, interactive histograms for each feature for the entire dataset, as seen in Fig. 3. When a user hovers over a bar, a tooltip shows what the value of the bar is and how many instances there are with that value in the entire dataset. Furthermore, clicking on one of the rows reveals a collapsible view of all the possible values for the feature. Users are also able to hover over the expanded values to see their location in the histogram.

Subgroup Generation. In addition to exploring the distribution of features in a dataset, the *Feature Distribution View* allows users to generate user-specified subgroups. Model architects are often aware of certain intersectional subgroups for which they want to verify fairness (C1). We define a subgroup as a subset of a dataset in which all instance share certain values, for example the subgroup of blue circles in Fig. 2.

Our interface allows users to generate both specific subgroups and all subgroups of multiple features by selecting a combination of features and values. For instance in Fig. 3, if a user checks the feature "race" and "sex", then mutually exclusive subgroups will be generated out of

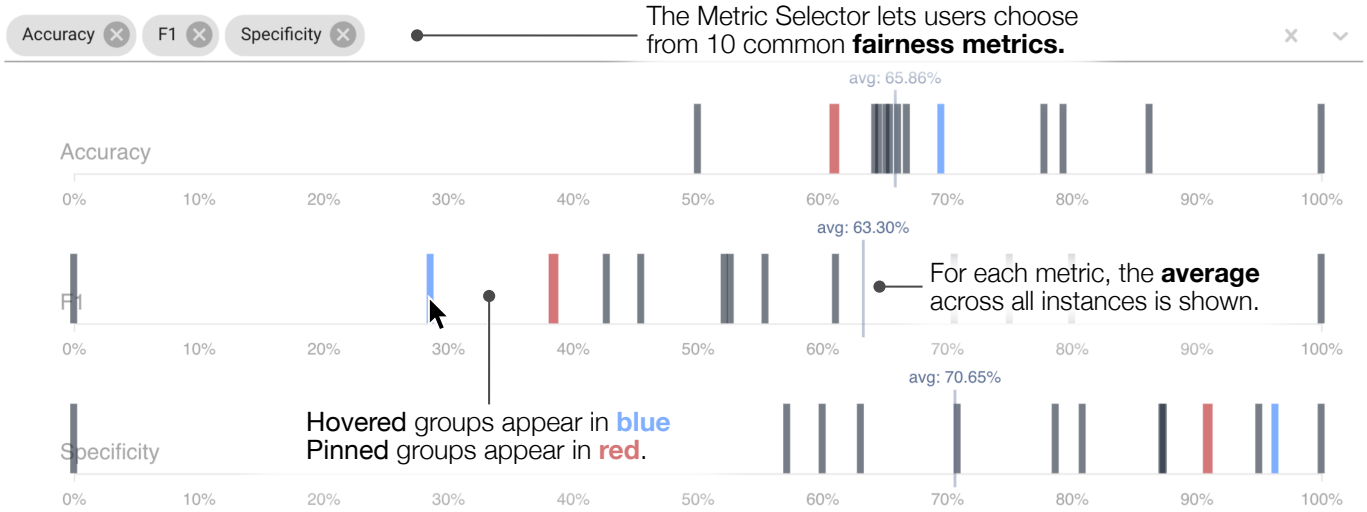


Fig. 4. In the *Subgroup Overview* users can see how different subgroups compare to one another according to various performance metrics. As more metrics are selected at the top, additional strip plots are added to the interface. As users hover or pin a subgroup in one strip plot, the same group is highlighted in the other plots.

all the instances in the dataset divided on their values for "race" and "sex". However, if a user wants to investigate a particular subgroup, they can select a specific value for "race" and "sex" to add a subgroup of all instances with those specific values. Users can pick any number and combination of features and values by which to define their subgroups, and thus are at liberty to define how general or specific the subgroups they want to explore are.

5.2 Subgroup Overview [G2]

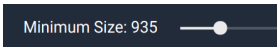
Once a user has generated subgroups, they should be able to quickly tell which subgroups the model is underperforming on, how they are performing for various metrics, and further investigate interesting subgroups (C2). The *Subgroup Overview* provides a high-level view of this information in the form of multiple interactive and dynamic strip plots (C2).

When a user clicks the "Generate Subgroups" button (Fig. 3), FAIRVIS splits the data into the specified subgroups and calculates various performance metrics for them. These groups are then represented in the multiple strip plots as lines corresponding to their performance for the respective metric.

Visualizing multiple fairness metrics. Due to the inherent trade-offs between different fairness requirements as shown by the *impossibility theorem*, users must choose which metrics they want to prioritize and investigate (C5). To facilitate this interaction, we allow users to select which metrics are displayed in the *Subgroup Overview* by adding and removing performance metrics through the bar seen at the top of Fig. 4. Selecting a new metric adds an additional strip plot for that metric with all the current subgroups. We also show the corresponding dataset average per metric in each strip plot to provide context as to how each subgroup is doing in relation to the overall dataset.

In total, users can select from the following metrics: Accuracy, Recall, Specificity, Precision, Negative Predictive Value, False Negative Rate, False Positive Rate, False Discovery Rate, False Omission Rate, and F1 score. These metrics were selected as they correspond to the most commonly used metrics in evaluating the equity and performance of binary classification models. All of these performance metrics are derived from the same base outcome rates of true positives, true negatives, false positives, and false negatives. If users find that they need different metrics for performance, they can add a new definition using the calculated base rates which will be available in the interface.

When a user hovers over a subgroup in one of the strip plots, the corresponding group is highlighted on every plot currently displayed. This allows users to see how an individual group performs on several different metrics at once [C2, C5]. To further investigate a subgroup, the user can click on a bar to pin the group and use the *Detailed Comparison View*.



Choice of visual encoding. We chose a strip plot to visualize performance metrics since it allows users to focus on the relative magnitude of subgroup performance in relation to other subgroups and the overall dataset performance. One of the shortcomings of strip plots is that they can become crowded and hard to use with a large number of subgroups. We address this issue with an important and useful interaction, filtering the strip plot by subgroup size. While it is important to consider subgroups of various sizes, groups that are only a few instances are usually not statistically significant enough to draw conclusions from. The size filtering mechanism can help users narrow their search space (C3) and improve the functionality of the strip plot.

While designing our system we considered different visual encodings for displaying subgroups, especially a scatterplot matrix. We decided to use a strip plot over a scatterplot matrix for several reasons:

1. **Predictable trends and insignificant outliers.** Since each of the performance metrics is derived from the same base rates, many of the relationships between metrics are arithmetic and not indicative of interesting patterns. We did investigate outliers and found that they did not systematically represent any interesting subgroups.

2. **Redundant encoding.** Scatterplot matricies redundantly encode information, as every metric is displayed multiple times. Our strip plot implementation only includes each metric once while still allowing users to see how the group performs in regards to other metrics.

Multiple strip plots allow us to display the most important information in a clean and quickly understandable manner; namely, how a given subgroup is performing for selected metrics and in relation to the overall dataset and other subgroups.

5.3 Suggested Subgroups [G3]

While many users may know of certain groups in their dataset they need to ensure fairness for, it is possible that the model architect has

little domain knowledge and doesn't know where to start. Since there are a combinatorially large number of subgroups in a dataset, it is daunting and often times not feasible to manually inspect groups for every combination of features.

To help the user find potentially biased subgroups, we generate subgroups algorithmically and present them to the user for investigation. The *Suggested and Similar Subgroup View* at the bottom of the interface displays these subgroups and allows the user to sort them by any fairness metric to discover significantly underperforming subgroups (C3).

5.3.1 Generating and Describing Suggested Subgroups

To create the suggested subgroups, we use a clustering-based generation technique. By clustering instances, we can generate groups with significant statistical similarity that can be described by a few dominant features. We can subsequently calculate their performance metrics and display them to the user.

We first cluster all the data instances by their feature values in one-hot encoded form. We use K-means as our clustering algorithm [15] with K-means++ as the seeding [4]. Users are able to choose the hyperparameter K to balance the number and size of generated subgroups — a smaller K produces larger, less defined groups while a larger K has the opposite effect. Users run the clustering as a pre-processing script before uploading their data to FAIRVIS.

We originally experimented with more sophisticated clustering algorithms like the density-based algorithms DBSCAN and OPTICS, which can generate arbitrarily shaped and sized clusters. While the statistical quality of the density-based clusters can be higher, we found that the flexibility provided by allowing users to modify K is more helpful for discovering important and useful subgroups.

Once the clusters have been generated, it is necessary to describe the makeup of the group to the user. A cluster's instances are made up of a variety of values for each feature, but some features may be more dominated by one value than others. We define a *dominated feature* as a feature that consists of mostly one value, the *dominant value* in a subgroup. For example, if a cluster is 99% male for the feature sex, sex is a dominated feature with a dominant value of male.

The most dominant features can be used to describe the makeup of a subgroup to the users. We rank how dominant features of a group are by calculating the entropy of each feature distribution over its values. Entropy is used since it describes how uniform a feature is. The closer a feature's entropy is to 0, the more concentrated the feature is in one value, making it more dominant in that subgroup.

We formalize the technique for finding dominant features as follows. Suppose we have a set of features, $\mathscr{F} = \{f_1, f_2, ..., f_i, ...\}$, with each feature, f_i, having a set of possible values, $V_i = \{v_{i1}, v_{i2}, ...\}$. We calculate the *feature entropy* for the k-th subgroup and i-th feature, $S_{k,i}$, as follows:

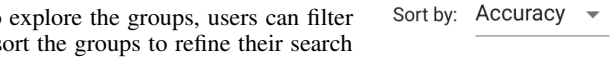
$$S_{k,i} = -\sum_{v \in V_i} \frac{N_{k,v}}{N_k} \log \frac{N_{k,v}}{N_k}, \tag{1}$$

where N_k is the number of instances in the k-th subgroup, and $N_{k,v}$ is the number of instances in the k-th subgroup with value v. For example, if all the instances of subgroup k have value $v_{3,1}$ (e.g., India), for the feature f_3 (e.g., native country), the feature entropy is 0 and f_3 is a dominant feature for the subgroup.

5.3.2 Displaying Suggested Subgroups

We display the generated subgroups in the *Suggested and Similar Subgroup View* at the bottom of the interface, as seen in Fig. 5. Since the generated subgroups are not strictly defined by a few features, it is important to show the feature distributions for each feature in a group. Each suggested subgroup has a list of its features and dominant value, along with a histogram of the value distribution for each feature. The features are sorted according to their dominance, with the dominant value being displayed under the feature name. This interface allows users to quickly see what values a subgroup is made up of and develop an idea of what subgroup may be underperforming.

To explore the groups, users can filter and sort the groups to refine their search space (C3). Since users may find certain metrics more important than others for certain problems, they can choose which metrics to sort the suggested groups by in ascending order (C5). For example, if for a given problem recall is an important metric, users can quickly find generated subgroups with the lowest recall.



Furthermore, users can use the same size slider used to filter the *Subgroup Overview* by size to filter the generated subgroups. Similar to the reasoning for filtering by size in the strip plot, very small groups may not be large enough to draw statistically significant conclusions from. Filtering the groups can remove noise and help users further refine their search space of problematic groups.

Users can hover over a suggested subgroup card to show its detailed performance metrics in the *Detailed Comparison View* and add the group to the *Subgroup Overview*. If a user wants to investigate the group further, they can click on the card, pinning the group and allowing them to compare it to other subgroups or export it for sharing.

5.4 Similar Subgroups [G4]

Once a user has discovered an interesting subgroup, it can be helpful to look at similar subgroups to either investigate the impact of certain features or to find more general groups with performance issues (C4). Finding similar groups is difficult since it is not a well defined task and can require searching a combinatorially large space.

To formalize similarity and refine the subgroup search space, we apply ideas from statistics and machine learning explainability to this task. When comparing suggested subgroups, we use similarity in the form of statistical divergence to compare how closely related groups are. For user-specified subgroups, we apply the concept of counterfactual explanations by finding groups with minimal value differences that have significantly different performance.

5.4.1 Finding Similar Subgroups

Similarity between subgroups can be thought of as the statistical distance between the feature distributions of groups; the more values two subgroups share, the more similar we consider them. Statistical distance can be measured in a variety of ways, but we found Jensen-Shannon (JS) divergence to be a good measure for our use case. As a derived form of Kulleback-Leibler divergence, JS divergence is a similar measure with the benefits of being bi-directional and always having a finite value. Since we often have zero-probability values, JS divergence makes calculating statistical similarity more straightforward and standardized.

We calculate similarity between groups by summing the JS divergence between all features for a pair of subgroups. This sum gives us a measure of how similar two subgroups are on aggregate. Formally, we calculate the total distance D between subgroups k and k' as follows, Where $G_{k,f}$ represents the value distribution of feature f in subgroup k.

$$D(k,k') = \sum_{f \in \mathscr{F}} \mathrm{JS}(G_{k,f} || G_{k',f}) \tag{2}$$

This definition of subgroup similarity applies most directly to the suggested subgroups that have some distribution over values for each feature. In practice for FAIRVIS, we have a combination of user-specified subgroups of selected features and suggested subgroups with feature distributions.

When comparing two suggested subgroups against each other, we can use the formal definition of JS divergence and sum the average distance of their feature distributions. For comparing user-specified and suggested subgroups against each other we can use a similar technique with a small optimization — since user-specified subgroups will have 0 probability for all values but the selected values in each feature, it is only necessary to calculate the JS divergence for the values present in the user-specified group.

User-specified subgroup comparison. The final potential case for comparison is between two user-specified subgroups. The use of JS divergence as a measure of similarity begins to break down and lose

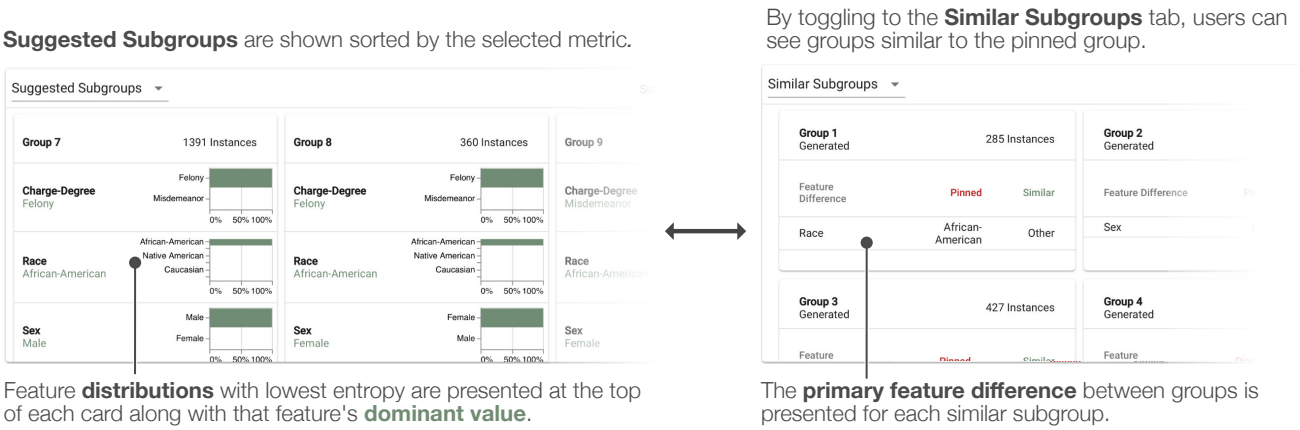


Fig. 5. Here we can see the *Suggested and Similar Subgroup View* for both suggested and similar subgroups. On the left we see suggested subgroups for the dataset along with their dominant values and feature distributions. On the right we see similar subgroups to the subgroup that is currently pinned. The primary feature difference is displayed to emphasize the counterfactual nature of the similar subgroups. Users can hover over any card to see detailed feature and performance information in the *Detailed Comparison View*. They can also sort the subgroups by any performance metric and filter by subgroup size to narrow their search space.

its utility for this use case. The divergence will only ever be one when groups have the same value for a feature or zero when they do not. This metric in practice just counts the number of features with the same value between two groups. While this measure provides some information about subgroup similarity, it is not as informative or accurate as it is when comparing distributions over features in the other two cases.

To provide a more useful comparison of groups, we use the idea of counterfactual explanations [27]. Counterfactual explanations are usually presented in the following form: What are the minimum number of features we have to change to switch the classification of an instance?

Since we are looking at subgroups of multiple instances instead of individual examples, we use a modified notion of counterfactuals for comparing user-specified subgroups: If we only switch one or two feature values for a subgroup, which similar groups have the most surprising changes in performance? This question can help users answer similar questions as they would for the groups found using JS divergence.

Calculating the counterfactual groups is straightforward — from the currently generated groups, we find which ones differ by one or two features from the selected group and display those to the user.

5.4.2 Displaying Similar Subgroups

Once similar subgroups have been found for a selected subgroup, we reuse the *Suggested and Similar Subgroup View* from Sect. 5.3 to display the groups to the user. Each subgroup is represented by a card containing a group number and the size of the subgroup. Since selecting a subgroup displays its information in the *Detailed Comparison View*, only the information most pertinent to deciding which subgroup to investigate should be displayed.

Continuing with the philosophy of treating similar groups as counterfactuals, we display the primary feature difference between two groups in the case of user-specified subgroups, and the most divergent feature for suggested subgroups. By displaying the feature difference, we emphasize the importance of that feature in the performance difference between the groups.

The same two primary interactions are available for exploring similar groups that are available for suggested groups: sorting and filtering (C3). Users can sort the groups by any fairness metric and filter the groups by size. As with the strip plot and suggested views, this mechanism helps users find statistically significant subgroups that the model is underperforming for in metrics the user finds important.

Similar subgroup importance. Similar subgroups can be informative in two primary manners: finding features which are important for performance and discovering more general subgroups. Given that we are looking at two similar subgroups, they likely only differ in one or two features. If the performance between these two groups is vastly different, it is indicative that the features which are different may contribute significantly to performance (C6). On the other hand, if the two groups have very similar performance, it may mean that a broader subgroup not split using the differing features is also underperforming and should be analyzed.

5.5 Detailed Subgroup Analysis and Comparison [G5]

The final step in discovering and formalizing group inequity is to examine the details of a subgroup's features and performance. We enable this interaction with the *Detailed Comparison View* on the right hand side of the system. In addition to allowing users to explore more detailed information about a subgroup, it allows them to compare two subgroups against each other.

A user is able to see the details for two groups in the *Detailed Comparison View*, the pinned and hovered group. A group can be pinned when a user clicks on it in the *Subgroup Overview* or *Suggested and Similar Subgroup View*, and is designated by a light red across the UI. The hovered group is designated by a light blue across the UI. Since the these are the only two distinct colors across the interface, users can easily see a selected group's information across various different views.

There are three primary components in the *Detailed Comparison View*, as seen in Fig. 6 The top-most component is a bar chart displaying how a group performs for selected performance metrics. While users can see the values of the fairness metrics in the strip plot, the bar chart allows users to see the specific values and enables comparison between groups with a grouped bar chart (C5). The grouped bar chart also enables direct comparison between the pinned and hovered subgroups without the distraction of other groups.

The second component in the *Detailed Comparison View* is a bar chart for the ground truth label balance of both selected subgroups. The label imbalance is important because it can often explain extreme values for metrics like recall and precision and can suggest reasons for bias (C6). For example, a subgroup with 95% negative values can get a 95% accuracy by classifying everything as negative, even though it will have a 0% sensitivity.

The final subgroup comparison interface is a table delineating and comparing the features of the pinned and hovered subgroups. For user-specified subgroups, this table shows the features and values that define the subgroup. For suggested subgroups, this shows the top 5 dominant feature values for that group, and users can see the full distribution in the *Suggested and Similar Subgroup View* view.

Subgroup feature distributions. There is additional information about the pinned and hovered subgroup in the *Feature Distribution View*. When a subgroup is hovered or pinned, a histogram of each feature's

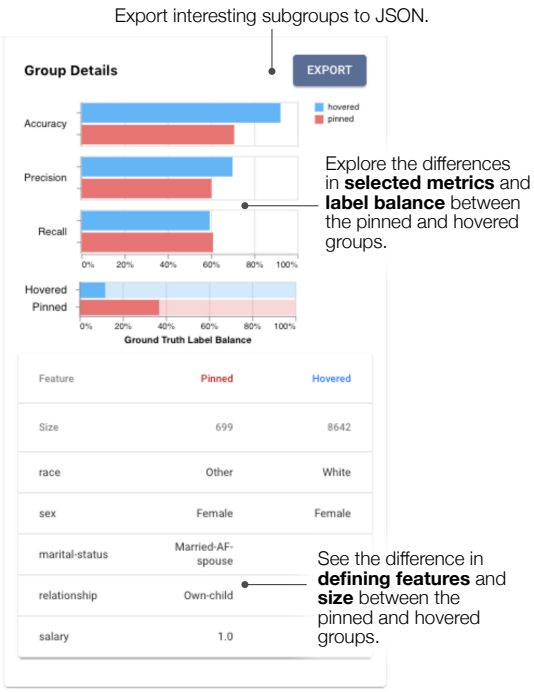


Fig. 6. In the *Detailed Comparison View* users can compare the performance and makeup of the pinned and hovered subgroups. Comparing the groups can provide users with an understanding of the causes for performance differences.

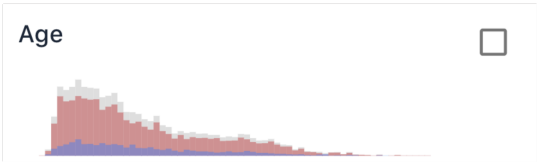


Fig. 7. When groups are pinned and hovered, users can compare their feature distributions in the *Feature Distribution View* . A subgroup's distribution can inform users about the diversity of the group's instances, an important factor in model performance.

distribution for that group is overlaid on the overall distribution (C2). When there is both a pinned and hovered subgroup, the histograms are overlaid with opacity, allowing users to see how similar the distributions are (Fig. 7).

The distribution of a subgroup's features can be an important indicator of why a subgroup is underperforming and suggest potential resolutions (C6). If a subgroup's ground truth labels are well balanced, there should be some diversity in the other features of a subgroup for the classifier to be able to discriminate between the two labels. For example, if all White males are also high school educated, married, and from the United States, and they are split between positive and negative classes, it is nearly impossible for a classifier to accurately predict the class for anyone in that subgroup.

An extra interaction in the *Detailed Comparison View* is an export button for sharing a discovered subgroup. Once a user has found a subgroup they want to investigate further or modify their model and dataset for, they likely want to share the details of the subgroup with others. The export button makes this process easy, allowing users to save a JSON file of the subgroup with important details about the subgroup. This includes the instances in the group, the defining features and values, the calculated metrics, and the feature distributions.

The *Detailed Comparison View* allows users to compare subgroups and make final, informed decisions about how well a machine learning model is performing in regards to fairness.

6 Use Cases

In this section we describe how FairVis can be used in practice with two example usage scenarios on distinct datasets and problem settings. The first scenario highlights how FairVis can be used to audit models for biases against known vulnerable groups in the context of a recidivism prediction system. The second use case shows how users without previous knowledge or intuitions about potential biases can use the system to find issues, for this example with an income prediction model.

6.1 Auditing for Known Biases in Recidivism Prediction

For our first example use case, consider a district court which uses an algorithmic recidivism prediction tool for helping decide who should be given bail. The team who uses the tool has recently heard of research showing that a popular system similar to what they are using, COMPAS, was found to have worse performance when making predictions for Black defendants [3], as described in Sect. 2. To ensure that their system does not have similar issues, they have hired Janel, an independent data scientist, to audit their recidivism prediction tool for potential biases.

The system in question takes information about a defendant and outputs a binary decision as to whether or not a defendant is deemed high risk, which judges subsequently use to decide whether to release a defendant on bail. The system was trained on the ProPublica dataset comprised of example defendants, and outputs COMPAS recidivism scores based on ground-truth labels corresponding to whether a defendant released on bail was arrested for another crime within 2 years of their release [24]. For any COMPAS score considered "high", the model takes this as a positive prediction for risk. The goal of the system is to have high accuracy and equitable treatment of defendants of different genders and races. While such a system may help remove implicit and explicit biases judges may have, Janel knows that judges selected which defendants in some pool were released to comprise this dataset, that arrests for crimes are not equally likely for all genders and races, and that this dataset and resulting models may therefore have encoded these biases. To audit the system for possible inequitable performance, Janel loads the dataset into FairVis and begins her analysis.

Known subgroup auditing. Janel is aware that in previous applications involving recidivism prediction, many tools have displayed imbalanced performance for certain genders and races. To test whether differing performance holds for this model and dataset, Janel uses the *Feature Distribution View* to generate all intersectional subgroups of race and sex. When the groups are added to the *Subgroup Overview* (Fig. 1B), Janel immediately sees that the groups are spread out broadly across various metrics, suggesting this model may have very different predictive performance on different subgroups. While Janel is interested in the accuracy of her model, she cares most about whether her model has large intra-group variation in terms of its false positive rate — how many of the people who are not risky are classified as risky, and are these mistakes distributed unevenly across the different demographic groups? A high false positive rate for this model indicates that many low-risk people (who might be good candidates for release on bail) would be labeled as high-risk by the model. If this model were used to help determine whether a person was seriously considered for release, false positives would correspond to low-risk candidates for release who might be passed over for bail.

To audit the false positive performance metric, Janel adds a strip plot for it using the metric selector. She then hovers over the bar in the false positive rate strip plot with the highest value, and sees that African-American males have a 43% false positive rate compared to the dataset average of 29%. Janel pins the subgroup to investigate it further and compare it to other groups.

By hovering over the other subgroups, Janel sees that the base rate (referring to having been arrested for another crime within 2 years of release) for African-American males is higher in this dataset than for other subgroups. Thus, if a model makes only one prediction for the entire subgroup, choosing to label the subgroup as positive (a prediction of high recidivism risk) will have higher accuracy than for other subgroups. Less extreme versions of this statement may still hold: to maximize accuracy for this subgroup, a model will use a larger

number of positive labels than negative labels. Janel is aware of many historical biases in the dataset used to train the model, and prepares a report detailing the imbalanced performance with respect to race, gender, and false positive rates she has found.

Investigating Suggested Subgroups. In addition to the biases she discovered using her domain knowledge, Janel turns to the *Suggested and Similar Subgroup View* panel to find other potentially problematic groups. Janel first sorts the suggested groups by their false positive rate, since she is most worried about that metric. While the first few groups with the highest false positive rate are made up of African-American males, corroborating her earlier findings, one of the following groups provides a different result.

The third generated group is relatively large with 249 instances, and has a high false positive rate of 39%. By inspecting the make-up of this group in the *Detailed Comparison View*, Janel sees that it is made up of Caucasian females with a felony charge. The label imbalance for this group is not a pronounced as for African-American males, giving Janel two hypotheses about sources of this high false positive rate. Her first hypothesis is that the rather small group was not large enough to have been given priority in training; the second is that the class of models considered during training may have been too simple to express the difference between classes in this subgroup. To investigate this group further, Janel saves this group using the export button.

6.2 Discovering Biases in Income Prediction

Next, let us consider Judy, a government employee who is in charge of offering loan forgiveness to people in her district who make less than $50,000 a year. When Judy needs to make decisions about which loans to forgive, she does not have annual income figures for those who have applied. She does, however, have a variety of demographic information about each candidate for loan forgiveness. She hypothesizes that she can train a machine learning model using the UCI Adult Dataset [10] that predicts whether someone makes under $50,000 a year, allowing her to allocate loan forgiveness to lower income candidates with higher fidelity.

Model training. Judy is an experienced data scientist, and does a thorough job of model selection. After testing different types of models and hyperparameters, she finds that a two-layer neural network performs best, with an overall accuracy of 85%. While Judy is encouraged by the high accuracy of her model, she is aware of recent news of algorithmic bias and wants to ensure that her model is treating different demographic groups with similar predictive performance. She decides to audit her model using FAIRVIS, and loads her dataset and predictions into the system.

Dataset exploration and subgroup creation. When first opening FAIRVIS, Judy uses the *Feature Distribution View* on the left to look at how balanced her dataset is. While Judy is unaware of any biases in her data, she immediately notices from looking at the feature histograms that the dataset has a disproportionate representation of males, with males making up more than 2/3 of all instances. To investigate the impact of this imbalance, Judy selects the feature for sex to generate male and female subgroups. When just looking at two subgroups, Judy sees in the *Subgroup Overview* that there is a gap of almost 10% in model accuracies between the male and female subgroups.

Suggested subgroups. After seeing the fairly large gap in the accuracy of her model between subgroups defined by just one feature, Judy is curious about what other combinations of features might lead to poor performance in her model. She turns to the *Suggested and Similar Subgroup View* to see what she can find. Keeping the default sorting of groups by lowest accuracy, Judy immediately notices that suggested Group 2 has an accuracy of around 72%, far below the average of 85%. By inspecting the feature distribution charts in the *Suggested and Similar Subgroup View* , Judy sees that this group is mostly comprised of female high school graduates working in the private sector. Since Judy wants to better understand why her model is performing poorly for this group, she tries exploring similar groups.

Similar subgroups. Using her discovery from the Suggested Subgroups tab, Judy wants to see how groups of high school educated females compare to one another across workclasses. She generates the subgroups in the *Feature Distribution View* and pins the subgroup of high school educated females in the private sector and inspects similar groups. Here, she notices that the similar groups with lowest accuracies are those also defined by high school educated females but who work in smaller sectors such as self-employment. Additionally, the labels for private sector high-school educated females are almost all positive. Looking at the *Feature Distribution View*, Judy sees that Private is by far the dominant value for the workclass feature. Judy therefore hypothesizes that the imbalance in workclass's distribution and imbalance in the base rates between the subgroups of workclass is causing her model's performance to vary. Judy notes these observations and aims to gather more data and try using a more expressive model to see if she can address these discrepancies.

7 Technical Implementation

FAIRVIS is a web-based system built using the open-sourced JavaScript framework *React*. Many additional libraries were used for building the system, including the following: *D3* for creating the strip plot and feature distributions; *Vega Lite* for the accuracy bar chart, label balance chart, and suggested group feature distributions; *Material.ui* for visual components and interface style. Additionally, the script for pre-processing the data and running clustering is written in Python and uses the SciKit Learn implementation of K-means.

The datasets used in the use cases are the ProPublica COMPAS dataset [24] and the UCI Adult dataset [10]. For the COMPAS dataset, the output scores were included and used directly for analysis. The original dataset ranks risk from 1-10, so we convert the task into binary classification between low (1-6) and high (7-10) risk. For the UCI Adult dataset we trained a two-layer neural network using the *PyTorch* library.

8 Limitations and Future Work

Improving and measuring the effectiveness of the subgroup generation technique. While we have found that the generated subgroups often provide useful suggestions, we have not thoroughly refined the technique. We have additionally not tested whether or not the generated groups align well with groups users find important. Collecting labeled data of datasets with outputs and important underperforming subgroups would allow us to quantify the effectiveness of our technique.

Supporting more types of problems and data. FAIRVIS currently only supports binary classification and tabular data. The current interface can be expanded to support multiclass classification, but additional visualizations views would need to be added for regression. It would additionally be nice to support some sort of graphical or textual data. The current interface works if the outputs of image classification are loaded with demographic data, but enabling the display of images could aid in auditing groups.

Scaling to millions of instances. The current implementation of FAIRVIS is able to scale to tens and hundreds of thousands of data points, but does not support even larger datasets very well. We are looking at improving the efficiency of the subgroup generation and suggestion technique to enable our system to continue to work in browser while at scale.

Suggesting and providing automatic resolutions. There are various techniques that exist to address bias in machine learning, many of which can be applied as a post-processing step to the output of a classifier. In addition, there are patterns as to what the potential reasons for bias are which could be learned by a model or codified into heuristics. We aim to implement some of the post-processing steps into FAIRVIS and add capability to highlight and suggest potential issues.

Acknowledgments

This work was supported by NSF grants IIS-1563816, CNS-1704701, and TWC-1526254, a NASA Space Technology Research Fellowship, and a Google PhD Fellowship.

References

[1] Facets - visualizations for ml datasets. `https://pair-code.github.io/facets/`, 2017. Accessed: 2019-03-31.

[2] 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. ACM, 2015.

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

[4] D. Arthur and S. Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pp. 1027–1035. Society for Industrial and Applied Mathematics, 2007.

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

[6] A. Beutel, J. Chen, T. Doshi, H. Qian, A. Woodruff, C. Luu, P. Kreitmann, J. Bischof, and E. H. Chi. Putting fairness principles into practice: Challenges, metrics, and improvements. In *AAAI/ACM Conference on Artificial Intelligence, Ethics, and Society (AIES)*, 2019.

[7] J. Buolamwini and T. Gebru. Gender shades: Intersectional accuracy disparities in commercial gender classification. In *Conference on Fairness, Accountability and Transparency*, pp. 77–91. ACM Press, New York, New York, 2018.

[8] A. Chouldechova. Fair prediction with disparate impact: A study of bias in recidivism prediction instruments. *Big data*, 5(2):153–163, 2017.

[9] Y. Chung, T. Kraska, N. Polyzotis, K. Tae, and S. E. Whang. Slice finder: Automated data slicing for model validation. In *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*, 2018.

[10] D. Dheeru and E. Karra Taniskidou. UCI machine learning repository. `http://archive.ics.uci.edu/ml`, 2017.

[11] W. Dieterich, C. Mendoza, and T. Brennan. Compas risk scales: Demonstrating accuracy equity and predictive parity. 2016.

[12] S. A. Friedler, C. Scheidegger, and S. Venkatasubramanian. On the (im) possibility of fairness. *arXiv preprint arXiv:1609.07236*, 2016.

[13] Google. What if tool. `https://pair-code.github.io/what-if-tool/`, 2018. Accessed: 2019-03-31.

[14] M. Hardt, E. Price, N. Srebro, et al. Equality of opportunity in supervised learning. In *Advances in neural information processing systems*, pp. 3315–3323, 2016.

[15] J. A. Hartigan and M. A. Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.

[16] 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*, 2018.

[17] M. Kahng, P. Y. Andrews, A. Kalro, and D. H. P. Chau. Activis: Visual exploration of industry-scale deep neural network models. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):88–97, 2018.

[18] M. Kahng, D. Fang, and D. H. P. Chau. Visual exploration of machine learning results using data cube analysis. In *Proceedings of the Workshop on Human-In-the-Loop Data Analytics*, p. 1. ACM, 2016.

[19] M. Kearns, S. Neel, A. Roth, and Z. S. Wu. Preventing fairness gerrymandering: Auditing and learning for subgroup fairness. In *International Conference on Machine Learning*, pp. 2569–2577, 2018.

[20] J. M. Kleinberg, S. Mullainathan, and M. Raghavan. Inherent trade-offs in the fair determination of risk scores. In *ITCS*, 2017.

[21] J. Krause, A. Dasgupta, J. Swartz, Y. Aphinyanaphongs, and E. Bertini. A workflow for visual diagnostics of binary classifiers using instance-level explanations. In *2017 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pp. 162–172. IEEE, 2017.

[22] M. J. Kusner, J. Loftus, C. Russell, and R. Silva. Counterfactual fairness. In *Advances in Neural Information Processing Systems*, pp. 4066–4076, 2017.

[23] K. Patel, N. Bancroft, S. M. Drucker, J. Fogarty, A. J. Ko, and J. Landay. Gestalt: integrated support for implementation and analysis in machine learning. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, pp. 37–46. ACM, 2010.

[24] ProPublica. Compas recidivism risk score data and analysis. `https://www.propublica.org/datastore/dataset/compas-recidivism-risk-score-data-and-analysis`, Mar 2019.

[25] D. Ren, S. Amershi, B. Lee, J. Suh, and J. D. Williams. Squares: Supporting interactive performance analysis for multiclass classifiers. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):61–70, 2017.

[26] H. Strobelt, S. Gehrmann, M. Behrisch, A. Perer, H. Pfister, and A. M. Rush. Seq2Seq-Vis: A Visual Debugging Tool for Sequence-to-Sequence Models. *ArXiv e-prints*, 2018.

[27] S. Wachter, B. Mittelstadt, and C. Russell. Counterfactual explanations without opening the black box: automated decisions and the gdpr. *Harvard Journal of Law & Technology*, 31(2):2018, 2017.

[28] B. Wilson, J. Hoffman, and J. Morgenstern. Predictive inequity in object detection. *arXiv preprint arXiv:1902.11097*, 2019.

[29] D. Xu, S. Yuan, L. Zhang, and X. Wu. Fairgan: Fairness-aware generative adversarial networks. In *2018 IEEE International Conference on Big Data (Big Data)*, pp. 570–575. IEEE, 2018.