

Metrics and Measurements for Algorithmic Fairness

Authors¹: mmitchellai@, dty@, mcmahan@, bbarbello@

With input from: Martin Wattenberg, Hartwig Adam, D. Sculley, Camille Francois, Brad Krueger, Alex Beutel [Draft]²


Last updated: 15.June

[go/ml-fairness-metrics](#)

//google3/learning/fairness

This document serves to describe the “ML” part of ML Fairness, functioning as an ML backbone of the [go/ml-fairness](#) project.

Table of Contents

1. Intended Audience
2. Background and Goal
3. Relevant Documents, Groups
4. Relevant Tools (referred to throughout; collected in separate document)
5. Definitions
6. Practices for Fairness in Machine Learning
7. Identifying Subgroups (separate document)
8. Metrics
 - 8.1. Discrete Output Systems (Classification)
 - 8.1.1. The Confusion Matrix
 - 8.1.2. Measuring ML systems for Bias 

Q1: Is your data collected to minimize the effects of Causes of Unfairness from Humans?

 - Problem Area: Data Collection

Q2: Are subgroups treated equally by your system?

 - Problem Area: Model and Input Data

Q3: Do you need more data or a better model?

 - Problem Area: Features and Variables

Q4: Do you need a different objective function?

 - Problem Area: Objective Function
 - 8.2. Continuous Output Systems (Scores)
 9. Testing Significance

¹ Please feel free to contribute to this document and add yourself to the list of authors if you do so.

² Originally based off of [Divya's Links Fairness doc](#).

10. Research Approaches for Fairness

11. Appendix: Evaluation Metrics, False Positives, and False Negatives

Tables

Table 1: Causes of Unfairness

Table 2: The Confusion Matrix

Table 3: Example Effects of Training Size

Audience

The intended audience for this doc is anyone who is interested in making the systems they work on more fair. This includes SWEs, PMs, policy folks, etc.

It is developed specifically in light of the machine learning fairness issues for people working with machine learning.

This document serves to describe the “ML” part of ML Fairness, helping to inform policy and preferred methods.

Background and Goal

Multiple projects in Google consider the role that “bias” plays in our technology.

The term “bias” in machine learning refers to many things. In this document, we focus on tools and metrics for algorithmic fairness.

From a machine learning perspective, there are a few ways to identify fairness:

1. Specifically for subgroups in the data ([see Subgroups doc](#)), measuring output on those subgroups using automatic evaluation metrics such as those available from the confusion matrix.
2. Slice Finding, where particularly error-prone regions of your dataset are presented.
3. Active Learning, where the system tells you the training instances that it’s struggling with the most, and you help it out.
4. Manually viewing and analyzing reported failures, reproducing those errors in data collected, and adding them to relevant evaluation subsets.
5. Thinking of possible biased failures, stress-testing them, and then hand-tweaking the model to handle those cases.

This document focuses on the first case, 1, while also calling out cases where 3, active learning, would be beneficial as well.

We break the problem down into a few parts:

1. Data (collection, annotation, processing)

2. Evaluation (what to measure and why)
3. Modeling (what is the machine learning modelling, and what is it not)
 - a. Architecture
 - b. Hyperparameters
 - c. Features
 - d. Variables
 - e. Objective Function

In all of the above, systems with unfairness in them stem in part from the following issues:³

Causes of Unfairness from “the world” that we might reflect in our projects when using some of the world’s data.	Causes of Unfairness in our procedures that we might reflect in our projects.
<ul style="list-style-type: none"> ● Implicit associations ● Implicit stereotypes ● Group Attribution error ● Out-group homogeneity bias ● Halo effect ● Stereotypical bias ● Prejudice ● Reporting Bias ● Selection Bias 	<ul style="list-style-type: none"> ● Correspondence bias ● In-group bias ● Bias blind spot ● Confirmation bias ● Subjective validation ● Experimenter’s bias ● Choice-supportive bias ● Insensitivity to sample size ● Neglect of probability ● Anecdotal fallacy ● Illusion of validity ● Automation bias

Table 1: Causes of Unfairness from Humans

The goal of this document is to help kick-start the conversation on how to we might begin to works towards systems where no subgroup of users, such as users in a [Fairness Vanguard system](#), receives disproportionately worse output/behavior from a system, or significantly worse outcomes. To do so, this document provides details of metrics relevant fairness, with links to more corresponding code and case examples increasing over time.

Why is this Important?

Google strives for algorithmic fairness across products. As we begin to craft policy around algorithmic fairness, this document starts to outline the nitty-gritty of how we can measure and promote optimally equal outputs for users -- at the level of the math, algorithms, and code.

³ Built with the help of The Cognitive Bias Codex: https://upload.wikimedia.org/wikipedia/commons/a/a4/The_Cognitive_Bias_Codex_-_180%2B_biases%2C_de_signed_by_John_Manoogian_III_%28jm3%29.png

Many checks and balances are being put in place to help provide an equitable experience across user subgroups. When those checks and balances require a deep dive into the model itself, this document provides the starting point of what to do.

Relevant Documents, Groups

Documents

1. [go/fair-not-default](#)
2. [go/links-fairness](#)
3. [go/ml-fairness-prd](#)
4. [go/algorithmic-unfairness-definition](#)
5. https://g3doc.corp.google.com/experimental/model_understanding/g3doc/tools.md
6. [Directional Awareness](#) (Kona models, Smart Reply)
7. [Algorithmic Bias Testing Playbook](#)
8. [\[Perspective\] ML Fairness](#)
9. [Equality of Opportunity in Machine Learning](#)
10. [Learning Fair Representations](#)
11. [Beyond Globally Optimal: Focused Learning for Improved Recommendations](#)
12. [Frustratingly Easy Domain Adaptation](#)

Groups

13. [go/ml-fairness](#)
14. [go/data-fairness](#)
15. [go/biasgang](#)
16. [go/fairness-vanguard](#)
17. [go/glassbox](#)
18. [go/jigsaw](#)
19. [Master I² Tracker](#)
20. [go/uhs](#)
21. [go/mlx](#)
22. [go/pair](#)

Datasets

23. [UCI Census Income Dataset](#)
24. UC Berkeley Admissions Dataset

Definitions

Algorithmic Unfairness

Algorithmic “Unfairness” at Google refers to algorithms that are unjust or prejudicial towards people. [For a detailed definition, see the Algorithmic Unfairness document](#). We approach this by grouping users into different subgroups, and working towards algorithms with output that does not disproportionately negatively affect any one subgroup.

Fair Performance

Fair performance is defined based on model performance and user experience.

- For *model performance*, the proportions of true positives, false positives, and false negatives for system prediction categories should be relatively equal across subgroups. We discuss this in the context of a confusion matrix below.
- For *user experience*, predictions over time should demonstrate that the relevance of learned user behavior increases while the relevance of user’s (known or inferred) personal attributes diminishes. (Such as race, gender, etc -- see [Static attributes definition below](#)).
- Performance is as close to equal as possible for an output category when the subgroups have roughly equivalent output for that category.
 - And subgroups may need to be discovered/re-created within the data.
 - Output should often include, for example, False Positive Rate and False Negative Rate.
- However, that may not always be possible, due to Intrinsic Hardness issues discussed later. In that case, we strive for *optimally* equal performance -- performance that is as close as can be to equal performance across subgroups.

User Subgroups

User subgroup categories may be either *pre-defined* or *discovered* in the data.

- *Pre-defined* subgroup categories are those defined manually, and include categories based on race, income, sexual preference, gender, religion, age, and political affiliation.
- *Discovered* subgroup categories are those based on the data available. This includes weakly supervised clusters that aggregate users based on similar appearance/language/interactions with the technology, self-identified categories, and fine-grained categories that don’t necessarily correspond to the pre-defined categories for broad subgroups. For example, for the broad subgroup “race”, we aim to *discover* subgroups without being bound by the U.S. Census categories
- For product impact now, we merge the two: Using broad pre-defined categories and consensually-annotated seed data to *snowball* the discovery of more and more group members.

- Moving forward in research, we are exploring fully unsupervised fairness, also called *prescient fairness* (term from Maya Gupta), where we have no annotated samples.

User Attributes

User attributes may be either *static* or *diachronic*.

- **Static attributes** are those that are relatively stable/unchanging for a user, or changing at a predictable/known rate. These include race, national origin, gender, age, sexual preference, religion, political affiliation.
- **Diachronic attributes** are that change over time, defined by the user’s behavior with the system.

Intrinsic Hardness

We define a subgroup as intrinsically hard if accuracy is not positively affected by changes in data size, model capacity, and feature adequacy.

Reporting Bias

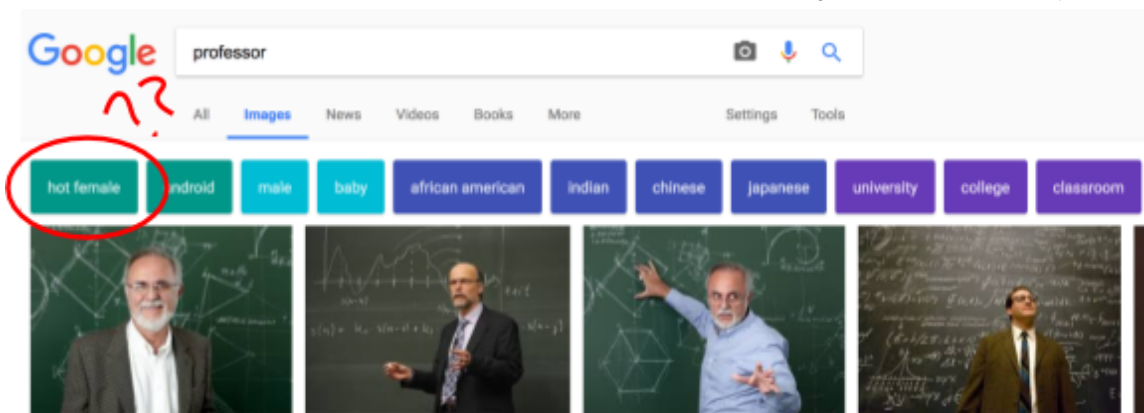
Reporting bias refers to the fact that what people talk about and share in the real world is a subset of the things that are true in the real world.

- We specifically tend to mention things that are outside of our day-to-day-norms; we do not tend to mention the things that “go without saying”.
- This can dramatically affect what our models learn from world data.
- For example, using text-based statistics, the probability of *murdering* is much higher than the probability of *exhaling*.

Word	Teraword	Knext	Word	Teraword	Knext
spoke	11,577,917	244,458	hugged	610,040	10,378
laughed	3,904,519	169,347	blinked	390,692	20,624
murdered	2,843,529	11,284	was late	368,922	31,168
inhaled	984,613	4,412	exhaled	168,985	3,490
breathed	725,034	34,912	was punctual	5,045	511

- In learning an embedding for a word like “gay”, the meaning will be overloaded with a “porn” connotation -- more so than for “straight”.
- *Note: The approach of looking up number of results could be useful for talking about unfairness examples without exposing any google products directly.*
- Doing an image search for “professor”, you will see significantly more males than in the true distribution of male professors; “hot female” professor is a common modifying phrase for “professor”, but “male” is further down (ostensibly because it is less

mentioned -- it is treated as more intrinsic, or goes-without-saying, for professor).



Practices for Fairness in Machine Learning

1. Fair data collection/annotation design
2. Fair training data and development data
3. Fair testing data
4. Adequate modeling, including architecture and hyperparameters
5. Fair training, with an objective function that is sensitive to optimally equivalent performance across user subgroups
6. Fair features
7. Fair variables
8. Fair evaluation

[Tests for 2-8 are detailed in the Steps for Examining ML Systems with Disproportional Outputs.](#)

Metrics

One way to create metrics relevant to fairness is to focus on creating similar/comparable experiences for all users.

For product, this can mean modeling each **user** as a single data point rather than each **user activity**. For example, in YouTube videos, we may want to model the users who watch YouTube as single data points, rather than modeling each video-watch as a single data point, which will favor users who watch YouTube more often.

Model Performance: Classification and Discrete Outputs

We seek to measure the differences in predictions across subgroups. To do this, we calculate a [confusion matrix](#) of predictions on each subgroup, and make adjustments based on what this tells us.

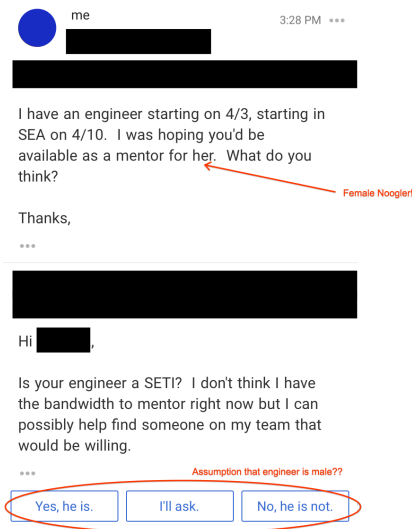
The confusion matrix is a detailed view of how a system is performing. Most groups are already using evaluation metrics that can be calculated from this, which we outline in the appendix. Most classification-based system evaluation is calculable directly from the confusion matrix.

For evaluating the fairness of a model's performance, we focus on the False Positive Rates (FPRs) and False Negative Rates (FNRs) between different subgroups. These measure whether things are being *overly predicted* for some subgroup (FPR); and whether things are being *overly left out* for some subgroup (FNR). Further extensions can be made to other evaluation metrics that are common in different products, [listed below](#). We aim to design algorithms so that different subgroups have roughly equal FPRs and roughly equal FNRs for different categories.

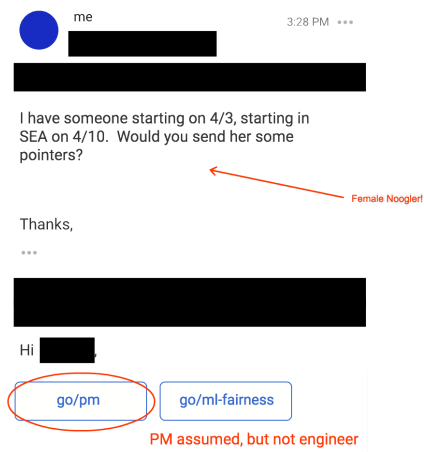
- When FPR for a given category y is high for some subgroup, the model is *overpredicting* y for that subgroup. See the [Appendix, False Positive Cases](#), for a detailed breakdown of different FPR cases. This happens for either *subgroup attributes*, when the model overpredicts an attribute is present for a subgroup, or *subgroup identity*, when the model overpredicts that a subgroup is present when it is not.

- **Examples.**

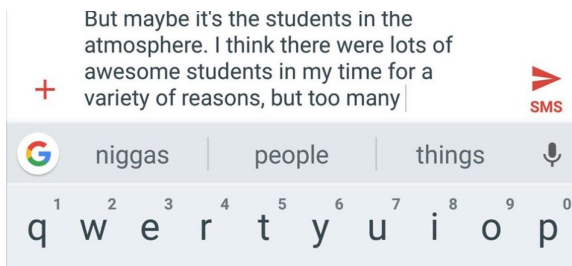
- **These demonstrate False Positives for the subgroup "female".**



Example: Subgroup Identity, high FPR. The predicted "he" pronoun in the SmartReply is an instance of the model overpredicting the subgroup *male* given the recognized category "engineer".



Example: Subgroup Attribute, high FPR. The predicted "PM" word in the SmartReply is an instance of the model overpredicting the category "PM" given the recognized subgroup *female*.



- **This image demonstrates a False Positive, potentially for a race subgroup. It was submitted to us from someone who does not use this term in their texting.**

- **When FNR for a given category *y* is high for some subgroup, the model is underpredicting *y* for that subgroup. See the [Appendix, False Negative Cases](#), for a detailed breakdown of different FPR cases.**

To calculate FNR, FPR, and related metrics, create a confusion matrix.

The Confusion Matrix

For each subgroup s , prediction category c :

		Predictions p_{sc}		Calculate
		Predictions Positive	Predictions Negative	
References r_{sc}	References Positive	The instances where the model predicts that something exists, and the reference says it does exist, are the True Positives. TP, True Positives = References Positive \cap Predictions Positive	The instances where the references say something exists, and the model does not predict it, are the False Negatives. FN, False Negatives = References Positive \cap Predictions Negative <i>Also known as Type II Error</i>	True Positive Rate/Sensitivity /Recall False Negative Rate/Miss Rate
	References Negative	The instances where the model predicts something exists, and the reference says it does not exist, are the False Positives. FP, False Positives = References Negative \cap Predictions Positive <i>Also known as a Type I error</i>	The instances where the references say something does not exist, and the model does not predict it, are the True Negatives. TN, True Negatives = References Negative \cap Predictions Negative	False Positive Rate/Fallout True Negative Rate/Specificity
Calculate		Precision/Positive Predictive Value, False Discovery Rate	Negative Predictive Value, False Omission Rate	LR+, LR-

Table 2. A Confusion Matrix. Create for each (subgroup, prediction) pair. Compare across subgroups for each prediction category.

In designing your project, make sure that you make a good decision about trade-offs between false positives/false negatives/true positives/true negatives. For example, you may want very a low false positive rate, but a high true positive rate. You may want a high precision, but a low recall is okay. Etc.

- 🙌🙌 Choose your evaluation metrics in light of these desired tradeoffs.

In what follows, we lay out a step-by-step approach for minimizing disproportional outcomes across subgroups. We refer to this as *optimally equal* values for FPR and FNR.

Steps for Examining ML Systems with Disproportional Outputs

Measure whether disproportionately poor prediction on one subgroup is a function of the data, architecture, hyperparameters, features, variables, or training procedure. Steps for this are outlined below.

At a high level, the idea is:

- (1) Collect the data well and preprocess it well
- (2) Check where there are potential problems (subgroup evaluation)
- (3) Examine the effect of data to determine whether to add more data and/or address modelling issues (model adequacy, model capacity).
- (4) If neither help, or adding more data is prohibitive, check feature and variable adequacy.
- (5) If none of this helps, consider updating your objective function.
- (6) If steps outlined in (2) through (5) do not improve performance on a subgroup, we categorize it as intrinsically hard -- a subgroup that is *more difficult* to get equivalent performance on, *all else equal*, including model design, optimal hyperparameters, and equal number of training data instances with other subgroups.

We assume that systems have a pre-defined, well-motivated evaluation metric that they are using.

- The evaluation metric should be designed so that it has the best product-specific trade-offs better true positives, true negatives, false positives, and false negatives (see the [Confusion Matrix](#)).
- For a product example, some systems may want to have *low recall* (missing a lot of stuff) in exchange for *high precision* (of the limited amount of stuff the system produces, it's all correct).
- In clinical domains, it's often preferred to have a low *false positive rate*, but a high *true positive rate* (see [Confusion Matrix](#)).
- Please see [The Appendix](#) for details on different evaluation metrics.

Step 1. Data collection and preprocessing *Addresses Fairness Question: Is your data collected to minimize the effects of Causes of Unfairness from Humans (Table 1)?*

Problem Area: Data Collection

Need to be added as its own doc.

From this step, we produce: **Training Data**, **Val-Train Data**, **Val-Test Data**, and Test Data (held out)

Usually the full dataset is split up to something like, 60% train, 10% val-train, 10% val-test, 20% test

Step 2. Subgroup Evaluation *Addresses Fairness Question: Are subgroups treated equally by your system?*

If any of these tests show unfairness across subgroups for a prediction category, move to [Step 3](#).

Many of these checks may be possible to do through go/mlx-lantern.

Global Model

- Train a full model using the **training data**.
- In the **val-test data**, for each subgroup s , for each prediction category s_c ,
- Create a **Confusion Matrix**.
 - Use the model trained on the training data to make **predictions on the val-test set**.
 - For each subgroup category s_c , pull out the predictions and the references (ground truth).
 - If this is a ranking/score system, then the category s_c is the bracket/position/tier.
 - i. Example: “Top 1”, “Top 5”, “Top 10”, “Will Get Loan”, “Will Not Get Loan”
 - ii. A common evaluation metric is Precision@K. That is, out of the top K results, what is your precision?
 - iii. Other options: FPR@K and FNR@K.
 - Example: For subgroup s based on Race, your category might be “has flowers”, and your outputs might be “True” and “False”:
 - i. The number of correct “True”s are your True Positives (TP).
 - ii. The number of correct “False”s are your True Negatives (TN).
 - iii. The number of incorrect “True”s are your False Positives (FP).
 - iv. The number of incorrect “False”s are your False Negatives (FN).
 - v. This is the [subgroup attributes binomial case](#).
 - vi. From these values, you calculate FPR and FNR.
 - Example: For subgroups s based on Race, your category might be “has X”, and your possible predictions might be “flowers”, “dishes”, and “puppies”. For the “flowers” prediction:
 - i. The number of correct “flowers” are your True Positives.
 - ii. The number of times you correctly do not predict “flowers” are your True Negatives.
 - iii. The number of incorrect “flowers” predictions are your False Positives.
 - iv. The number of times you incorrectly do not predict “flowers” when they are actually there are your False Negatives.
 - v. This is the [subgroup attributes multinomial case](#).
 - vi. From these values, you calculate FPR and FNR.
- Calculate the target task evaluation metric, the False Positive Rate (FPR) and False Negative Rate (FNR) for each subgroup s , prediction category c :
- **False Positive Rate** = $FP / (TN + FP)$
- **False Negative Rate** = $FN / (TP + FN)$
- The FPR values here ideally will be roughly equal for all subgroups.

- Similarly, the FNR values here ideally will be roughly equal for all subgroups, if they can be.
- Ditto for whatever target evaluation metric you're also working with.
- If they're not, then that is one source of system unfairness.

Subgroup-specific Models

Same as above Step, but train individual models, one for each subgroup, testing on the **val-test data**.

- *If the subgroup does not have enough instances to train an ML model (thousands of examples), this is not feasible.*
- If there is enough data, then you can check out whether a subgroup-specific model is succeeding when a global model is failing, and if so, make a move to bring in subgroup-specific models.
- If the subgroup-specific model is also doing poorly, move on to Step 3.

If any of these tests show unfairness across subgroups for a prediction category, move to [Step 3](#).

Step 3. Check Effect of Model and Data. *Addresses Fairness Question: Do you need more data or a better model?*

Retrain and test new models as you gradually increase the training data.

- For each subgroup, plot the target evaluation metric, the FNR, and FPR values on the y-axis, vs. the amount of training data (# examples) on the x-axis.
 - Create Plots For:
 - Subgroup category predictions, model trained only on data from the subgroup.
 - Subgroup category predictions, model trained on the overall data.
 - For each of the above, model initialized from the overall population model.
- Could additionally put all the subgroups on the same plot to observe the differences between subgroups.
- For these plots, overlay and include:
 - The same plot for all data
 - The same plot for all data but the subgroup



Table 3. Example outcomes from plotting the subgroup False Positive Rate vs. the full population model (rest-of). Depending on the difference between the curves, you could find evidence for *more data*, a *better model*, or *intrinsic hardness* of the problem.

Problem Area: Model Inadequacy

- There is model inadequacy if you see:
 - Evaluation on a subgroup improves (e.g., the False Positive Rate goes down) when the model is trained specifically on that subgroup, but the improvement is much weaker for that subgroup when the model is trained on the full dataset
 - The curve is not showing a good trend ([see orange line in Table 2](#))
- This can be due to a couple things relevant to the model itself:
 - Insufficient model capacity in the global model.
 - **ML Technique:** Change the model architecture to allow for increased modelling capacity.
 - Deeper or wider models
 - Check out [go/wide-n-deep](#) to leverage the benefits of both.
 - Inadequate hyperparameters: A model may have adequate capacity, but is tuned to perform well for larger subgroups that make up the bulk of the overall data.
 - [Integrate details from Beyond Globally Optimal.](#)
 - Choose hyperparameters to do well across subgroups, not just to do well on the aggregate, overall data.]
 - Use [go/vizier](#) to make selections.
- This may also be solvable with simple changes to the data:
 - **ML Technique:** Threshold/cap the number of data points for the largest subgroups.

- If your model is saturated with data from the largest subgroups, it may “overfit”, or it may be spending all its modeling power on those subgroups, at the expense of the others.
- **ML Technique:** Duplicate/augment data instances of the minority class.
 - Simply duplicating instances can help, although testing in this case should be rigorous: The concern with simply duplicating instances is that the model will “overfit”, meaning it will create strong correlations between things that are not actually correlated in the test/run-time data.
 - More details on domain adaptation in these cases would be useful to include here; not yet done.

Problem Area: Data Inadequacy

- If evaluation metric values are not levelling off for the subgroup, but actually seem to be improving as you increase training data, you now have some motivation to augment data for that subgroup. [See the red line in Table 2.](#)
 - **ML Technique:** Incrementally increase the training data for a given subgroup, plotting the changes in FNR and FPR as before until equal output performance is achieved.
 - The ideal amount of data needed can be estimated from the trend line on your plot.
 - **Not working?** If performance begins degrading for the rest of the development data, or for other subgroups, go to [Step 4.](#)
 - If data augmentation cost is prohibitive, or if doing so does not seem to improve equitability, go to [Step 4.](#)

If none of these solutions fix the problem, or if the estimated amount of data you would need is cost-prohibitive, move to [Step 4.](#)

Step 4. Check Feature and Variable Adequacy. *Addresses Fairness Question: Do you need better features or variables?*

- If one subgroup gets significantly lower evaluation scores given the same amount of training data, or performance is not improving as we scale up the amount of training data, this suggests we need new features or new variables, more data won't necessarily be enough.
 - Example: Smart Reply seems to be preferring one outcome for women, another outcome for men, and one is better than the other. In deep learning space, “gender” might not be an explicit feature -- it may have been learned implicitly -- and this is creating an unfairness issue.

Problem Area: Feature and Variable Inadequacy

- To guide the creation of new variable and features, make some of the implicit explicit:
 - Use [RankLab](#) to see what features are being given the most weight.
 - Do Feature Ablation (also possible with RankLab) to remove features and see the

- effects on down stream performance.
- **Use active learning** to find weak data points that should be labelled, bolstered.
 - In active learning, the places where the model struggles the most during training are output to humans -- to provide additional annotations, or more training data in light of the model's confusion.
- Predict implicit categories that may be at play, such as those within sensitive groups.
 - Examine correlation between implicit categories and desired categories.
 - Based on your findings, you might, e.g., preferentially select features or data that move away from these correlations; or remove data/features that enforce these correlations.
- Manually inspect content given similar representations by your system.
 - Using e.g., pairwise cosine distance between last hidden layers, where one instance produces an incorrect output and the other instance produces a correct output, and find those instances that are closest together.
- Manually inspect errors in low-confidence cases and high-confidence cases.
- As discussed above, when we learn from naturally occurring data, they will already be biased because of *reporting bias*. This means biased features, reflecting the bias in your data.
- **ML Technique:** "Debias" your features, for example, debias your embeddings.
 - See: [Man is to computer programmer as woman is to homemaker? Debiasing word embeddings](#)
 - Code being developed at [google3/learning/fairness](#).
 - Possible partners: YouTube, Rephil
- **ML Technique:** Develop new features, iterate from Steps 2-4 to measure their effect.
- **ML Technique:** Develop auxiliary tasks to predict new variables, use a multi-task learning framework, iterate Steps 2-4 to measure effect.
 - See [go/tf-multitask](#) and Lantern's support for multi-headed models.
 - For example, if your deep learning model is overpredicting LGBT content as toxic, update the model to predict "LGBT/Straight", *as well as* "positive"/"negative"/"neutral", *and* toxicity. This can be trained jointly end-to-end so as not to need too much additional compute resources.
 - If you want to get fancy, more coarse-grained tasks should be earlier in your model than the more fine-grained tasks, rather than all at last layer of the model.
- **ML Technique (less preferred):** Develop auxiliary tasks to predict new variables, use a pipeline framework, iterate Steps 2-4 to measure effect.
 - For another example, if your pipeline system is overpredicting LGBT content as toxic, first predict LGBT; use those scores as input features to a second step to

predict “positive/negative/neutral”; use those scores and the previous LGBT scores as input features to predict toxicity.

- **ML Technique:** Active learning to discover samples that should be additionally annotated.
 - In active learning, the places where the model struggles the most during training are output to humans -- to provide additional annotations, or more training data in light of the model’s confusion.

Step 5: Objective Function *Addresses Fairness Question: Do you need a different objective function?*

Problem Area: Objective Function

- This one’s tricky.
- Here’s the tl;dr: Machine learning is used to *train a model*. The model is trained by using an *objective function*. This objective function (OF) can be problematic.
- Many objective functions do some form of “maximum likelihood estimation” (MLE). Roughly, this pushes the model to have its parameters/weights make the training data the *most likely* output of the system.
 - Several areas of concern here.
 - **Overfitting:** Spurious correlations in your training data are treated as meaningful correlations, even when they are not (or are, e.g., prejudicial/harmful).
 - **Overgeneralization:** If your training data has a distribution over outcomes such as “70% A, 15% B, 10% C, 5% D”, this can get overgeneralized in your model as “80% A, 20% B” -- the less likely and minority cases can get washed out.
- This is a big move, but one option is to change the objective function.
 - Incorporate explicit fairness constraints.
 - See [Equal Opportunity](#) and [Equalized Odds](#).

If Steps 2-5 have all been tried, and still one subgroup does not show signs of improving, or continues to get significantly lower accuracy → The data for a subgroup has “Intrinsic Hardness”. ([See green line in Table 2.](#))

- Putting it another way: if you have a bunch of smart ML researchers try to train the best model they possibly can for the “harder” subgroup, and they still can’t get the same accuracy as the “easier” subgroup, then the problem is intrinsically harder. Go to [Research Approaches for Equitability](#).

For now, **ML Technique:** focus energy on modelling that subgroup in particular, including increasing data, changes to model, which may include additional predictions and constraints in a multi-task learning framework, and changes to features.

Model Performance: Continuous Outputs

Systems that output a real value rather than a hard class label require a somewhat different procedure. These are often regression-based systems, and examples include pricing, probability estimation like pCTR, risk scores, etc.

In each case, the work here applies to evaluating the final category decisions based on these scores. For example, in a system that uses scores to determine whether something is “in” or “out”, those “in” and “out” decisions are what we can evaluate fairness on, using the classification-based methods above. Precision@K is a common metric to use, meaning that out of the top K outputs from your system, what is the [precision](#) within that K? False Positive Rates and False Negative Rates, as described above, can be useful for the task of determining fairness: False Negative values get at how much the system is missing.

In further work, we will add more details about continuous outputs. For now, the recommended practice is to use the classification-based fairness evaluation, using the output category decisions from a score-based system.

User Experience

Steps towards Equitable User Experiences

User Experience Test: Measure whether any subgroups are receiving unfair information from the system.

Subgroup experiences should also be fair. Testing this can involve measuring what percentage of things that the users are exposed to are biased towards some X . On the other hand is the related issue of testing whether users within a subgroup are receiving reasonably well-informed diversity⁴, as additional system objectives are met (e.g., user engagement).

Similarly, an end-to-end ranking metric may aim to show results that provide equal opportunities, equal positive experiences, for each subgroup. This can be checked in part by examining the probabilities for different labels, based on subgroup memberships.

For evaluating the fairness of a user’s experience, we focus on measuring the effect of static user attributes on model prediction, and calculating the correlation between output predictions and users’ static attributes.

Step 1. Measure how static attribute effect much static attributes are influencing system decisions for different subgroups.

⁴ Not all diverse options should be equally encouraged for a user, hence we say “reasonably well-informed” diversity.

1. Let u = observed user behavior with the system, a = sensitive attributes, and q = a correct predicted value
2. $p(q|u,a)$ should tend towards $p(q|u,a')$ over time.
 - a. Probability of output given mutable and immutable should tend towards Probability of output given mutable and any other immutable.
3. This is [Equality of Opportunity in Machine learning](#).
4. Work towards user experiences that are agnostic to variations across sensitive attributes, unless it's an attribute that should be treated differently (e.g., accessibility, neuroatypicality).
 - a. One way to do this is to swap correlated behaviors for one subgroup into the behaviors for another subgroup; do this round-robin so that each subgroup has representations for behaviours correlated with another subgroup's behavior.
 - b. The probability of q across these swaps should remain the same.
 - c. To be tested regularly, e.g., every week.

Step 2. If a static attribute remains relevant, [ML Technique](#): fine-tune or retrain the model with the addition of the observed user behavior types as additional attributes.

Testing Significance

- **tl;dr:** Testing significance on the same data more than once requires correction to handle increased chances of finding significance with multiple tests. Methods to handle this include:
 - Bonferroni Correction ([ML Technique](#)): For your desired p -value, simply divide by the number of tests you are running. So, if your $p=.05$, and you number of tests is 10, then your Bonferroni-corrected p -value is: $.05/10 = .005$
 - Bonferroni-corrected p -value = *given p /number of tests*
 - Fisher's Combination Test
- For any test you run to be scientifically valid, you must see if the null hypothesis can be rejected. The null hypothesis simply states that the observed differences between groups are due to random chance.
 - Example Null Hypothesis: *There is no statistically significant relationship between the number of features and precision on subpopulations A and B.*
 - Example Null Hypothesis: *Model is equally good for all users.*
- From the null hypothesis, you can derive the alternative hypothesis:
 - Example Alternative Hypothesis: *This is a statistically significant relationship between the number of features and precision on subpopulations A and B.*
 - Example Alternative Hypothesis: *Model is not equally good for all users.*
- When you test significance, you are essentially asking "how likely is it that these observations are due to random chance?".
- However, the more you test significance, the more the chances of finding significance increases. This is also known as [p-hacking](#), and has been written about extensively. Use the Bonferroni Correction or similar to correct for this.

- **Monte Carlo Simulation:** Look at different slices/views of data and measure there. Under the assumption of equitability/fairness across predictions, we would find that accuracy is roughly the same across different simulations.
- Once we discern that something is not due to random chance, turn to: Is this because that subsample is intrinsically hard, or extrinsically hard?

Research Approaches for Fairness

Ordered so that more straightforward work is at top, more speculative work closer to bottom.

Latent Attributes

ML Technique: Train models to predict attributes for each labeled prediction, then explore the attribute predictions on the valtest set to dig into what assumptions the model is implicitly making, and how this differs across groups. Adjust model to explicitly model and address these attributes when making predictions.

- This can be done in a multi-task learning framework, where several things are predicted, including the target category and a relevant subgroup.
- For example, given a model that predicts “professor”, fine-tune it to predict subgroups that are relevant to fairness groups, such as “gender”, by predicting the fine-grained subgroups of, e.g., “male” and “female”.
 - If “professor” and “male” predictions align, or the average of the last hidden layers for each are close together in vector space, then you know your model is biased in that direction.

Equalized Odds

[\[Modified from Equal Opportunity paper\]](#)

Based on protected attributes. We say that a predictor \hat{Y} satisfies *equalized odds* with respect to protected attribute A and outcome if \hat{Y} and A are independent conditional on Y .

Unlike demographic parity, equalized odds allows \hat{Y} to depend on A but only through the target variable Y . As such, the definition encourages the use of features that allow to directly predict Y , but prohibits abusing A as a proxy for Y .

As stated, equalized odds applies to targets and protected attributes taking values in any space, including binary, multi-class, continuous or structured settings. The case of binary random variables Y , \hat{Y} and A is of central importance in many applications, encompassing the main conceptual and technical challenges. As a result, we focus most of our attention on this case, in which case equalized odds are equivalent to:

$$P(\hat{Y} = 1 | A = 0, Y = y) = P(\hat{Y} = 1 | A = 1, Y = y), y \in \{0, 1\}$$

For the outcome $y = 1$, the constraint requires that \hat{Y} has equal *true positive rates* across the two demographics $A = 0$ and $A = 1$. For $y = 0$, the constraint equalizes *false positive rates*. The definition aligns nicely with the central goal of building highly accurate classifiers, since $\hat{Y} = Y$ is always an acceptable solution. Equalized odds enforces that the accuracy is equally high in all demographics, punishing models that perform well only on the majority.

Equal Opportunity

Based on protected attributes. A relaxation of equalized odds, requiring non-discrimination only within the “advantaged” outcome group; say when $Y = 1$. This leads to a relaxation of our notion that we call “equal opportunity”.

We say that a binary predictor \hat{Y} satisfies *equal opportunity* with respect to A and Y if

$$P(\hat{Y} = 1 | A = 0, Y = 1) = P(\hat{Y} = 1 | A = 1, Y = 1).$$

Equal opportunity is a weaker notion of non-discrimination, and thus typically allows for stronger utility (see [case study](#) below).

A score R satisfies equalized odds if R is independent of A given Y . If a score obeys equalized odds, then any thresholding $\hat{Y} = I\{R > t\}$ of it also obeys equalized odds (as does any other predictor derived from R alone).

Entropy Measures

A number of algorithms leveraging entropy (e.g., cross-entropy error, perplexity, surprisal, KL divergence) can be used to stress-test how well different trained models are fit to the training data in each group by comparing these metrics across different batches within a group.

Variance

Looking at within-group variance, as well as entropy differences within different subgroups, can help to define which groups might benefit from having some instances move to another cluster/group membership, or new cluster/group membership

Reporting Bias

[[Adapted from Reporting Bias paper](#)]

- Train models to predict latent (sparsely labelled) attributes that may or may not be observed/annotated/mentioned.
 - Examples: *sexual, emotional, violent*
- A human-biased prediction h can be factored as:

- **Presence v – Is the object present?**
- **Relevance r – Is the object relevant for a human?**

How *relevant* is the concept, given presence status? Is concept *present*?

$$h(y|I) = \sum_{j \in \{0,1\}} r(y|z = j, I)v(z = j|I)$$

This builds up the “latent” model of whether the concept is present or not -- even if it’s not explicitly mentioned/annotated.

Prediction Bias

Compute and compare prediction bias for different subgroups. It is reasonable to compute and compare prediction bias $E[H(X) | A] - E[Y | A]$ across suitably large populations A .

Area Between F*R Rates

A model tested on a particular subgroup, vs. a model tested on the rest of the data with that subgroup removed, will show different rates of change as the amount of training data increases.

Target-subgroup-only Models vs. Full Dataset, False Positive Rate as Training Data Increases

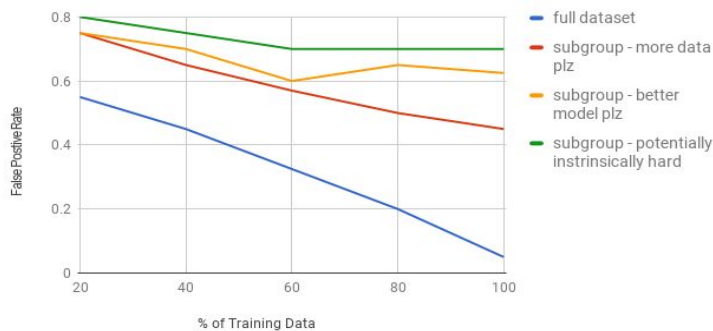


Table 2. Example different rates of improvements in subgroup predictions (target-subgroup) vs. predictions for the full dataset.

The difference between the slope of the *rest-of* predictions and the *target-subgroup* predictions provides a direct measure of the equitability of data increase.

GANs to Augment Training data

Use generative adversarial network training with minibatch discrimination (to improve sample diversity) within each subgroup. As in regular GAN training, the generator creates additional faces for each batch within a cluster, and the discriminator tries to distinguish whether the faces have been artificially generated, or belong to the cluster. Once the model is done training, we can use the network to then generate new training instances to subgroups.

Note that it is difficult to measure a classifier's performance when the evaluation data is very small, since a small amount of evaluation data is not necessarily going to be a representative sample of the population. The approach would be to hold out as much testing data as possible (of real data), while training on the synthetic GAN data.

CycleGan

Models like CycleGAN can literally turn apples into oranges. We could use something like CycleGAN to turn records from one subgroup into their equivalent from another subgroup. This would allow you to ask questions like "would this person have been granted parole if they belonged to a different race?" etc.

Appendix

Common evaluation metrics for classification-based systems

- **False Positive Rate, $FPR = FP/(TN+FP)$**
 - "Fall Out"
 - "Probability of False Alarm"
 - This metric measures how often the system makes predictions that should not be made.
- **False Negative Rate, $FNR = FN/(TP+FN)$**
 - "Miss Rate"
 - This metric measures how often the system misses predictions that should be made.
- **Precision = $TP/(TP+FP)$**
 - This metric measures that, for all the predictions made, how correct they are.
- **Recall, Sensitivity, True Positive Rate = $TP/(TP+FN)$**
 - This metric measure that, for all the predictions made, how much is being left out.
- **Specificity = $TN/(TN+FP)$**
 - This metric measures that, for all predictions made, how much is correctly left out.
- **Mean Average Precision (mAP)**
 - This metric measures the mean of the average precision scores for each query.
- **F-score (F1) = $2 * (Precision * Recall) / (Precision + Recall)$**
 - This metric measures the harmonic mean of precision and recall, with

- both equally weighted.
 - Other harmonic mean F-scores could weight them differently, e.g., prioritizing precision by not recall.
- Receiver Operating Characteristic (ROC) Curve
 - Graphical plot for False Positive Rate and True Positive Rate
- Area Under the ROC Curve (AUC)
 - This metric measures the area under the ROC curve -- is used as a measure of accuracy.

False Positive Cases

When FPR for a given category y is high for some subgroup, the model is *overpredicting* y for that subgroup. This happens for either *subgroup attributes*, when the model overpredicts an attribute is present for a subgroup, or *subgroup identity*, when the model overpredicts that a subgroup is present when it is not.

Case 1	Case 2	Case 3	Case 4
Subgroup Attributes, Binomial	Subgroup Attributes, Multinomial	Subgroup Identity, Binomial	Subgroup Identity, Multinomial
The model incorrectly guesses that an attribute of a subgroup is present when it is not.		The model incorrectly guesses that a particular subgroup is present when it is not.	
Examples			
<i>For instance of subgroup_a, model incorrectly guesses that the instance is "toxic" (when it is not).</i>		<i>For an instance of subgroup_b, model incorrectly guesses that subgroup_a is present (when it is not).</i>	
category: is_toxic values: True, False ref: is_toxic=False y: is_toxic=True	category: is values: toxic, awesome, so-so ref: is=awesome y: is=toxic	category: sub_a values: True, False ref: sub_a=False y: sub_a=True	category: subgroup_id values: sub_a, sub_b, sub_c ref: subgroup_id=sub_b y: subgroup_id=sub_a
<i>For instance of subgroup_a, category "holding"</i>	<i>For instance of subgroup_a, category "holding",</i>		

<i>flowers</i> ", model incorrectly guesses that they are holding flowers when they are not.	model incorrectly guesses that the value is "flowers" when it is not.	
category: holding_flowers values: True, False ref: holding_flowers=False y: holding_flowers=True	category: holding values: flowers, puppies, plates ref: holding=puppies y: holding=flowers	

False Negative Cases

When FNR for a given prediction category *y* is high for some subgroup, the model is *underpredicting y* for that subgroup.

Case 1	Case 2	Case 3	Case 4
Subgroup Attributes, Binomial	Subgroup Attributes, Multinomial	Subgroup Identity, Binomial	Subgroup Identity, Multinomial
The model regularly incorrectly guesses that an attribute of a subgroup is not present when it is.		The model regularly incorrectly guesses that a subgroup is not present when it is.	
Examples			
For an instance of subgroup_a, the model incorrectly guesses that people from a particular subgroup are not <i>holding flowers</i> when they are.		For an instance of subgroup_a, the model incorrectly misses that subgroup_a is present.	
category: holding_flowers values: True, False ref: holding_flowers=True y: holding_flowers=False	category: holding values: flowers, puppies, plates ref: holding=flowers y: holding=puppies	category: subgroup_a values: True, False ref: subgroup_a=True y: subgroup_a=False	category: subgroup_id values: subgroup_a, subgroup_b, subgroup_c ref: subgroup_id=subgroup_a y:

			<i>subgroup_id=subgroup_b</i>
--	--	--	--------------------------------------

