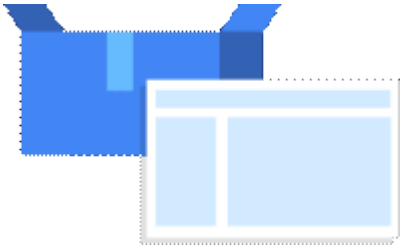
Google Inclusive ML

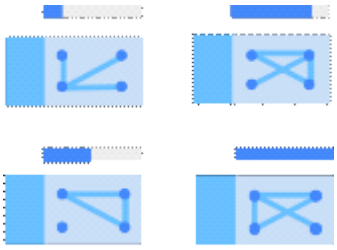
Sunday, October 4, 2020 9:55 PM

Clipped from: https://cloud.google.com/inclusive-ml/

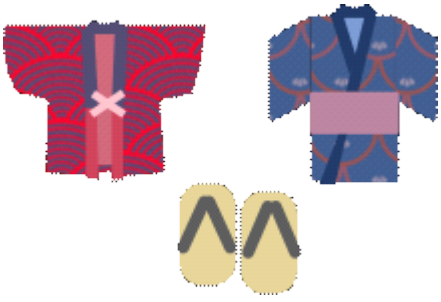
At Google, we've been thinking hard about the principles that motivate and shape our work in artificial intelligence (AI). We're committed to a human-centered approach that foregrounds responsible AI practices and products that work well for all people and contexts. These values of responsible and inclusive AI are at the core of the AutoML suite of machine learning products, and manifest in the following ways.



AutoML expands the kinds of organizations and individuals who can make AI work for them by offering an easy-to-use, codeless user experience that requires no prior machine learning experience.



Using algorithmic techniques such as transfer learning and Learning to Learn, AutoML lowers the bar to entry by enabling organizations to build custom models with smaller datasets than typically would be required.



AutoML gives you the ability to easily produce meaningful, contextually relevant ML systems. For instance, if you see that our generic model doesn't capture slang or language in your domain, you can create a custom model that includes the linguistic features you care about. If you find that generic clothing classification models don't work for the clothing worn by your community, you can train a model that does a better job.

As part of our mission to bring the benefits of machine learning to everyone, we care deeply about mitigating pre-existing biases around societal categories that structure and impact all our lives. At Google, this area of research is called Machine Learning Fairness. In this page, we share our current thinking on this topic and our recommendations for how to apply AutoML to conversations with regard to fairness in machine learning.

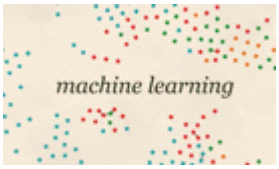
What is fairness in machine learning?

Fairness in machine learning is an exciting and vibrant area of research and discussion among academics, practitioners, and the broader public. The goal is to understand and prevent unjust or prejudicial treatment of people related to race, income, sexual orientation, religion, gender, and other characteristics historically associated with

discrimination and marginalization, when and where they manifest in algorithmic systems or algorithmically aided decision-making.

These algorithmic challenges emerge in a variety of ways, including societal bias embedded in training datasets, decisions made during the development of an ML system, and through complex feedback loops that arise when an ML system is deployed in the real world.

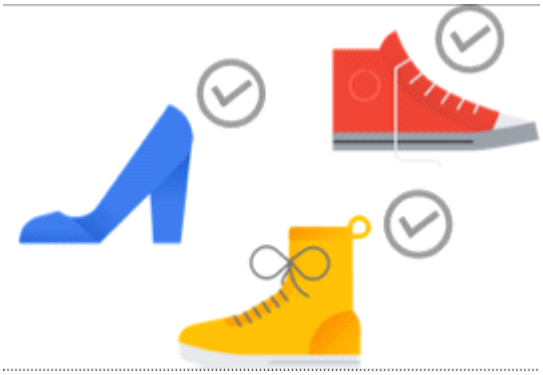
In pursuing fairness in machine learning, we see a diversity of valid perspectives and goals. For instance, we might train ML classifiers to predict equally well across all social groups. Or, informed by research on the impact of historical inequities, we might aim to design ML systems that try to correct or mitigate adverse outcomes going forward. These and many other approaches are important and often interrelated.



video

Watch “Machine learning and human bias”

For further information, see Google's Responsible AI Practices and Recommended Fairness Practices, Google video on machine learning and human bias, and Moritz Hardt and Solon Barocas' "Fairness in ML Tutorial."



Fairness in ML & AutoML

In AutoML, we have an opportunity to promote inclusion and fairness in different ways. As noted earlier, if the machine learning models you can

access today don't fully address the needs of your community or users, due to historical absences or misrepresentation in data, you can create custom models that do a better job. In any custom model you create using AutoML, you can pursue fairness goals by including data that helps the model predict equally well across all categories relevant to your use case. These fairness-related actions may help mitigate the risk of the following types of negative consequences associated with some ML systems.

Representational harm

This type of harm occurs when an ML system amplifies or reflects negative stereotypes about particular groups. For example, ML models generating image search results or automated text suggestions are often trained on previous user behavior (e.g. common search terms or comments), which can lead to offensive results. And along with offending individual users in the moment, this kind of representational harm also has diffuse and long-term societal effects on large groups of people.

Opportunity denial

Machine learning systems are increasingly used to make predictions and decisions that have real-life consequences and lasting impacts on individuals' access to opportunities, resources, and overall quality of life.

Disproportionate product failure

In some cases, unfairness is a matter of basic usability and access. For example, some soap dispensers used in public restrooms have been shown to have disproportionately high failure rates for individuals with darker skin tones.

Now let's discuss steps you can take to promote fairness as you build your custom models in AutoML and use them in your ML systems. We'll focus on mitigating bias in training datasets, evaluating your custom models for disparities in performance, and things to consider as you use your custom model.

What are some first steps in assessing your use case for fairness in machine learning?

Consider your product's context and use.

In some cases, as described above,fairness is a matter of basic usability and access.

In other cases, fairness intersects with laws and regulations that restrict the use of data that directly identifies or is highly correlated with sensitive characteristics, even if that data would be statistically relevant. People with certain of those characteristics may also be legally protected against discrimination in some contexts (e.g., "protected classes").

In yet other cases, unfairness isn't immediately obvious, but requires asking nuanced social, political and ethical questions about how your ML system might be used in practice or how it may allow bias to creep in over time. For example, if you're using AI to generate automated text or translations, it's important to consider what types of bias or stereotypes might be ethically problematic (e.g. associating gender with job types, or religion with political views)

When you're getting started building your own ML system, review discrimination-related regulations in both your region and the locations your application will serve, as well as existing research or product information in your domain to learn about common fairness issues.

Consider the following key questions.

Here are some more key questions that are worth asking. If you answer "yes" to any of them, you might want to consider conducting a more thorough analysis of your use case for potential bias-related issues.

Does your use case or product specifically use any of the following data: biometrics, race, skin color, religion, sexual orientation, socioeconomic status, income, country, location, health, language, or dialect?

Does your use case or product use data that's likely to be highly correlated with any of the personal characteristics that are listed above (for example, zip code and other geospatial data are often correlated with socioeconomic status and/or income; similarly, image/video data can reveal information about race, gender, and age)?

Could your use case or product negatively impact individuals' economic or other important life opportunities?

Now let's look at ways you can work to promote

fairness in machine learning as you move through the different steps in the AutoML workflow.

Data Guidelines

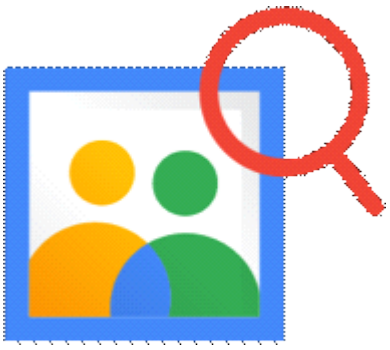
Let's start with the first step in AutoML: putting together your training data. While no training data will be perfectly "unbiased," you can greatly improve your chances of building a better, more inclusive product if you carefully consider potential sources of bias in your data and take steps to address them.

What kind of biases can exist in data?

Biased data distribution

This occurs when your training data isn't truly representative of the population that your product seeks to serve. Think carefully about how your data was collected. For example, if you have a dataset of user-submitted photos and you filter it for image clarity, this could skew your data by over-representing users who have expensive cameras. In general, consider how your data is distributed with respect to the groups of users your product will serve. Do you have enough data for each relevant group? There are often subtle, systemic reasons why your dataset might not capture the full diversity of your use case in the real world.

To mitigate this, you could try to acquire data from multiple sources, or filter data carefully to ensure you only take the most useful examples from overrepresented groups.



Biased data representation

It's possible that you have an appropriate amount of data for every demographic group you can think of, but that some groups are represented less positively than others. Consider a dataset of microblog posts about actors. It might be the case that you did a great job collecting a 50-50 split of male and female performers, but that when you dig into the content, you find that posts about female performers tend to be more negative than those about male performers. This could lead your model to learn some form of gender bias.

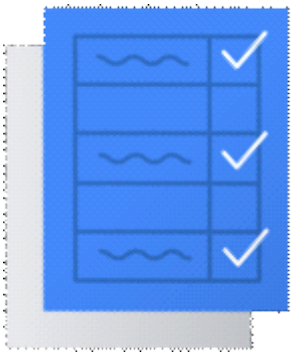
For some applications, different representations between groups might not be a problem. In medical classification, for instance, it's important to capture subtle demographic differences to make more accurate diagnoses. For other applications, however, biased negative associations may have financial or educational repercussions, limit economic opportunity, and cause emotional and mental anguish.

Consider hand-reviewing your data for these negative associations if it's feasible, or applying rule-based filters to remove negative representations if you think it's right for your application.

Proxy variables

It's easy to think that once you've removed variables that encode protected demographic information, your model must be free of bias. But many variables are strongly correlated with demographics, including location, education level, and income, to name just a few. If you have access to demographic information about your data, it's always a good idea to analyze your results based on that information to make sure your model treats

different groups equitably.



Biased labels

An essential step in creating training data for AutoML is labeling your data with relevant categories. Minimizing bias in these labels is just as important as ensuring that your data is representative. Understand who your labelers are. Where are they located? What languages do they speak natively? What ages and genders are they? Homogeneous rater pools can yield labels that are incorrect or skewed in ways that might not be immediately obvious.

Ideally, make sure your labelers are experts in your domain or give instructions to train them on relevant aspects, and have a secondary review process in place to spot-check label quality. The more complicated the data is to label, the harder you should work to make sure your labelers understand their job; drawing bounding boxes and labeling text entities might not be intuitive to everyone, so be sure to break down every task and anticipate common questions. Aim to optimize for objectivity over subjectivity in decision-making. Training labelers on "unconscious bias" has also been shown to help improve the quality of labels with respect to diversity goals. And, allowing labelers to self-report issues and ask clarifying questions about instructions can also help minimize bias in the labeling process.

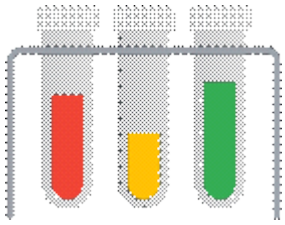
Tip: If you're using the human labeling service in AutoML, consider the following guidelines as you write your instructions.

Make sure your labeling instructions and training materials include detailed, specific context about your use case, description of your end-users, and illustrative examples that help labelers keep the diversity of your user base in mind.

Review any comments you receive from raters in order to identify areas of confusion, and pay close attention to any sensitive categories as you spot check, approve, and reject the data labels you receive back.

Once your dataset is ready, consider specifying the test/train split

In the ML Beginner's guides (Vision, Natural Language, Translation, Video Intelligence, Tables), we discussed how your dataset is divided in the machine learning process. As noted, in AutoML, you can either have Google automatically split your dataset or manually specify the test/train split. If your use case warrants, you might want to consider the second option.



While you're splitting your data manually, consider the guidance we've covered so far in order to create diverse and inclusive test sets. If you use all your best inclusive data for training, you could lose out at test time if you get an overly rosy picture of your model's performance on underrepresented subgroups. If you have scarce data about a particular subgroup, performing the train/test split yourself will help ensure that your data is spread representatively between your training and test sets. In some AutoML products, such as AutoML Tables, you can also try specifying custom weights for rare types of data to give them more significance in the training process.

Review your training data

- Do all your categories have the recommended number of data items? Do your categories and images/videos/text represent the diversity of your user base? Is the distribution approximately equal across classes? Does your training data (images, videos, text, sentence pairs) match the type of data you want your model to make predictions on?

Evaluate: Assess your model's performance

Evaluating your model for fairness requires thinking deeply about your particular use case and what impact your model could have on your end users when it gets things wrong. This means understanding the impact of different types of errors for different user groups. This is where it's important to think about potential fairness issues. For example, do model errors affect all users equally, or are they more harmful to certain user groups?

Once you've thought this issue through, you'll be better able to decide what performance metrics it makes sense to optimize for (for instance, precision vs. recall), evaluate trade-offs between them, and to examine examples of errors to check for bias.

Use case: Passport photo evaluation

Let's say you want to create a tool to help people edit and print passport photos. Each country has its own rules about photo dimensions, framing, acceptable background colors, acceptable facial expressions, and other things that may or may not be in the picture. You want to alert people before they send in a passport application that their photo may not be acceptable.

False positive:

A false positive in this case would be when the system marks a photo as unacceptable when in fact the country's passport authority would have

accepted it. No big deal — the retake is even more likely to be usable.

False negative:

A false negative in this case would be a failure to detect an unusable picture. The customer goes to the expense of printing a photo and submitting an application, only to have it rejected. Worst case, they miss a planned trip because they couldn't get a passport in time.

Fairness considerations: In this case, it would be important to check whether the model produces false negatives more frequently for certain groups of people, for example based on race or gender. You can do this in AutoML by examining individual false negatives to check for problematic patterns.

Optimize for: In this case, you would likely want to optimize for Recall. This aims to reduce the number of false negatives, which in this scenario are the more problematic errors.

Use case: Kids' content filter

Suppose you're building a reading app for kids and want to include a digital library of age-appropriate books. You want to design a text classifier that selects children's books from a database of adult and children's books based on each book's title and description.

False positive:

A false positive in this case would be an adult book that is incorrectly classified as a children's book and therefore gets added to the kids' reading app, thus potentially exposing children to age-inappropriate

content. Parents would be very upset and likely delete the app.

False negative:

A false negative in this case would be a children's book that gets flagged incorrectly as an adult book and is therefore excluded from the in-app library. Depending on the book, this could be a minor inconvenience (e.g. excluding an obscure sequel of an unpopular series) or much more problematic — for example, if the children's book includes content that is considered controversial by some people but is also generally accepted as having clear educational or social value.

Fairness considerations: While at first glance this may seem like a simple case, it highlights some of the complexities of evaluating use cases for fairness. On the one hand, there's a clear need to avoid false positives (minimizing the likelihood that children are exposed to age-inappropriate content). On the other hand, false negatives can also be harmful. For example, if the text classifier tends to flag children's books with LGBTQ themes (for instance, stories about children with two parents of the same gender) as inappropriate, this is problematic. Similarly, if books about certain cultures or locations are excluded more commonly than others, this is equally concerning.

Optimize for: In this case, you would likely want to optimize for Precision. Your app will only surface a small fraction of all the children's books in the world, so you can afford to be picky about which ones to show to users. However, you'd also want to consider UX solutions for how to surface books that might require a parent's input. For example, you could add a feature which recommends that parents read a book with children, so they can talk through the issues the book raises.

Use case: Survey distribution

Suppose you're working on distributing a survey and want to build a model to choose participants who are most likely to reply. You aren't allowed to consider income as a factor for participant choices, but your data has an "Income" column. In AutoML Tables, you remove the "Income" column from training. But when you slice the data by income to check that it didn't affect the results, you discover that your model didn't choose evenly across income buckets. How did this happen?

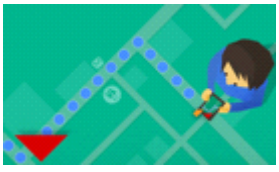
Proxy variables: Although you removed the "Income" column from consideration, your data might still include many other variables that provide clues about the income of the individuals in your dataset. Do you have their zip code, level of education, or even age? Any of these variables could be correlated with income. When you want to make sure your model is choosing a sample that cuts equally across all demographic slices, look carefully at AutoML Tables' "Analyze" tab to check for correlations. And make sure to evaluate your model carefully for biases before you use it in production.

Research

Google Research blog post on attacking discrimination with smarter machine learning

Developer blog

Google Developers blog post on measuring bias in text embeddings

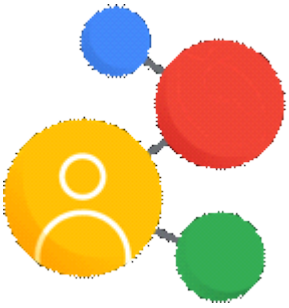


Article

[Google Design blog post on human-centered machine learning]

Predict: Smoke test your model

Once you've evaluated your model's performance on fairness using the machine learning metrics in AutoML, you can try out your custom model with new content in the Predict tab. While doing so, consider the following fairness recommendations:



Think carefully about your problem domain and its potential for unfairness and bias. You know your area best. Is your image or video classifier likely to be affected by the races or genders of people in the content? Is your text classifier likely to be sensitive to terms that refer to demographic groups? Does the language pair for which you're building a translator have cultural differences that may be highlighted, or a mismatched set of pronouns that might end up exposing an underlying societal bias? Come up with cases that would adversely impact your users if they were found in production, and test those on the Predict page or in your own unit tests.

Remember that your users could be negatively impacted not just by offensive or unfair predictions but by the absence of a clear prediction (false negatives), as well. If you find that your results aren't aligned with the experience you'd like to create for all of your end-users, you can further debias your dataset by adding more data to relevant classes, or use your model in a way that corrects for any issues you've found.

Use: Your model in production

Implement simple fixes. If your model isn't perfect, retraining it with new data isn't the only answer. Sometimes a simple pre- or post-processing step to remove certain words or types of images can be an effective solution.

Adjust the score thresholds of your model to find an acceptably 'fair' balance between precision and recall, given your understanding of how different error types impact your users.

Once your model is built and serving predictions, your data distribution may change subtly over time, and your model may no longer reflect the relevant contexts of your application. Be sure to monitor your model's performance over time to ensure that it's doing as well as you expect, and collect feedback from your users to identify potential issues that may require new data and retraining.

Sometimes corner cases emerge that you just didn't think about. Come up with an incident response plan if you're concerned that your model may misbehave in a manner that may adversely impact your users and your business.

Feedback

This is a living document and we're learning as we go. We'd love your feedback on the guidance we've given here. Send an email to inclusive-ml-feedback@google.com to tell us about your

experience in creating your custom models, what worked, and what didn't. We look forward to your feedback!