AI Fairness & Bias

Welcome to AI Fairness & Bias.

01 Overview of Fairness and Bias

02 Identify Bias – TFDV Tool

03 Identify Bias – What-if Tool

04 Identify Bias – TFMA Tool

05 Mitigate Bias – Data Intervention

06 Mitigate Bias – Threshold Calibration

07 Mitigate Bias – Model Remediation

08 Lab: Mitigate Bias with MinDiff in TensorFlow

This module consists of eight lessons.

In this module, you learn to ...

01	Define types of unfair bias in AI
02	Discuss why AI fairness is important for machine learning and difficult to achieve
03	Identify possible causes of biases in AI systems
04	Recognize AI fairness best practices
05	Explore tools and techniques to mitigate bias in datasets and models.

Today you will learn to...

- Define types of unfair bias in AI
- Discuss why AI fairness is important for machine learning and difficult to achieve
- Identify possible causes of biases in AI systems
- Recognize AI fairness best practices
- Explore tools and techniques to mitigate bias in datasets and models.

01 Overview of Fairness and Bias

02 Identify Bias – TFDV Tool

03 Identify Bias – What-if Tool

04 Identify Bias – TFMA Tool

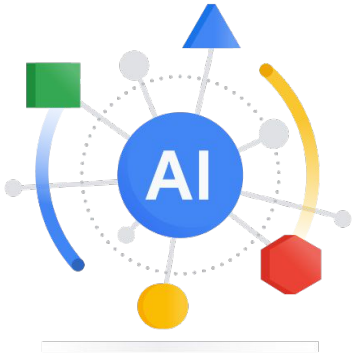
05 Mitigate Bias – Data Intervention

06 Mitigate Bias – Threshold Calibration

07 Mitigate Bias – Model Remediation

08 Lab: Mitigate Bias with MinDiff in TensorFlow

Let’s start with an overview of fairness and bias.



1. Be socially beneficial.
2. Avoid creating or reinforcing unfair bias.
3. Be built and tested for safety.
4. Be accountable to people.
5. Incorporate privacy design principles.
6. Uphold high standards of scientific excellence.
7. Be made available for users that accord with these principles.

Fairness relates to the second of Google's AI principles: "Avoid creating or reinforcing unfair bias;" although it's worth noting that the principles often intersect.

Let's first define what is bias.

What is bias?

Stereotyping, prejudice or favoritism towards some things, people, or groups over others.

Definition from
https://developers.google.com/machine-learning/glossary/fairness

Bias is stereotyping, showing prejudice, or favoritism towards some things, people, or groups over others. It is easy to say, but usually challenging to realize the biases we have, because they are hidden inside our own perspectives.

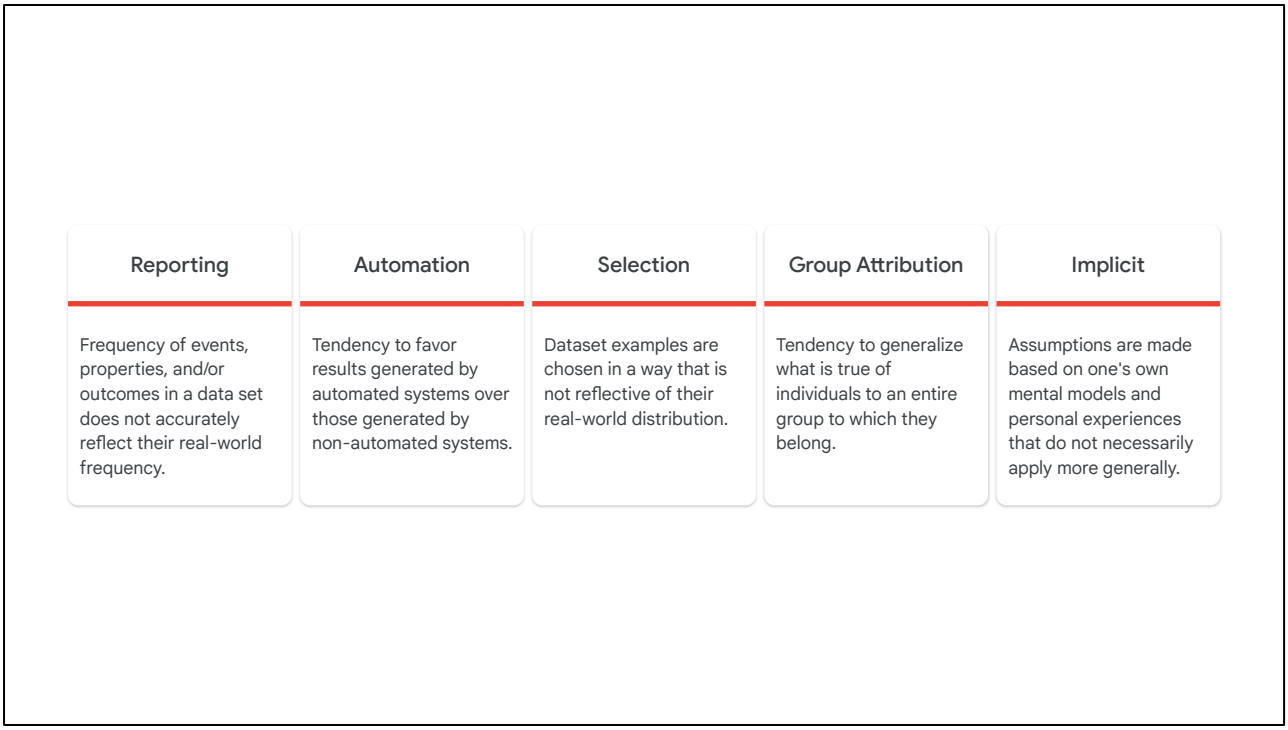
Picture... a shoe

Let’s do a quick experimentation. Picture a shoe.
What kind of shoe did you picture?

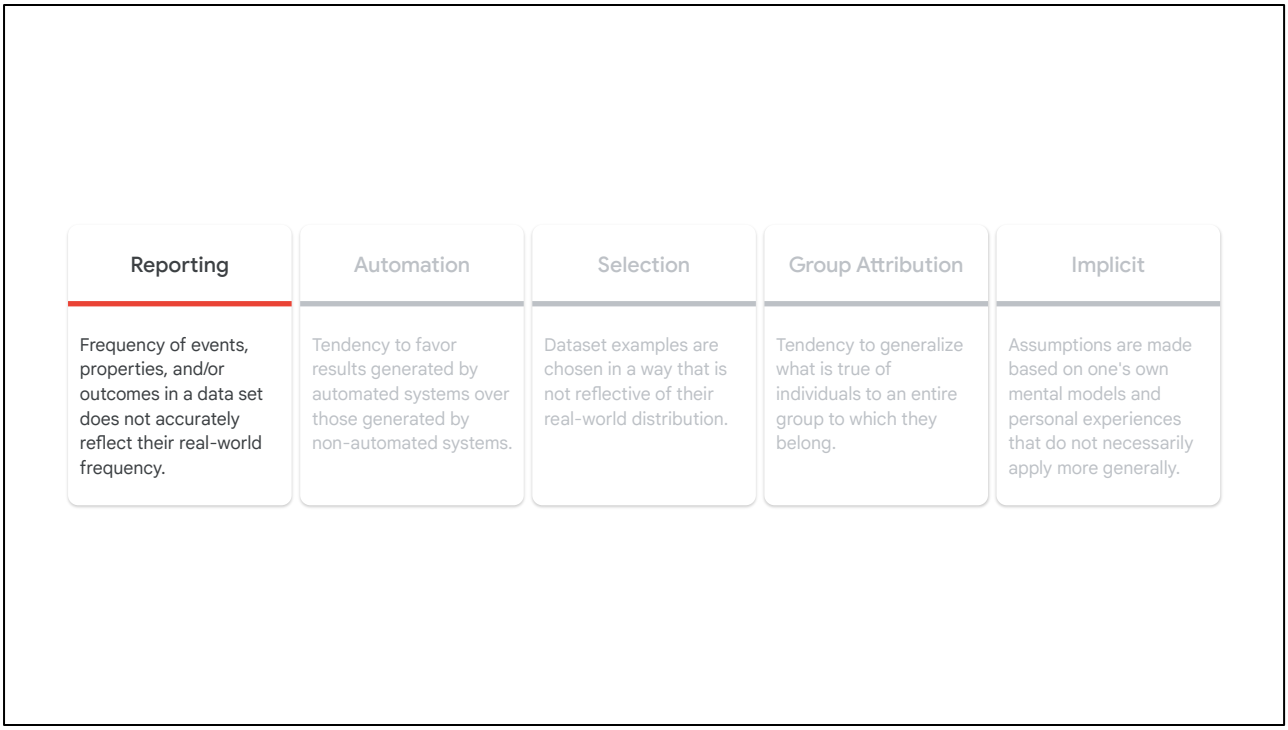
Here are some examples of a shoe. Which one is closest to what you visualized? Something like a sneaker on the left? Or a high-heel on the right?

Although not all biases are harmful, it is important to note that we naturally obtain some tendencies in perceptions, thought processes, and common senses, depending on a group you belong to, or information you get.

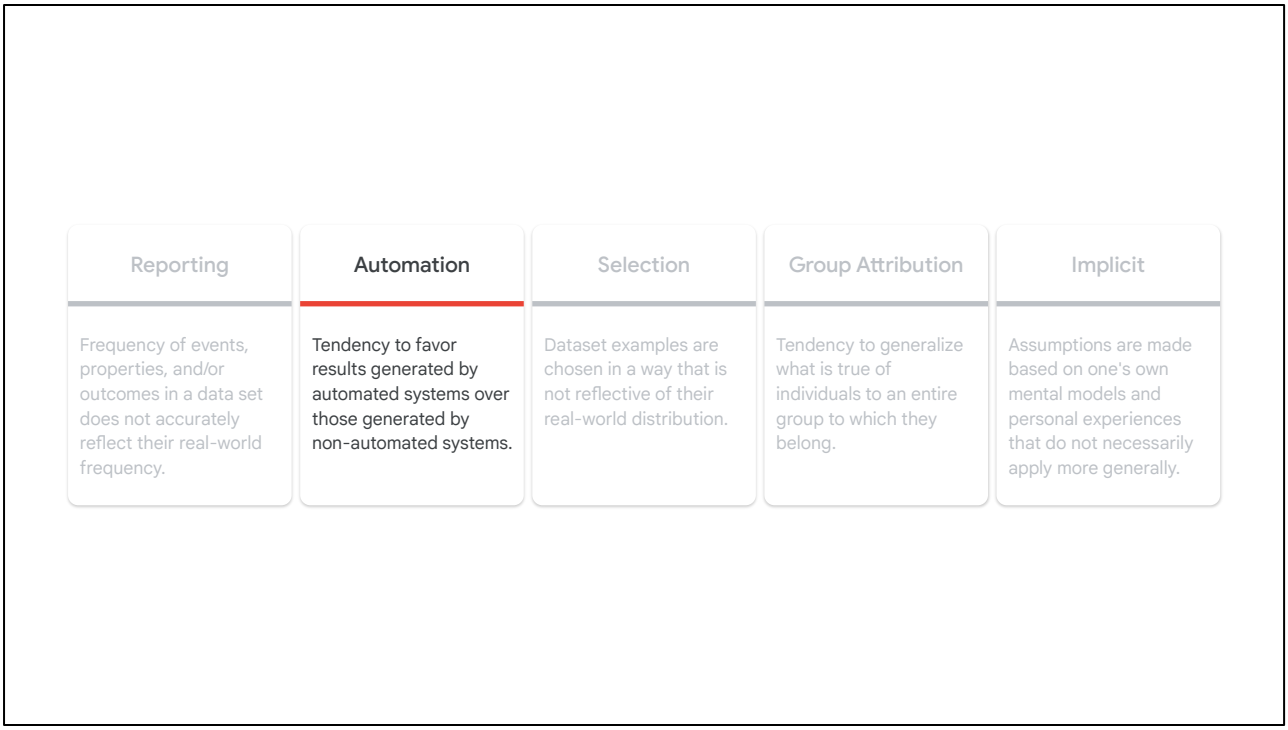
What if you are asked to visualize “a scientist”? Did you envision a particular gender group or racial group?



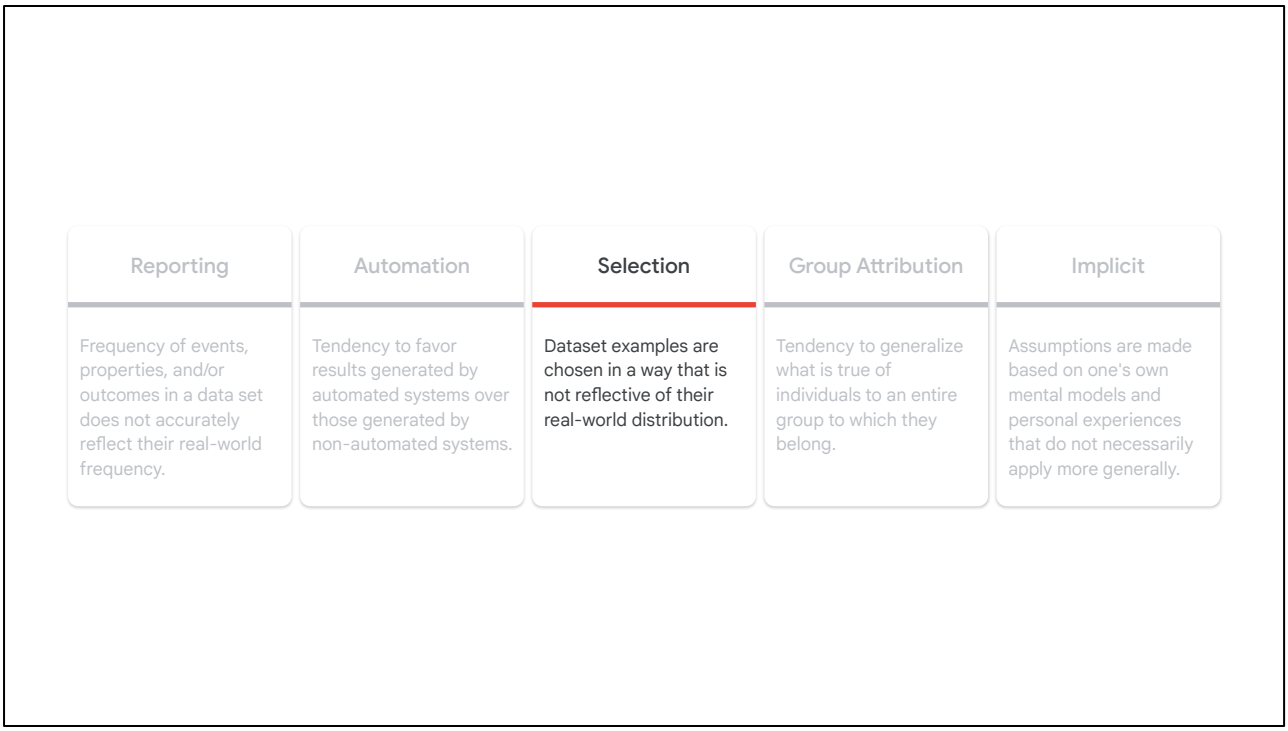
There are over 100 different types of human biases listed in Wikipedia’s catalog of cognitive biases.
When developing products for AI systems, it's important to be aware of common human biases that can manifest in your data and in your model. This way, you can take proactive steps to mitigate the effects of biases. Let’s look at the five most common biases.



Reporting bias occurs when the frequency of events, properties, and/or outcomes captured in a data set does not accurately reflect their real-world frequency. This bias can arise because people tend to focus on documenting circumstances that are unusual or especially memorable, assuming that the ordinary is generally accepted.



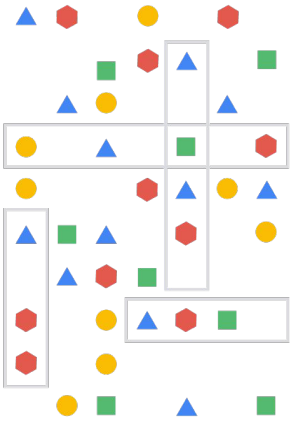
Automation bias is a tendency to favor results generated by automated systems over those generated by non-automated systems, irrespective of the error rates of each.



Selection bias occurs if a dataset's examples are chosen in a way that doesn’t reflect their real-world distribution.

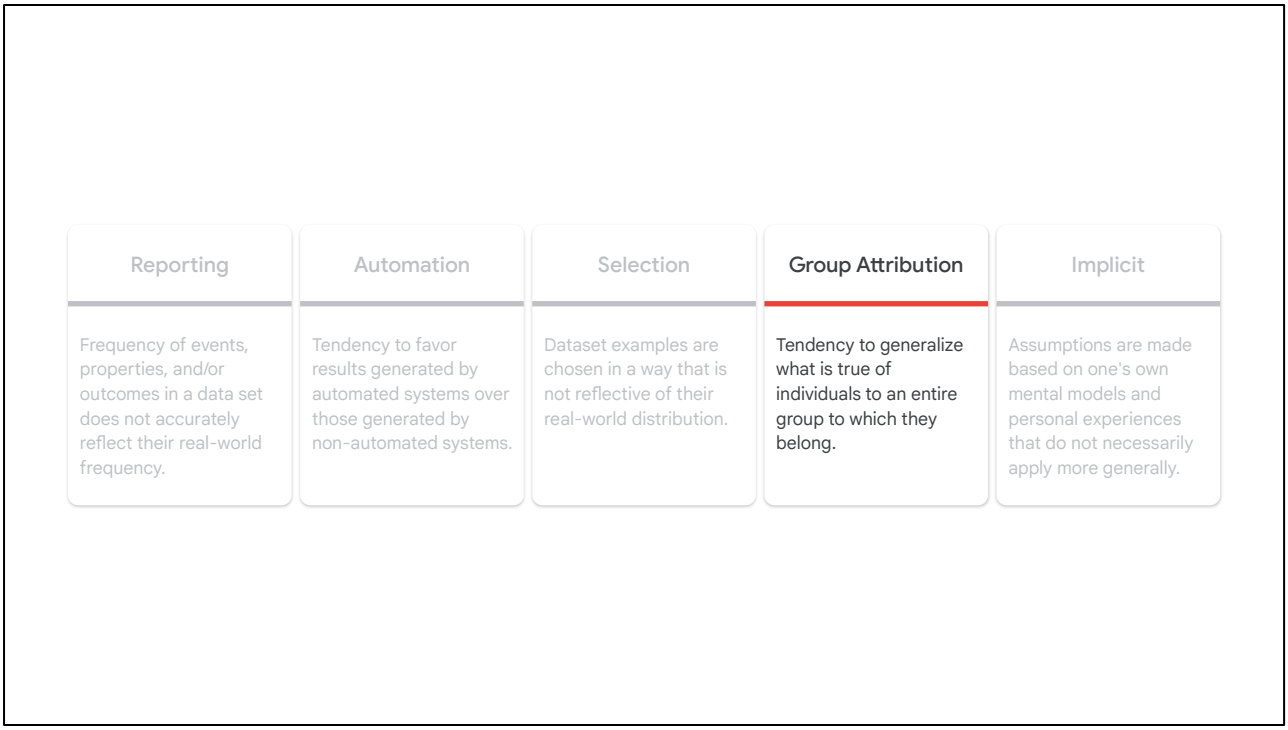
Forms of selection bias

- Coverage bias
- Non-response bias (or participation bias)
- Sampling bias



Selection bias can take many different forms such as:

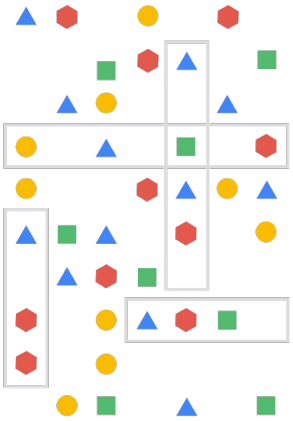
- Coverage bias - this occurs when data is not selected in a representative fashion.
- Non-response bias (or participation bias) - this happens when data ends up being unrepresentative due to participation gaps in the data-collection process.
- And sampling bias - which occurs when proper randomization is not used during data collection.



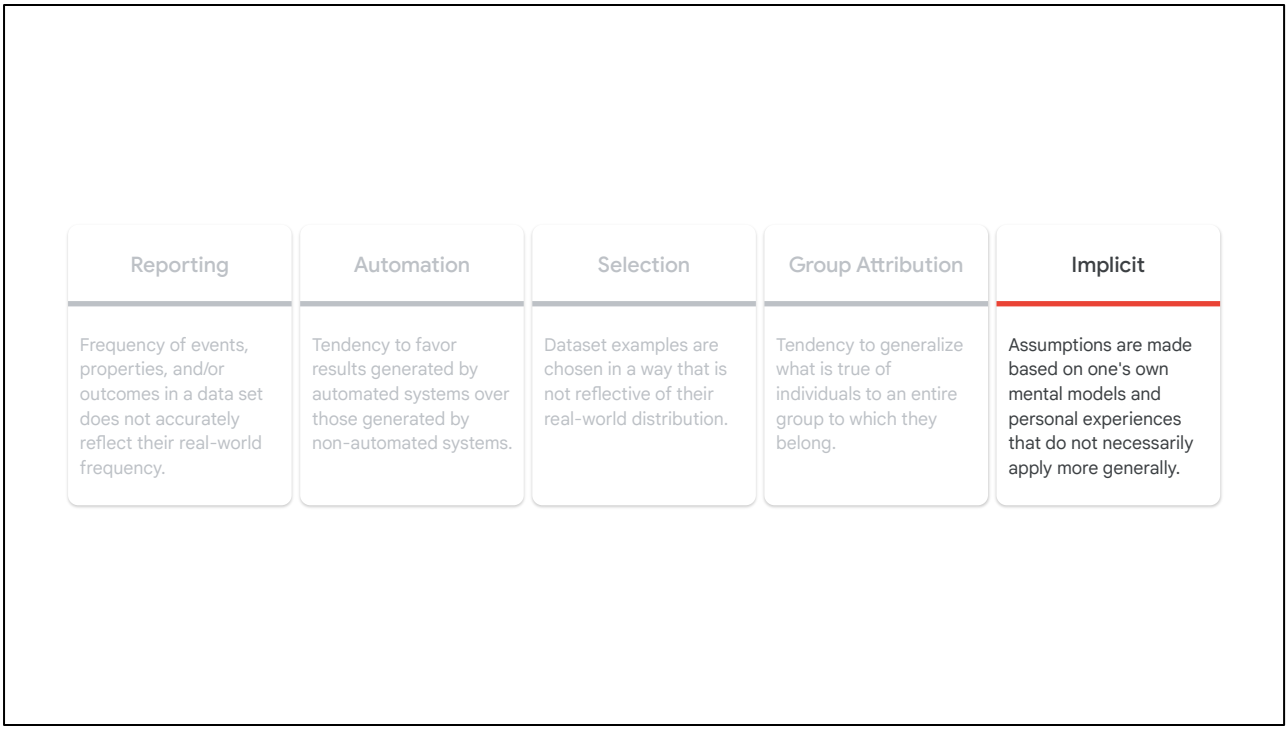
Group attribution bias is a tendency to generalize what is true of individuals to an entire group to which they belong. Two key manifestations of this bias are in-group bias and out-group homogeneity bias

Forms of group attribution bias

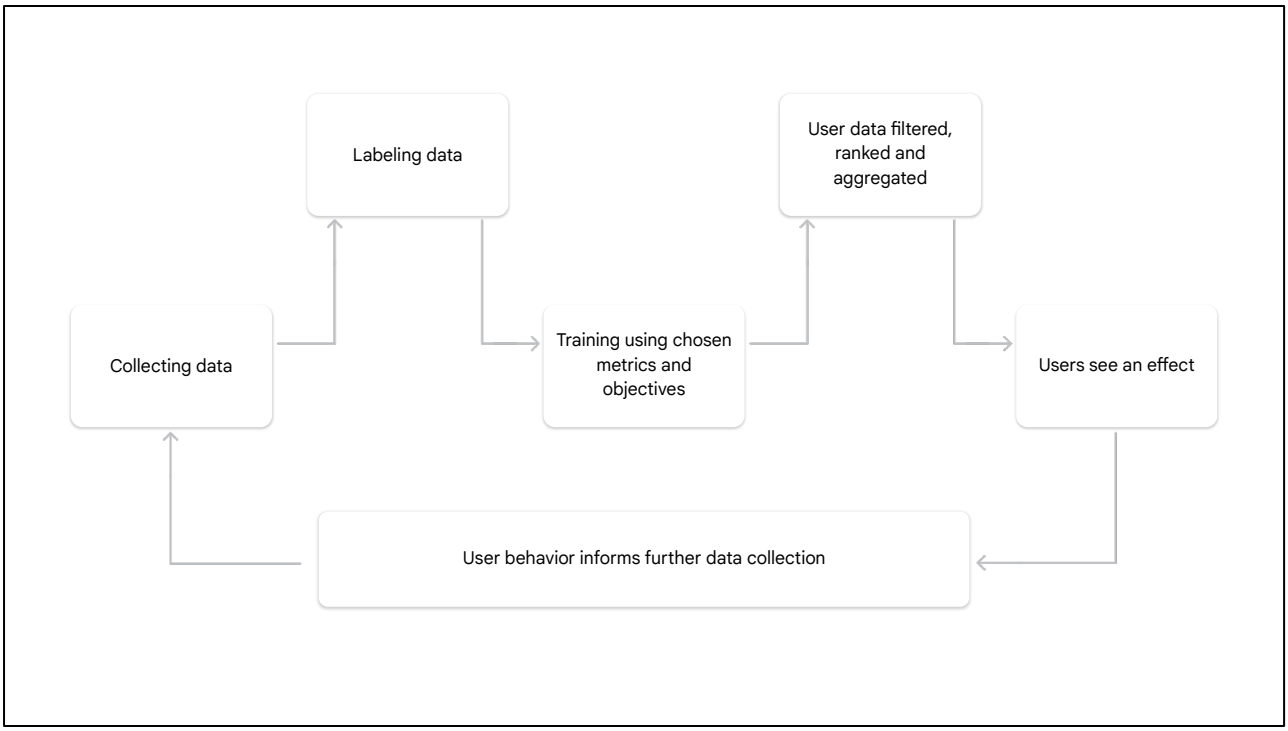
- In-group bias
- Out-group homogeneity bias



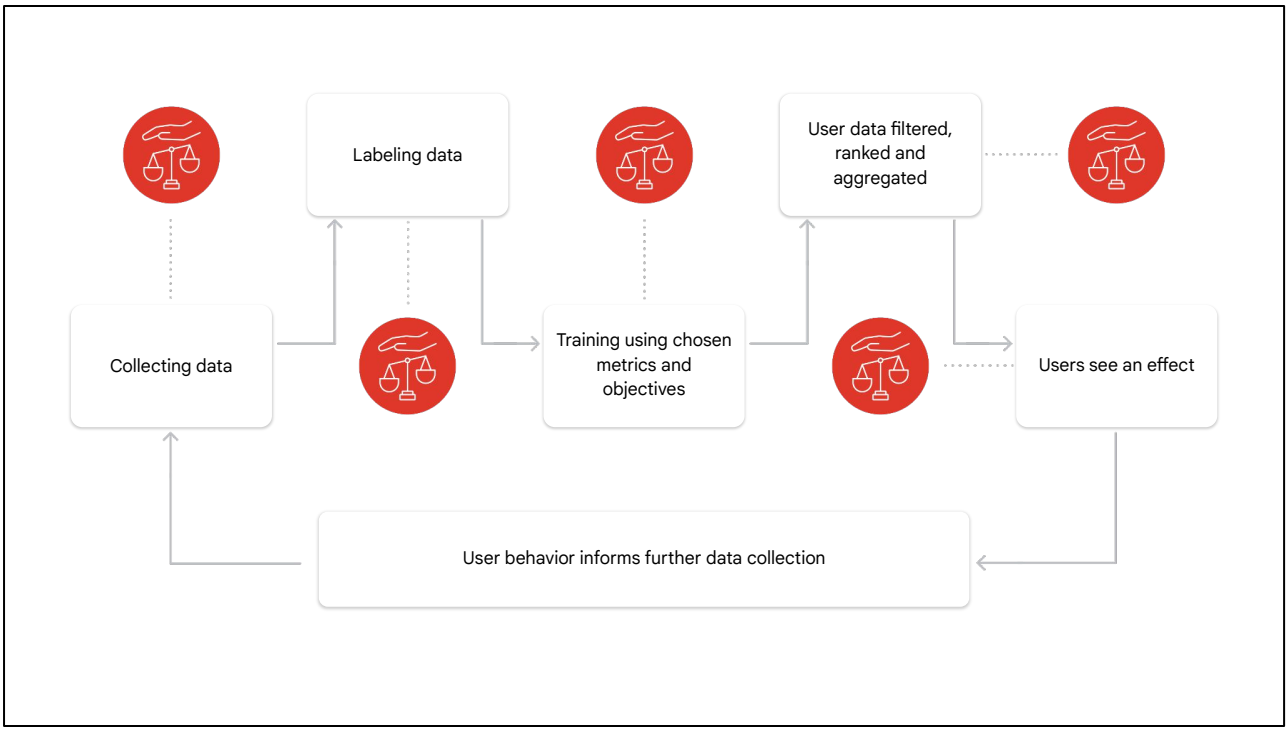
- In-group bias occurs when there is a preference for members of a group to which you also belong, or for characteristics that you also share.
- Out-group homogeneity bias occurs when there is a tendency to stereotype individual members of a group to which you do not belong or to see their characteristics as more uniform.



And implicit bias occurs when assumptions are made based on one's own mental models and personal experiences that do not necessarily apply more generally.
A common form of implicit bias is confirmation bias, where model builders unconsciously process data in ways that affirm preexisting beliefs and hypotheses. In some cases, a model builder might actually keep training a model until it produces a result that aligns with their original hypothesis; this is called experimenter's bias.

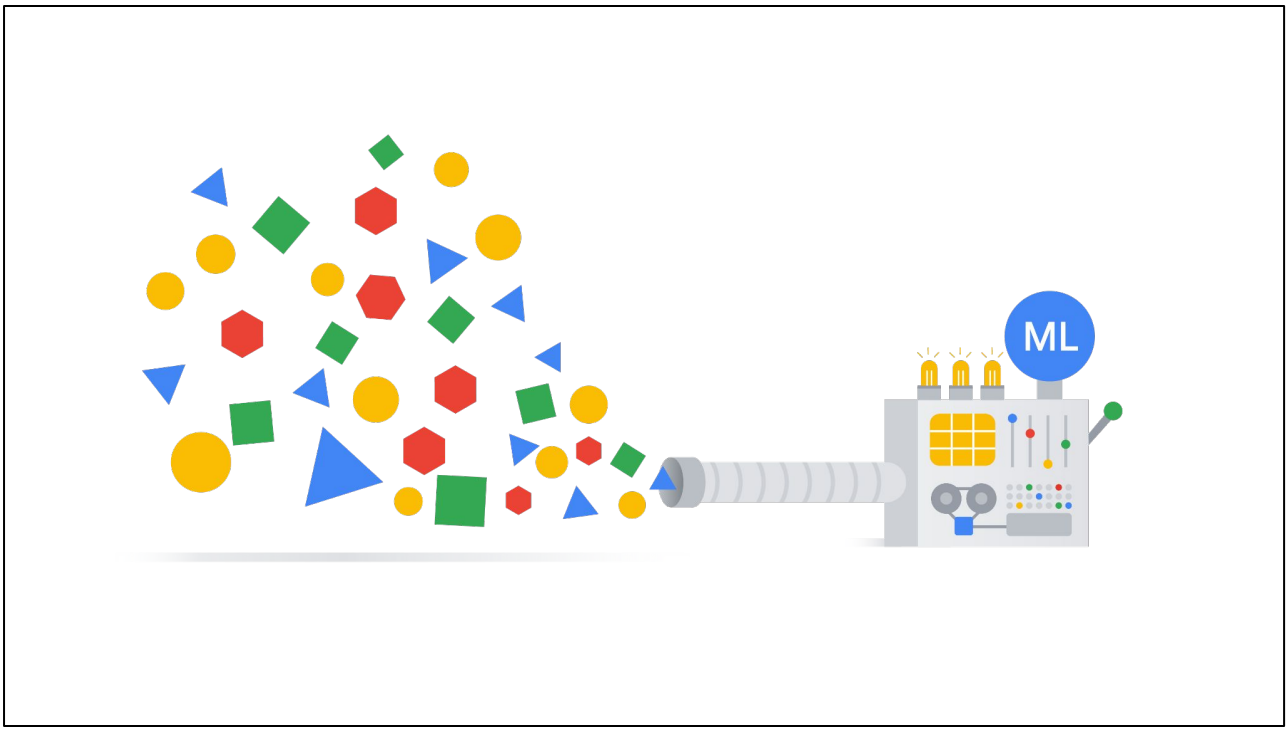


So, how about AI systems? As you know, they usually have multiple steps, including data collection and labeling, training, evaluation, and deployment.
Throughout the lifecycle, bias can enter into the system as a systematic error introduced by sampling or reporting procedures.

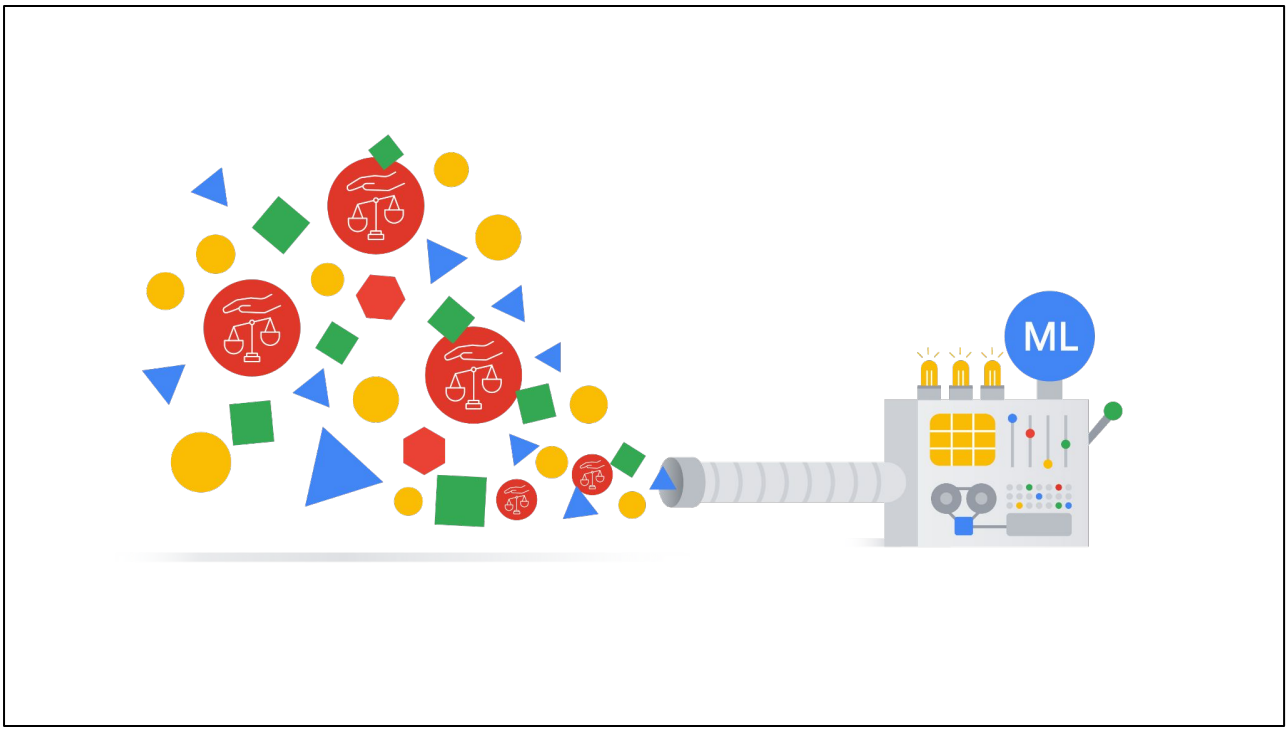


If the data collection has wrong assumptions or implementation, the dataset itself ends up having a lot of biases, such as selection bias.
Engineers can inadvertently train models by feeding the biased data set, and machine learning models will easily find and repeat the bias patterns in predictions.

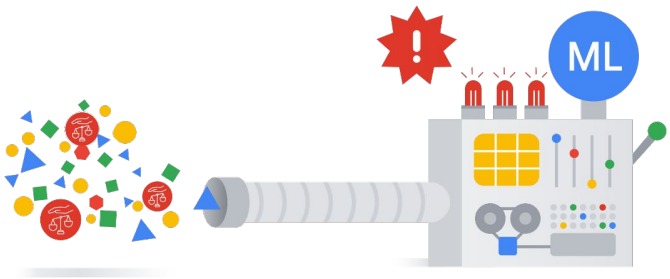
When building models, bias can also enter in the way model components and logic are defined.
Bias can even appear after deployment from feedback loops and model iterations.



You can rarely identify a single cause of, or a single solution to, these bias problems.



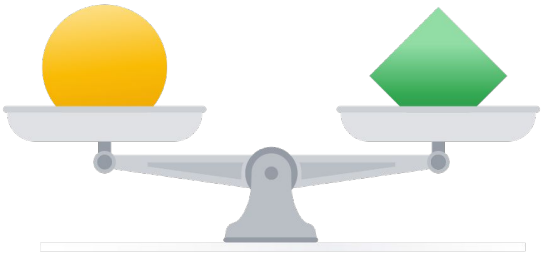
What happens far more often is the existence of various causes that interact in ML systems



to produce problematic outcomes. This results in a range of solutions needed.

AI Fairness

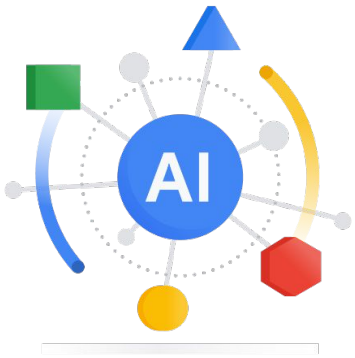
Decisions made by computers after a machine-learning process may be considered unfair if they were based on variables considered sensitive.



Definition from Wikipedia

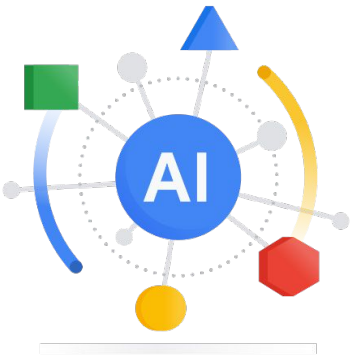
It's important to note that not all types of bias are necessarily unfair.
The definition of fairness is slightly more restrictive: decisions made by computers after a machine-learning process might be considered unfair if they were based on variables considered sensitive.
Examples of such variables include gender, ethnicity, sexual orientation, or disability.
With these variables, fairness also encompasses equity and inclusion.

As the impact of AI increases across sectors and societies...



It is important to work towards AI systems that are fair and inclusive for all as the impact of AI increases across sectors and societies.
Beyond recommending books and television shows, AI systems can be used for more critical tasks, such as matching people to jobs and partners, identifying fraudulent transactions, or identifying a person who is crossing a street on a red light.

As the impact of AI increases across sectors and societies...



Opportunity

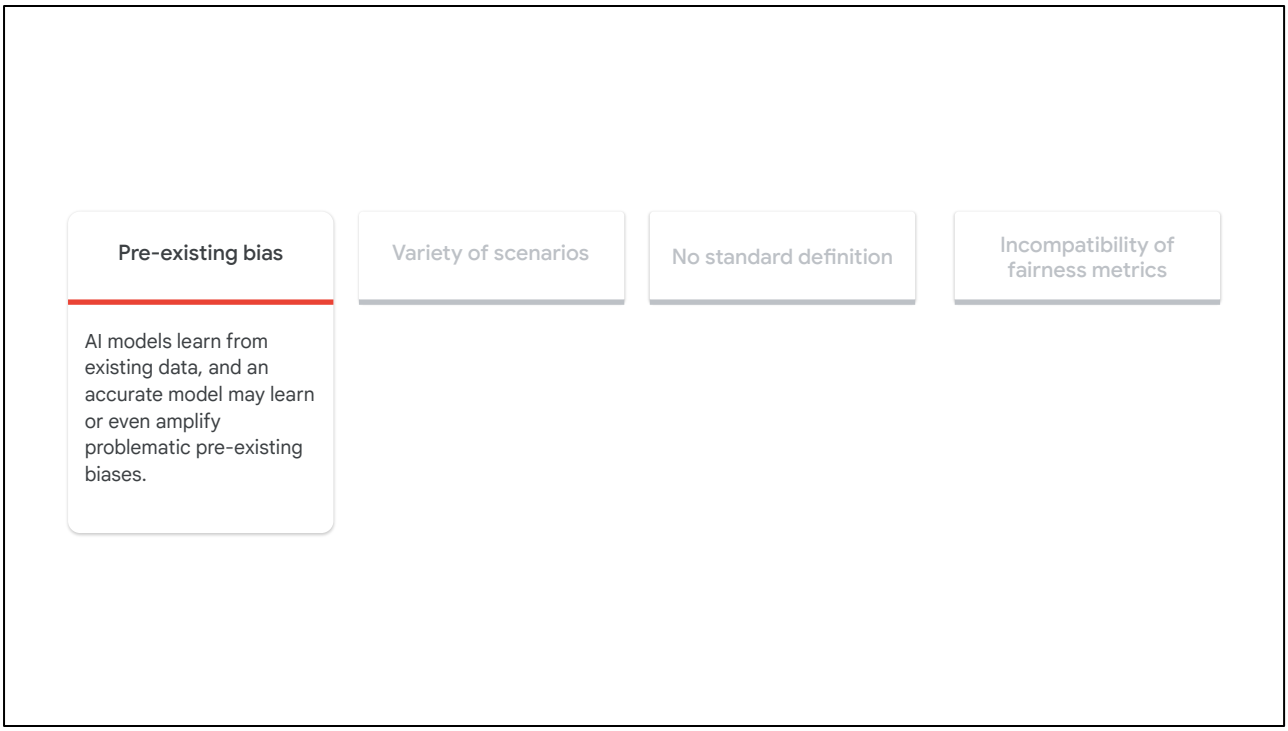
To be fairer and more inclusive at a broader scale.

Risk

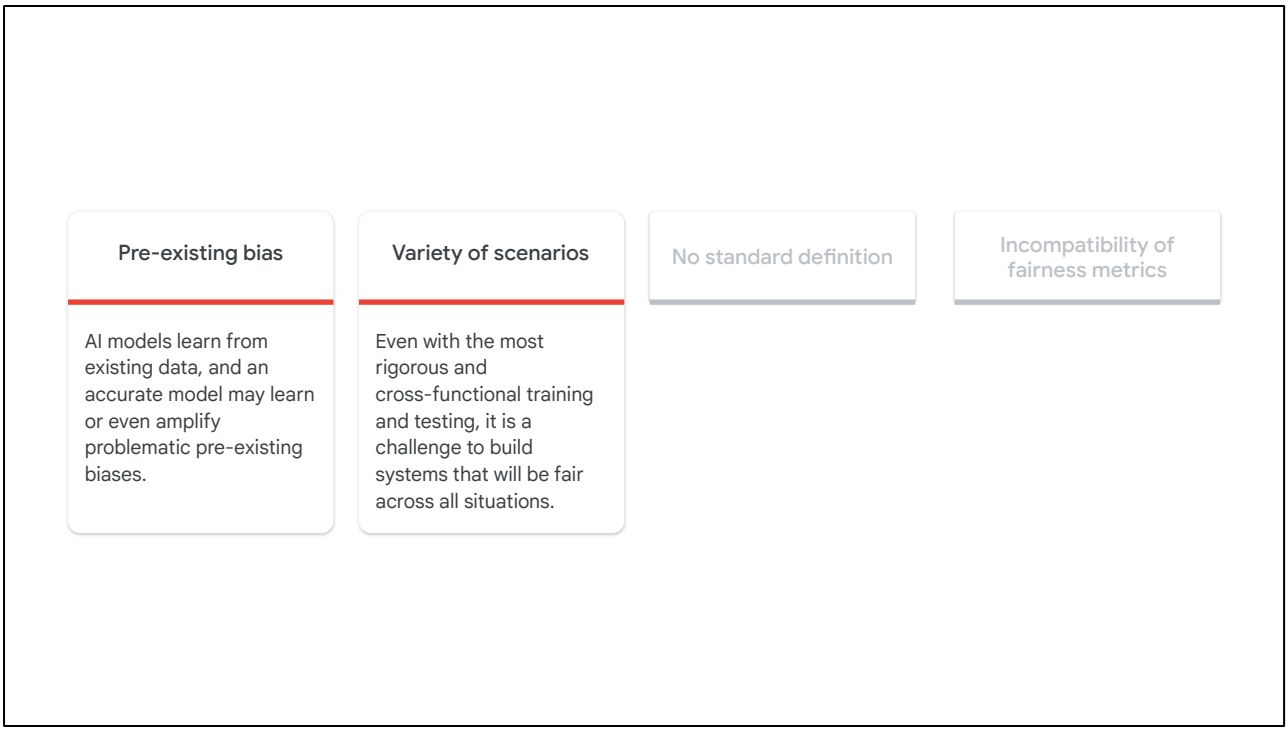
Has the potential to have a negative wide-scale impact.

So what is the opportunity that exists? AI systems have the potential to be fairer and more inclusive at a broader scale than decision-making processes based on ad hoc rules or human judgments.
And the risk that exists? Unfairness in such systems can also have a negative wide-scale impact.

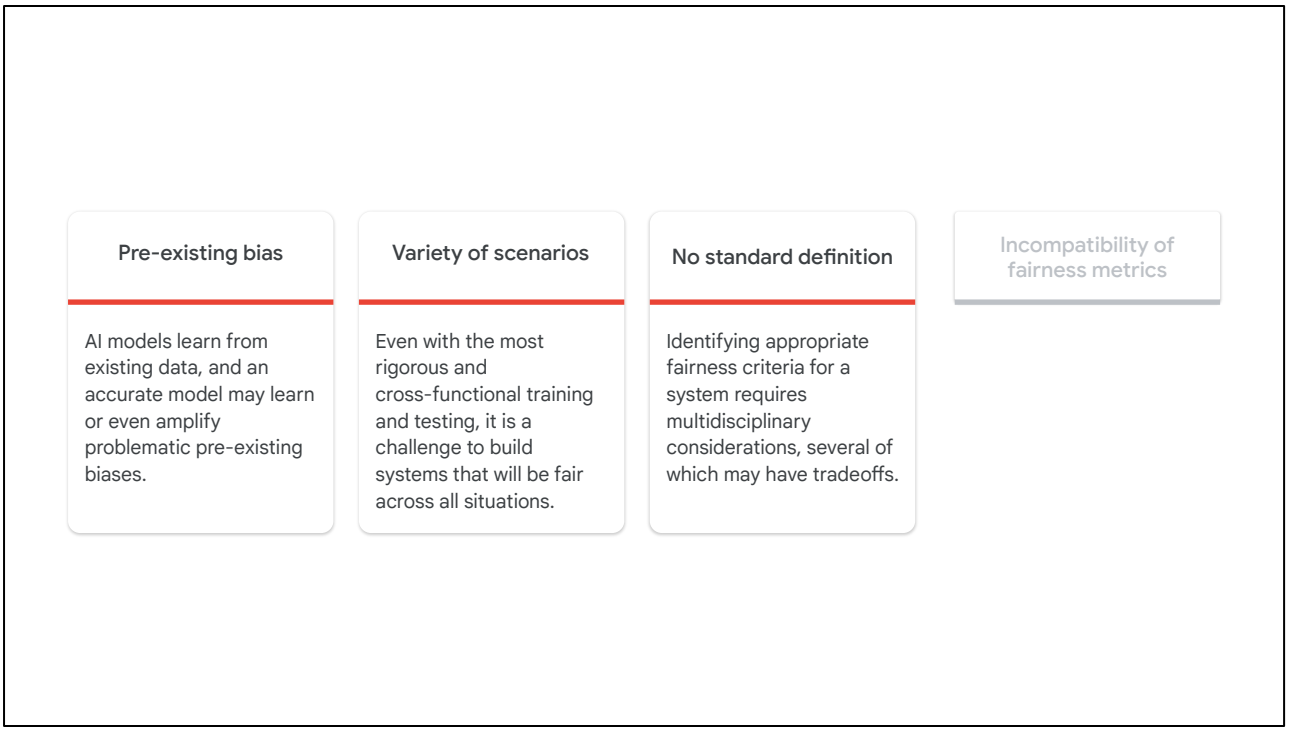
In all this, AI fairness is actually a difficult task to achieve.



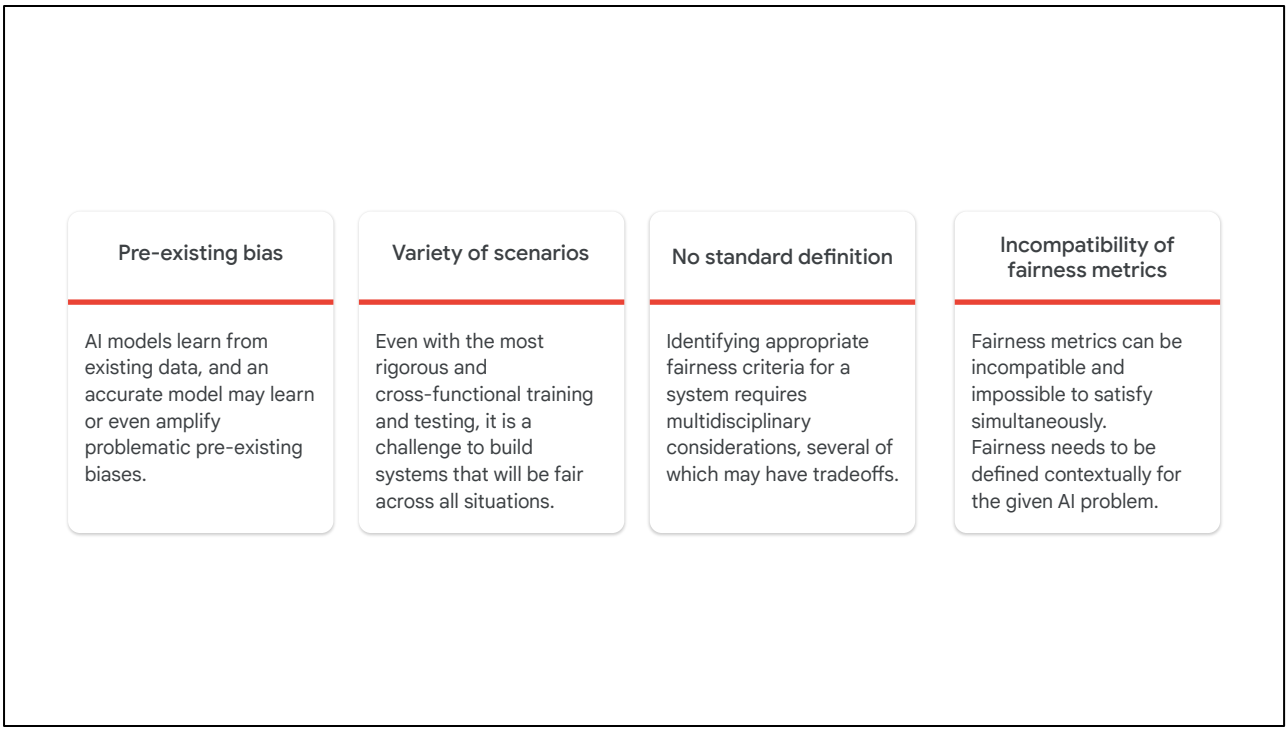
First of all, AI models learn from existing data, and an accurate model might learn or even amplify problematic pre-existing biases.



Second, even with the most rigorous and cross-functional training and testing, it is a challenge to build systems that will be fair across all situations.



Third, there is no standard definition of fairness, whether decisions are made by humans or machines. Identifying appropriate fairness criteria for a system requires accounting for considerations such as user experience, cultural, social, historical, political, legal, and ethical. Several of which may have tradeoffs. Even for situations that seem simple, people may disagree about what is fair, and it may be unclear what point of view should dictate policy, especially in a global setting.



Finally, fairness metrics can be incompatible and impossible to satisfy simultaneously. Hence, it's impossible to define a universal metric for fairness. Don't be discouraged though: it just means that fairness needs to be defined contextually for the given AI problem.

01 Overview of Fairness and Bias

02 Identify Bias – TFDV Tool

03 Identify Bias – What-if Tool

04 Identify Bias – TFMA Tool

05 Mitigate Bias – Data Intervention

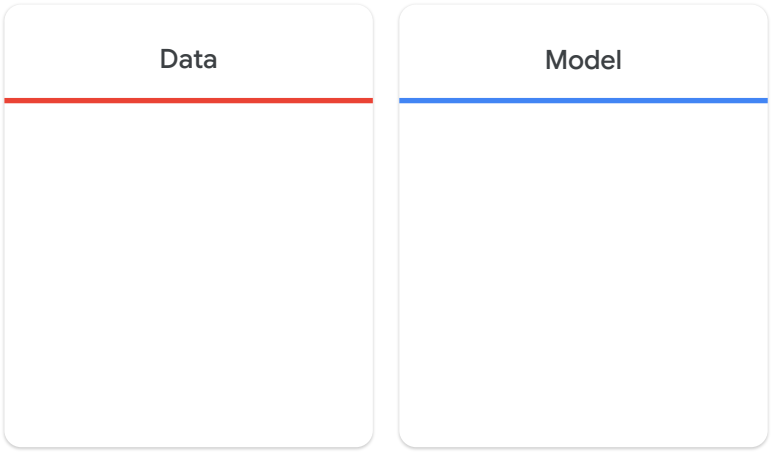
06 Mitigate Bias – Threshold Calibration

07 Mitigate Bias – Model Remediation

08 Lab: Mitigate Bias with MinDiff in TensorFlow

Now, let's talk about how to identify bias problems in AI systems.

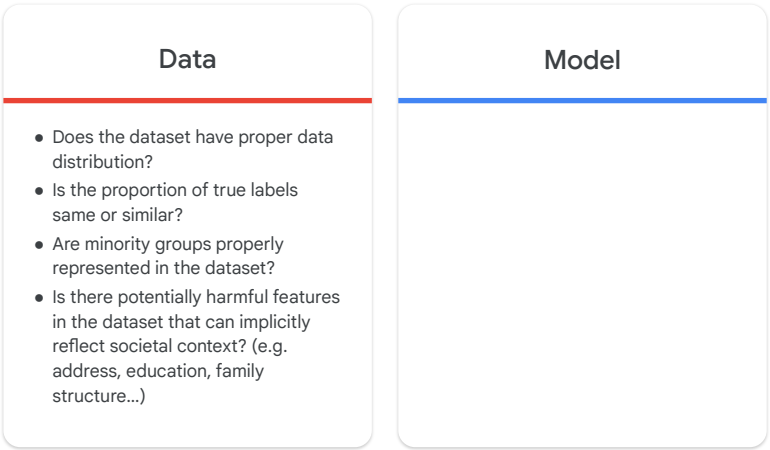
How can you identify biases in machine learning?



There are many possible causes of biases in AI systems.
Let's look from these two perspectives: data and model.

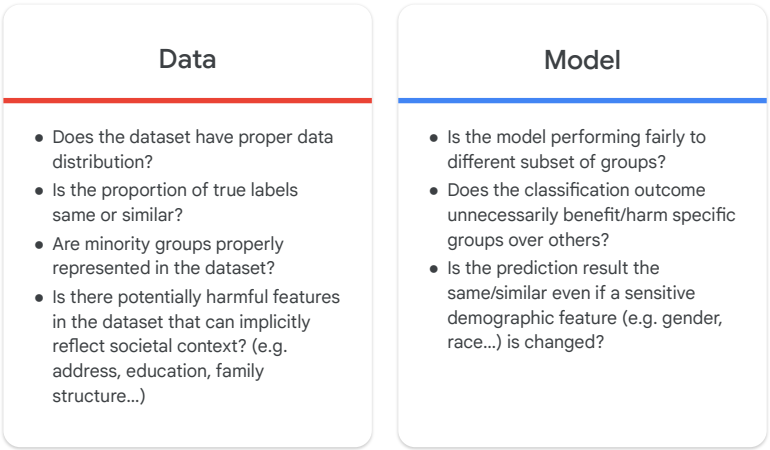
Bias identification always starts with asking good questions.

How can you identify biases in machine learning?



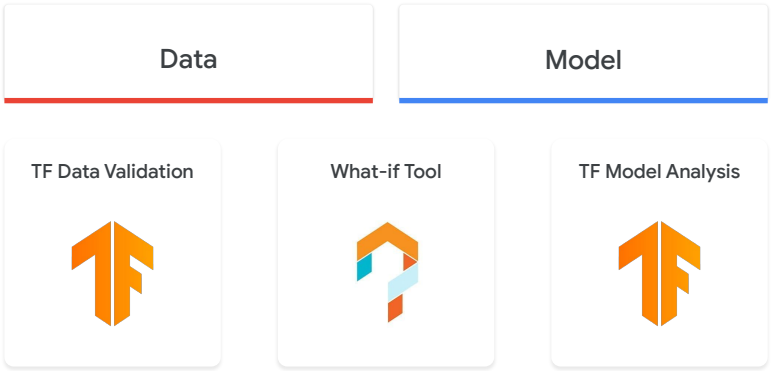
For example, be careful about whether the dataset has proper data distributions. Is it skewed towards a subgroup of the population? Does it have selection bias problems? Also, you should be careful about the proportions. Even if a dataset contains a sufficient number of data points for the minority population, the label proportion can be different from the one of the majority population.

How can you identify biases in machine learning?



And you can check for bias issues in different ways, even after training a model
You can check the model performance in different subsets of the dataset to verify if it doesn't have any harmful biases. You can also check if it doesn't unnecessarily benefit or harm a specific population.

What are good tools to identify fairness?



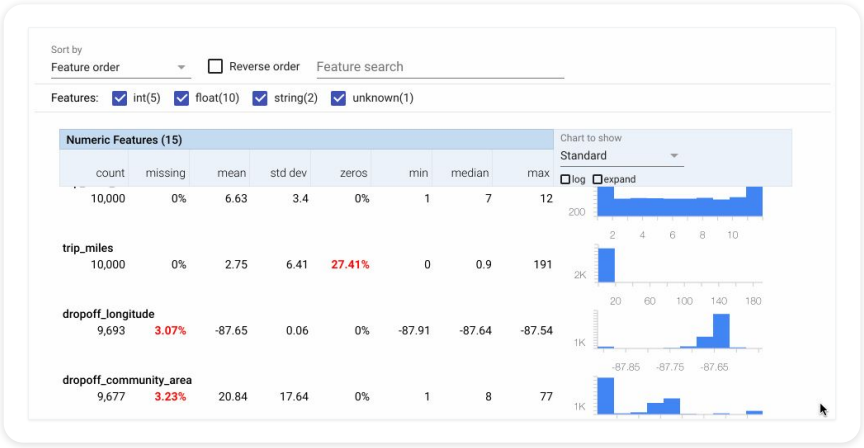
To help us identify bias issues, you can leverage a few open source tools.

- TensorFlow Data Validation is a useful tool for exploring and validating ML data at scale.
- The What-If Tool by Google helps you investigate your data and model bias through data and prediction visualization, and helps you apply counterfactual analysis.
- And TensorFlow Model Analysis is designed to analyze model behavior by applying granular model evaluation.

Although the focus here is on the TensorFlow ecosystem tools, the tools mentioned are not limited to the TensorFlow library, and are compatible with other frameworks. More importantly, the idea itself is universal. So you can find other alternatives if you prefer other libraries. You can even implement an alternative on your own after you feel you properly understand what to do.

TensorFlow Data Validation:

can help you understand data distribution and statistics



Let's start from Tensorflow Data Validation or TFDV. It helps you understand data distribution and statistics.
Although this tool is not developed for fairness purposes only, it is always the starting point to help you investigate.

TensorFlow Data Validation:

A highly-scalable open-source data validation library

- Scalable calculation of summary statistics of training and test data
- Integration with a data viewer for distributions, statistics, and faceted comparison of feature pairs
- Automated data-schema generation, and schema viewer
- Anomaly detection and viewer for missing features, out-of-range values, wrong feature types, ...

```
stats = tfdv.generate_statistics_from_tfrecord(
    data_location=path,
)
```

For starters, with TFDV you can easily calculate summary statistics from a dataset. Although this code example shows you the tf_record format, which is Tensorflow's own, super efficient, binary storage format, you can also use a .csv file or a dataframe directly.
All you need is a path to your data.

TensorFlow Data Validation:

A highly-scalable open-source data validation library

- Scalable calculation of summary statistics of training and test data
- Integration with a data viewer for distributions, statistics, and faceted comparison of feature pairs
- Automated data-schema generation, and schema viewer
- Anomaly detection and viewer for missing features, out-of-range values, wrong feature types, ...

```
# Slice on country feature
# (i.e., every unique value of the feature)
slice_fn1 = slicing_util.get_feature_value_slicer(
    features={'country': None}
)

# Slice on the cross of country and state feature
# (i.e., every unique pair of values of the cross)
slice_fn2 = slicing_util.get_feature_value_slicer(
    features={'country': None, 'state': None}
)

# Slice on specific values of a feature
slice_fn3 = slicing_util.get_feature_value_slicer(
    features={'age': [10, 50, 70]}
)

stats_options = tfdv.StatsOptions(
    slice_functions=[slice_fn1, slice_fn2, slice_fn3]
)
```

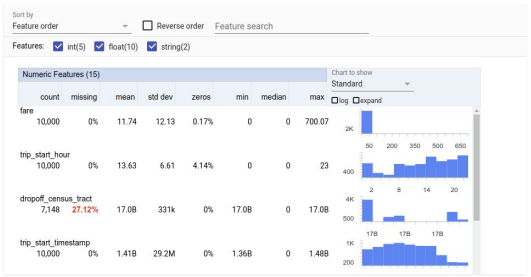
With TFDV, you can also calculate statistics by data slices.
For example, you can slice data by country and state to help understand data proportion differences in various locations.

TensorFlow Data Validation:

A highly-scalable open-source data validation library

- Scalable calculation of summary statistics of training and test data
- Integration with a data viewer for distributions, statistics, and faceted comparison of feature pairs
- Automated data-schema generation, and schema viewer
- Anomaly detection and viewer for missing features, out-of-range values, wrong feature types, ...

```
tfdv.visualize_statistics(stats)
```



After you calculate your stats, simply visualize them by using the method visualize_statistics.

The dashboard contains two tables: one for numeric features, and one for categorical features. The dashboard also has a control panel. The tables and dashboard are separated due to different information being shown for the two data types.

Now in the table, a row corresponds to a feature. It contains some calculated statistics about the values of that feature across the entire dataset, along with a number of charts to show the distribution of values.

For numerical values, TFDV calculates mean, standard deviation, number of zeros, and the min, median, and max values.

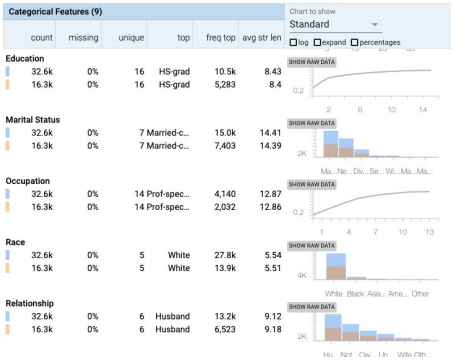
The tool automatically tags some stats that might be problematic by bolding them and coloring them red. For example, if a feature had a high number of zero values (also known as null values), it might be automatically tagged.

TensorFlow Data Validation:

A highly-scalable open-source data validation library

- Scalable calculation of summary statistics of training and test data
- Integration with a data viewer for distributions, statistics, and faceted comparison of feature pairs
- Automated data-schema generation, and schema viewer
- Anomaly detection and viewer for missing features, out-of-range values, wrong feature types, ...

```
tfdv.visualize_statistics(stats)
```



For categorical features, you are given the number of missing values, number of unique values, top/most frequent category, frequency of the top category, and average strength length.

Notice this new screenshot and how it is different from the previous screenshot. Here within the same view, you can see a direct comparison of multiple data slices, represented by blue and orange bars.

TensorFlow Data Validation:

A highly-scalable open-source data validation library

- Scalable calculation of summary statistics of training and test data
- Integration with a data viewer for distributions, statistics, and faceted comparison of feature pairs
- Automated data-schema generation, and schema viewer
- Anomaly detection and viewer for missing features, out-of-range values, wrong feature types, ...

```
schema = tfdv.infer_schema(stats)
```

```
feature {
  name: "payment_type"
  value_count {
    min: 1
    max: 1
  }
  type: BYTES
  domain: "payment_type"
  presence {
    min_fraction: 1.0
    min_count: 1
  }
}
```

With T F D V, you can also automatically infer the schema given stats. This is particularly useful for datasets with lots of features, where manually defining the schema can be a tedious task. Instead, automatically generate it using infer schema, then refine as needed!

This schema is especially useful when you want to automate data validation steps in an automated M L Ops pipeline. After defining the ideal characteristics of the dataset, you can automatically check the data issues from the next M L Ops pipeline execution.

TensorFlow Data Validation:

A highly-scalable open-source data validation library

- Scalable calculation of summary statistics of training and test data
- Integration with a data viewer for distributions, statistics, and faceted comparison of feature pairs
- Automated data-schema generation, and schema viewer
- Anomaly detection and viewer for missing features, out-of-range values, wrong feature types, ...

```
anomalies = tfdv.validate_statistics(
   statistics=other_stats, schema=schema,
)
```

```
payment_type  Unexpected string values
Examples contain values missing from the
schema: Prcard (<1%).
```

```
options = tfdv.StatsOptions(schema=schema)
anomalous_stats = tfdv.validate_examples_in_csv(
       data_location=input, stats_options=options
)
```

```
tfdv.get_feature(schema,payment_type).skew_comparat
or.infinity_norm.threshold = 0.01
skew_anomalies = tfdv.validate_statistics(
   statistics=stats_1, schema=schema,
   serving_statistics=stats_2,
)
```

Lastly, T F D V lets you easily detect and act on anomalies by using the schema defined.

Anomalies can be out of domain unexpected features, a skew of the entire dataset, or a proportional gap between multiple data slices.

By using this capability in a machine learning pipeline, especially in an automated system, you detect data problems before you train models.

Again, although T F D V offers useful functionalities, if you prefer, you can also develop a similar data validation system by using different libraries.

01 Overview of Fairness and Bias

02 Identify Bias – TFDV Tool

03 Identify Bias – What-if Tool

04 Identify Bias – TFMA Tool

05 Mitigate Bias – Data Intervention

06 Mitigate Bias – Threshold Calibration

07 Mitigate Bias – Model Remediation

08 Lab: Mitigate Bias with MinDiff in TensorFlow

Let’s look at the next tools. The What-If Tool can be used to visually analyze the interaction of datasets and ML models.

What-If Tool:

Open-source tool to visually probe ML datasets and models

- Visualize inference results
- Edit a datapoint and see how your model performs
- Explore the effects of single features
- Arrange examples by similarity
- View confusion matrices and other metrics
- Test algorithmic fairness constraints

With the What-If Tool, you can do many explorations by using the Datapoint Editor and the Performance and Fairness tabs.

Let’s look at each.

What-If Tool:

Open-source tool to visually probe ML datasets and models

- Visualize inference results
- Edit a datapoint and see how your model performs
- Explore the effects of single features
- Arrange examples by similarity
- View confusion matrices and other metrics
- Test algorithmic fairness constraints

After loading a model and dataset, the first view you see allows you to visualize inference results using Facets Dive.

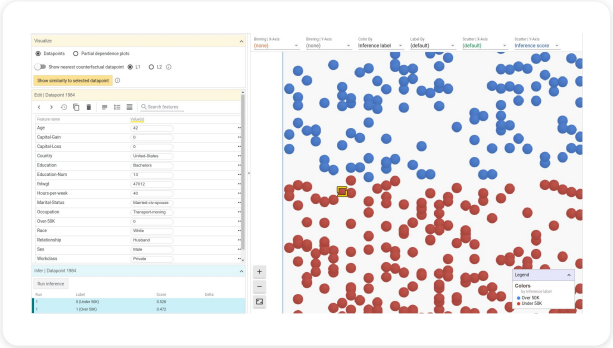
In this example, each datapoint is colored by the category that the model predicted for it. Points are laid out top to bottom following the inference score. This means that points at the bottom have a prediction close to 0 and very likely belong to the negative label. Points at the very bottom have a prediction close to 1 and very likely belong to the positive label.

With this view, you could easily say that the model is often very confident since most points are close to 0 or 1, and that the model predicts more points belong to the red label.

What-If Tool:

Open-source tool to visually probe ML datasets and models

- Visualize inference results
- Edit a datapoint and see how your model performs
- Explore the effects of single features
- Arrange examples by similarity
- View confusion matrices and other metrics
- Test algorithmic fairness constraints

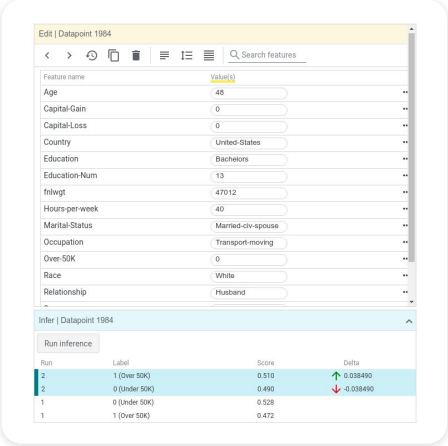


From the Facets Dive view, you can investigate any data point. In this case, you would probably select difficult use cases such as data points near the decision boundary.
To see datapoint details, click on the datapoint and a panel appears to the left of the visualization.
You can then modify any value, and see if the prediction changes:

What-If Tool:

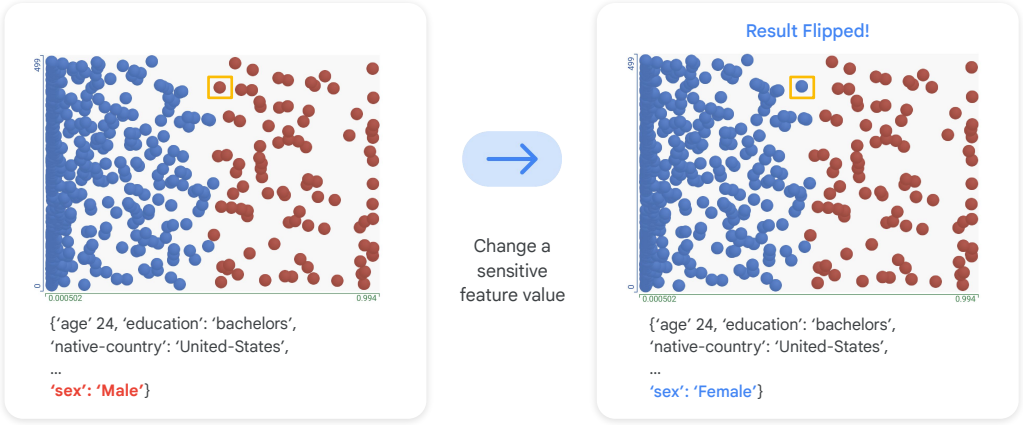
Open-source tool to visually probe ML datasets and models

- Visualize inference results
- Edit a datapoint and see how your model performs
- Explore the effects of single features
- Arrange examples by similarity
- View confusion matrices and other metrics
- Test algorithmic fairness constraints



For example, what if you change the age? Or the gender group of this data point?

Counterfactual Analysis helps you understand hidden model biases



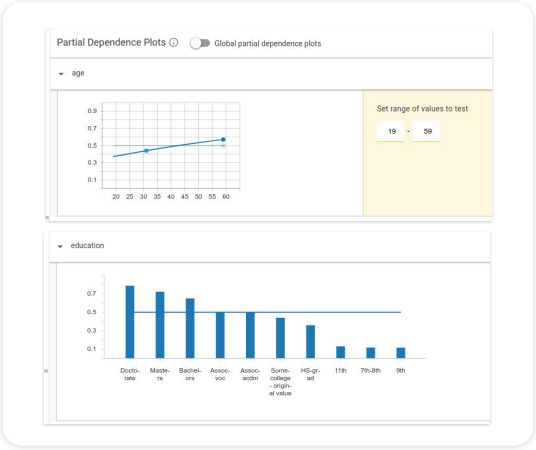
You can change a specific feature to see if the prediction is affected by that. For example, by changing the “sex” feature from Male to Female, the model prediction was flipped from positive to negative.
This kind of ‘what-if’ analysis is called counterfactual analysis, and it helps us understand the model behavior and identify bias problems.

You can also apply this method to the entire dataset, and quantify the ratio of the flipped result when changing a sensitive feature value. This fairness metric is called “flip ratio”, and helps you understand the model’s vulnerability.

What-If Tool:

Open-source tool to visually probe ML datasets and models

- Visualize inference results
- Edit a datapoint and see how your model performs
- Explore the effects of single features
- Arrange examples by similarity
- View confusion matrices and other metrics
- Test algorithmic fairness constraints



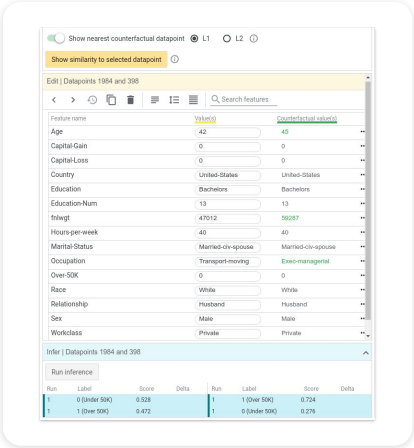
You can also explore the effects of single features for a prediction result through partial dependence plots. This technique can also help us find bias problem, which we will explain in the next interpretability module.

The top example shows a partial dependence plot for a numerical value, and the second example shows a plot for a categorical value.
You can easily see that the model has learned a positive correlation between age and positive prediction, and that higher degree is correlated to higher positive prediction.

What-If Tool:

Open-source tool to visually probe ML datasets and models

- Visualize inference results
- Edit a datapoint and see how your model performs
- Explore the effects of single features
- Arrange examples by similarity
- View confusion matrices and other metrics
- Test algorithmic fairness constraints



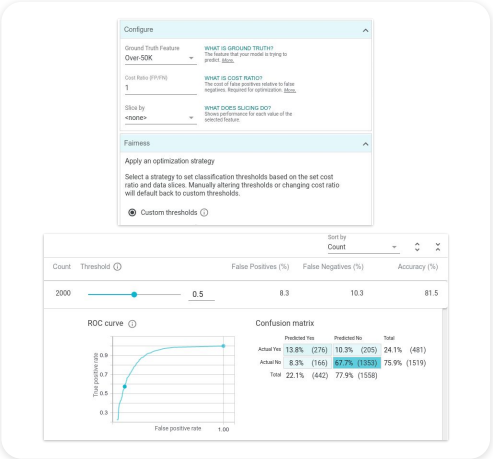
And in the DataPoint Editor tab, you can find the most similar example to a datapoint, either using L1 or L2 distance.
The tool compares the two points side-by-side, and the green text represents features where the two data points differ.

Let’s now look at the “Performance and Fairness” tab for the last two points.

What-If Tool:

Open-source tool to visually probe ML datasets and models

- Visualize inference results
- Edit a datapoint and see how your model performs
- Explore the effects of single features
- Arrange examples by similarity
- View confusion matrices and other metrics
- Test algorithmic fairness constraints



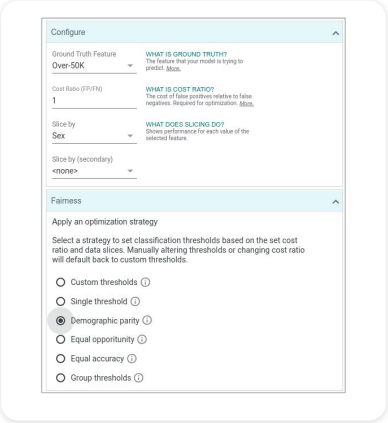
Here you can view confusion matrices and other metrics overall, or per group. You can also slide the classification threshold around, and the metrics will be updated accordingly.

You also have the ability to automatically calculate the optimal classification threshold given a desired cost ratio, like, for example, the relative cost of a false positive versus a false negative. The default cost ratio in the tool is 1, which means that false negatives and false positives are equally undesirable.

What-If Tool:

Open-source tool to visually probe ML datasets and models

- Visualize inference results
- Edit a datapoint and see how your model performs
- Explore the effects of single features
- Arrange examples by similarity
- View confusion matrices and other metrics
- Test algorithmic fairness constraints



Finally, you can test algorithmic fairness constraints by using the slicing capability in the “Performance & Fairness” tab.
You can analyze model performance for any (sensitive) feature. For example, if you use sex, you can see that the model is more accurate on females than males, in the sense that it has less false positives and false negatives.

You can also automatically calculate the optimal classification threshold given a fairness goal such as demographic parity, which we will discuss in the next section.

01 Overview of Fairness and Bias

02 Identify Bias – TFDV Tool

03 Identify Bias – What-if Tool

04 Identify Bias – TFMA Tool

05 Mitigate Bias – Data Intervention

06 Mitigate Bias – Threshold Calibration

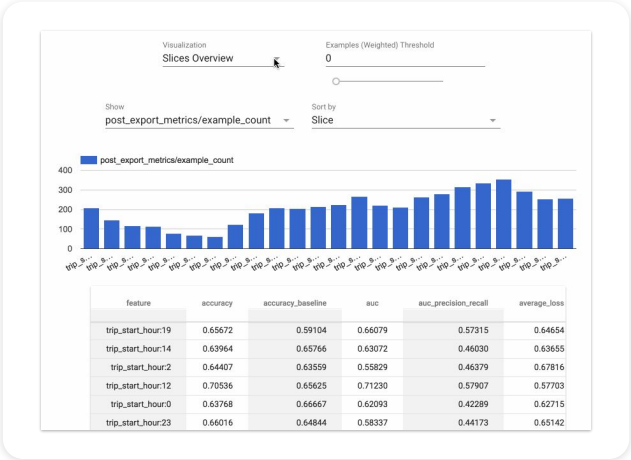
07 Mitigate Bias – Model Remediation

08 Lab: Mitigate Bias with MinDiff in TensorFlow

The TensorFlow Model Analysis, or TFMA, library can help you analyze trained model performance from multiple perspectives, including fairness.

TensorFlow Model Analysis:

Highly-scalable open-source model analysis library



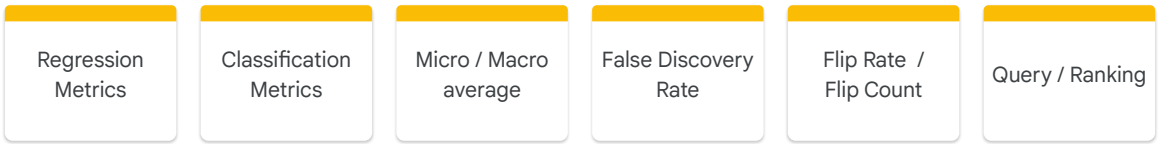
feature	accuracy	accuracy_baseline	auc	auc_precision_recall	average_loss
trip_start_hour:19	0.65672	0.59104	0.66079	0.57315	0.64654
trip_start_hour:14	0.63964	0.65766	0.63072	0.46030	0.63655
trip_start_hour:2	0.64407	0.63559	0.55829	0.46379	0.67816
trip_start_hour:12	0.70536	0.65625	0.71230	0.57907	0.57703
trip_start_hour:0	0.63768	0.66667	0.62093	0.42289	0.62715
trip_start_hour:23	0.66016	0.64844	0.58337	0.44173	0.65142

The TFMA library is designed to support tensorflow based models, but can be easily extended to other frameworks such as pandas dataframe or scikit-learn models. For TF Keras models, it can automatically apply the training metrics at evaluation time, and doesn't require additional steps to compute with pre-calculated predictions. The results are automatically saved with the model.

TensorFlow Model Analysis:

Highly-scalable open-source model analysis library

Supported metrics includes:



TFMA supports a wide variety of metrics including all the standard TensorFlow metrics. This includes regression metrics and classification metrics, and fairness related metrics, like flip rate.

Metrics settings can be customized, and you can also define custom metrics entirely.

TensorFlow Model Analysis:

A highly-scalable open-source model analysis library

- Run model analysis on a single serving model
- Validate model performance on defined metrics
- Compare two models
- Perform fairness analysis with Fairness Indicators

Let’s look at TFMA’s capabilities in more detail.

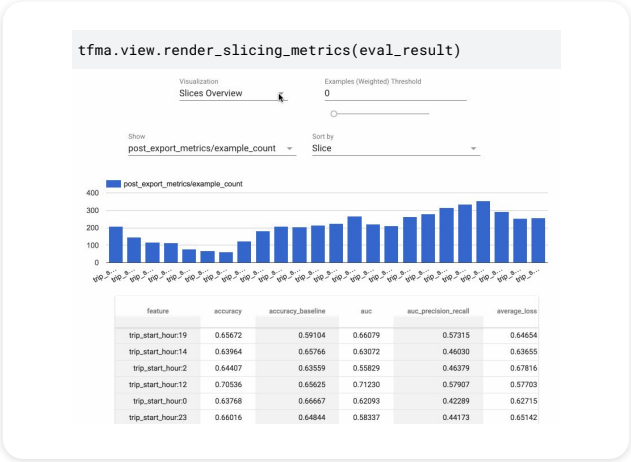
TensorFlow Model Analysis:

A highly-scalable open-source model analysis library

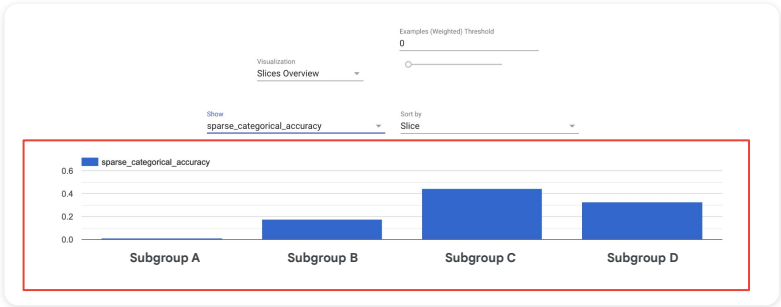
- Run model analysis on a single serving model
- Validate model performance on defined metrics
- Compare two models
- Perform fairness analysis with Fairness Indicators

After training a model, you can first run model analysis on the model. This step will create an evaluation result that can be saved for later use.

TMFA results visualization



This result can be visualized by using a TFMA widget that enables you to interactively check model performance metrics.

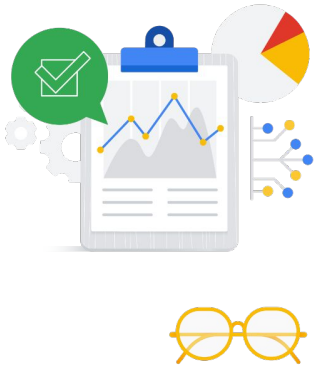


You can also slice the performance by using a sensitive feature, such as racial group. With this capability, you can easily check if there is a critical performance gap among different groups.

TensorFlow Model Analysis:

A highly-scalable open-source model analysis library

- Run model analysis on a single serving model
- Validate model performance on defined metrics
- Compare two models
- Perform fairness analysis with Fairness Indicators

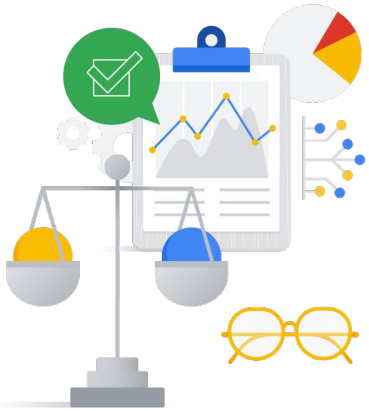


You can automate the model performance validation process using defined metrics thresholds.
For example, in an automated machine learning pipeline, if T F M A finds a critical performance gap, you can stop the model update process before deploying the model.
This capability is important when you are building an automated machine learning system.

TensorFlow Model Analysis:

A highly-scalable open-source model analysis library

- Run model analysis on a single serving model
- Validate model performance on defined metrics
- Compare two models
- Perform fairness analysis with Fairness Indicators

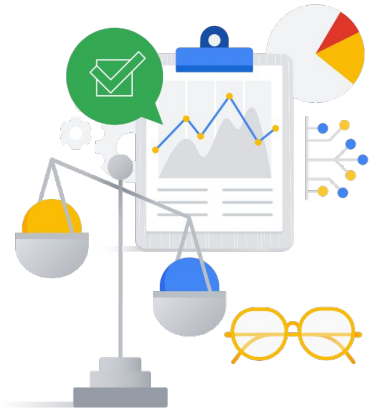


You can also compare the performance of two models, often used to check the performance improvement of a new model over the preceding model.

TensorFlow Model Analysis:

A highly-scalable open-source model analysis library

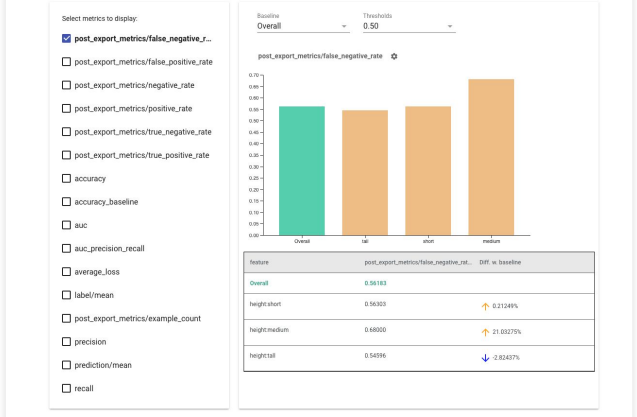
- Run model analysis on a single serving model
- Validate model performance on defined metrics
- Compare two models
- Perform fairness analysis with Fairness Indicators



You can also link the TFMA analysis result to a supplemental library, Fairness Indicators, where you can investigate a model using multiple fairness metrics.

```
from tensorflow_model_analysis.addons.fairness.view
import widget_view

tfma.view.render_slicing_metrics(eval_result)
```



The Fairness indicator also provides an interactive widget, where you can investigate the model performance. This capability can also be incorporated into the What-if Tool.

01 Overview of Fairness and Bias

02 Identify Bias – TFDV Tool

03 Identify Bias – What-if Tool

04 Identify Bias – TFMA Tool

05 Mitigate Bias – Data Intervention

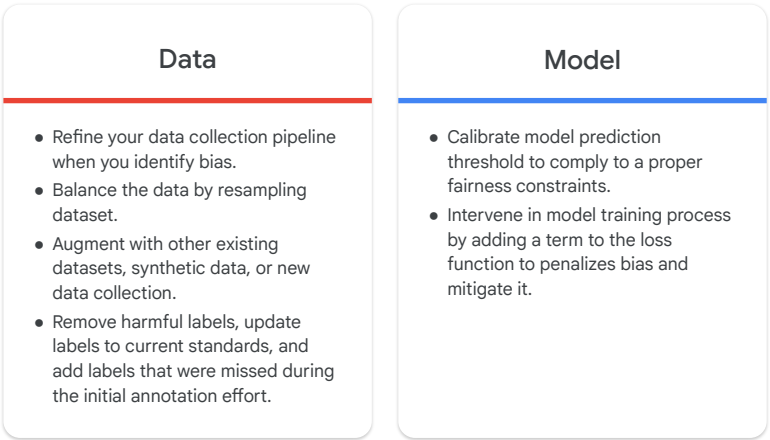
06 Mitigate Bias – Threshold Calibration

07 Mitigate Bias – Model Remediation

08 Lab: Mitigate Bias with MinDiff in TensorFlow

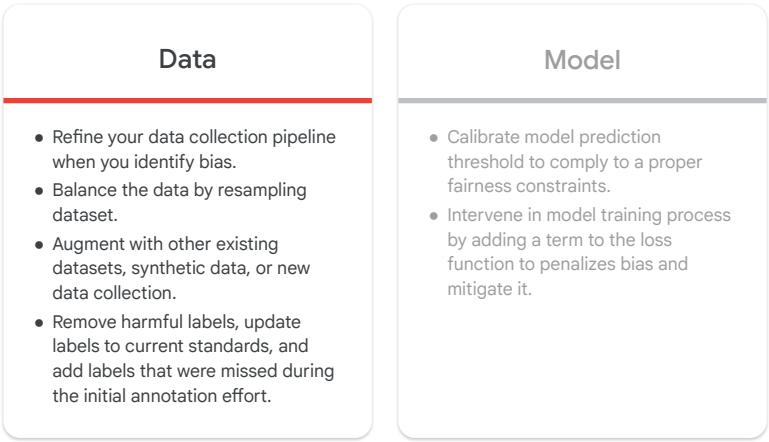
Now we will look at another set of tools and techniques to actually mitigate bias.

How can you identify biases in machine learning?



Let's follow the same format: Data and Model.
You can try to mitigate bias issues by applying several techniques. Although the actual approach depends on the use case, usually it involves multiple methods.

How can you identify biases in machine learning?

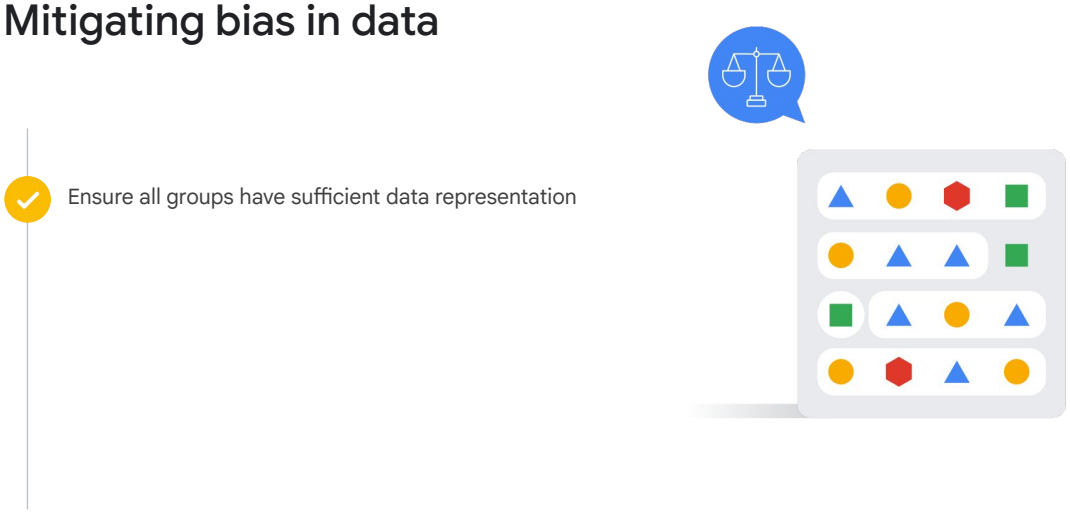


First, let's talk about methods to mitigate biases by intervening data.

Mitigating bias in data

Teams should continuously improve and diversify the data sources and annotations included in their training datasets, as data gaps can be harmful. This is due to overrepresentation and/or insufficient training data for specific subgroups that might lead to drastic performance differences across various subgroups.

Mitigating bias in data

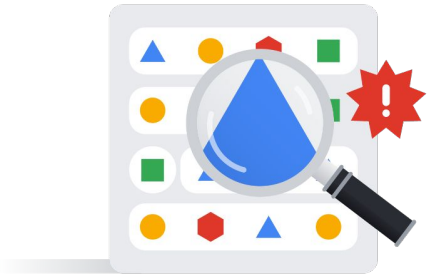


To improve the training data for machine learning models, you should aim for these goals

- Ensure all groups have sufficient data representation

Mitigating bias in data

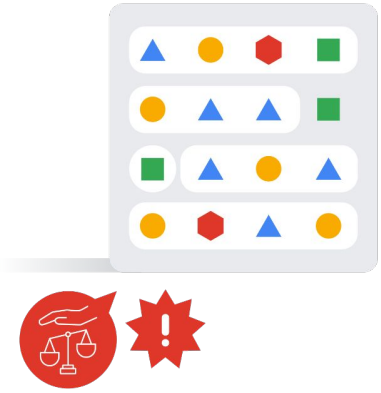
- Ensure all groups have sufficient data representation
- Ensure data does not contain harmful labels or media that are outside your intended use case



- Ensure data does not contain harmful labels or media that are outside your intended use case, such as pejorative labels, or poor quality data.

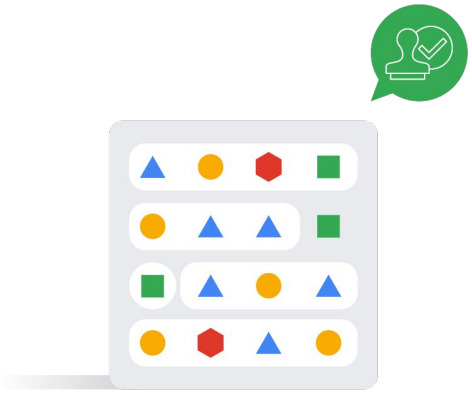
Mitigating bias in data

- Ensure all groups have sufficient data representation
- Ensure data does not contain harmful labels or media that are outside your intended use case
- Ensure your data does not skew towards harmful biases



- Ensure your data does not skew towards harmful biases, like stereotypes or biased correlations, as it may lead to unintended correlations in your model.

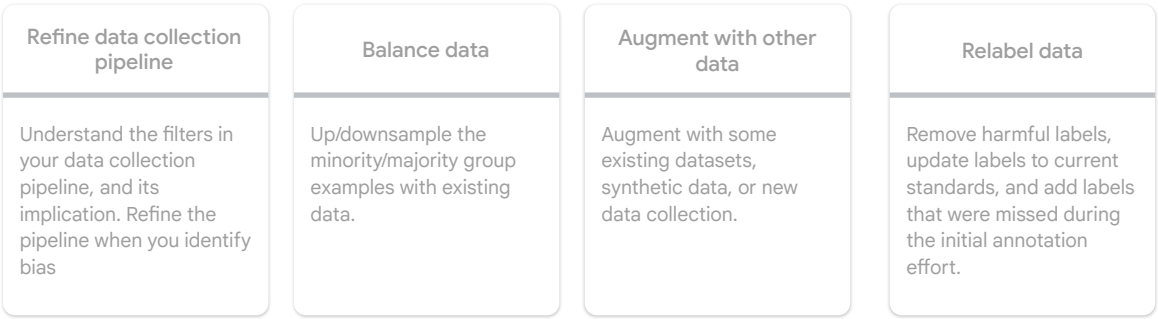
Mitigating bias in data



- Ensure all groups have sufficient data representation
- Ensure data does not contain harmful labels or media that are outside your intended use case
- Ensure your data does not skew towards harmful biases
- Ensure your data does not have unintended biases in either the authorship, the way the data was captured or the content of the data itself.

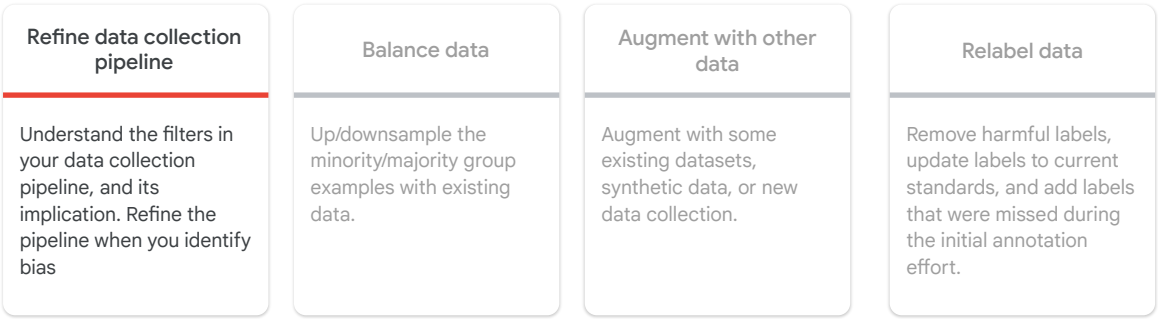
- And ensure your data does not have unintended biases in either the authorship (who collected the data), the way the data was captured (how it was captured), the way it was described (who annotated/labeled it), or the content of the data itself (what was captured).

Mitigating bias in data



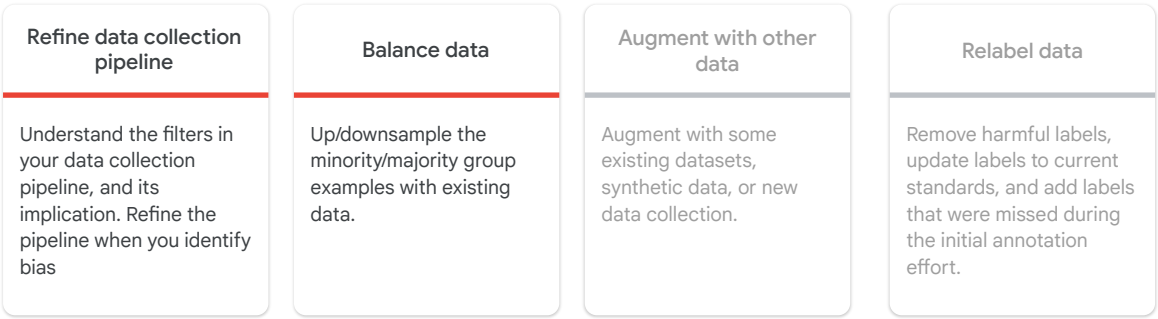
To achieve these goals, here are some techniques that help.

Mitigating bias in data

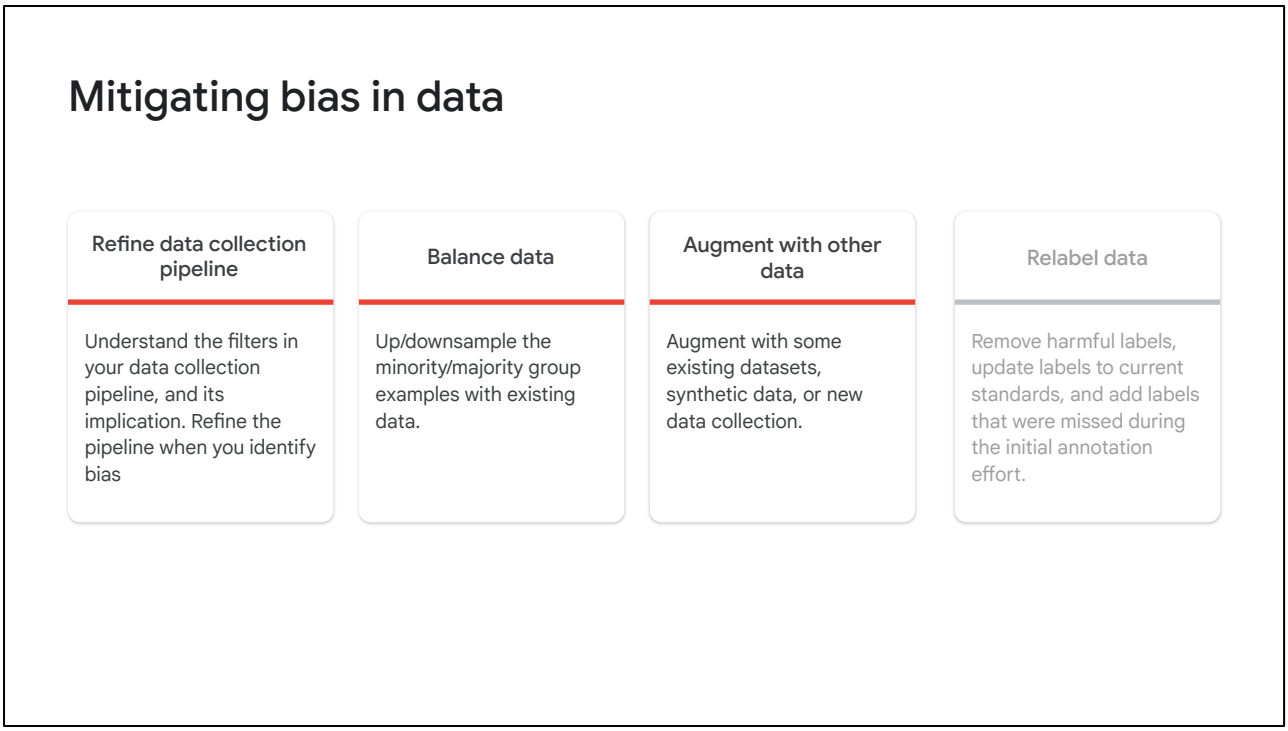


- Refine your data collection pipeline when you identify bias, especially when associated with data filters in the pipeline.

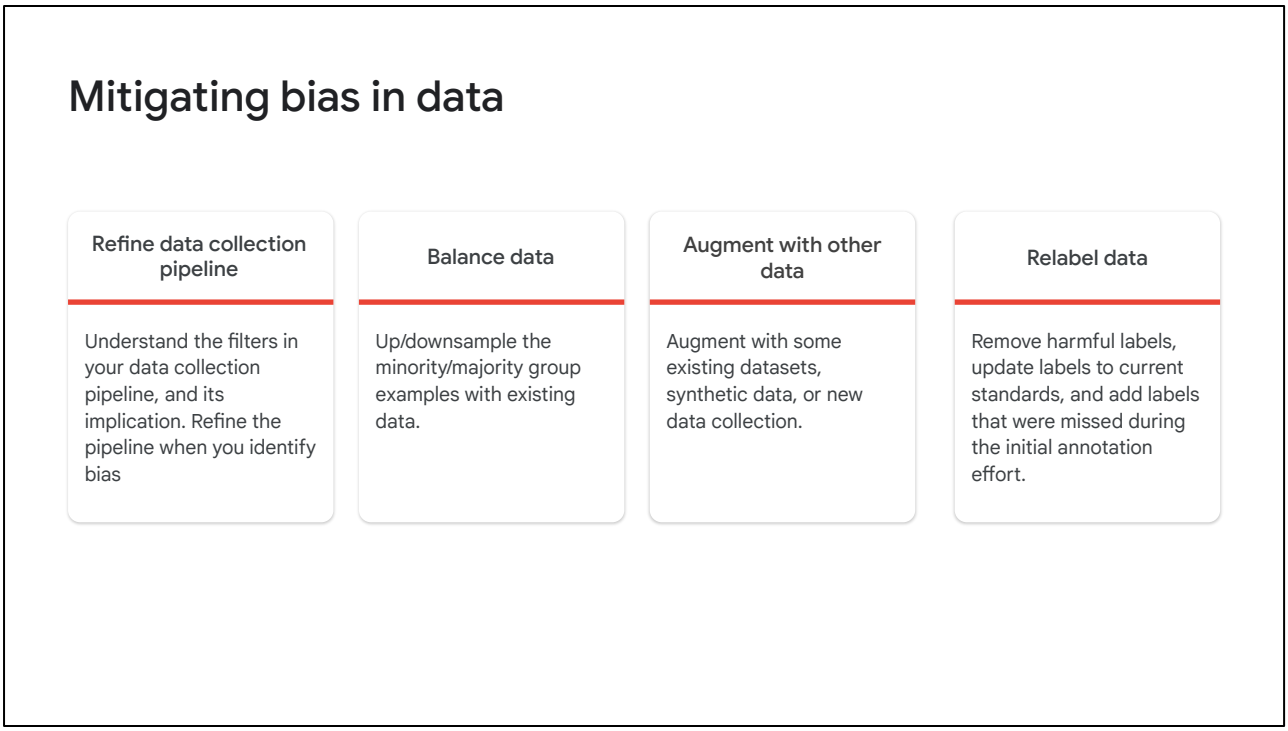
Mitigating bias in data



- Resample the dataset to balance data.



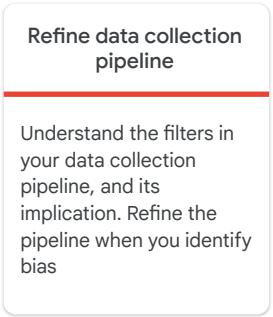
- Augment with more data, like other existing datasets, synthetic data, or new data collection.



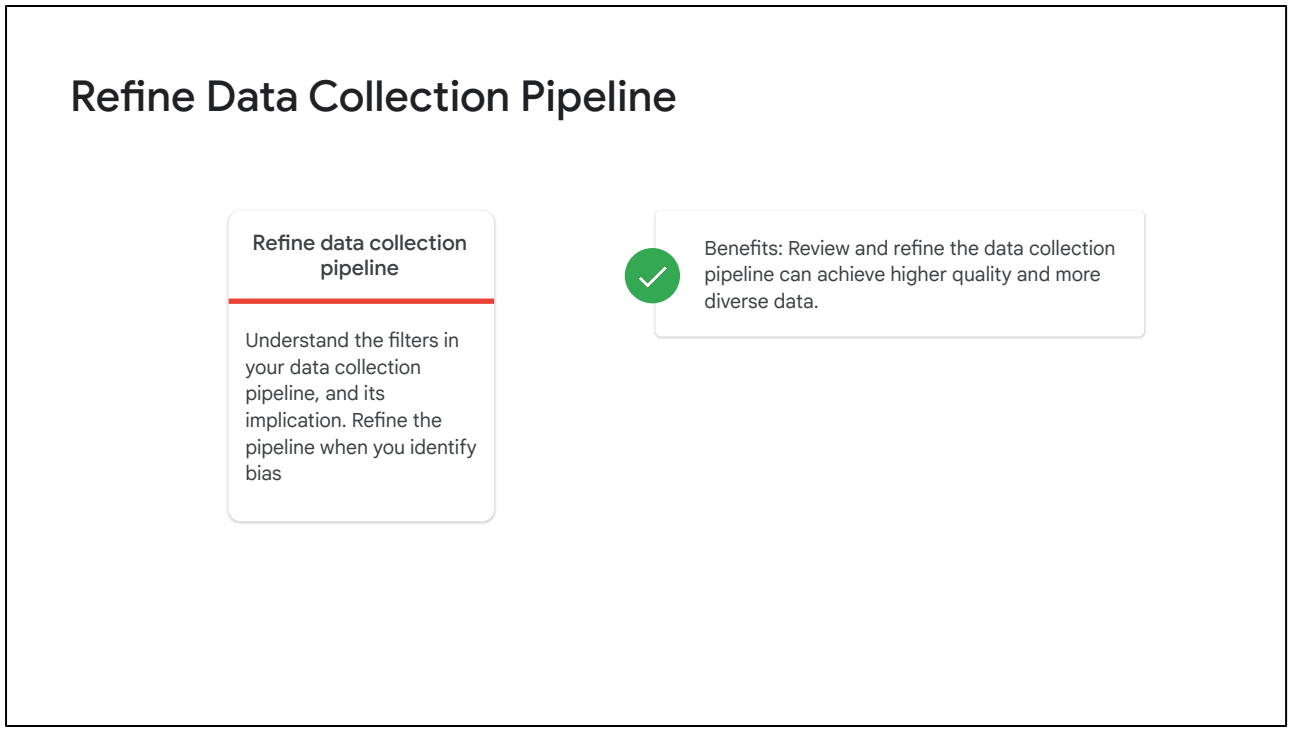
- Relabel your data by removing harmful labels, updating labels to current standards, and adding labels that were missed during the initial annotation effort.

Let’s dive into each of the techniques mentioned.

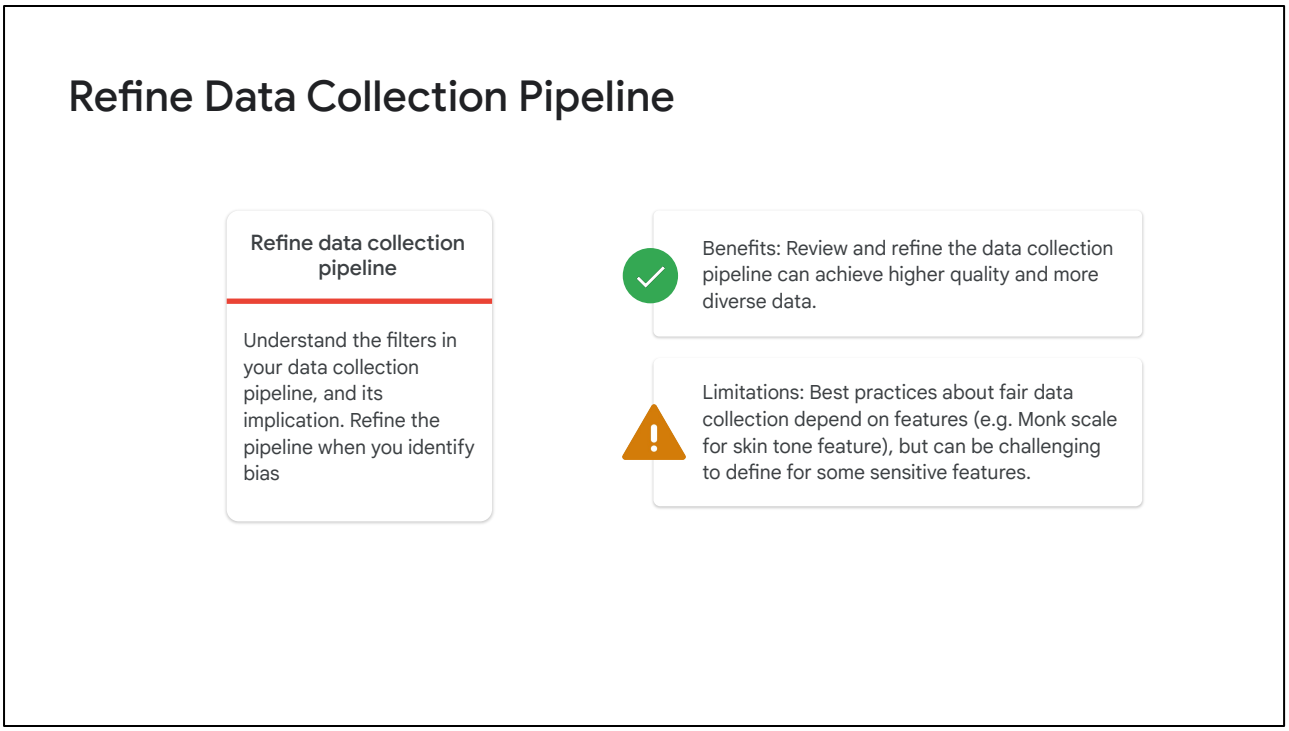
Refine Data Collection Pipeline



First, review and refine the pipeline when you identify bias in your data.
When there are filters to include or exclude data in your data collection pipeline, ensure that you think through the implications of what is being filtered in or out, both intended and unintended.
Every time you add a filter to your data, you heavily bias it in some way.

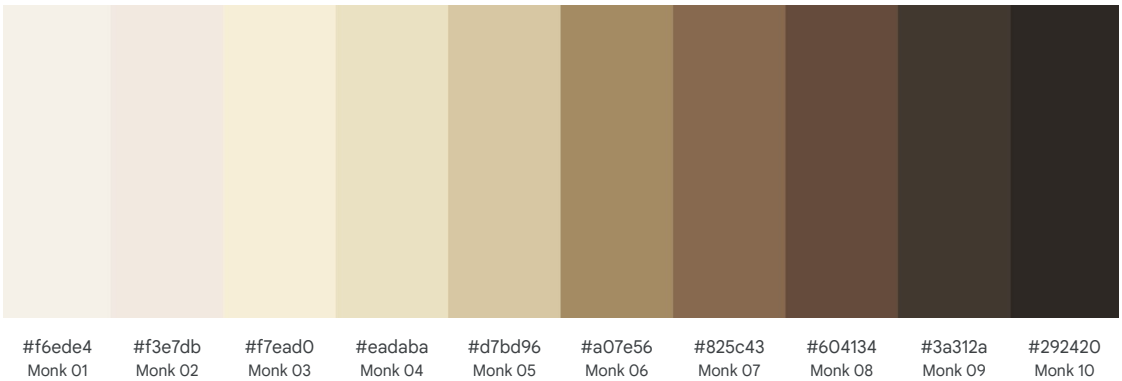


Refining the pipeline can help achieve higher quality and more diverse data.



There are also limitations on refining the pipeline. Best practices about fair data collection depend on features.

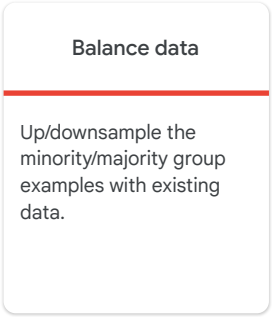
Monk Skin Tone (MST) scale



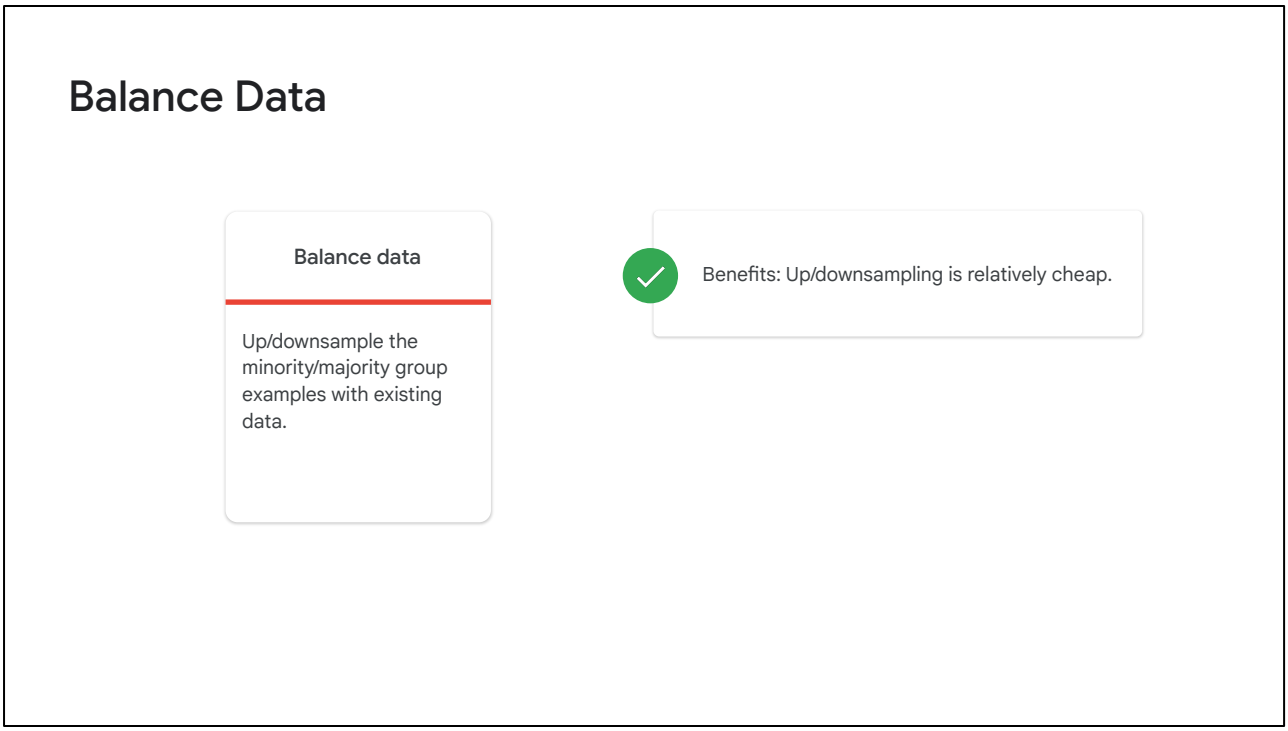
For example, if you want to collect a fair human face image dataset, you can use the Monk Skin Tone (MST) scale. Developed by Dr. Ellis Monk, in partnership with Google, the MST scale provides a 10-shade scale of skin tones that can be used to evaluate datasets and ML models for better representation.

However, this kind of best practice might not be available for some sensitive features, and it can be challenging to define it from the beginning.

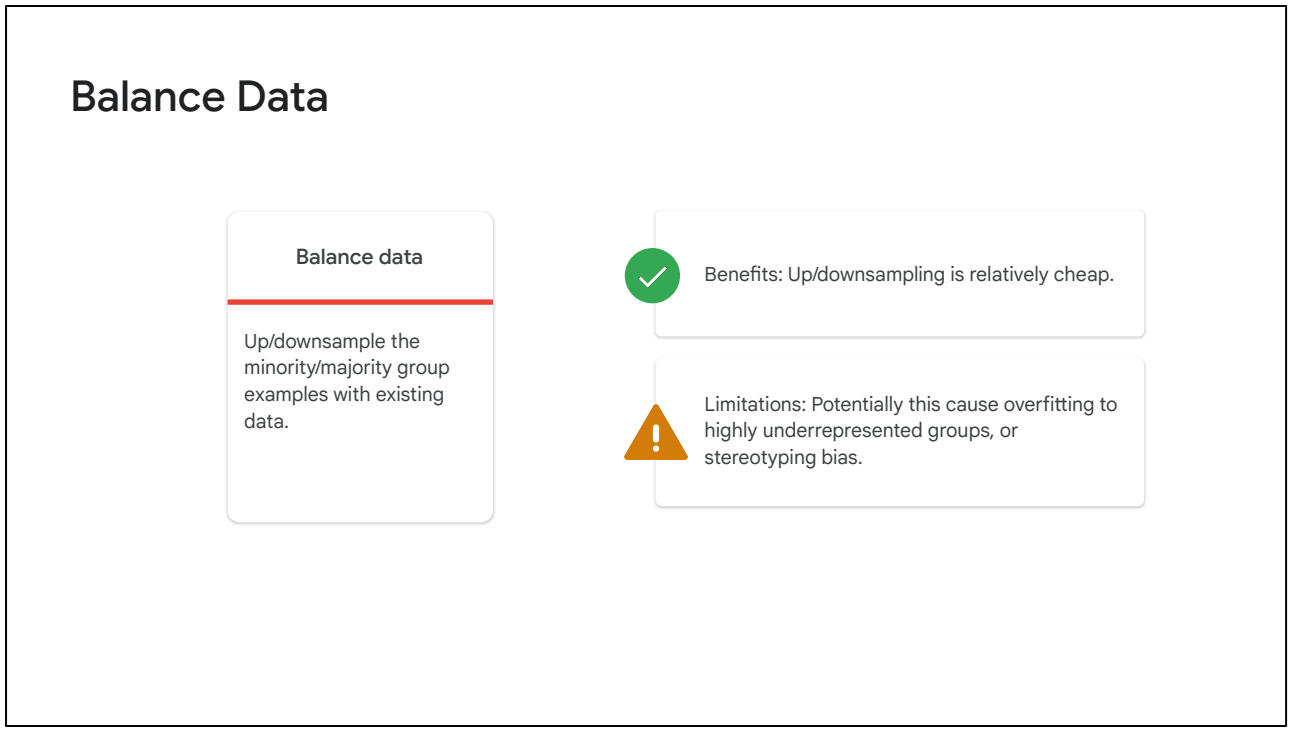
Balance Data



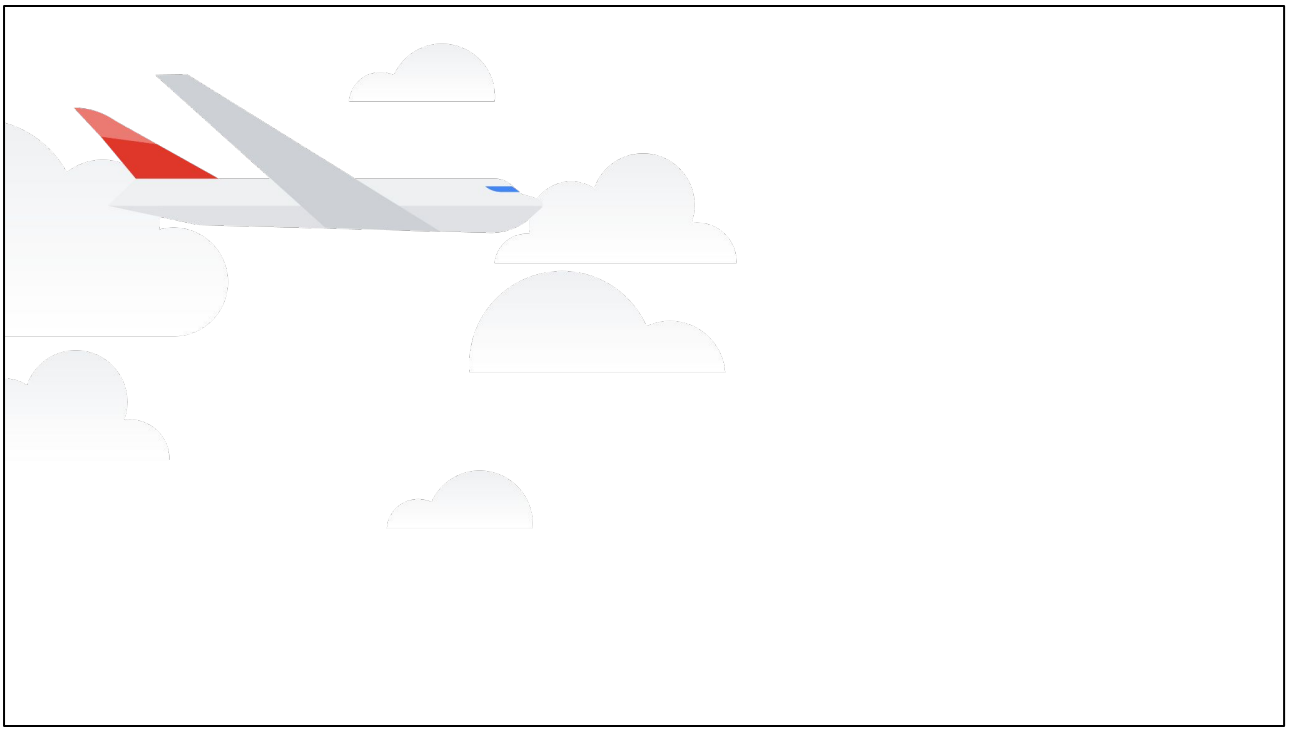
You can balance the data by resampling the dataset. For example, you can upsample or downsample the minority or majority group examples with existing data. Resampling can occur at various stages of the ML workflow and often depends on the use of the dataset.



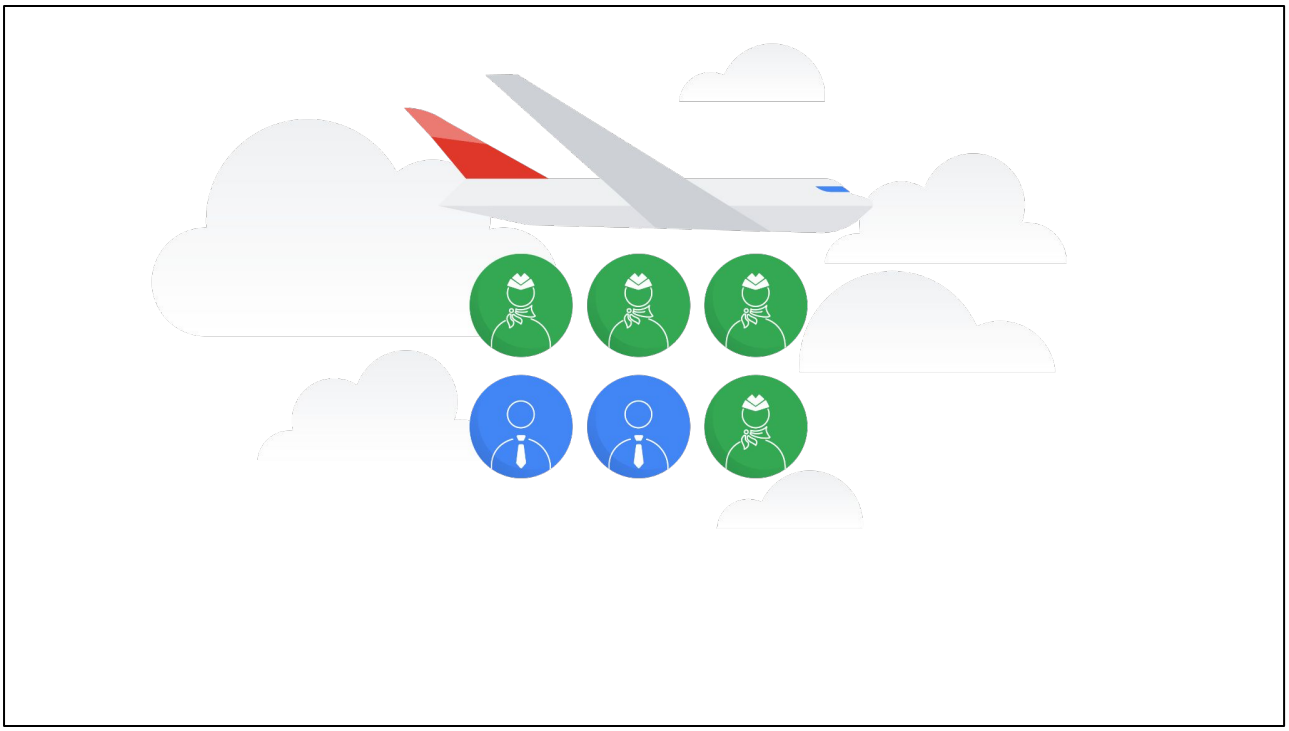
The benefit of this technique is that it's relatively cheap. No new data is required if you are upsampling underrepresented groups.



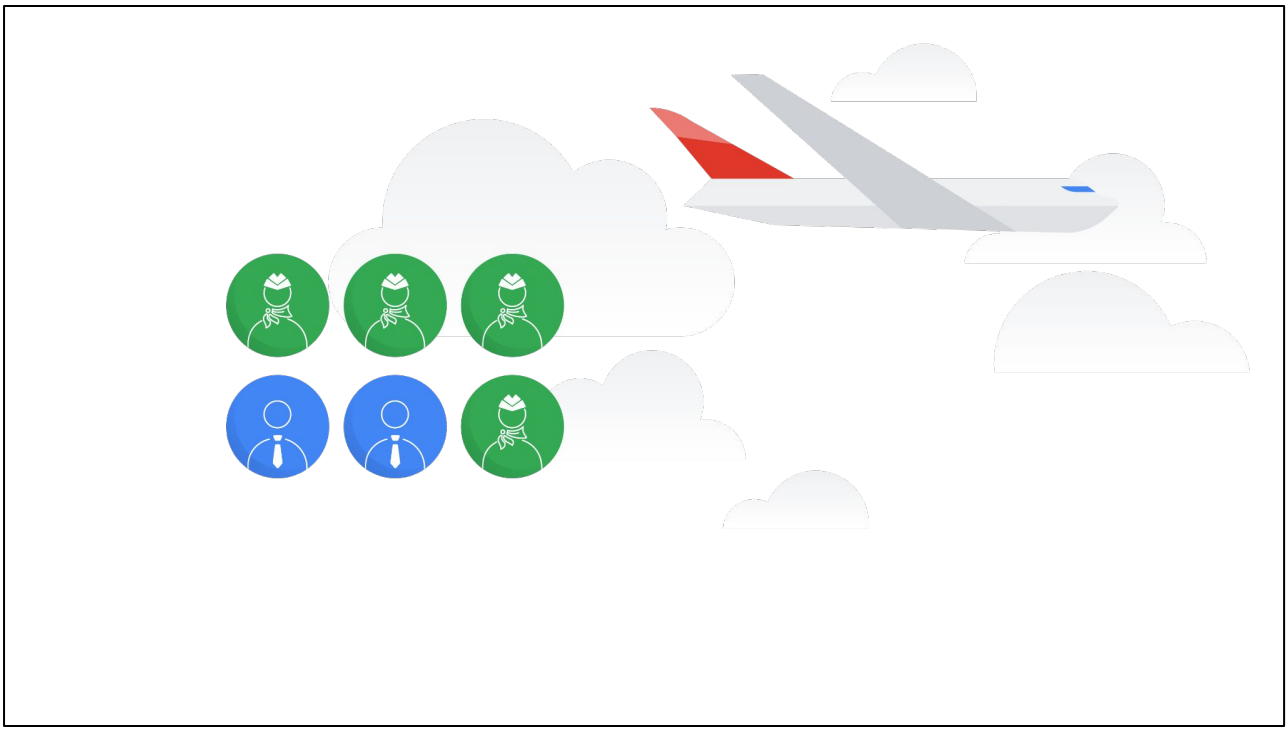
However, be aware of a limitation potential to overfit highly underrepresented groups, and pay attention to amplification leading to stereotyping bias.



Here is an example of balancing data.

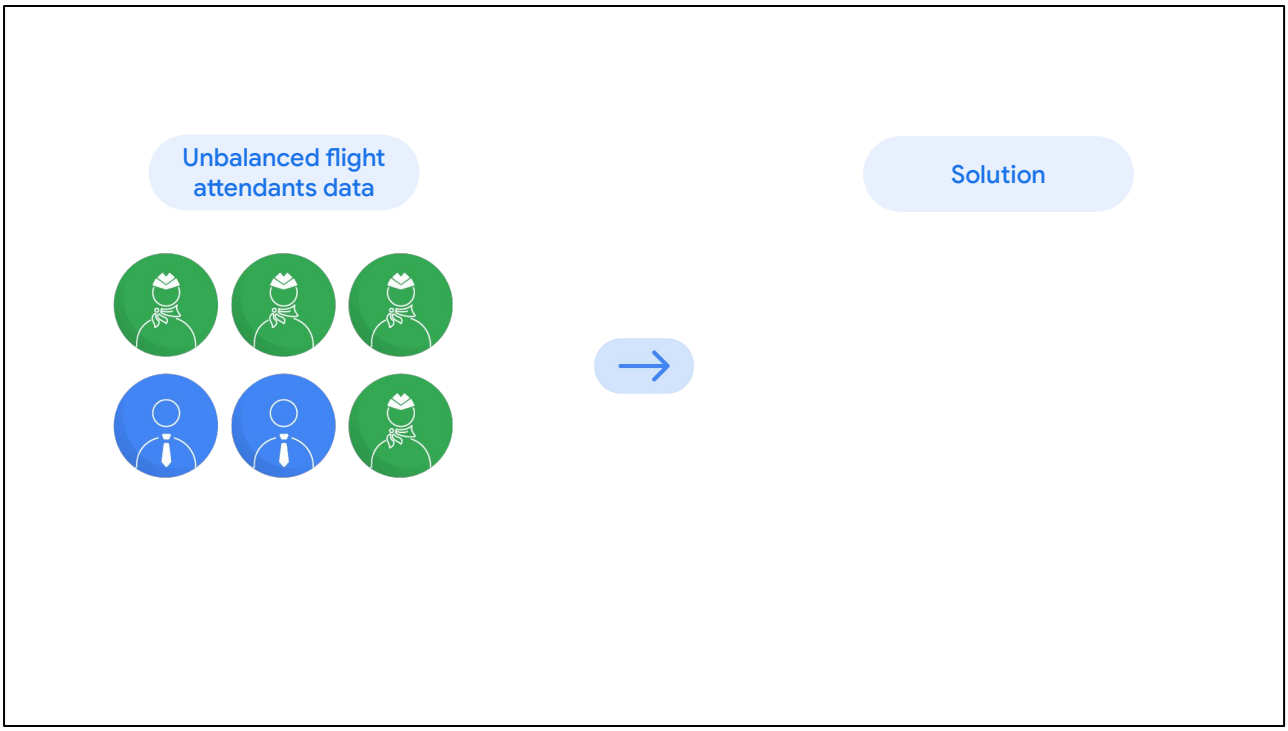


Imagine you are training a machine learning model using a dataset with flight attendant data.

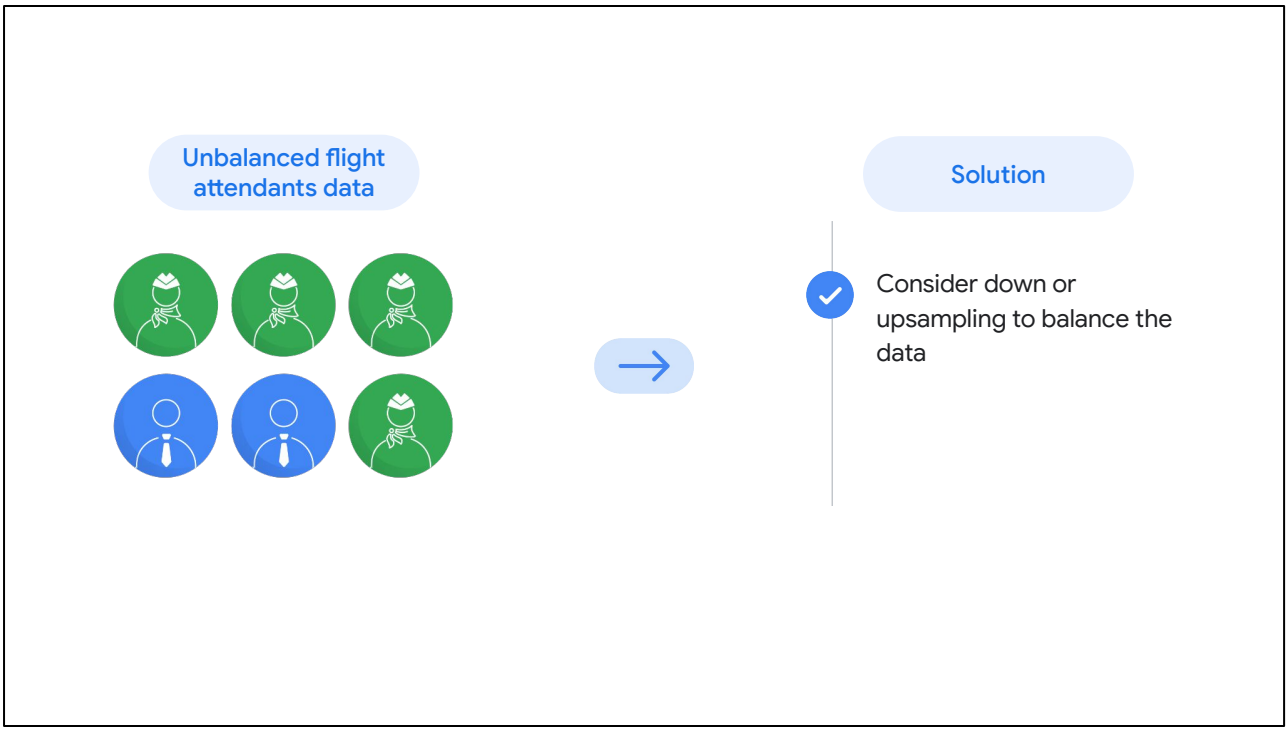


In this dataset,

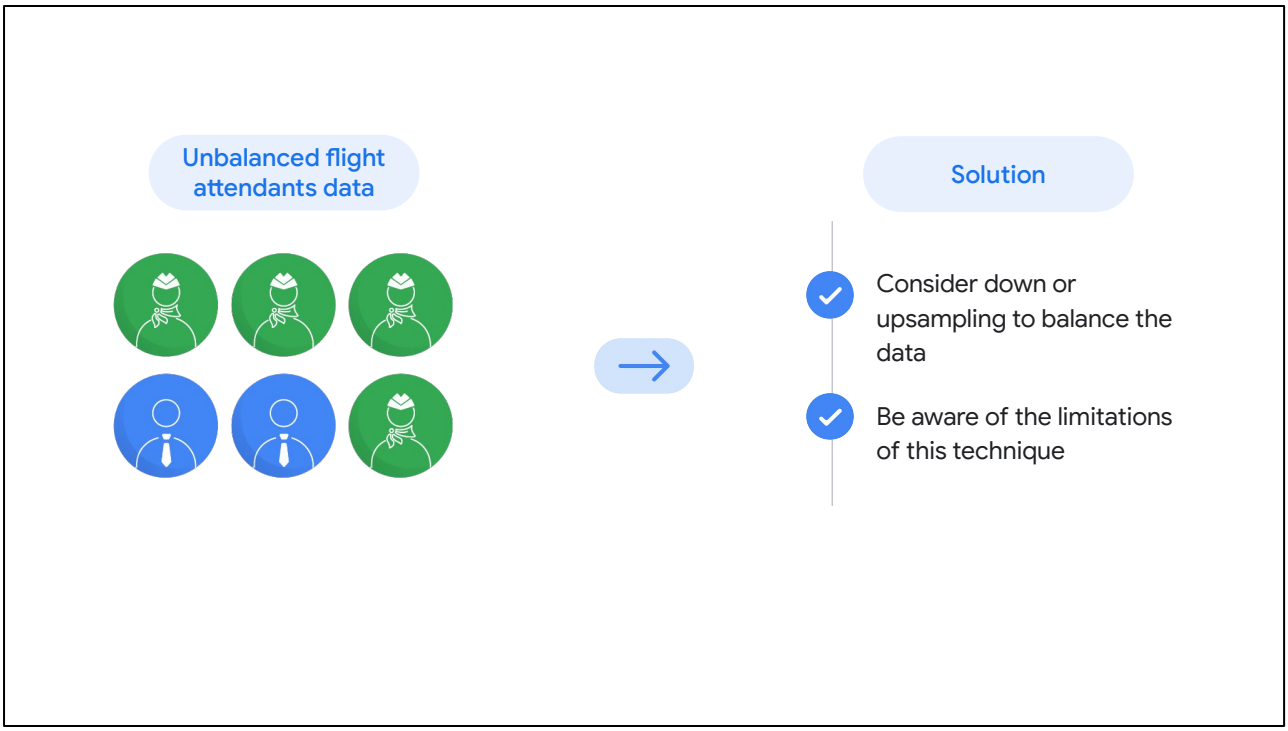
only a handful of male flight attendants exist in the data.



For this case,

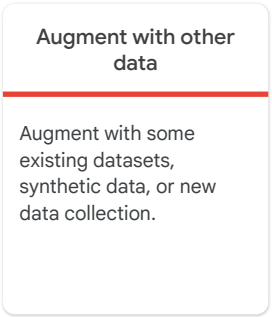


it would be beneficial to consider downsampling the female attendants group, or upsampling the other group.



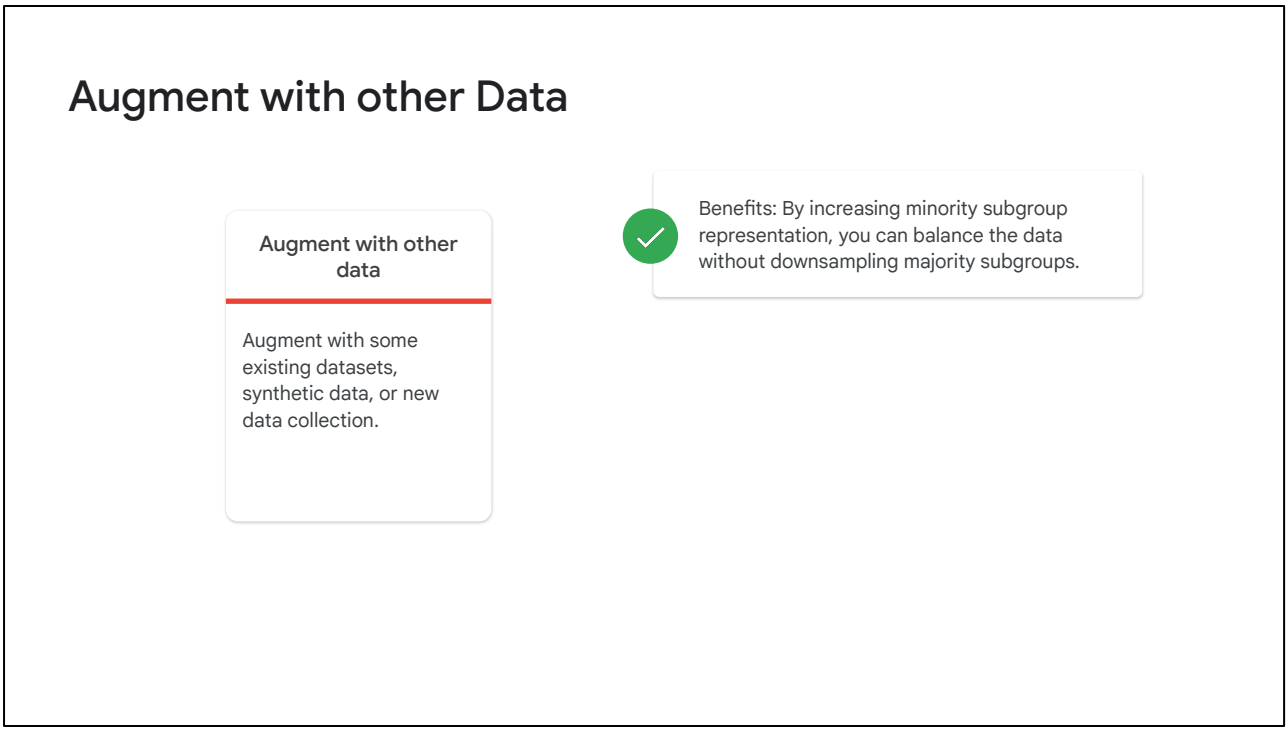
A limitation of this technique is that each individual example could be overrepresented. To offset this, it's recommended to explore and collect more data for the minority group, and augment with data.

Augment with other Data

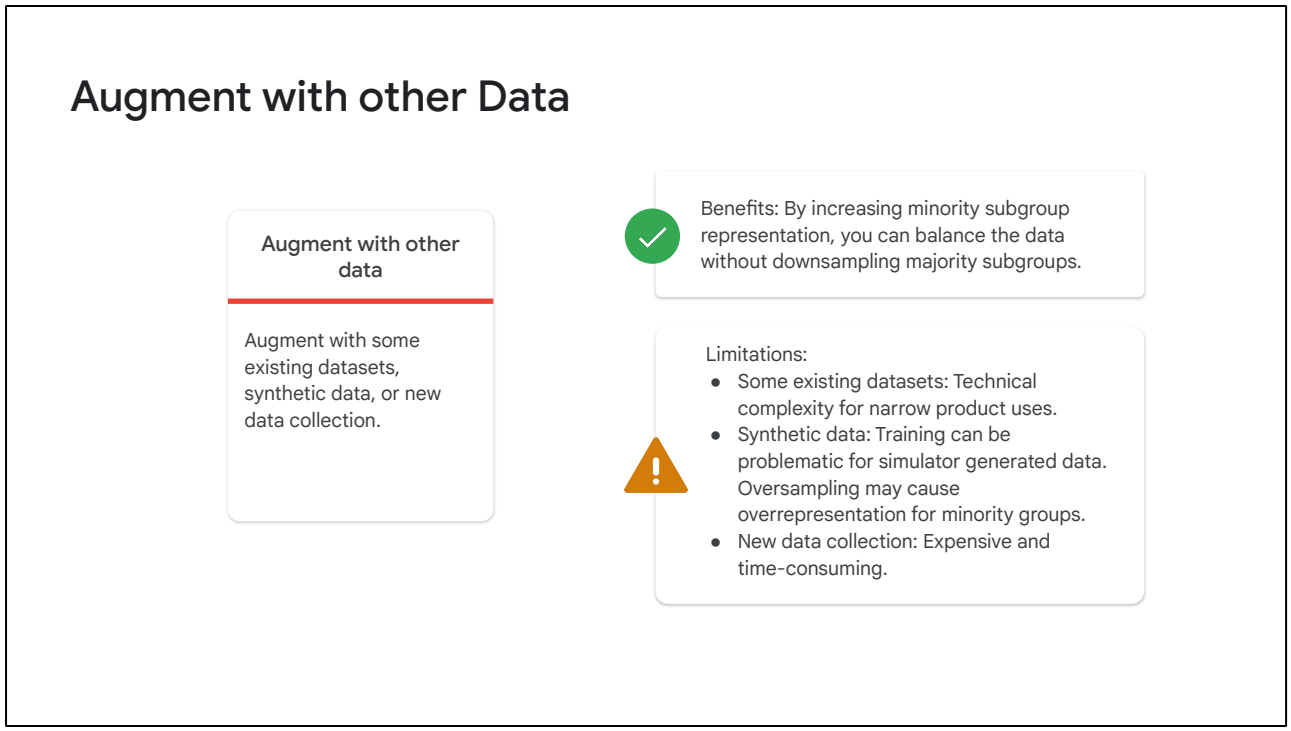


Augmenting with other existing datasets, synthetic data or new data collection, is an important technique for mitigating bias in data. However, note that augmenting with data enlarges the data pool. Whereas balancing data by resampling, operates on the existing data.

And by collecting data from multiple datasets, you could increase representation of minority subgroups, or failure cases. To add new examples for underrepresented minority subgroups you can generate synthetic data. And if budget and time are not pressing concerns, you can also add new examples by collecting new data.

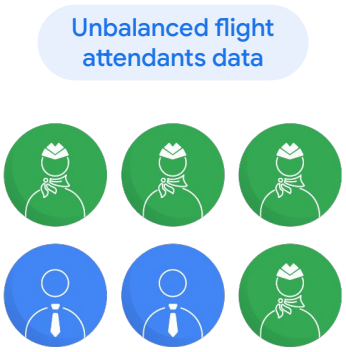


There are a few benefits to augmenting with more data. Additional data with improved fairness characteristics might be already available for widespread use. By increasing minority subgroup representation, you can balance the data without downsampling majority subgroups. And augmenting with synthetic data can be potentially less expensive than curating a new dataset to fill in these gaps.

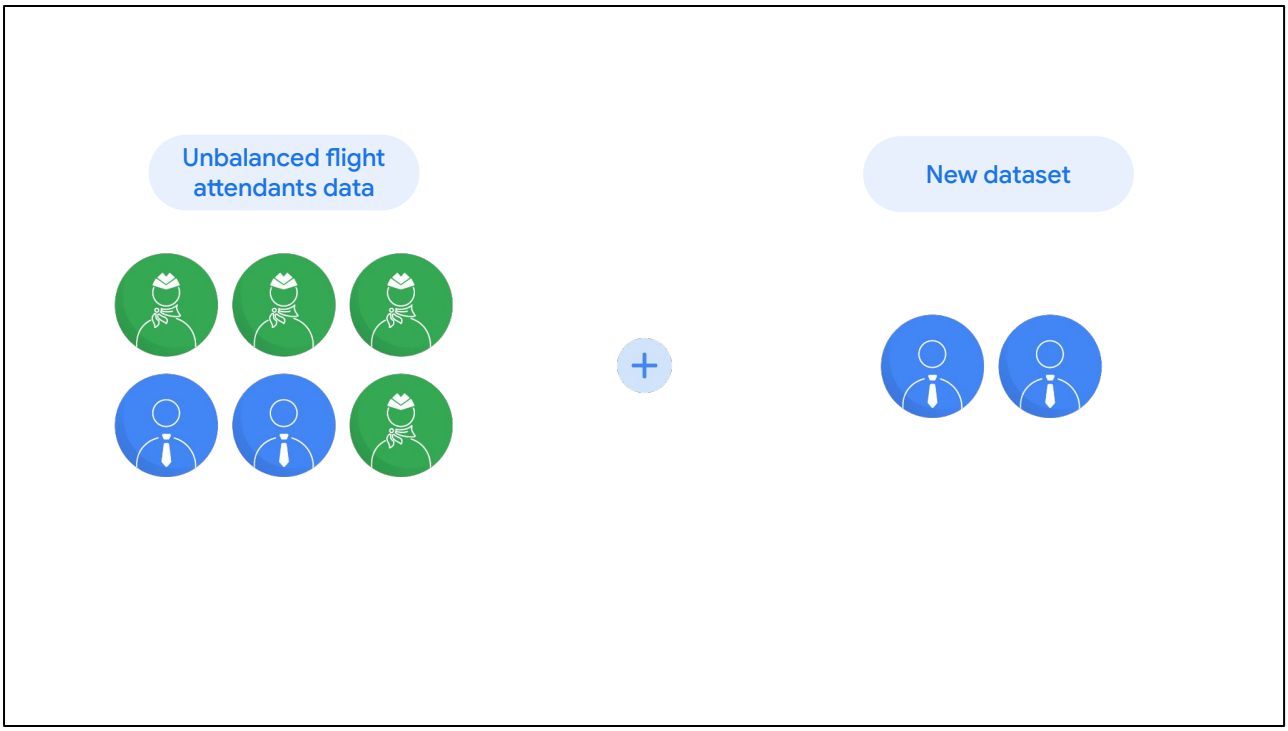


However, each of the techniques has its own limitations.

For narrow product uses with rare data attributes, usage of additional out-of-domain data may introduce technical complexity on the relevant ML tasks to overcome domain gaps.
Training on synthetic data can be problematic, especially for simulator generated data. For example, you should pay attention to whether the trained model is treating synthetic images differently from “natural” images, to ensure the model doesn’t “cheat” by simply changing scores for synthetic images. And oversampling might cause overrepresentation for the minority group, like upsampling.
Of all the listed techniques, new data collection is the most expensive and time-consuming.

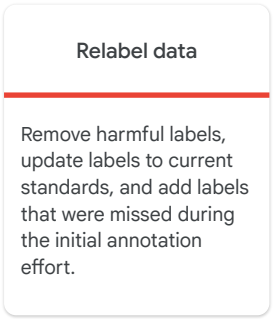


Coming back to our previous example, we could also try a different approach to improve representation and reduce bias in our training model, by augmenting our data with a new dataset with more male flight attendants.

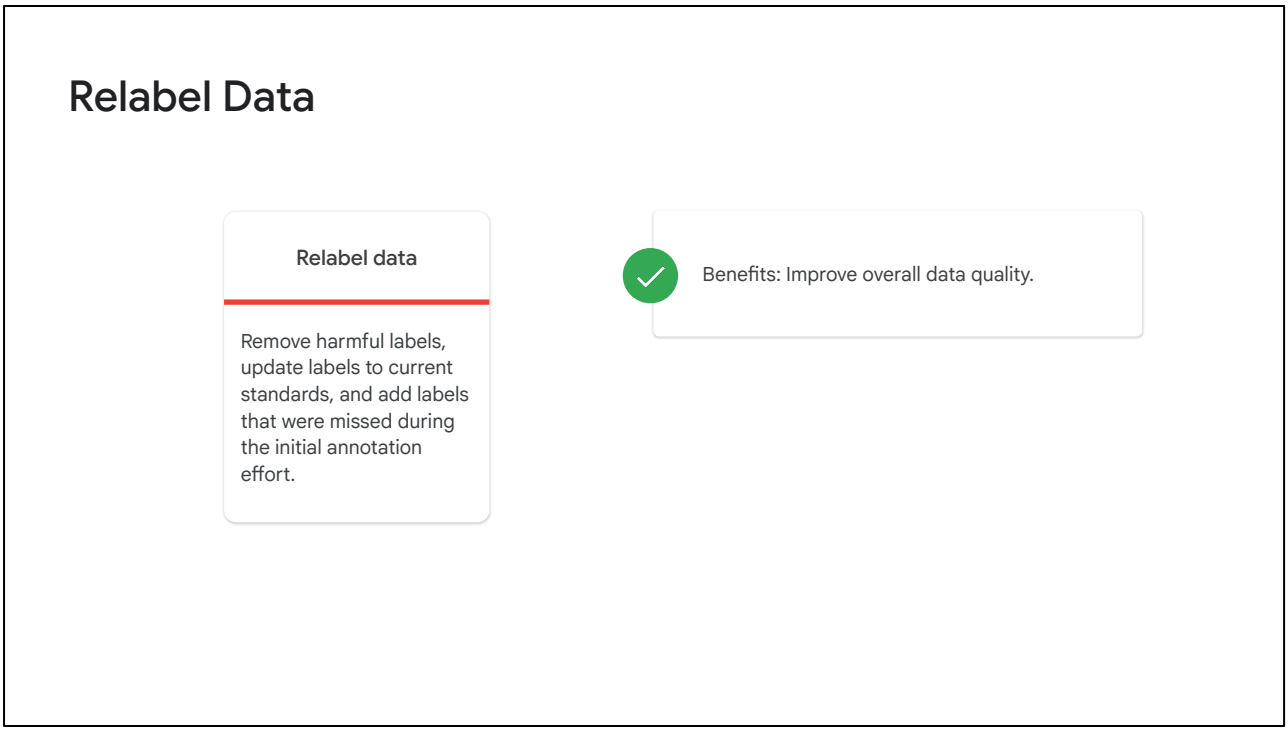


Coming back to our previous example, we could also try a different approach to improve representation and reduce bias in our training model, by augmenting our data with a new dataset with more male flight attendants.

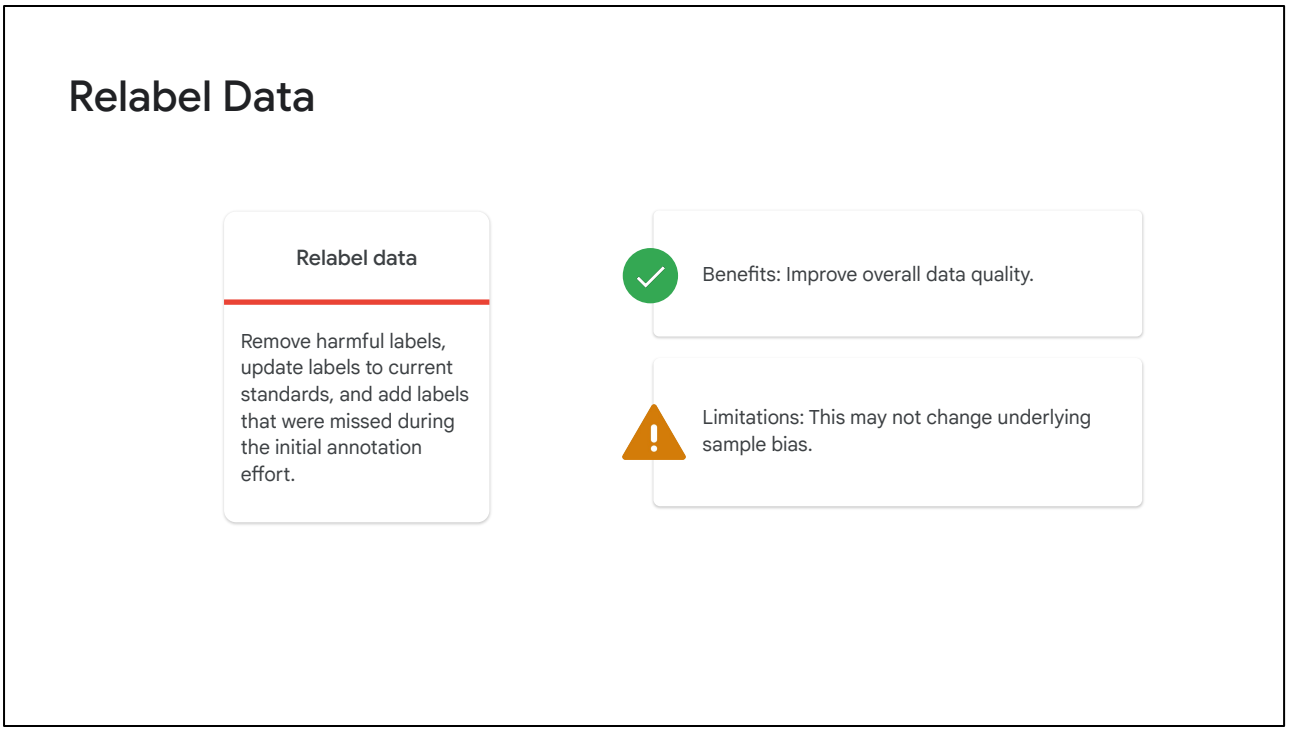
Relabel Data



Last but not least, relabel your data by removing harmful labels, updating labels to current standards, and adding labels that were missed during the initial annotation effort.



This method can improve overall data quality.



However, this may not change underlying sample bias.

Let’s take a look at a real-life example of how relabelling data can be used to reduce bias in machine learning models, is through sentiment analysis.

"This movie is a hilarious romp through the life of a group of stereotypical millennials. They're lazy, entitled, and obsessed with their smartphones. But despite their flaws, they're also loveable and relatable. The movie is full of laugh-out-loud moments that will have you rolling in the aisles. It's the perfect movie to watch if you need a good laugh."

Imagine you're training a machine learning model to classify movie reviews as positive or negative.

"This movie is a hilarious romp through the life of a group of stereotypical millennials. They're lazy, entitled, and obsessed with their smartphones. But despite their flaws, they're also loveable and relatable. The movie is full of laugh-out-loud moments that will have you rolling in the aisles. It's the perfect movie to watch if you need a good laugh."

Sentient Analysis pass leads to negative review.

Your training data might contain reviews with biased language, such as using stereotypical phrases, or making generalizations about certain groups of people.

"This movie is a hilarious romp through the life of a group of stereotypical millennials. They're lazy, entitled, and obsessed with their smartphones. But despite their flaws, they're also loveable and relatable. The movie is full of laugh-out-loud moments that will have you rolling in the aisles. It's the perfect movie to watch if you need a good laugh."

Sentient Analysis pass leads to negative review.

Human review changes to positive review, trains model.

To reduce bias, you could relabel some of the reviews by having human experts carefully review them, then assign more neutral labels. This results in helping the model learn to associate sentiment with the actual content of the reviews, instead of biased language.

01 Overview of Fairness and Bias

02 Identify Bias – TFDV Tool

03 Identify Bias – What-if Tool

04 Identify Bias – TFMA Tool

05 Mitigate Bias – Data Intervention

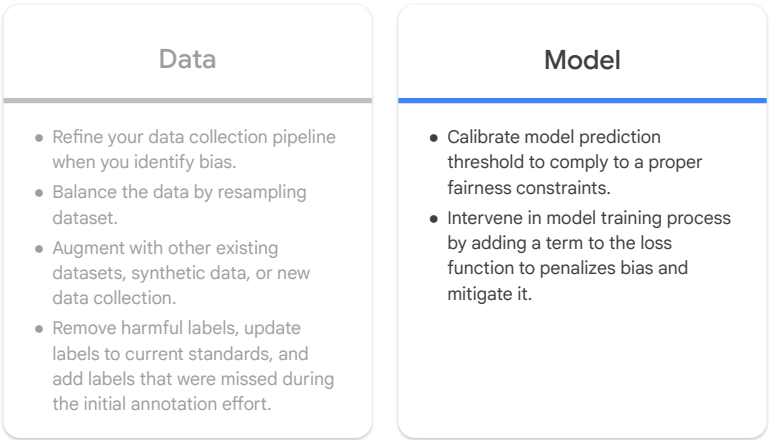
06 Mitigate Bias – Threshold Calibration

07 Mitigate Bias – Model Remediation

08 Lab: Mitigate Bias with MinDiff in TensorFlow

Now that we have learned how to mitigate bias from the data,

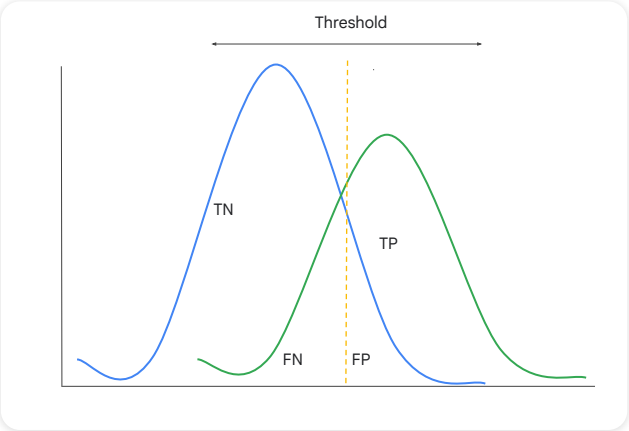
How can we mitigate biases in machine learning?



let’s see the techniques intervening machine learning models use to mitigate issues in different ways.

Threshold Calibration:

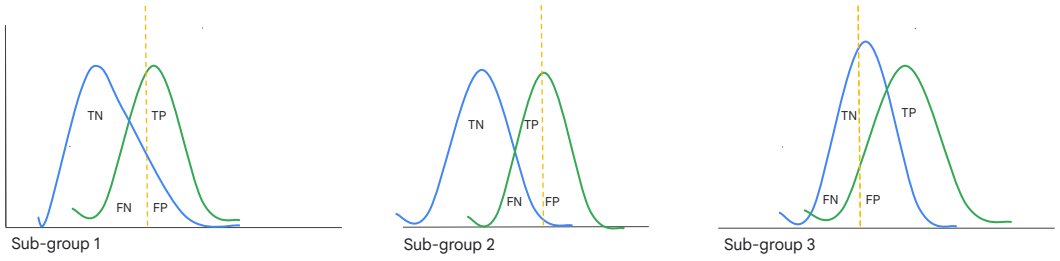
Find the threshold that brings the fair result



First, even after training and fixing a machine learning model, there is still room to control classification behavior, by calibrating thresholds.
Threshold calibration defines the proportion of the prediction results by categorizing them into True Positives, False Positives, False Negatives, and True Negatives.

After you determine these metrics on the entire dataset, ensure that you go one step further by calculating the metrics across the different subgroups within your data.

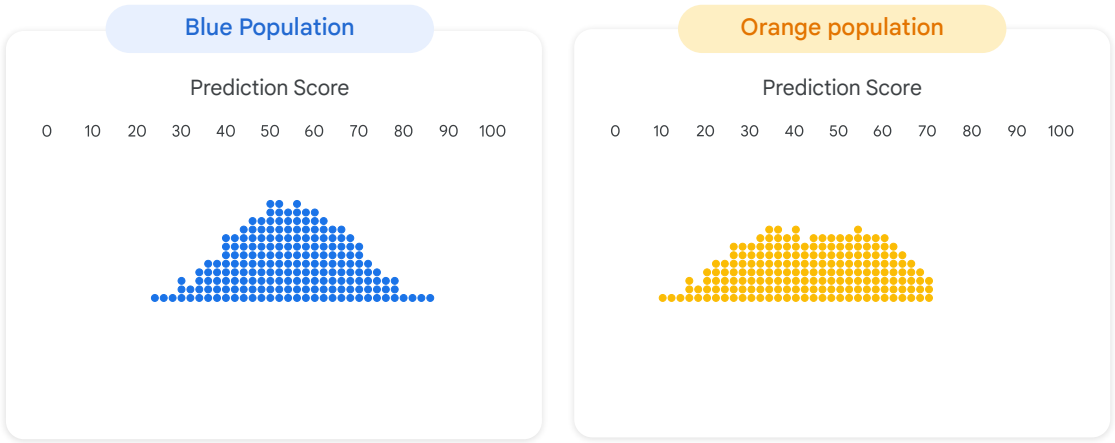
For example, you might find that a false negative rate at zero point one (0.1) is acceptable for the problem you're trying to solve with your machine learning system. Given that overall rate, how does that rate look across the subgroups?



As shown in these plots, and after you compute your metrics, you can visualize the distribution of your evaluation metrics across each subgroup, as depicted by the blue and green colors. Each color represents a different subgroup within the data.

As you can see, predictions on each subgroup result in different proportions. That could mean that the best and fair threshold can vary among different subgroups. By keeping this in mind, you are one step closer to identifying ways in which you could make your machine learning system more inclusive.

Fairness in loan approvals...

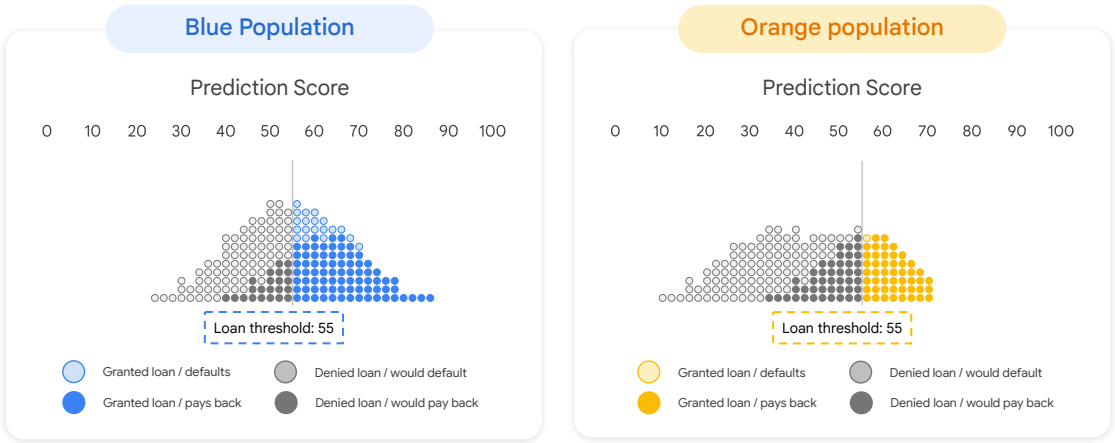


To help you think through the thresholding from fairness purpose, and to highlight what happens when you change variables, let's look at an example where a machine learning model decides whether a bank approves or rejects loan applications.

Imagine that we have two groups of people, identified here by the colors blue, and orange. Notice how the prediction score distributions are different between the two groups.

Fairness constraint 1:

Group unaware—everyone held to the same threshold



Let's say we set the same threshold for the two groups at 55.

With the same threshold, the blue and orange groups have different distributions in how they'll be able to pay back loans. However, the differences in group distribution and loan thresholds are a result of historical and systemic biases that have somehow made it into our data. Because of this, it might be unfair to ignore the differences between the two groups.

Let's examine the distributions of these loan thresholds a bit more.

- The light gray color represents those who are denied loans as they would default anyhow.
-
- The dark gray color represents those who are denied loans, but won't default.
-
- The light blue and orange colors represent those who are granted loans, but default.
-
- And the dark blue and orange colors represent those who are granted the loan, and pay it back.
-

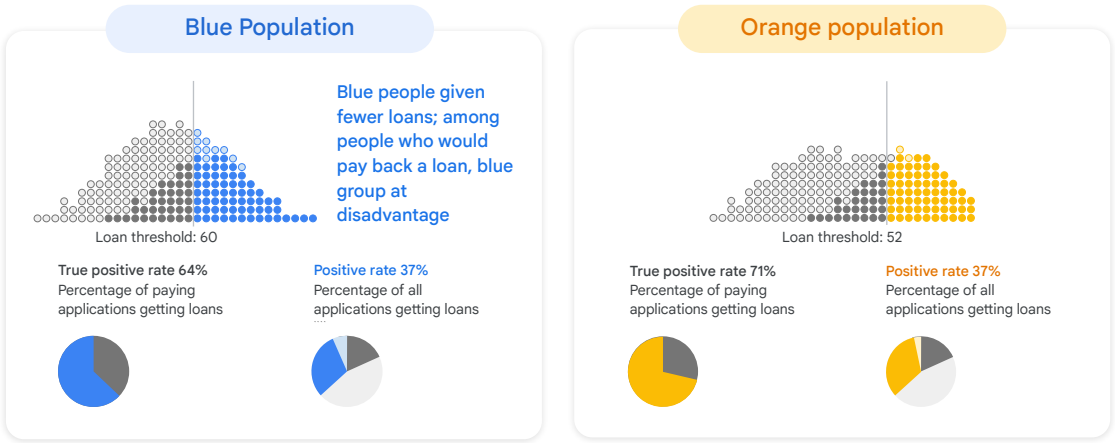
Examining these distributions between groups is important and very applicable in real life. For instance, in the United States, different race and gender groups might

have different rates for paying back their loans due to wealth inequality, which is the result of historical discrimination. Here's another example. Women tend to live longer than men. Therefore, would it make sense to consider insurance rates and policies in the same way for both men and women? These two examples highlight the concept of group unaware, or, a simple threshold to set the same value across all groups.

Looking back at the plots on the screen, the blue group is now being given more loans overall, but the orange group has been given fewer loans. Among people who would pay back a loan, the orange group is at a disadvantage. This is because the real differences that exist between the two groups are being ignored.

Fairness constraint 2:

Demographic parity—same percentage of loans approved



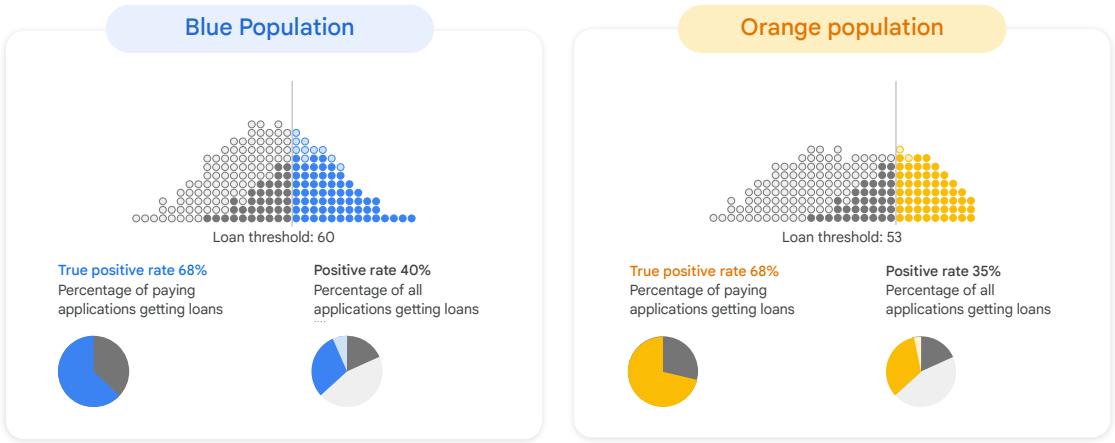
Now let's consider another fairness definition: demographic parity. Let's have the bank issue the same percentage of loans to both blue and orange groups.

What happens is that the ratio of loans given to each group becomes the same, but among the people who would pay back a loan, the blue group is now at a disadvantage. This is because the model only looks at loans given, not rates at which loans are paid back.

This demographic parity strategy is helpful in a use case where having equal final percentages of groups is important.

Fairness constraint 3:

Equality of opportunity—of the people who can pay, same percentage approved

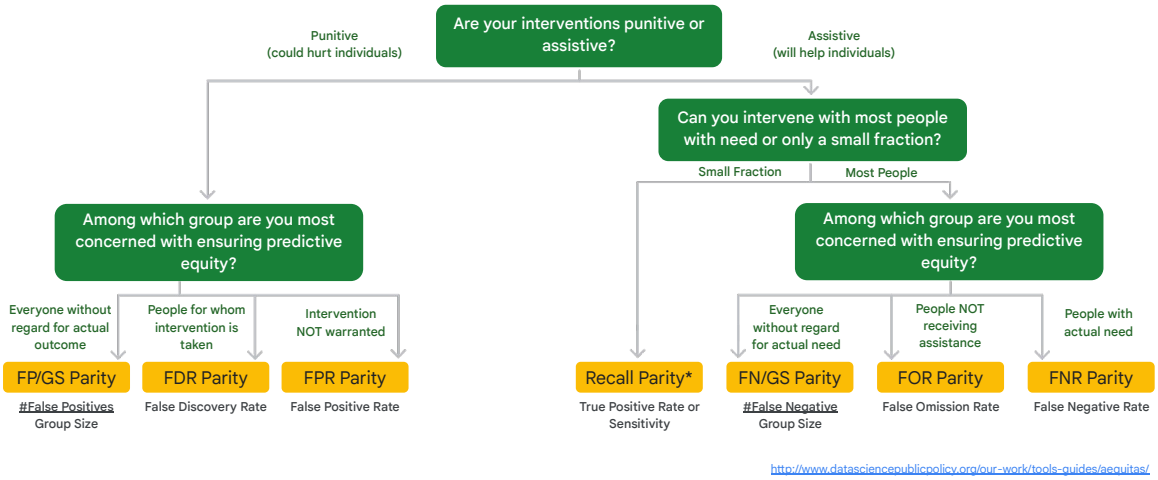


Here's a third fairness definition. Equality of opportunity.
This means offering the same percentage of loans to those who can pay them back in both groups. In other words, set the same recall.

Now you can see that among people who would pay back a loan, blue and orange groups do equally well.

When to use what?:

Check Aequitas Fairness Tree guideline



http://www.datasciencepublicpolicy.org/our-work/tools-guides/aequitas/

Throughout this loan example, we looked at different constraints to determine the level of fairness when issuing loans to two different groups.

Each of these constraints has different mathematical properties and different outcomes. One is not necessarily better than the other. You also can't satisfy all properties and objectives at the same time.

And these are not the exhaustive list of the constraints. And, of course, which constraint you choose depends a lot on the context of the ML use case.

The Aequitas Fairness Tree Guideline offers you guidelines about which metrics you should prioritize on a specific use case. The decision tree includes conditions, such as if positive predictions assist or punish people, and if your system can intervene with most people with need, or only a small fraction.

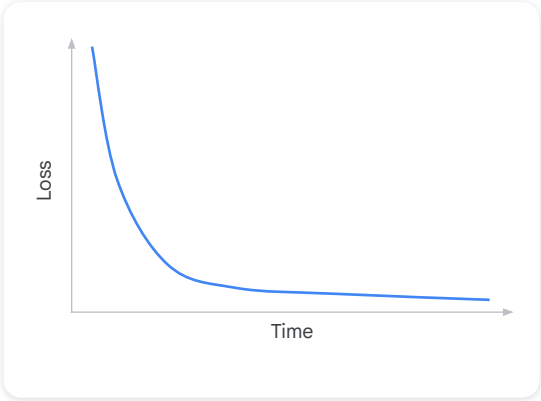
Threshold calibration is an easy way to adjust model output to a more desirable way, without changing the model itself.

What if we also want to intervene in the model training process to actually train a fairer system?

TensorFlow Model Remediation provides two methods: MinDiff and CLP

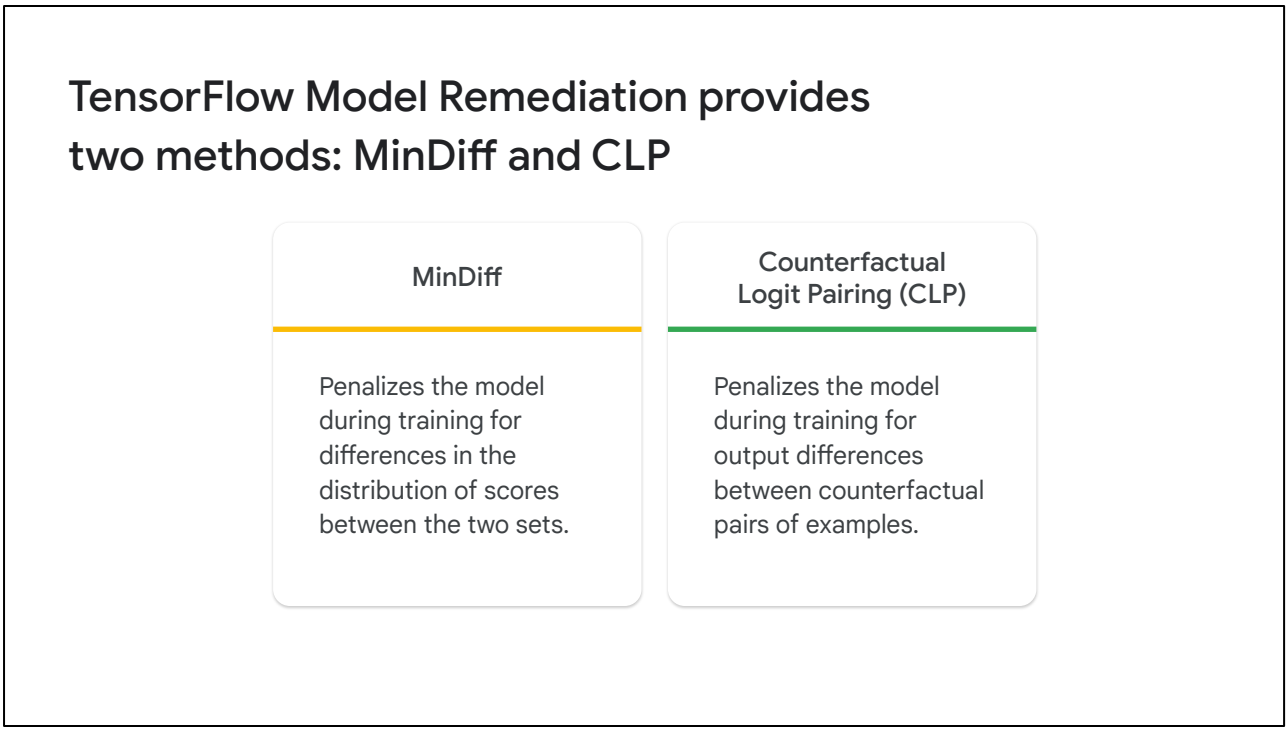
Machine Learning aims to minimize loss during training.

Why not adding a penalty term that represent a magnitude of bias?



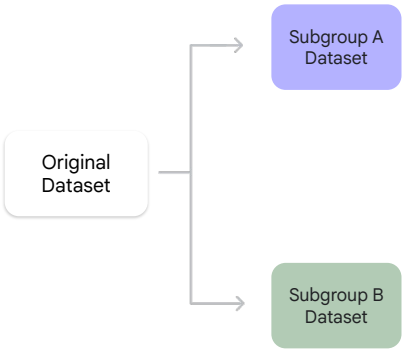
Remember that neural network models are trained to minimize loss values. And if this is the case, does this mean we can change the training process itself by changing the loss function?
The answer is yes! We can consider adding a penalty term that represents the magnitude of bias in some way, so that the training process can try to minimize not only the model performance but also biases.

It is similar to the idea to reduce model complexity by adding regularization terms like L1 or L2 norms.



There could be many ideas about this additional term for fairness. Here are two methods which are covered by the Tensor Flow Model Remediation library: MinDiff and Counterfactual Logit Pairing. The two methods have different ideas and goals about fairness, but both add a supplemental term to original loss functions.

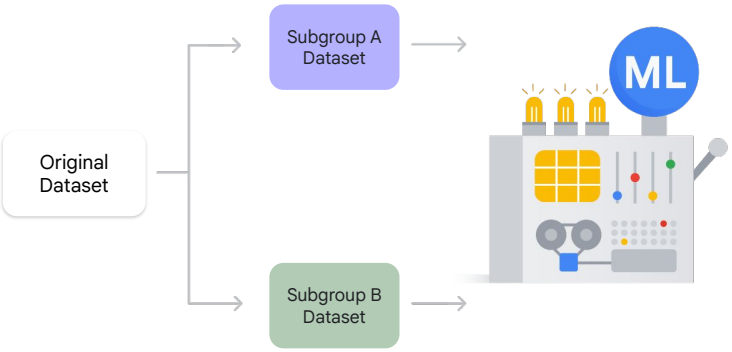
Applying MinDiff to Improve Model Fairness



Let's start with MinDiff method.
MinDiff focuses on the distribution difference of model prediction with respect to different subset of dataset.

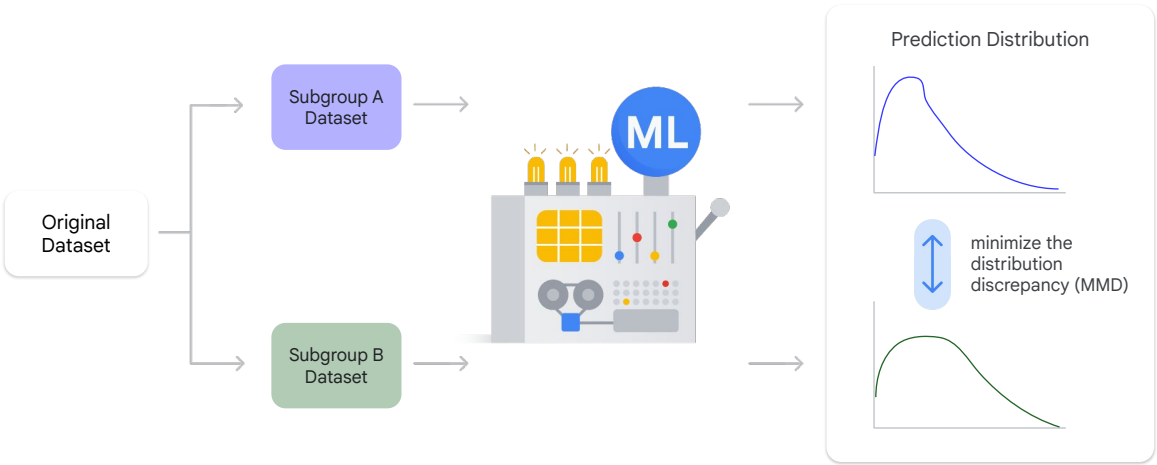
Let's say we are training a model with a dataset that has a sensitive feature, such as gender group or racial group. After training, we split the dataset into two: subgroup A and subgroup B by using the sensitive feature,

Applying MinDiff to Improve Model Fairness



and passed them to the model to get predictions.

Applying MinDiff to Improve Model Fairness



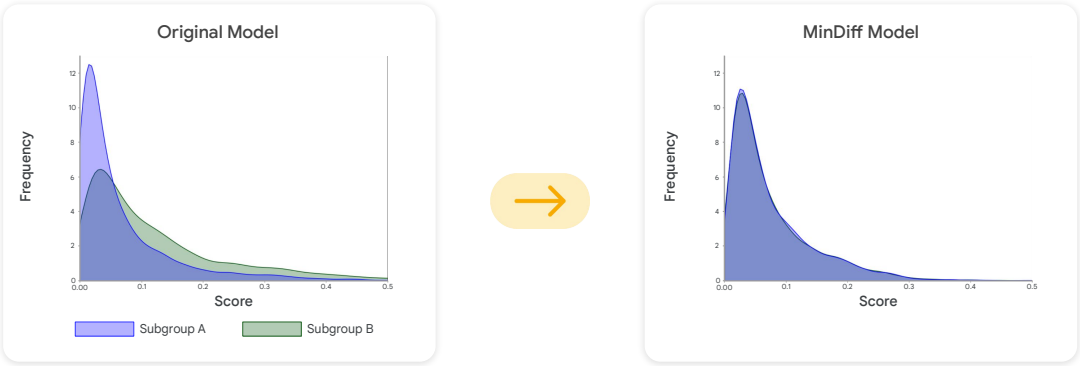
From this, we can then depict different prediction distributions on these two datasets.

The idea of MinDiff is, If the model is fair and robust enough, the output distribution of different subsets shouldn't be very different. Now in some situations, a sensitive feature reflects a social context related to the prediction objective, and the model ends up capturing the pattern difference. This then becomes reflected in the distribution of the predictions.

MinDiff uses additional loss terms to minimize MMD

MinDiff term

$$Loss(X, Y) = L_{primary}\,(f(X), Y) + \lambda MMD(f(X_0), f(X_1)|Y = 0)$$



It does this by adding a penalty term that represents the discrepancy between two distributions by using Mean Maximum Discrepancy. The term is added to the original primary loss function, such as binary cross entropy loss, and minimized in the training phase. The balance between the primary loss term and this MinDiff term can be controlled with lambda value.

With this term, we can expect the output distributions would be close, compared to the original model without MinDiff.

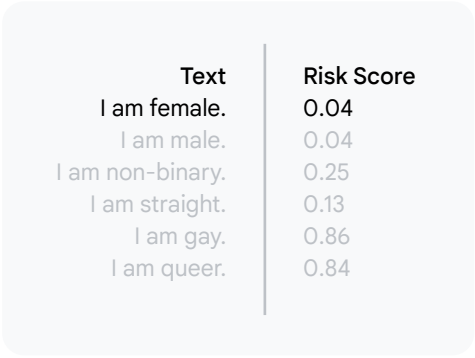
Counterfactual testing can help us check model bias

How would the prediction change if the **identity token** were something else?

Let's move on to the next method: Counterfactual Logit Pairing or CLP. We already discussed the term counterfactual. In counterfactual analysis, we try to identify bias issues by modifying a sensitive feature, then checking how it affects model predictions. For example, let's say we are creating a machine learning model to validate the toxicity of text sentences, which Google's jigsaw team is actually working on and published as a Perspective API.

Counterfactual testing can help us check model bias

How would the prediction change if the identity token were something else?



Using this as an example, if a machine learning model returns results that contain toxicity in text sentences, it will be regarded as unfair and harmful to some identities. Although the sentence structure is the same, and the identities themselves are neutral, the risk scores are different when an identity token is changed. Again, because machine learning captures patterns from datasets, if harmful data patterns exist, it will be reflected in the result. And in this case, some identity terms could be unfairly linked to negative scores.

Counterfactual analysis provides us this kind of information by setting up a proper “What-if” scenario; it helps us identify if the model's predictions are unfairly influenced by the presence of these identity terms, even when they are used in a neutral context.

Counterfactual testing can help us check model bias

How would the prediction change if the identity token were something else?

Text	Risk Score
I am female.	0.04
I am male.	0.04
I am non-binary.	0.25
I am straight.	0.13
I am gay.	0.86
I am queer.	0.84

Counterfactual testing can help us check model bias

How would the prediction change if the identity token were something else?

Text	Risk Score
I am female.	0.04
I am male.	0.04
I am non-binary.	0.25
I am straight.	0.13
I am gay.	0.86
I am queer.	0.84

Counterfactual testing can help us check model bias

How would the prediction change if the identity token were something else?

Text	Risk Score
I am female.	0.04
I am male.	0.04
I am non-binary.	0.25
I am straight.	0.13
I am gay.	0.86
I am queer.	0.84

Counterfactual testing can help us check model bias

How would the prediction change if the identity token were something else?

Text	Risk Score
I am female.	0.04
I am male.	0.04
I am non-binary.	0.25
I am straight.	0.13
I am gay.	0.86
I am queer.	0.84

Counterfactual testing can help us check model bias

How would the prediction change if the identity token were something else?

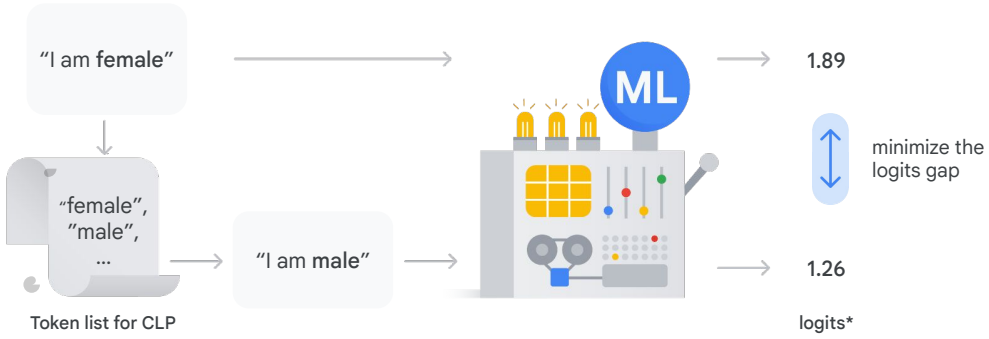
Text	Risk Score
I am female.	0.04
I am male.	0.04
I am non-binary.	0.25
I am straight.	0.13
I am gay.	0.86
I am queer.	0.84

Counterfactual testing can help us check model bias

How would the prediction change if the **identity token** were something else?

Text	Risk Score
I am female.	0.04
I am male.	0.04
I am non-binary.	0.25
I am straight.	0.13
I am gay.	0.86
I am queer.	0.84

Counterfactual Logit Pairing for Model Remediation



* logits: classification model's output before sigmoid/softmax function

CLP also focuses on this perspective.
The idea of this approach is "If a model is fair and robust enough, the output result won't be changed much even if we change a sensitive feature value".

Based on this idea, it adds an additional loss term that represents the gap of logits between original and counterfactual examples.
And remember that logits is a classification model's output before applying a normalization function, such as a sigmoid or softmax function.

Counterfactual Logit Pairing addresses counterfactual fairness like flip rate

CLP Loss

$$Loss(X, Y) = L_{primary}\left(f(X), Y\right) + \lambda CLP(g(x), g(x'))$$

$f(x) = \sigma(g(x))$

$g(x)$: model that produces a logit

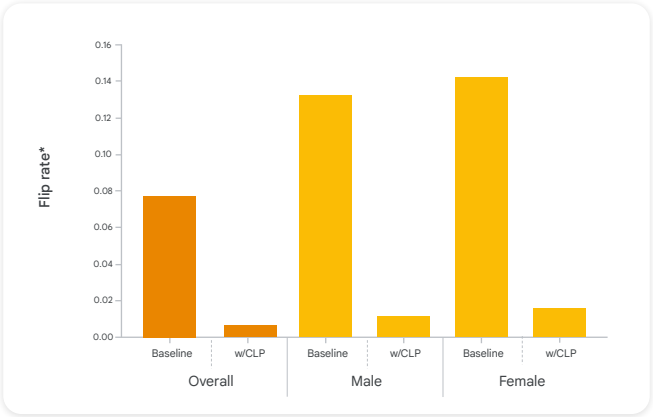
$\sigma(\cdot)$: sigmoid / softmax function

x : original data

x' : counterfactual data

By minimizing this new loss term, and the primary loss, we can expect the model would be more robust in terms of counterfactual fairness,

Counterfactual Logit Pairing addresses counterfactual fairness like flip rate



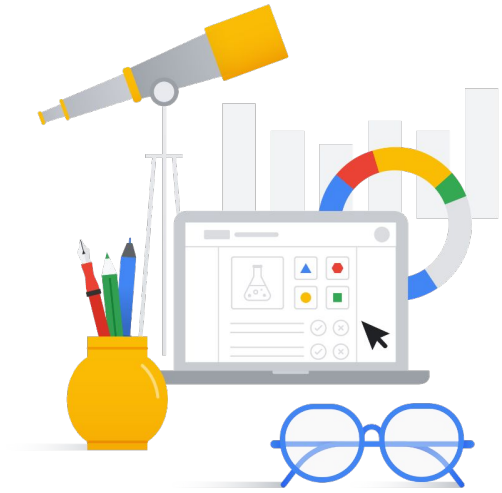
*** Flip Rate:** The probability that the classifier gives a different prediction if the identity attribute in a given feature were changed.

which can be measured by some counterfactual metrics, such as flip rate. As shown in the graph, Counterfactual Logit Pairing can help models achieve lower flip rate.

Summary: MinDiff and CLP

	MinDiff	CLP
When should you use this technique?	To ensure that a model predicts the preferred label equally well for all values of a sensitive attribute. To achieve group **equality of opportunity.**	To ensure that a model's prediction does not change between "counterfactual pairs" (where the sensitive attribute referenced in a feature is different). To achieve a form of **counterfactual fairness.**
How does it work?	Penalizes the model during training for differences in the *distribution* of scores between the two sets.	Penalizes the model during training for output differences between counterfactual *pairs of examples*.

So, although MinDiff and CLP have similar ideas to add a loss term that represents the magnitude of some sort of bias, the target and idea are slightly different. MinDiff focuses on prediction distribution so that it leads to achieving fairness constraints such as equality of opportunity, whereas CLP focuses on counterfactual fairness.

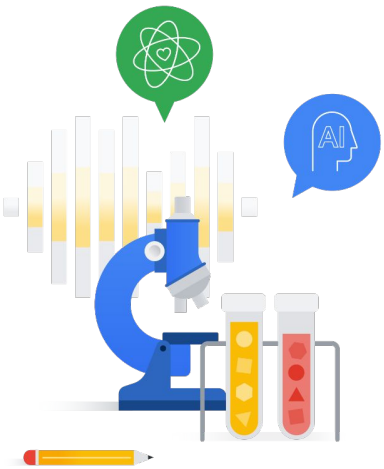


Hands-on lab:

Mitigate Bias with MinDiff in TensorFlow

Let’s perform an exercise in a hands-on lab.

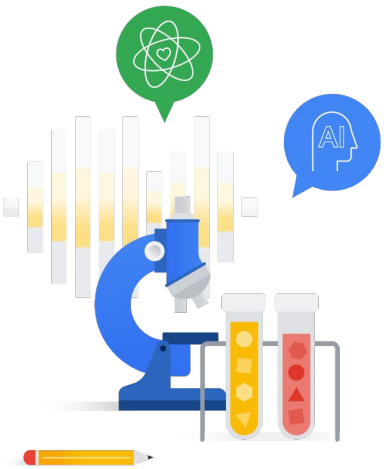
In this Lab, you will:



This lab helps you learn how to mitigate bias using MinDiff technique by leveraging TensorFlow Model Remediation library. In this lab, you will:

In this Lab, you will:

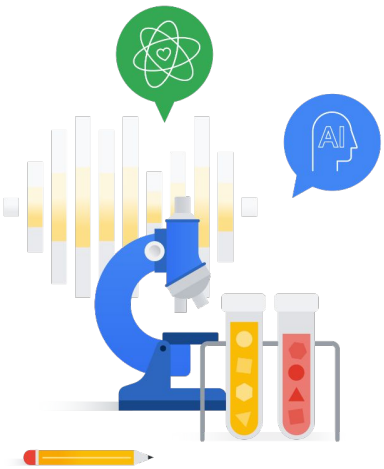
- Explore the toxicity text dataset.



- Explore the toxicity text dataset.

In this Lab, you will:

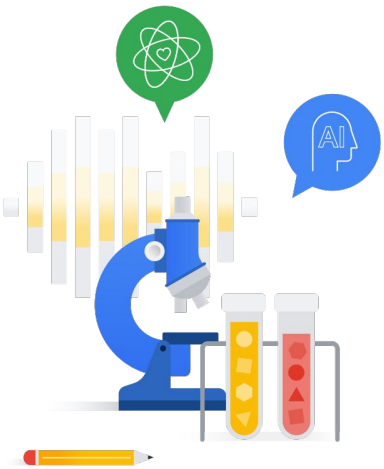
- Explore the toxicity text dataset.
- Build and train a toxicity classification model.



- Build and train a toxicity classification model.

In this Lab, you will:

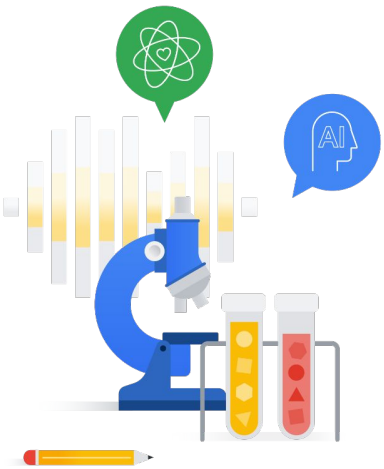
- Explore the toxicity text dataset.
- Build and train a toxicity classification model.
- Check the model bias by plotting the prediction results.



- Check the model bias by plotting the prediction results.

In this Lab, you will:

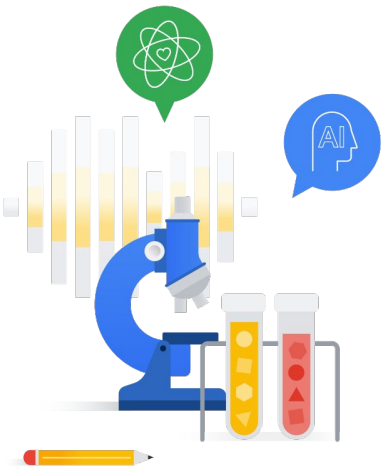
- Explore the toxicity text dataset.
- Build and train a toxicity classification model.
- Check the model bias by plotting the prediction results.
- Apply MinDiff technique using TensorFlow Model Remediation library.



- Apply MinDiff technique using TensorFlow Model Remediation library.

In this Lab, you will:

- Explore the toxicity text dataset.
- Build and train a toxicity classification model.
- Check the model bias by plotting the prediction results.
- Apply MinDiff technique using TensorFlow Model Remediation library.
- Compare the result between the baseline and MinDiff models.



- Compare the result between the baseline and MinDiff models.