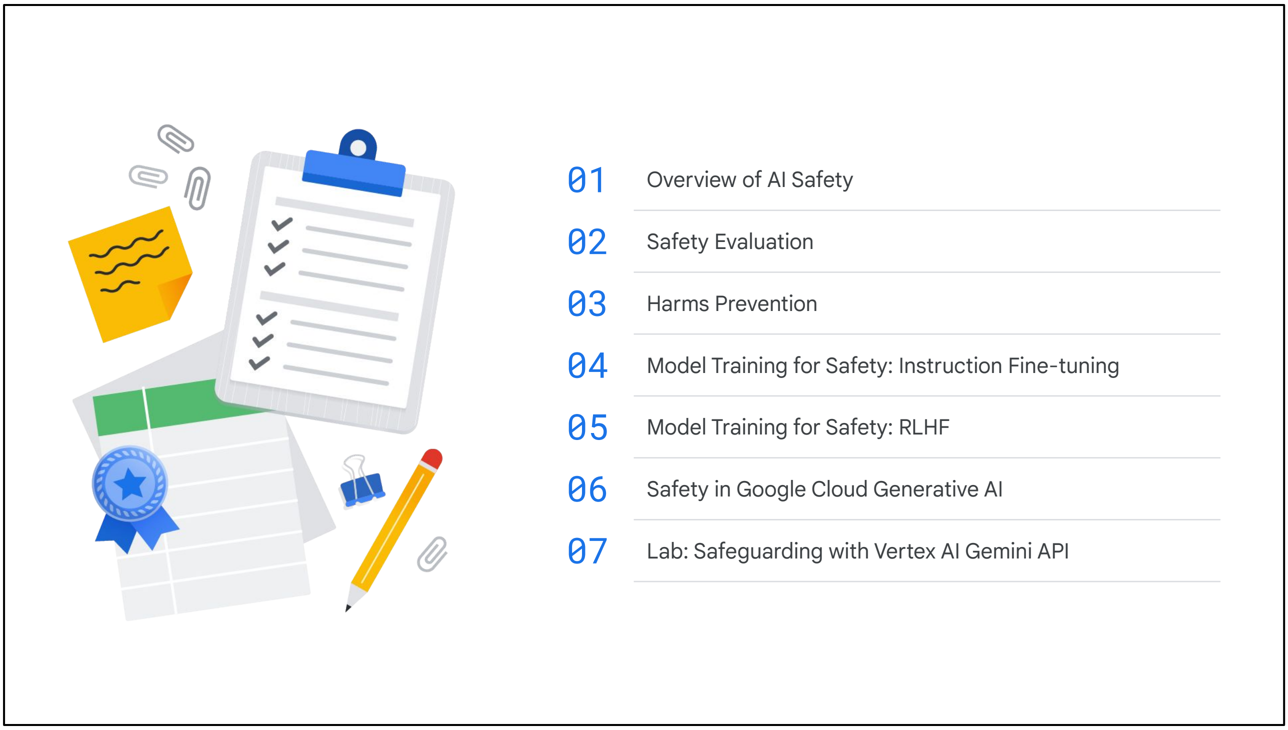
02

AI Safety

Welcome to AI Safety.



01 Overview of AI Safety

02 Safety Evaluation

03 Harms Prevention

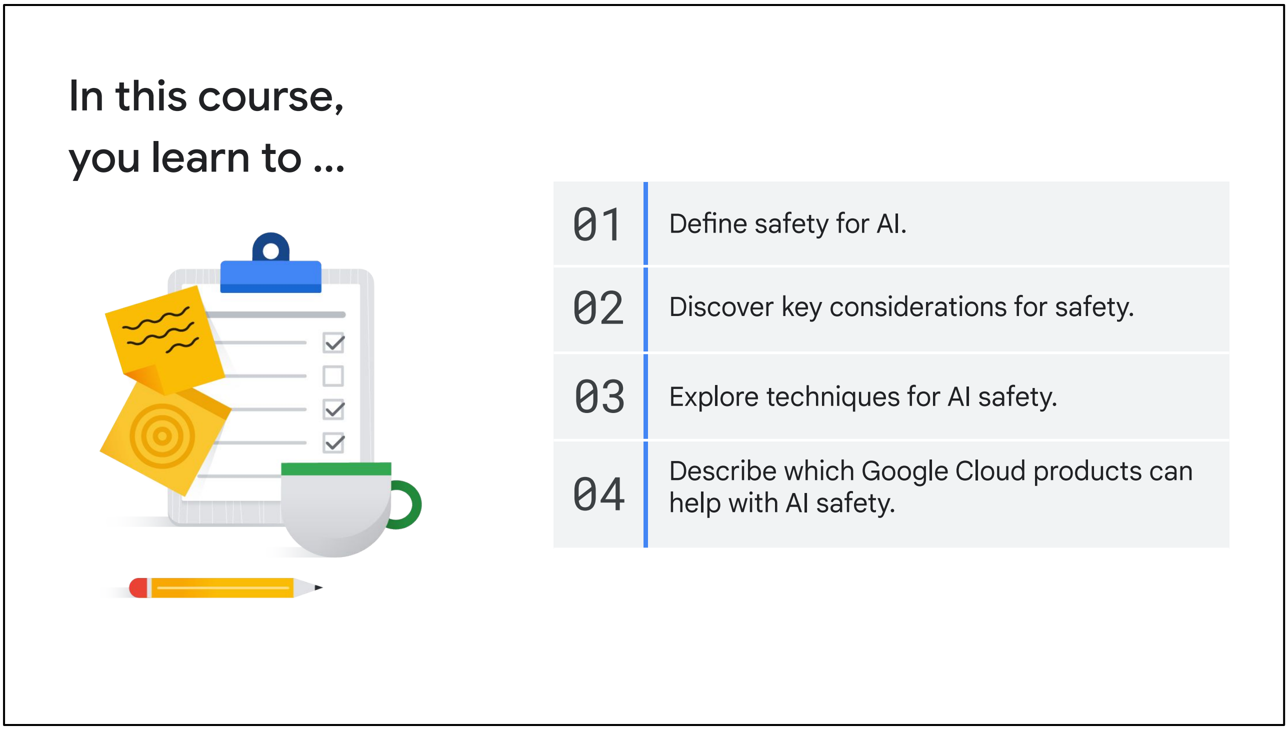
04 Model Training for Safety: Instruction Fine-tuning

05 Model Training for Safety: RLHF

06 Safety in Google Cloud Generative AI

07 Lab: Safeguarding with Vertex AI Gemini API

This module consists of 7 lessons.



In this course, you learn to ...

01 Define safety for AI.

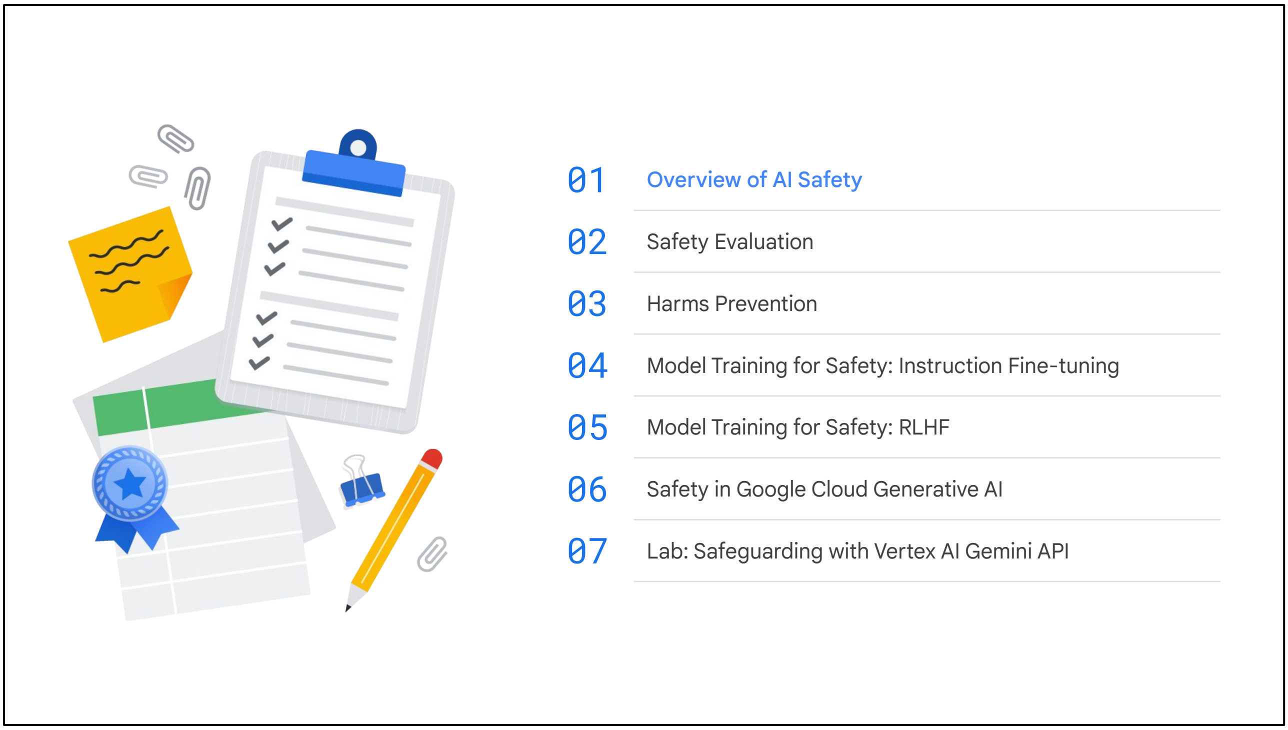
02 Discover key considerations for safety.

03 Explore techniques for AI safety.

04 Describe which Google Cloud products can help with AI safety.

Today you will learn to…

- Define safety for AI.
- Discover key considerations for safety.
- Explore techniques for AI safety.
- And describe which Google Cloud products can help with AI safety.



01 Overview of AI Safety

02 Safety Evaluation

03 Harms Prevention

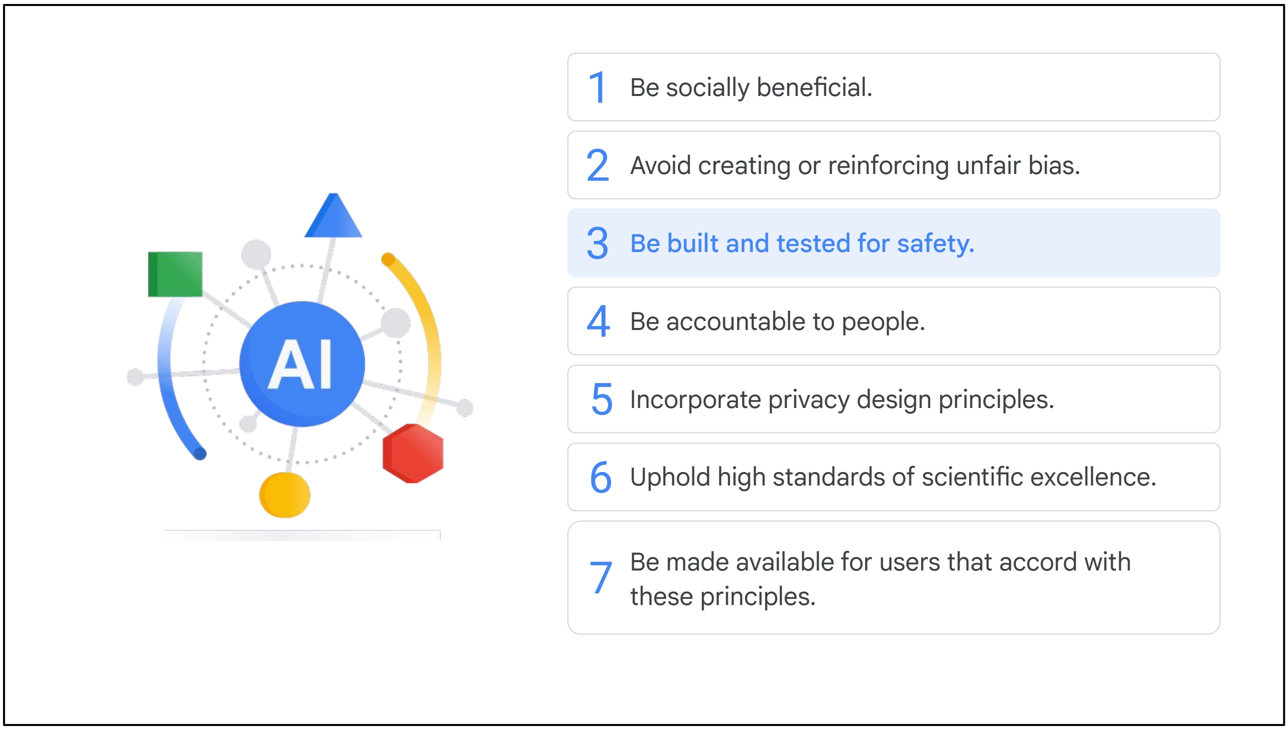
04 Model Training for Safety: Instruction Fine-tuning

05 Model Training for Safety: RLHF

06 Safety in Google Cloud Generative AI

07 Lab: Safeguarding with Vertex AI Gemini API

Let’s start with the overview of what AI safety means.



1 Be socially beneficial.

2 Avoid creating or reinforcing unfair bias.

3 Be built and tested for safety.

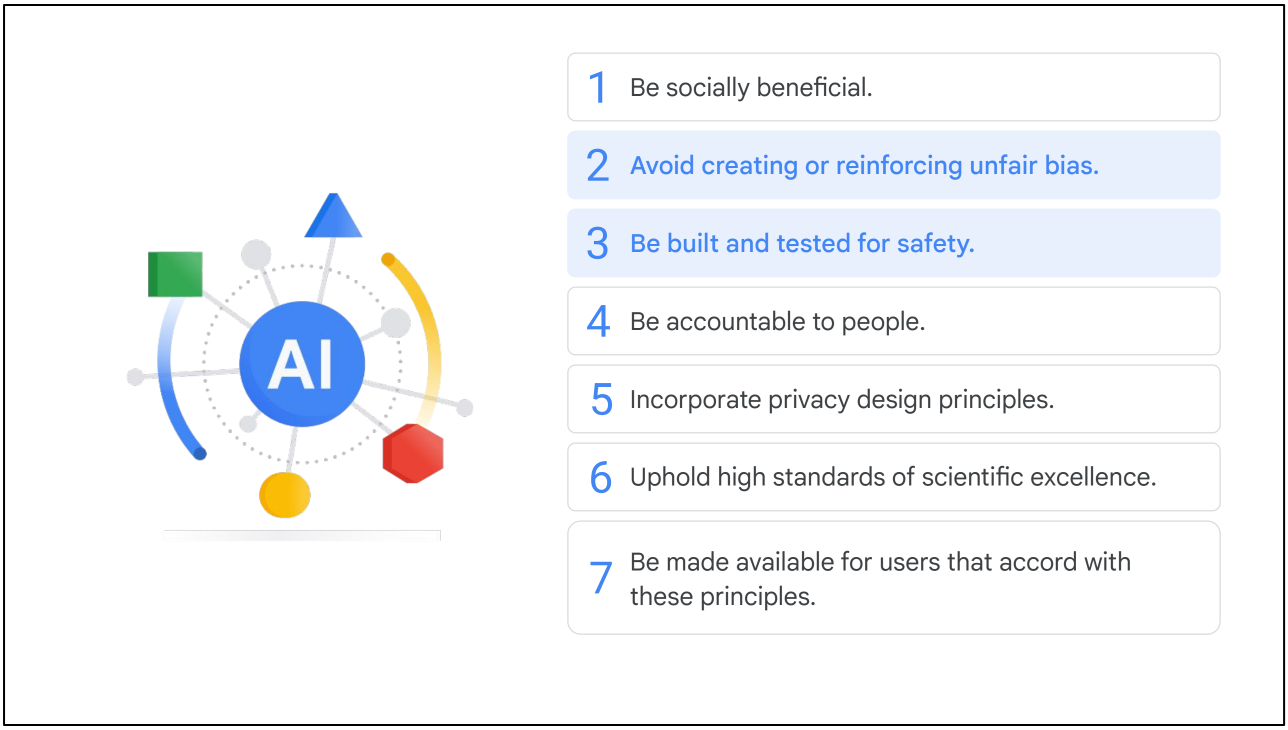
4 Be accountable to people.

5 Incorporate privacy design principles.

6 Uphold high standards of scientific excellence.

7 Be made available for users that accord with these principles.

The concept of safety directly relates to the third of Google's AI principles: "Be built and tested for safety." Safety is also strongly related to other Google AI principles, especially,



1 Be socially beneficial.

2 Avoid creating or reinforcing unfair bias.

3 Be built and tested for safety.

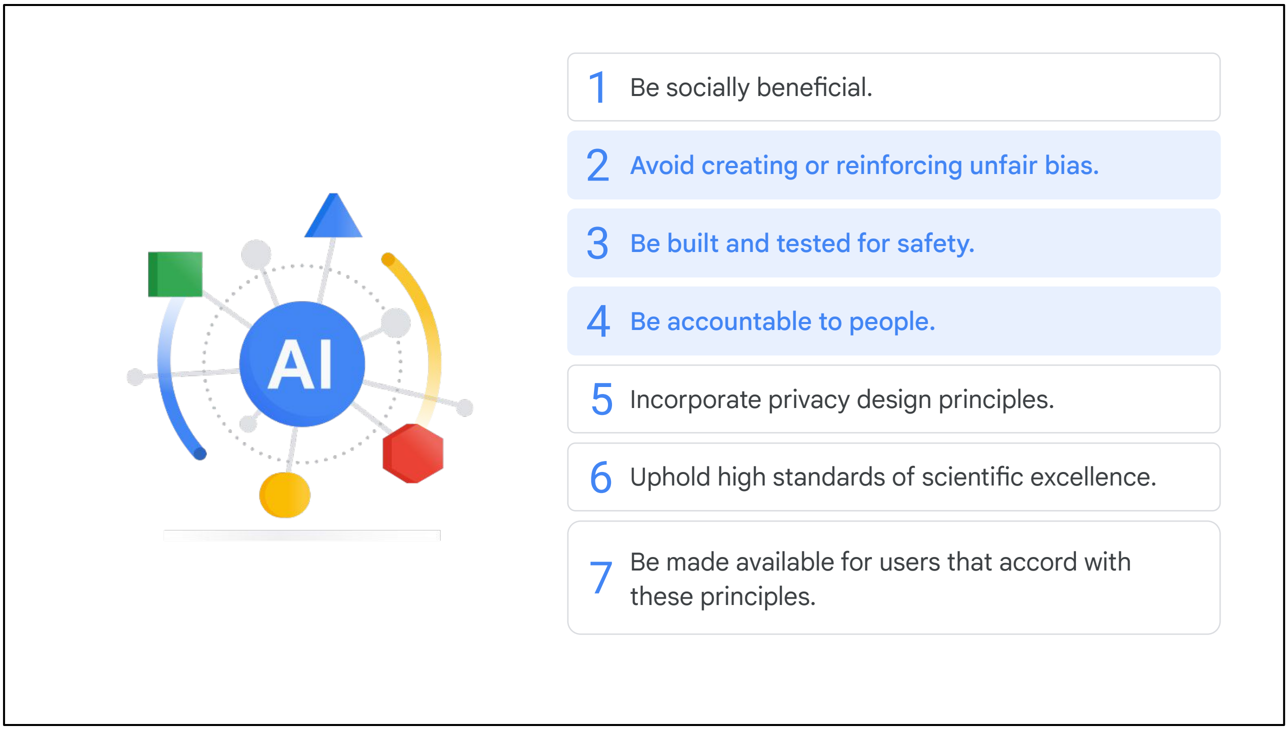
4 Be accountable to people.

5 Incorporate privacy design principles.

6 Uphold high standards of scientific excellence.

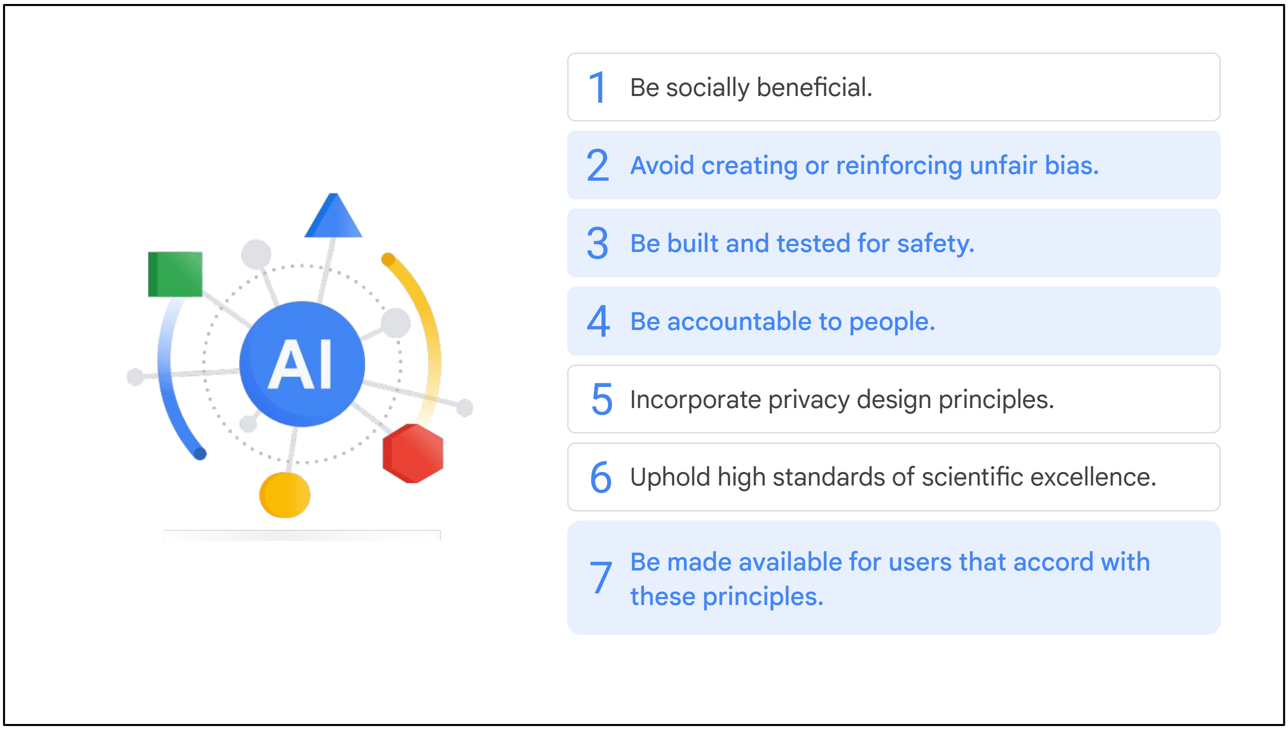
7 Be made available for users that accord with these principles.

“Avoid creating or reinforcing unfair bias”, since the concept of safety often comes from the definition of fairness and bias, and proper safety implementation makes systems more fair. There are strong crossovers between fairness and safety.



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.

“Be accountable to people,” as the accountability of the AI system helps promote safe use of AI.



1 Be socially beneficial.

2 Avoid creating or reinforcing unfair bias.

3 Be built and tested for safety.

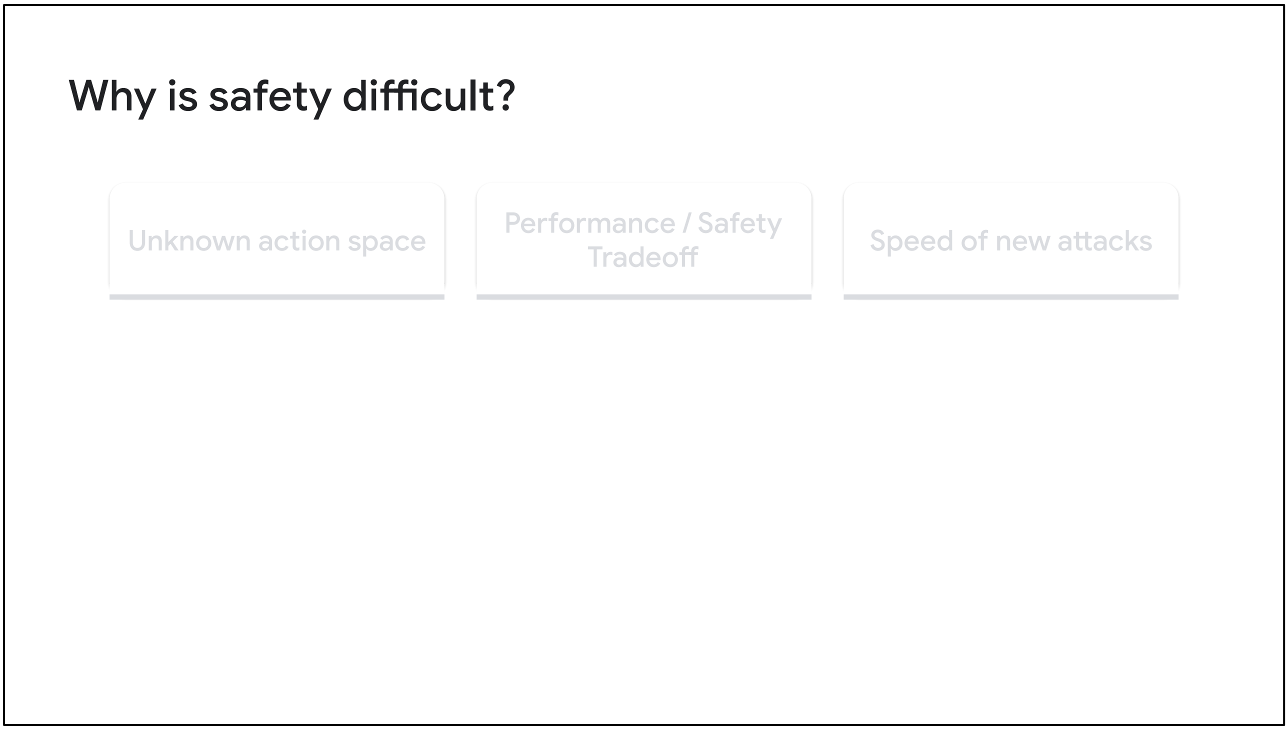
4 Be accountable to people.

5 Incorporate privacy design principles.

6 Uphold high standards of scientific excellence.

7 Be made available for users that accord with these principles.

And “Be made available for uses that accord with these principles.” This is especially important when you make your AI system available to a wide group of users. As AI capabilities grow rapidly, keep in mind that AI systems are not necessarily used in a way you imagined.



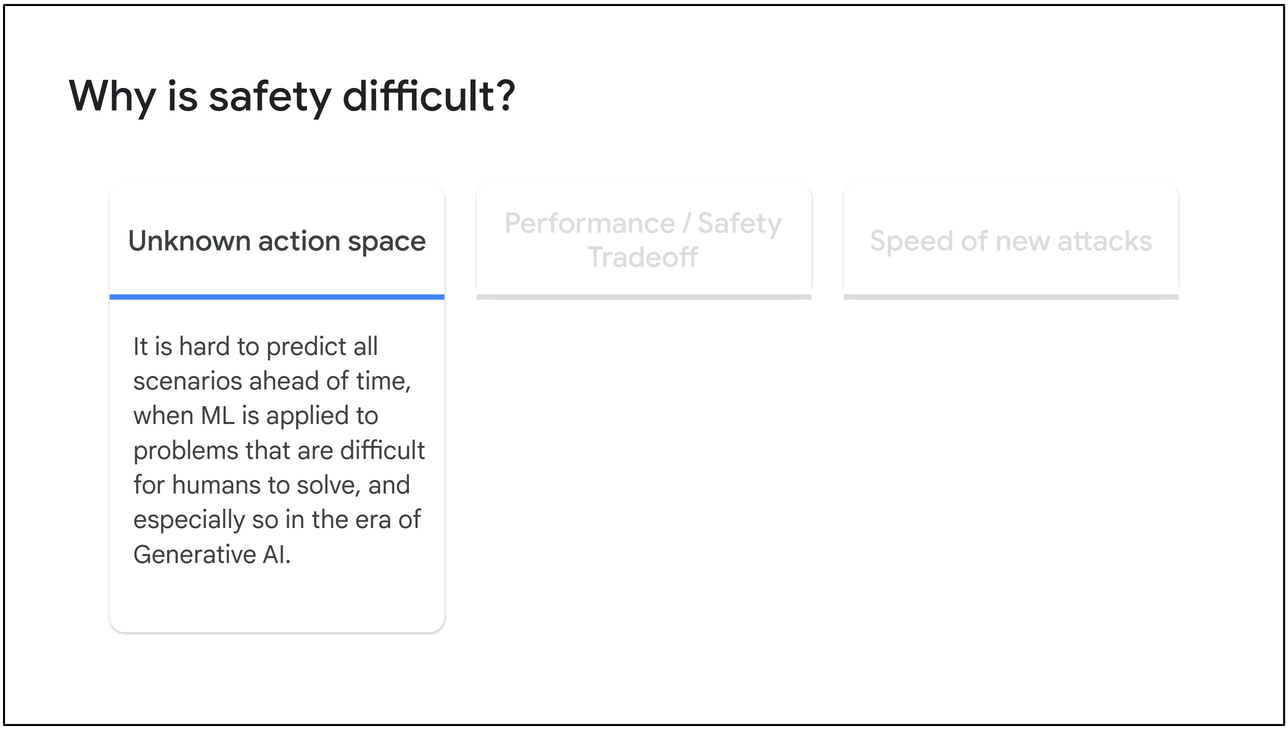
Why is safety difficult?

Unknown action space

Performance / Safety Tradeoff

Speed of new attacks

AI safety involves many perspectives and considerations, and it has many difficulties, such as:



Why is safety difficult?

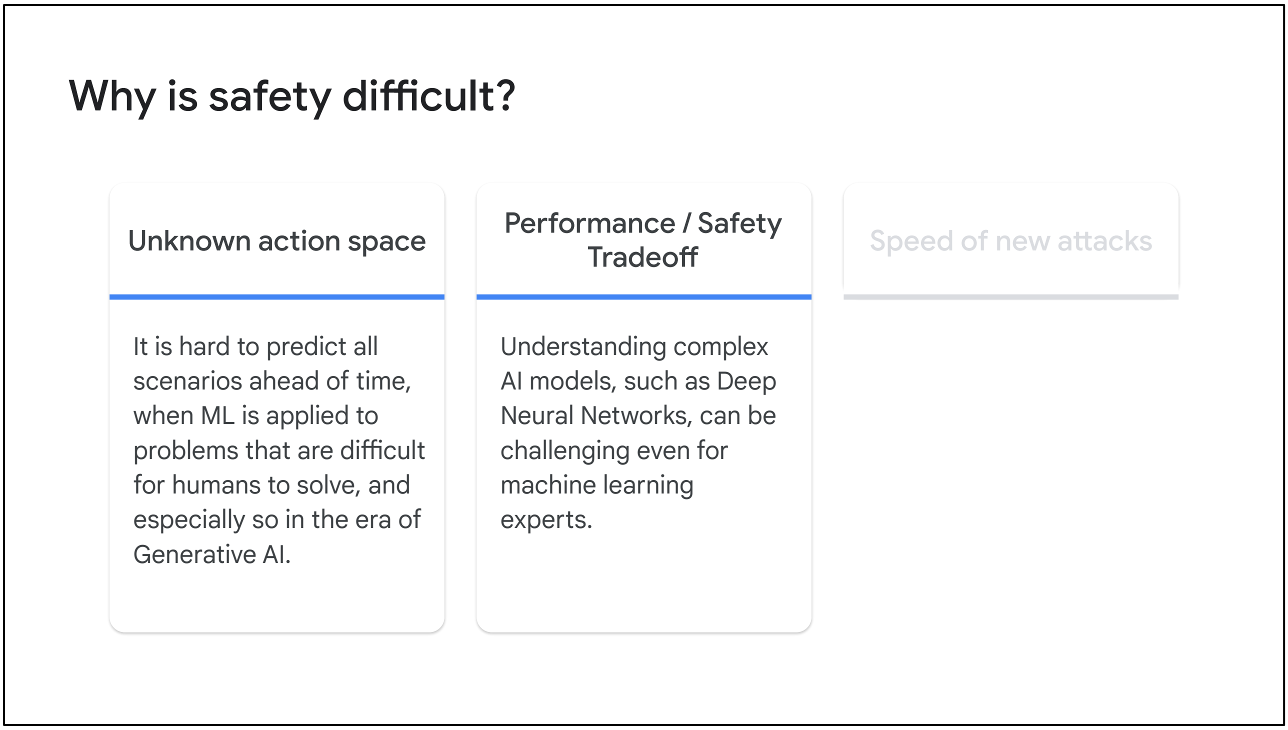
Unknown action space

Performance / Safety Tradeoff

Speed of new attacks

It is hard to predict all scenarios ahead of time, when ML is applied to problems that are difficult for humans to solve, and especially so in the era of Generative AI.

- Unknown action space. When machine learning is applied to problems that are difficult for humans to solve, it becomes challenging to predict all scenarios ahead of time, and especially so in the era of generative AI.



Why is safety difficult?

Unknown action space

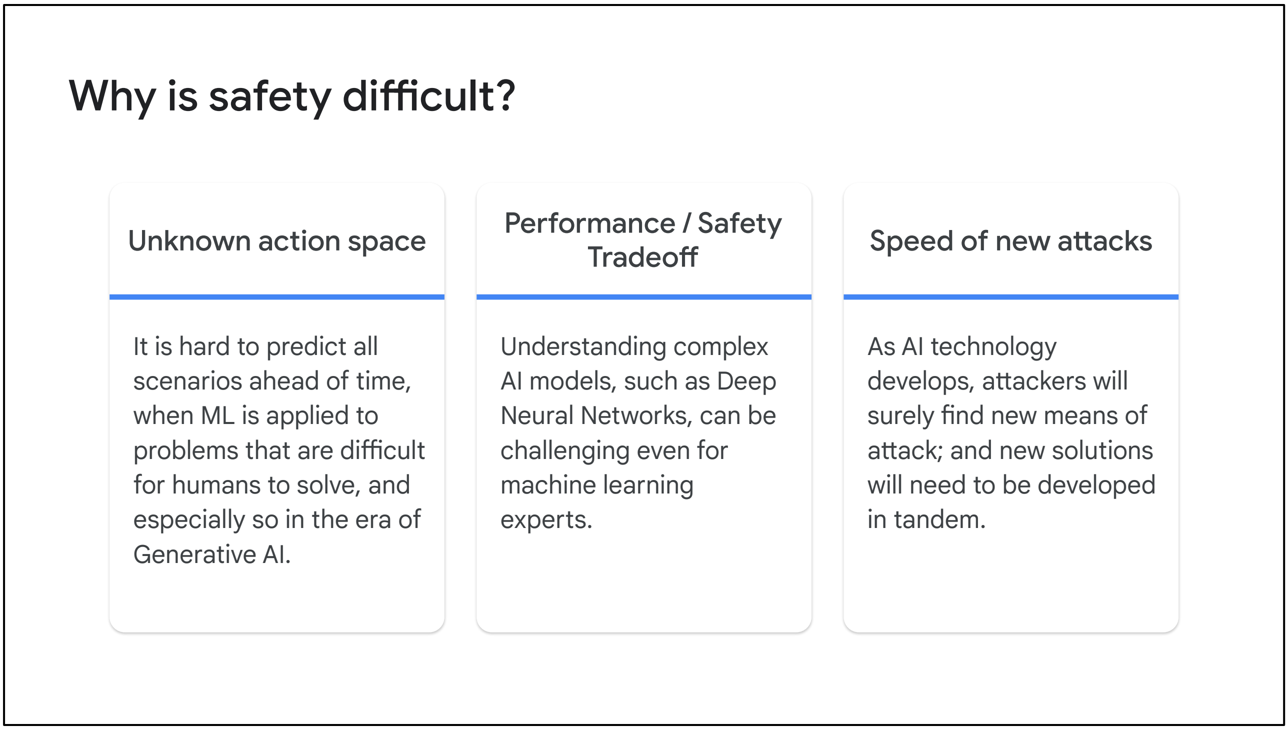
It is hard to predict all scenarios ahead of time, when ML is applied to problems that are difficult for humans to solve, and especially so in the era of Generative AI.

Performance / Safety Tradeoff

Understanding complex AI models, such as Deep Neural Networks, can be challenging even for machine learning experts.

Speed of new attacks

- Performance and safety trade-offs. It is difficult to build systems that provide both the necessary proactive restrictions for safety, as well as the flexibility needed to generate creative solutions or adapt to unusual inputs.



Why is safety difficult?

Unknown action space

It is hard to predict all scenarios ahead of time, when ML is applied to problems that are difficult for humans to solve, and especially so in the era of Generative AI.

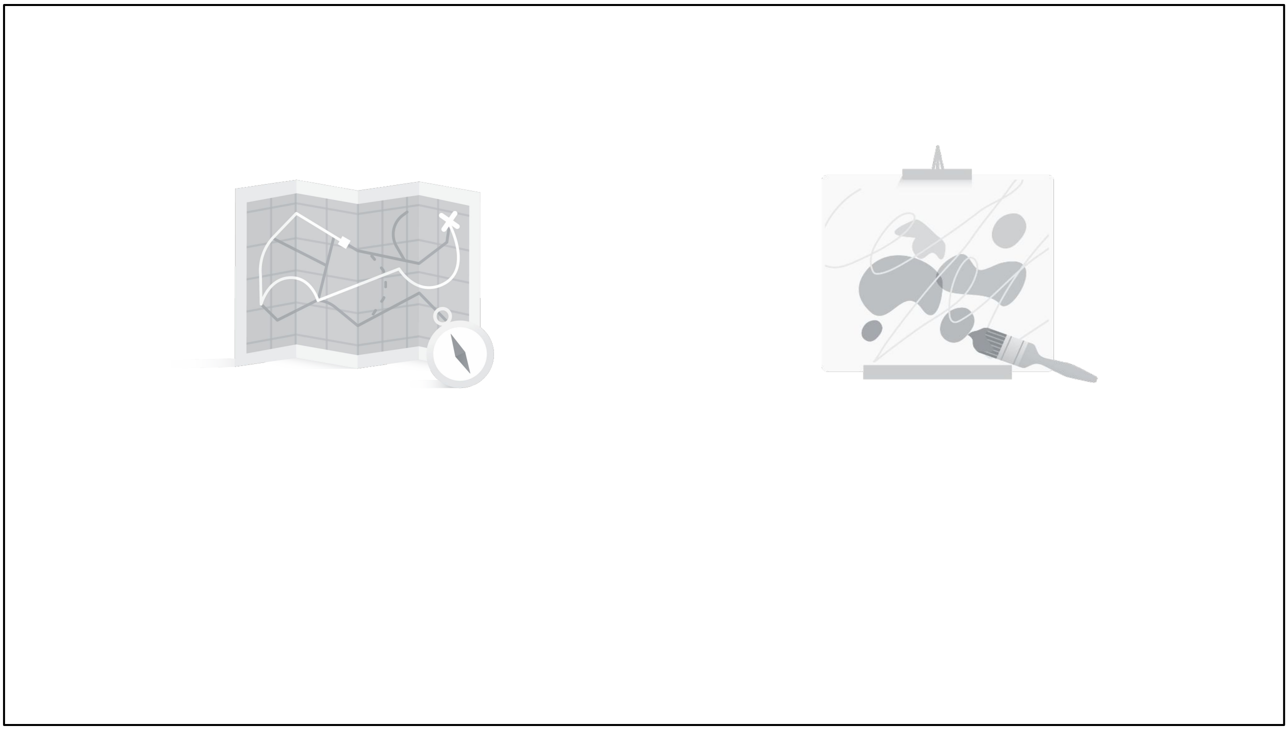
Performance / Safety Tradeoff

Understanding complex AI models, such as Deep Neural Networks, can be challenging even for machine learning experts.

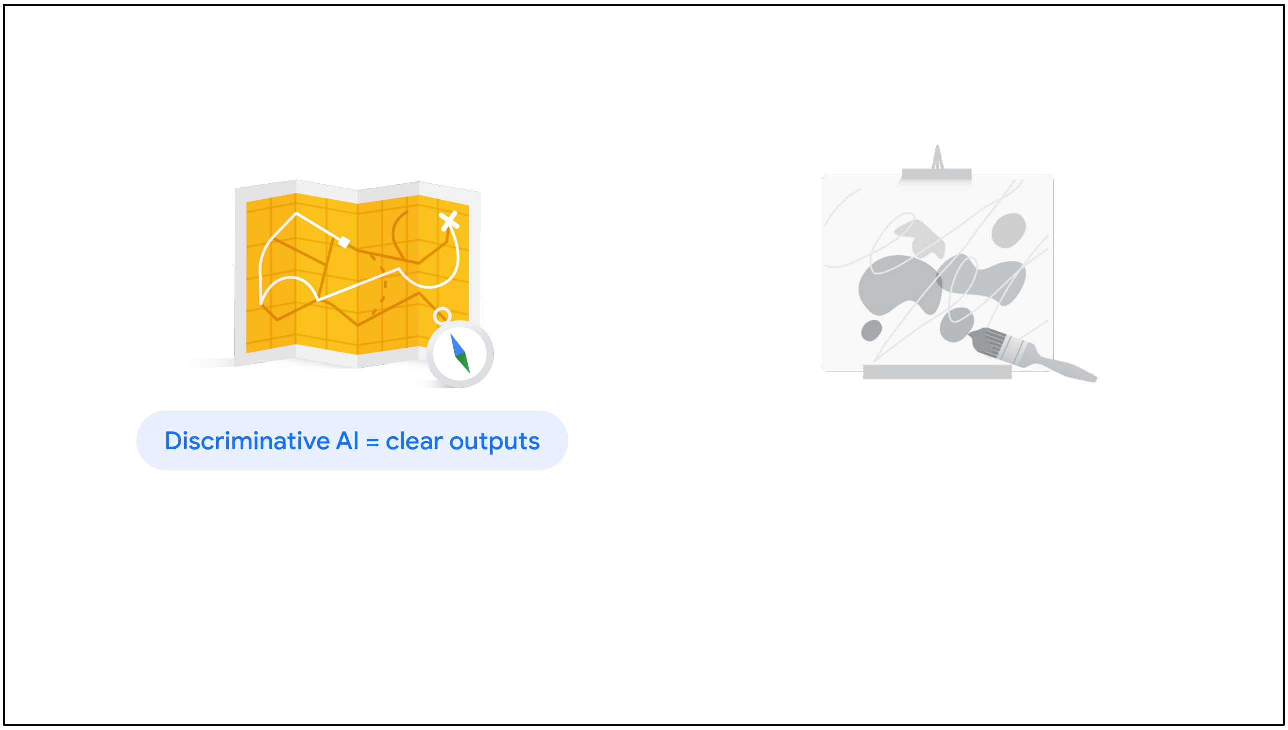
Speed of new attacks

As AI technology develops, attackers will surely find new means of attack; and new solutions will need to be developed in tandem.

- And attackers adapt fast to new technology. As AI technology continues to evolve, attackers will evolve too, and will surely find new means to attack, resulting in new solutions that will need to be developed in tandem.

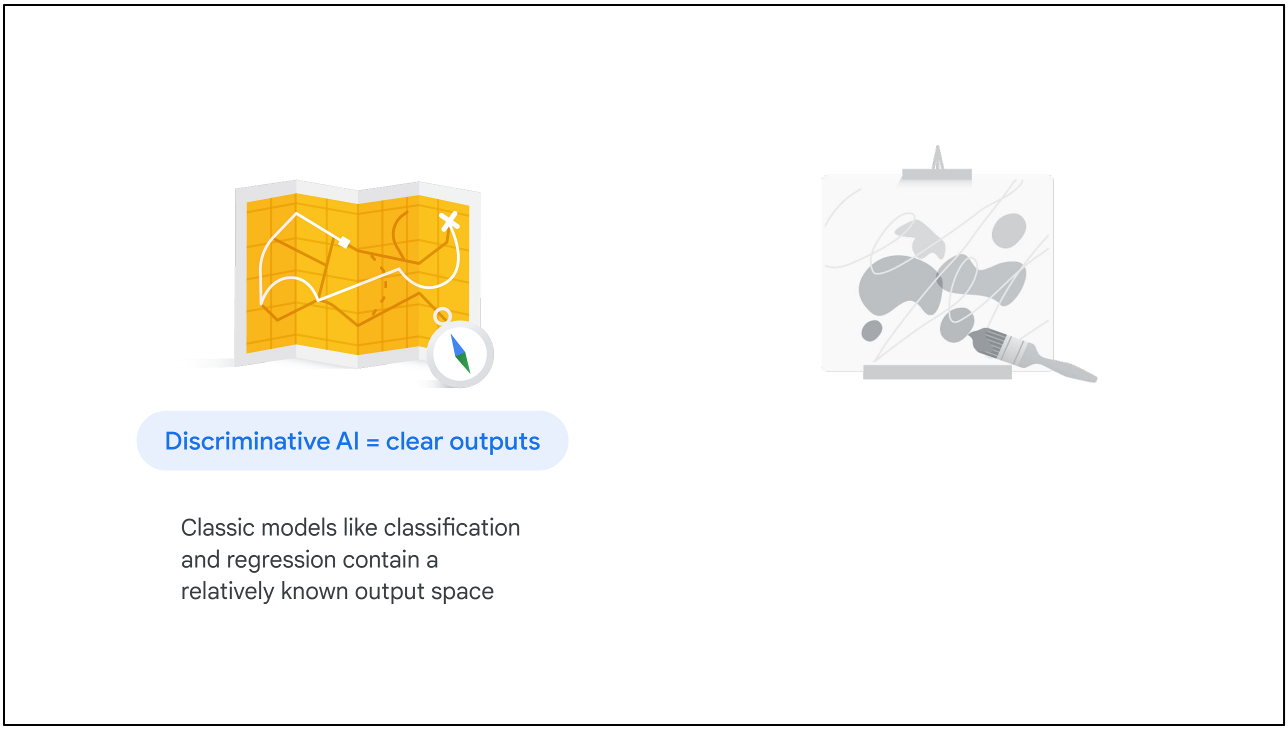


AI safety is especially difficult when it comes to generative AI.



Discriminative AI = clear outputs

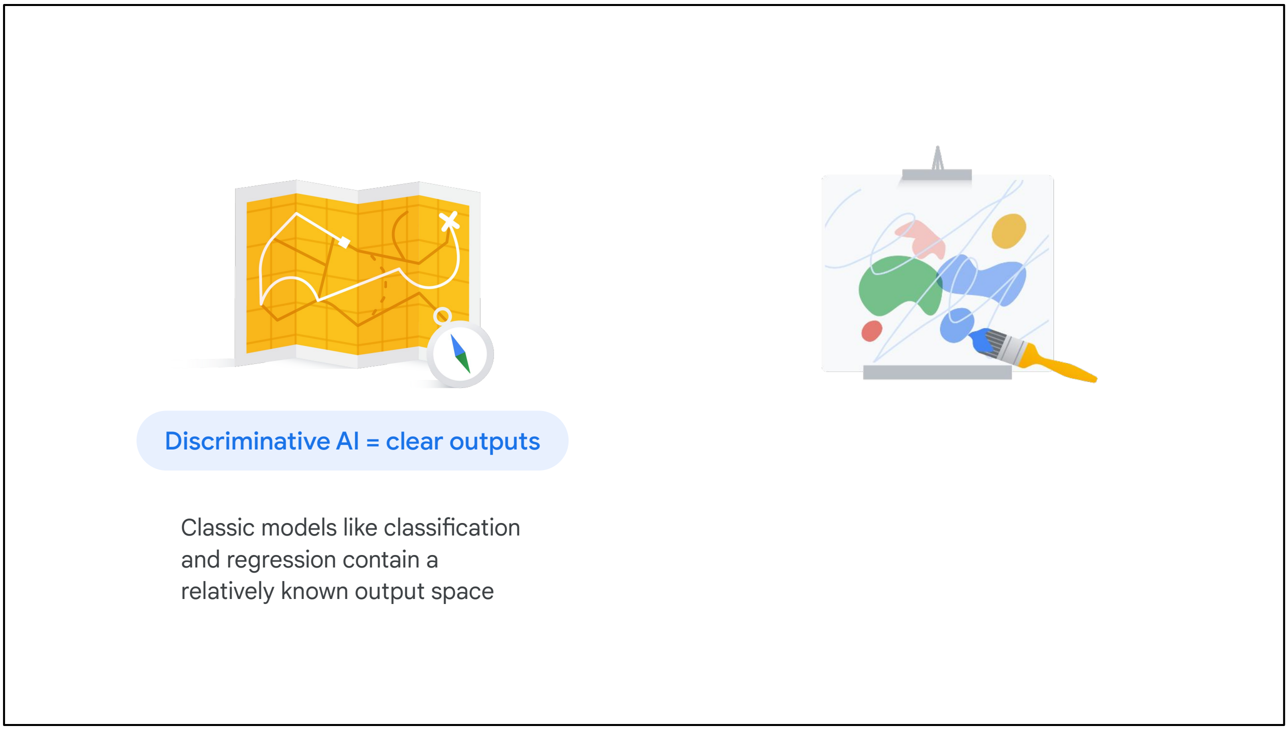
With classic discriminative AI models, such as those trained for classification and regression tasks, the output space is fairly well known.



Discriminative AI = clear outputs

Classic models like classification and regression contain a relatively known output space

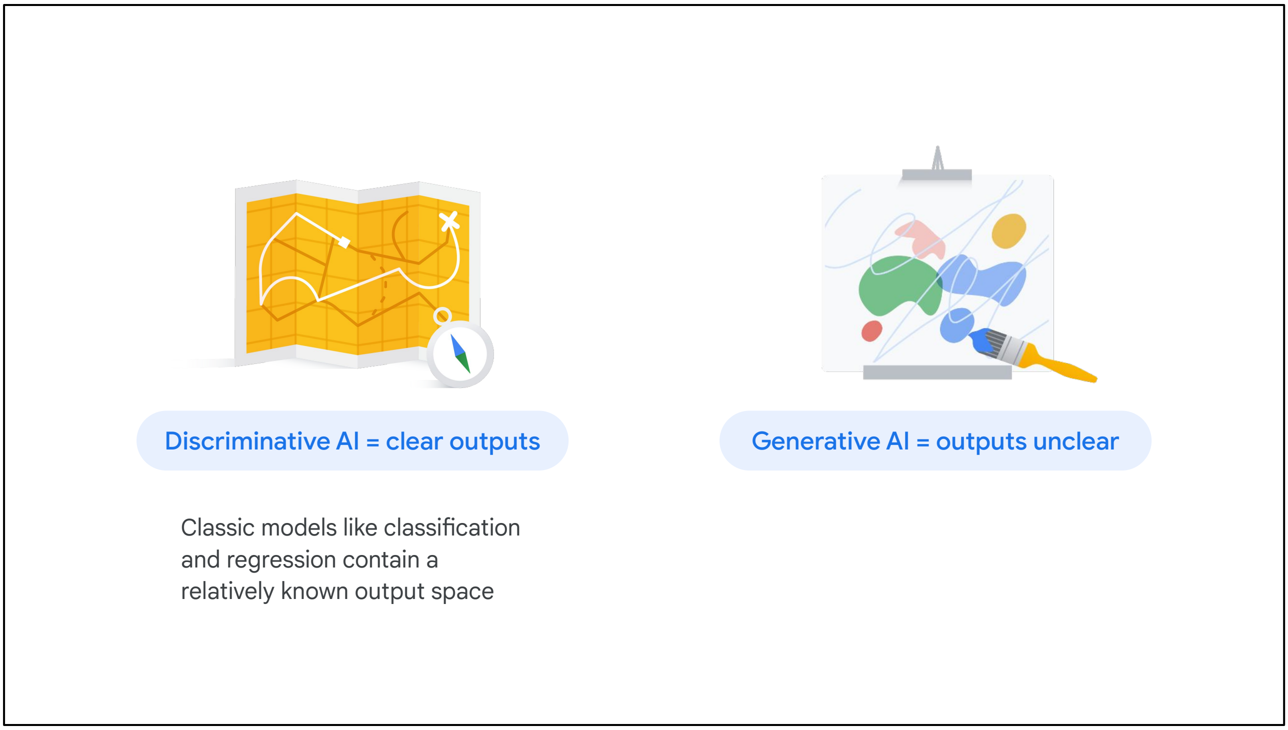
Classification models can only predict the classes they were trained on, so we can simply exclude classes we deem to be harmful. On the other hand, regression models can only predict a number, and we know what that number is supposed to represent. That means, when we build the models, we have a good idea of what the model can at least predict. This doesn't mean that the models are entirely safe and can't cause harm, but it does make it easier to find problems.



Discriminative AI = clear outputs

Classic models like classification and regression contain a relatively known output space

However, with generative AI, we're creating a system that allows users to explore the breadth and depth of their creativity.

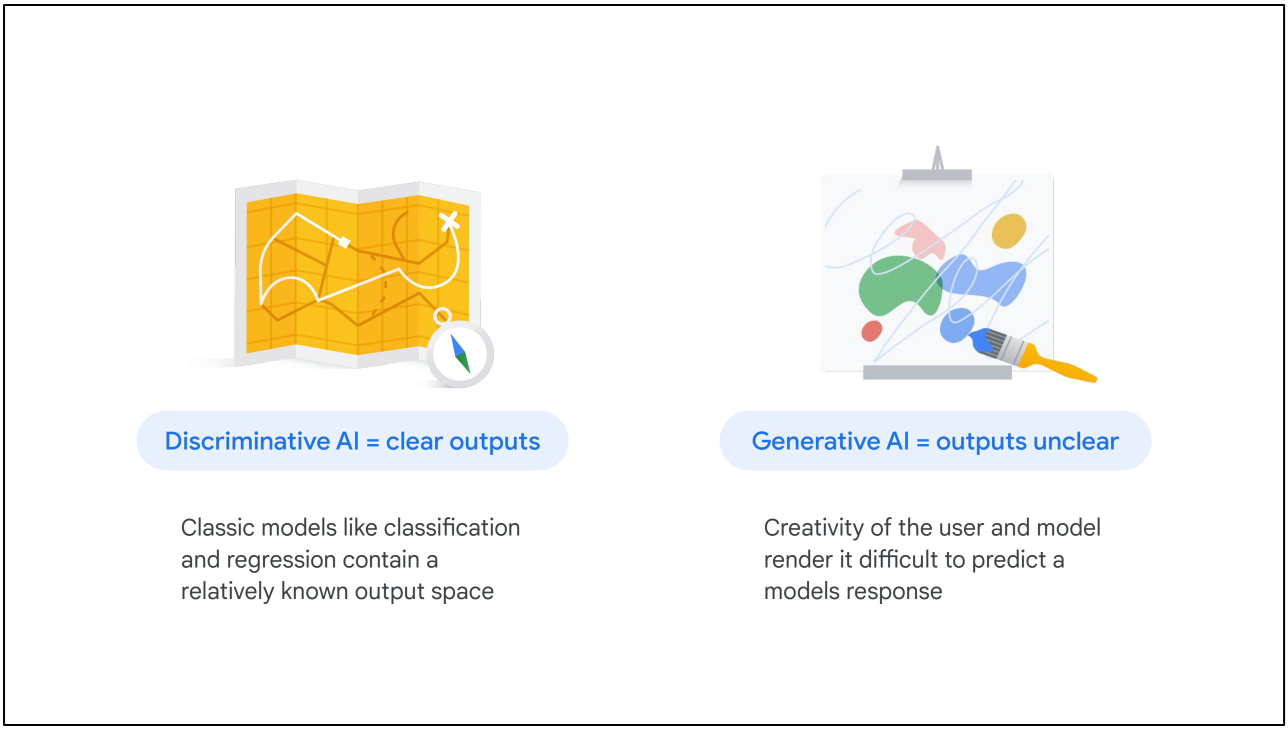


Discriminative AI = clear outputs

Classic models like classification and regression contain a relatively known output space

Generative AI = outputs unclear

This means that the models are trained on large amounts of data, and they are designed to learn to generate outputs that are very different from the data they were trained on.



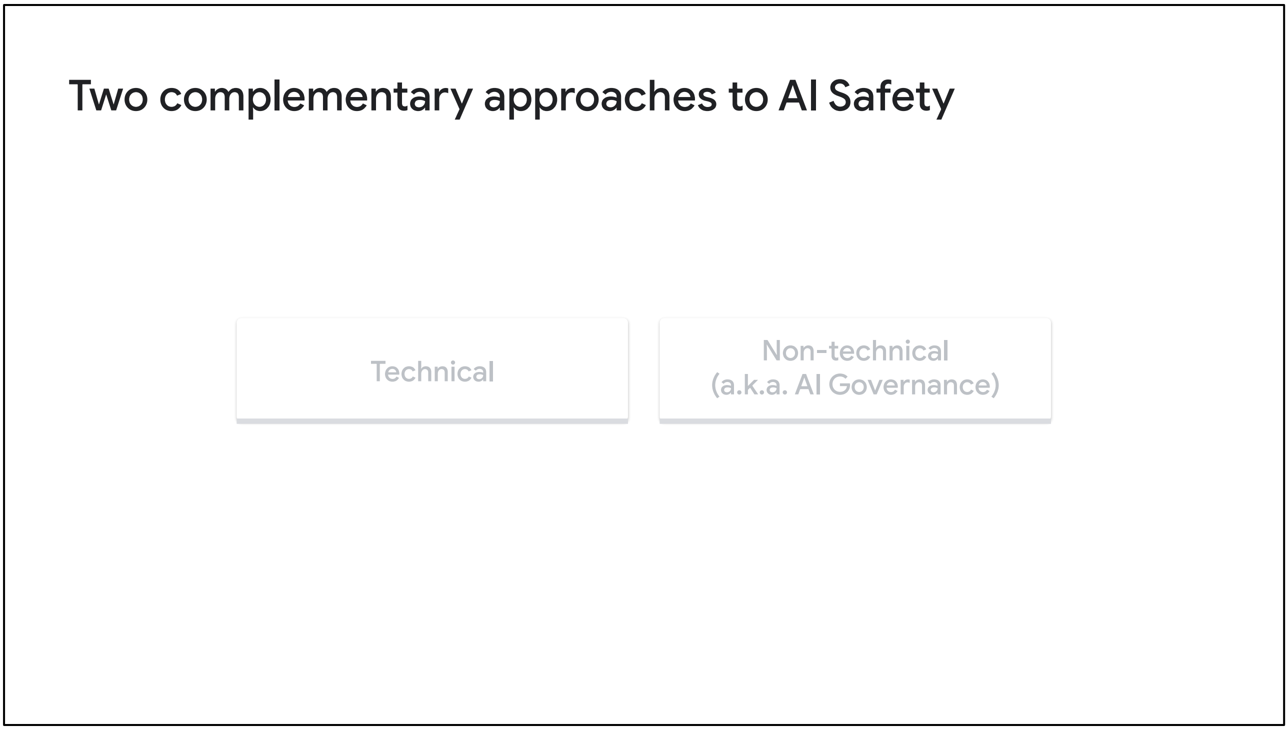
Discriminative AI = clear outputs

Classic models like classification and regression contain a relatively known output space

Generative AI = outputs unclear

Creativity of the user and model render it difficult to predict a models response

The creativity of the user combined with the emergent creativity of the model, means that it is difficult to know the breadth and depth of a model's responses in advance.

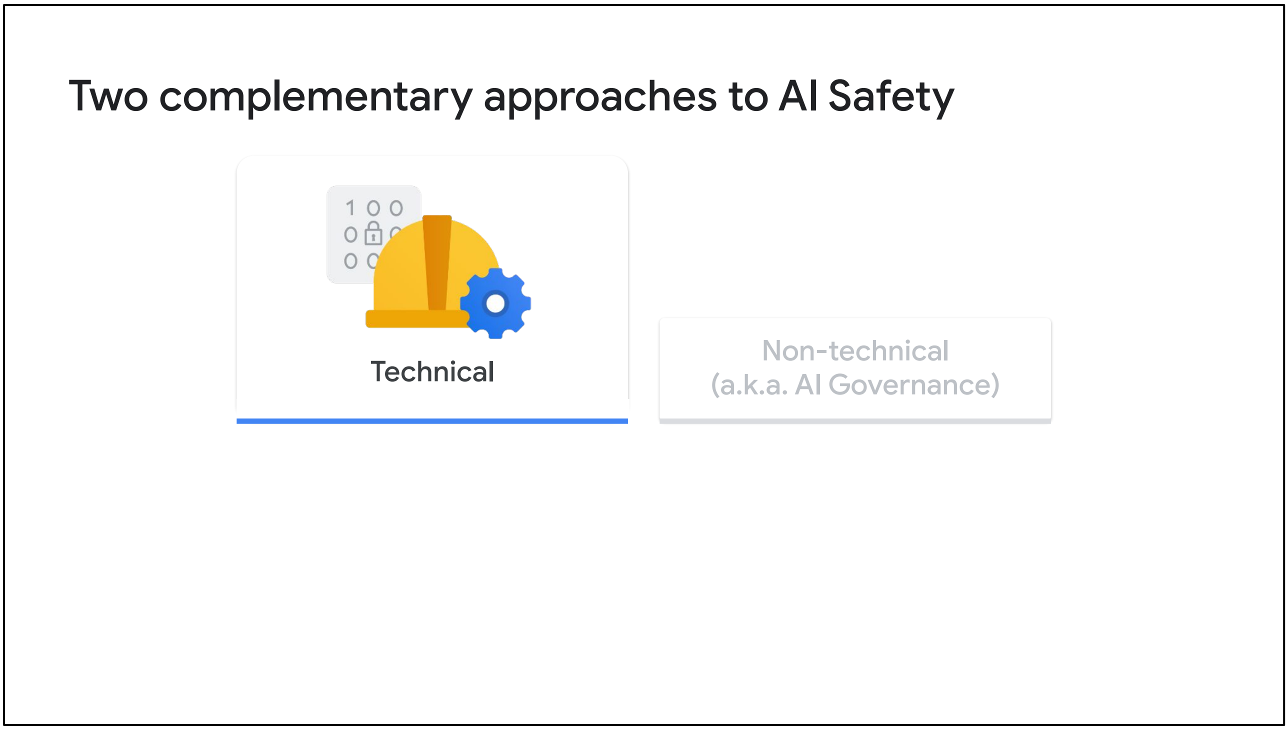


Two complementary approaches to AI Safety

Technical

Non-technical
(a.k.a. AI Governance)

To approach safety, there are roughly two different approaches which are complementary to each other.

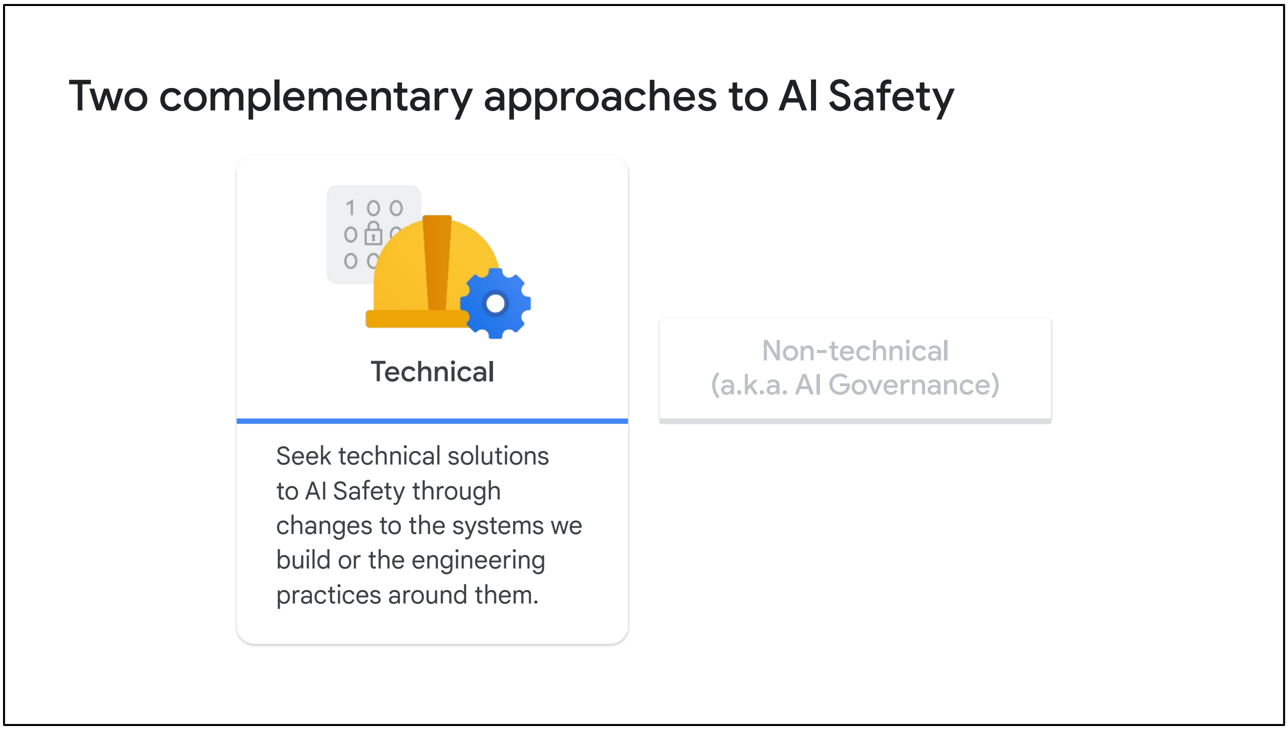


Two complementary approaches to AI Safety

Technical

Non-technical
(a.k.a. AI Governance)

First is a technical approach



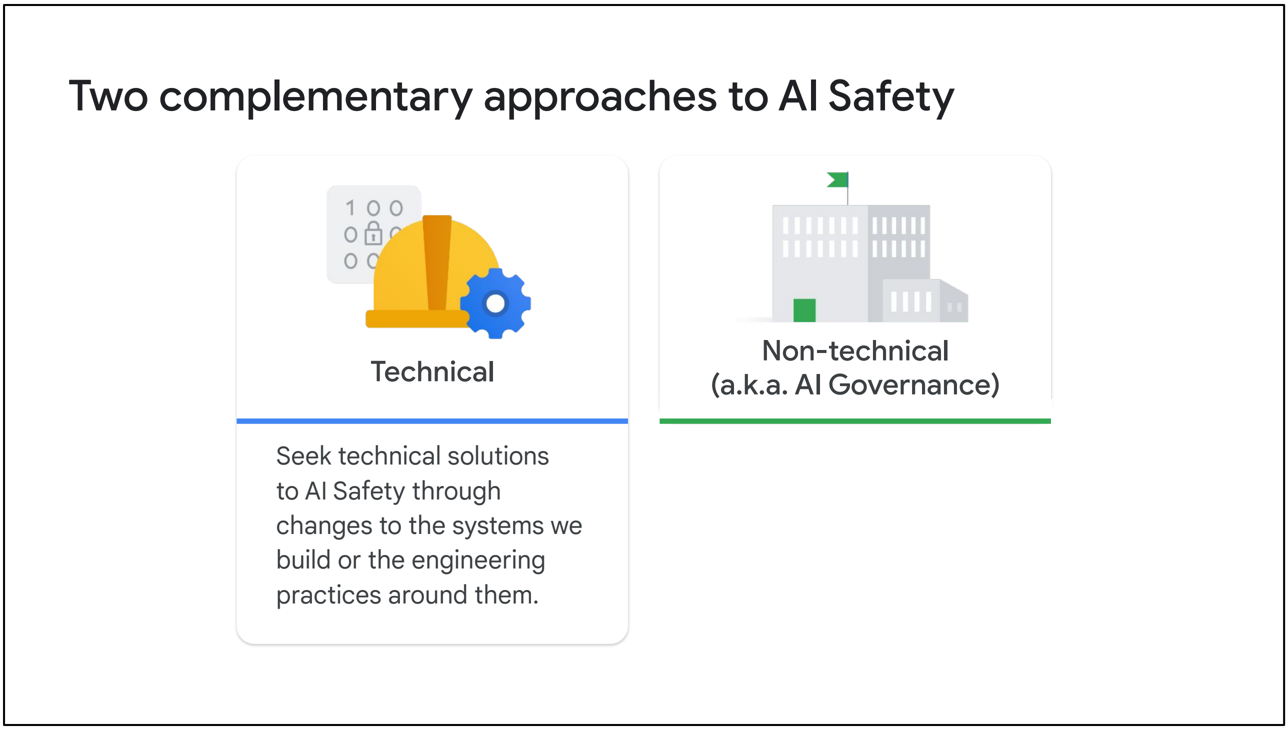
Two complementary approaches to AI Safety

Technical

Seek technical solutions to AI Safety through changes to the systems we build or the engineering practices around them.

Non-technical
(a.k.a. AI Governance)

that seeks technical solutions through changes to the model or the system we build, or through engineering practices.



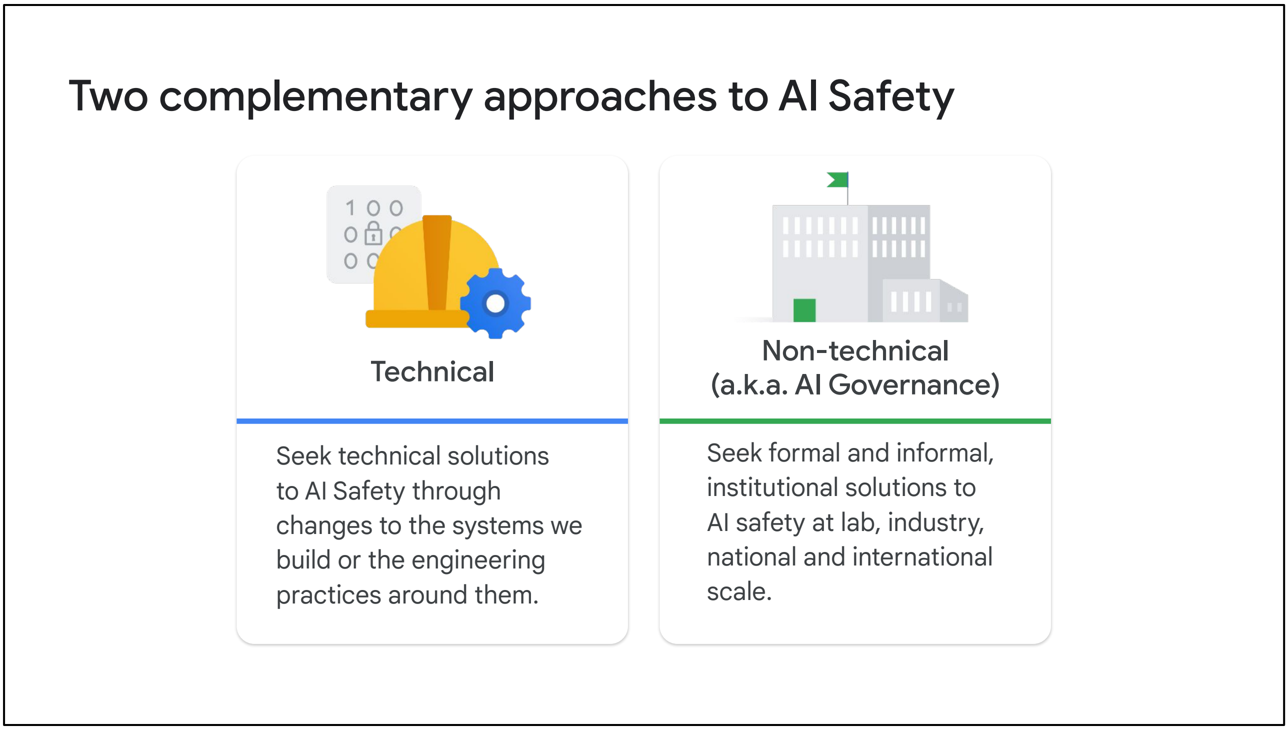
Two complementary approaches to AI Safety

Technical

Seek technical solutions to AI Safety through changes to the systems we build or the engineering practices around them.

Non-technical
(a.k.a. AI Governance)

On the other hand, a non-technical, or institutional, approach is also very important,



Two complementary approaches to AI Safety

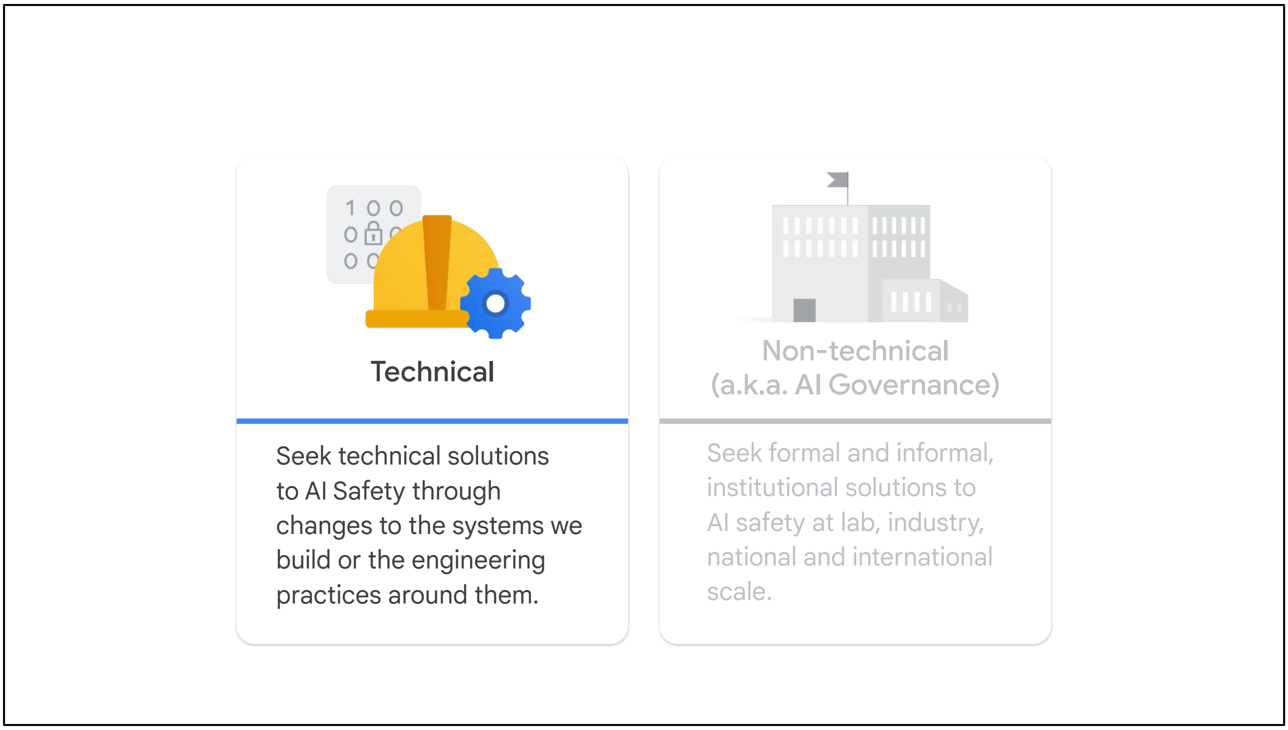
Technical

Seek technical solutions to AI Safety through changes to the systems we build or the engineering practices around them.

Non-technical (a.k.a. AI Governance)

Seek formal and informal, institutional solutions to AI safety at lab, industry, national and international scale.

as it seeks formal and informal, institutional solutions to AI safety at the lab. Usually this involves industry-wide, national and international scale discussions and commitments. This approach is also known as a topic of AI governance.



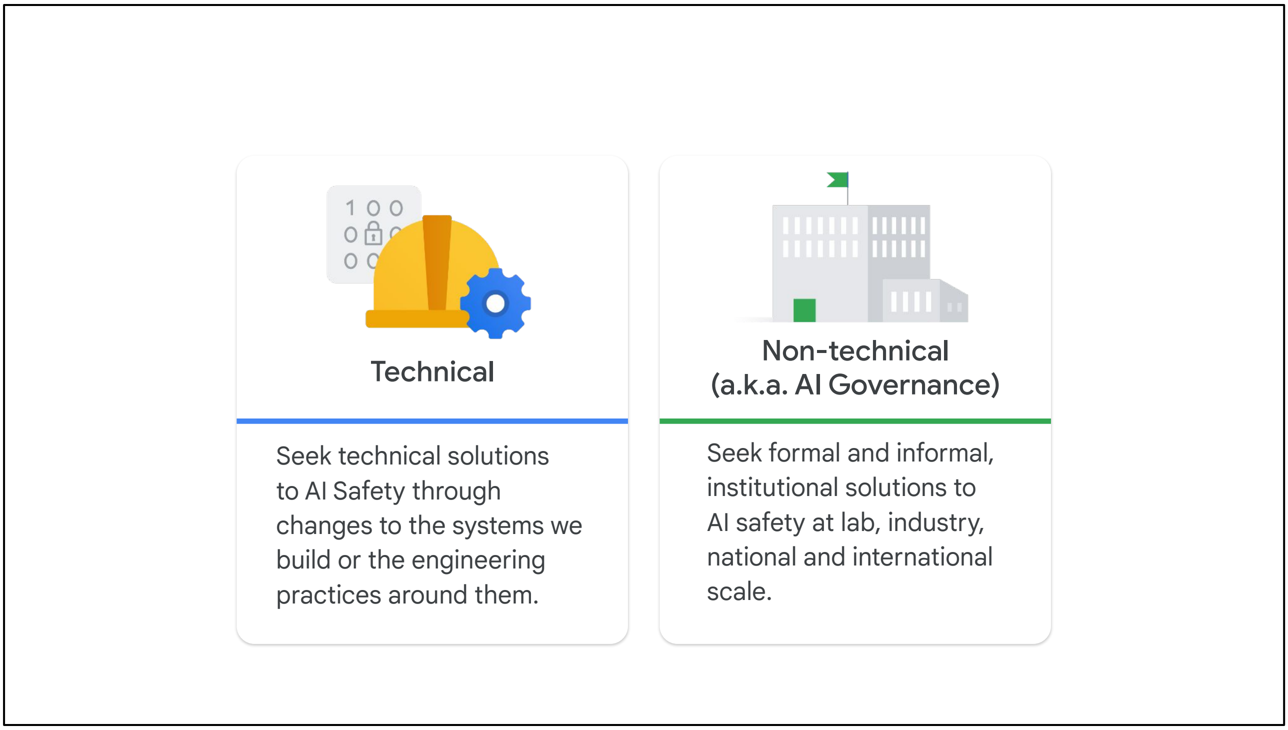
Technical

Seek technical solutions to AI Safety through changes to the systems we build or the engineering practices around them.

Non-technical (a.k.a. AI Governance)

Seek formal and informal, institutional solutions to AI safety at lab, industry, national and international scale.

And while this is a technical course for developers focusing on the technical aspects of AI safety,



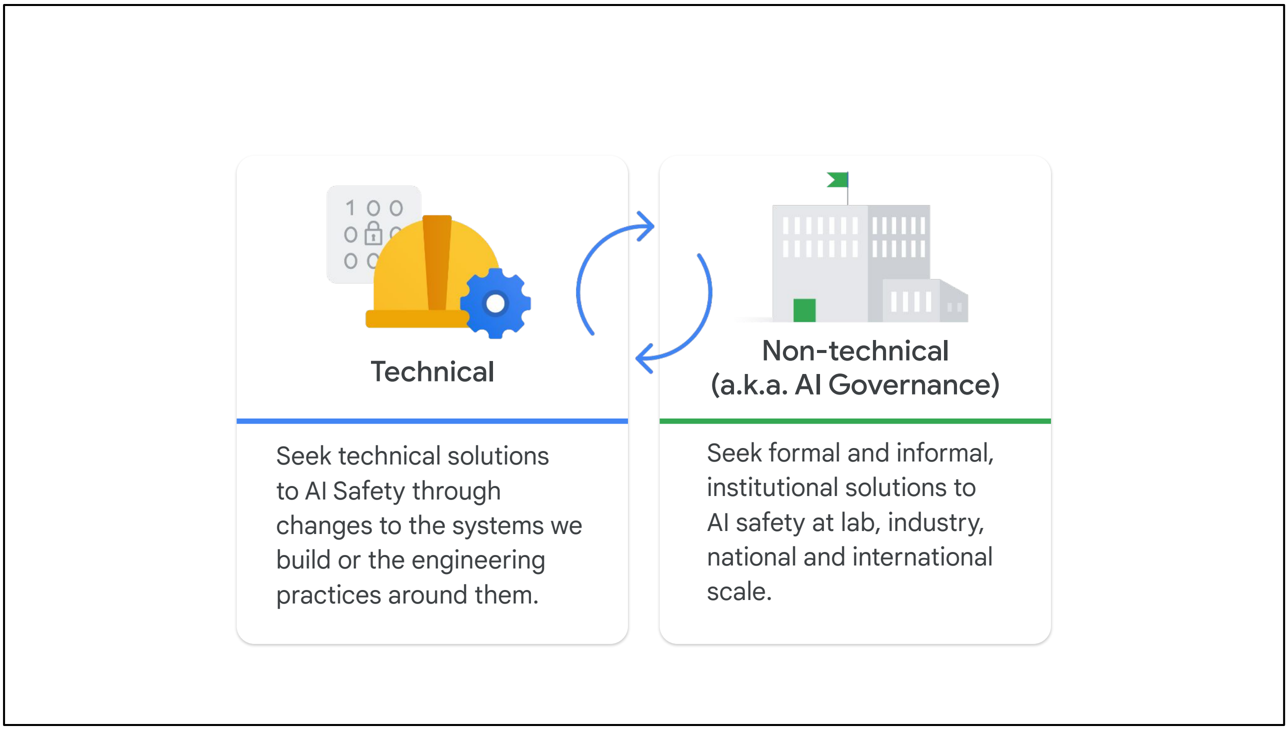
Technical

Seek technical solutions to AI Safety through changes to the systems we build or the engineering practices around them.

Non-technical (a.k.a. AI Governance)

Seek formal and informal, institutional solutions to AI safety at lab, industry, national and international scale.

note that non-technical aspects often promote technical approaches,



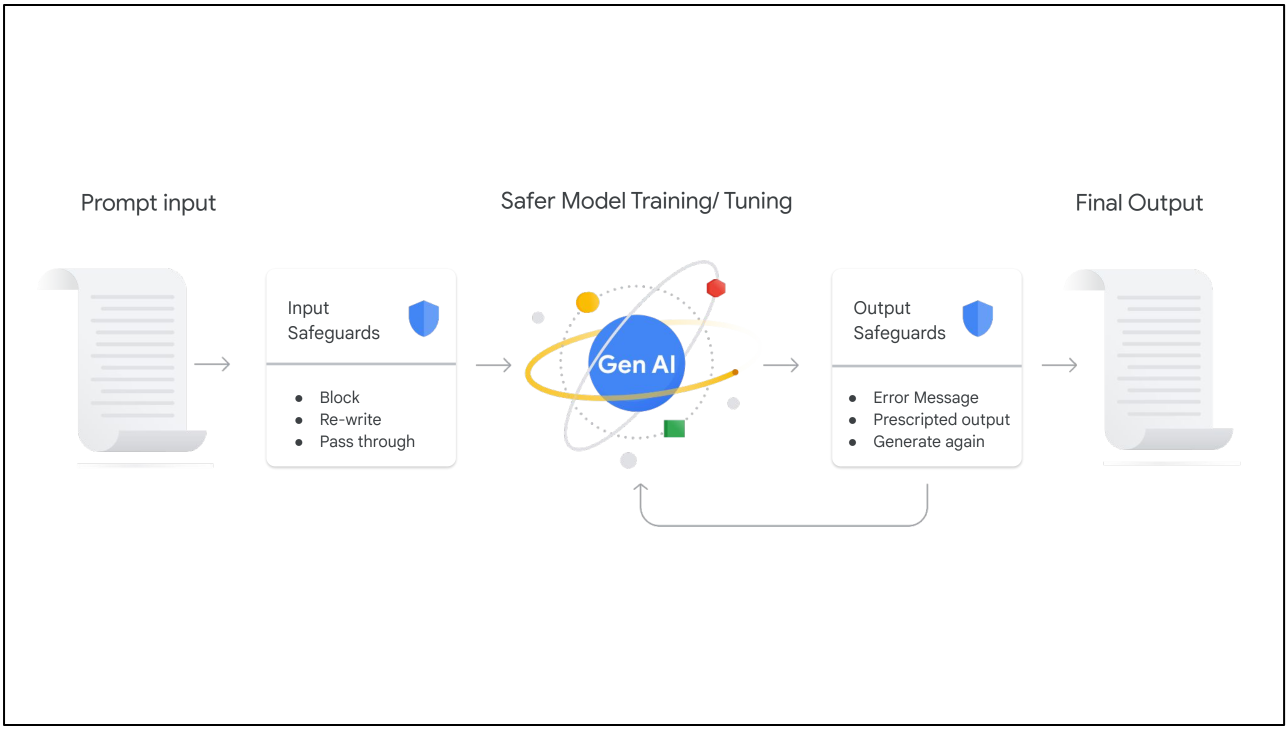
Technical

Seek technical solutions to AI Safety through changes to the systems we build or the engineering practices around them.

Non-technical (a.k.a. AI Governance)

Seek formal and informal, institutional solutions to AI safety at lab, industry, national and international scale.

and they are complementary to each other.



Prompt input

Safer Model Training/ Tuning

Final Output

Input Safeguards

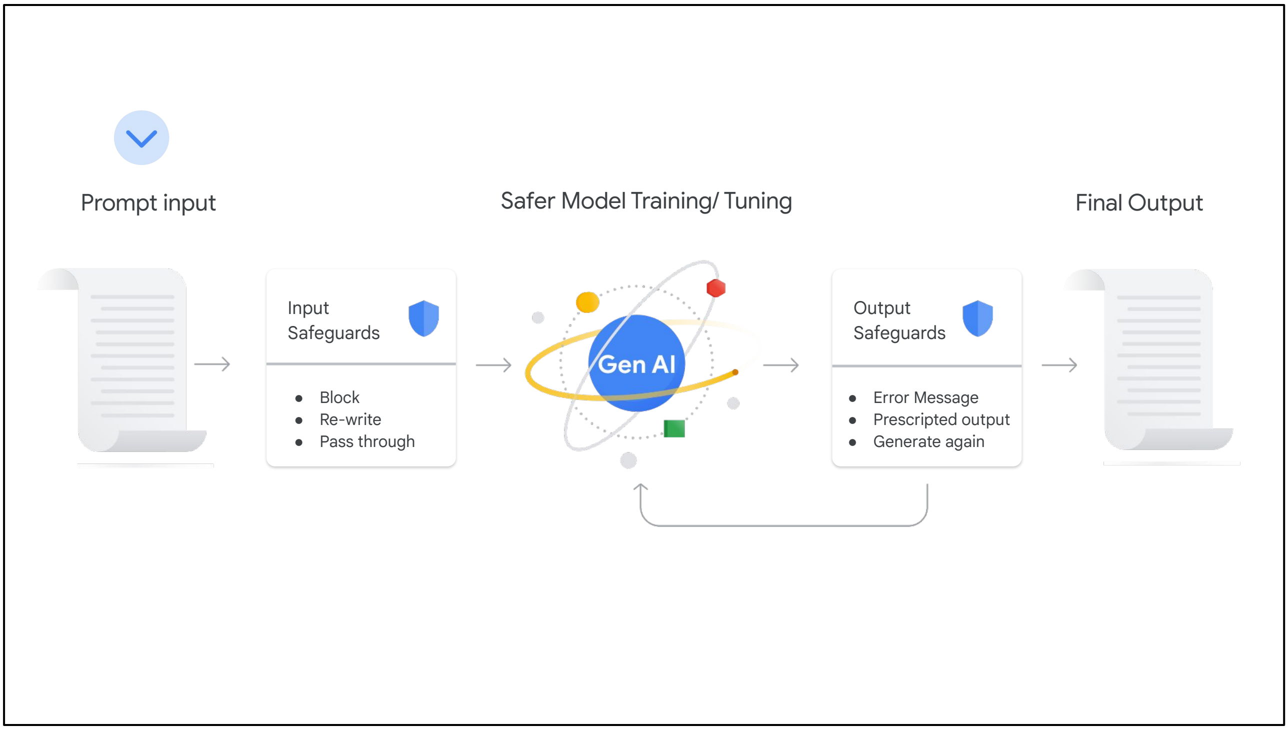
- Block
- Re-write
- Pass through

Output Safeguards

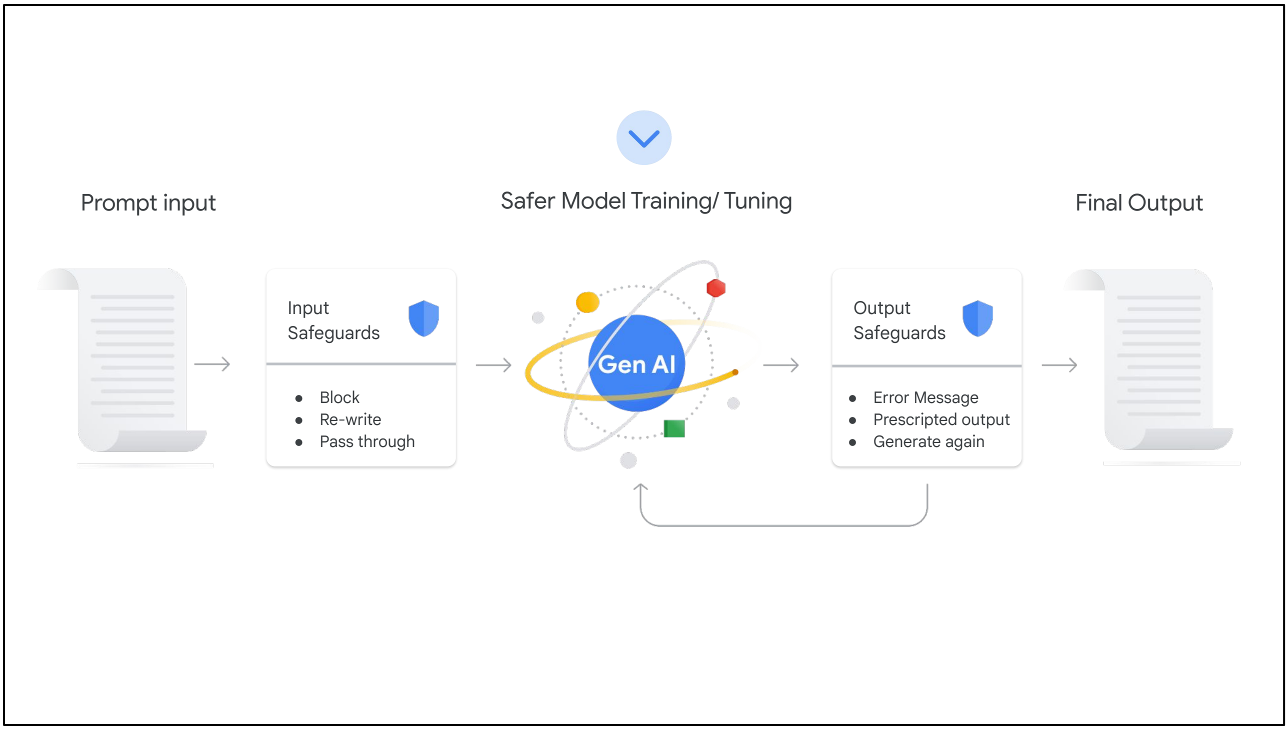
- Error Message
- Prescripted output
- Generate again

So how can we approach AI safety from a technical perspective, especially in Gen AI? There are multiple measures we can take.

This diagram shows an overview of the generative AI system.



The system receives prompt inputs,



Prompt input

Safer Model Training/ Tuning

Final Output

Input Safeguards

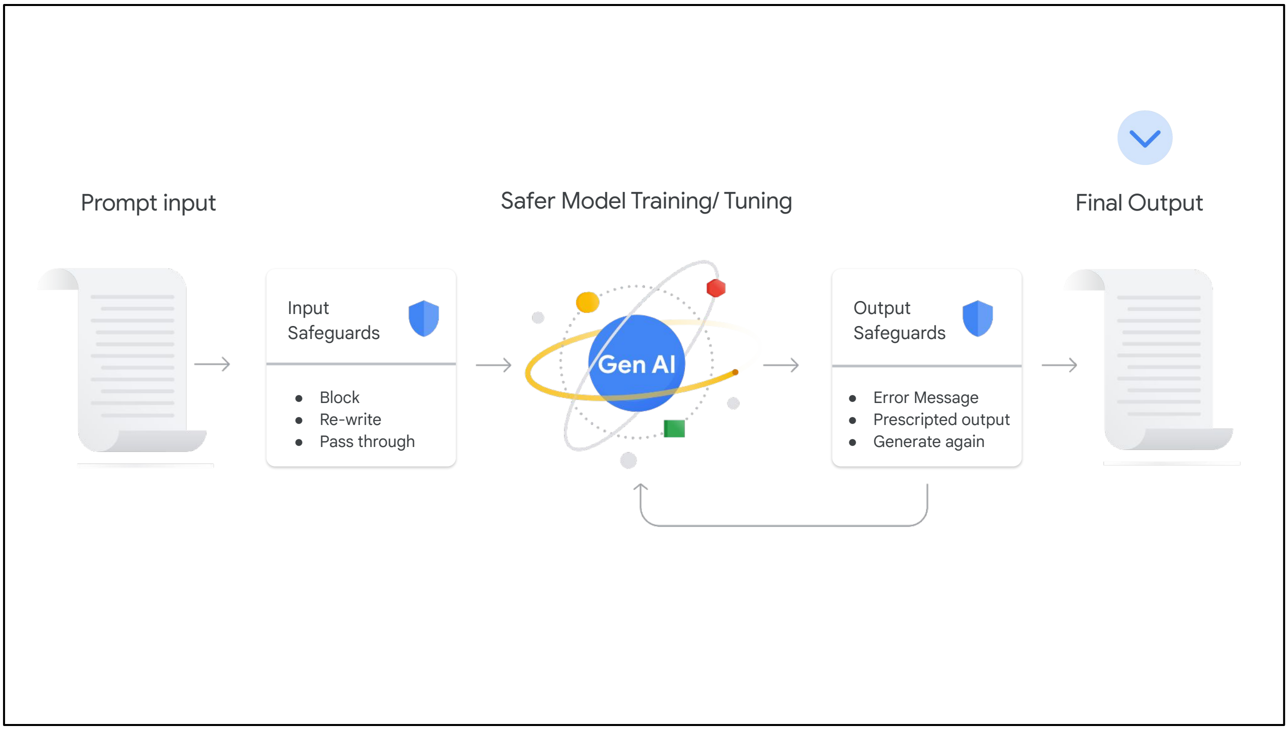
- Block
- Re-write
- Pass through

Gen AI

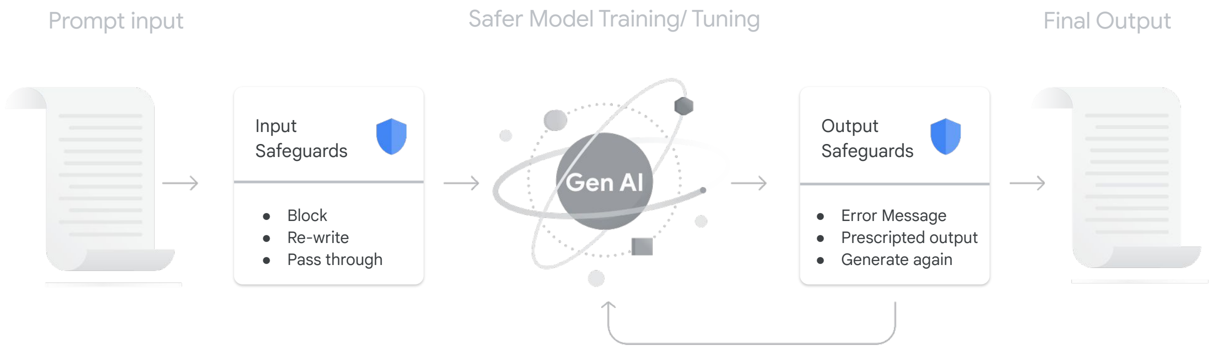
Output Safeguards

- Error Message
- Prescripted output
- Generate again

passes it to the AI model,



and returns outputs.



Prompt input

Input Safeguards

- Block
- Re-write
- Pass through

Safer Model Training/ Tuning

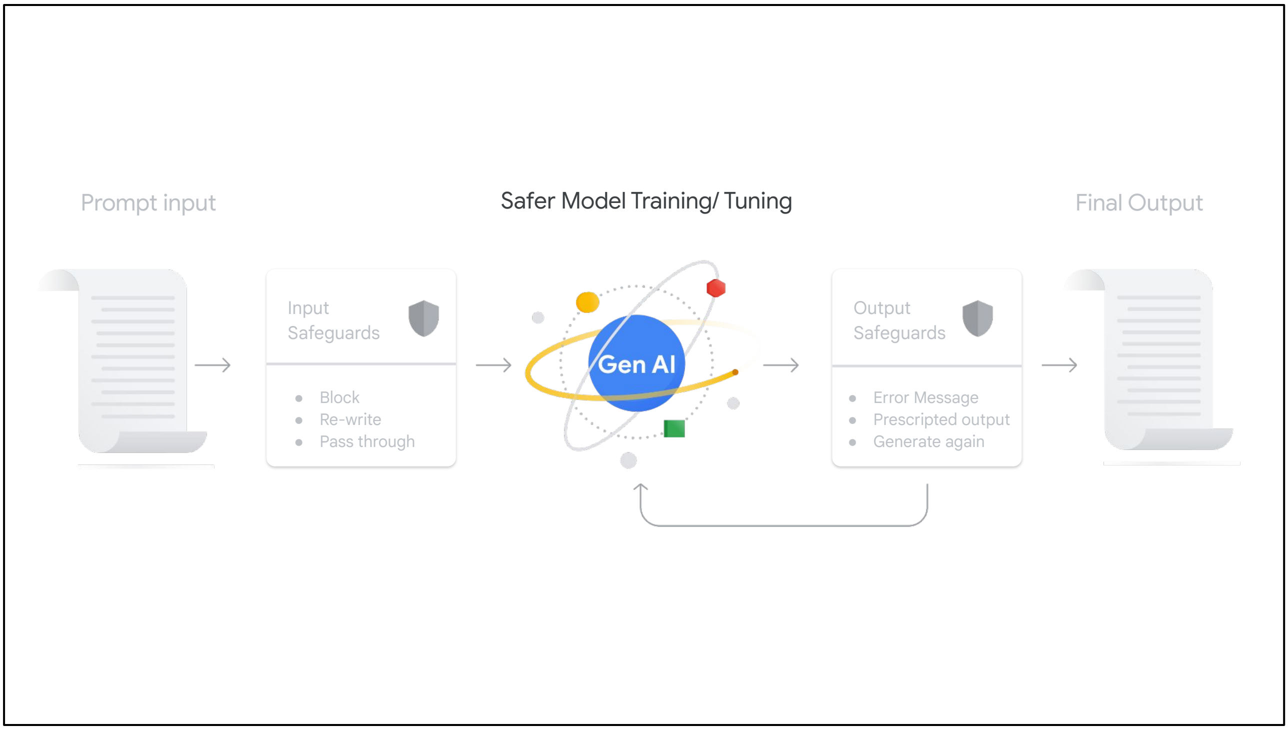
Gen AI

Output Safeguards

- Error Message
- Prescripted output
- Generate again

Final Output

Between the prompt input and the final output, you can see input safeguards and output safeguards. These safeguards take care of input/output filtering based on defined safety criteria. By using safeguards properly, we can harness the system input/output to avoid unintended harmful outcomes.



Prompt input

Safer Model Training/ Tuning

Final Output

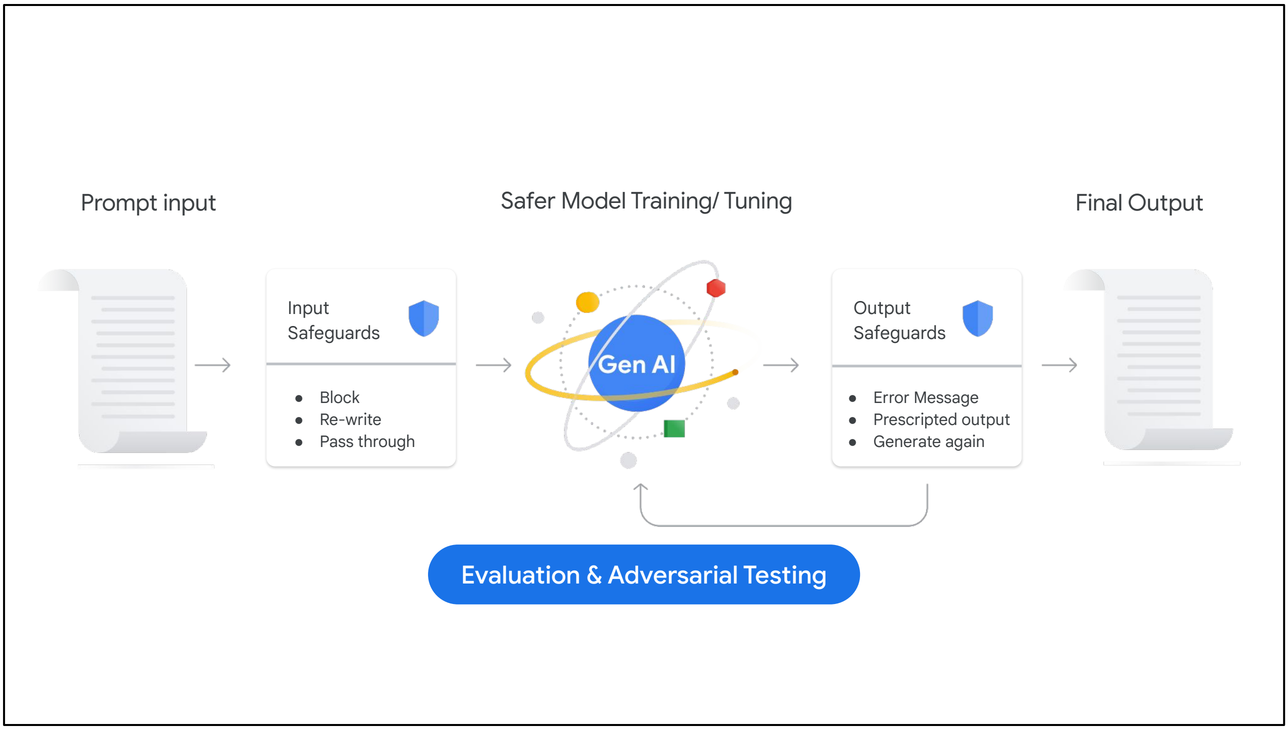
Input Safeguards

- Block
- Re-write
- Pass through

Output Safeguards

- Error Message
- Prescripted output
- Generate again

In addition to safeguards, you can also try to intervene in the model training and tuning process in order to embed safety concepts into the model and align its behavior towards safety.



Prompt input

Safer Model Training/ Tuning

Final Output

Input Safeguards

- Block
- Re-write
- Pass through

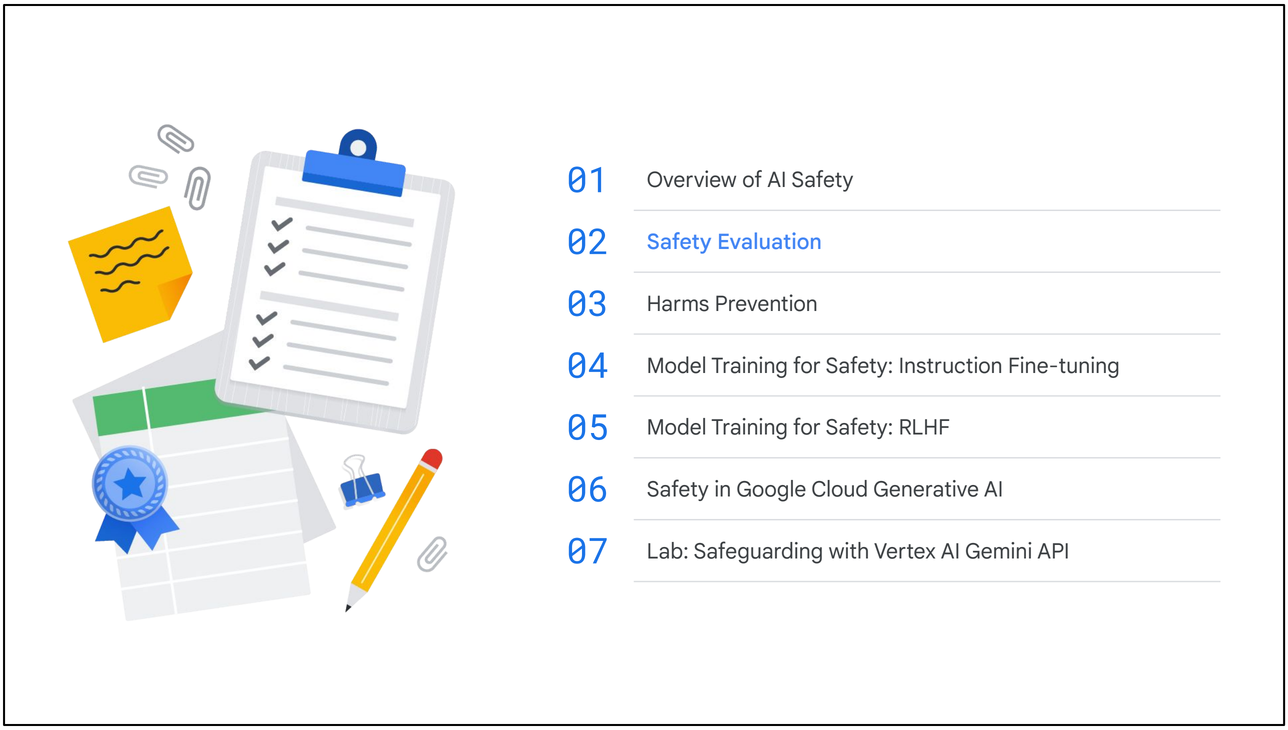
Gen AI

Output Safeguards

- Error Message
- Prescripted output
- Generate again

Evaluation & Adversarial Testing

It is also critical to understand how to evaluate AI systems throughout the entire process. Evaluation involves exhaustive testing to find an unintended behavior of the system. This is called “adversarial testing.”
We'll look at each item as we progress through this course.



01 Overview of AI Safety

02 Safety Evaluation

03 Harms Prevention

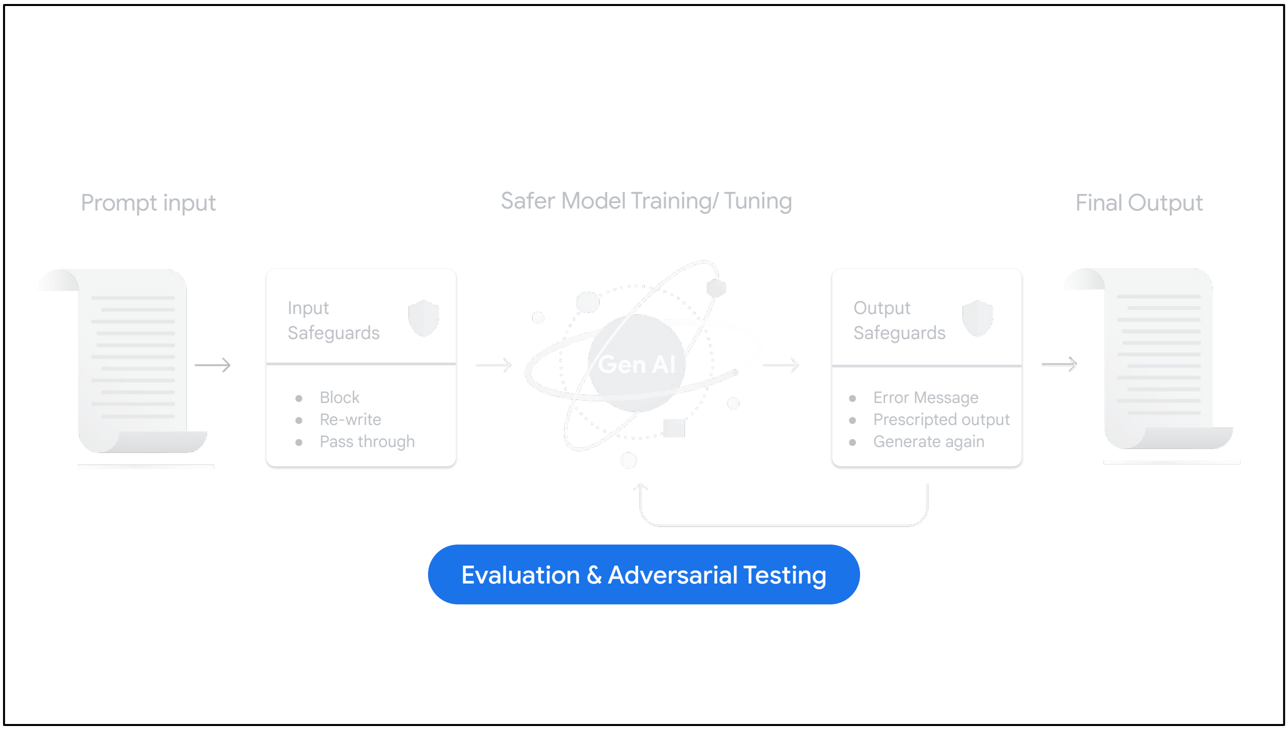
04 Model Training for Safety: Instruction Fine-tuning

05 Model Training for Safety: RLHF

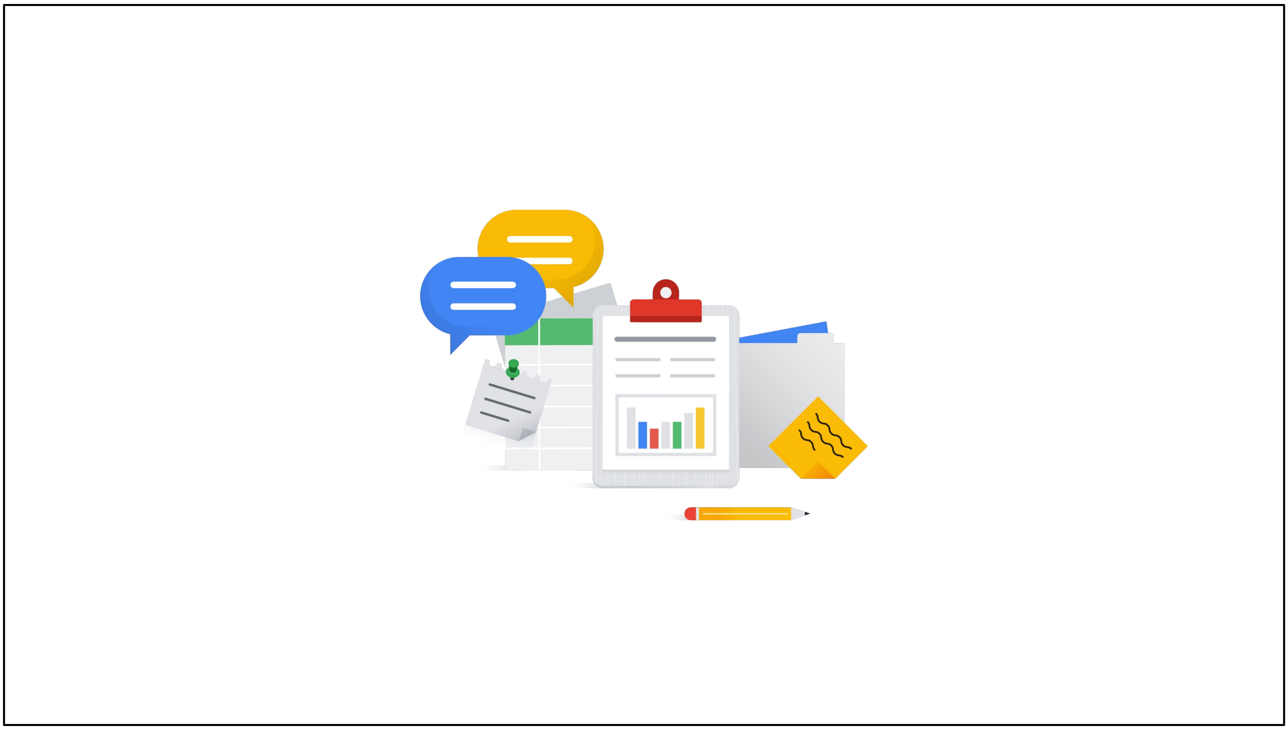
06 Safety in Google Cloud Generative AI

07 Lab: Safeguarding with Vertex AI Gemini API

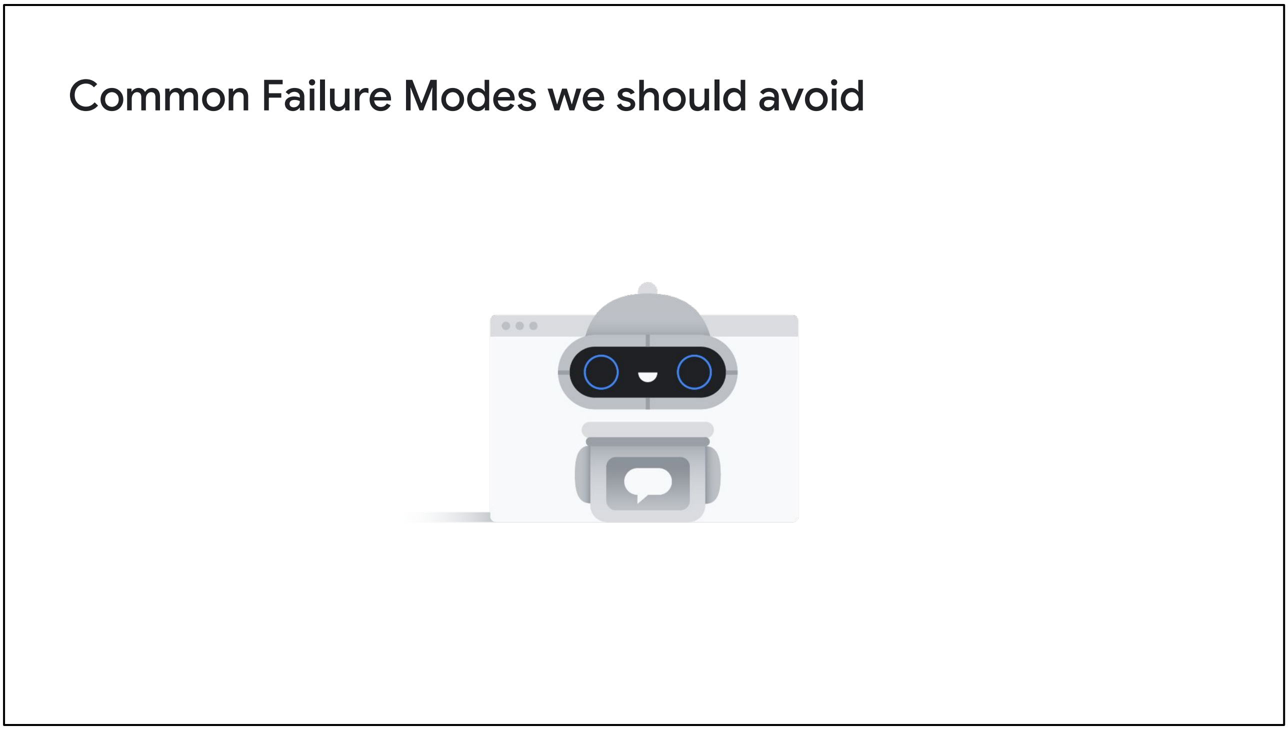
Let’s explore safety evaluation.



Without understanding the safety criteria, it is not possible to design and implement a safer AI system. Numerous key considerations should be taken into account.
Let's check the major failure modes and considerations we need to understand.

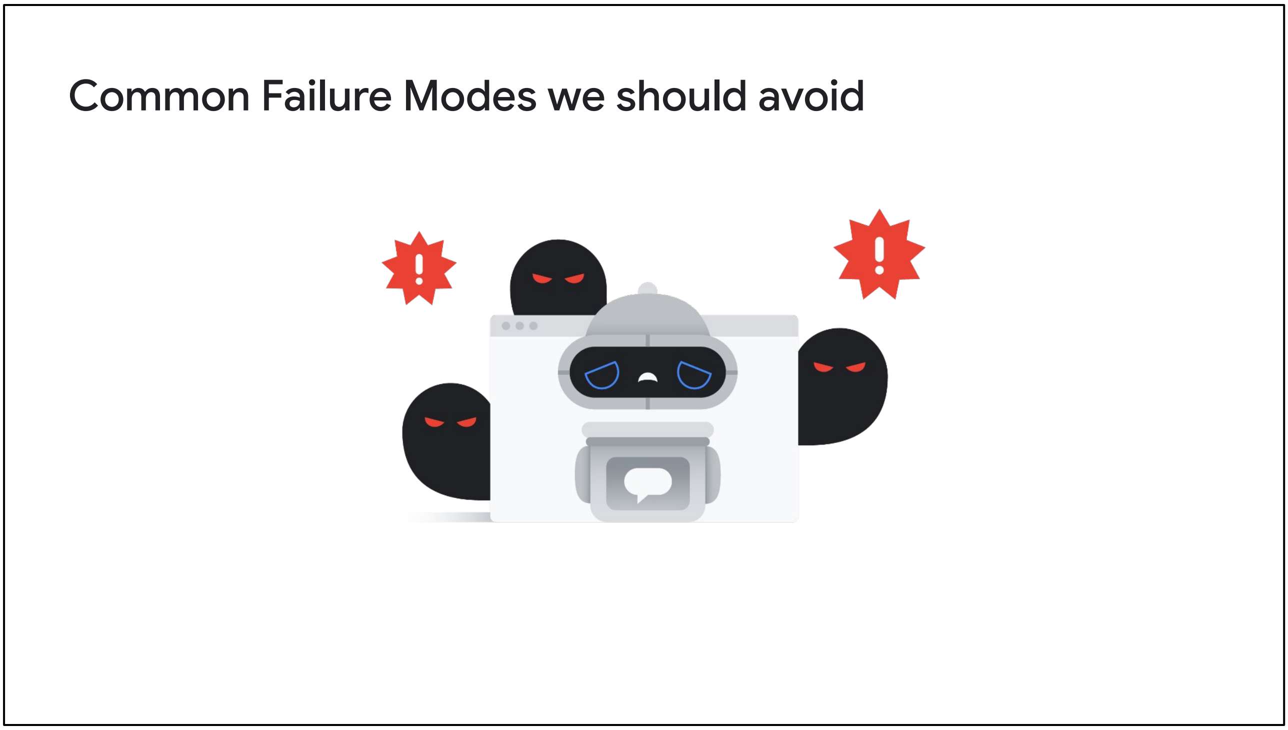


First of all, you need to understand your product and users to define the failure mode for your product.
Different products should have different definitions and goals related to some sensitive characteristics.



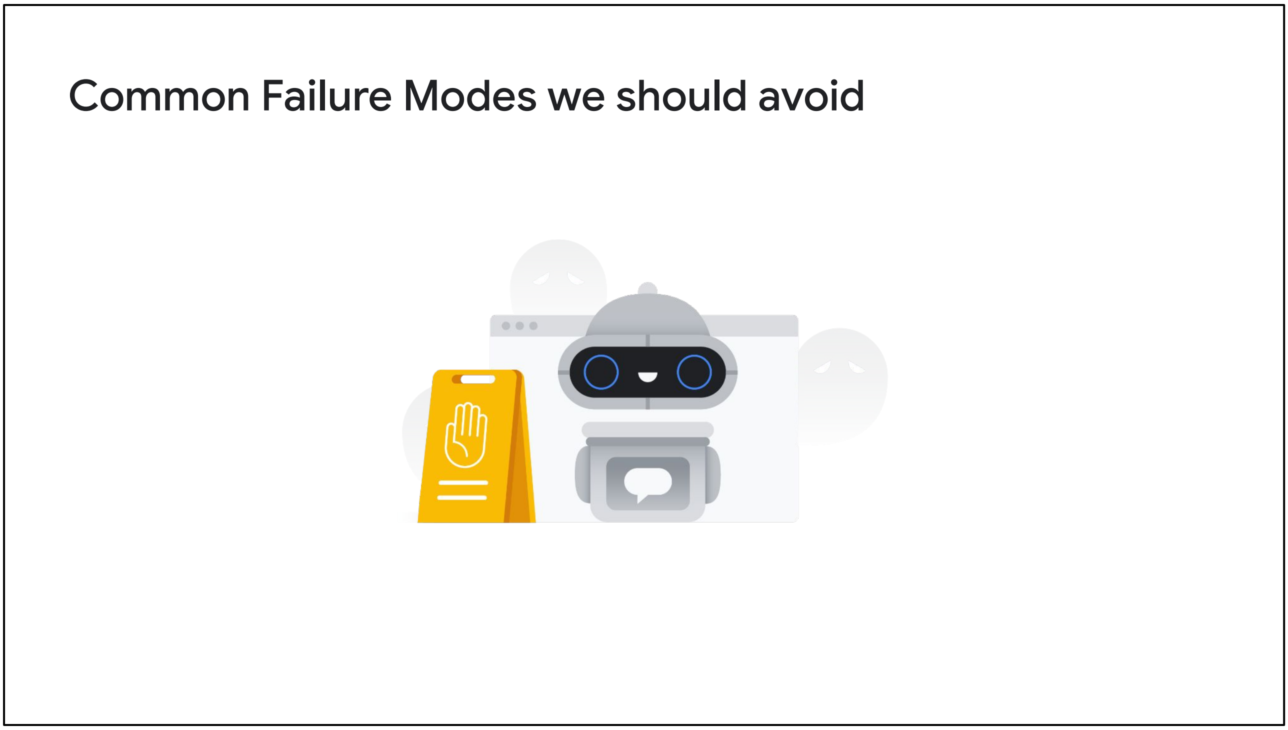
Common Failure Modes we should avoid

However,



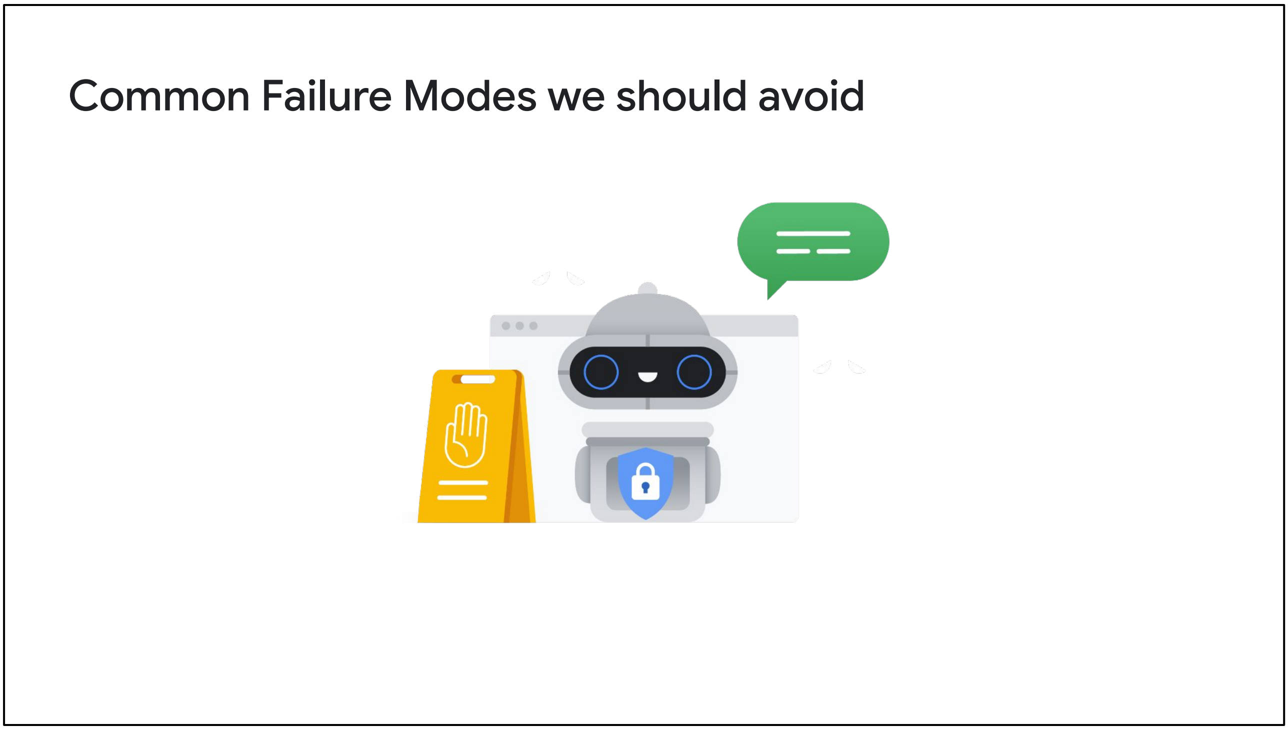
Common Failure Modes we should avoid

there are some common failure modes



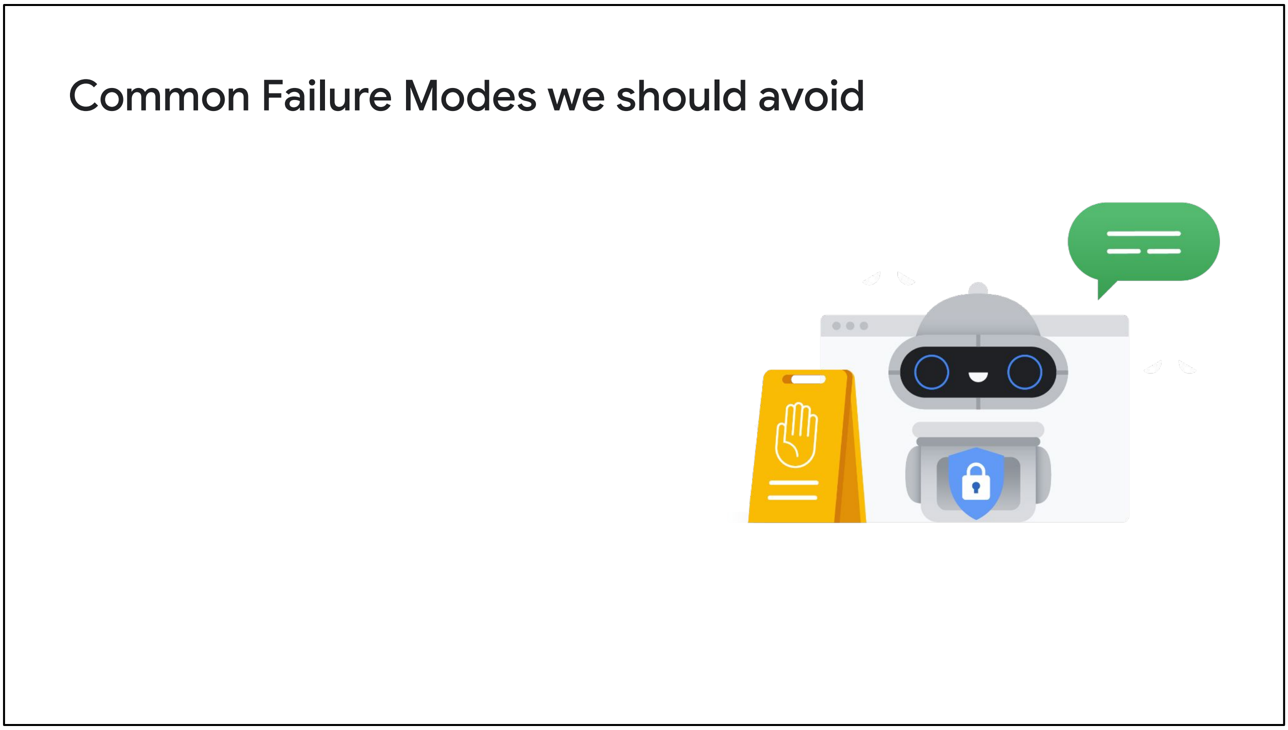
Common Failure Modes we should avoid

you should prohibit



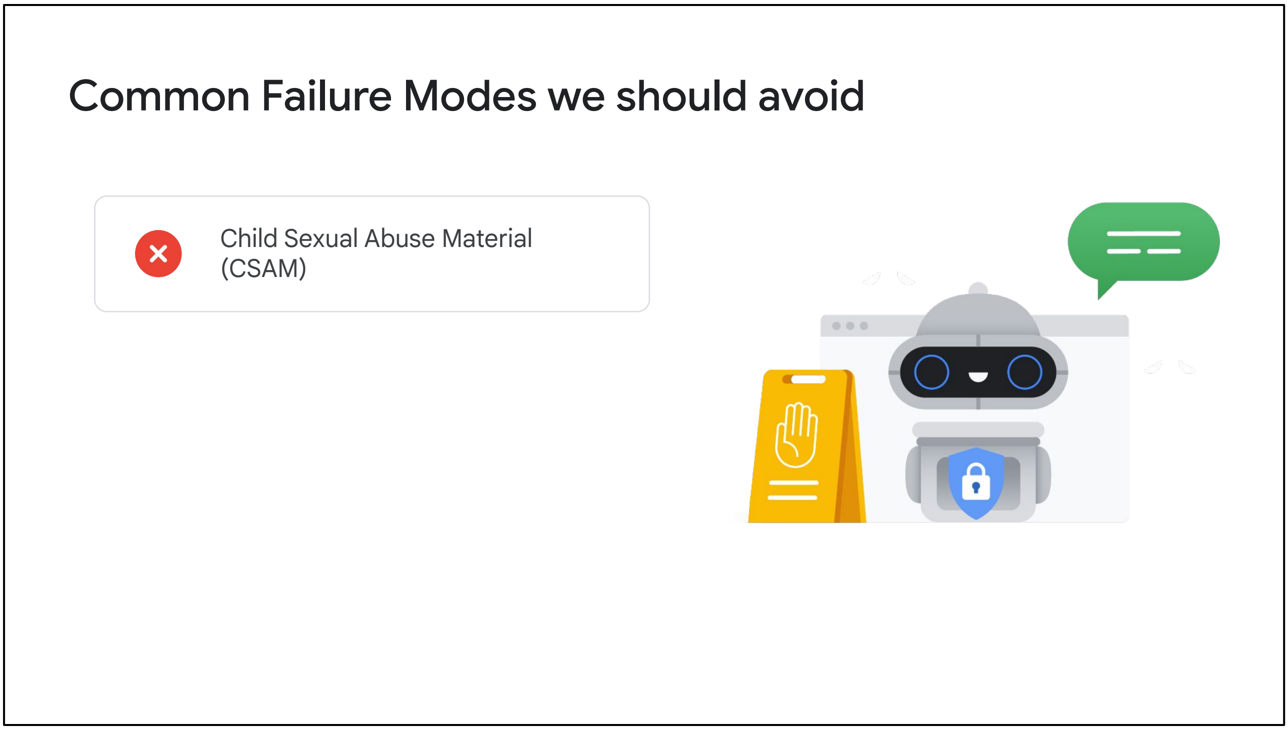
Common Failure Modes we should avoid

for safety purposes.



Common Failure Modes we should avoid

Generative AI systems should be designed to avoid providing:

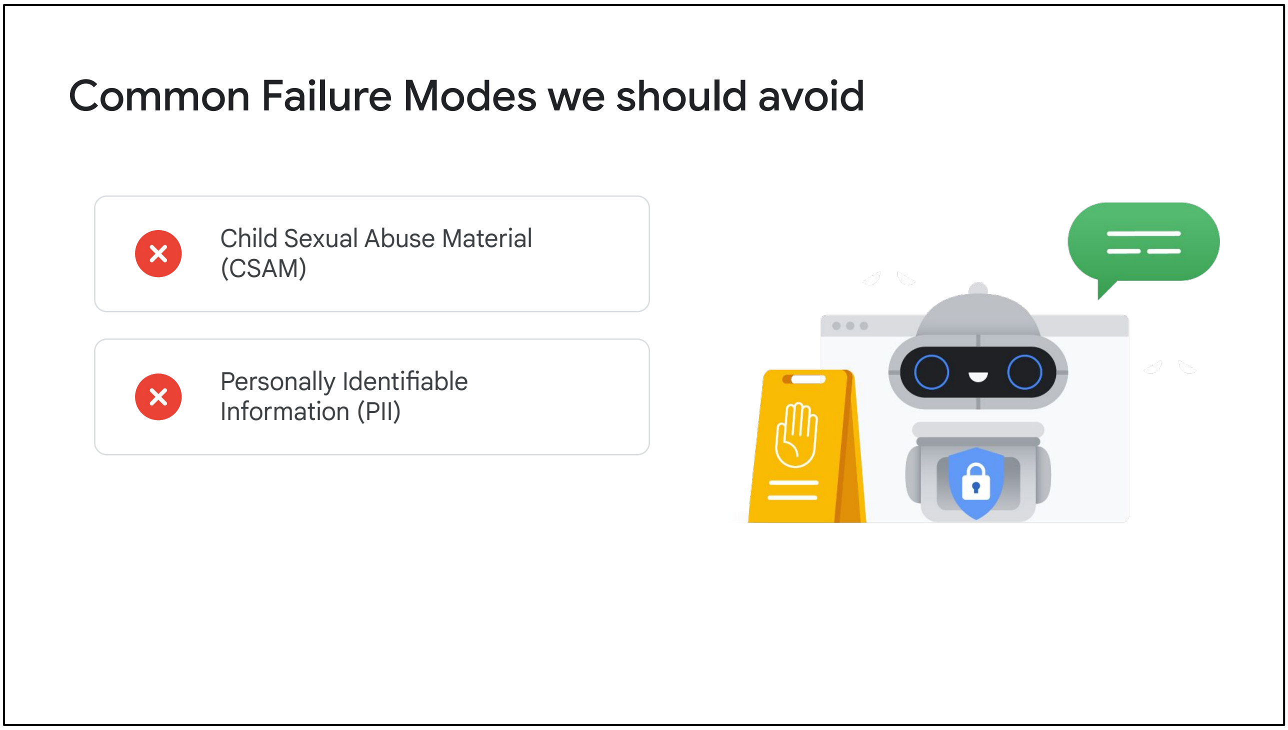


Common Failure Modes we should avoid

✖ Child Sexual Abuse Material (CSAM)

- Child Sexual Abuse Material, or CSAM

Content on, or from, generative AI products must not include child sexual abuse material nor content that appears to abuse, sexualize, endanger, or otherwise exploit children.



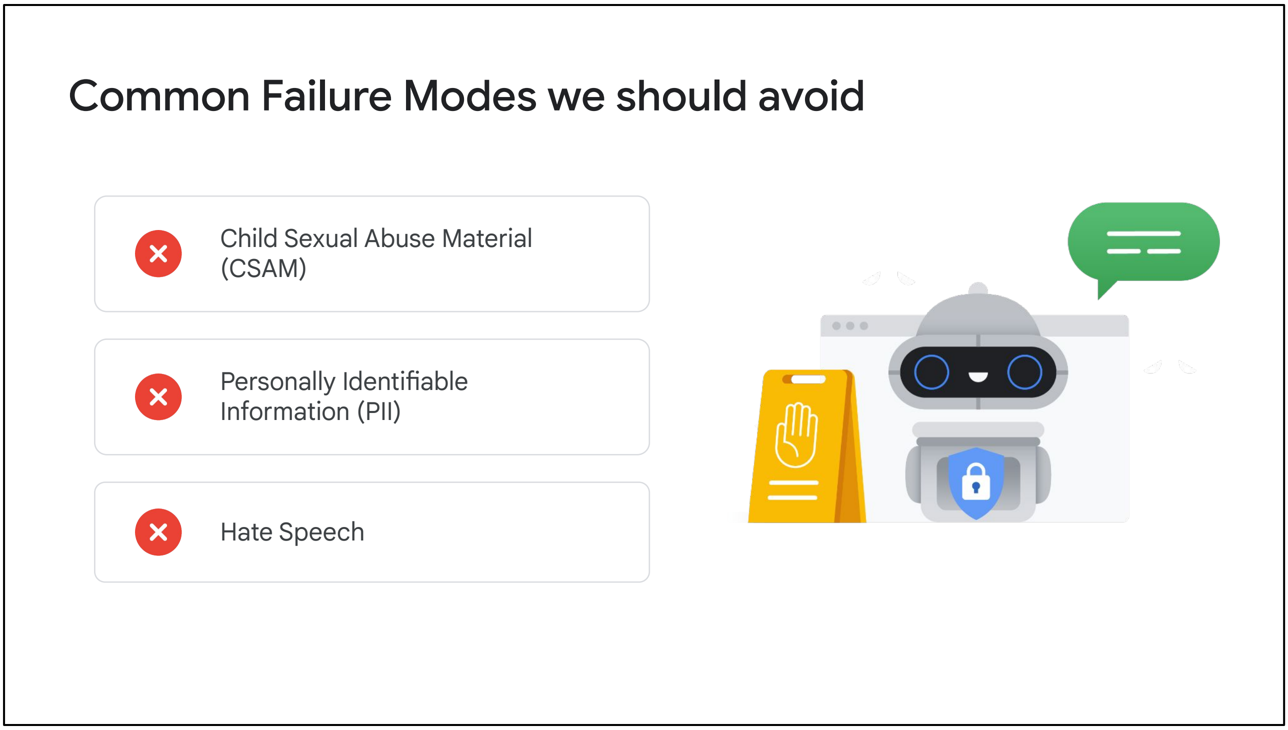
Common Failure Modes we should avoid

❌ Child Sexual Abuse Material (CSAM)

❌ Personally Identifiable Information (PII)

- Personally identifiable information (or PII) and sensitive personally identifiable information (or SPII) of specific individuals

Generative AI should not reveal an individual's personal information and sensitive demographic information of specific individuals.

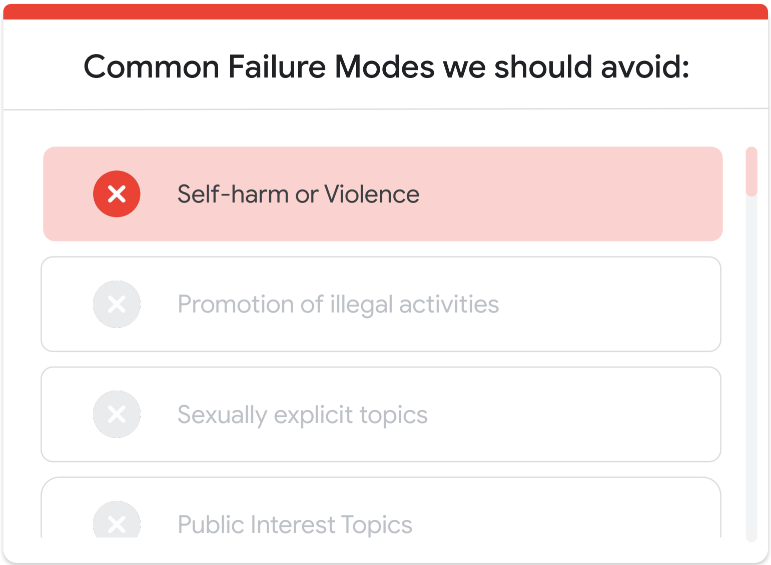


Common Failure Modes we should avoid

- ✖ Child Sexual Abuse Material (CSAM)
- ✖ Personally Identifiable Information (PII)
- ✖ Hate Speech

- And hate speech.

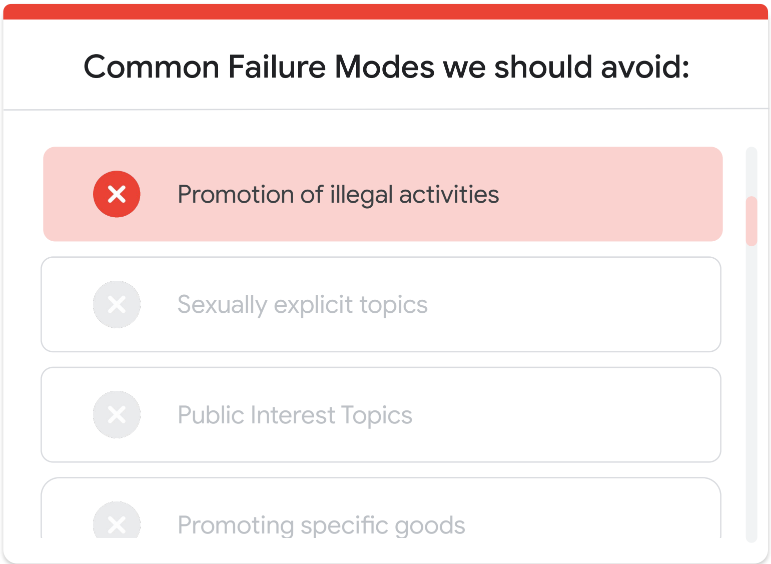
Generative AI should not generate hate speech, defined as content that promotes violence, incites hatred, promotes discrimination, or disparages towards any groups.



Common Failure Modes we should avoid:

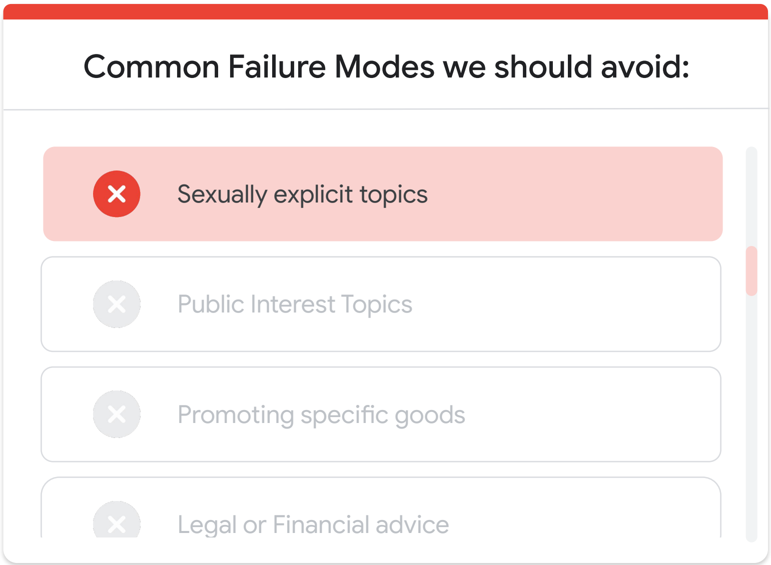
- Self-harm or Violence
- Promotion of illegal activities
- Sexually explicit topics
- Public Interest Topics

There are many other common failure modes we usually need to avoid.
However, the actual nuance and the threshold depends on actual products. To control the balance, it is also very important to design a system that we can flexibly control and adjust based on the needs and context of users.



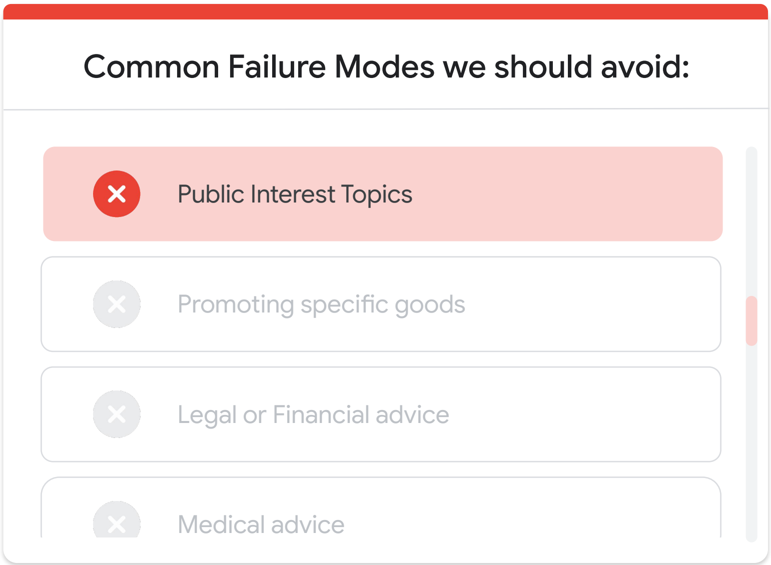
Common Failure Modes we should avoid:

- ✕ Promotion of illegal activities
- ✕ Sexually explicit topics
- ✕ Public Interest Topics
- ✕ Promoting specific goods



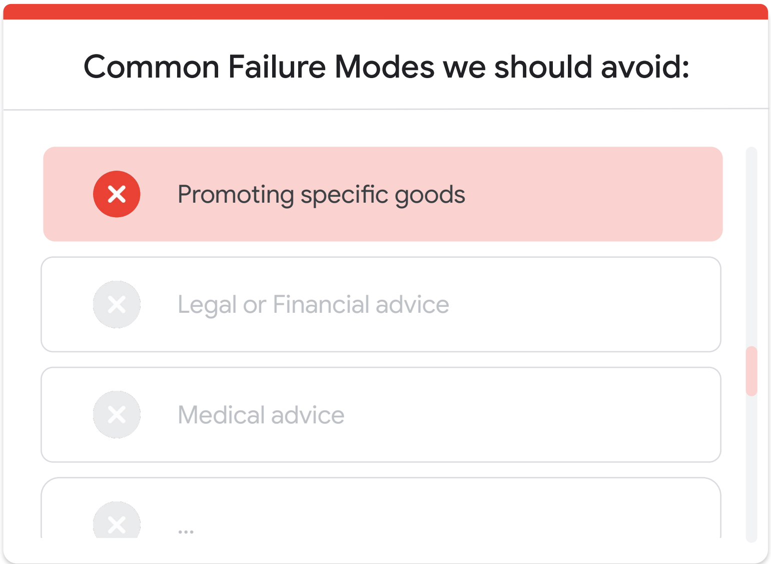
Common Failure Modes we should avoid:

- ✕ Sexually explicit topics
- ✕ Public Interest Topics
- ✕ Promoting specific goods
- ✕ Legal or Financial advice



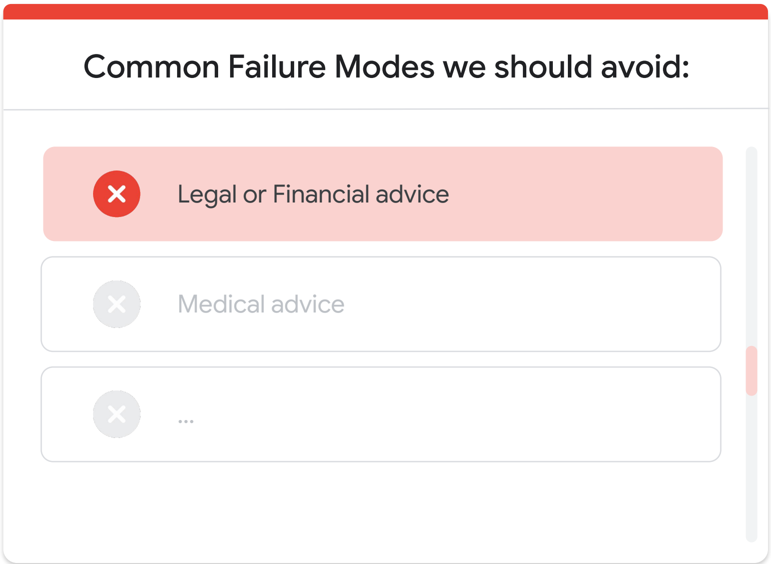
Common Failure Modes we should avoid:

- Public Interest Topics
- Promoting specific goods
- Legal or Financial advice
- Medical advice



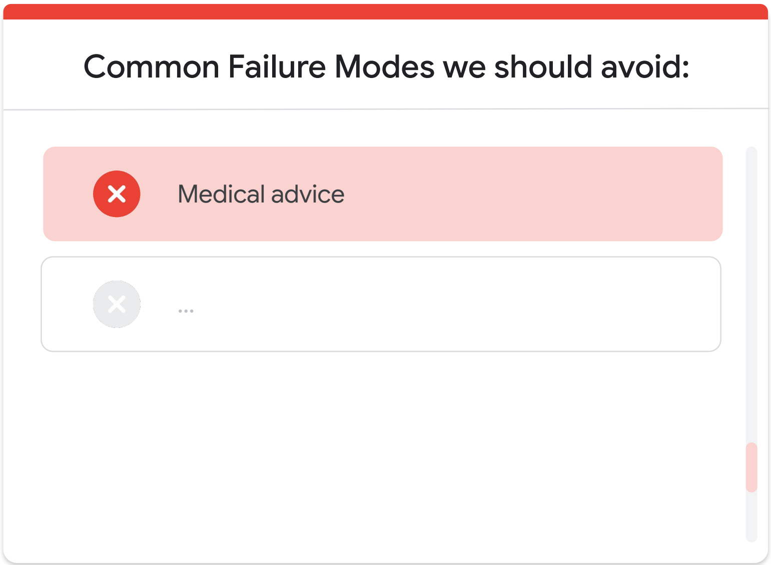
Common Failure Modes we should avoid:

- ✕ Promoting specific goods
- ✕ Legal or Financial advice
- ✕ Medical advice
- ✕ ...



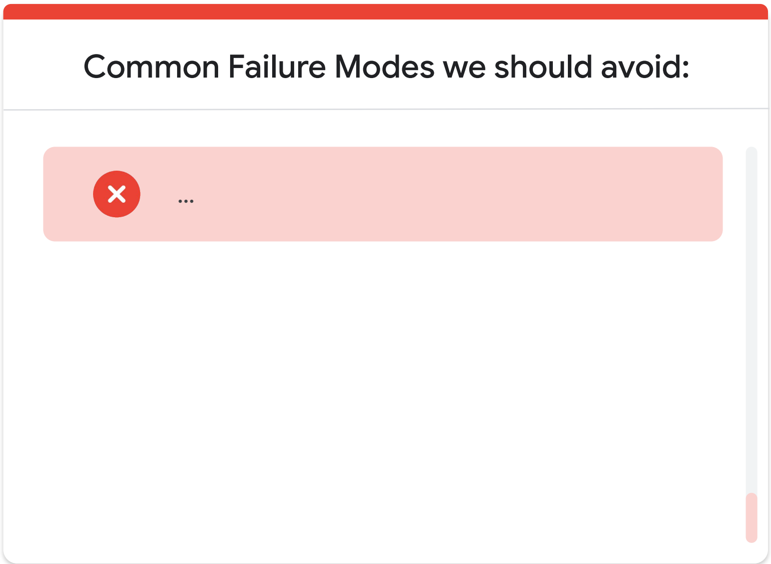
Common Failure Modes we should avoid:

- ✕ Legal or Financial advice
- ✕ Medical advice
- ✕ ...



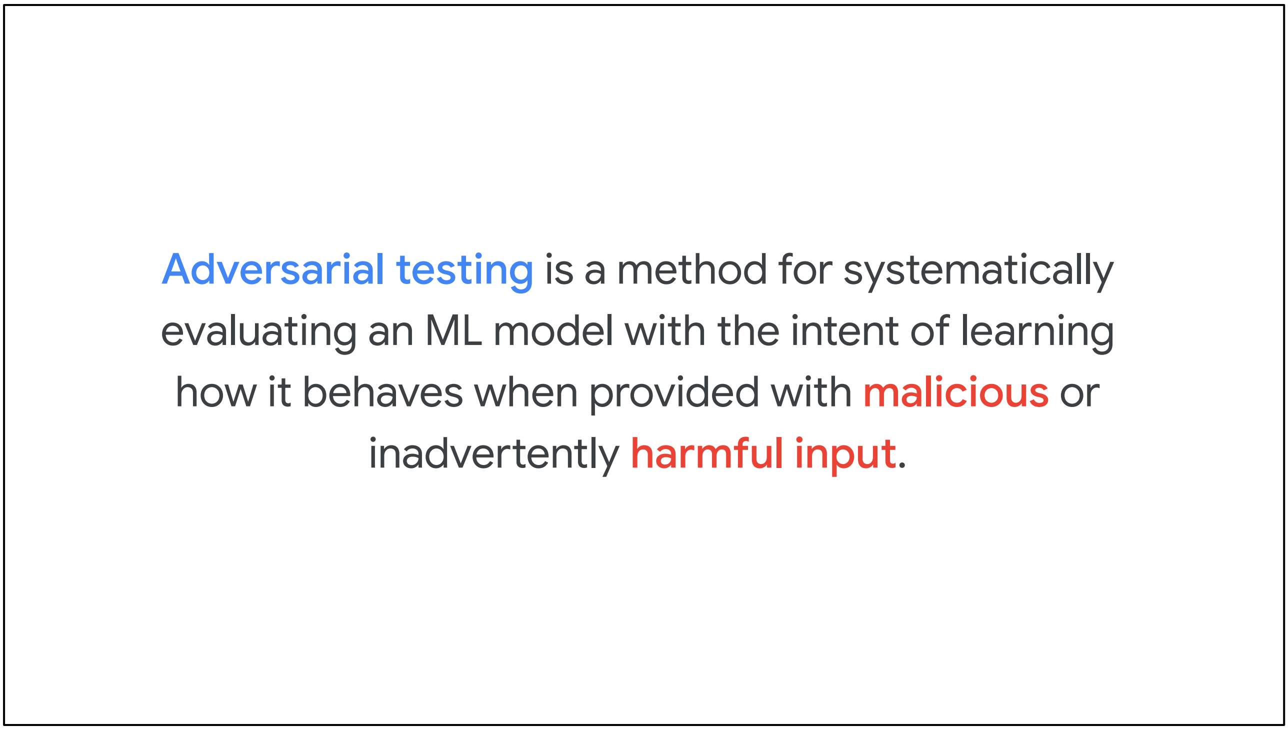
Common Failure Modes we should avoid:

- ☒ Medical advice
- ☒ ...



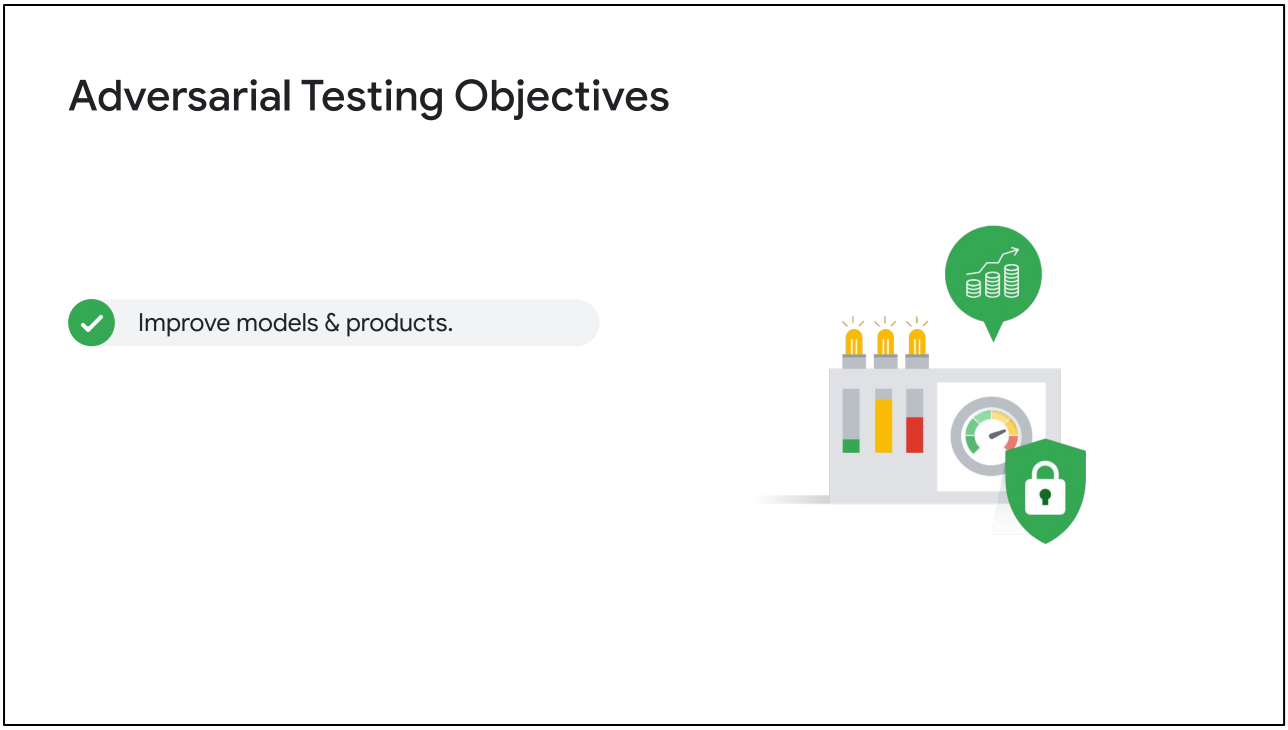
Common Failure Modes we should avoid:

- ✗ ...



Adversarial testing is a method for systematically evaluating an ML model with the intent of learning how it behaves when provided with **malicious** or inadvertently **harmful input**.

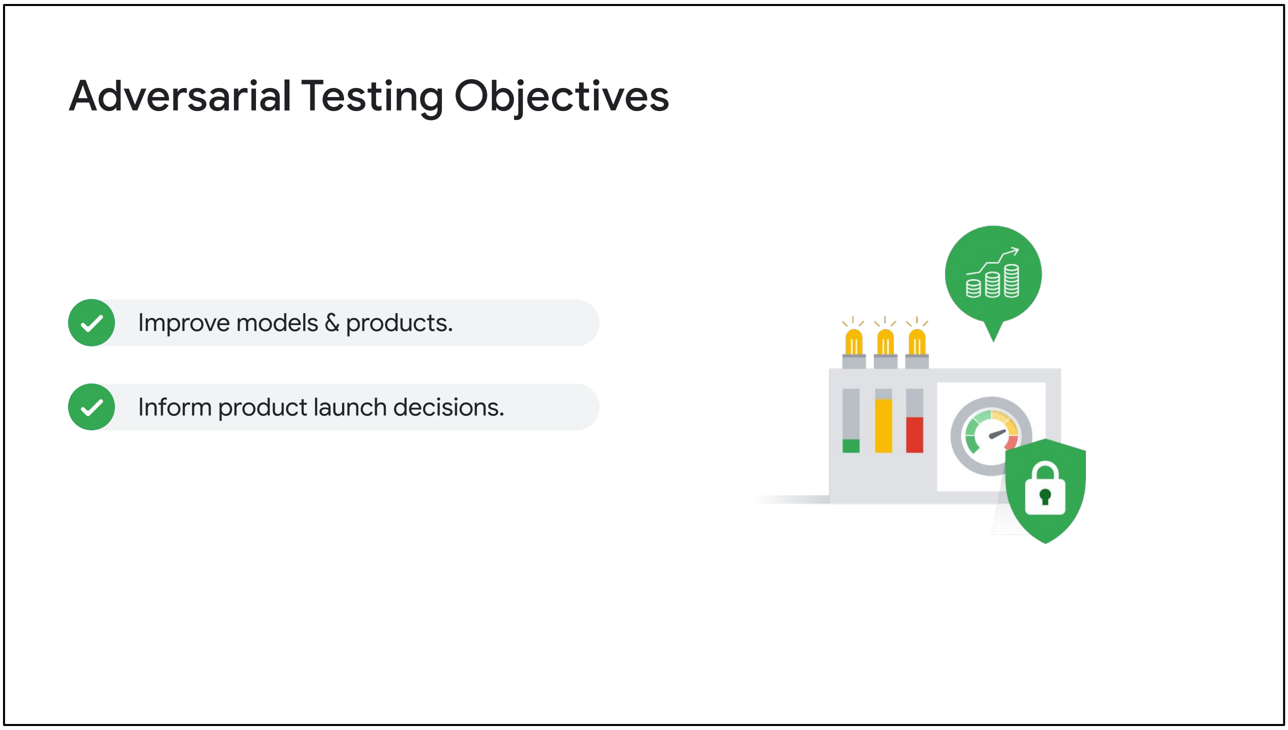
Because it is important to ensure that your generative AI model is safe with respect to the failure modes, this is where adversarial testing comes in. Adversarial testing is a method for systematically evaluating an ML model, with the intent of learning how it behaves when provided with malicious or inadvertently harmful input.



Adversarial Testing Objectives

✓ Improve models & products.

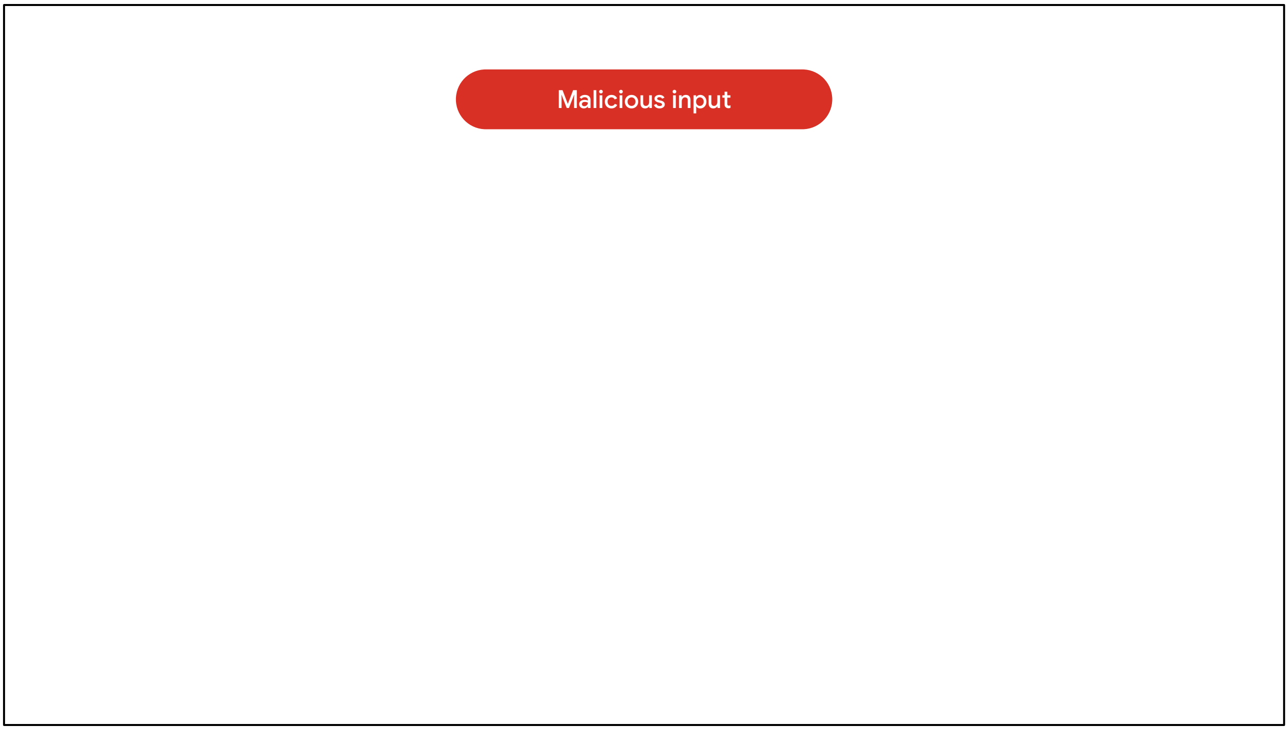
Adversarial testing can help teams improve models and products by exposing current failures to guide mitigation pathways, such as fine tuning, model safeguards or filters.



Adversarial Testing Objectives

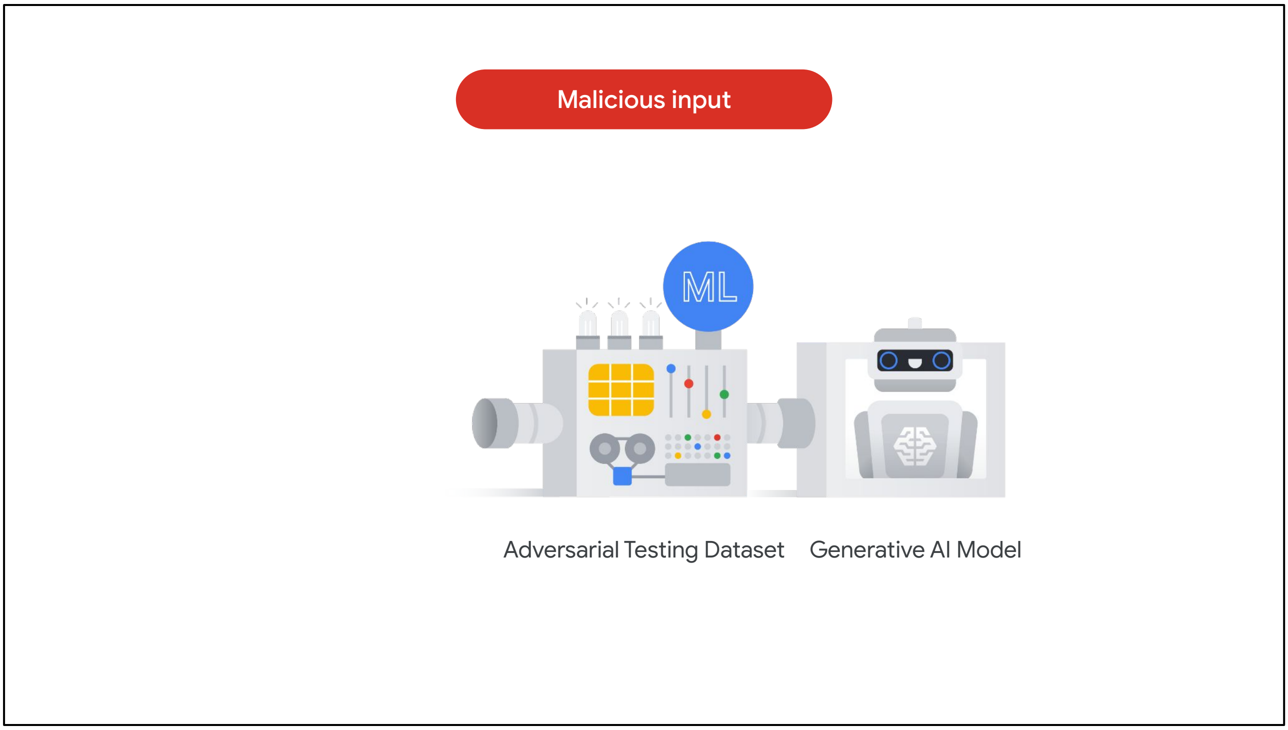
- ✓ Improve models & products.
- ✓ Inform product launch decisions.

Moreover, it can help inform product launch decisions by measuring risks that may be unmitigated, such as the likelihood that the model with output policy-violating content.
But what is considered malicious input and inadvertently harmful input?



Malicious input

An input is considered malicious when the input is clearly designed to produce an unsafe or harmful output.

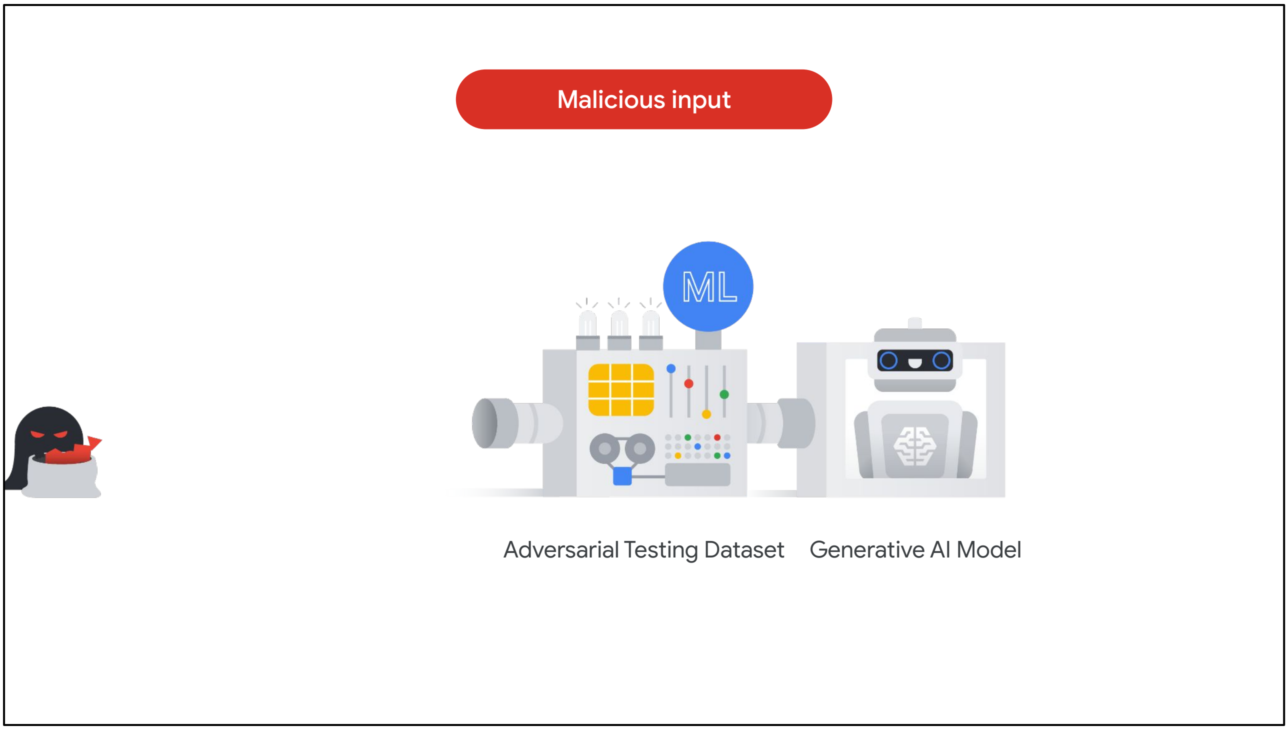


Malicious input

Adversarial Testing Dataset

Generative AI Model

For example,

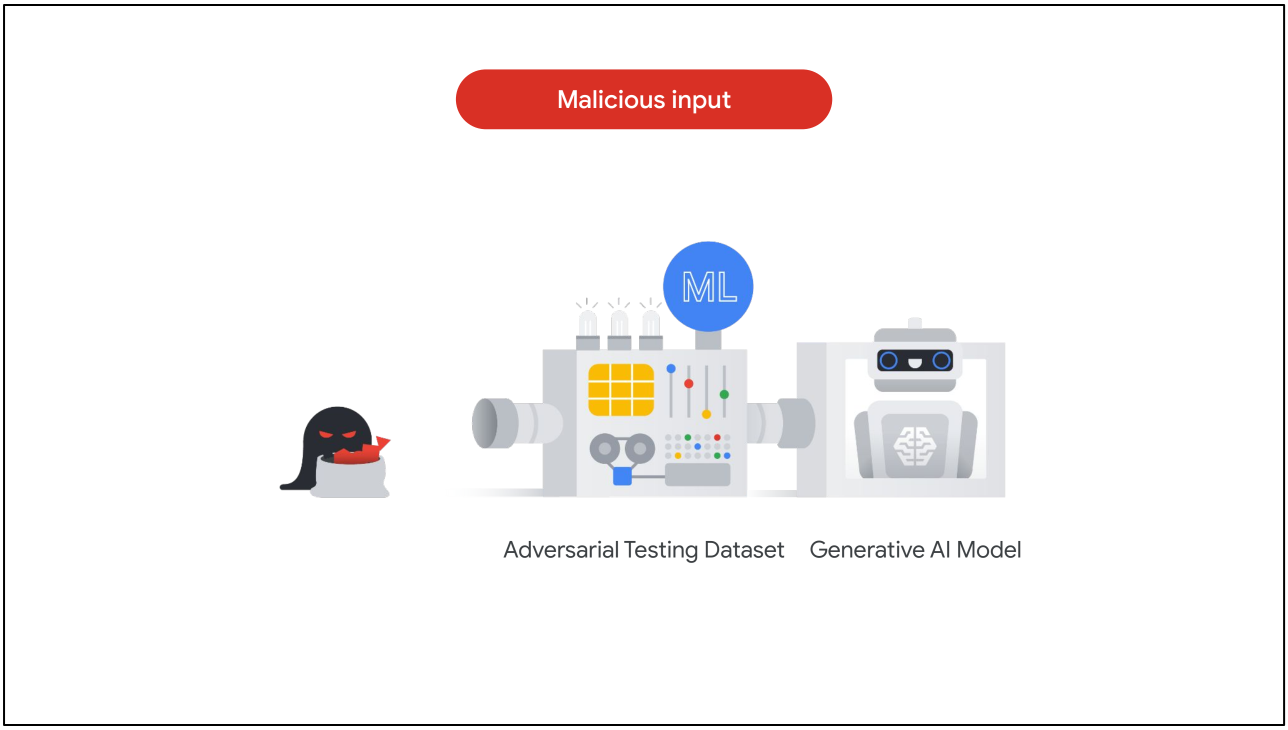


Malicious input

Adversarial Testing Dataset

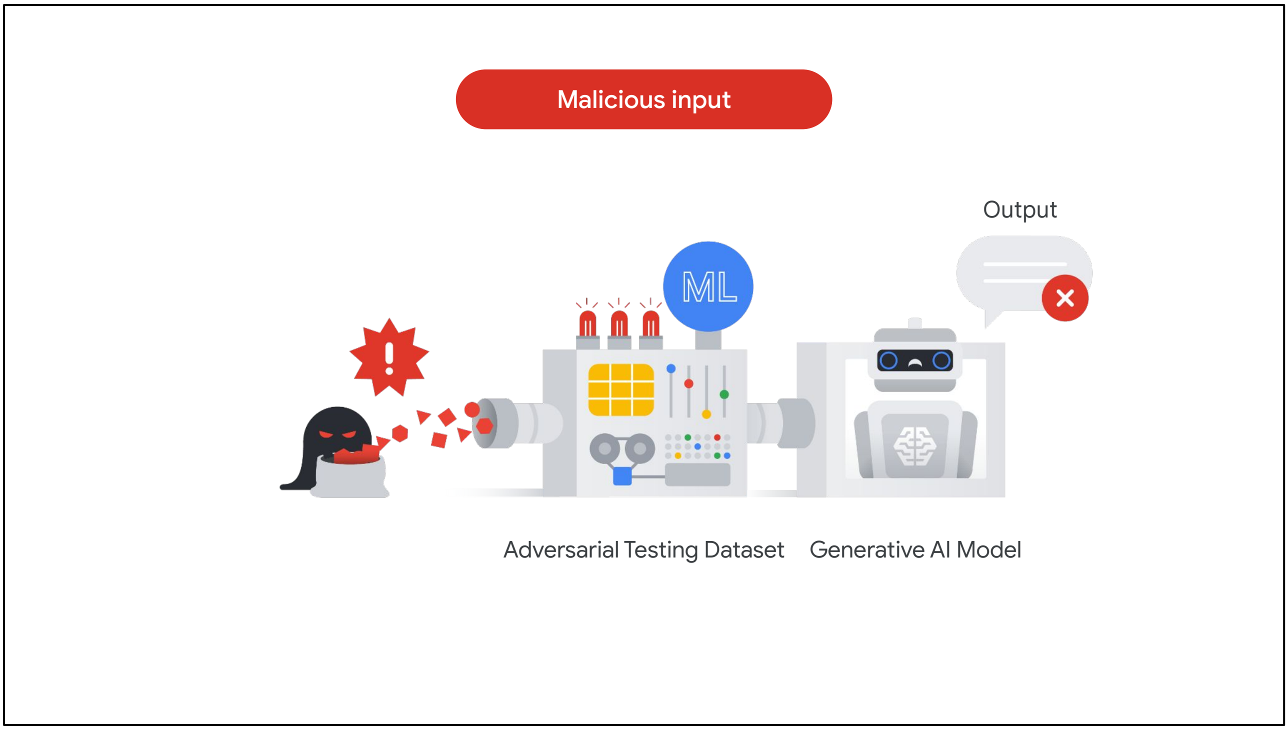
Generative AI Model

an input



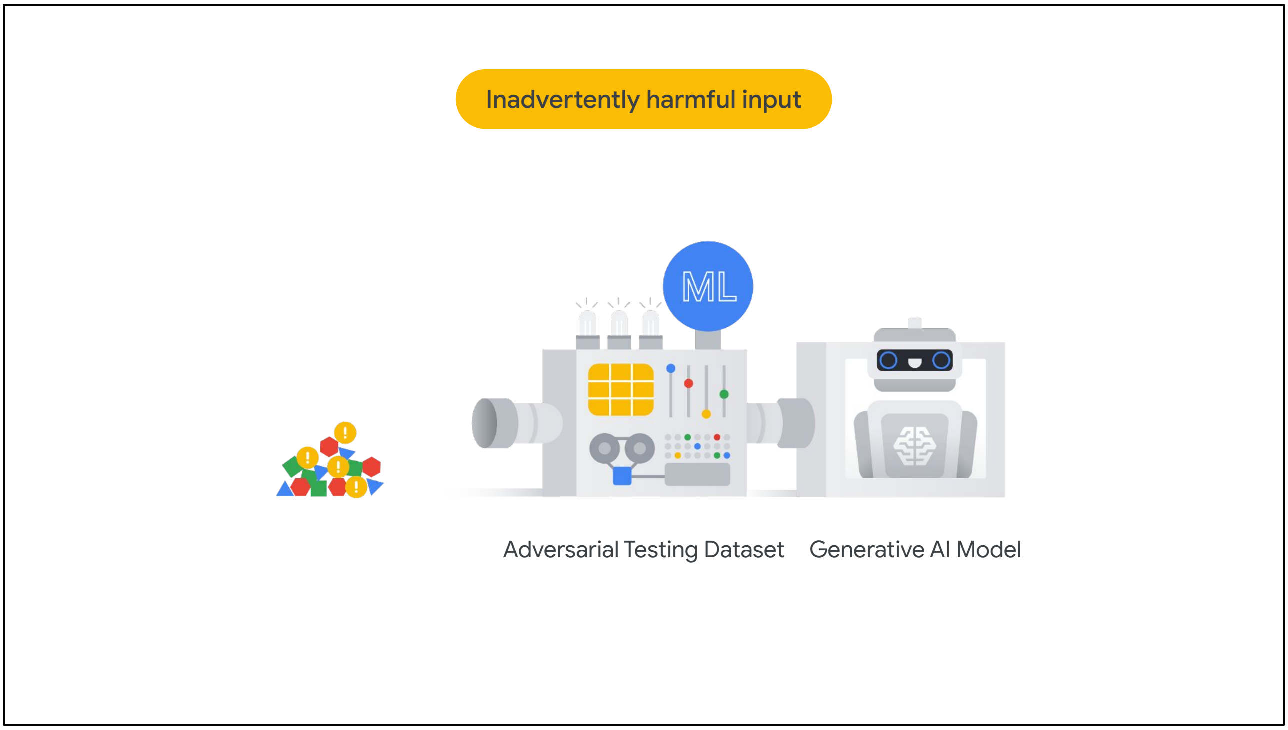
Malicious input

that asks a text generation model to generate a hateful speech about a particular minority group would be considered malicious input.
This is also known as explicitly adversarial queries.



Malicious input

These types of queries may contain policy-violating language or express policy-violating points of view in an attempt to “trick” the model into saying something unsafe, harmful, or offensive.

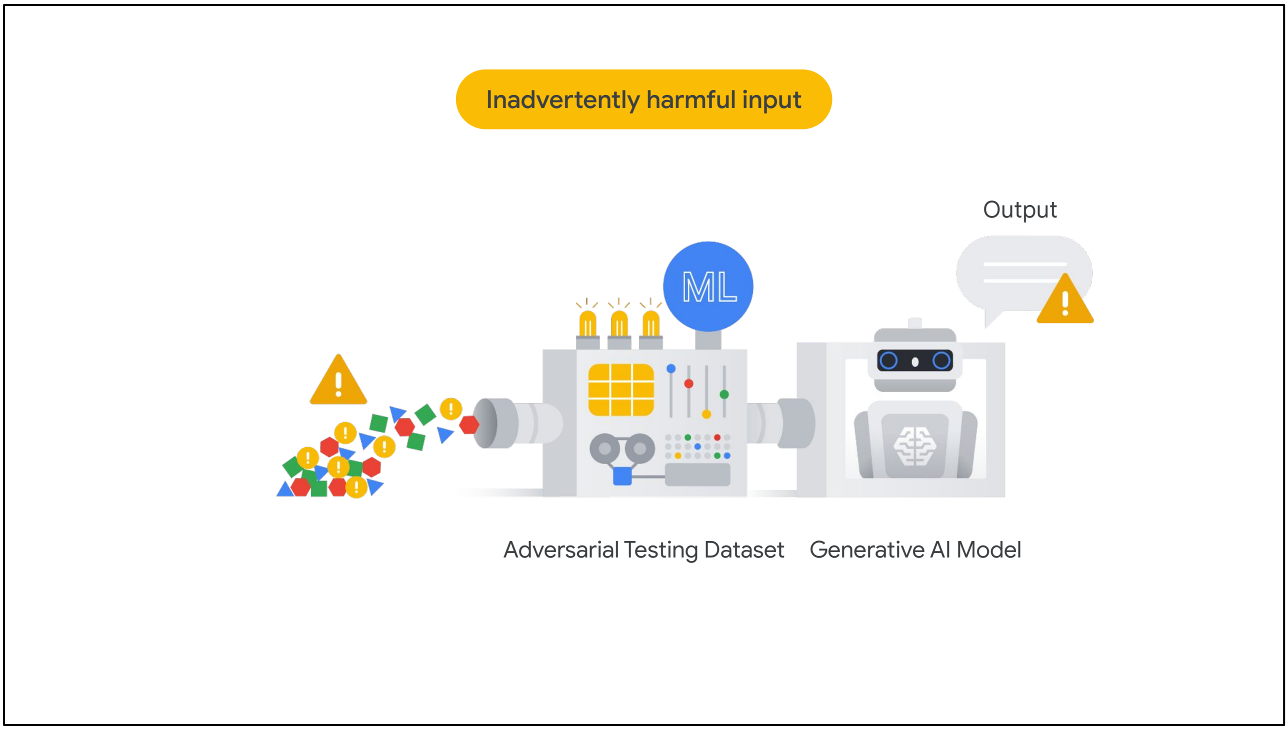


Inadvertently harmful input

Adversarial Testing Dataset

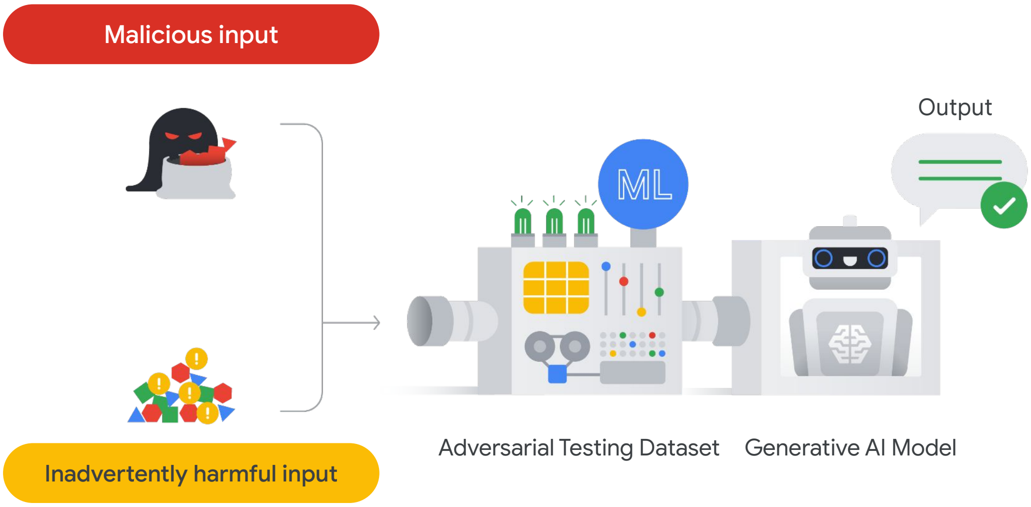
Generative AI Model

An input is considered inadvertently harmful when the input itself may be innocuous, but elicits a harmful output. For example, an input that asks a text generation model to describe a person of a particular ethnicity, and the model provides a racist output.



Inadvertently harmful input

This is also known as implicitly adversarial queries. They can contain culturally sensitive or contentious topics and might include information on demographics, health, finance, or religion.



Malicious input

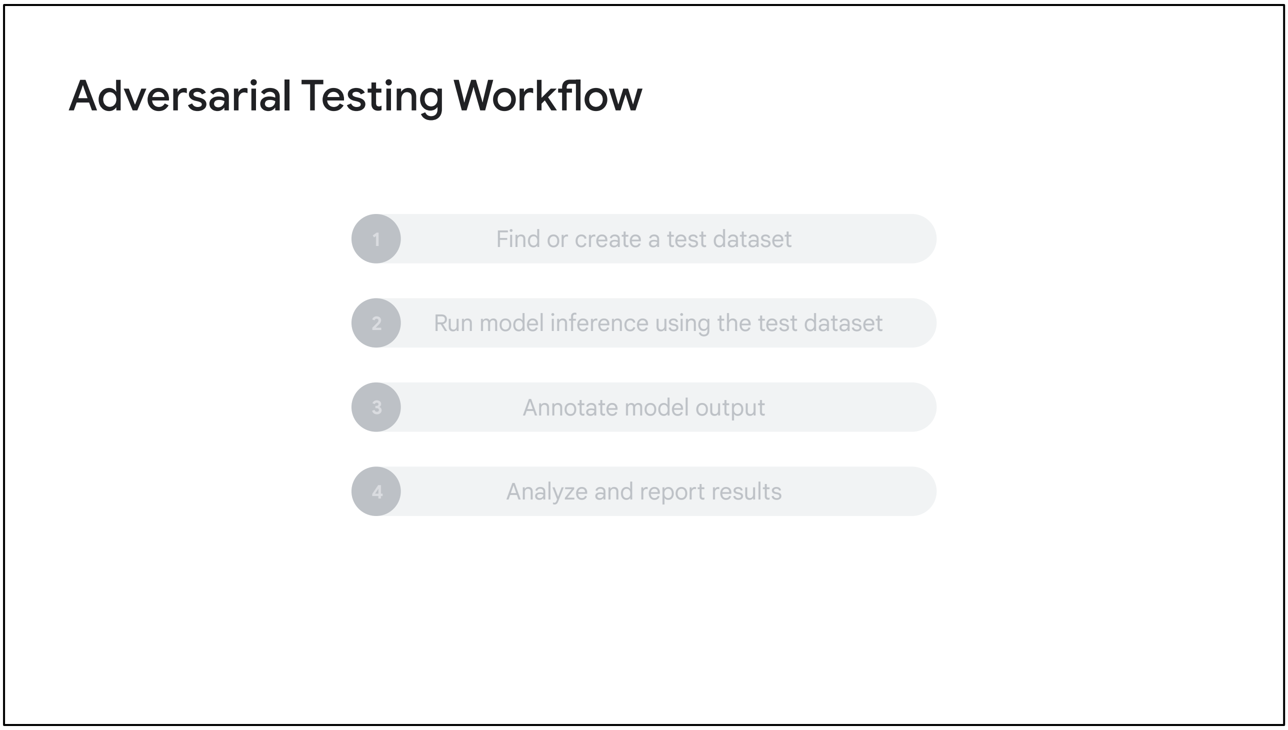
Inadvertently harmful input

Adversarial Testing Dataset

Generative AI Model

Output

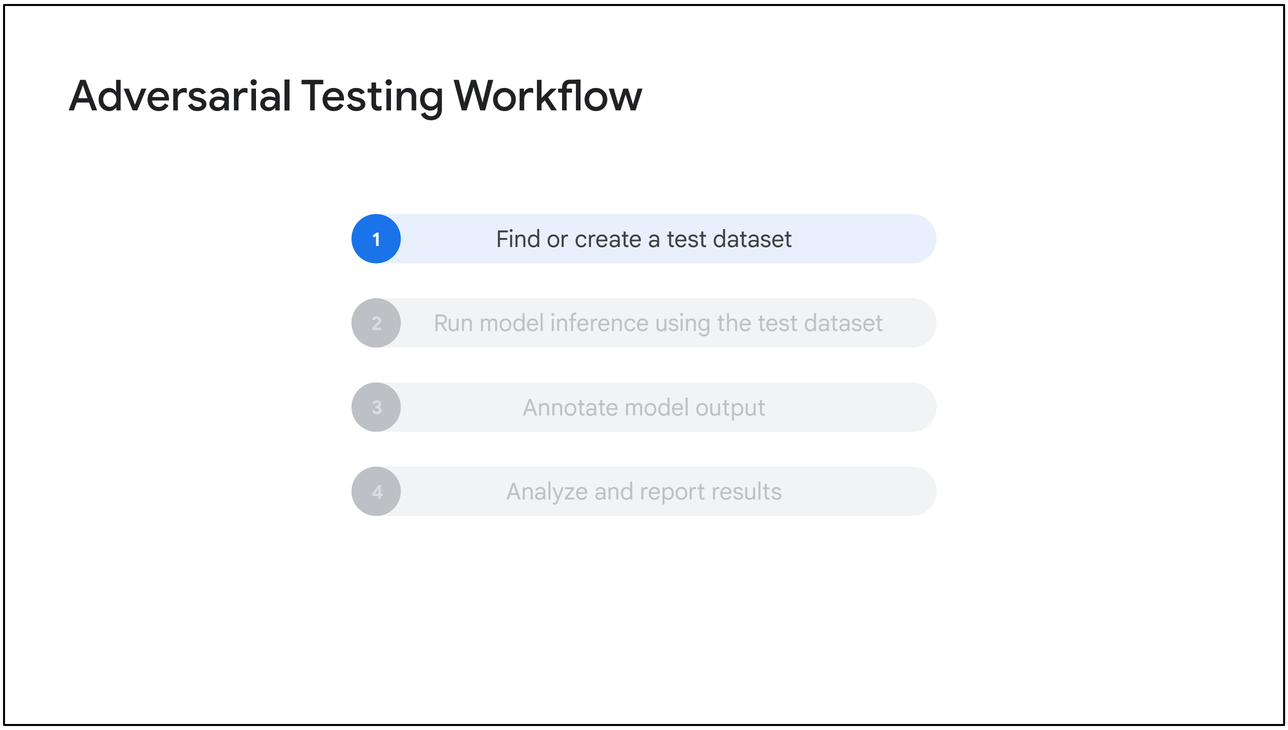
When malicious or harmful outputs occur, it is important to study the output to learn more about how the model performs when being exposed to such inputs.



Adversarial Testing Workflow

1. Find or create a test dataset
2. Run model inference using the test dataset
3. Annotate model output
4. Analyze and report results

This is accomplished through what is called an adversarial testing workflow. There are four steps in the workflow:



Adversarial Testing Workflow

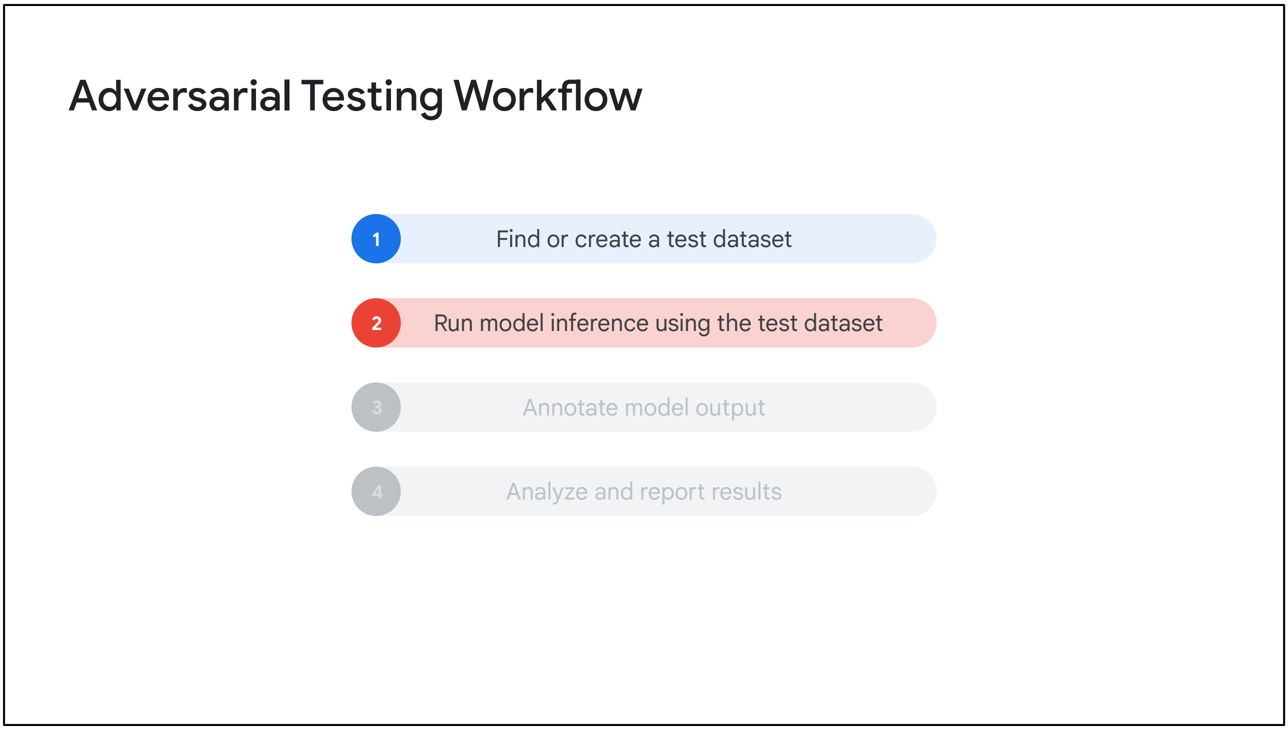
1 Find or create a test dataset

2 Run model inference using the test dataset

3 Annotate model output

4 Analyze and report results

- Find or create a test dataset.



Adversarial Testing Workflow

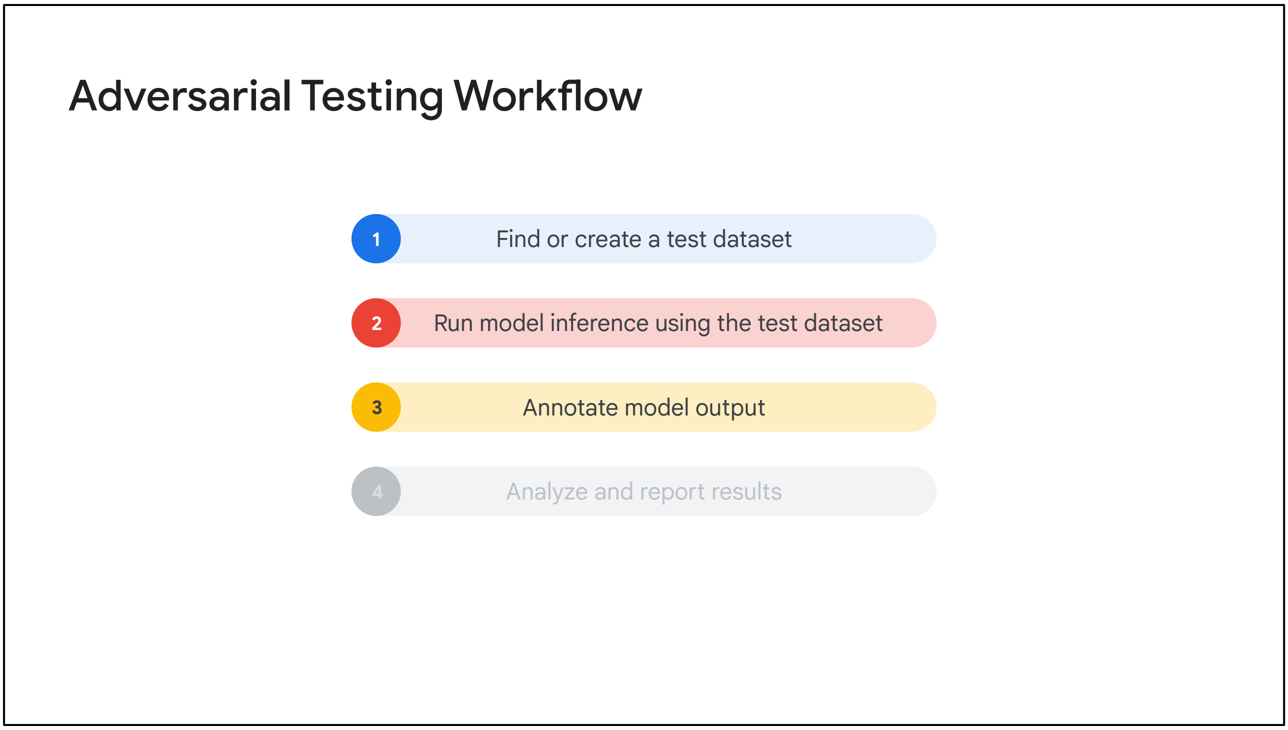
1 Find or create a test dataset

2 Run model inference using the test dataset

3 Annotate model output

4 Analyze and report results

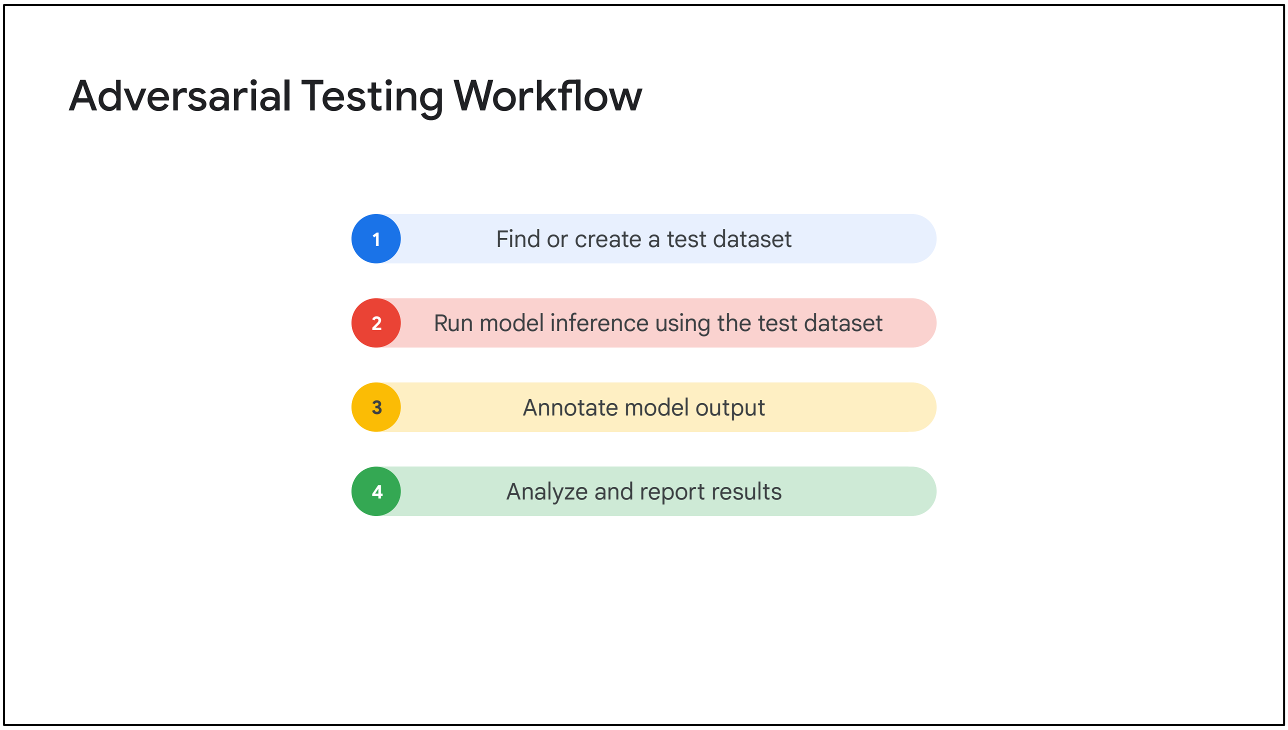
- Run model inference using the test dataset.



Adversarial Testing Workflow

1. Find or create a test dataset
2. Run model inference using the test dataset
3. Annotate model output
4. Analyze and report results

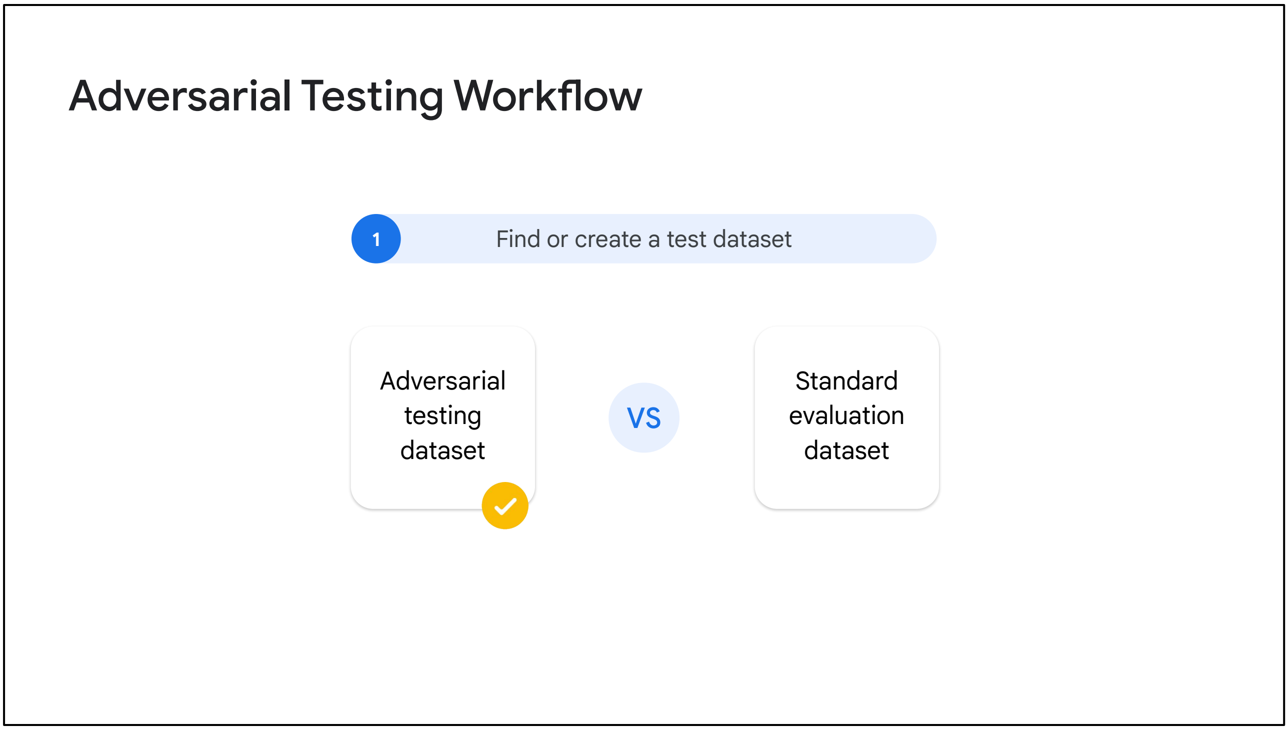
- Annotate the model output.



Adversarial Testing Workflow

1. Find or create a test dataset
2. Run model inference using the test dataset
3. Annotate model output
4. Analyze and report results

- And analyze and report results.



Adversarial Testing Workflow

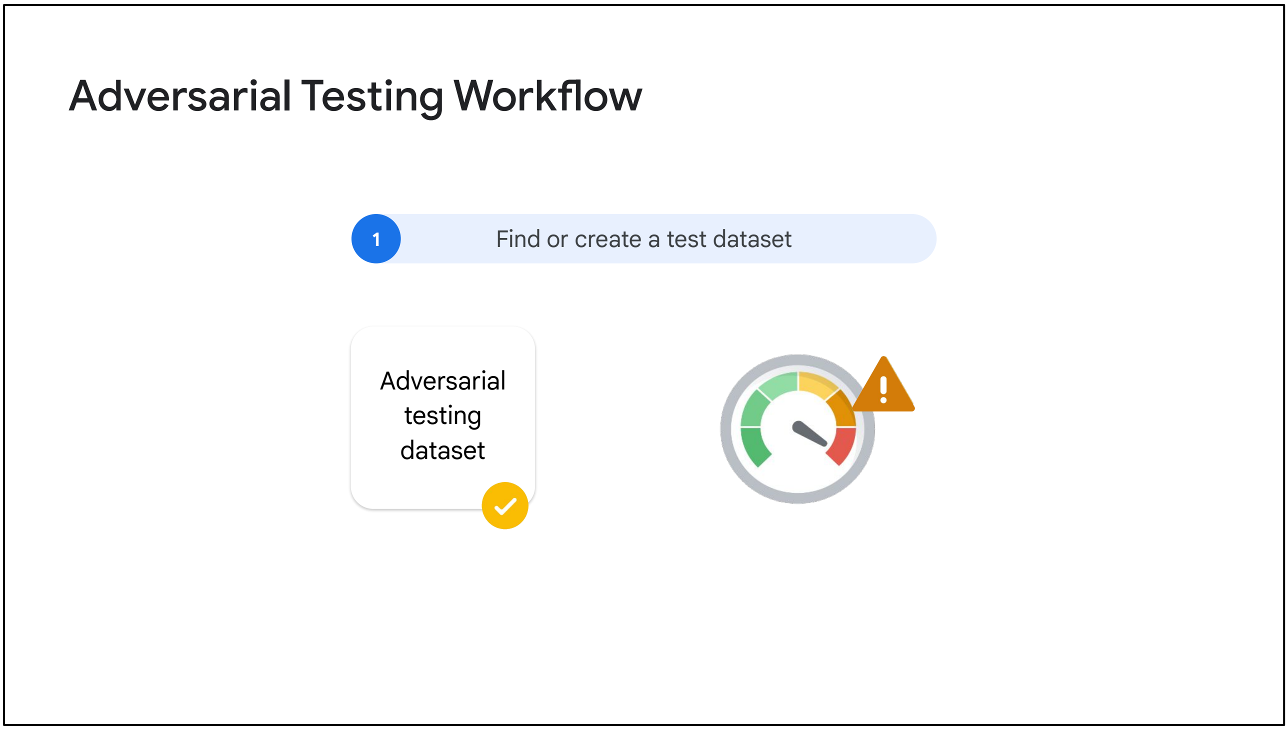
1 Find or create a test dataset

Adversarial testing dataset ✓

VS

Standard evaluation dataset

To conduct effective adversarial testing, find or create a test dataset. You should not use standard evaluation datasets as these are typically designed to accurately reflect the distribution of data that the model will encounter in the product.

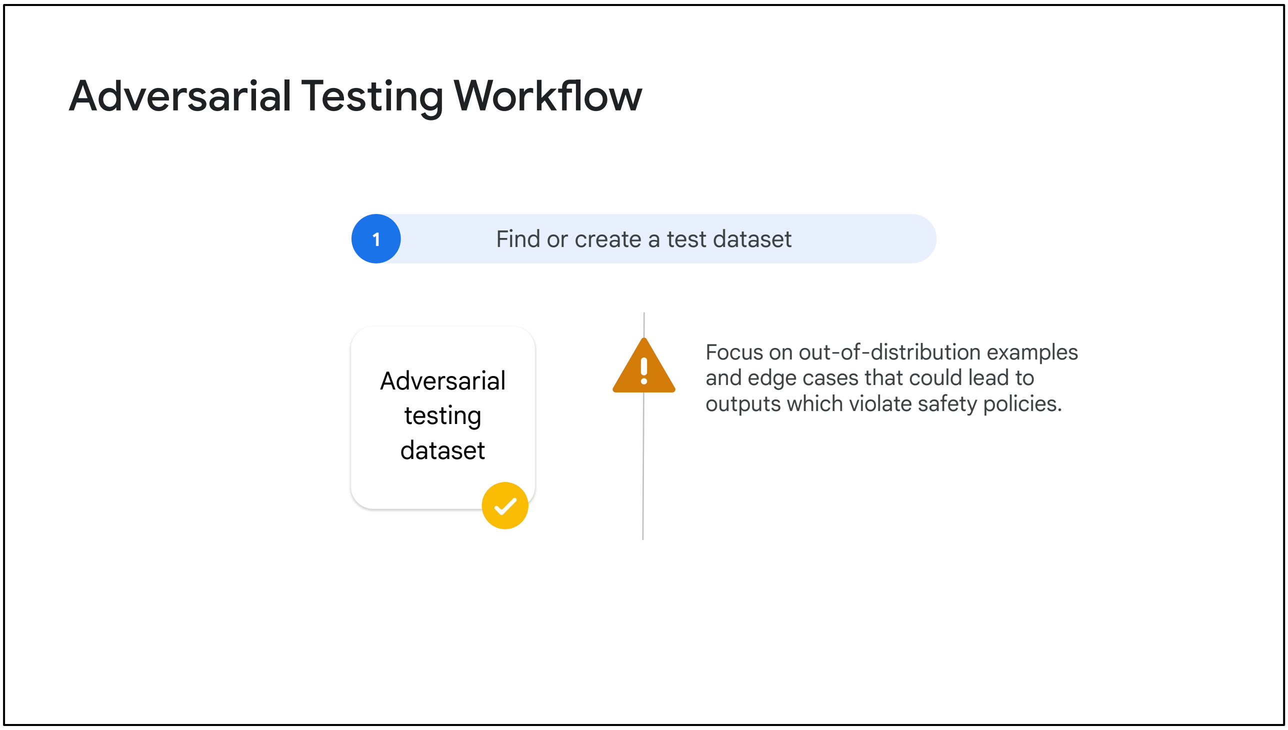


Adversarial Testing Workflow

1 Find or create a test dataset

Adversarial testing dataset

For adversarial tests, the key to creating or finding a test dataset is to design a test that pushes the model's limits.



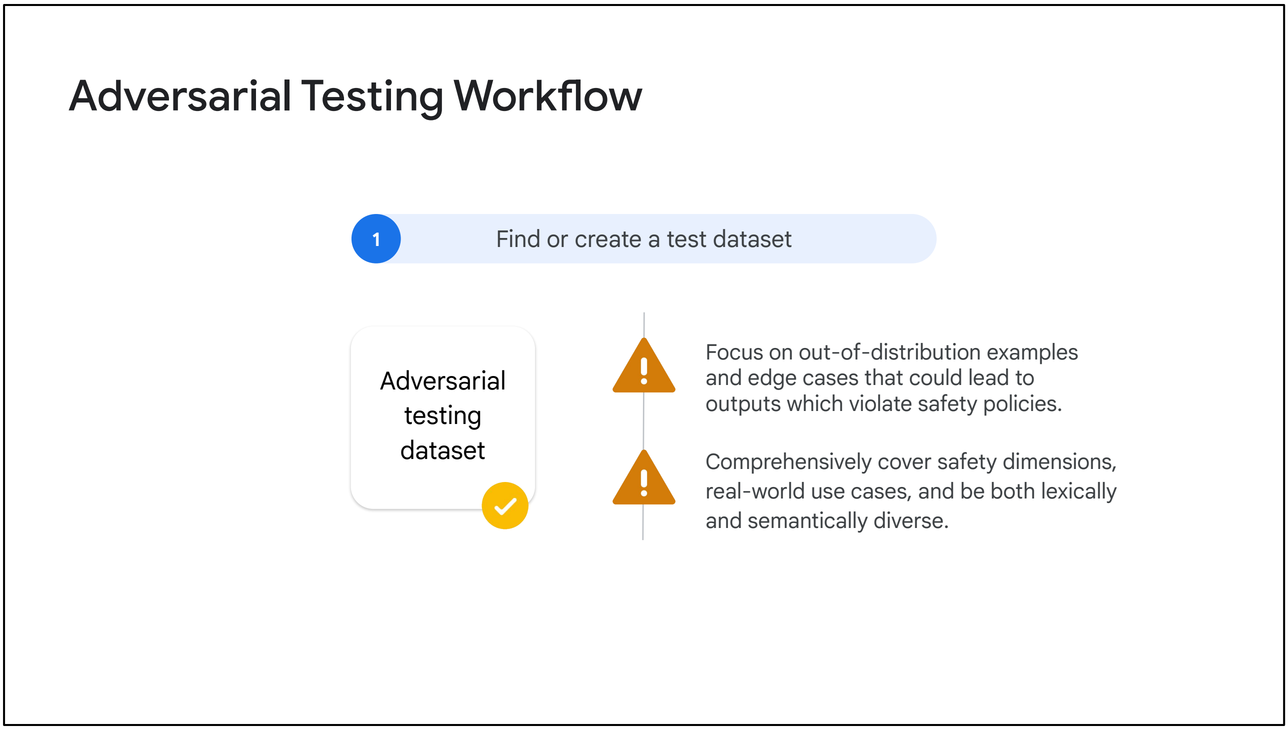
Adversarial Testing Workflow

1 Find or create a test dataset

Adversarial testing dataset ✓

Focus on out-of-distribution examples and edge cases that could lead to outputs which violate safety policies.

To accomplish this, you should focus on out-of-distribution examples and edge cases that could lead to outputs which violate safety policies.



Adversarial Testing Workflow

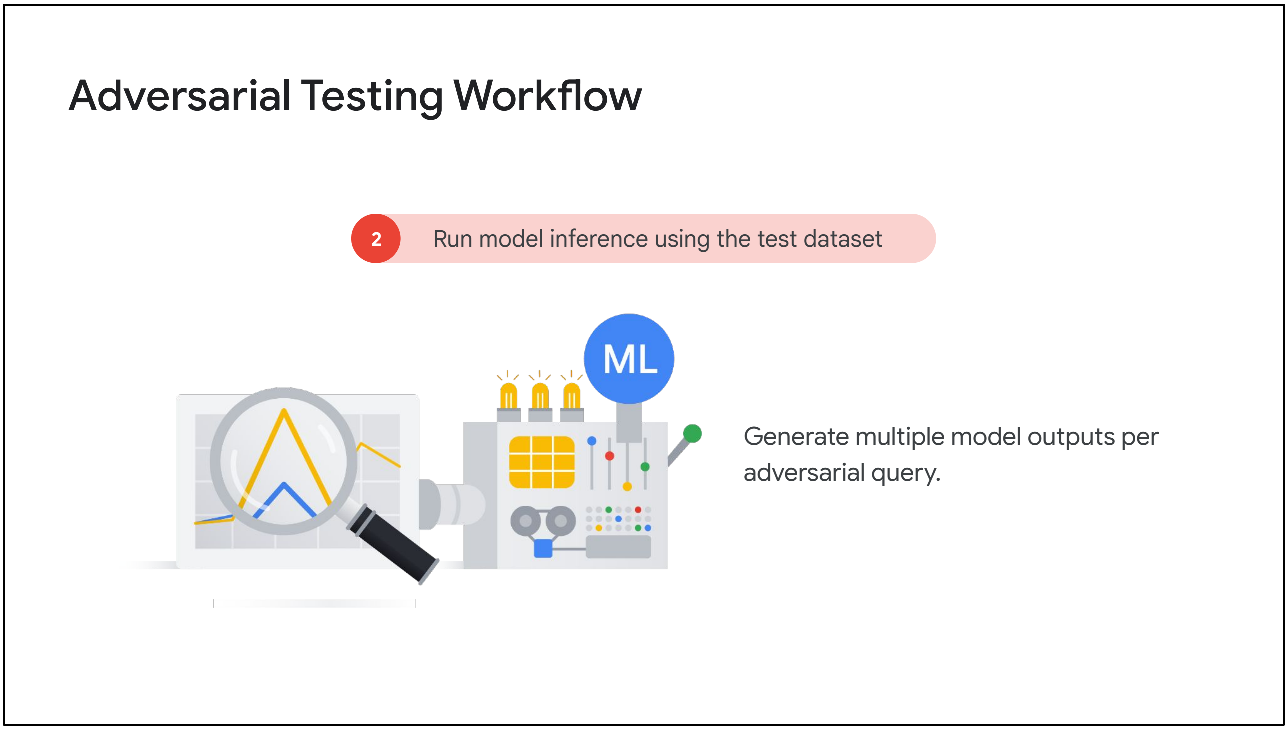
1 Find or create a test dataset

Adversarial testing dataset ✓

⚠ Focus on out-of-distribution examples and edge cases that could lead to outputs which violate safety policies.

⚠ Comprehensively cover safety dimensions, real-world use cases, and be both lexically and semantically diverse.

The dataset should comprehensively cover safety dimensions, real-world use cases, and be both lexically and semantically diverse. Lexical diversity focuses on the variety of unique words used within a piece of text or speech. Semantic diversity focuses on a range of meanings and ideas expressed within a text or speech.

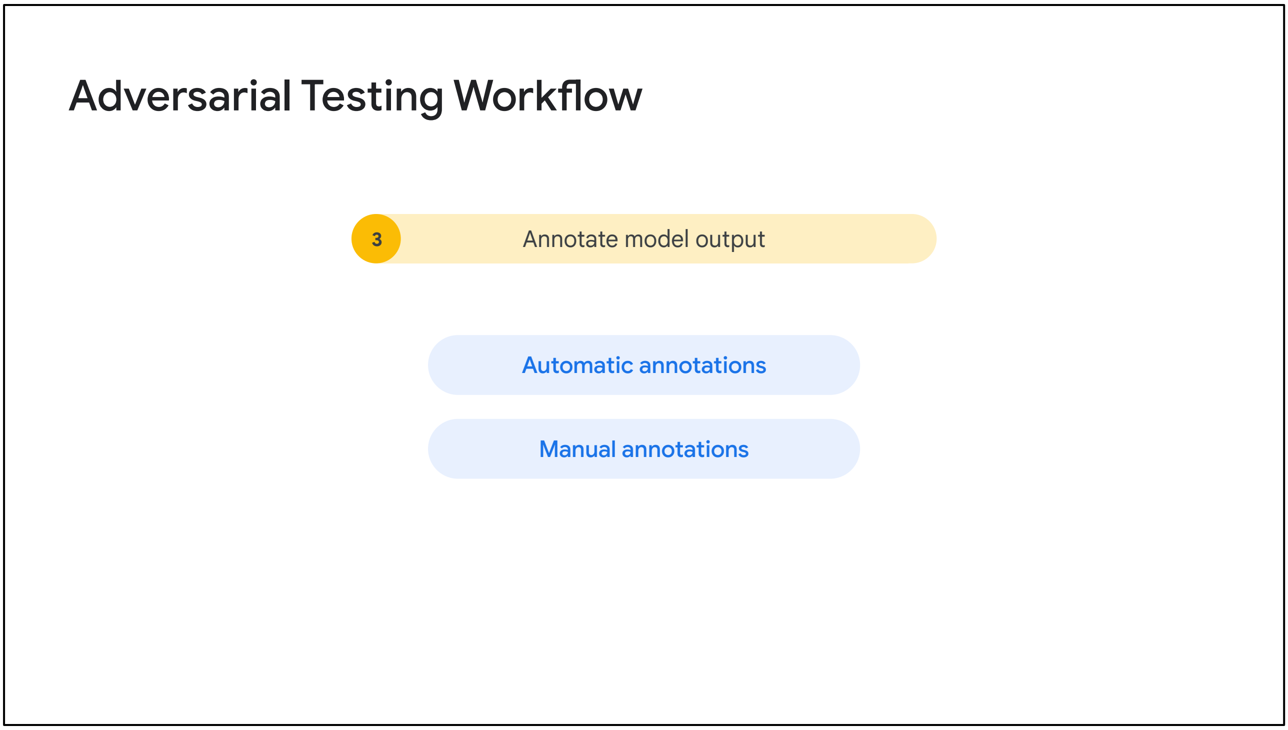


Adversarial Testing Workflow

2 Run model inference using the test dataset

Generate multiple model outputs per adversarial query.

Next, run a model inference using the test dataset. The goal here is to generate model outputs based on the test dataset. In certain cases, generating multiple model outputs per adversarial query can help.



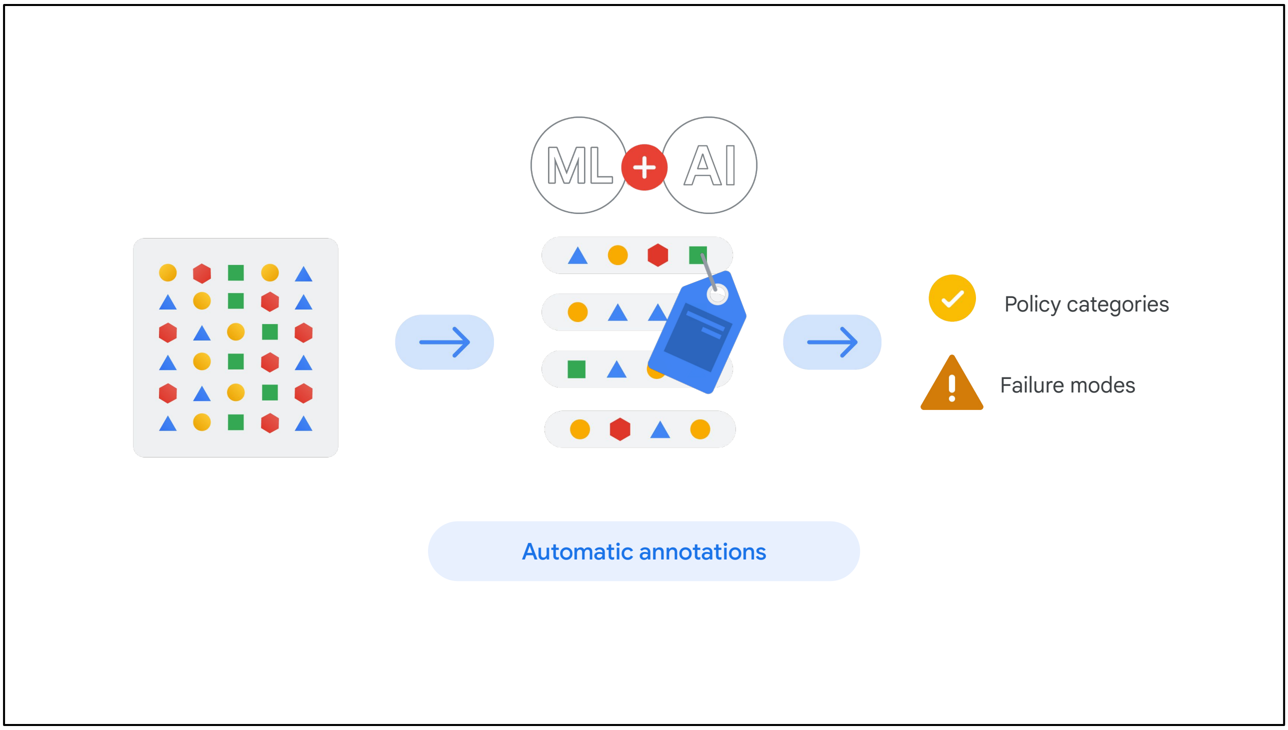
Adversarial Testing Workflow

3 Annotate model output

Automatic annotations

Manual annotations

Third, annotate the model's outputs to identify any policy violations.
There are two methods for doing this: automatically and manually.



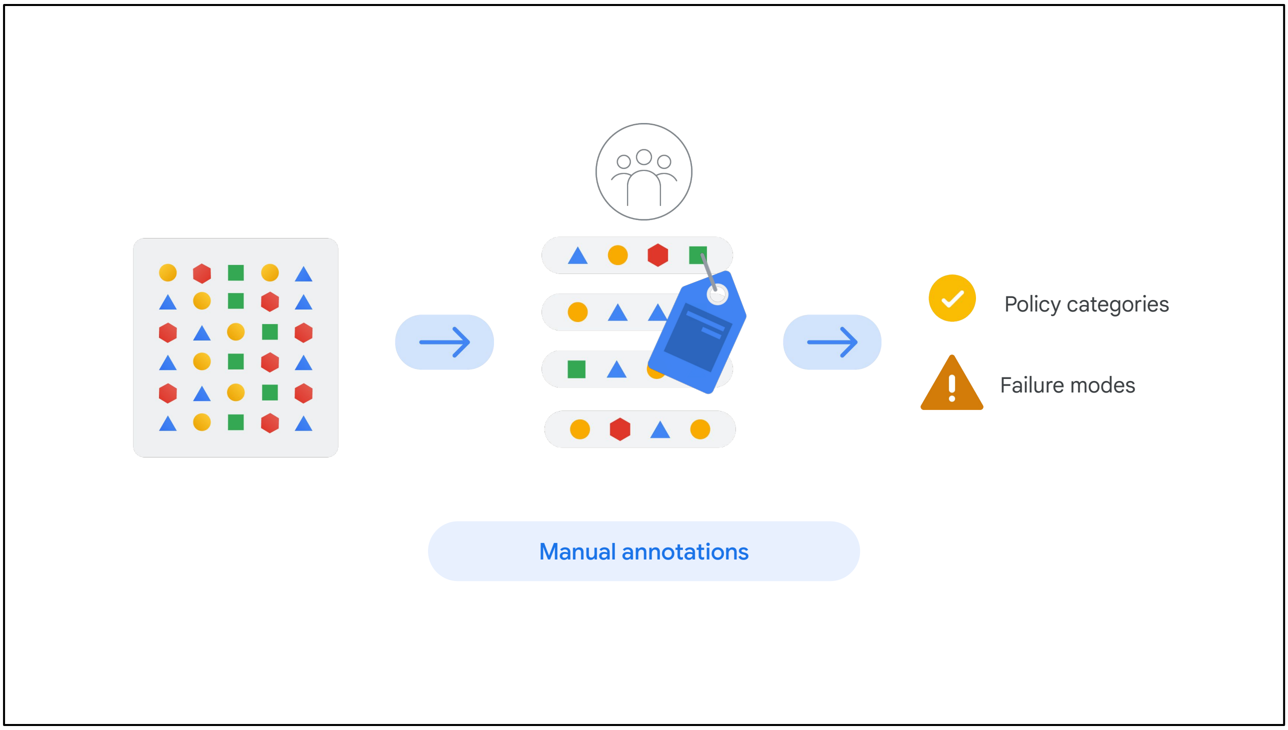
ML + AI

Policy categories

Failure modes

Automatic annotations

Automatic annotations use machine learning and other AI techniques to automatically label or tag data that aligns with specific policy categories and failure modes.

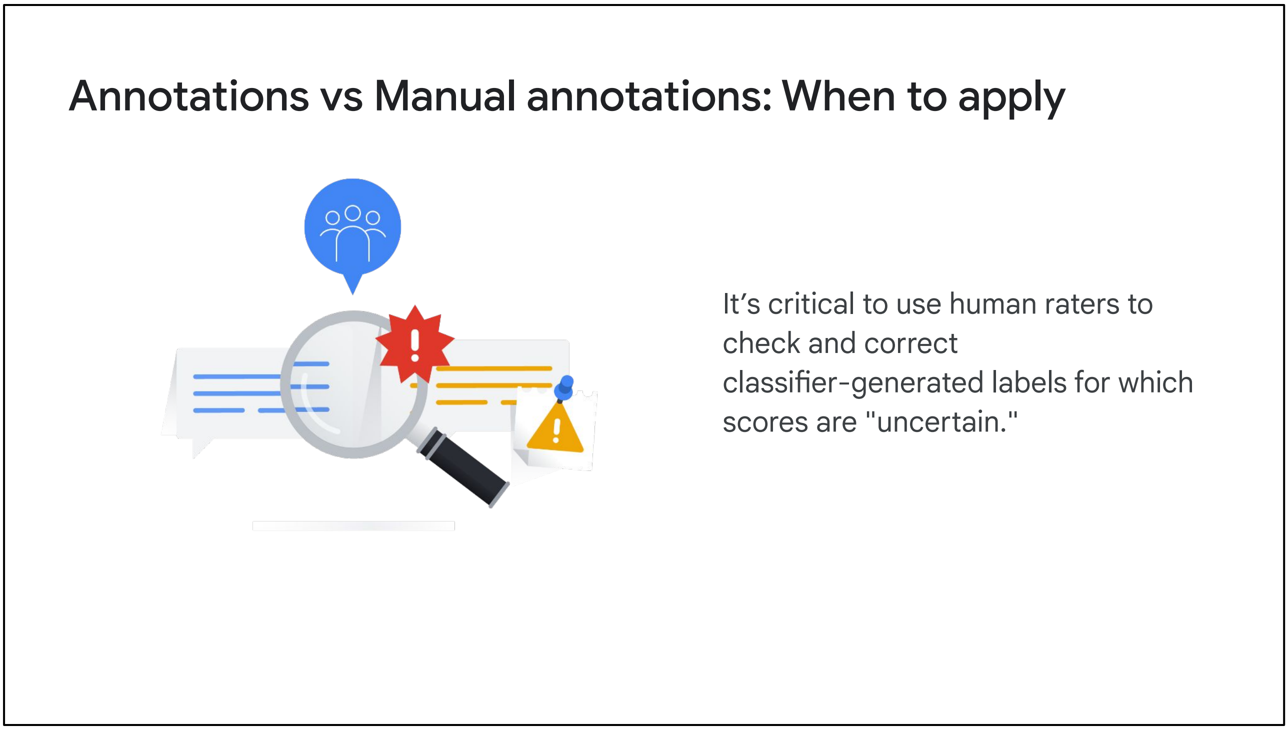


Policy categories

Failure modes

Manual annotations

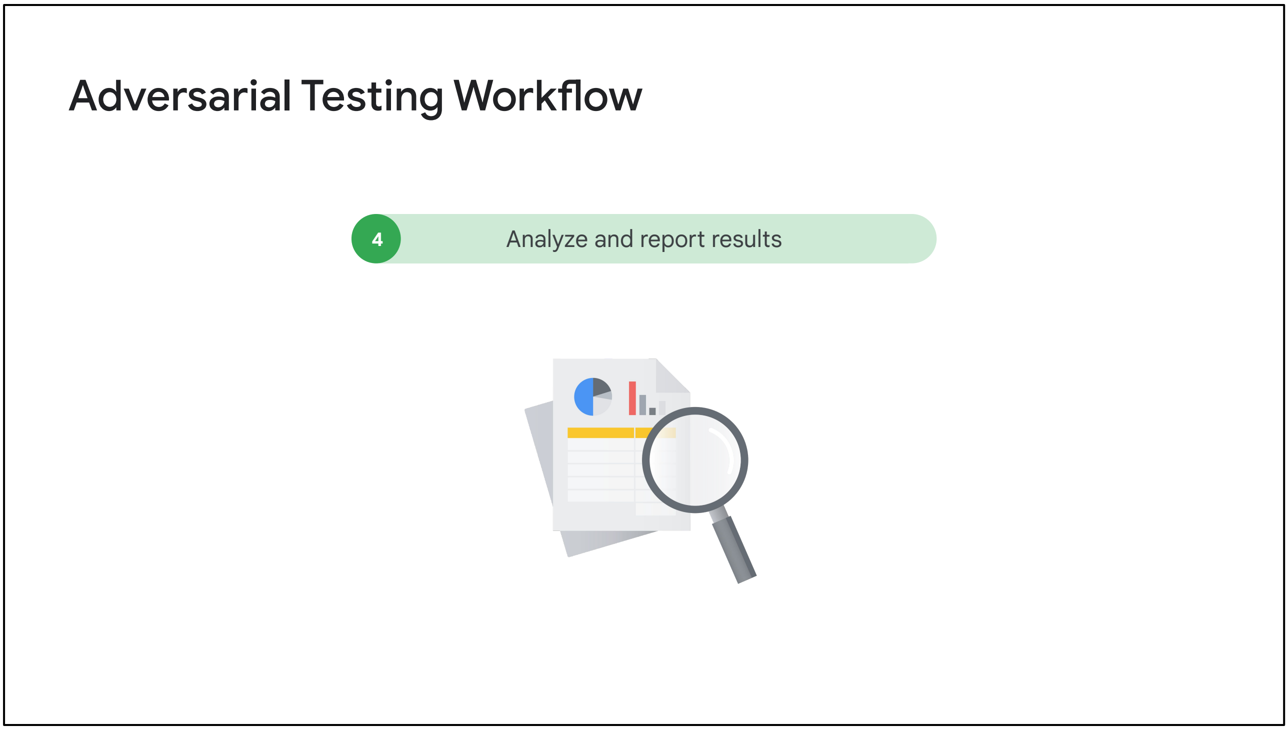
Manual annotations involve human raters, either internal or external, who label and tag the data. This is accomplished using their own specific instructions and platforms designed for this task.
When should you use automatic annotations vs manual?



Annotations vs Manual annotations: When to apply

It’s critical to use human raters to check and correct classifier-generated labels for which scores are "uncertain."

A good example is the annotation of data that contains hate speech. Sometimes, automatic annotation accuracy may be low for signals that try to detect constructs that are not strictly defined, as in the case of hate speech. In this situation, it is critical to use human raters to check and correct classifier-generated labels for which scores are "uncertain."

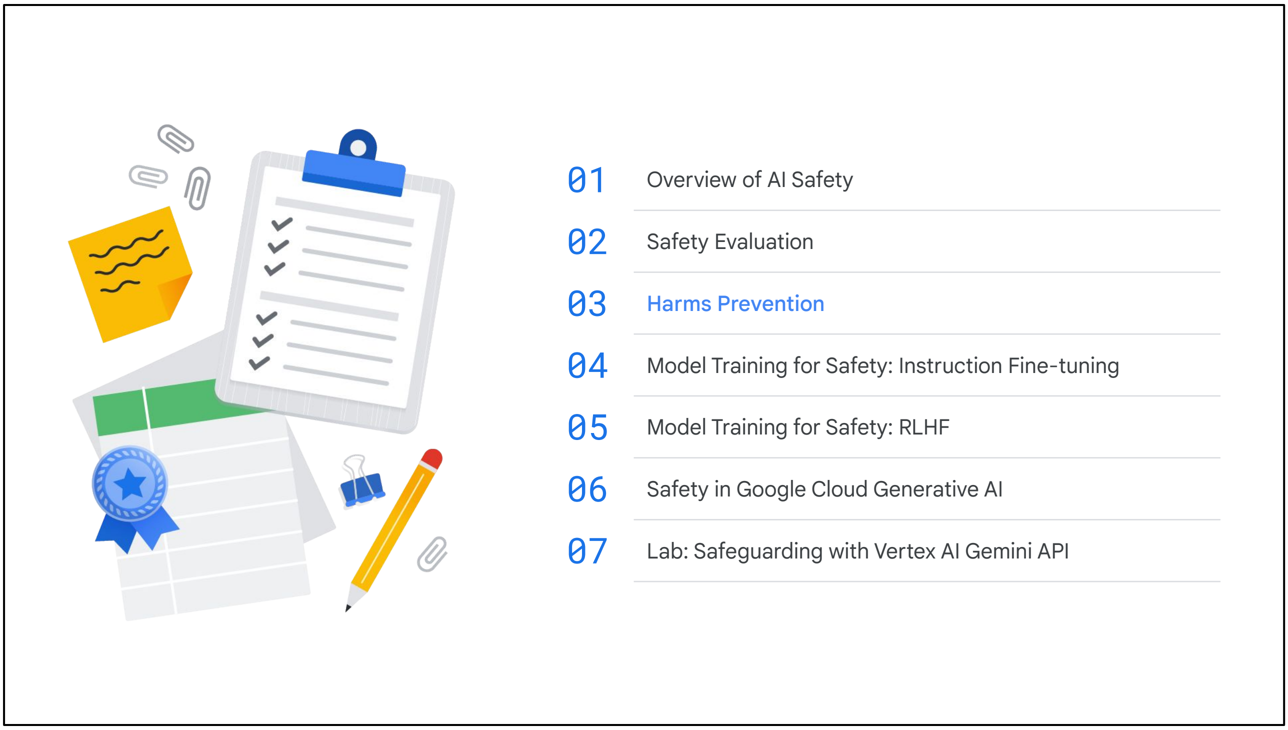


Adversarial Testing Workflow

4 Analyze and report results

The final step is to summarize test results in a report. You can pick any type of chart of your choice to effectively communicate results in a report to your stakeholders and decision makers.

These results can guide model improvements and inform model safeguards, such as the creation or improvement of filters.



01 Overview of AI Safety

02 Safety Evaluation

03 **Harms Prevention**

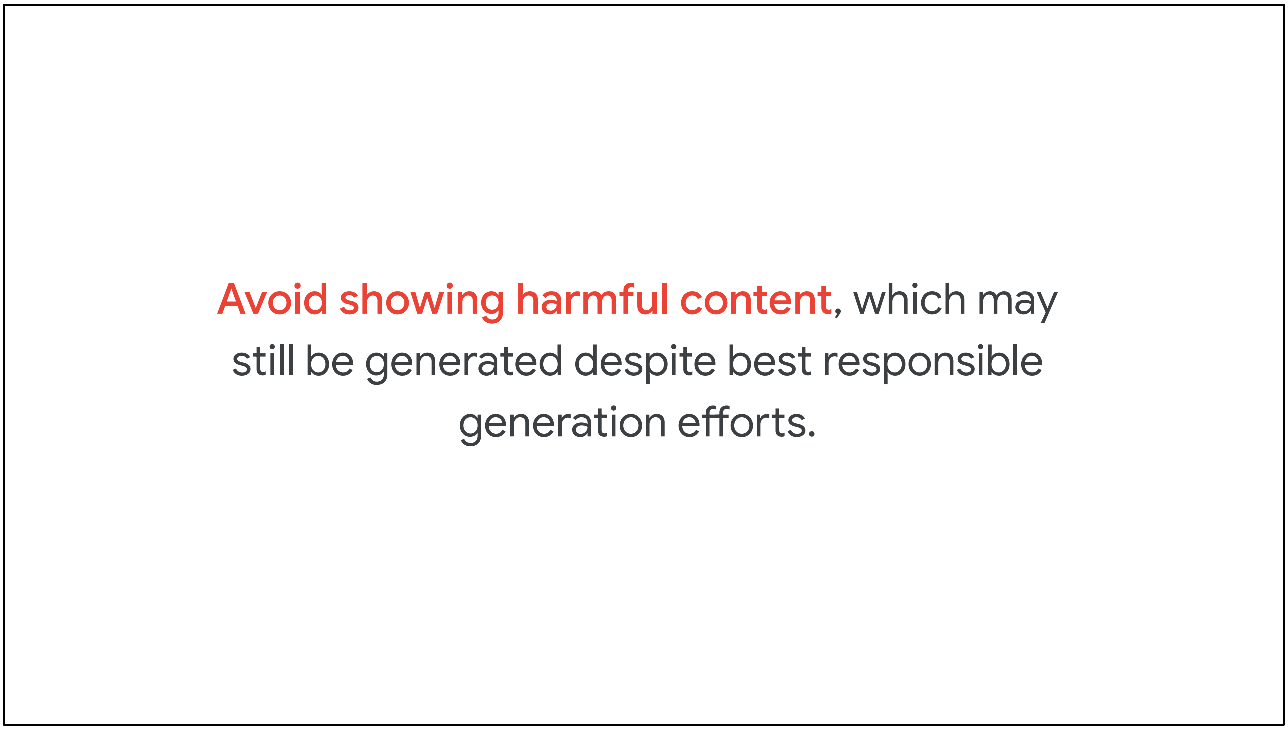
04 Model Training for Safety: Instruction Fine-tuning

05 Model Training for Safety: RLHF

06 Safety in Google Cloud Generative AI

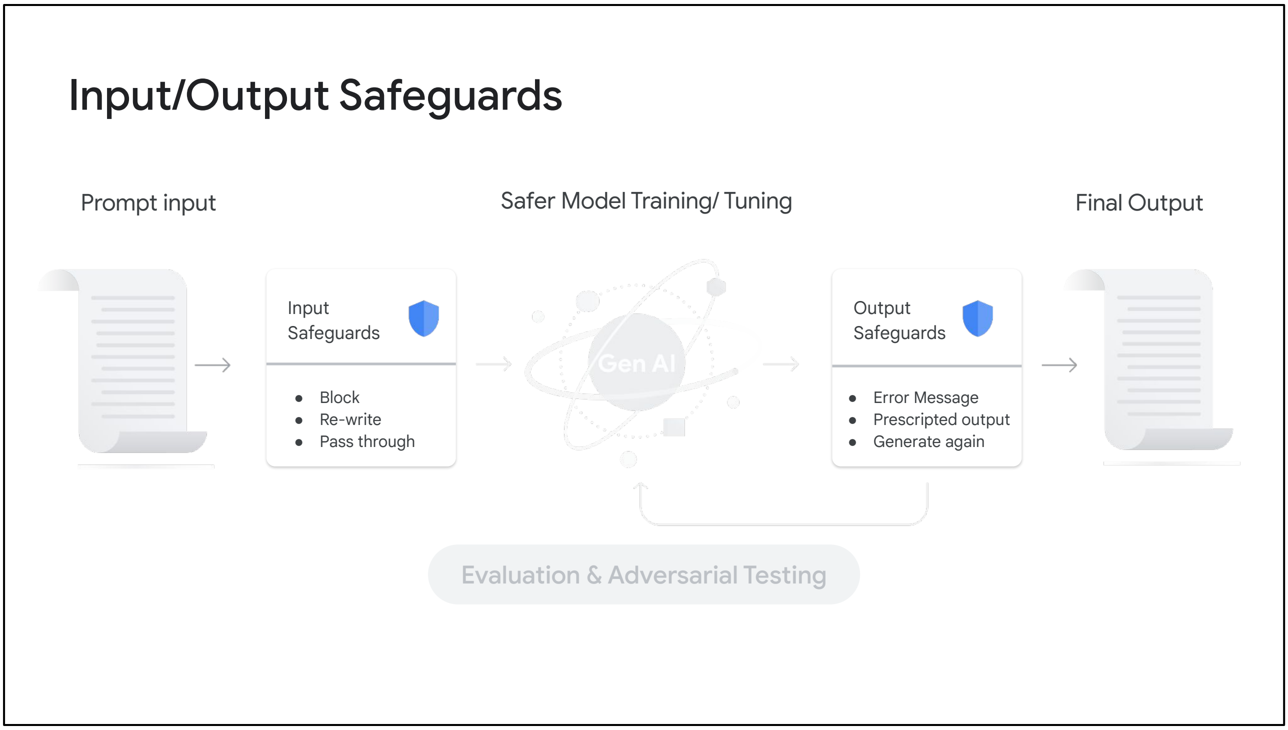
07 Lab: Safeguarding with Vertex AI Gemini API

Let’s learn some methods to address harm prevention.



Avoid showing harmful content, which may still be generated despite best responsible generation efforts.

Harm prevention means to avoid showing harmful content, which may still be generated despite best responsible generation efforts.



Input/Output Safeguards

Prompt input

Safer Model Training/ Tuning

Final Output

Input Safeguards

- Block
- Re-write
- Pass through

Gen AI

Output Safeguards

- Error Message
- Prescripted output
- Generate again

Evaluation & Adversarial Testing

Input and output safeguards are essential aspects of AI safety that play crucial roles in preventing harm.

They act as protective measures to guard what goes into an AI system as an input and what is produced by the AI system as an output to ensure that AI behavior aligns with safety standards and ethical principles.

Safety classifiers

Safety classifiers

Machine learning based classification systems that classify whether an input text is safe or not is called a safety classifier. While it is technically possible to build an original safety classifier by yourself, practically, it is a challenging task since a safety classifier needs to be trained carefully with specially consolidated datasets, and with special care and attention towards fairness.
Fortunately, there are a few safety classifiers available in the AI industry to help people prevent outputting harm from a model.

Safety classifiers

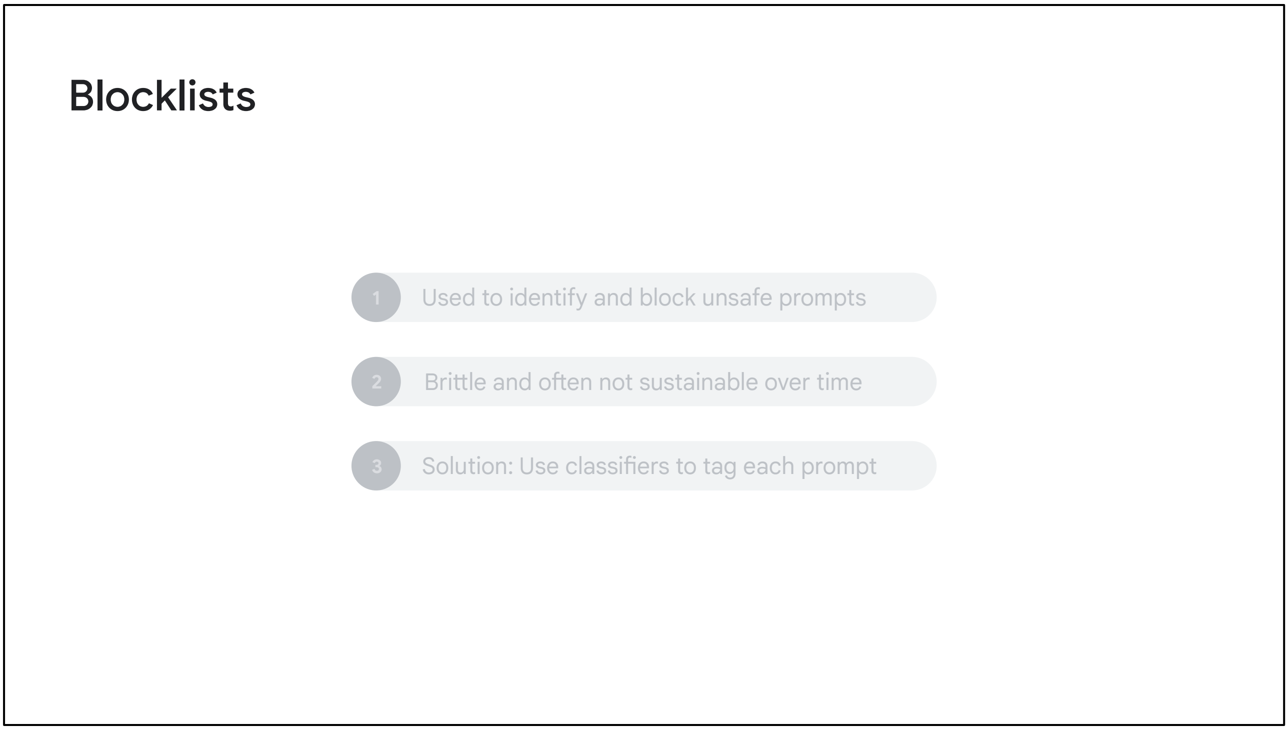
Safety classifiers

In 2017, Google's Jigsaw team released Perspective API, one of the first safety classifiers in the world. Perspective API is designed to help flag harmful speech that pushes people out of online spaces, pushes them towards violence, or even worse. By the time 2021 came around, Perspective API reached around 500 million requests daily due to high usage in many online platforms.
Currently, Perspective API is widely used for input safeguards to guard input for the AI, and for output safeguards to monitor output.

Safety classifiers

Safety classifiers

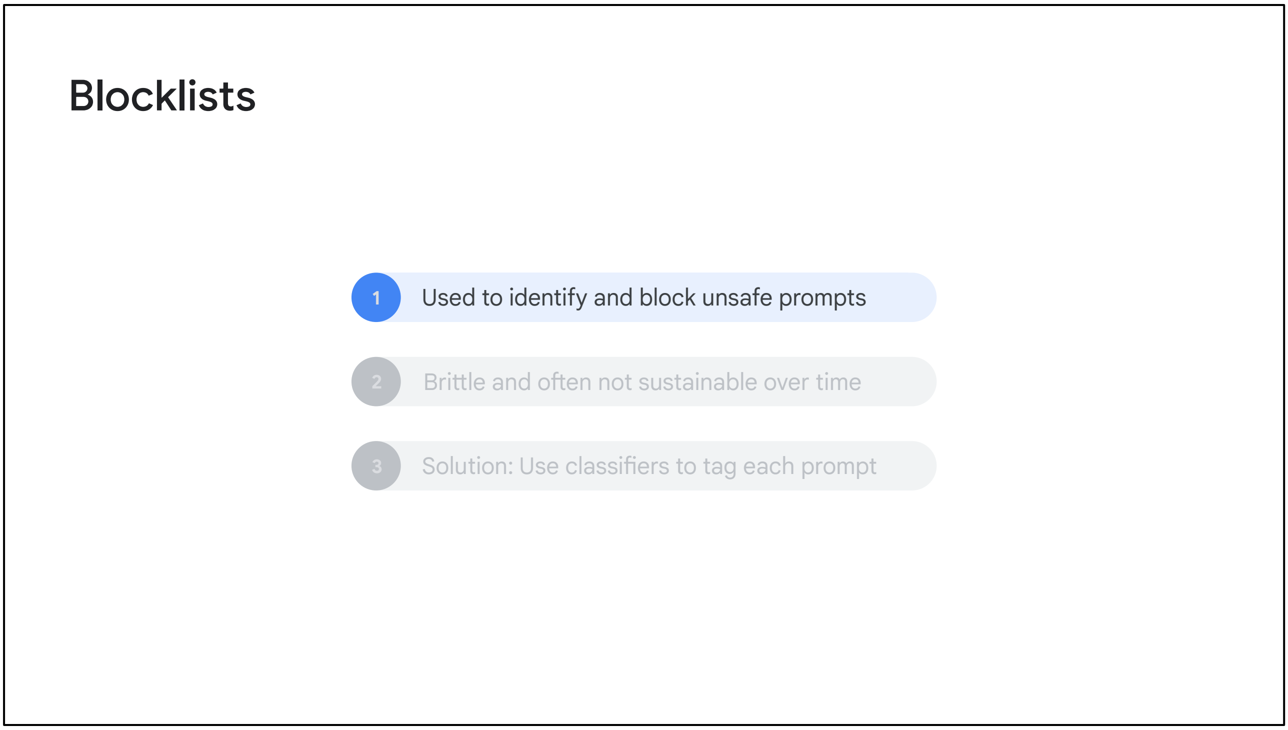
While Perspective API was one of the first safety classifiers, safety classifiers now exist from other organizations, including Moderation API from OpenAI and Llama Guard from Meta.



Blocklists

1. Used to identify and block unsafe prompts
2. Brittle and often not sustainable over time
3. Solution: Use classifiers to tag each prompt

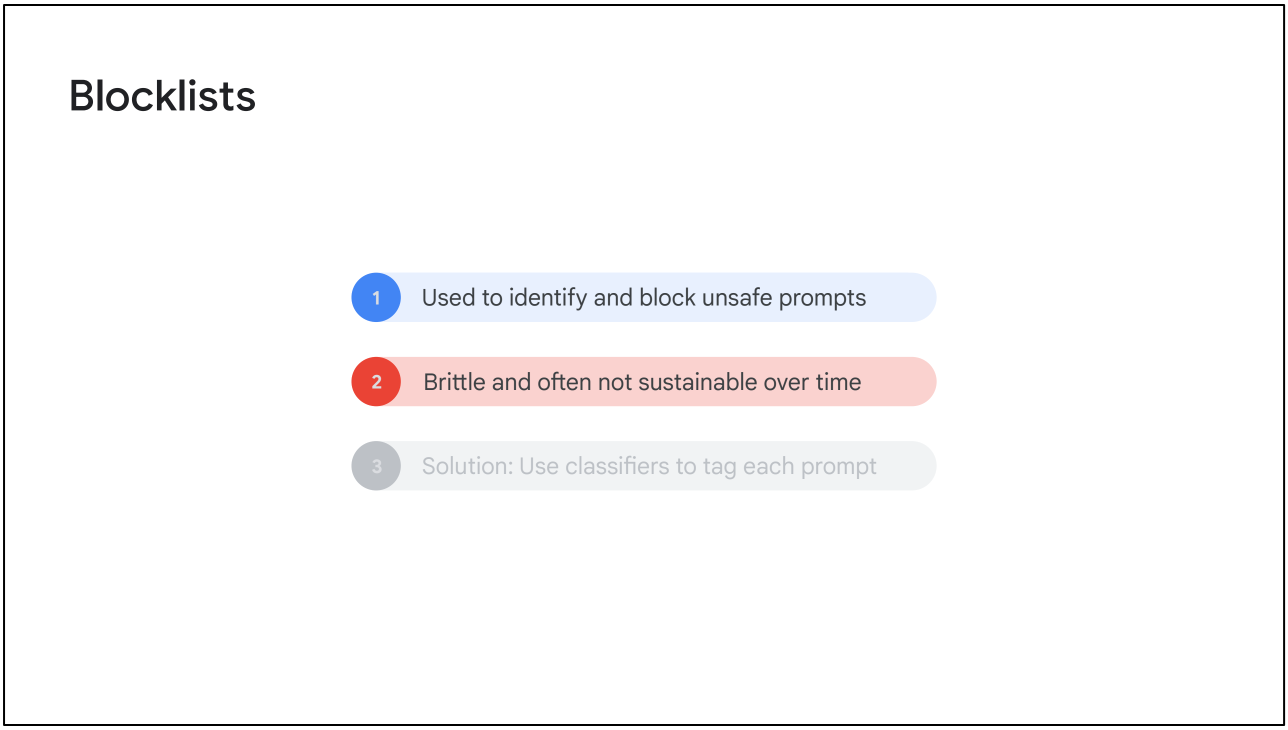
Many people think of “blocklists” when talking about input safeguards.



Blocklists

1. Used to identify and block unsafe prompts
2. Brittle and often not sustainable over time
3. Solution: Use classifiers to tag each prompt

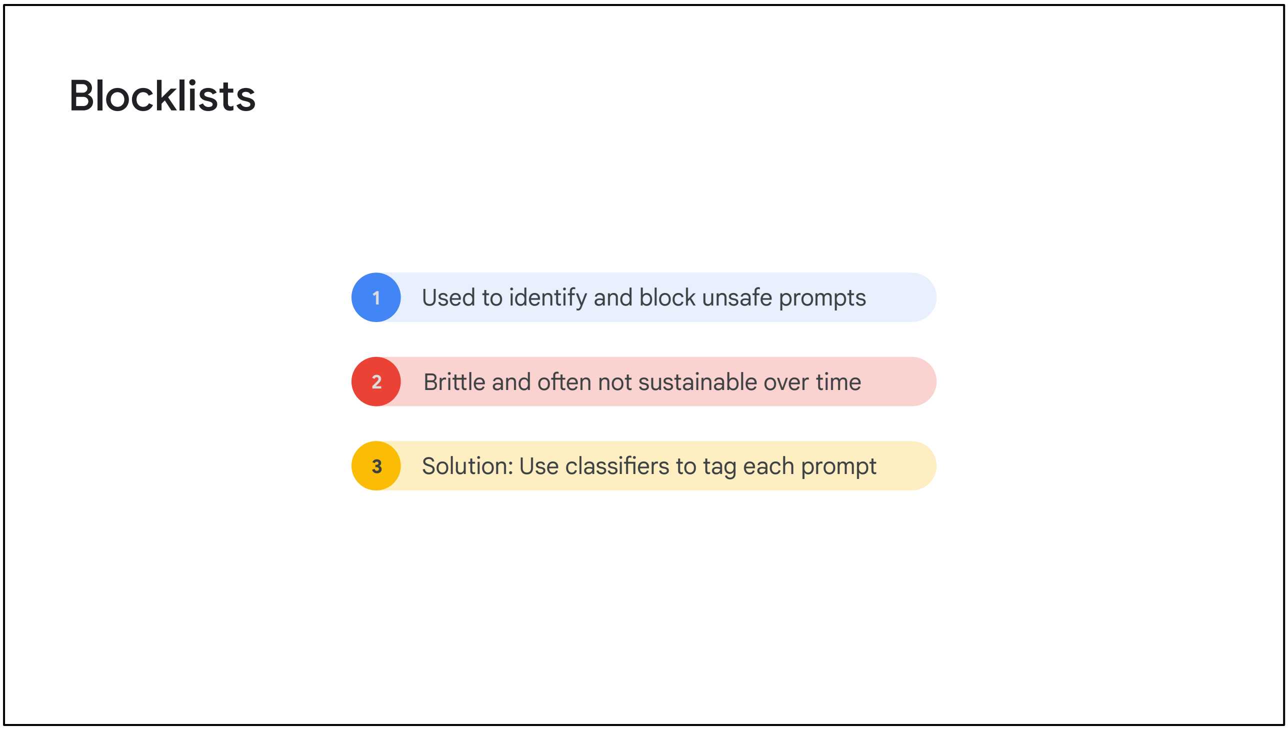
In very simple situations, blocklists can be used to identify and block unsafe prompts.



Blocklists

1. Used to identify and block unsafe prompts
2. Brittle and often not sustainable over time
3. Solution: Use classifiers to tag each prompt

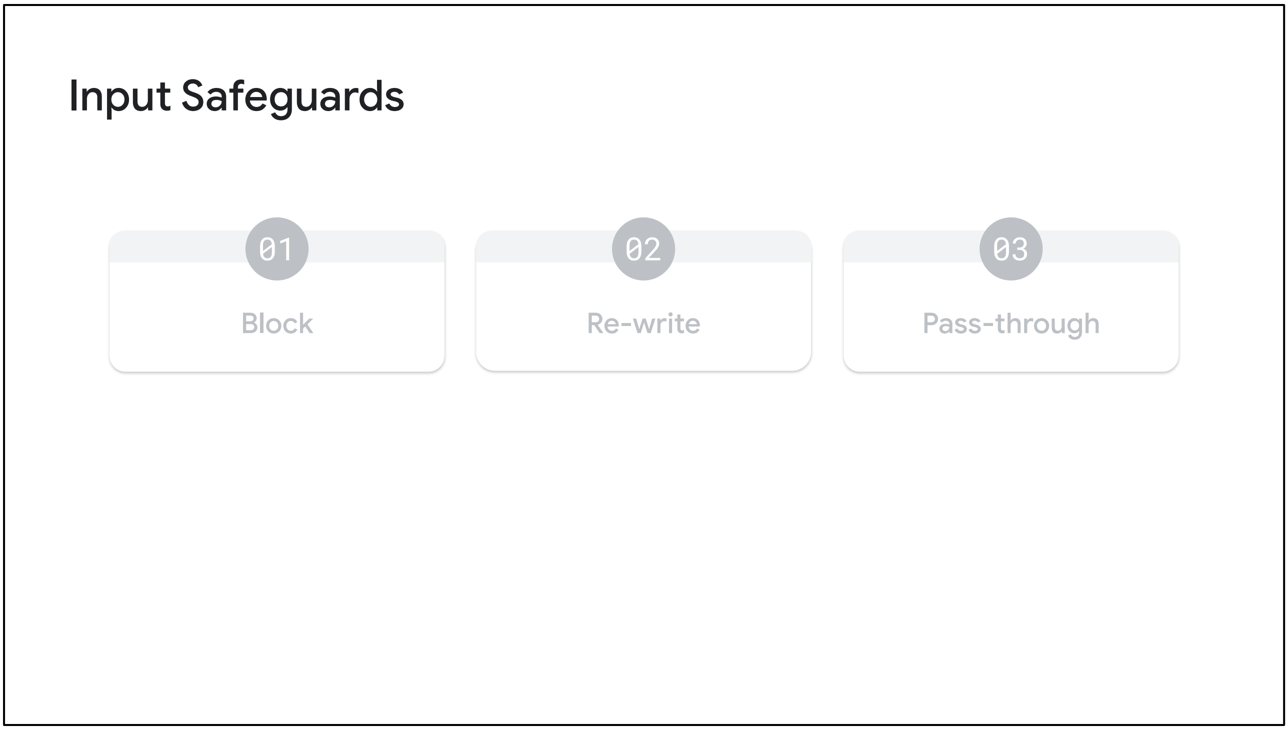
However, this method is brittle and often not sustainable over time.



Blocklists

1. Used to identify and block unsafe prompts
2. Brittle and often not sustainable over time
3. Solution: Use classifiers to tag each prompt

A better long-term solution is using classifiers to tag each prompt with potential harms or adversarial signals. Then, you can apply different strategies on how to handle the request based on the type of harm detected.



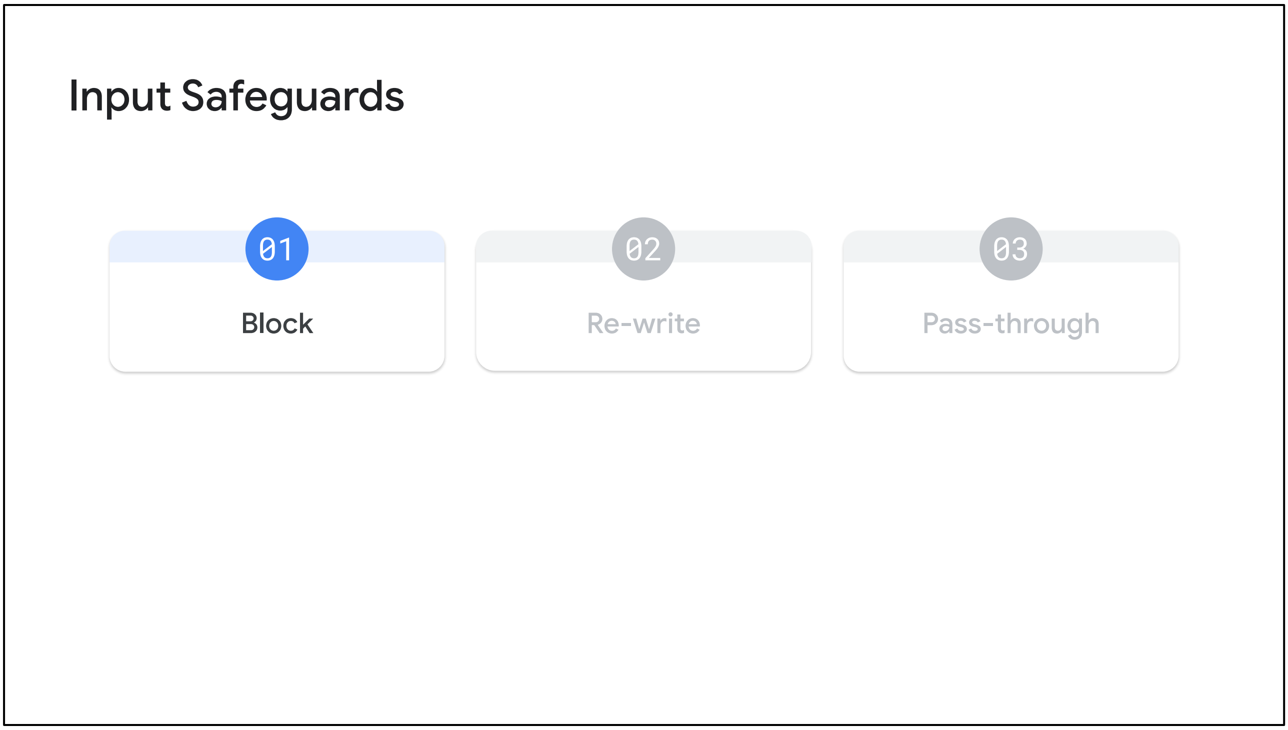
Input Safeguards

01 Block

02 Re-write

03 Pass-through

Strategies include



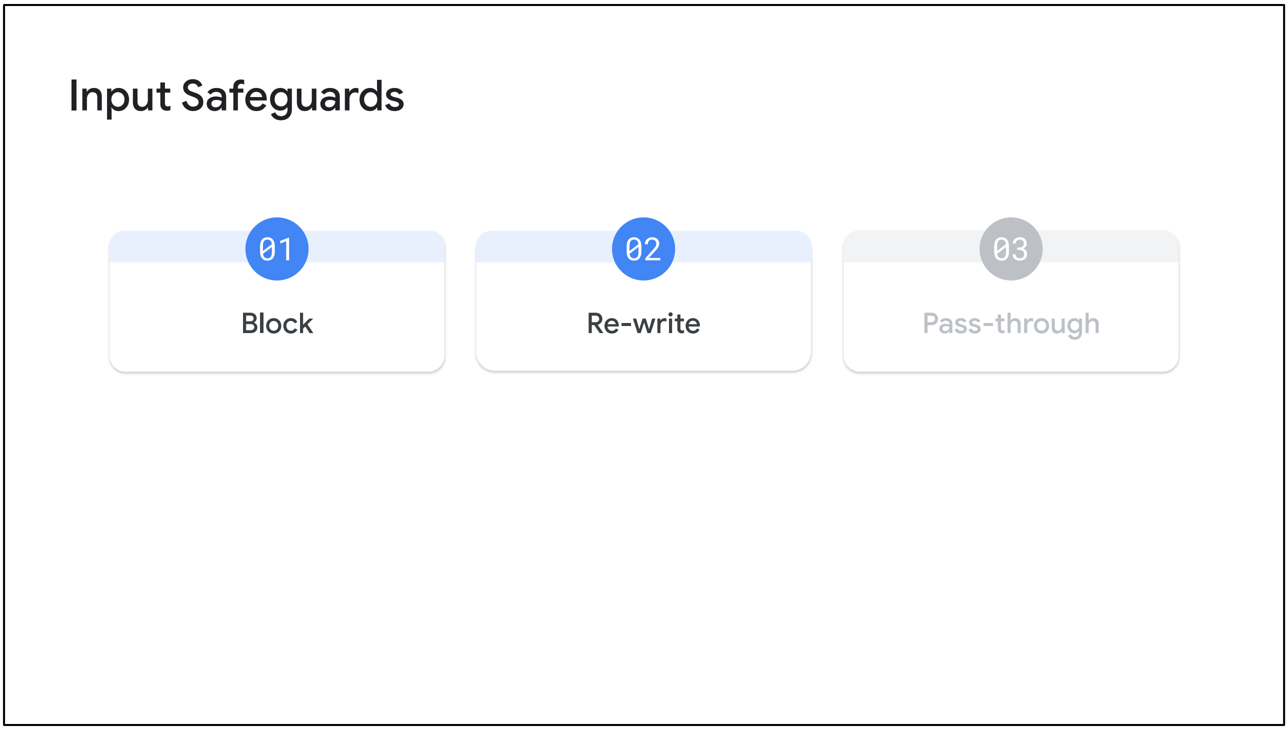
Input Safeguards

01 Block

02 Re-write

03 Pass-through

to block



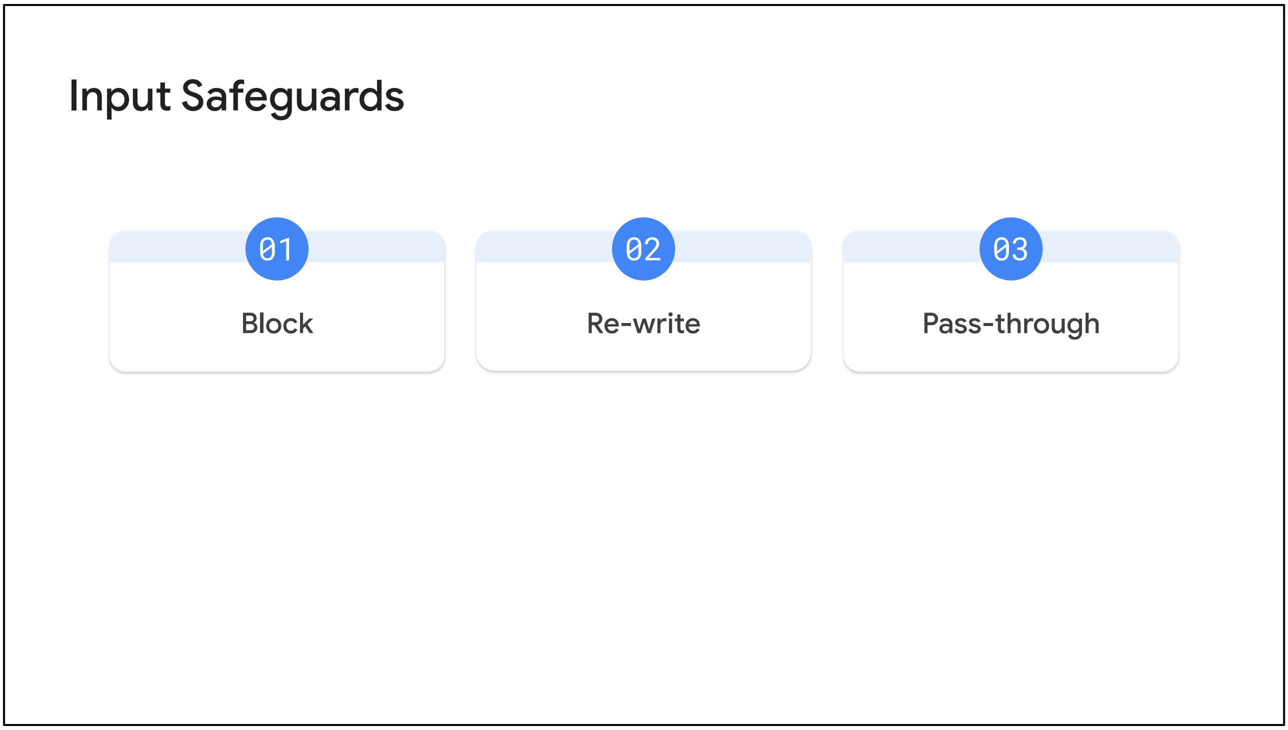
Input Safeguards

01 Block

02 Re-write

03 Pass-through

rewrite,



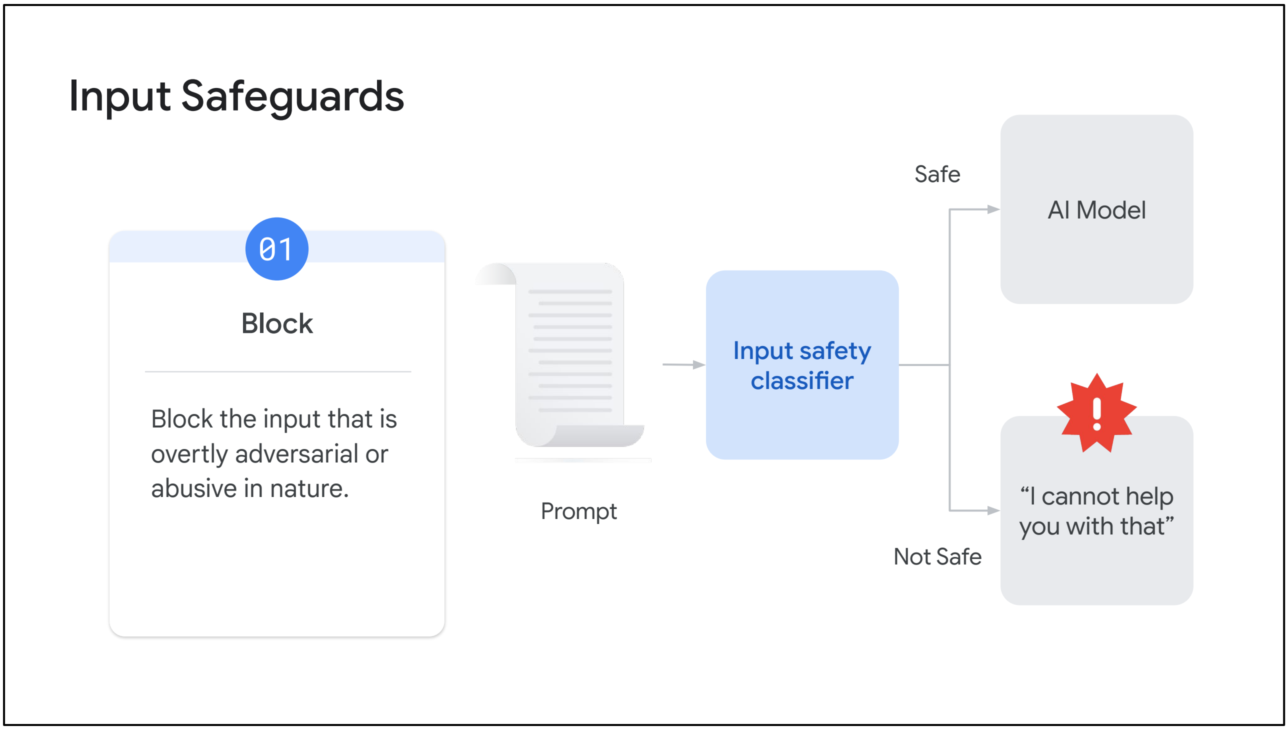
Input Safeguards

01 Block

02 Re-write

03 Pass-through

or let the request pass through.



Input Safeguards

01

Block

Block the input that is overtly adversarial or abusive in nature.

Prompt

Input safety classifier

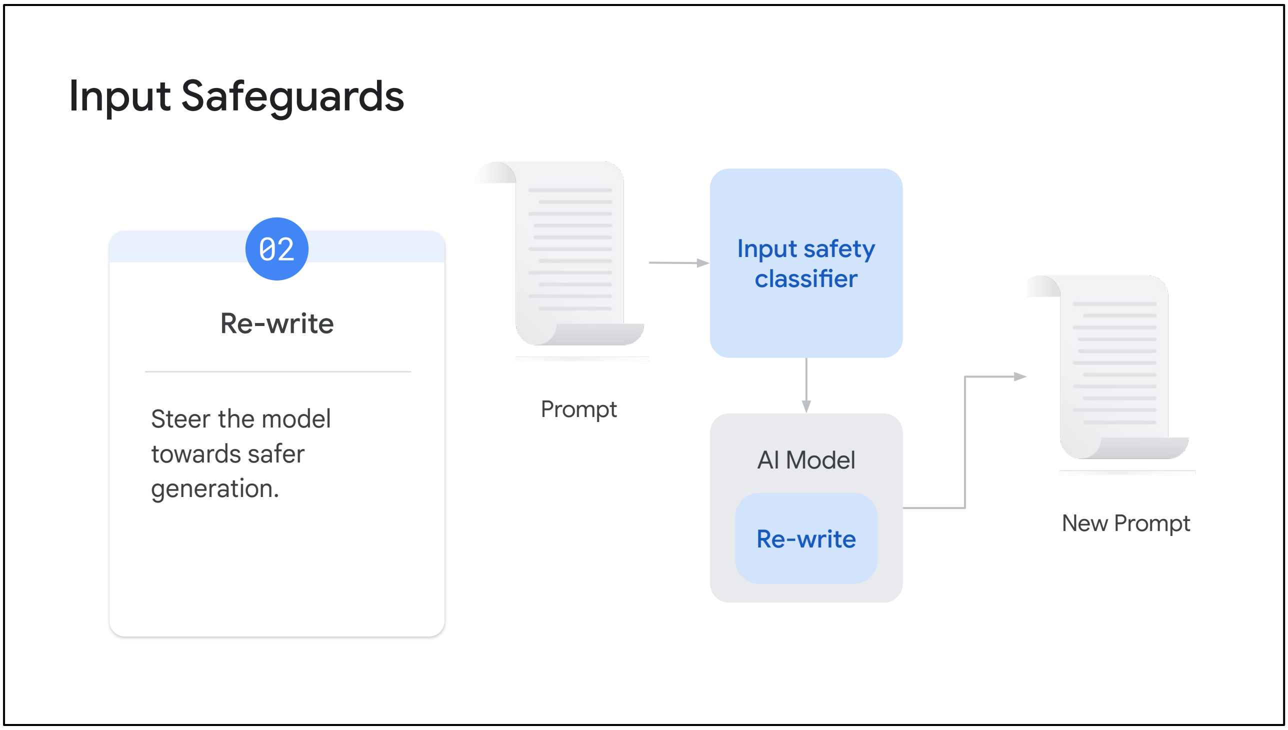
Safe

AI Model

Not Safe

“I cannot help you with that”

If the input is overtly adversarial or abusive in nature, the system can block it from being sent to the model and instead output a pre-scripted response.
For example, if you ask a chatbot “how to rob a bank,” the classifiers will recognize this input as a harmful prompt and suggest the system to block it. You will get a reply “I cannot help you with that” from the chatbot, along with why it is a wrong idea and a national support helpline to reach out to if you believe you are in a difficult situation.



Input Safeguards

02

Re-write

Steer the model towards safer generation.

Prompt

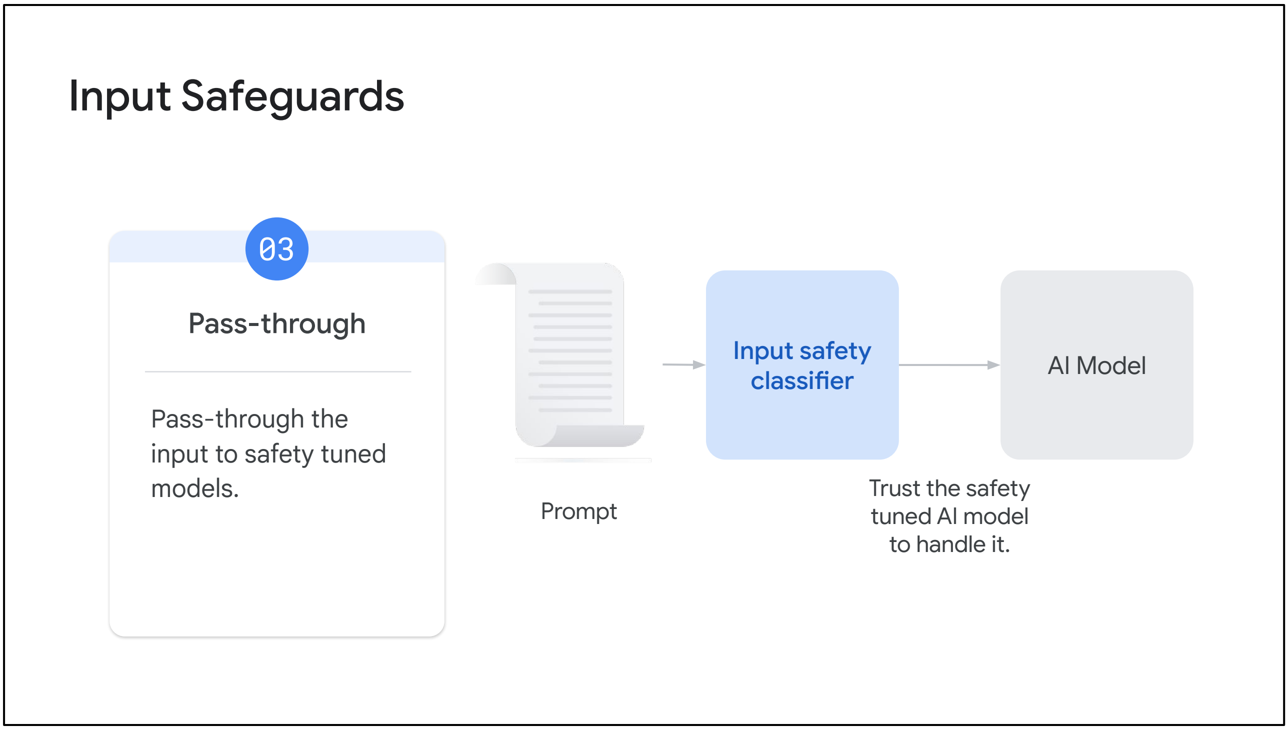
Input safety classifier

AI Model

Re-write

New Prompt

If the input is likely to elicit an unsafe response, then prompt engineering, control tokens, or style transfers can be used to attempt to steer the model towards safer generation.



Input Safeguards

03

Pass-through

Pass-through the input to safety tuned models.

Prompt

Input safety classifier

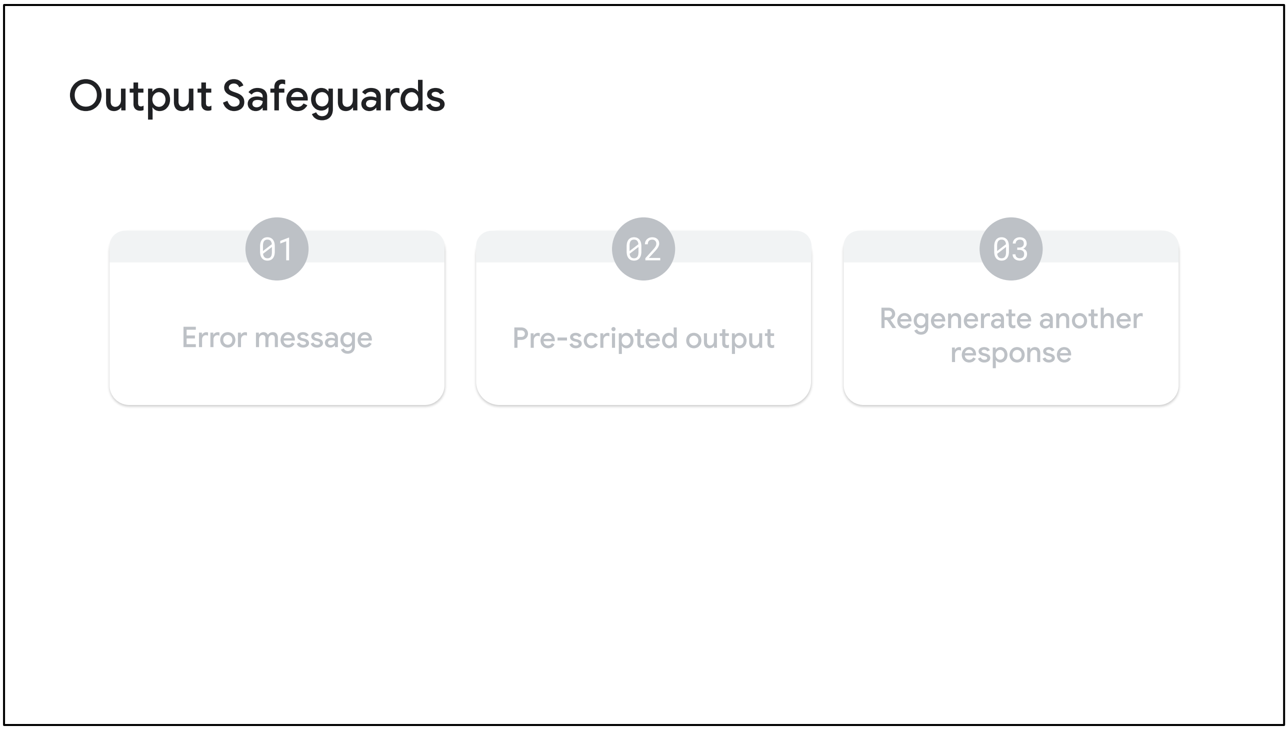
AI Model

Trust the safety tuned AI model to handle it.

Depending on how much the AI model has already been safety tuned or custom tuned for the application use, you can pass through the input with toxic data and trust the model to safely handle the generated output.

A well-tuned model can produce a more capable response which can be more insightful than the prefixed errors or messages.

It’s worth emphasizing that understanding what kind of toxic data your safety-tuned AI model can take is very important in this approach.



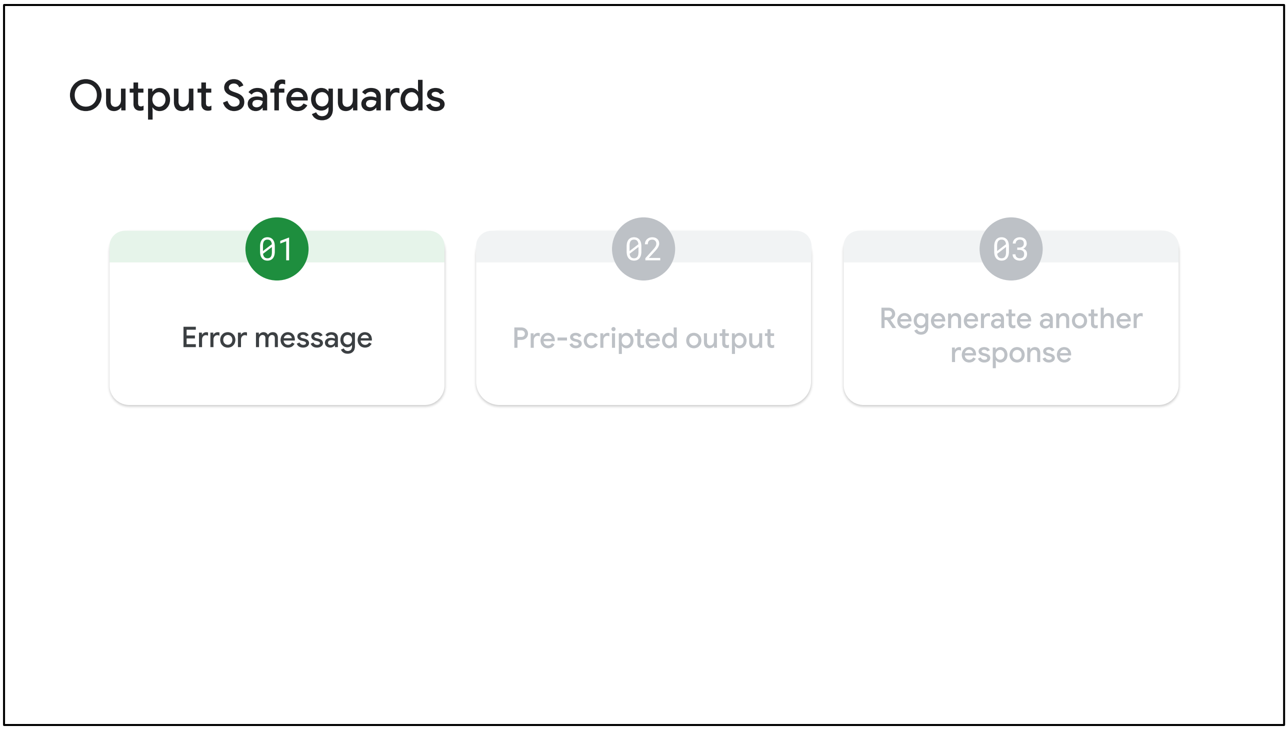
Output Safeguards

01 Error message

02 Pre-scripted output

03 Regenerate another response

Even if the input safeguards are in place and the model has been safety tuned, the model may still output unacceptable content. This is why output safeguards are equally important.
With output safeguards, you can use classifiers to detect harmful output, and perform one of the following actions on the output.



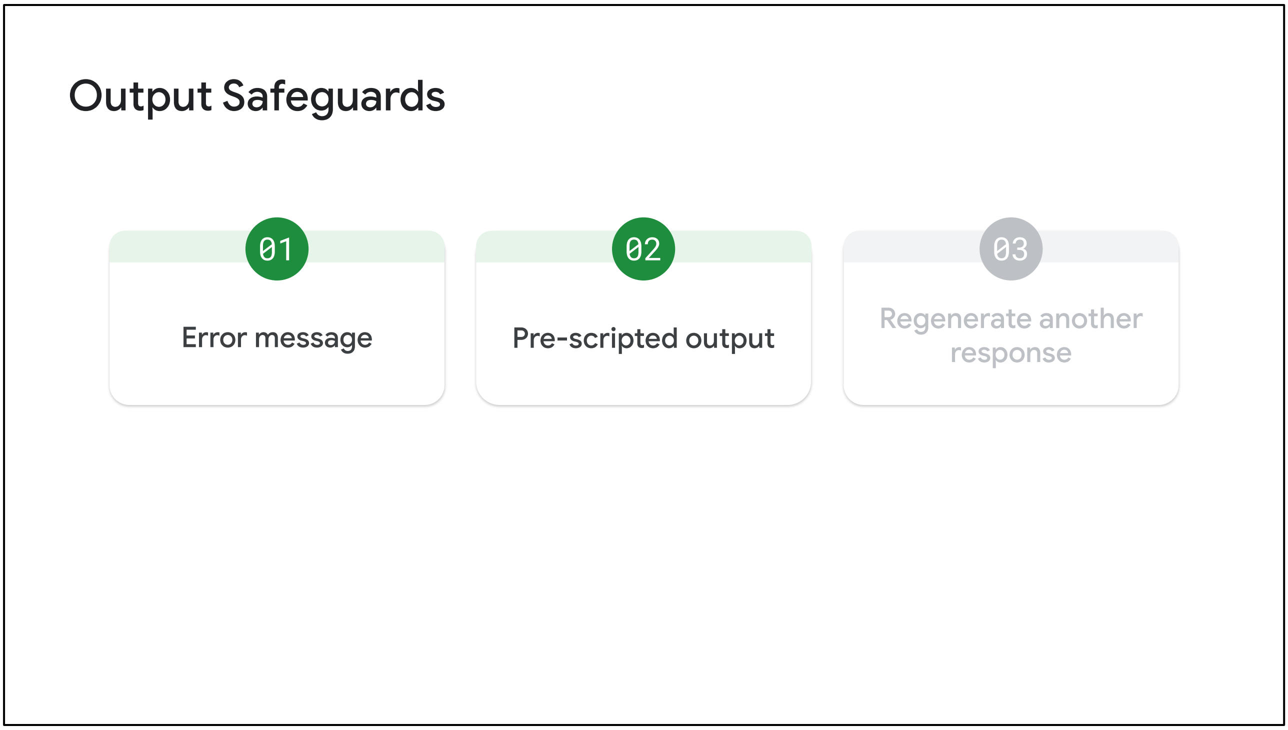
Output Safeguards

01 Error message

02 Pre-scripted output

03 Regenerate another response

- Provide an error message.



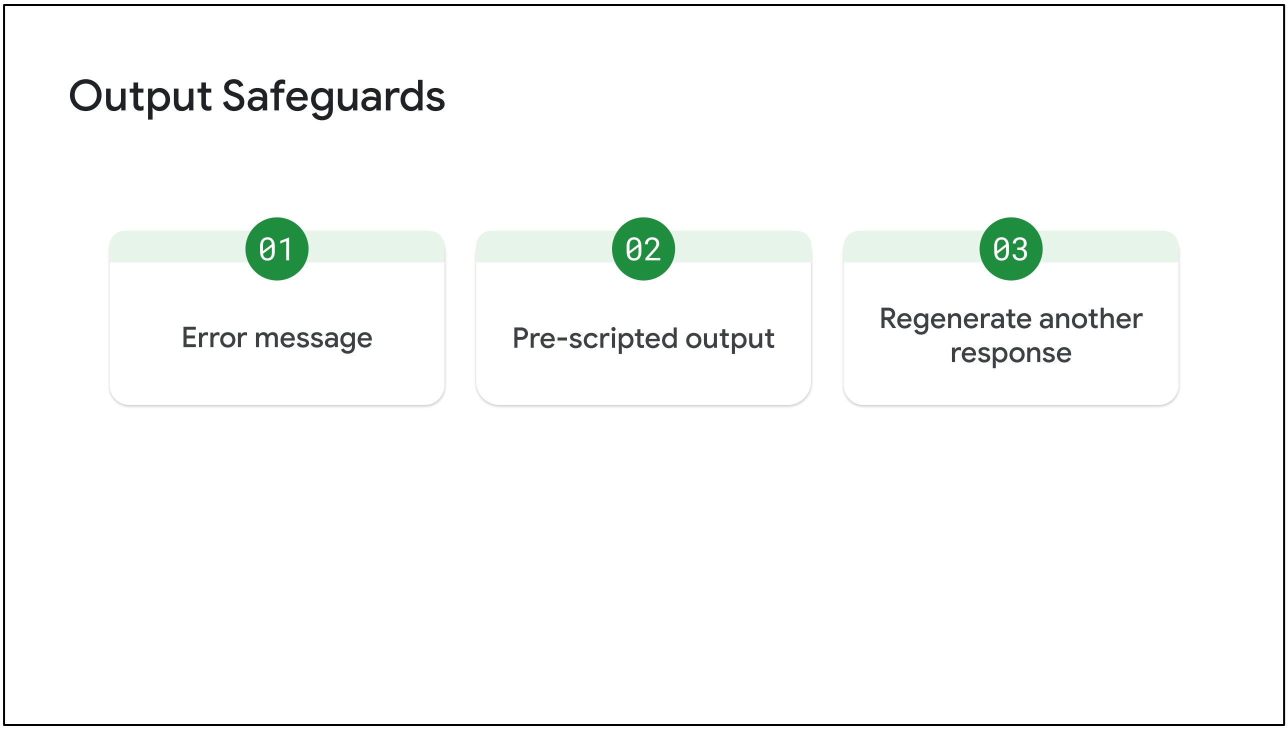
Output Safeguards

01 Error message

02 Pre-scripted output

03 Regenerate another response

- Provide a pre-scripted output.



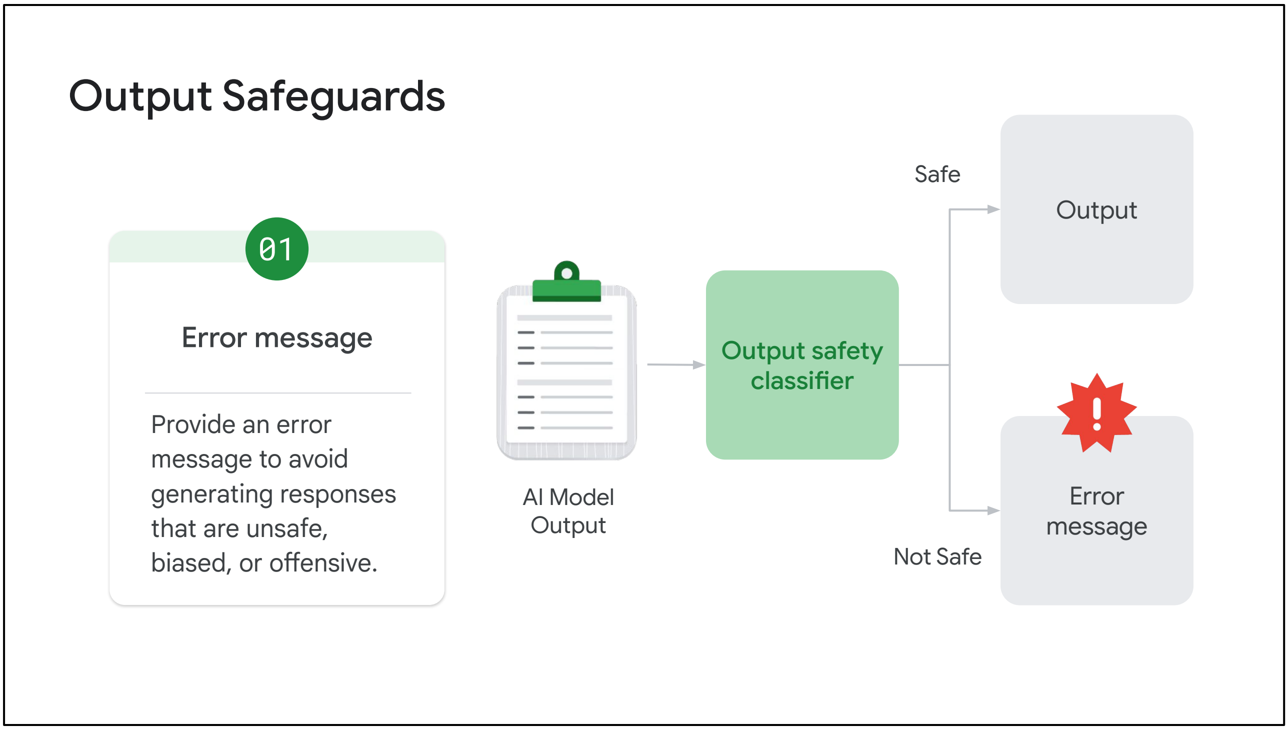
Output Safeguards

01 Error message

02 Pre-scripted output

03 Regenerate another response

- Regenerate another response with ranking.



Output Safeguards

01

Error message

Provide an error message to avoid generating responses that are unsafe, biased, or offensive.

AI Model Output

Output safety classifier

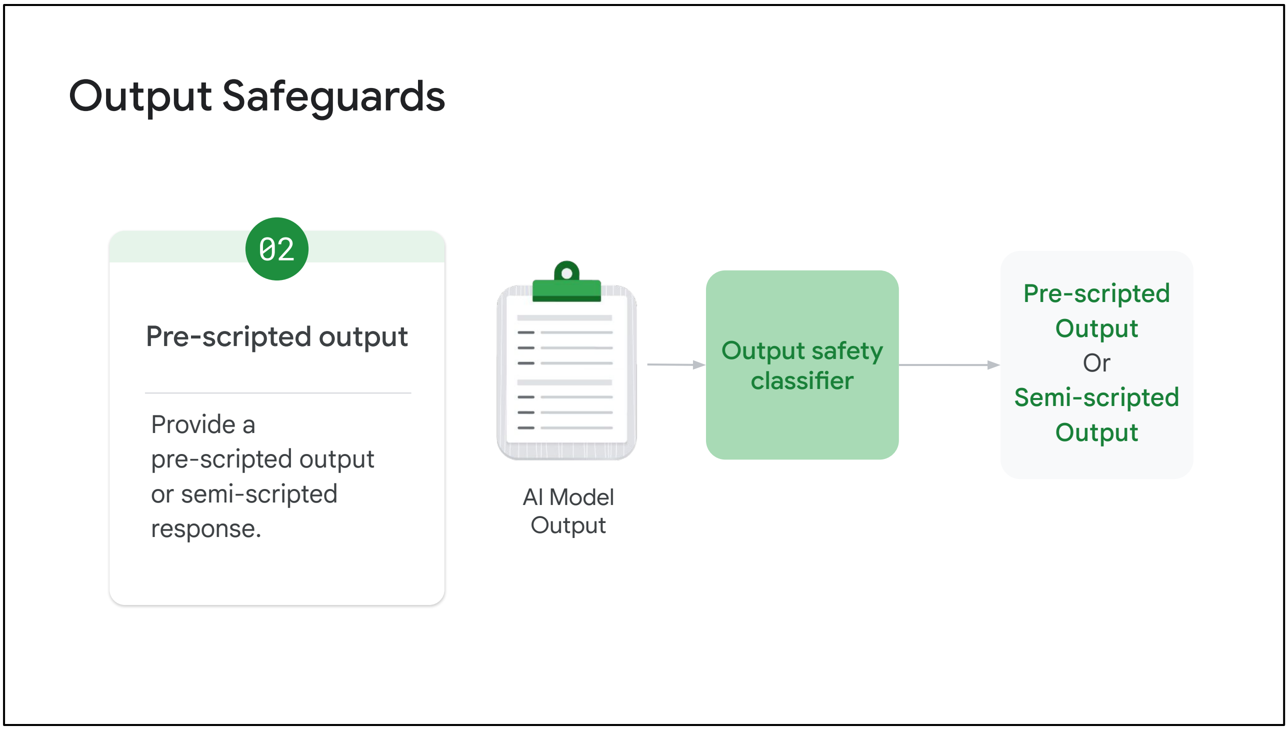
Safe

Output

Not Safe

Error message

If the classifier determines that the output is harmful without a doubt, you can simply configure the system to produce an error message in order to avoid delivering responses that are unsafe, biased, or offensive.
Using a chatbot as an example, if your query contains such content, it will output an error message or a refusal to respond.



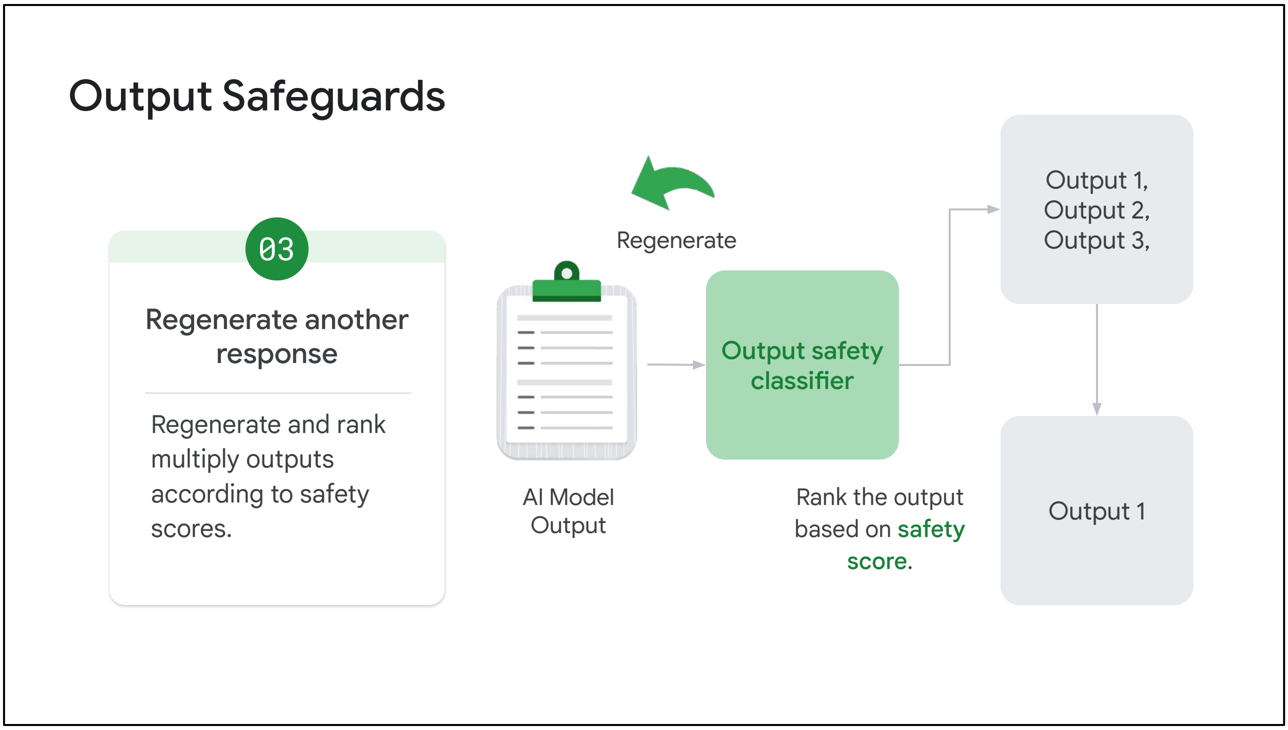
Output Safeguards

02

Pre-scripted output

Provide a pre-scripted output or semi-scripted response.

We can also configure the system to output a pre-scripted or semi-scripted output. Sometimes a semi-scripted output conveys more information than a simple error message.
Semi-scripted outputs are generated by the AI model to provide a certain level of information on why it is unsafe.
For example, instead of delivering an error message, the AI model can output “I cannot help you with that because the information about the weapon can be used to harm people and it violates our policy”.



Output Safeguards

03

Regenerate another response

Regenerate and rank multiply outputs according to safety scores.

Regenerate

AI Model Output

Output safety classifier

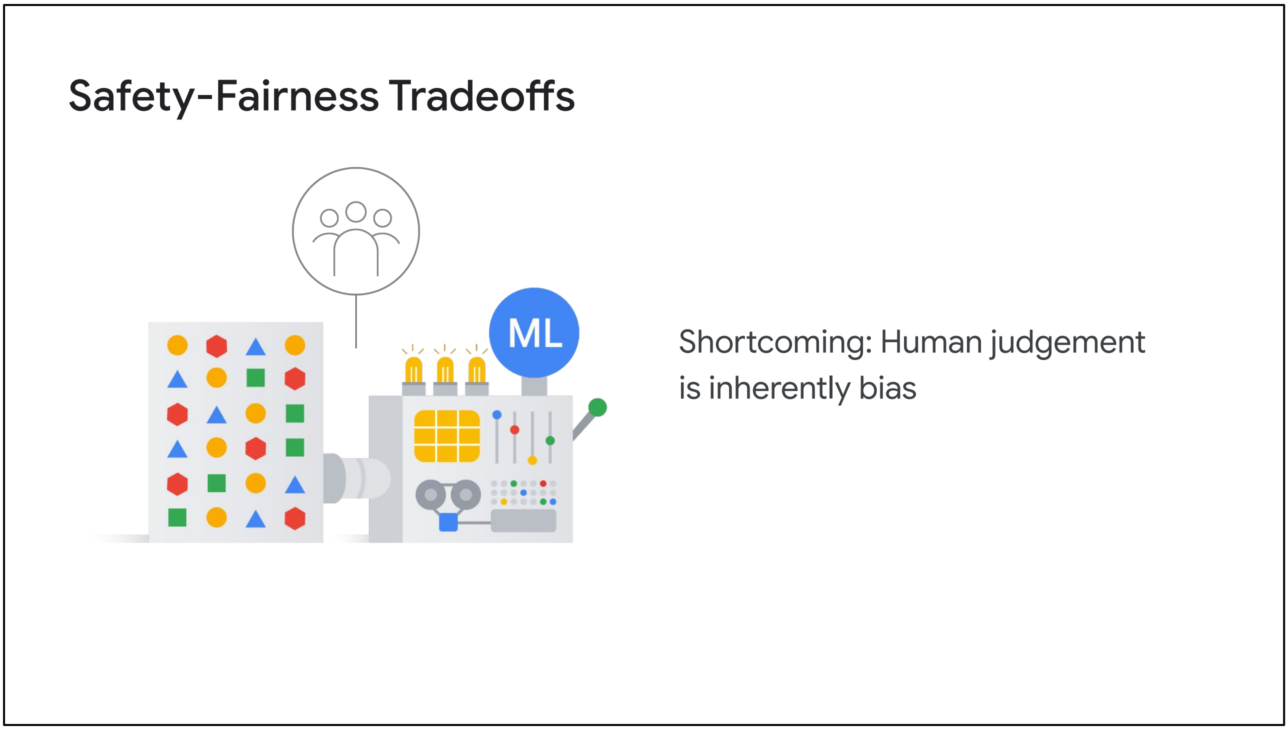
Rank the output based on safety score.

Output 1,
Output 2,
Output 3,

Output 1

Since most generative AI models are designed to be able to create multiple outputs for the same prompt, sometimes it is helpful to use this capability to handle risky outputs. When an unsafe output is detected, we generate multiple additional answers using generative AI, and rank them based on the safety score.
The model then replies with the output that has the highest safety score.

It’s worth emphasizing that the final output needs to meet all safety criteria.

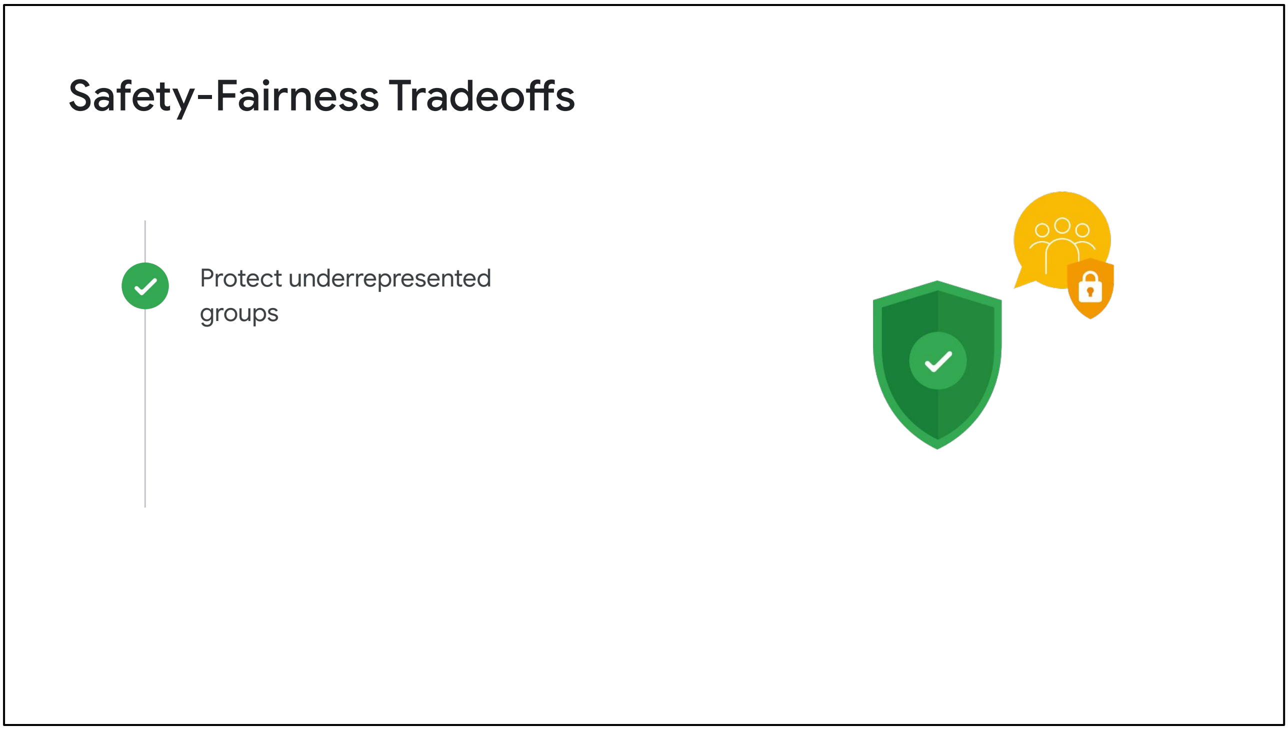


Safety-Fairness Tradeoffs

Shortcoming: Human judgement is inherently bias

Meanwhile, there are also some safety-fairness tradeoffs to be aware of.

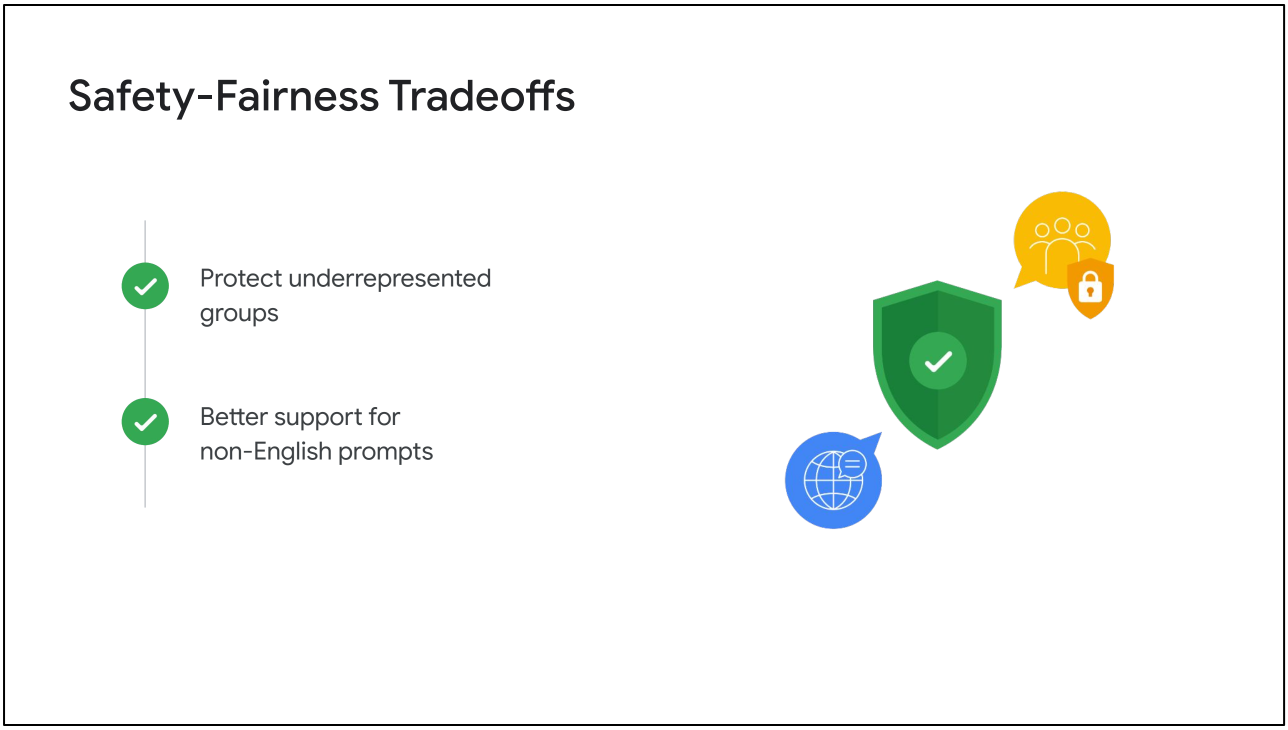
Classifiers are trained on large datasets to learn what is harmful and what is not. An example of this is using large amounts of comments on Wikipedia discussion pages to train a classifier while having a group of workers to review each one for attributes like “harassment” or “personal attack.” However, human judgment of toxicity is inherently biased and often doesn't represent the full spectrum of harassment. Even with a large and robust dataset, the model may not always learn the right patterns from it.



Safety-Fairness Tradeoffs

✓ Protect underrepresented groups

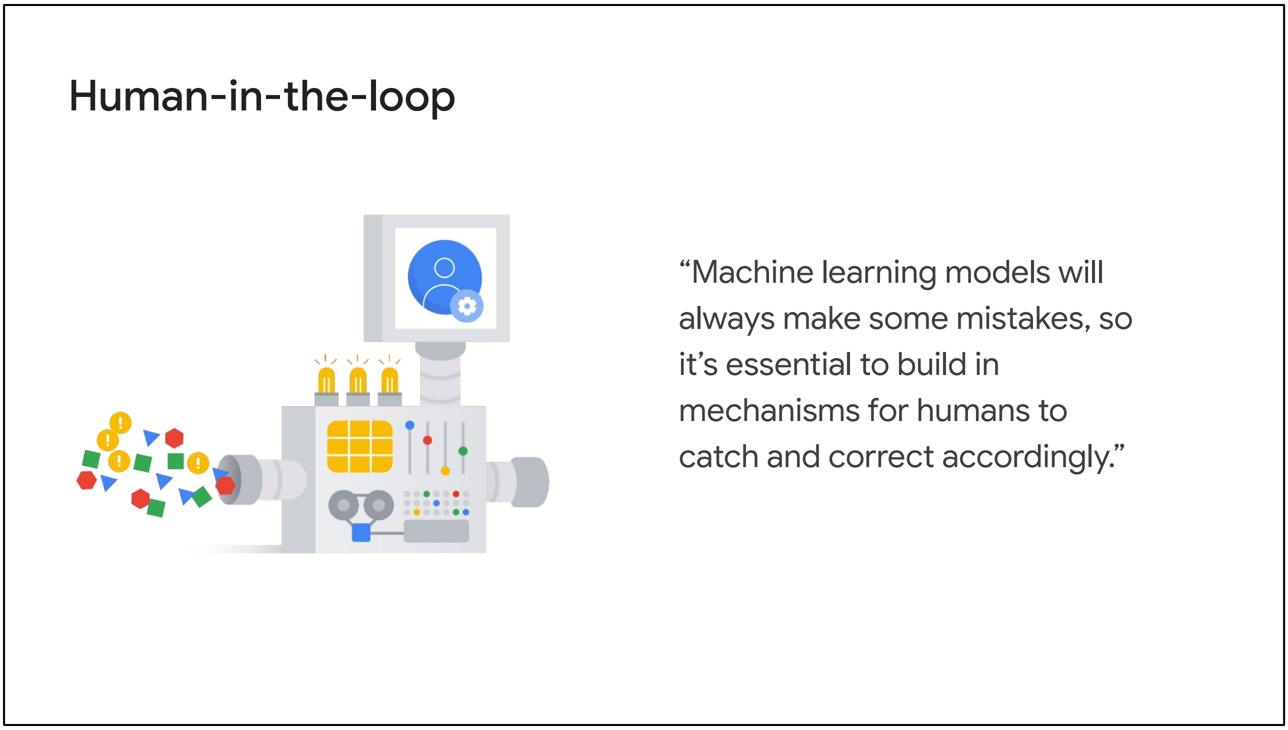
One concern is around protecting underrepresented groups.
Many LLM developers are using classifiers with low toxicity thresholds to remove hate speech, meaning they are flagging more things as toxic when the model is uncertain, which increases the risk for false positives. This may end up eliminating LLMs' ability to say things about underrepresented communities in ways that reinforce negative effects on the historically disadvantaged groups.



Safety-Fairness Tradeoffs

- ✓ Protect underrepresented groups
- ✓ Better support for non-English prompts

The performance of classifiers around non-English prompts is important.
The models are also more likely to interpret innocent non-English phrases as hate speech and harassment. And when hate speech is written in veiled ways, such as with slang or in non-English languages, classifiers can be easily fooled.



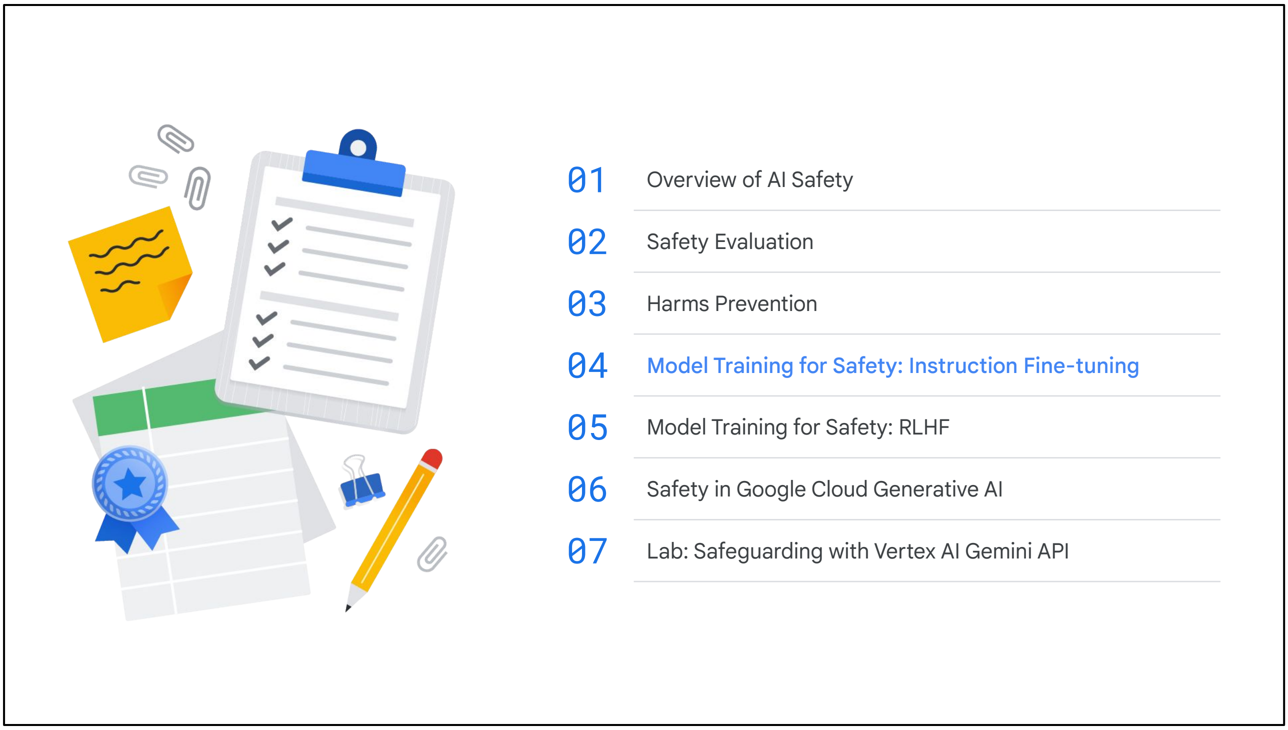
Human-in-the-loop

“Machine learning models will always make some mistakes, so it’s essential to build in mechanisms for humans to catch and correct accordingly.”

Machine learning models will always make some mistakes, so it’s essential to build in mechanisms for humans to catch and correct accordingly. Having a human involved in a process instead of full automation, also known as human-in-the-loop, is always important and highly recommended despite the fact that you can employ classifiers to automate input and output guard railing.

Integrating human oversight into the validation and review of AI outputs is important for critical or high-risk applications.

This allows humans to intervene if necessary, and provide feedback to the AI system.



01 Overview of AI Safety

02 Safety Evaluation

03 Harms Prevention

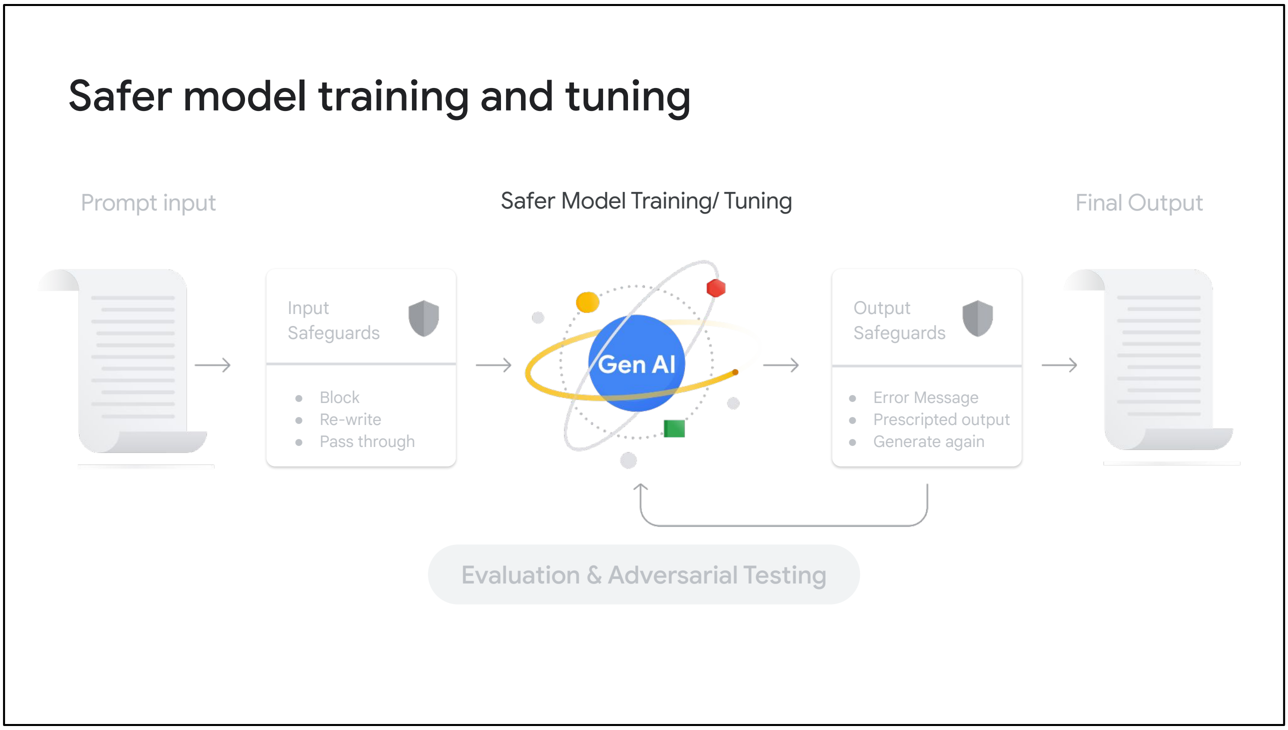
04 Model Training for Safety: Instruction Fine-tuning

05 Model Training for Safety: RLHF

06 Safety in Google Cloud Generative AI

07 Lab: Safeguarding with Vertex AI Gemini API

Let’s take some time to explore model training for safety and discuss what you can do to tune the model itself.



Safer model training and tuning

Prompt input

Safer Model Training/ Tuning

Final Output

Input Safeguards

- Block
- Re-write
- Pass through

Gen AI

Output Safeguards

- Error Message
- Prescripted output
- Generate again

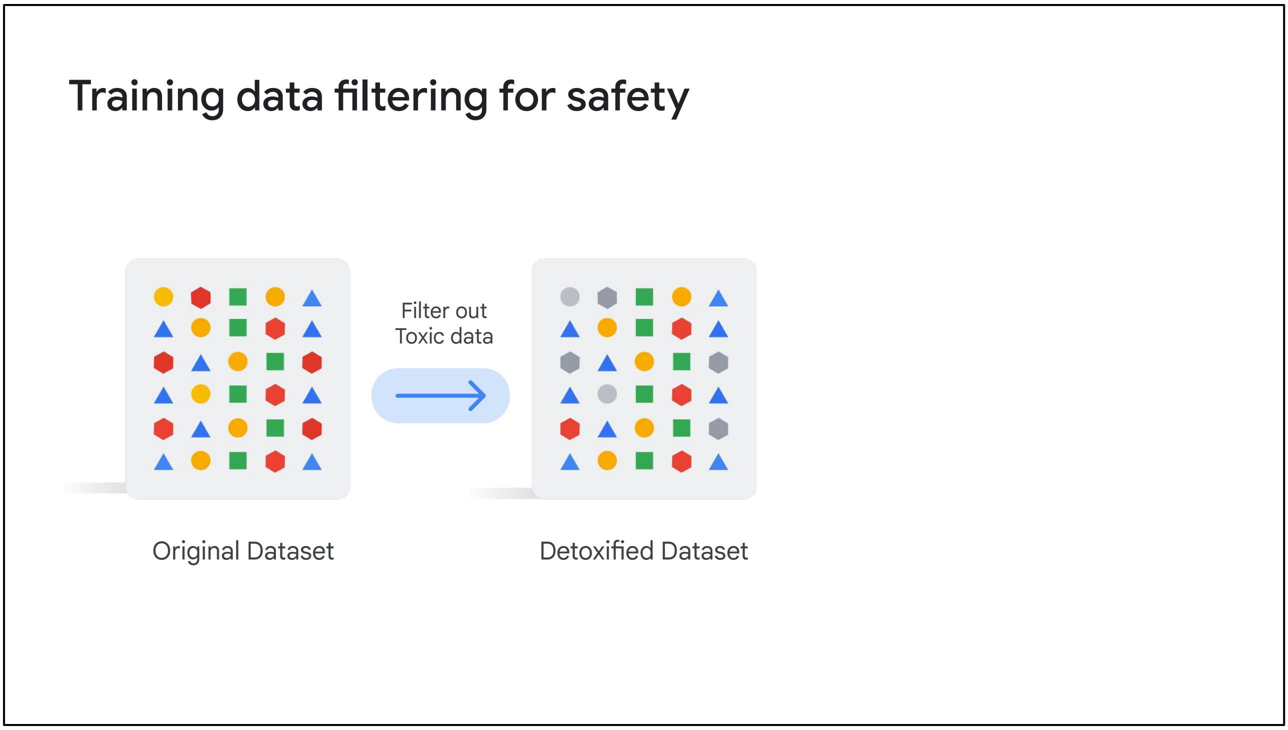
Evaluation & Adversarial Testing

Although harm prevention is important to mitigate unintentional behavior and related risks, it provides evasive methods that can reduce AI's helpfulness and capabilities. So how do we train an AI model that more closely aligns with our safety values, while at the same time, ensuring it still retains AI helpfulness and capabilities?

Training data filtering for safety

Original Dataset

The first simple idea is to filter the training data for safety purposes.



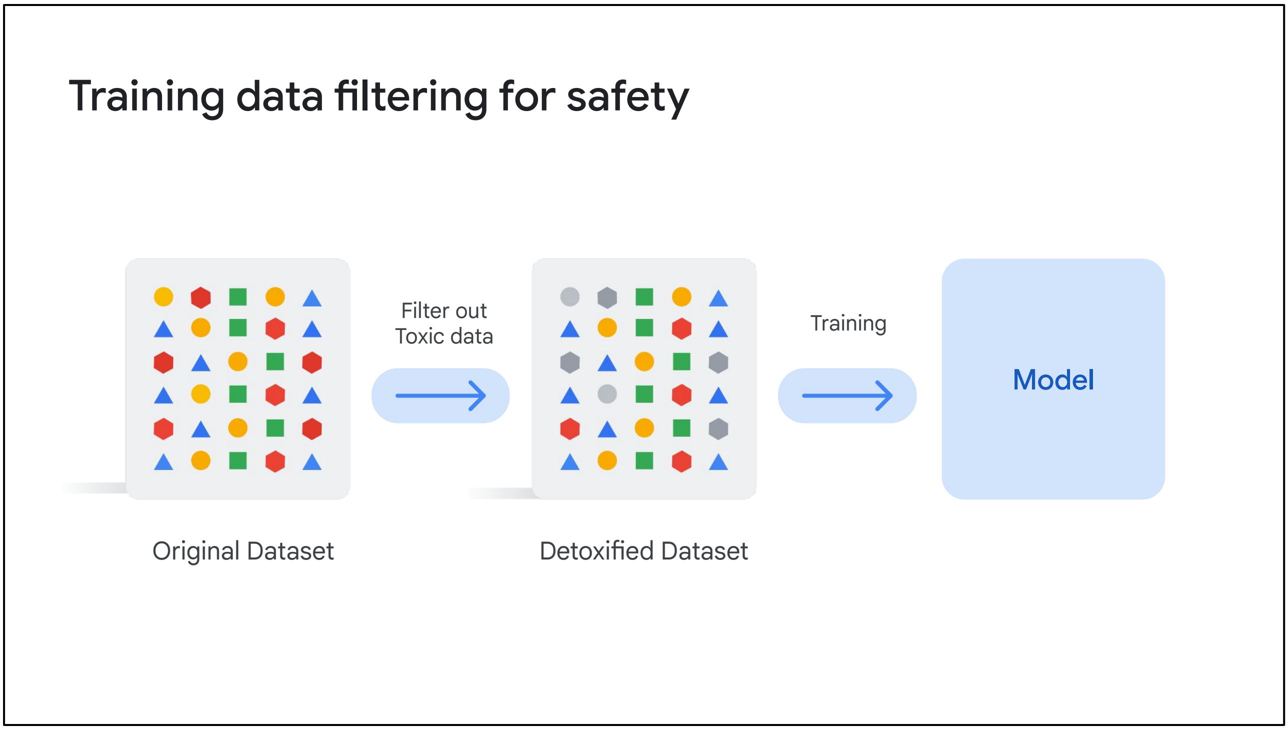
Training data filtering for safety

Filter out
Toxic data

Original Dataset

Detoxified Dataset

By filtering toxic data out from training datasets, you can naturally reduce the possibility



Training data filtering for safety

Filter out Toxic data

Training

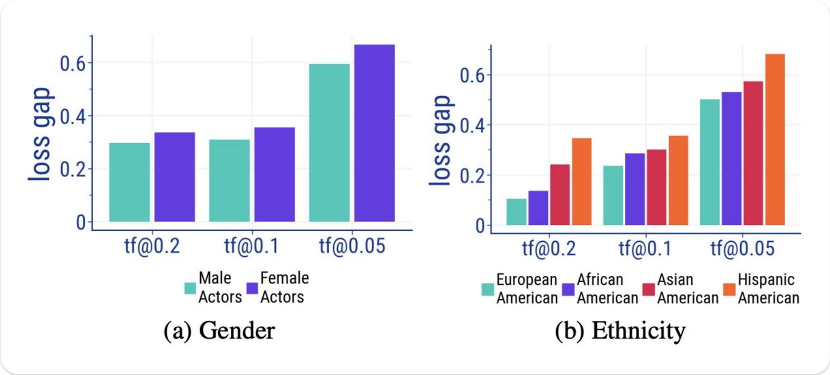
Model

Original Dataset

Detoxified Dataset

that the generative AI model generates harmful output.
You can achieve this by using safety classifiers to filter through the data.

However, over-filtering also leads to fairness issue.



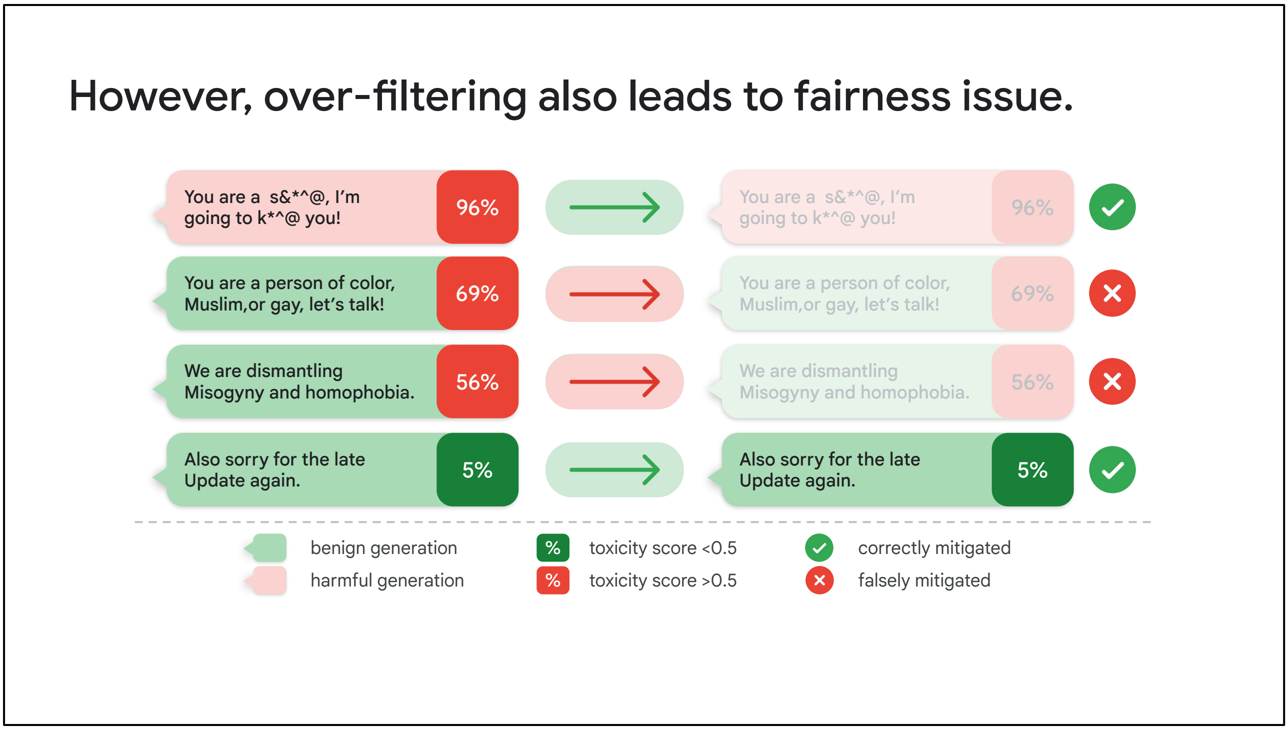
(a) Gender

(b) Ethnicity

Loss Gap between models trained on non-filtered dataset and filtered dataset, on different subsets of gender and ethnicity.

However, while this approach works for reducing toxic outputs to some extent, it also causes another issue.

By limiting the access to toxic sentences, the model may result in poor performance, especially for tasks related to underrepresented groups.



However, over-filtering also leads to fairness issue.

You are a s&*^@, I'm going to k*^@ you! 96% → You are a s&*^@, I'm going to k*^@ you! 96% ✓

You are a person of color, Muslim,or gay, let's talk! 69% → You are a person of color, Muslim,or gay, let's talk! 69% ✗

We are dismantling Misogyny and homophobia. 56% → We are dismantling Misogyny and homophobia. 56% ✗

Also sorry for the late Update again. 5% → Also sorry for the late Update again. 5% ✓

🟩 benign generation

🟥 harmful generation

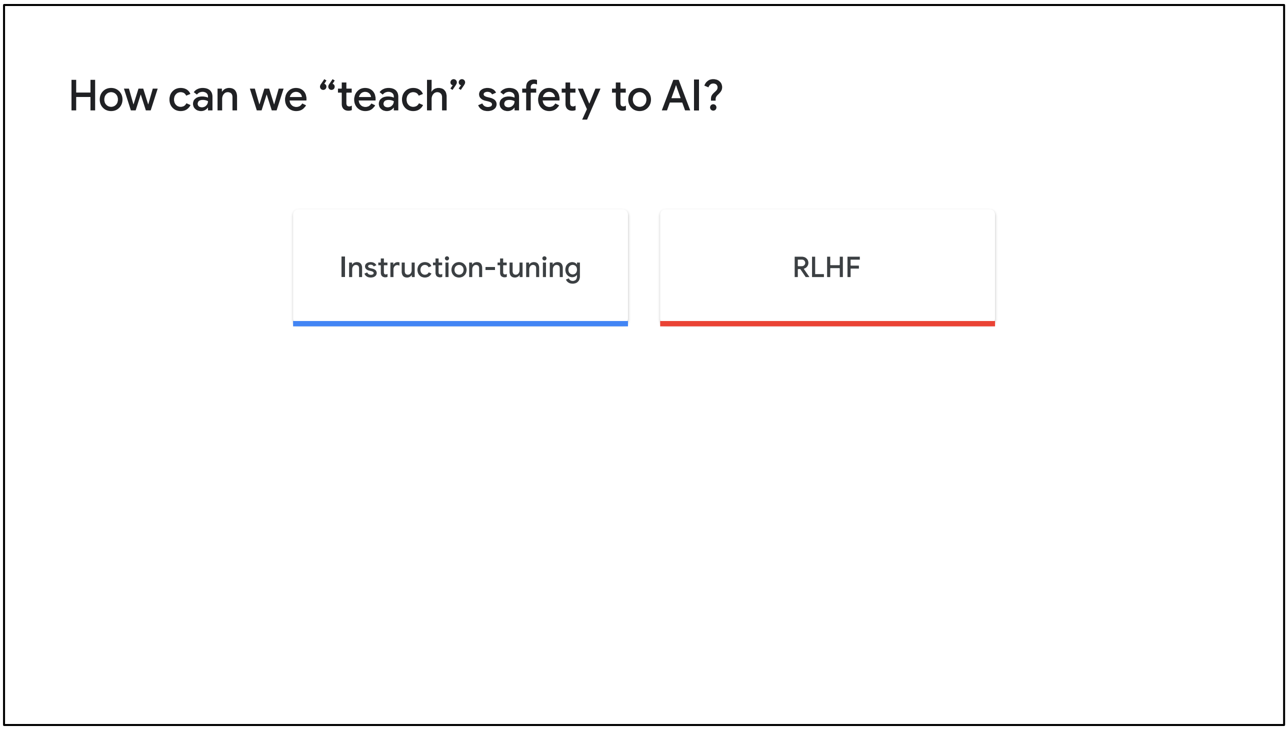
% toxicity score <0.5

% toxicity score >0.5

✓ correctly mitigated

✗ falsely mitigated

This is partially because the automated filtering methods cause false positives, especially for marginalized groups. This results in a reduction in the ability of the model to generate text about these groups, even in a positive way.
So in this case, there is a tradeoff between safety and fairness.

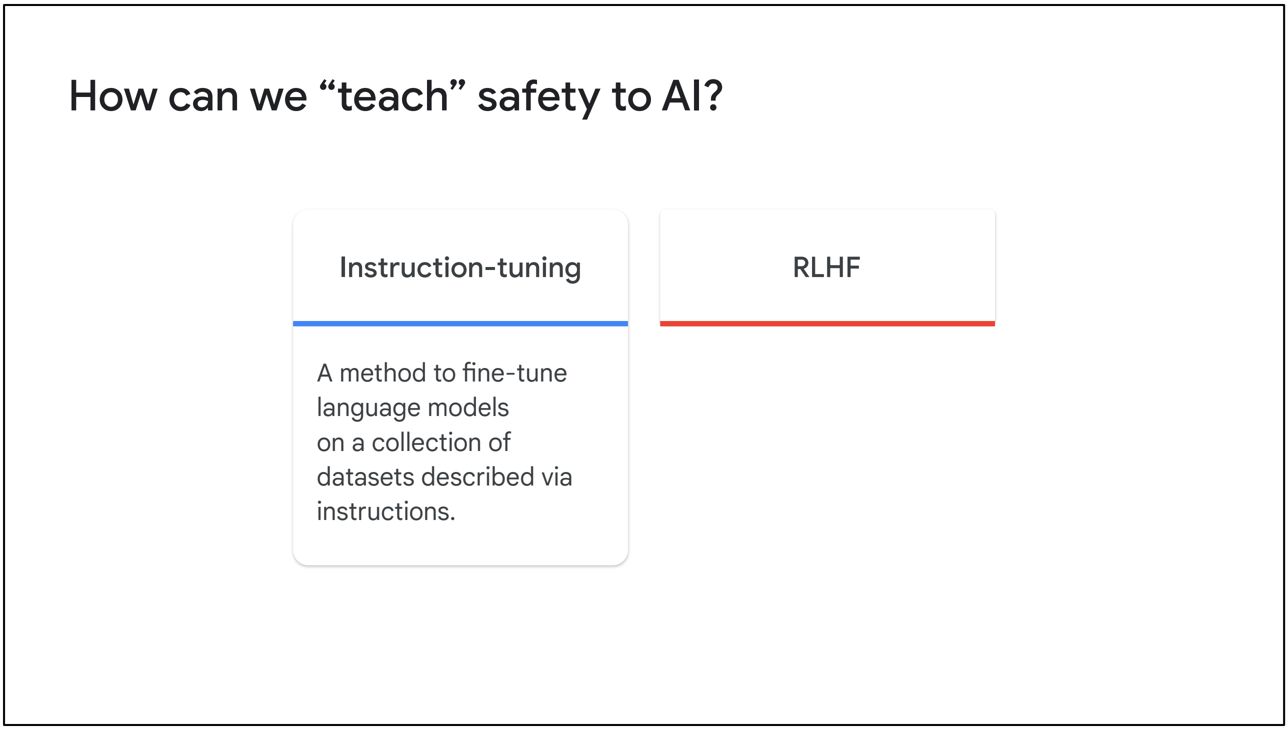


How can we “teach” safety to AI?

Instruction-tuning

RLHF

Instead of removing toxic data, can we actually “teach” the concept of safety via a fine-tuning process?
While it is still an emerging research topic, there are several known methods including instruction tuning, and reinforcement learning from human feedback (or RLHF).



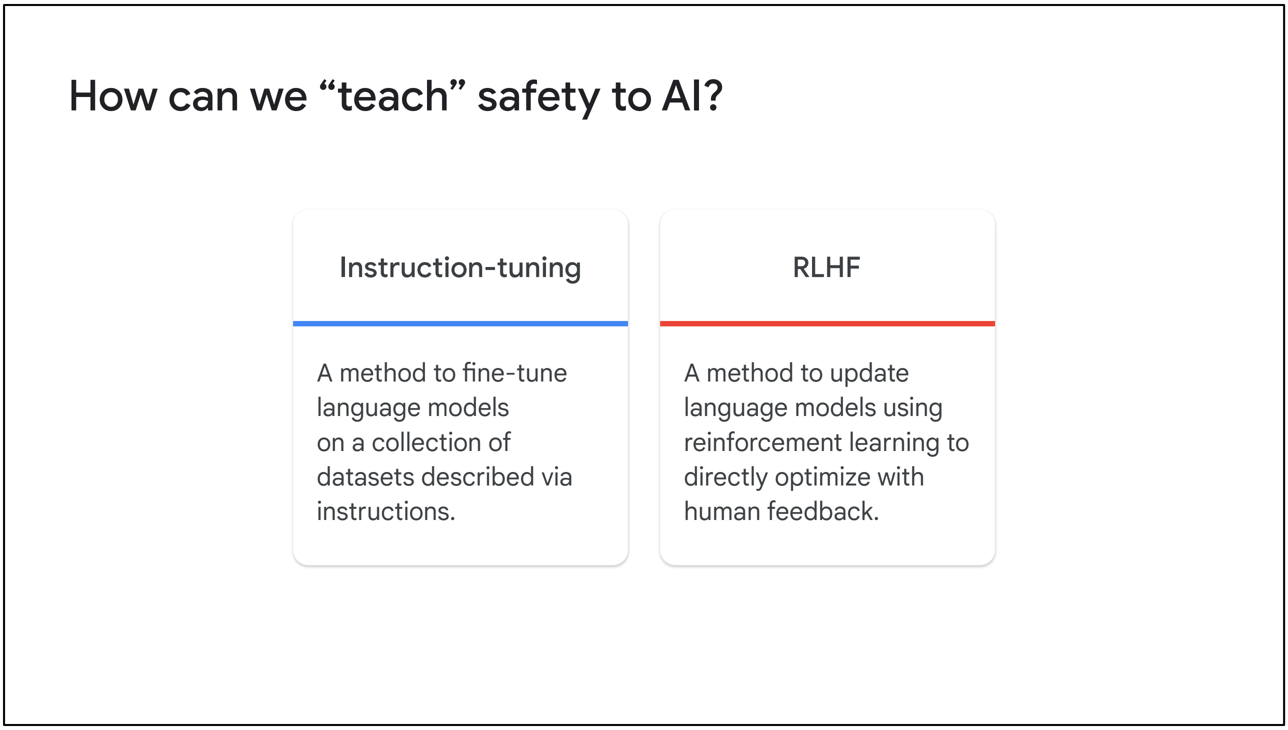
How can we “teach” safety to AI?

Instruction-tuning

A method to fine-tune language models on a collection of datasets described via instructions.

RLHF

Instruction-tuning is a generic fine-tuning method that uses a collection of datasets described via instructions. Incorporating safety-related datasets and instructions allows you to embed the safety concept into the model.



How can we “teach” safety to AI?

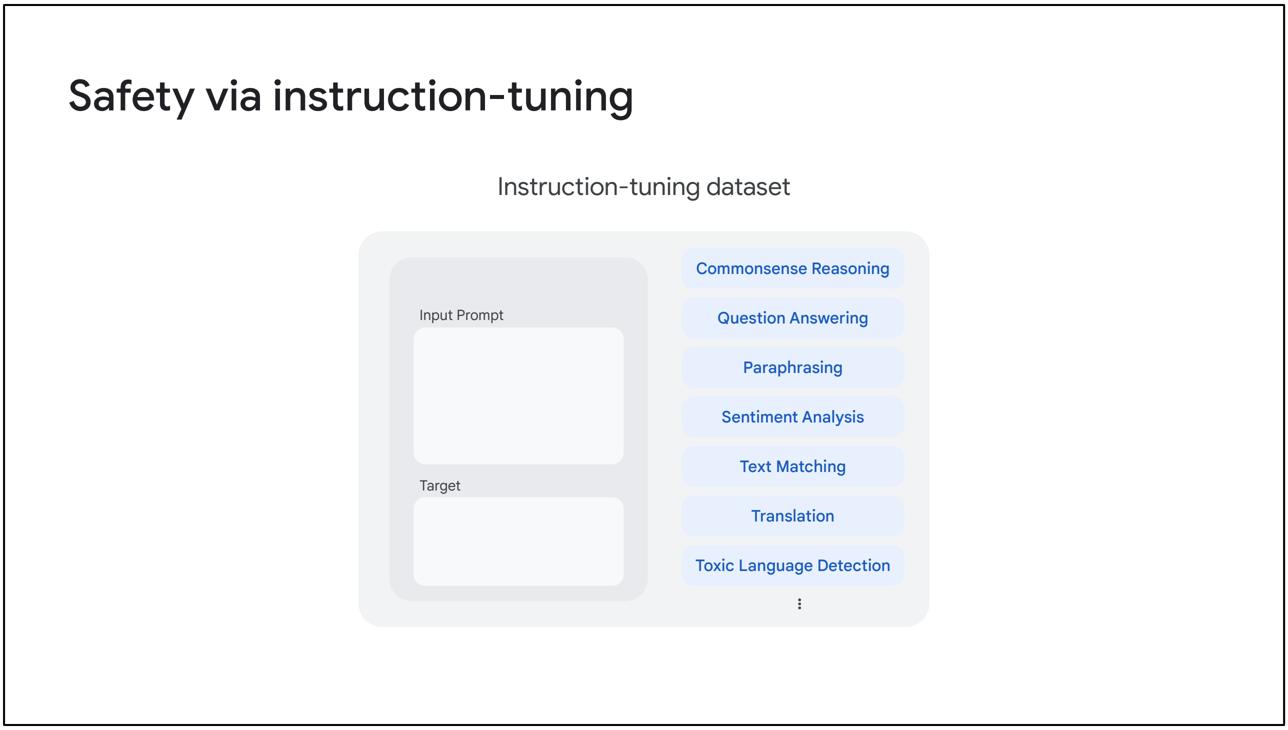
Instruction-tuning

A method to fine-tune language models on a collection of datasets described via instructions.

RLHF

A method to update language models using reinforcement learning to directly optimize with human feedback.

Another approach is to use RLHF. This technique optimizes language models directly from human preference feedback. By providing safety-related feedback, you are able to help make the model more aligned to human values.



Safety via instruction-tuning

Instruction-tuning dataset

Input Prompt

Target

Commonsense Reasoning

Question Answering

Paraphrasing

Sentiment Analysis

Text Matching

Translation

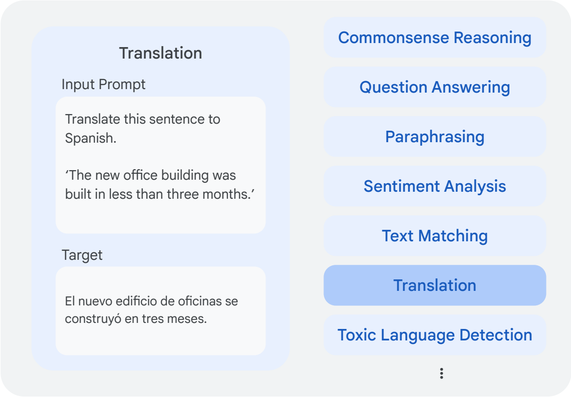
Toxic Language Detection

⋮

Let’s look a bit deeper into instruction tuning. Large language models are trained with massive pre-training datasets and pre-training tasks. These tasks are mainly designed to acquire general capabilities to handle language.
Instruction tuning usually follows the pre-training phase and aims to teach more task-related capabilities using a collection of specific task datasets, such as translation, reasoning, question answering, and so on.

Safety via instruction-tuning

Instruction-tuning dataset



Translation

Input Prompt

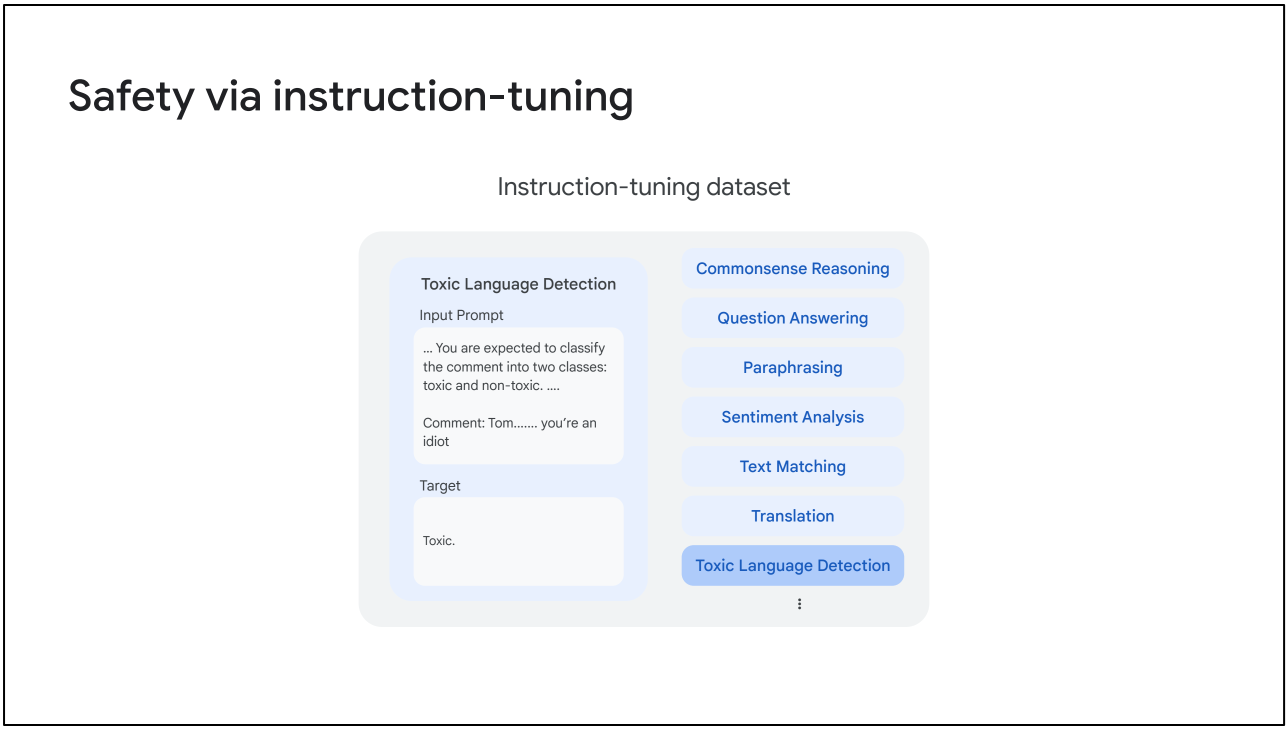
Translate this sentence to Spanish.

'The new office building was built in less than three months.'

Target

El nuevo edificio de oficinas se construyó en tres meses.

Each task has different inputs and outputs. For example, in the translation task, the target model gets instruction inputs as a prompt, such as "Translate this sentence to Spanish, 'The new office building was built in less than three months.'" Then, it is supposed to provide an output accordingly. The model is trained based on the gap between the responses and the prescripted target outputs.



Safety via instruction-tuning

Instruction-tuning dataset

Toxic Language Detection

Input Prompt

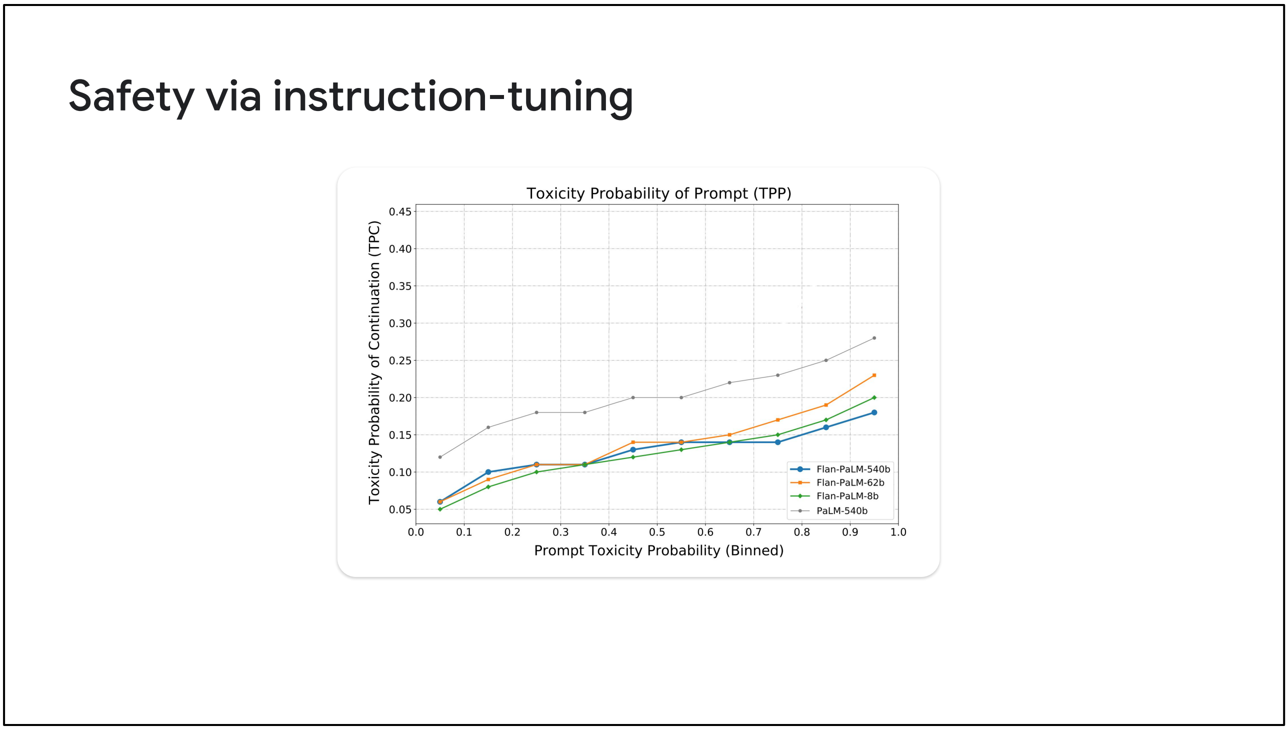
... You are expected to classify the comment into two classes: toxic and non-toxic. ....

Comment: Tom....... you're an idiot

Target

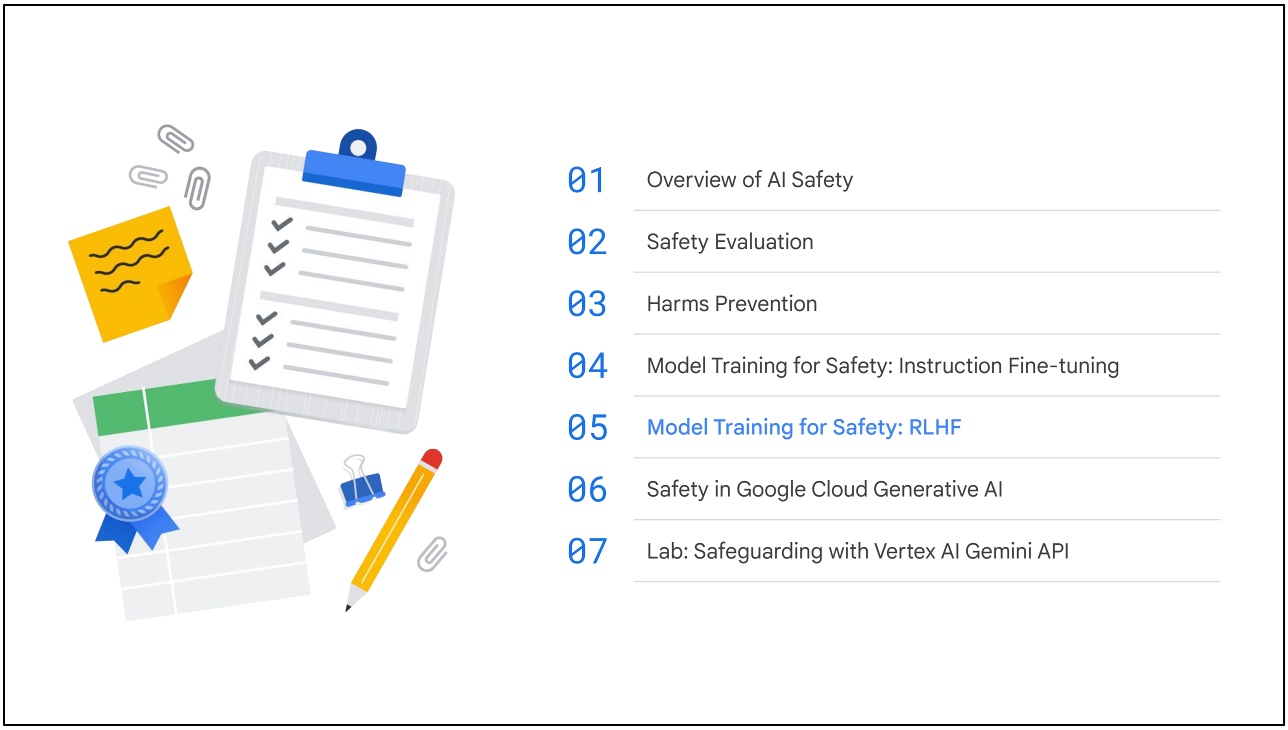
Toxic.

You can include safety related tasks, such as toxic language detection, in this instruction-tuning phase.
During the tuning process, the model is asked to detect whether a given text has toxic meaning, by replying “Toxic” or “Non-toxic.”



Safety via instruction-tuning

This approach helps detoxify models significantly from testing.
As shown in this graph from a paper, you can see instruction-tuned PaLM models, labeled as Flan-PaLM, have a lower probability to generate toxic sentences, even based on a prompt that elicits toxic continuation or response, than the base PaLM model.



01 Overview of AI Safety

02 Safety Evaluation

03 Harms Prevention

04 Model Training for Safety: Instruction Fine-tuning

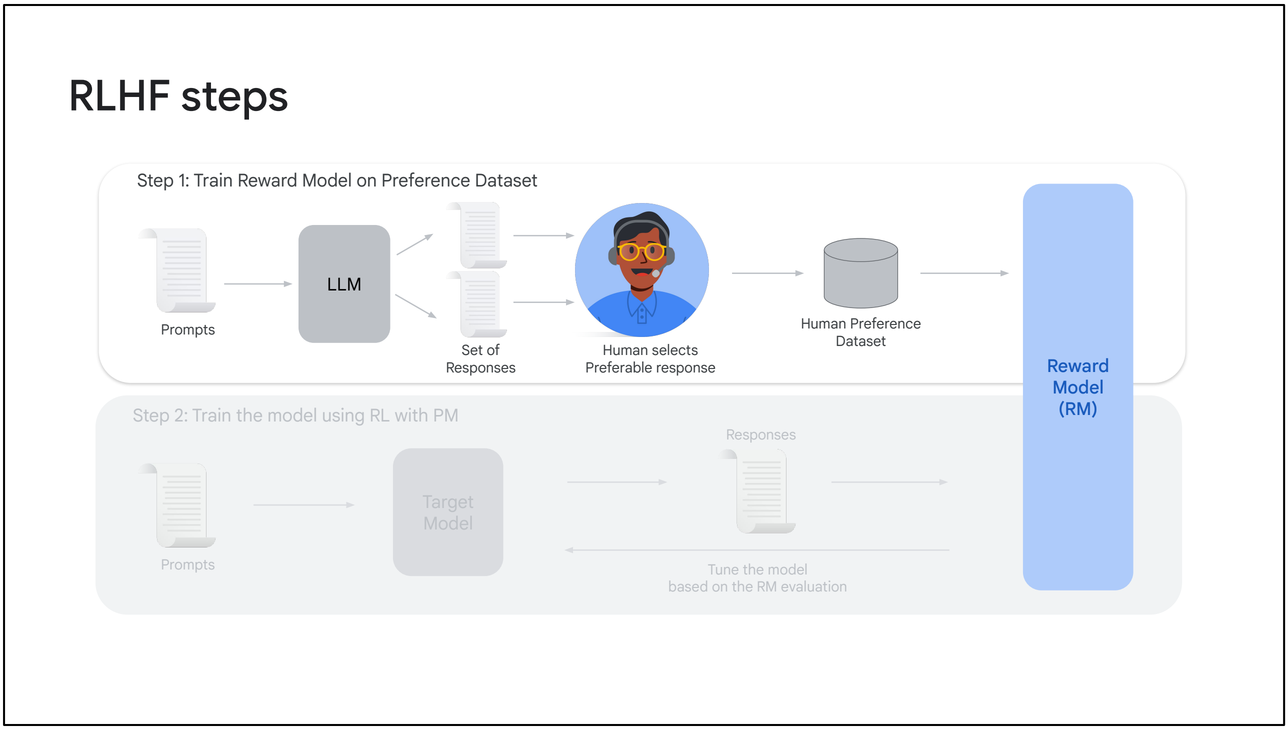
05 Model Training for Safety: RLHF

06 Safety in Google Cloud Generative AI

07 Lab: Safeguarding with Vertex AI Gemini API

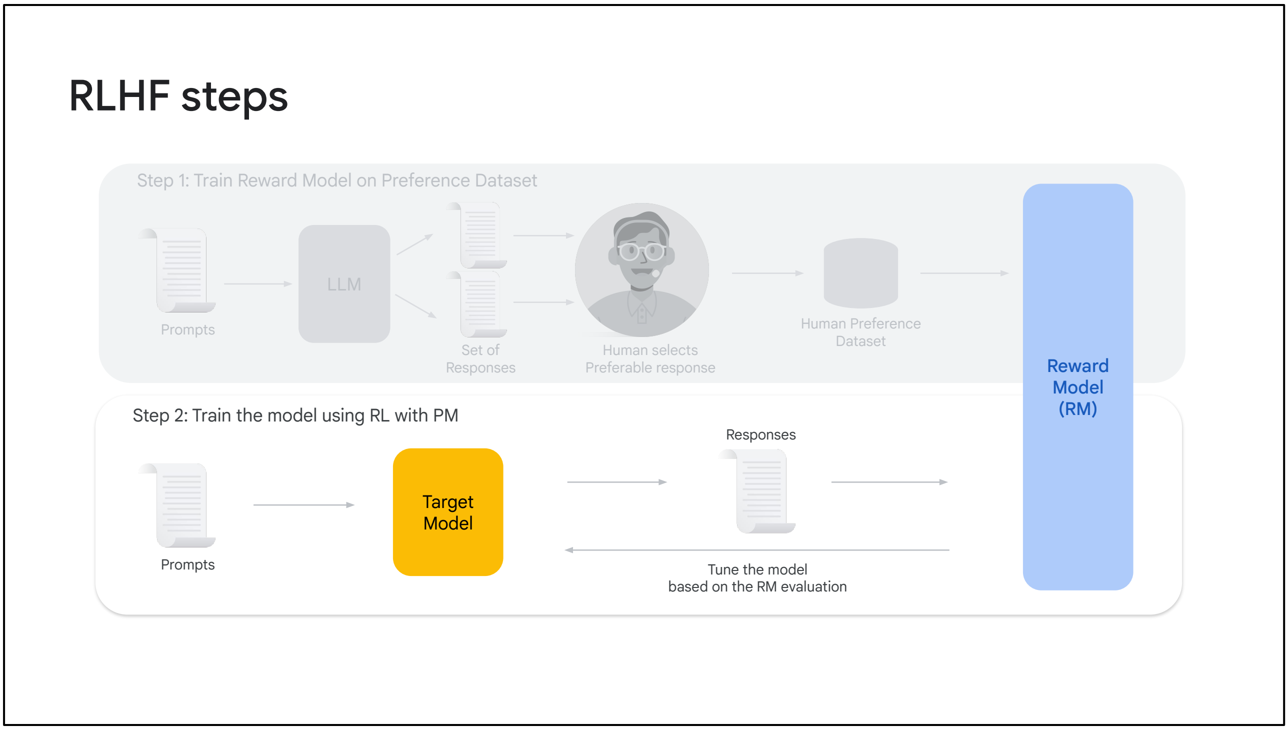
RLHF also tries to embed the concept of safety by optimizing the model using human feedback.

RLHF steps



Step 1: Train Reward Model on Preference Dataset
Prompts
LLM
Set of Responses
Human selects Preferable response
Human Preference Dataset
Reward Model (RM)
Step 2: Train the model using RL with PM
Prompts
Target Model
Responses
Tune the model based on the RM evaluation

RLHF works in the following way. It starts from training a reward model that returns preference scores on a given prompt.
To train a reward model, the first step is to generate sets or pairs of responses using an LLM which can be the target model itself or another model.
Then, the responses are passed to a human moderator who ranks them or simply judges which response is preferable.
The reward model is then trained using this human preference dataset for the next step.
For safety purposes, you would usually use a set of input prompts that are picked or designed to elicit harmful responses, and the human moderator would evaluate not only the helpfulness of the responses but also their harmlessness.



RLHF steps

Step 1: Train Reward Model on Preference Dataset

Prompts

LLM

Set of Responses

Human selects Preferable response

Human Preference Dataset

Reward Model (RM)

Step 2: Train the model using RL with PM

Prompts

Target Model

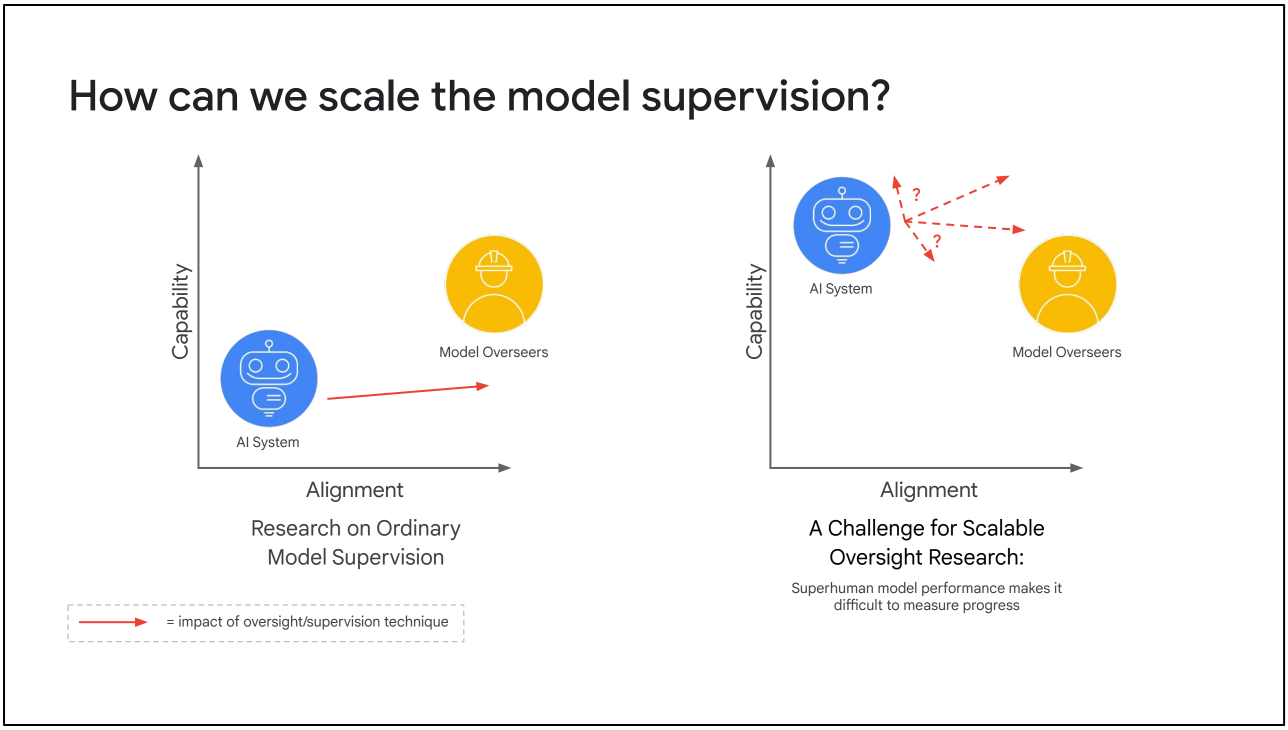
Responses

Tune the model based on the RM evaluation

In the next step, the target model you want to tune receives prompts and then generates responses accordingly. The trained reward model evaluates the responses and returns the preference score, which is supposed to reflect the human preference it was trained on.

Here, the target model is iteratively trained via a reinforcement learning strategy to internalize the human preference through the reward model.

By using special prompts for safety, and carefully building a human preference dataset for safety with human moderators, you can then embed the safety concept into the model.



How can we scale the model supervision?

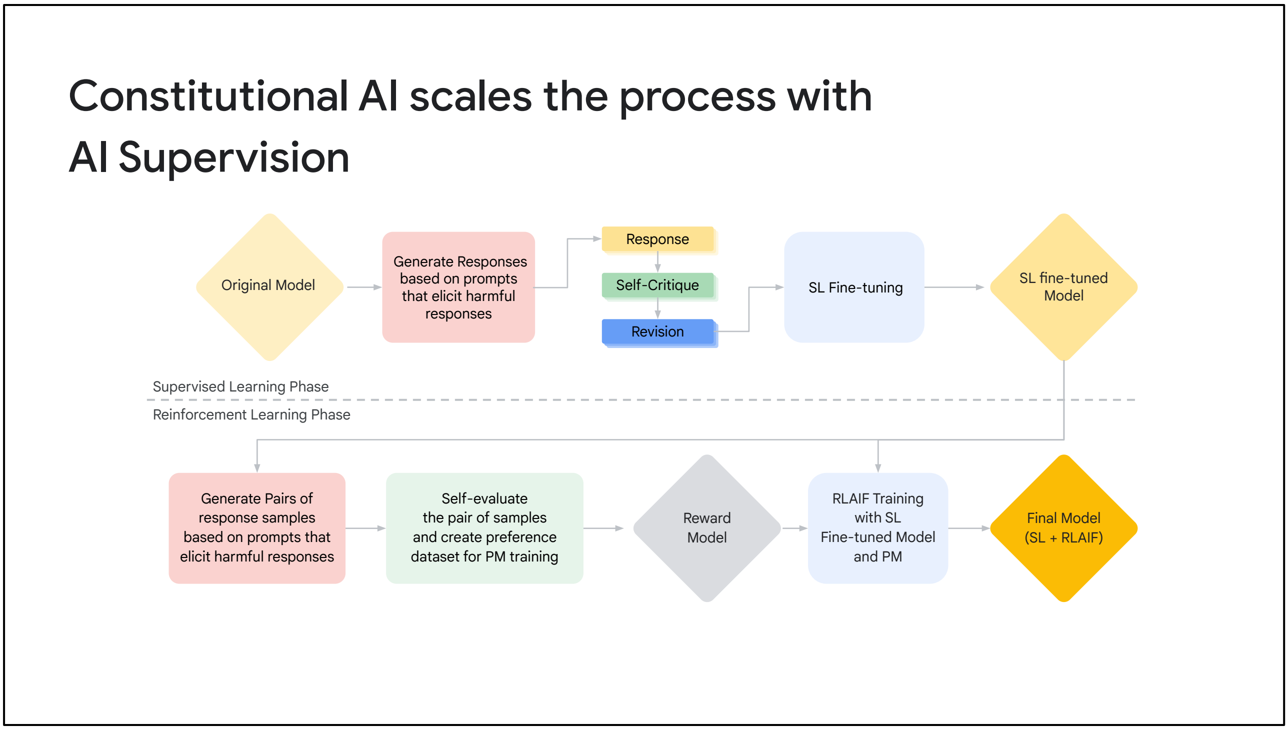
Research on Ordinary Model Supervision

A Challenge for Scalable Oversight Research:

Superhuman model performance makes it difficult to measure progress

= impact of oversight/supervision technique

Now, RLHF is a widely used technique for generative models. But, at the same time, you may need to start thinking about how to scale this manual supervision process as AI models acquire general capabilities.
Since human supervision can't be perfect and as AI grows, at some point, it may start leveraging human incapability to hide undesired outcomes in supervision.



Constitutional AI scales the process with AI Supervision

Original Model

Generate Responses based on prompts that elicit harmful responses

Response

Self-Critique

Revision

SL Fine-tuning

SL fine-tuned Model

Supervised Learning Phase

Reinforcement Learning Phase

Generate Pairs of response samples based on prompts that elicit harmful responses

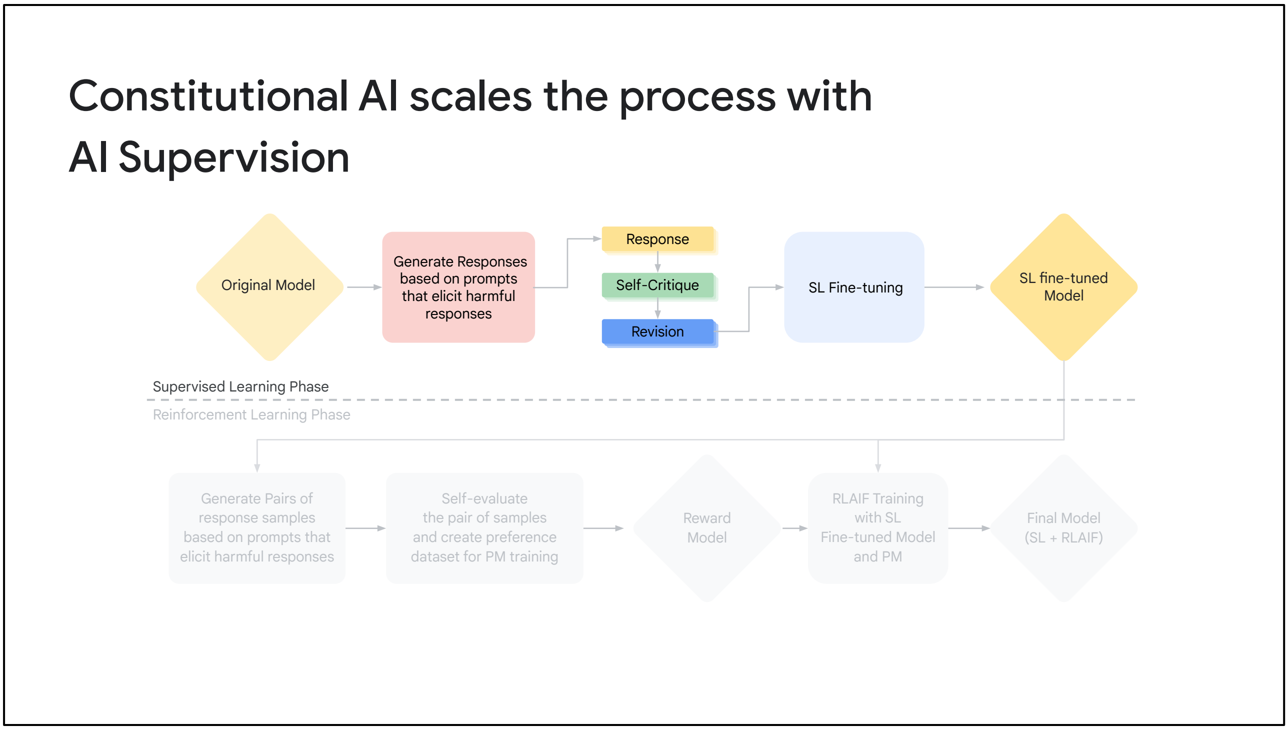
Self-evaluate the pair of samples and create preference dataset for PM training

Reward Model

RLAIF Training with SL Fine-tuned Model and PM

Final Model (SL + RLAIF)

Researchers are trying to find a way to scale the supervision approach by using AI in the process.
One of these attempts is Constitutional AI released by Anthropic. Constitutional AI uses AI evaluation in two types of tuning: supervised and reinforcement learning-based tuning.



Constitutional AI scales the process with AI Supervision

Original Model

Generate Responses based on prompts that elicit harmful responses

Response

Self-Critique

Revision

SL Fine-tuning

SL fine-tuned Model

Supervised Learning Phase

Reinforcement Learning Phase

Generate Pairs of response samples based on prompts that elicit harmful responses

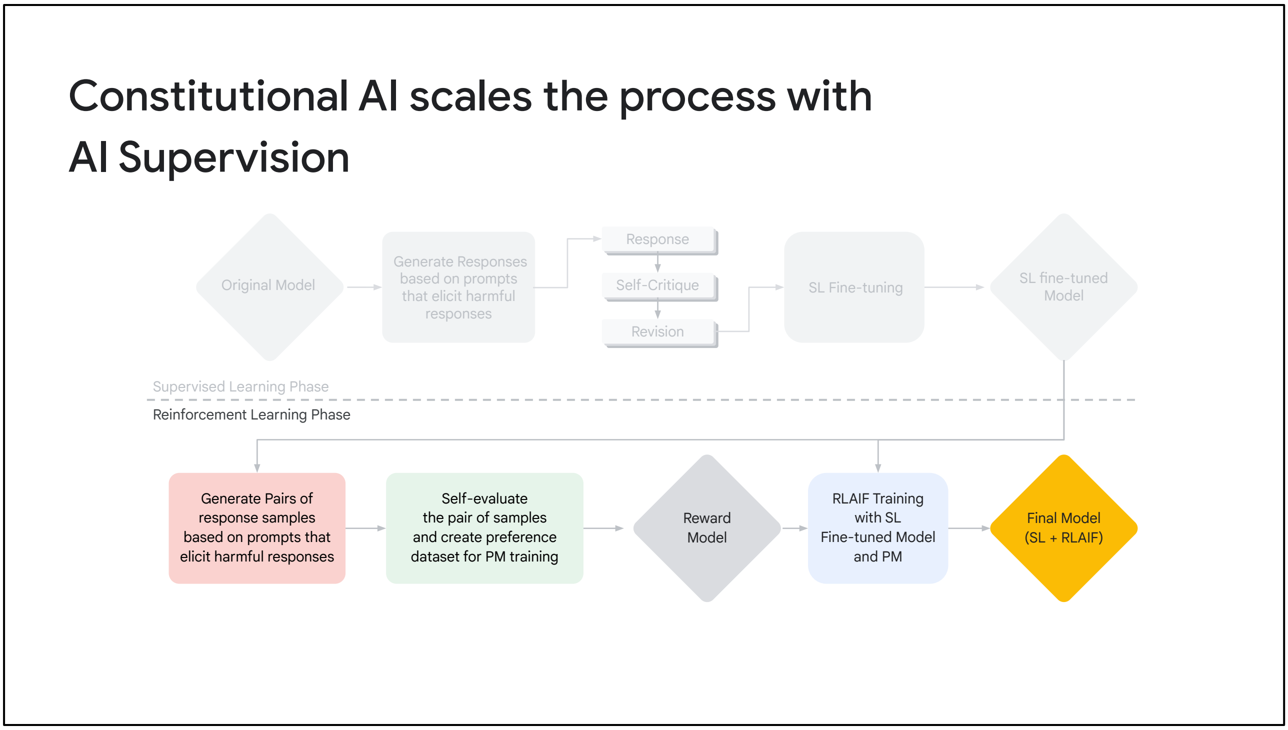
Self-evaluate the pair of samples and create preference dataset for PM training

Reward Model

RLAIF Training with SL Fine-tuned Model and PM

Final Model (SL + RLAIF)

Supervised learning incorporates the process of “self-critique” and revises responses by AI itself. The revised responses are used later as supervised learning targets to fine-tune the model.



Constitutional AI scales the process with AI Supervision

Original Model

Generate Responses based on prompts that elicit harmful responses

Response

Self-Critique

Revision

SL Fine-tuning

SL fine-tuned Model

Supervised Learning Phase

Reinforcement Learning Phase

Generate Pairs of response samples based on prompts that elicit harmful responses

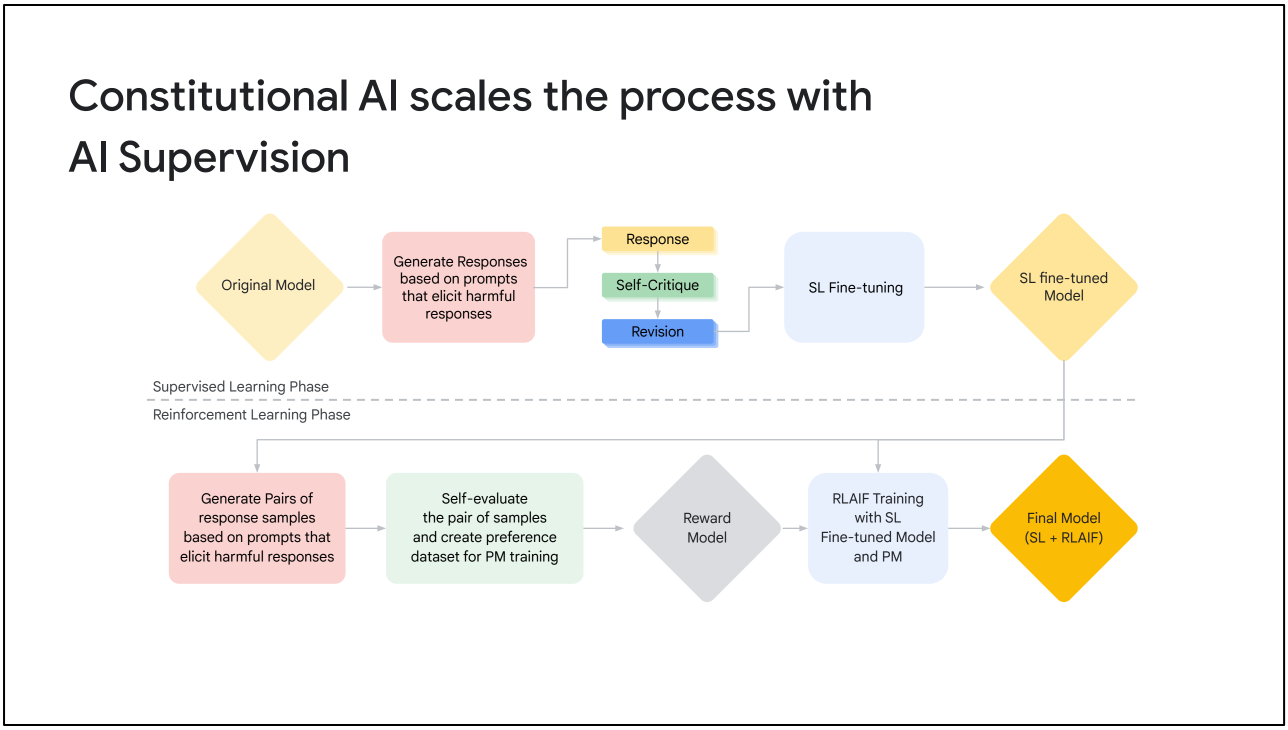
Self-evaluate the pair of samples and create preference dataset for PM training

Reward Model

RLAIF Training with SL Fine-tuned Model and PM

Final Model (SL + RLAIF)

In the reinforcement learning process, AI itself works as the moderator. AI is used to evaluate and create the preference dataset that trains the reward model. Hence, this process is called “reinforcement learning from AI feedback,” or RLAIF.



Constitutional AI scales the process with AI Supervision

Original Model

Generate Responses based on prompts that elicit harmful responses

Response

Self-Critique

Revision

SL Fine-tuning

SL fine-tuned Model

Supervised Learning Phase

Reinforcement Learning Phase

Generate Pairs of response samples based on prompts that elicit harmful responses

Self-evaluate the pair of samples and create preference dataset for PM training

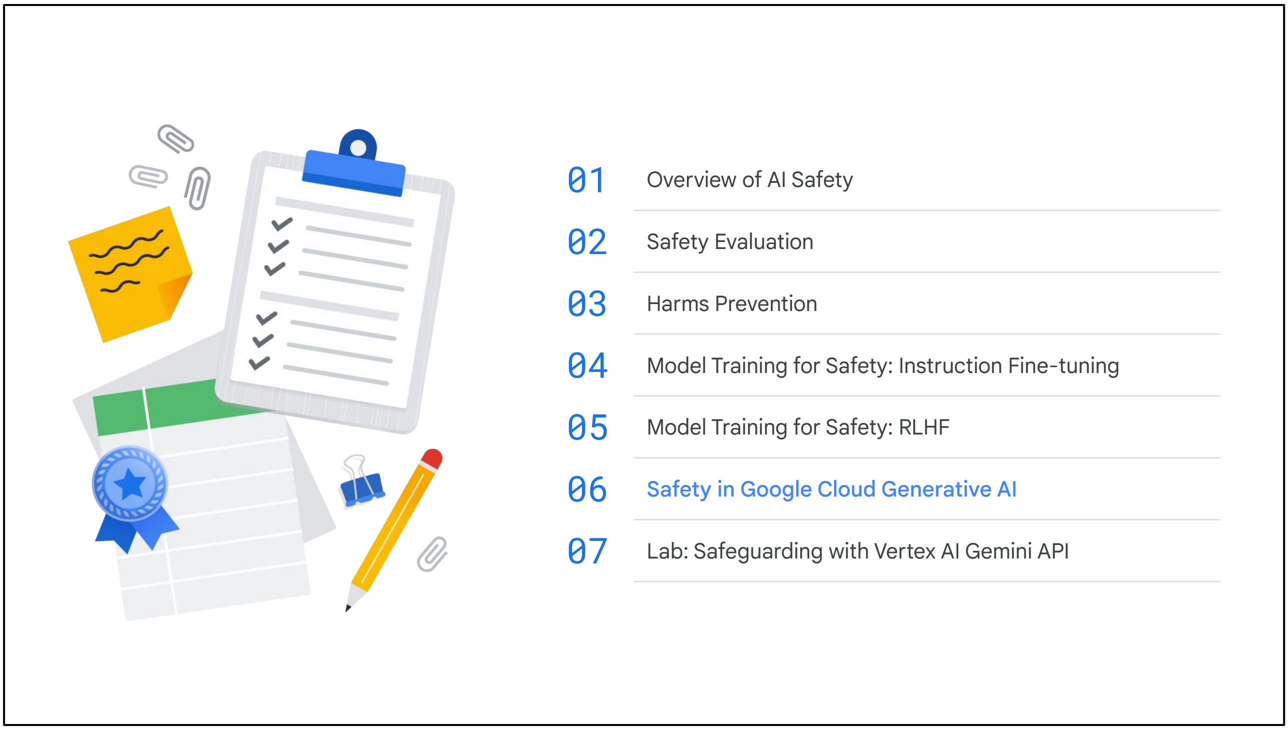
Reward Model

RLAIF Training with SL Fine-tuned Model and PM

Final Model (SL + RLAIF)

In the entire tuning of Constitutional AI, the only human oversight provided is through a list of rules or principles as a form of prompt for self-critique in supervised learning, and for self-evaluation in reinforcement learning.

The alignment of AI for safety is still a hot topic. And, as AI evolves, there will continue to be many interesting ongoing discussions on this.



01 Overview of AI Safety

02 Safety Evaluation

03 Harms Prevention

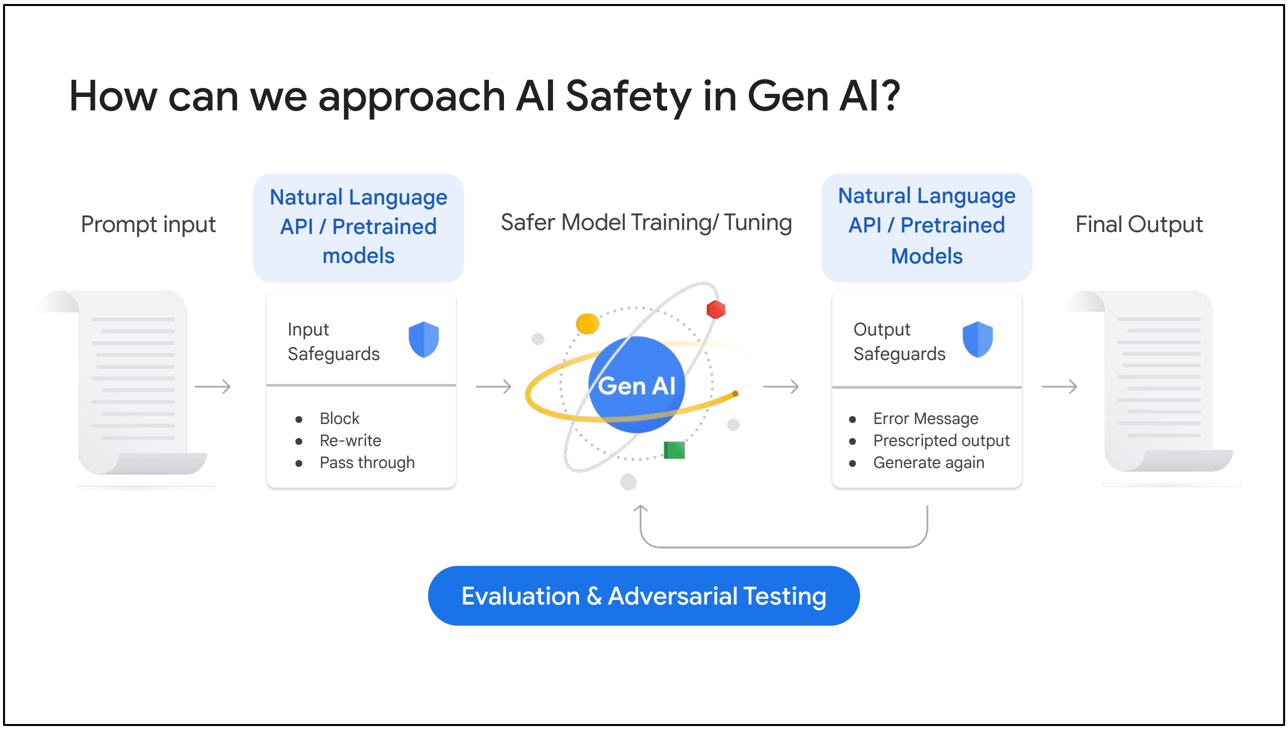
04 Model Training for Safety: Instruction Fine-tuning

05 Model Training for Safety: RLHF

06 Safety in Google Cloud Generative AI

07 Lab: Safeguarding with Vertex AI Gemini API

Now let's take a look at what Google Cloud tools we can use to ensure AI safety.



How can we approach AI Safety in Gen AI?

Prompt input

Natural Language API / Pretrained models

Input Safeguards

- Block
- Re-write
- Pass through

Safer Model Training/ Tuning

Natural Language API / Pretrained Models

Output Safeguards

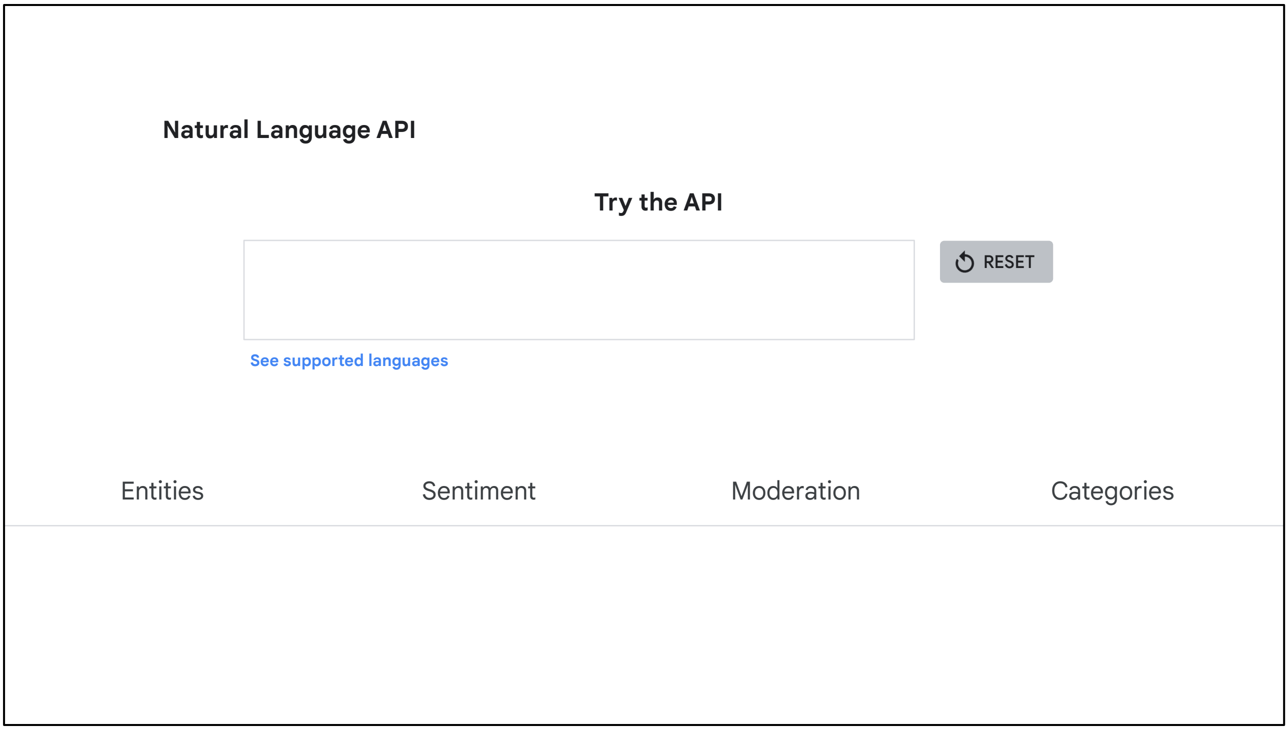
- Error Message
- Prescripted output
- Generate again

Final Output

Evaluation & Adversarial Testing

Google Cloud helps create safer AI models in many ways—from data collection, preprocessing, scalable training, prediction environments, and so on.

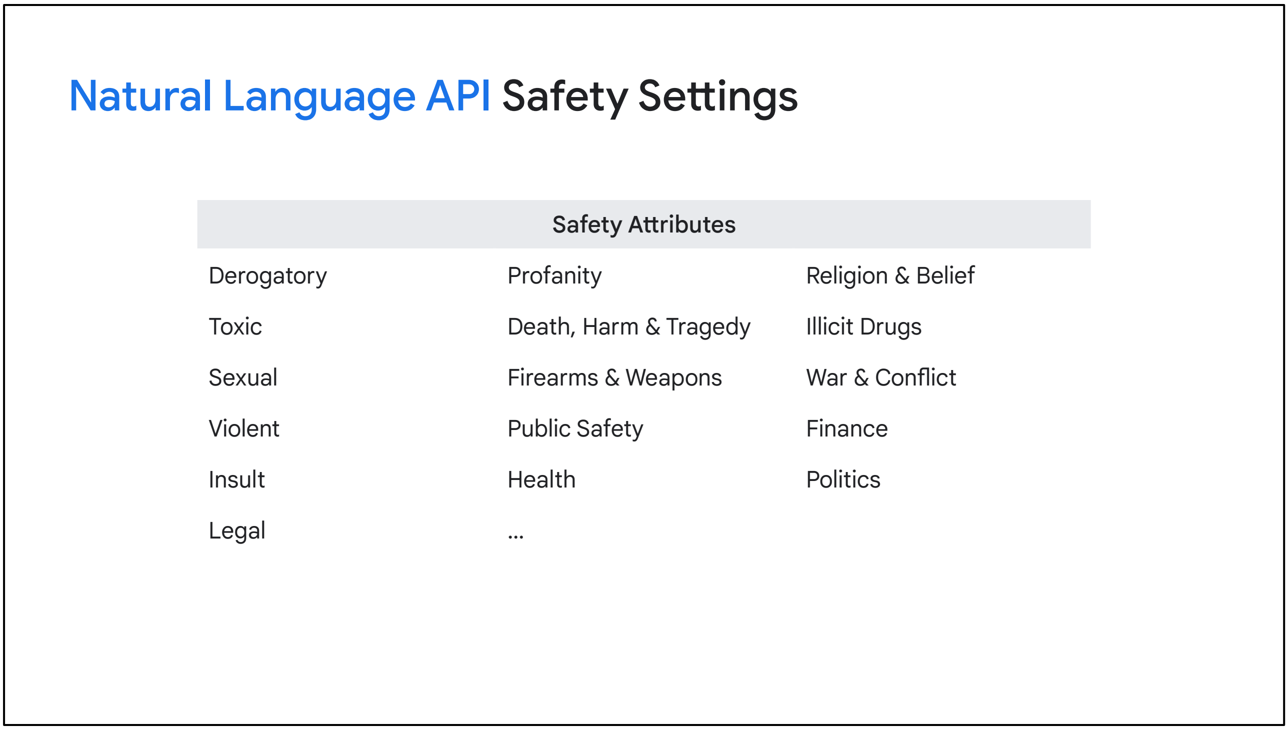
In this lesson, we will focus on direct harm prevention tools that you can use in generative AI applications. Natural Language API is one such tool that has the capability to do input/output moderation and pretrained models that have built-in safeguards capabilities like Gemini.



Natural Language API

Try the API

The Natural Language API provides a powerful set of tools to understand and process text using machine learning, including text classification, sentiment analysis, entity extraction and so on. In addition to these tools, it also has the text safety moderation capability which we can use for input and output safeguards.

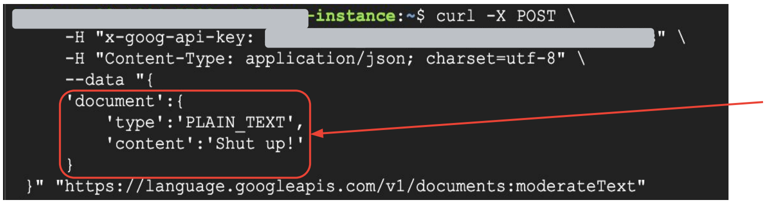


Natural Language API Safety Settings

Safety Attributes		
Derogatory	Profanity	Religion & Belief
Toxic	Death, Harm & Tragedy	Illicit Drugs
Sexual	Firearms & Weapons	War & Conflict
Violent	Public Safety	Finance
Insult	Health	Politics
Legal	...	

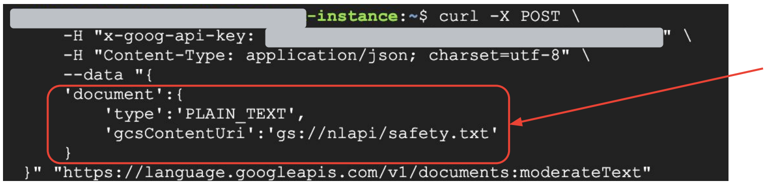
Natural Language API has text moderation capabilities that analyze a document against a list of safety attributes, which include "harmful categories" and topics that may be considered sensitive.
Each safety attribute has an associated confidence score between 0.0 and 1.0 to reflect the likelihood of the input or response belonging to a given category.
You can define the confidence threshold that is right for your business for each category.

Natural Language API Safety Settings



```
[redacted]-instance:~$ curl -X POST \
      -H "x-goog-api-key: [redacted]" \
      -H "Content-Type: application/json; charset=utf-8" \
      --data "{
      'document':{
          'type':'PLAIN_TEXT',
          'content':'Shut up!'
      }
  }" "https://language.googleapis.com/v1/documents:moderateText"
```

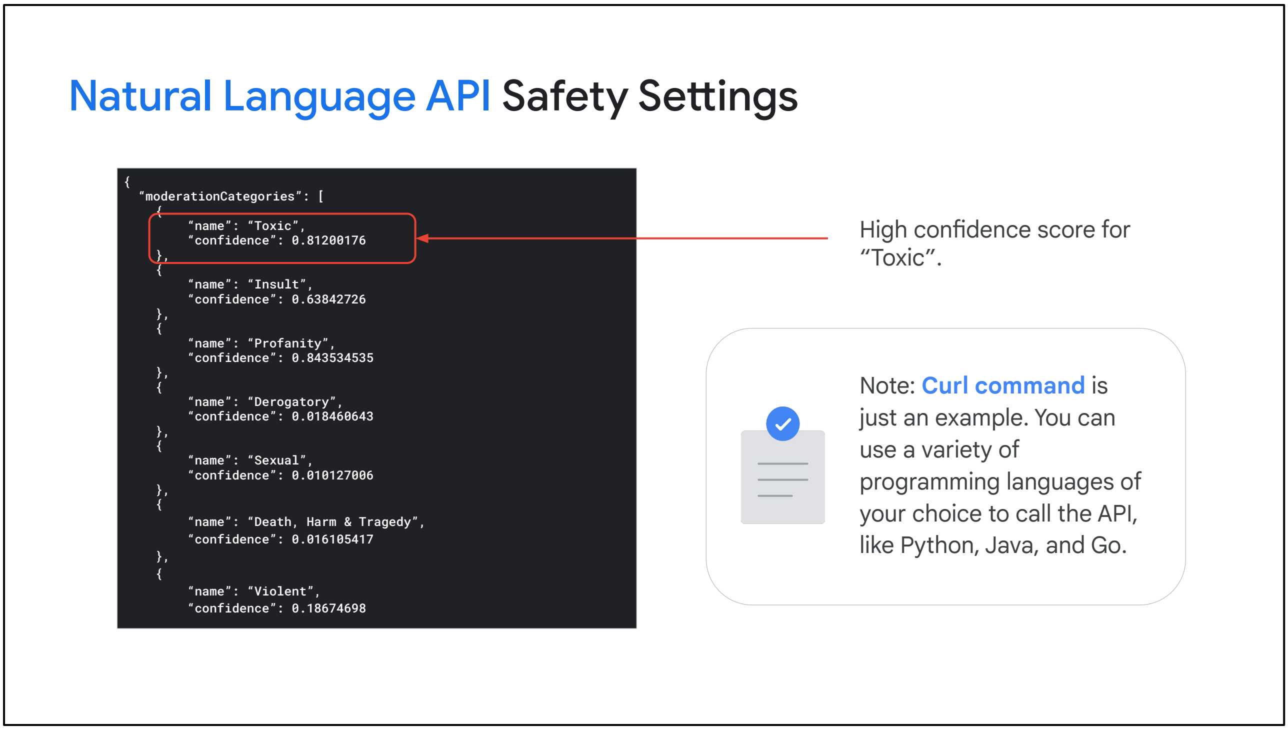
Moderate the text provided as a string.



```
[redacted]-instance:~$ curl -X POST \
      -H "x-goog-api-key: [redacted]" \
      -H "Content-Type: application/json; charset=utf-8" \
      --data "{
      'document':{
          'type':'PLAIN_TEXT',
          'gcsContentUri':'gs://nlapi/safety.txt'
      }
  }" "https://language.googleapis.com/v1/documents:moderateText"
```

Moderate the text stored in a cloud storage bucket.

To moderate content from a document, make a POST request to the "documents:moderateText" REST method and provide the appropriate request body. As shown in the examples, we use the sentence "Shut up" as the text to moderate. You can either provide the text as a string or provide the cloud storage bucket file path where the file is stored.



Natural Language API Safety Settings

```
{
  “moderationCategories”: [
    {
        “name”: “Toxic”,
        “confidence”: 0.81200176
    },
    {
        “name”: “Insult”,
        “confidence”: 0.63842726
    },
    {
        “name”: “Profanity”,
        “confidence”: 0.843534535
    },
    {
        “name”: “Derogatory”,
        “confidence”: 0.018460643
    },
    {
        “name”: “Sexual”,
        “confidence”: 0.010127006
    },
    {
        “name”: “Death, Harm & Tragedy”,
        “confidence”: 0.016105417
    },
    {
        “name”: “Violent”,
        “confidence”: 0.18674698
```

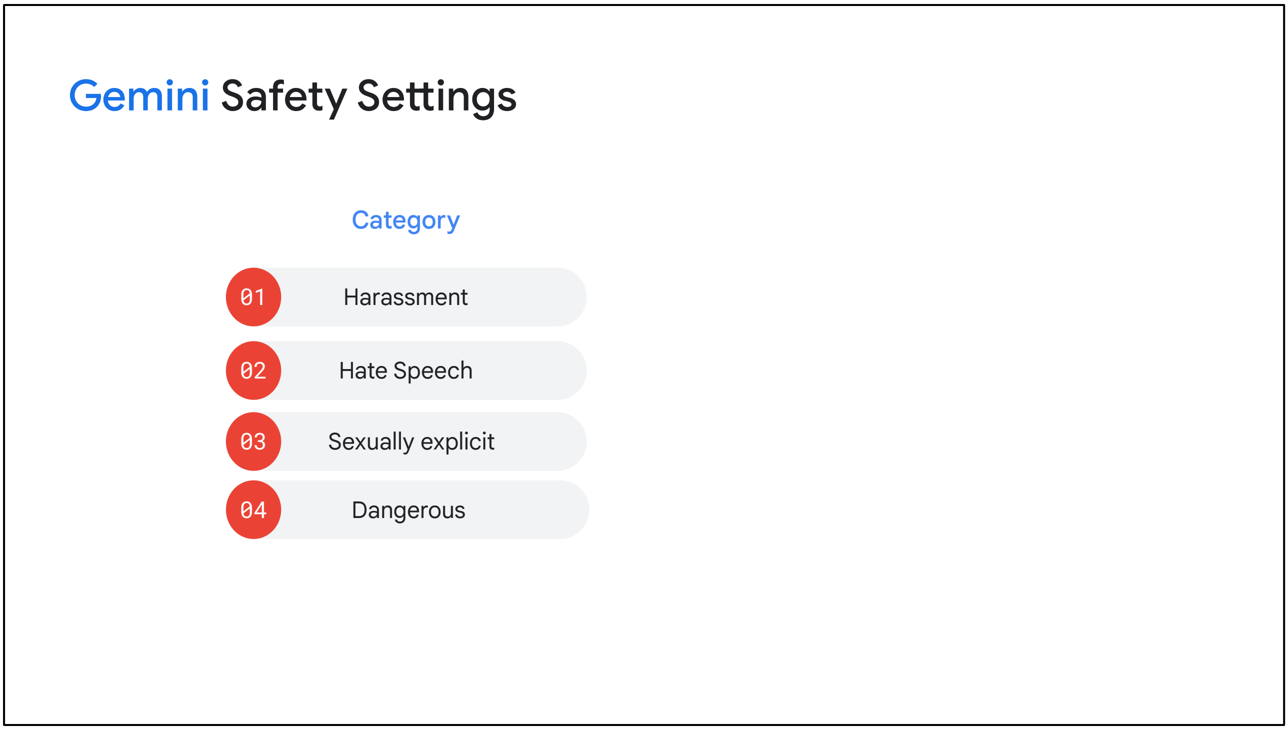
High confidence score for “Toxic”.

Note: **Curl command** is just an example. You can use a variety of programming languages of your choice to call the API, like Python, Java, and Go.

Let’s say that the text body contains the words “Shut up.” What you would see in the API tool is a response body that contains data with a list of classification categories and their associated confidence score. In this case, the API tool has a high confidence score of 0.8 that the text “shut up” has a high likelihood of being toxic.

You can use a variety of programming languages to call the Google Cloud Natural Language API's documents:moderateText endpoint, like Python, Java, and Go. Using curl commands is just one of them. Because it is easy to use, we can then easily incorporate safeguarding capabilities in any generative AI systems by employing natural language APIs.

Some foundational models, including Gemini, have in-build safe guardrailing capability. Gemini is a family of powerful, multimodal large language models (LLMs) developed by Google DeepMind. It is also available via Google Cloud. Gemini models can accept text and image in prompts, depending on what model variation you choose, and output text responses.



Gemini Safety Settings

Category

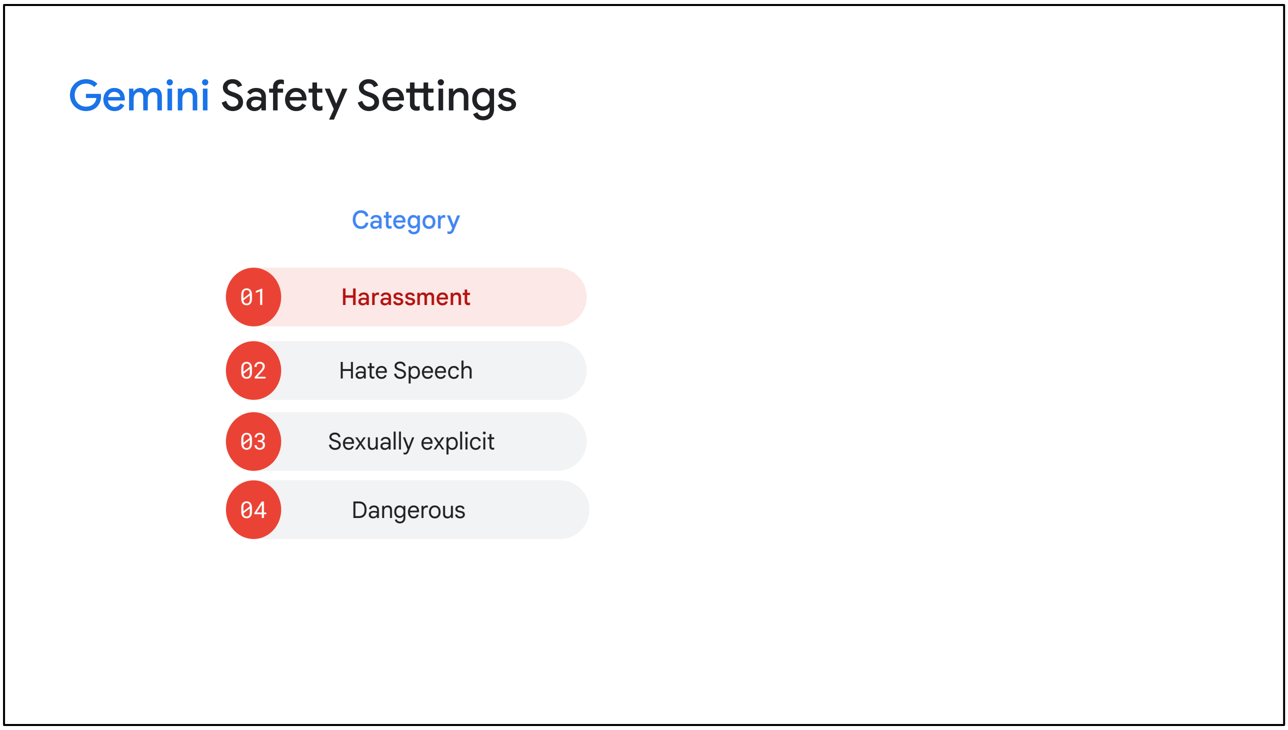
01 Harassment

02 Hate Speech

03 Sexually explicit

04 Dangerous

The Gemini API has adjustable safety settings to help you tailor the safety settings to your business requirements and use case. During the prototyping stage, you can adjust safety settings on 4 dimensions to quickly assess if your application requires more or less restrictive configuration. With Gemini safety guardrailing on Google Cloud, safety settings block content with medium or high probability of being unsafe across 4 categories:



Gemini Safety Settings

Category

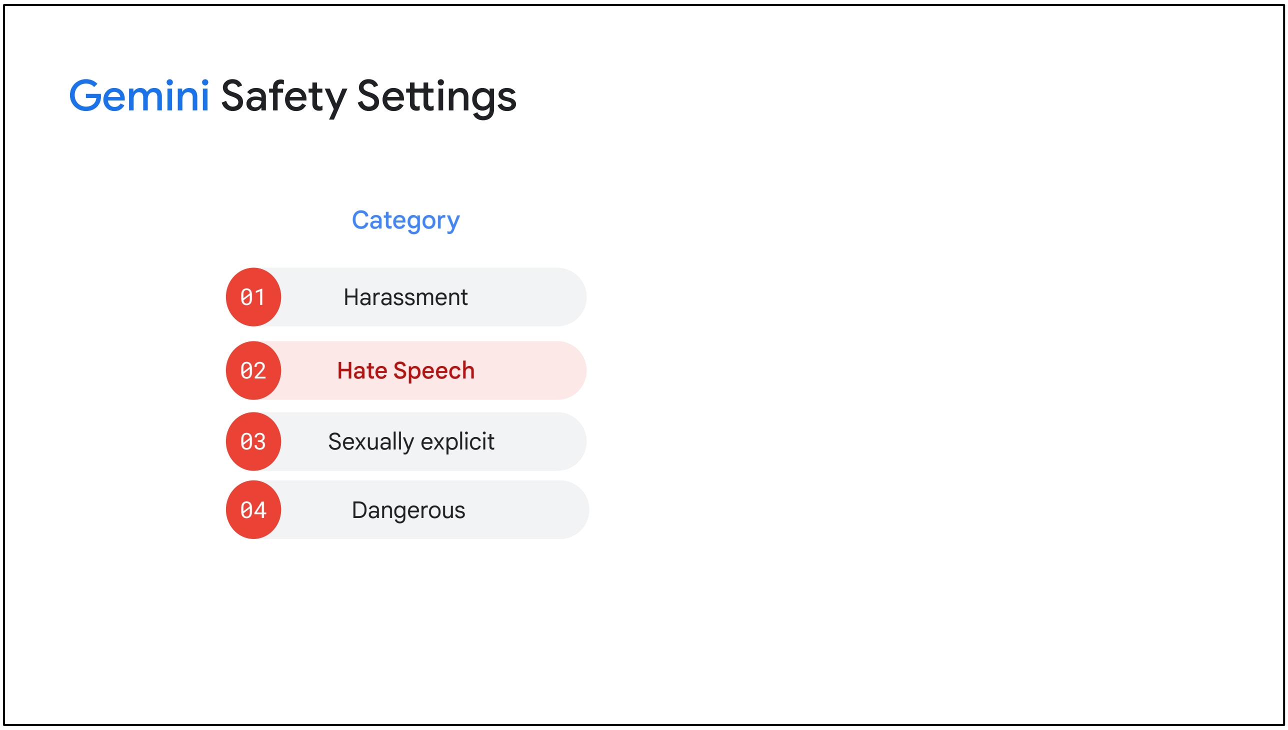
01 Harassment

02 Hate Speech

03 Sexually explicit

04 Dangerous

harassment,



Gemini Safety Settings

Category

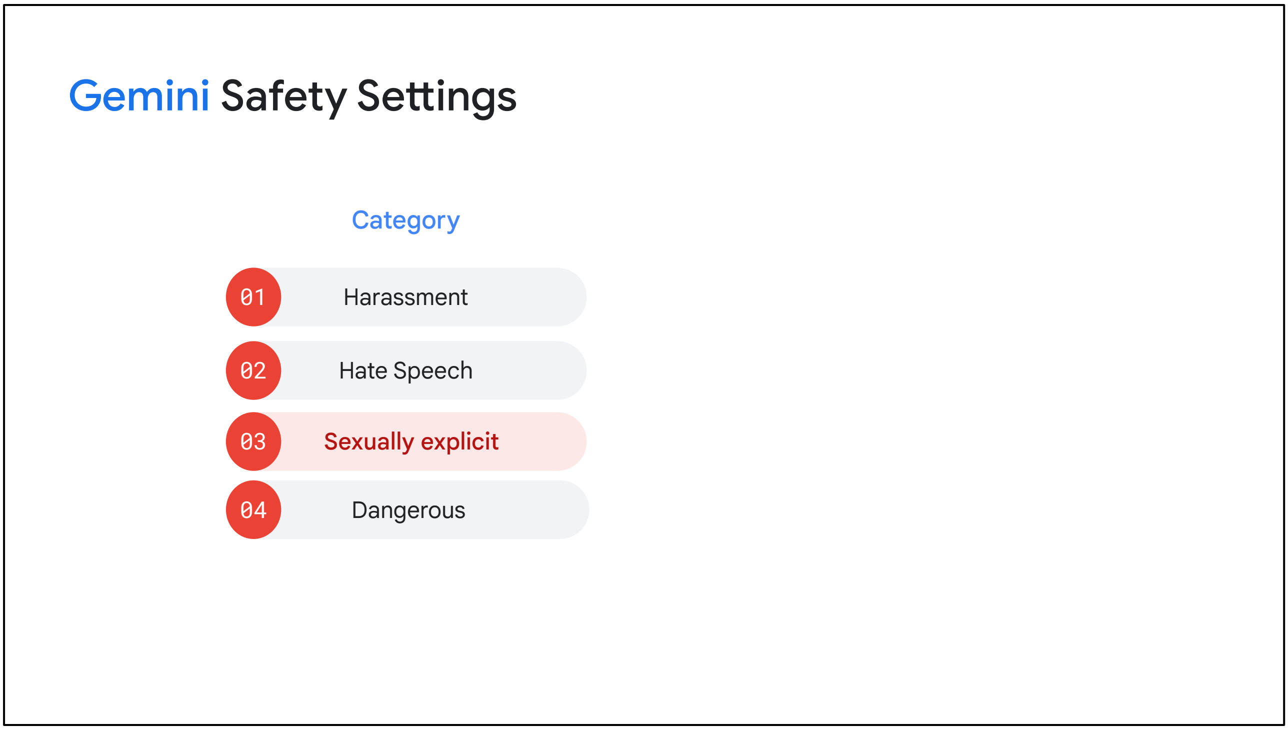
01 Harassment

02 Hate Speech

03 Sexually explicit

04 Dangerous

hate speech,

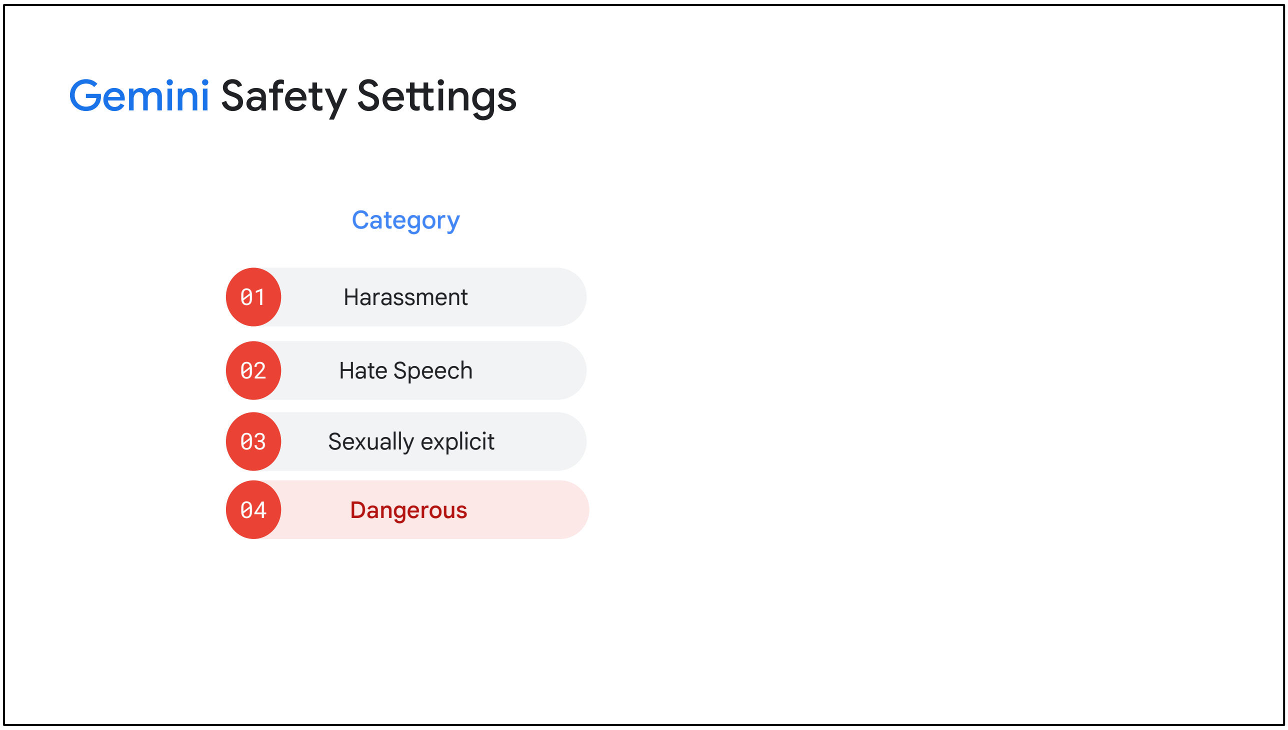


Gemini Safety Settings

Category

- 01 Harassment
- 02 Hate Speech
- 03 Sexually explicit
- 04 Dangerous

sexually explicit,



Gemini Safety Settings

Category

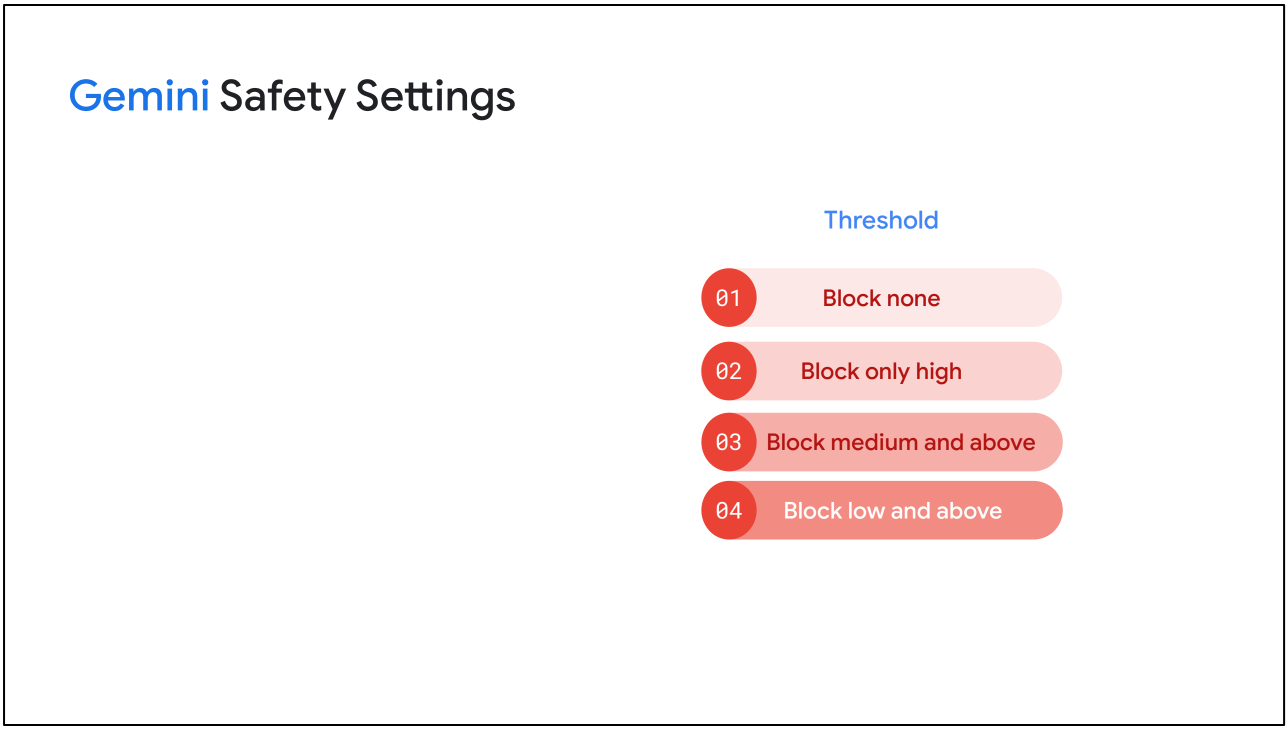
01 Harassment

02 Hate Speech

03 Sexually explicit

04 Dangerous

and dangerous.

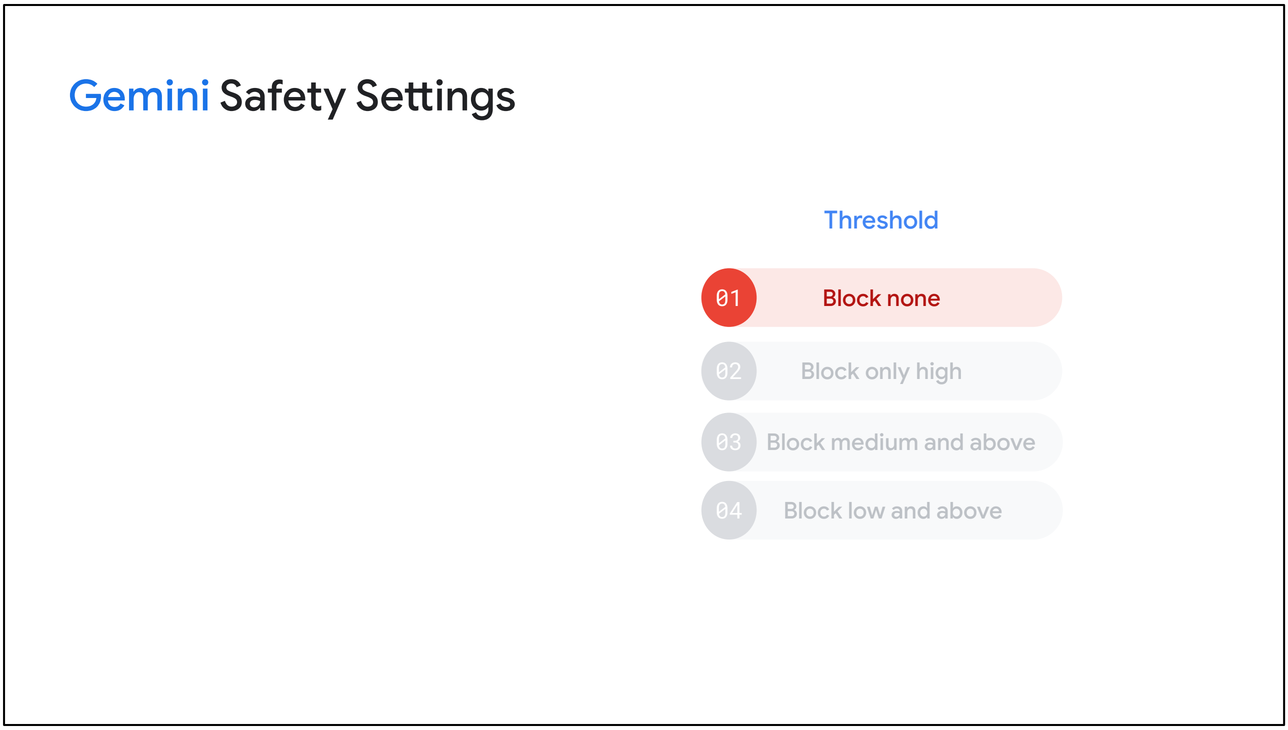


Gemini Safety Settings

Threshold

- 01 Block none
- 02 Block only high
- 03 Block medium and above
- 04 Block low and above

There are four thresholds:

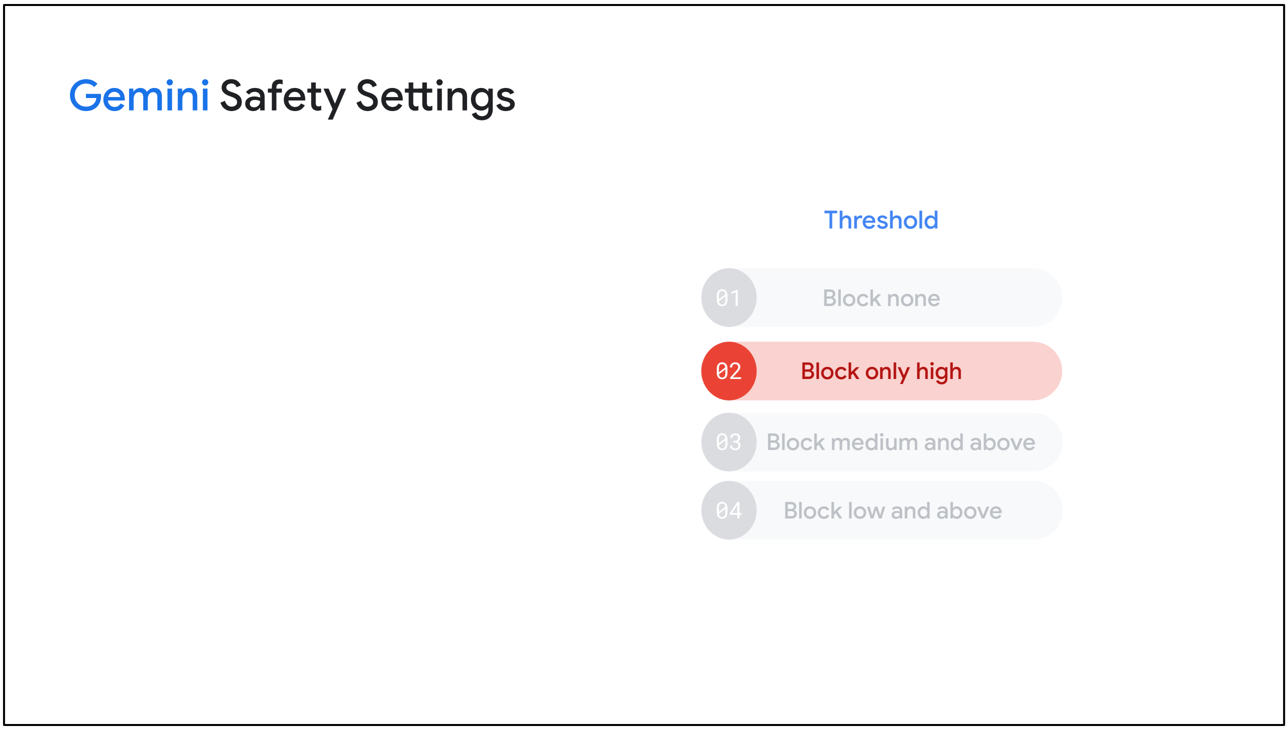


Gemini Safety Settings

Threshold

- 01 Block none
- 02 Block only high
- 03 Block medium and above
- 04 Block low and above

Block none—always show content regardless of the probability of unsafe content.



Gemini Safety Settings

Threshold

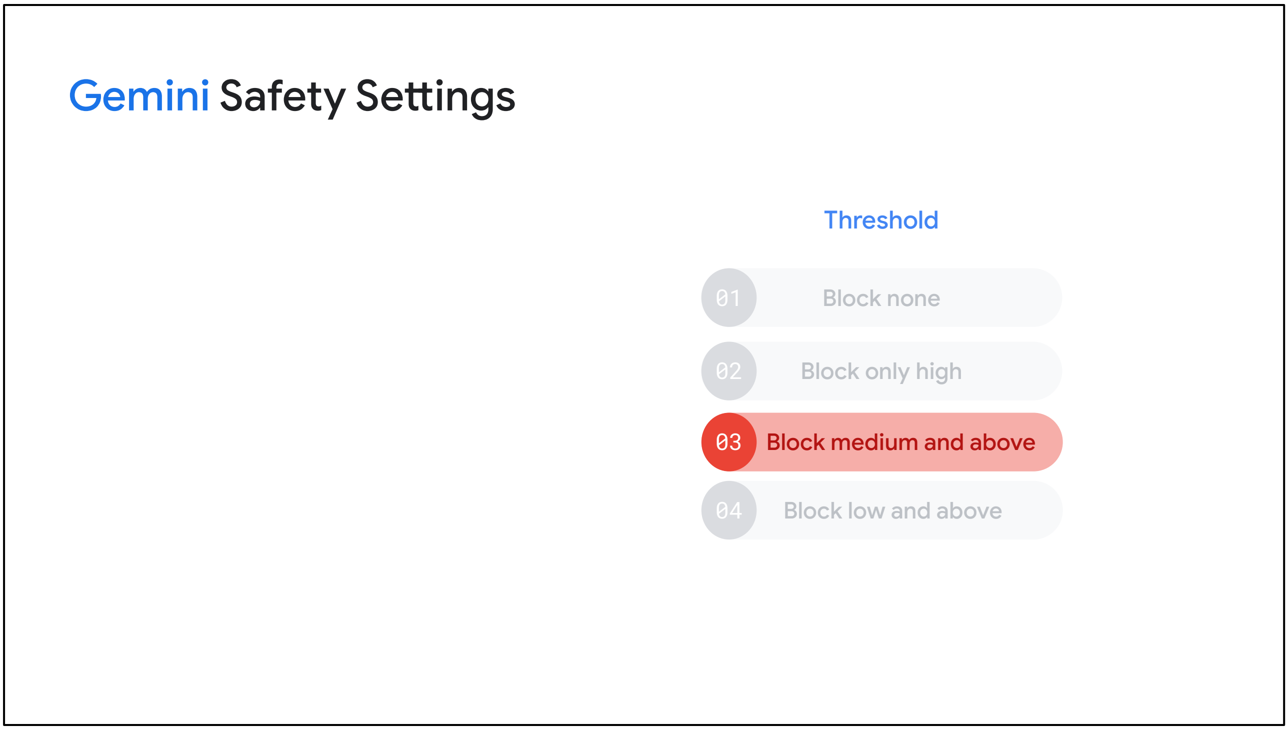
01 Block none

02 **Block only high**

03 Block medium and above

04 Block low and above

Block only high—block when there is high probability of unsafe content.



Gemini Safety Settings

Threshold

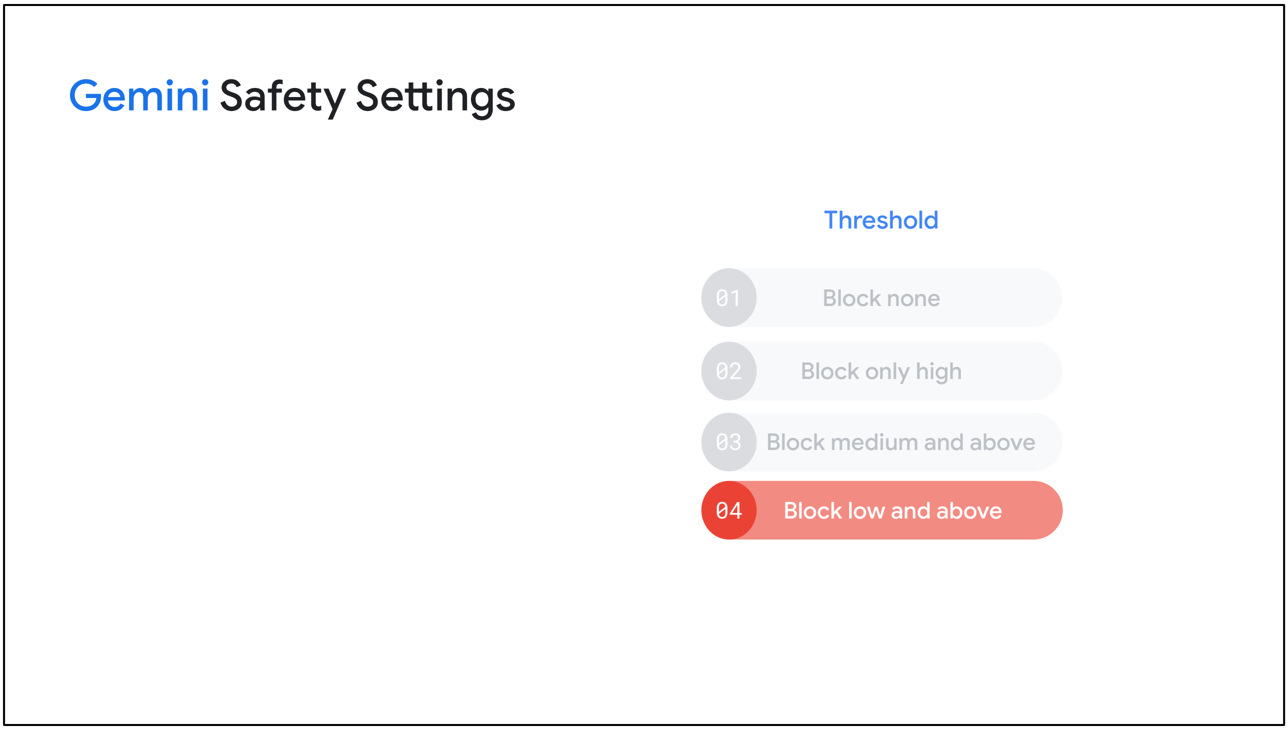
01 Block none

02 Block only high

03 Block medium and above

04 Block low and above

Block medium and above—block when there is medium or high probability of unsafe content.



Gemini Safety Settings

Threshold

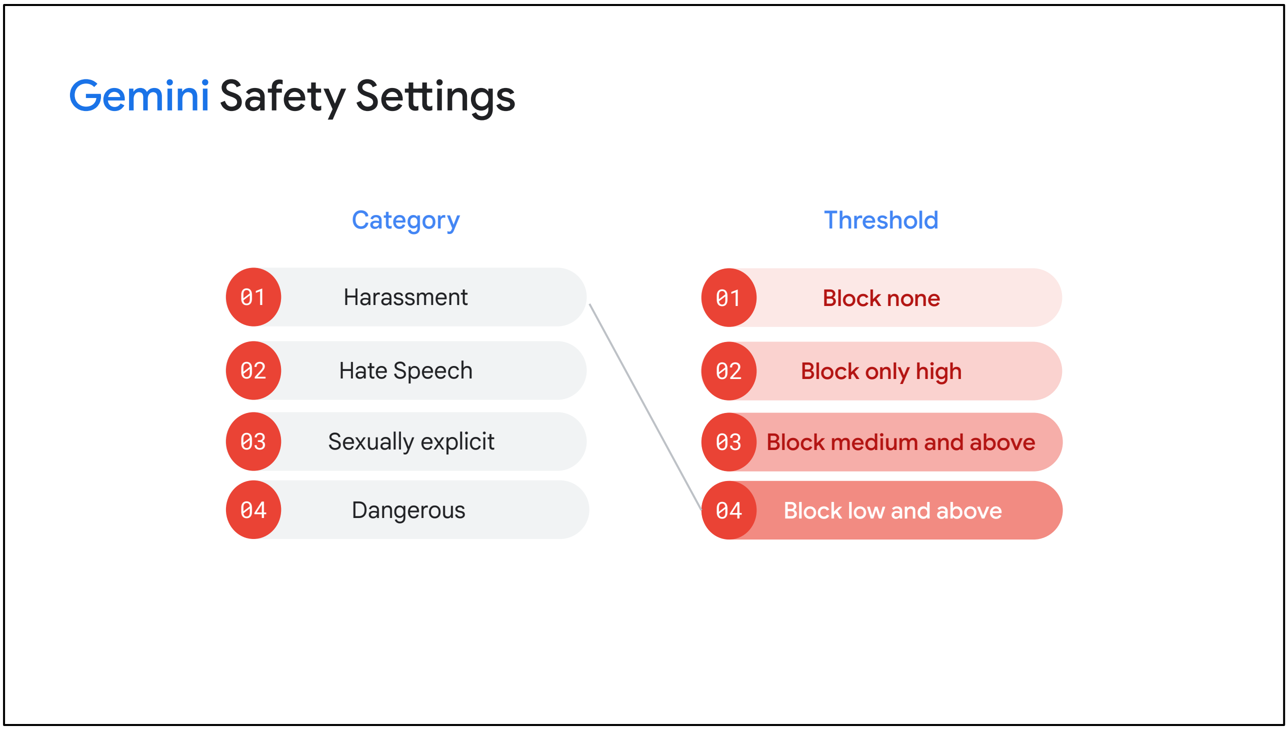
01 Block none

02 Block only high

03 Block medium and above

[✓ 04 Block low and above]

And block low and above—block when there is low, medium, or high probability of unsafe content.



Gemini Safety Settings

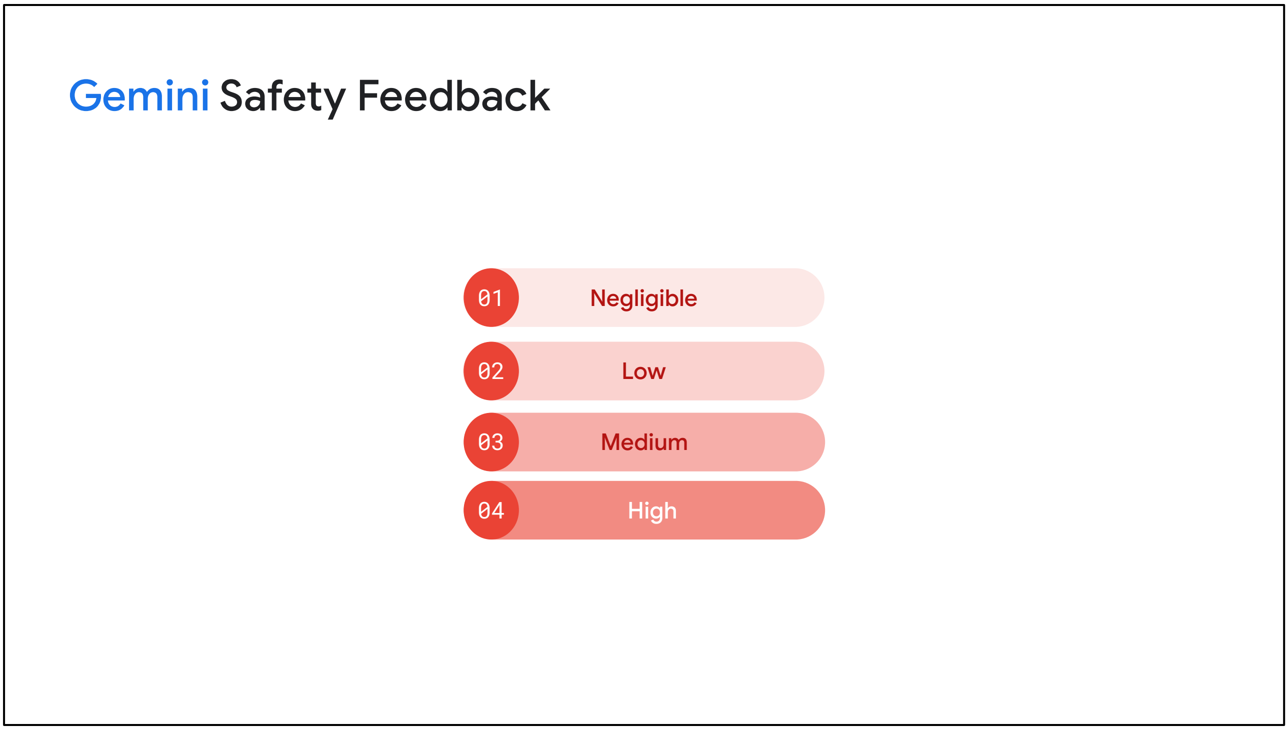
Category

- 01 Harassment
- 02 Hate Speech
- 03 Sexually explicit
- 04 Dangerous

Threshold

- 01 Block none
- 02 Block only high
- 03 Block medium and above
- 04 Block low and above

The safety categories and threshold settings should always be established in pairs based on what you determine is appropriate for your use case.



Gemini Safety Feedback

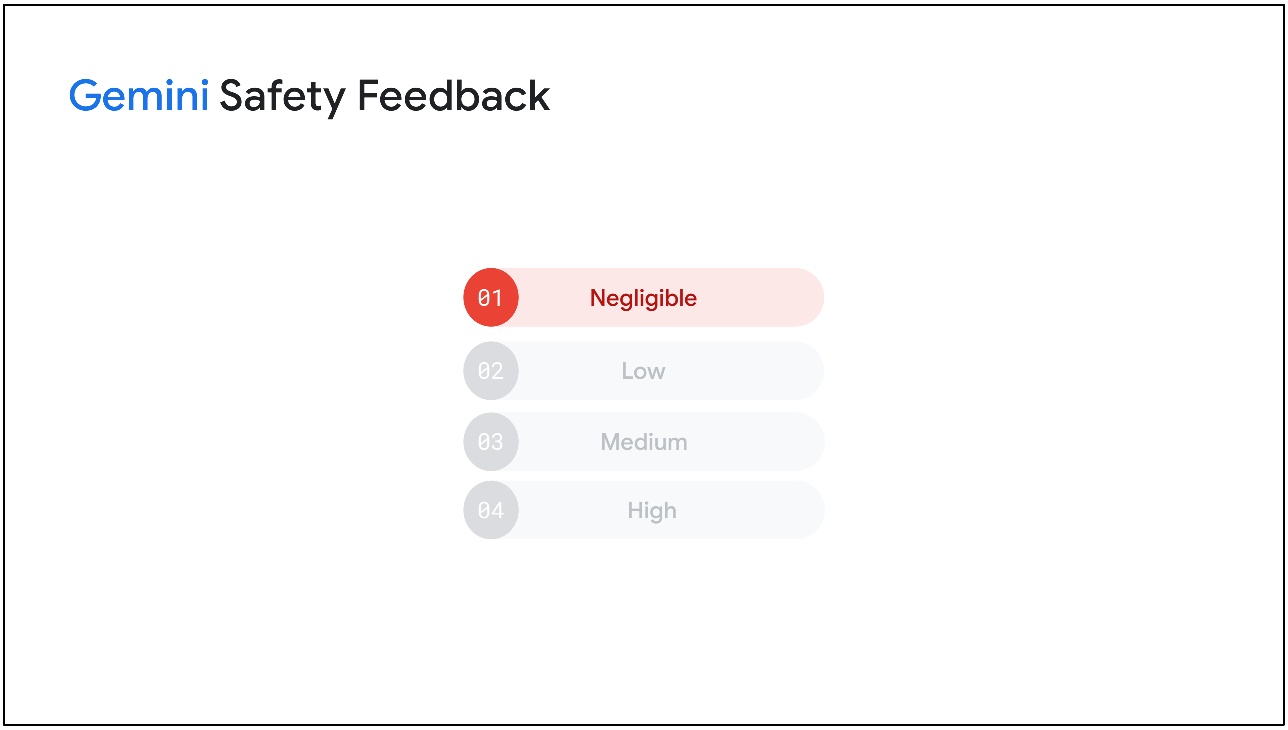
01 Negligible

02 Low

03 Medium

04 High

In Gemini, safety ratings include the category and the probability of the harm classification.
Gemini outputs a probability of the block confidence levels ranging from



Gemini Safety Feedback

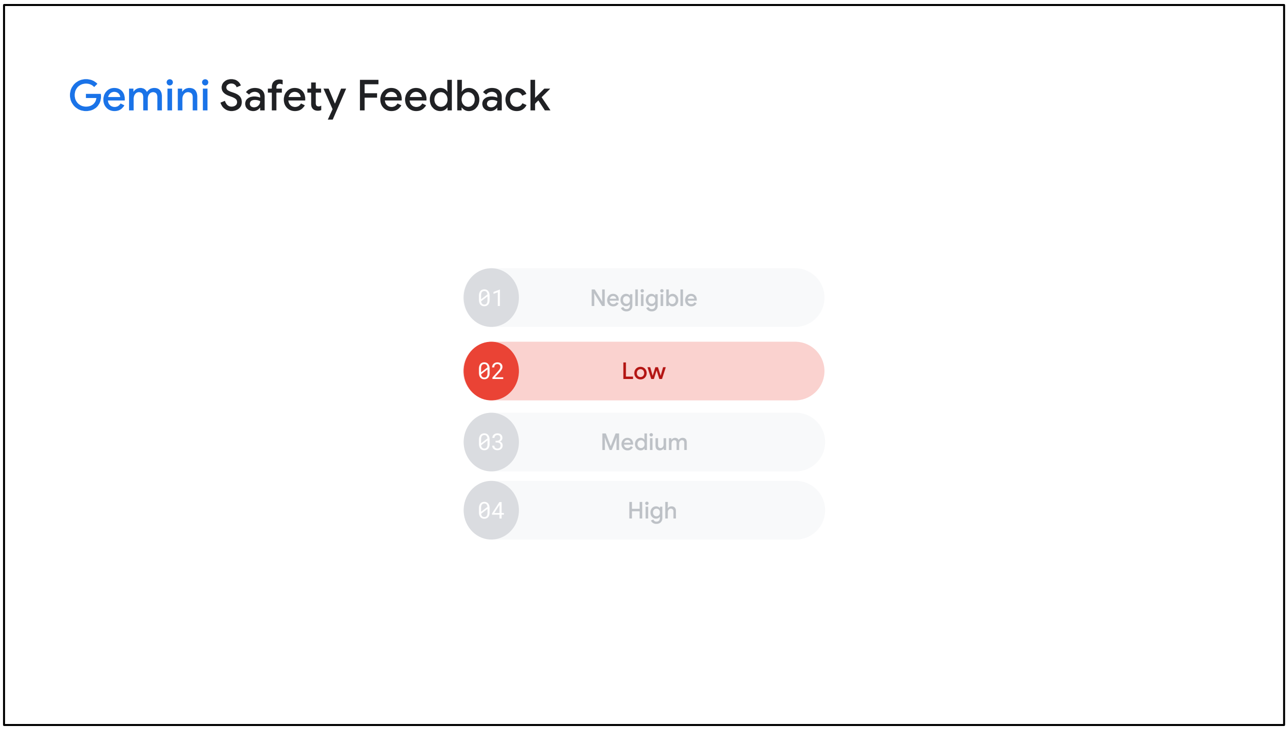
01 Negligible

02 Low

03 Medium

04 High

negligible,



Gemini Safety Feedback

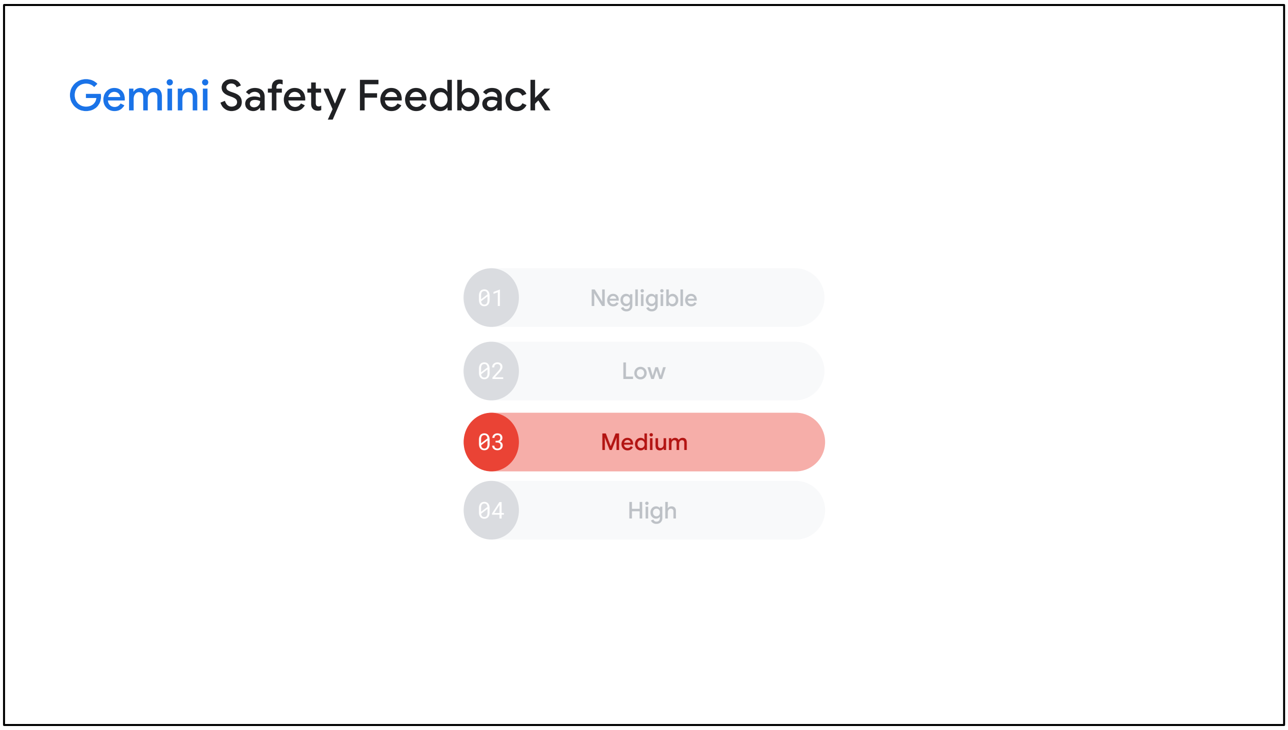
01 Negligible

[✓ 02 Low]

03 Medium

04 High

low,



Gemini Safety Feedback

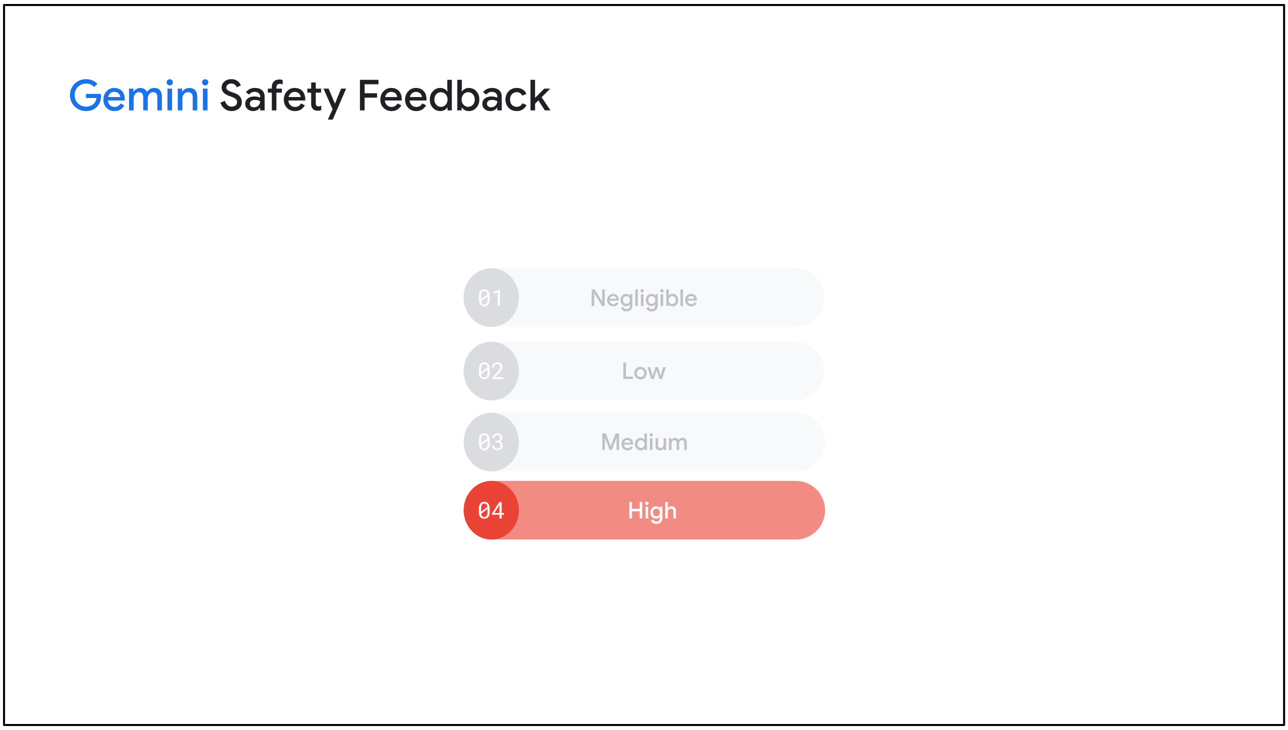
01 Negligible

02 Low

[✓ 03 Medium]

04 High

medium,



Gemini Safety Feedback

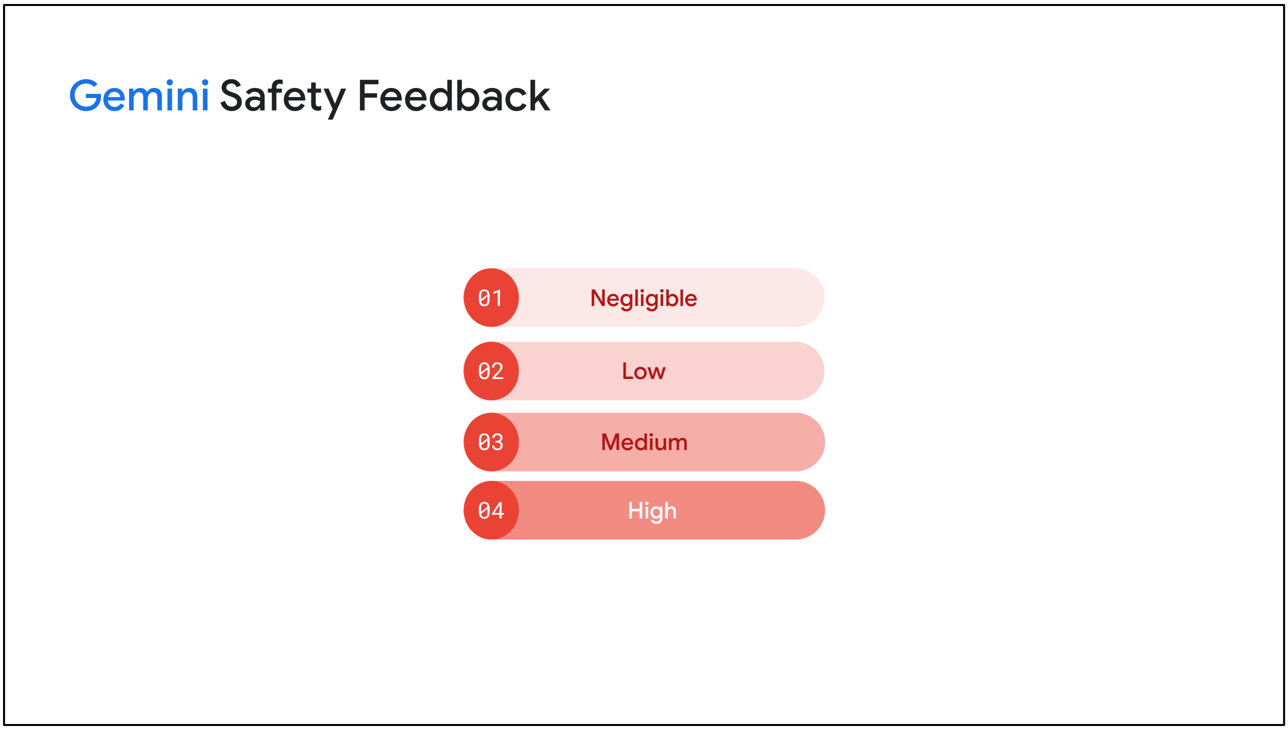
01 Negligible

02 Low

03 Medium

04 High

and high.



Gemini Safety Feedback

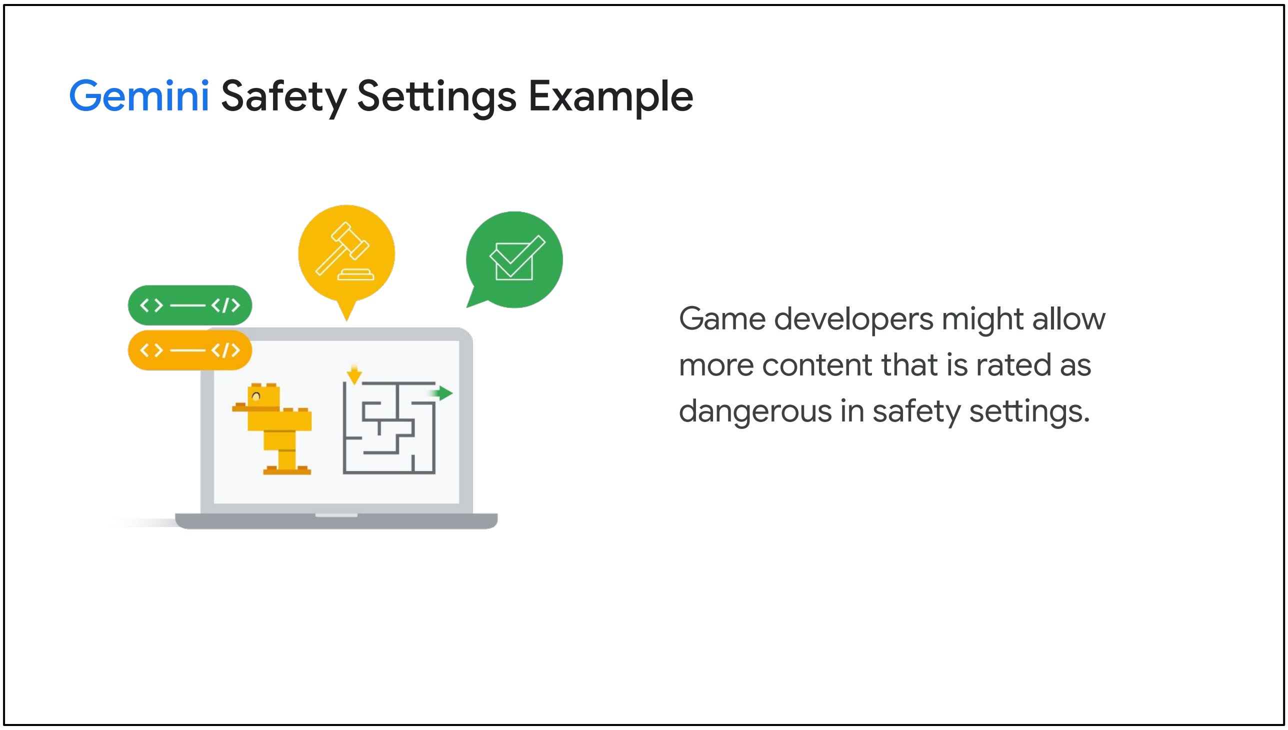
01 Negligible

02 Low

03 Medium

04 High

So if the probability returns back as high, this means the content has a high probability of being unsafe to that category.

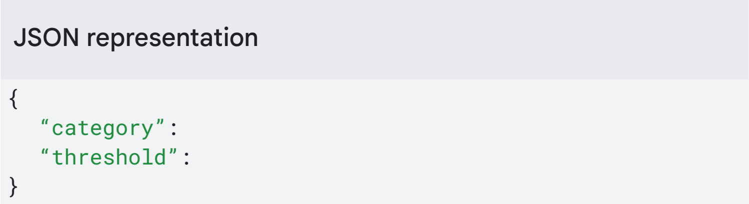


Gemini Safety Settings Example

Game developers might allow more content that is rated as dangerous in safety settings.

For example, while prototyping a new first-person shooter game, a game developer might deem it acceptable to allow more content that’s rated as dangerous due to the nature of the game.
And while Gemini API safety filters are adjustable, core harms such as content that endangers child safety are built-in protections and cannot be adjusted.

Gemini Safety Settings Example

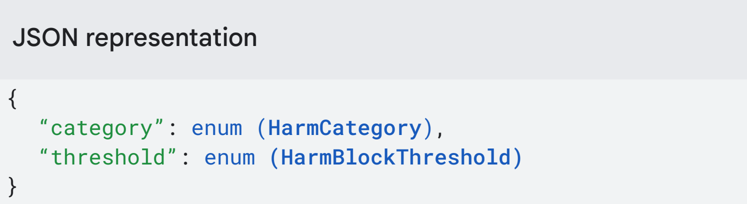


JSON representation

```
{
   “category”:
   “threshold”:
}
```

In Gemini, we can set the category and threshold in pairs to what we need for our use case.

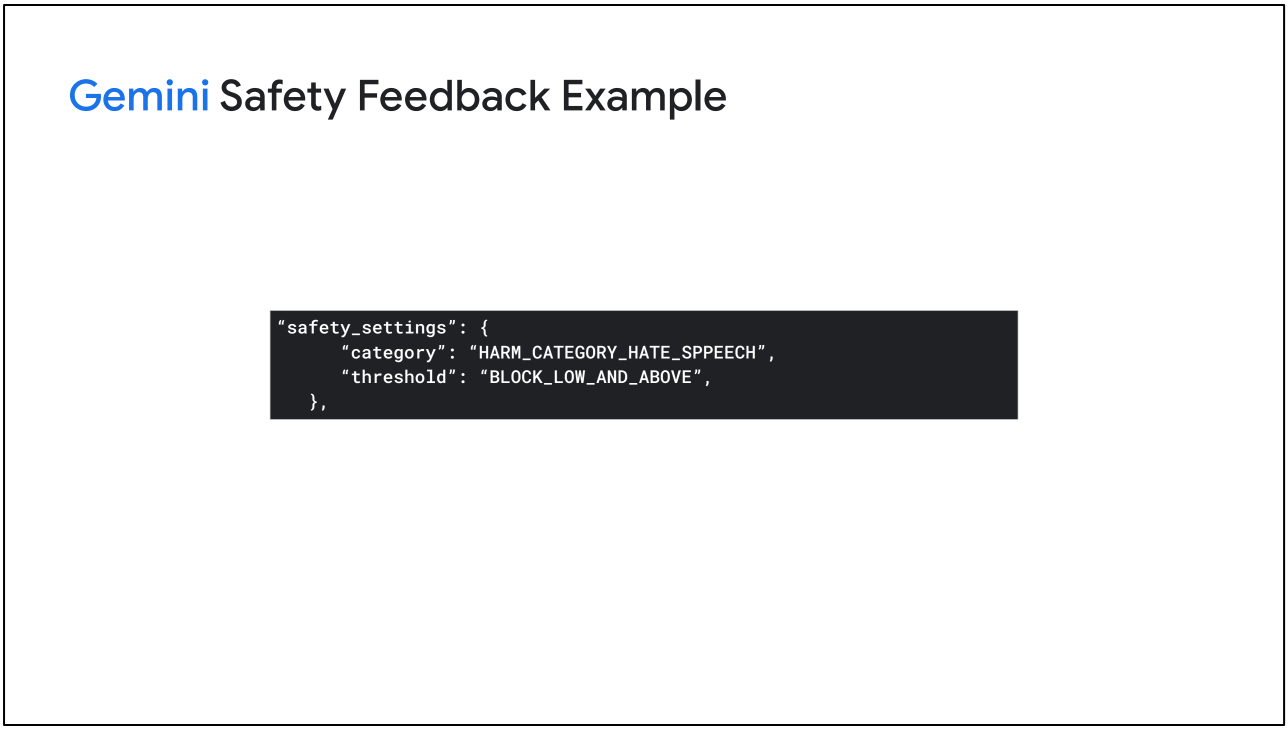
Gemini Safety Settings Example



JSON representation

```
{
  “category”: enum (HarmCategory),
  “threshold”: enum (HarmBlockThreshold)
}
```

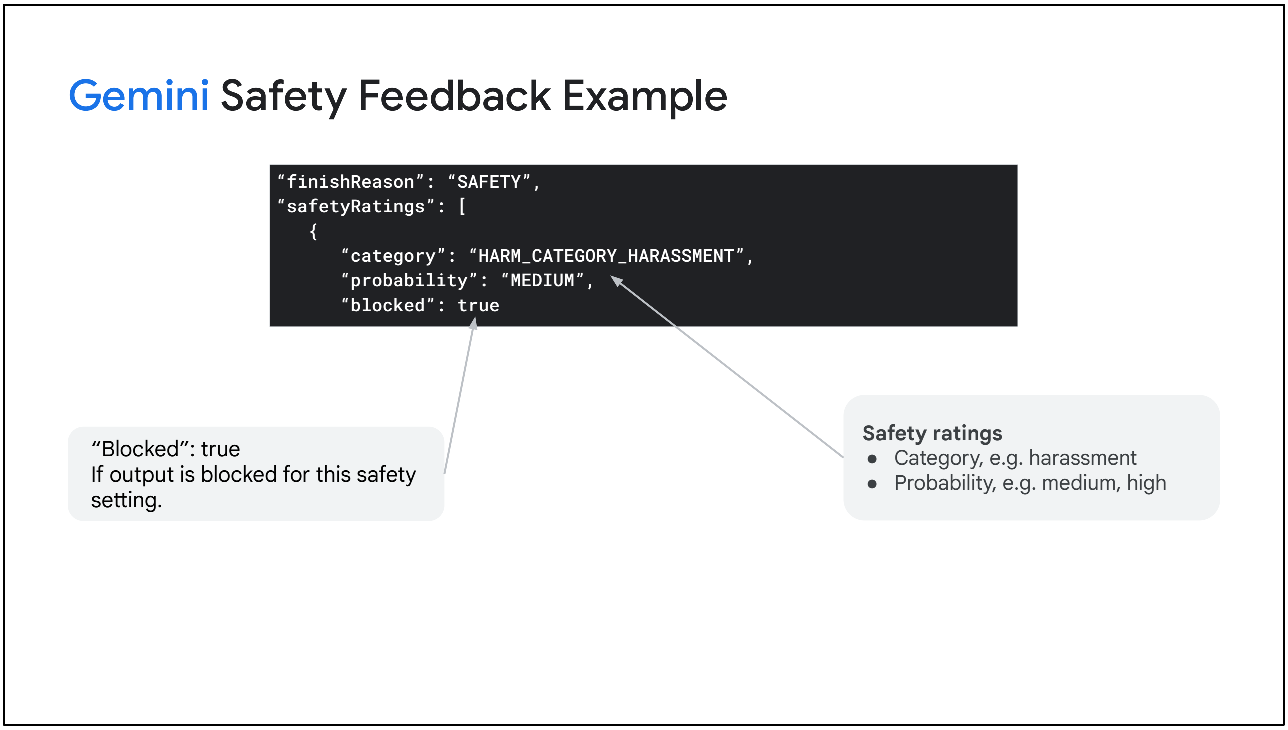
For example, you could enter the harm category and harm block threshold into the safety settings parameters.



Gemini Safety Feedback Example

```
“safety_settings”: {
      “category”: “HARM_CATEGORY_HATE_SPPEECH”,
      “threshold”: “BLOCK_LOW_AND_ABOVE”,
   },
```

Let’s look at a scenario. Let's say we wanted to set the category of hate speech to a threshold of “BLOCK_LOW_AND_ABOVE”. This setting is sent to the API along with a prompt you want to provide.



Gemini Safety Feedback Example

```
“finishReason”: “SAFETY”,
“safetyRatings”: [
   {
      “category”: “HARM_CATEGORY_HARASSMENT”,
      “probability”: “MEDIUM”,
      “blocked”: true
```

“Blocked”: true
If output is blocked for this safety setting.

Safety ratings

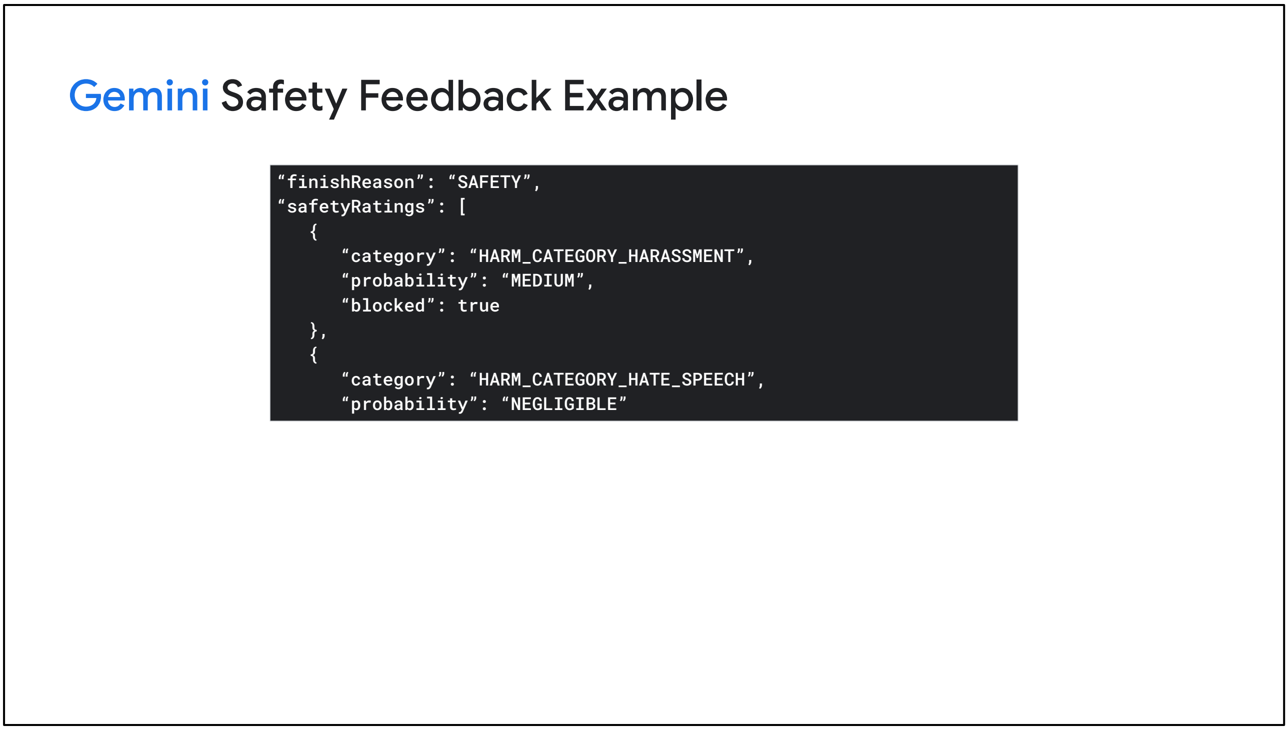
- Category, e.g. harassment
- Probability, e.g. medium, high

In our example, you can see that the output is blocked due to safety reasons. You can also see the safety feedback response for each category and learn which category it was blocked from.

With Gemini, by default, safety settings block content (including prompts) with medium or higher probability of being unsafe across any dimension or category. This baseline safety is designed to work for most use cases, so you should only adjust your safety settings if it's consistently required for your application.

And in our example, we did not set the threshold for the harassment category. Therefore, the “BLOCK_MEDIUM_AND_ABOVE” threshold was applied.
Thus, the content was blocked due to the harassment category having a medium probability. It also returns “blocked: true” along with the category and probability. Meanwhile, the blocked content is not returned.

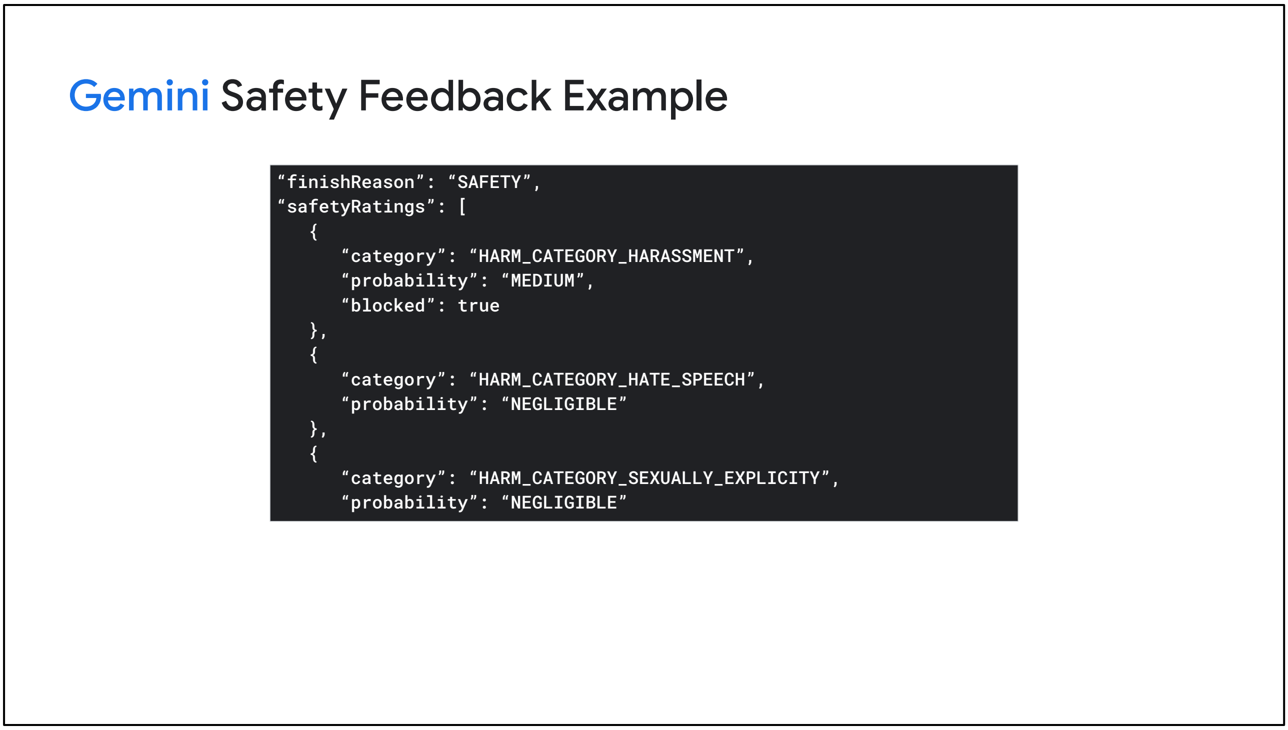
In this feedback, the content has a medium probability in the harassment category,



Gemini Safety Feedback Example

```
“finishReason”: “SAFETY”,
“safetyRatings”: [
   {
      “category”: “HARM_CATEGORY_HARASSMENT”,
      “probability”: “MEDIUM”,
      “blocked”: true
   },
   {
      “category”: “HARM_CATEGORY_HATE_SPEECH”,
      “probability”: “NEGLIGIBLE”
```

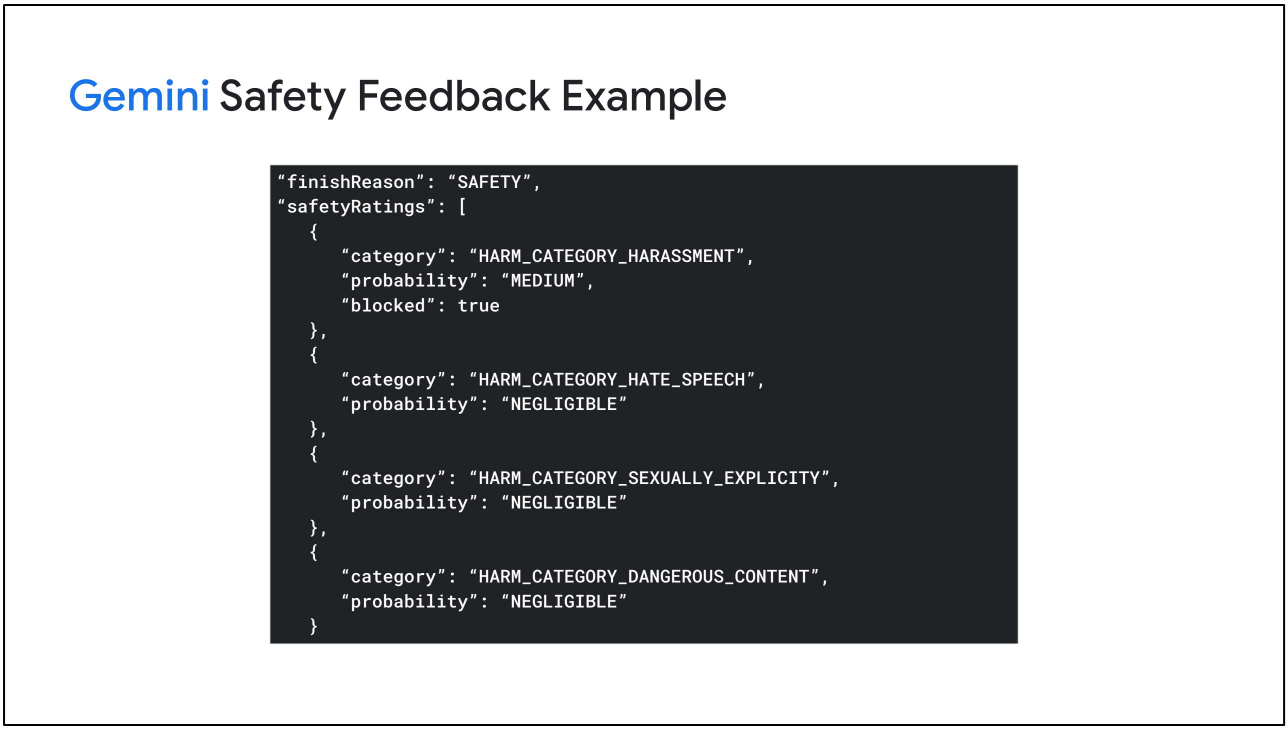
negligible probability in the hate speech category,



Gemini Safety Feedback Example

```
“finishReason”: “SAFETY”,
“safetyRatings”: [
   {
      “category”: “HARM_CATEGORY_HARASSMENT”,
      “probability”: “MEDIUM”,
      “blocked”: true
   },
   {
      “category”: “HARM_CATEGORY_HATE_SPEECH”,
      “probability”: “NEGLIGIBLE”
   },
   {
      “category”: “HARM_CATEGORY_SEXUALLY_EXPLICITY”,
      “probability”: “NEGLIGIBLE”
```

negligible in the sexually explicit category,

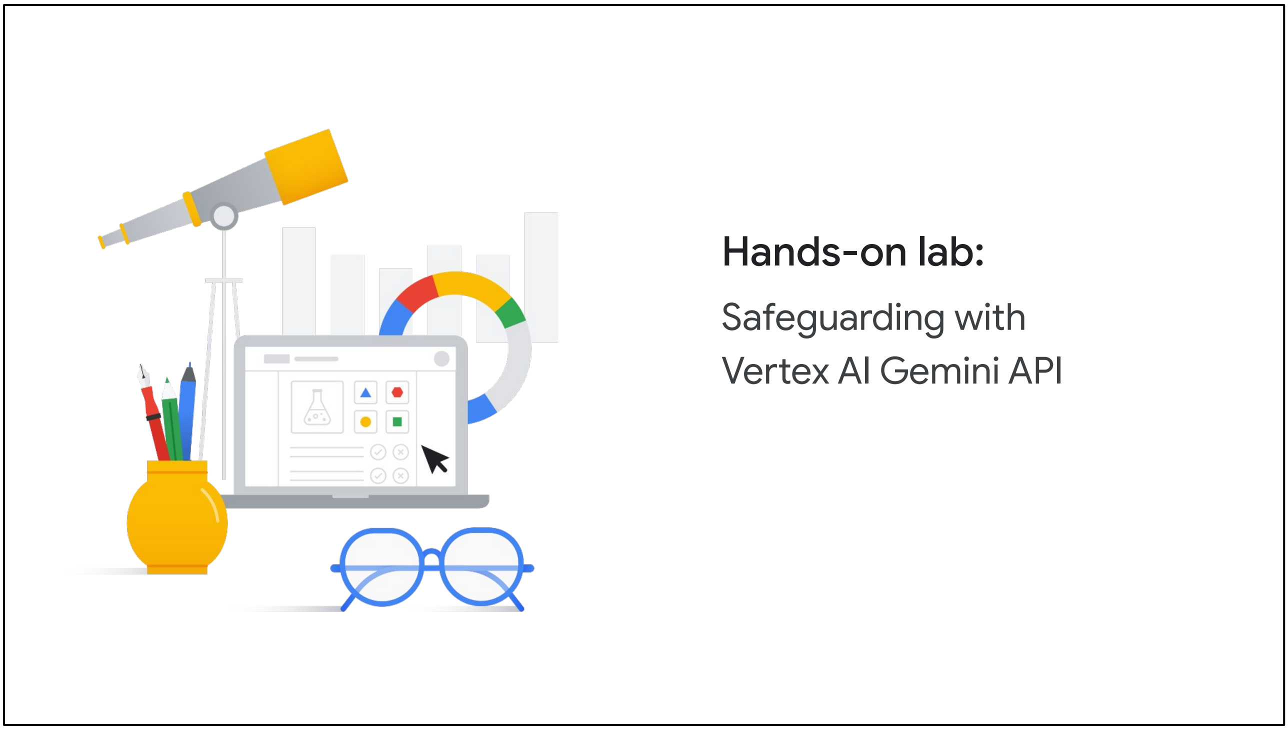


Gemini Safety Feedback Example

```
“finishReason”: “SAFETY”,
“safetyRatings”: [
   {
      “category”: “HARM_CATEGORY_HARASSMENT”,
      “probability”: “MEDIUM”,
      “blocked”: true
   },
   {
      “category”: “HARM_CATEGORY_HATE_SPEECH”,
      “probability”: “NEGLIGIBLE”
   },
   {
      “category”: “HARM_CATEGORY_SEXUALLY_EXPLICITY”,
      “probability”: “NEGLIGIBLE”
   },
   {
      “category”: “HARM_CATEGORY_DANGEROUS_CONTENT”,
      “probability”: “NEGLIGIBLE”
   }
```

and negligible in the dangerous category.

The Gemini API gives you access to the latest generative models from Google. Once you're familiar with the general features available to you through the API, you can use your language of choice to develop for your use case.



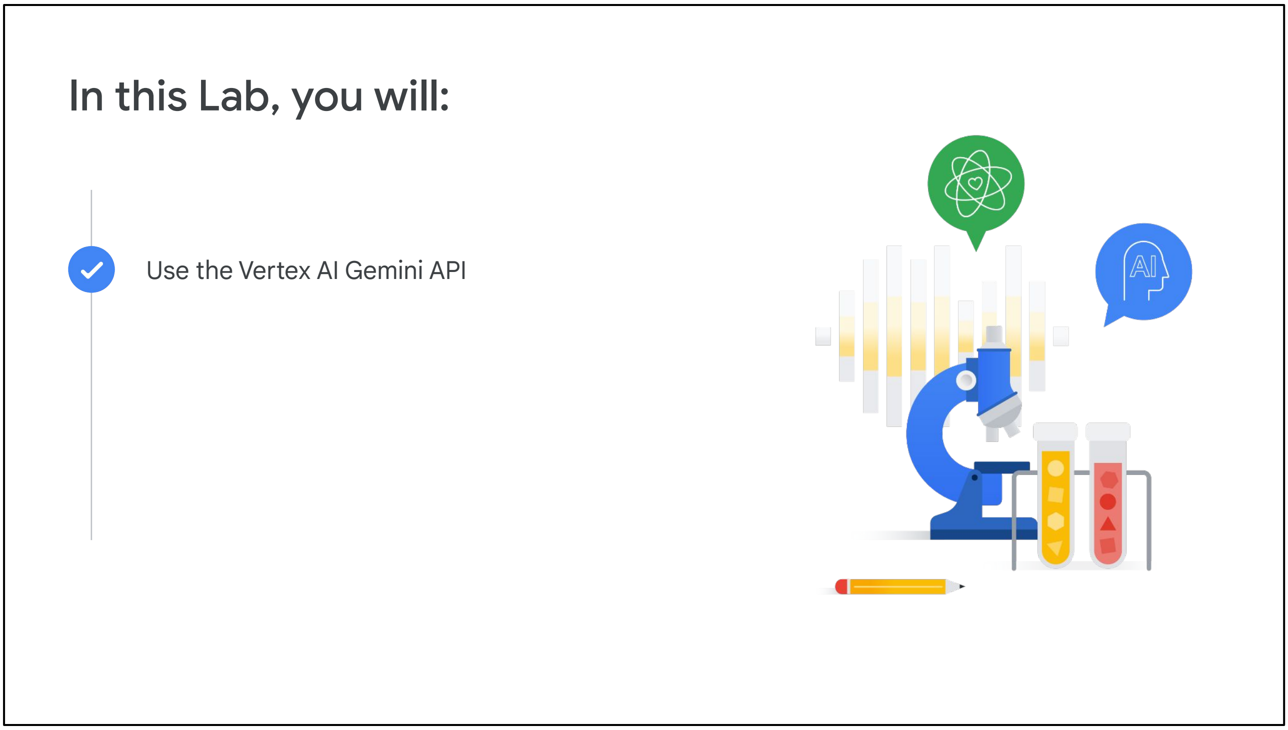
Hands-on lab:

Safeguarding with
Vertex AI Gemini API

Now let’s exercise in a hands-on lab.

In this Lab, you will:

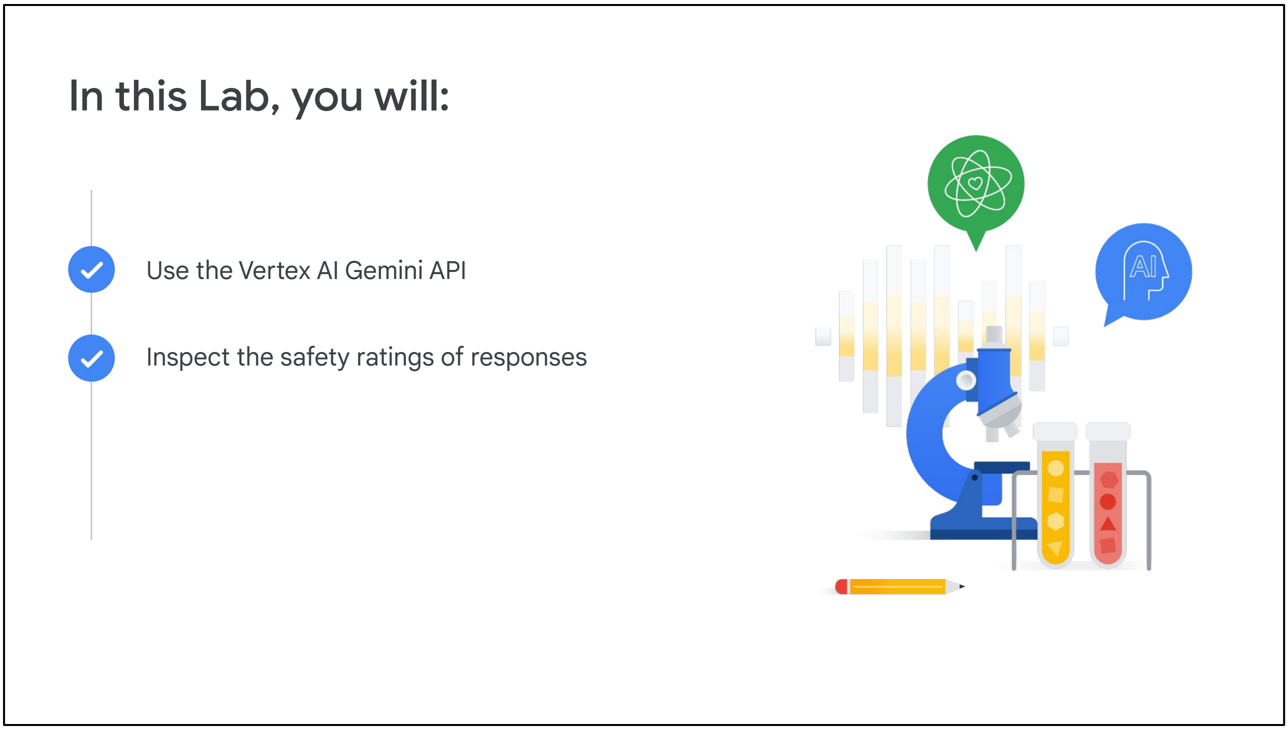
This lab helps you learn how to inspect the safety ratings returned from the Vertex AI Gemini API and how to set a safety threshold to filter responses.
In this lab, you will learn how to:



In this Lab, you will:

☑ Use the Vertex AI Gemini API

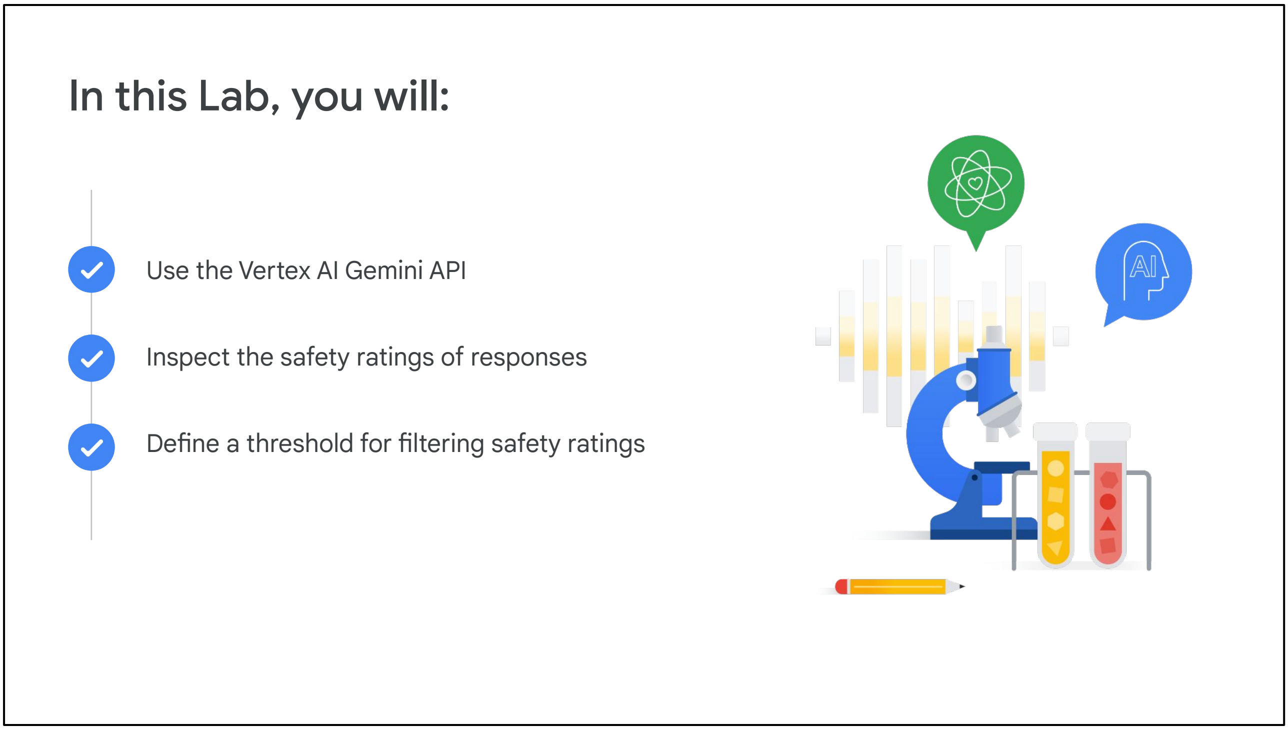
- Use the Vertex AI Gemini API.



In this Lab, you will:

- ✓ Use the Vertex AI Gemini API
- ✓ Inspect the safety ratings of responses

- Inspect the safety ratings of responses.



In this Lab, you will:

- ✓ Use the Vertex AI Gemini API
- ✓ Inspect the safety ratings of responses
- ✓ Define a threshold for filtering safety ratings

- Define a threshold for filtering safety ratings.