



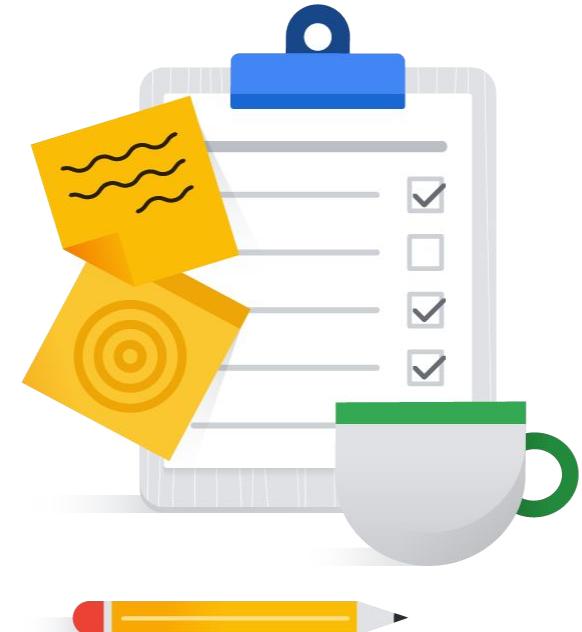
# Vertex AI for ML Workloads

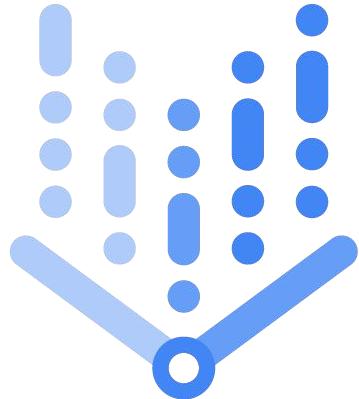
# Vertex AI for ML Workloads

01 Vertex AI on Google Cloud

02 Vertex AI Model Workflow Tools

03 Vertex AI for Generative AI





# Vertex AI

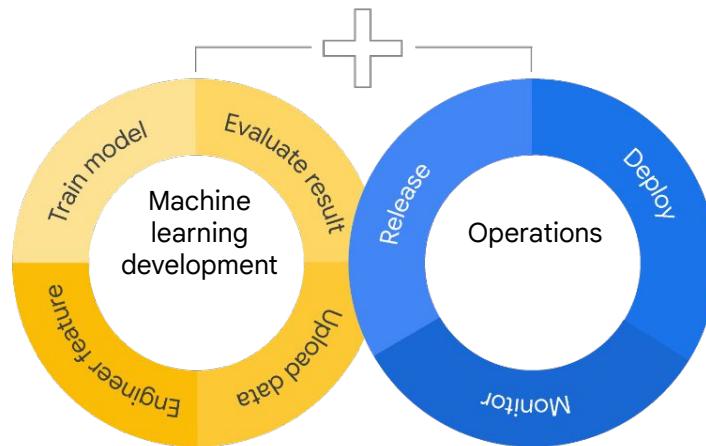
Google Cloud's fully-managed, unified  
machine learning platform

# Improve models with MLOps

Models need to evolve over time, which requires a process for managing change



MLOps phases:



Continuous integration (CI)

Continuous training (CT)

Continuous delivery (CD)

# Vertex AI provides an easy-to-use platform

## Vertex AI

Data Science Tools: Notebooks + integration with data services

Experiment

Train

Deploy

MLOps

## Google Cloud Infrastructure

Data platform tools

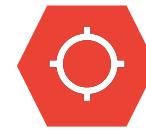
Compute

Storage

Network and Security

# Machine learning workflow

You can use Vertex AI to manage the following stages:



---

Manage and serve data for faster model training.

Train an ML model on your data:

- Train the model
- Evaluate model accuracy
- Tune hyperparameters

Upload and store your models in Vertex AI.

Send batches of data to your models for prediction.

Deploy your trained model to an endpoint for online predictions.

Split traffic between models for testing.

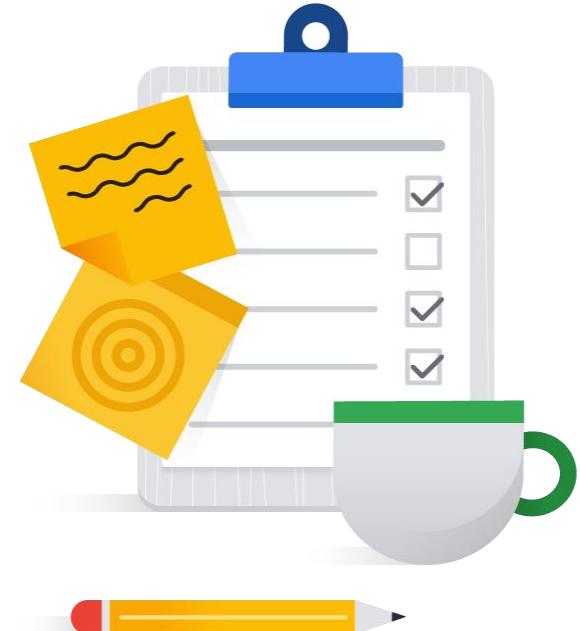
Manage your models and endpoints.

# Vertex AI for ML Workloads

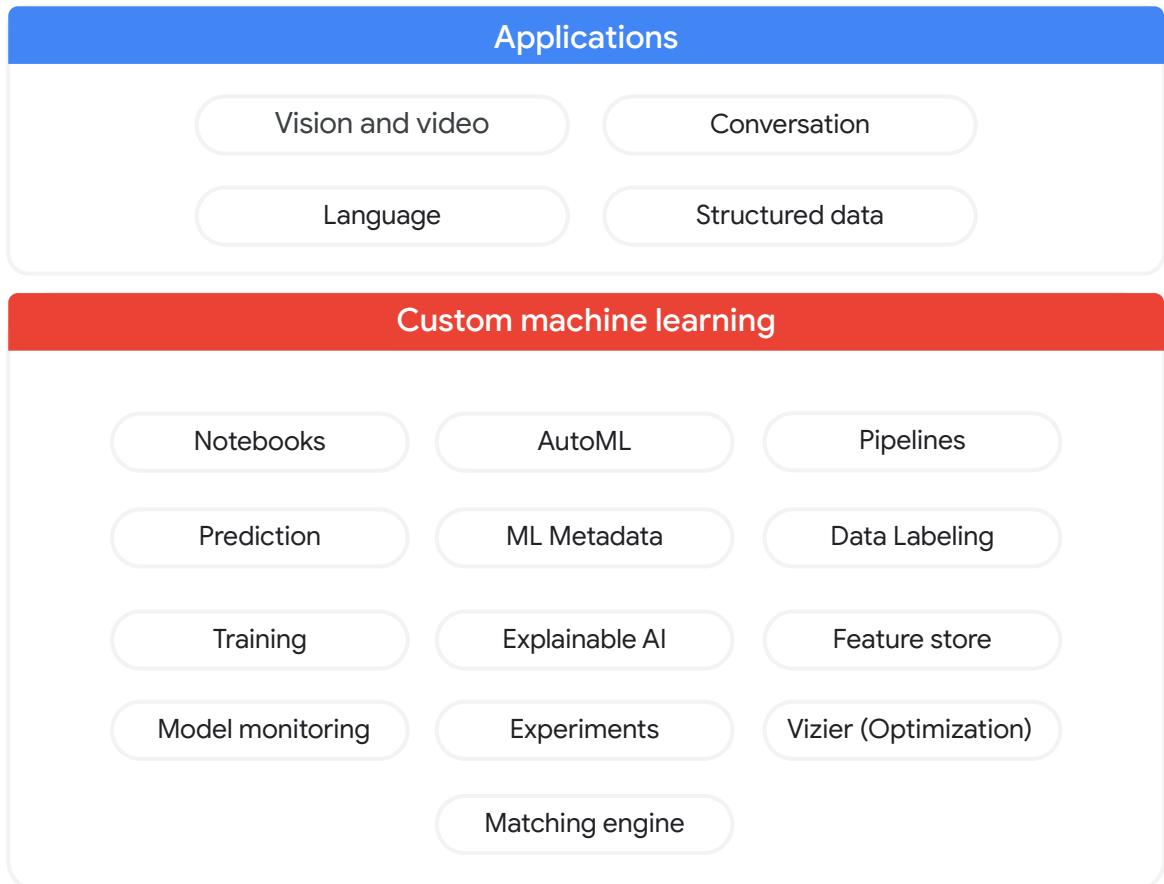
01 Vertex AI on Google Cloud

02 [Vertex AI Model Workflow Tools](#)

03 Vertex AI for Generative AI



# Vertex AI



# Vertex AI Workbench

## Exploratory Analysis

Use industry-standard Jupyter notebooks to explore your data and further transform it at scale with other Google Cloud services.

## Experiment Locally

Build and test your models locally, then migrate the training to use the full power of Google Cloud.

## Collaborate

ML is not a solo task. Share notebooks with your team when you need more eyes on a task, or for inspiration when you find something interesting!

# Feature Store

01

## Ingest Features

Batch? No problem.  
Streaming? Of course.

02

## Serve Features

Serve features for  
training.

Serve features for  
prediction.

03

## Share Features

Don't reinvent the  
feature. Share it!

# Training: AutoML

## Image

- Classification
- Object detection

## Video

- Action recognition
- Classification
- Object tracking

## Text

- Classification
- Entity extraction
- Sentiment analysis

## Tabular

- Classification
- Regression
- Forecasting

# Training: Custom

01

## Prebuilt Runtimes

TensorFlow, PyTorch, scikit-learn, and XGBoost environments are provided.

02

## Custom containers

Need more control of the environment for your training? Hand Vertex AI a container image!

03

## Hyperparameter tuning

Prebuilt or custom, tune your model's hyperparameters as part of the training job.

04

## Distributed training

Models can also be trained on large datasets using distributed infrastructure with Vertex AI.

# Training: Extra services

## ML Metadata

Store training process artifacts, configurations, and metrics for posterity and repeatability.

## Experiments

Compare different models' performances across a range of trial configurations.

## Vizier

Explore hyperparameter values systematically to help find optimal configurations.

# Model Serving

01

## Model Registry

Store and version trained models.

02

## Predictions

Batch or streaming, Vertex AI can provide compute power to meet your model's needs.

03

## Explainable AI

Discover which features contribute most to predictions.

04

## Model Monitoring

Track model inputs for statistical similarity to training data.

# Vertex AI Pipelines

## Directed Acyclic Graphs (DAG)

Automate, monitor, and experiment with interdependent parts of a ML workflow by building a DAG of steps.

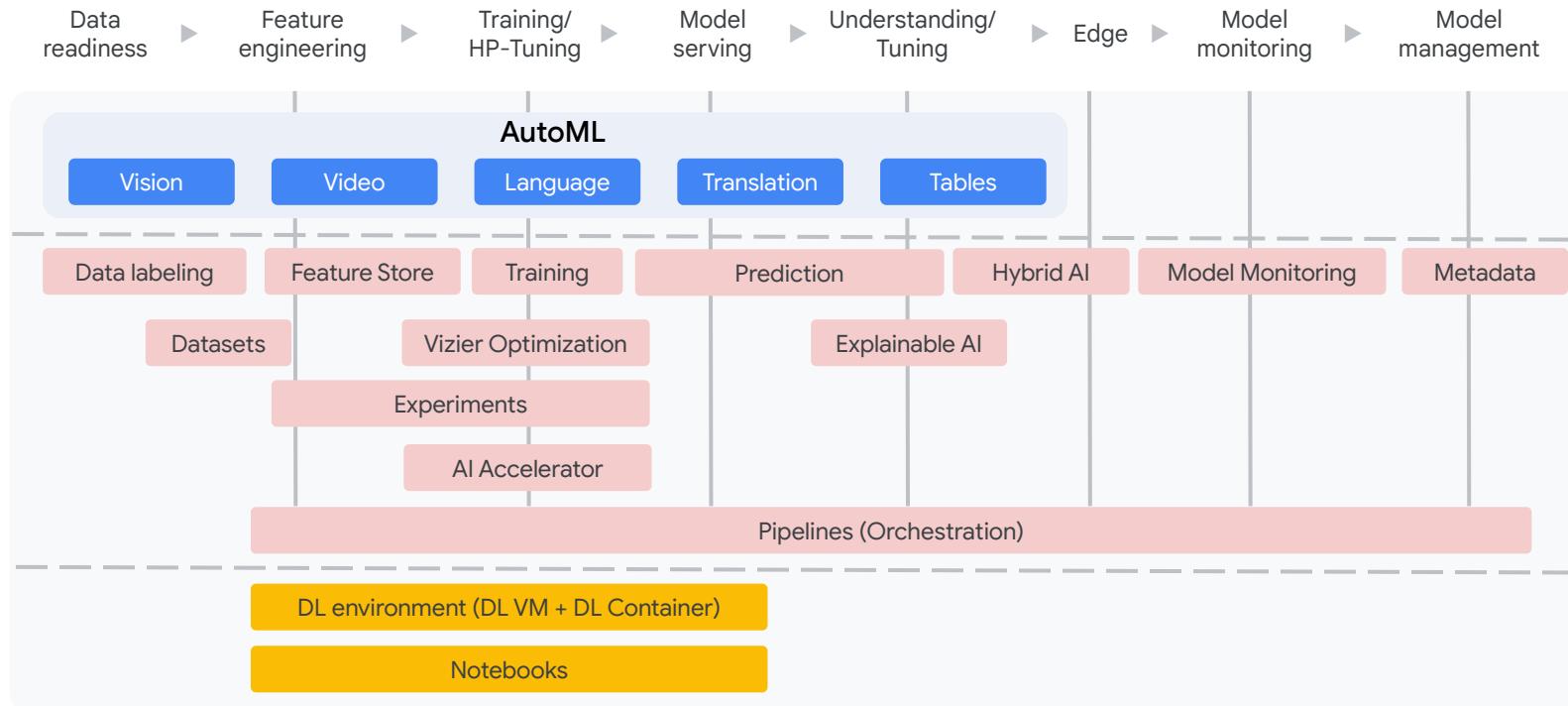
## Scalability

Google Cloud serverless power for running your workflows.

## Reusability

Pipelines can accept variable inputs so are easy to reuse for different, but similar, tasks.

# Vertex AI Model Workflow



# Vertex AI for ML Workloads

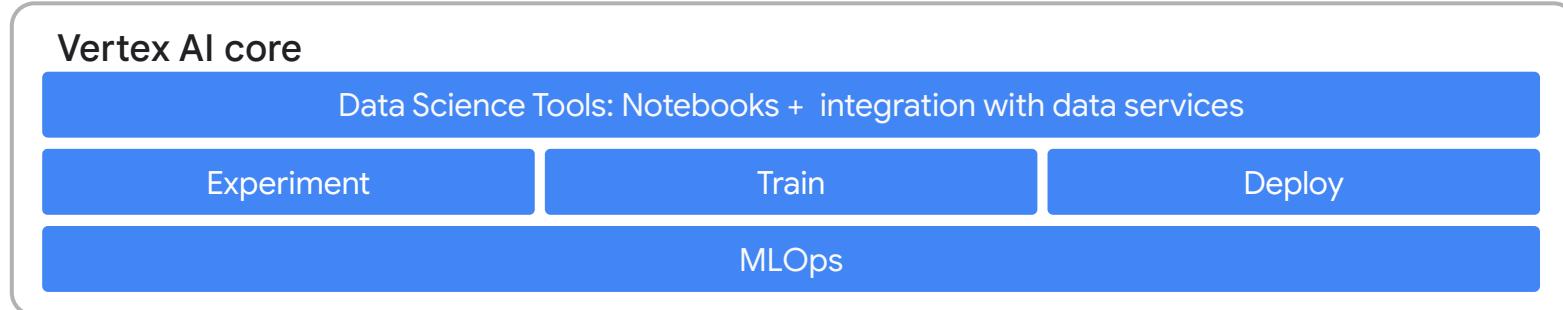
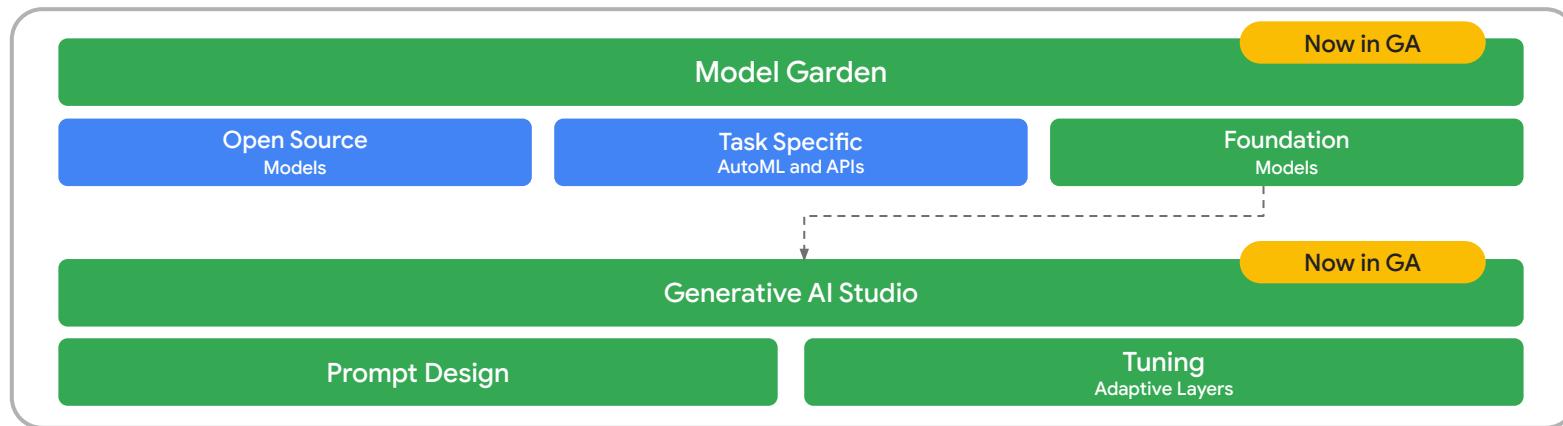
01 Vertex AI on Google Cloud

02 Vertex AI Model Workflow Tools

03 **Vertex AI for Generative AI**



# Generative AI with Vertex AI



# Model Garden

## Foundation

Google's state-of-the-art multimodal models

## Pre-trained APIs

Google's pre-trained, task-specific models

## Open Source

Enterprise-ready open source models

## Third-Party

Coming soon!

# Generative AI Studio

## Explore

Test models, build prototypes, and iterate on designs. No coding or deployment required!

## Tune

Bring your own data to tune the model outputs.

## Deploy

Custom models are in Model Registry, ready to be deployed!

# Data Governance

“

We [Google] don't use data that you provide us to train our own models without your permission.

[Google AI/ML Privacy Commitment](#)

# Responsible AI

<https://ai.google/responsibility>

- 01 Be socially beneficial.

---

- 02 Avoid creating or reinforcing unfair bias.

---

- 03 Be built and tested for safety.

---

- 04 Be accountable to people.

---

- 05 Incorporate privacy design principles.

---

- 06 Uphold high standards of scientific excellence.

---

- 07 Be made available for uses that accord with these principles.





# Model Garden

# Model Garden

- 01 **Introduction to Model Garden**
- 02 Available Model Types
- 03 Working with Gen AI Studio
- 04 Working with Model Registry
- 05 Intro: Course Use Cases



Search models

 Suggestions [text embedding](#) [essay outline](#) [BERT](#)
**Modalities**

Language	23
Vision	46
Tabular	2
Documents	2
Speech	1

**Tasks**

Generation	22
Classification	26
Detection	17
Extraction	7
Recognition	6
Translation	2
Embedding	2
Segmentation	3

**Features**

API available	9
Open source	34
Notebook support	33
Pipeline support	2
One-click deployment	1

**Foundation models**

Pre-trained multi-task models that can be further tuned or customized for specific tasks.

[Foundation](#) [Language](#)
**PaLM 2 for Text**

Fine-tuned to follow natural language instructions and is suitable for a variety of language tasks, such as: classification, extraction, summarization and content...  
text-bison@001

[VIEW DETAILS](#)
[Foundation](#) [Language](#)
**PaLM 2 for Chat**

Fine-tuned to conduct natural conversation. Use this model to build and customize your own chatbot application.  
chat-bison@001

[VIEW DETAILS](#)
[Foundation](#) [Language](#)
**Embeddings for text**

Text embedding is an important NLP technique that converts textual data into numerical vectors that can be processed by machine learning algorithms, especially large models...  
textembedding-gecko@001

[VIEW DETAILS](#)
[Foundation](#) [Speech](#)
**Chirp**

Chirp is a version of a Universal Speech Model that has over 2B parameters and can transcribe in over 100 languages in a single model.  
chirp-rnnt1

[VIEW DETAILS](#)
[Foundation](#) [Vision](#)
**Embeddings for Image**

Generates vectors based on images, which can be used for downstream tasks like image classification, image search, and so on.  
imageembedding-001

[VIEW DETAILS](#)
[▼ SHOW ALL \(25\)](#)
**Fine-tunable models**

Models that data scientists can further fine-tune through a custom notebook or pipeline.

[Classification](#) [Vision](#)
**tflhub/EfficientNetV2**

EfficientNet V2 are a family of image classification models, which achieve better parameter efficiency and faster training speed than prior arts.  
tensorflow-hub/efficientnetv2

[Classification](#) [Vision](#)
**tfvision/vit**

The Vision Transformer (ViT) is a transformer-based architecture for image classification.  
tfvision/vit-s16

[Detection](#) [Vision](#)
**tfvision/SpineNet**

SpineNet is an image object detection model generated using Neural Architecture Search.  
tfvision/spinenet49

[Detection](#) [Vision](#)
**tfvision/YOLO**

YOLO algorithm is a one-stage object detection algorithm that can achieve real-time performance on a single GPU.  
tfvision/scaled-yolo

[Foundation](#) [Language](#) [Vision](#)
**Stable Diffusion Inpainting**

Stable Diffusion Inpainting is a latent diffusion model capable of inpainting images given any text input and a mask image.  
runwayml/stable-diffusion-inpainting

## Modalities

Language	23
Vision	46
Tabular	2
Documents	2
Speech	1

## Tasks

Generation	22
Classification	26
Detection	17
Extraction	7
Recognition	6
Translation	2
Embedding	2
Segmentation	3

## Features

API available	9
Open source	34
Notebook support	33
Pipeline support	2
One-click deployment	1

Search models

Suggestions [text embedding](#) [essay outline](#) [BERT](#)

## Foundation models

Pre-trained multi-task models that can be further tuned or customized for specific tasks.

Foundation Language

## PaLM 2 for Text

Fine-tuned to follow natural language instructions and is suitable for a variety of language tasks, such as: classification, extraction, summarization and content...  
text-bison@001

[VIEW DETAILS](#)

Foundation Language

## PaLM 2 for Chat

Fine-tuned to conduct natural conversation. Use this model to build and customize your own chatbot application.  
chat-bison@001

[VIEW DETAILS](#)

Foundation Language

## Embeddings for text

Text embedding is an important NLP technique that converts textual data into numerical vectors that can be processed by machine learning algorithms, especially large models...  
textembedding-gecko@001

[VIEW DETAILS](#)

Foundation Speech

## Chirp

Chirp is a version of a Universal Speech Model that has over 2B parameters and can transcribe in over 100 languages in a single model.  
chirp-rnnt1

[VIEW DETAILS](#)

Foundation Vision

## Embeddings for Image

Generates vectors based on images, which can be used for downstream tasks like image classification, image search, and so on.

imageembedding-001

[VIEW DETAILS](#)

▼ SHOW ALL (25)

## Fine-tunable models

Models that data scientists can further fine-tune through a custom notebook or pipeline.

Classification Vision

## tfhub/EfficientNetV2

EfficientNet V2 are a family of image classification models, which achieve better parameter efficiency and faster training speed than prior arts.

[tensorflow-hub/efficientnetv2](#)

Classification Vision

## tfvision/vit

The Vision Transformer (ViT) is a transformer-based architecture for image classification.

[tfvision/vit-s16](#)

Detection Vision

## tfvision/SpineNet

SpineNet is an image object detection model generated using Neural Architecture Search.

[tfvision/spinenet49](#)

Detection Vision

## tfvision/YOLO

YOLO algorithm is a one-stage object detection algorithm that can achieve real-time performance on a single GPU.

[tfvision/scaled-yolo](#)

Foundation Language Vision

## Stable Diffusion Inpainting

Stable Diffusion Inpainting is a latent diffusion model capable of inpainting images given any text input and a mask image.

[runwayml/stable-diffusion-inpainting](#)

Modality	Count	Description
Language	23	
Vision	46	<a href="#">View outline</a> <a href="#">BERT</a>
Tabular	2	
Documents	2	can be further tuned or customized for specific tasks.
Speech	1	
Task	Count	Description
Generation	22	<a href="#">Usage</a> Fine-tuned to conduct natural conversation. Use this model to build and customize your own chatbot application. <a href="#">chat-bison@001</a>
Classification	26	<a href="#">Usage</a> Text embedding is an important NLP technique that converts textual data into numerical vectors that can be processed by machine learning algorithms, especially large models... <a href="#">textembedding-gecko@001</a>
Detection	17	
Extraction	7	
Recognition	6	Further fine-tune through a custom notebook or pipeline.
Translation	2	
Embedding	2	<a href="#">Image</a> The Vision Transformer (ViT) is a transformer-based architecture for image classification. <a href="#">tfvision/vit-s16</a>
Segmentation	3	<a href="#">Image</a> SpineNet is an image object detection model generated using Neural Architecture Search. <a href="#">tfvision/spinenet49</a>
Model	Count	Description
PaLM 2 for Chat	Foundation	Language
Embeddings for text	Foundation	Language
Chirp	Foundation	Speech
Embeddings for Image	Foundation	Vision
tfvision/vit	Classification	Vision
tfvision/SpineNet	Detection	Vision
tfvision/YOLO	Detection	Vision
Stable Diffusion Inpainting	Foundation	Language
Stable Diffusion Inpainting	Foundation	Vision

## Modalities

Language	23
Vision	46
Tabular	2
Documents	2
Speech	1

## Tasks

Generation	22
Classification	26
Detection	17
Extraction	7
Recognition	6
Translation	2
Embedding	2
Segmentation	3

## Features

API available	9
Open source	34
Notebook support	33
Pipeline support	2
One-click deployment	1

Search models

Suggestion

Found:

Pre-trained

Found

PaLM

Fine-tu

instruc

variety

classifi

summa

text-bis

VIEW

▼ SHOW

Fine-tu

Models th

Classif

tfhub

Efficien

classifi

better f

faster

## Foundation models

Pre-trained multi-task models that can be further tuned or customized for specific tasks.

[Foundation](#) [Language](#)

### PaLM 2 for Text

Fine-tuned to follow natural language instructions and is suitable for a variety of language tasks, such as: classification, extraction, summarization and content...  
text-bison@001

[VIEW DETAILS](#)[Foundation](#) [Language](#)

### PaLM 2 for Chat

Fine-tuned to conduct natural conversation. Use this model to build and customize your own chatbot application.

chat-bison@001

[VIEW DETAILS](#)[Foundation](#) [Language](#)

### Embeddings for text

Text embedding is an important NLP technique that converts textual data into numerical vectors that can be processed by machine learning algorithms, especially large models...  
textembedding-gecko@001

[VIEW DETAILS](#)

## Fine-tunable models

tensorflow-hub/efficientnetv2

tfvision/vit-s16

tfvision/spinenet49

tfvision/scaled-yolo

runwayml/stable-diffusion-inpainting

# Model Card

## PaLM 2 for Text

Fine-tuned to follow natural language instructions and is suitable for a variety of language tasks, such as: classification, extraction, summarization and content generation.

[OPEN PROMPT DESIGN](#)[VIEW API CODE](#)[OVERVIEW](#)[USE CASES](#)[DOCUMENTATION](#)

### Overview

**text-bison** is the name of the PaLM 2 for text large language model that understands and generates language. It's a foundation model that performs well at a variety of natural language tasks such as sentiment analysis, entity extraction, and content creation. The type of content that **text-bison** can create includes document summaries, answers to questions, and labels that classify content.

The PaLM 2 for text is ideal for tasks that can be completed with one API response, without the need for continuous conversation. For text tasks that require back-and-forth interactions, use the [PaLM 2 for chat](#).

### Use cases

- **Summarization:** Create a shorter version of a document that incorporates pertinent information from the original text. For example, you might want to summarize a chapter from a textbook. Or, you could create a succinct product description from a long paragraph that describes the product in detail.
- **Question answering:** Provide answers to questions in text. For example, you might automate the creation of a Frequently Asked Questions (FAQ) document from knowledge base content.
- **Classification:** Assign a label to provided text. For example, a label might be applied to text that describes how grammatically correct it is.

Resource ID

text-bison@001

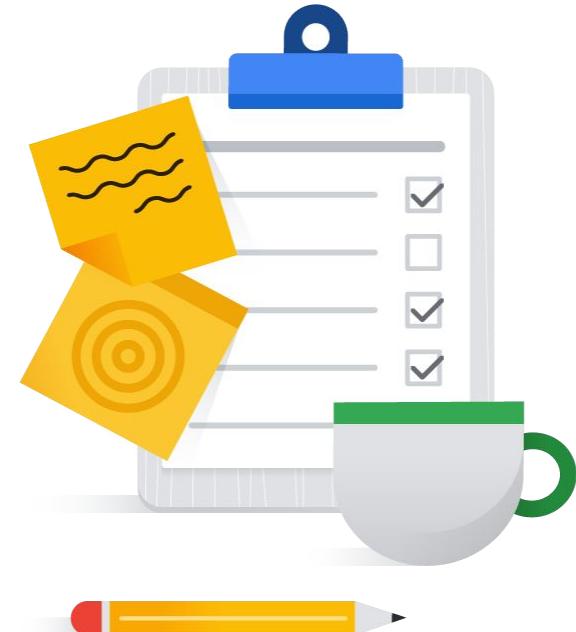
Tags

Task

Classification    Detection  
Extraction    Generation  
Recognition    Translation

# Model Garden

- 01 Introduction to Model Garden
- 02 Available Model Types
- 03 Working with Gen AI Studio
- 04 Working with Model Registry
- 05 Intro: Course Use Cases



# Model Garden

## Foundation

Google's state-of-the-art multimodal models

## Pre-trained APIs

Google's pre-trained, task-specific models

## Open Source

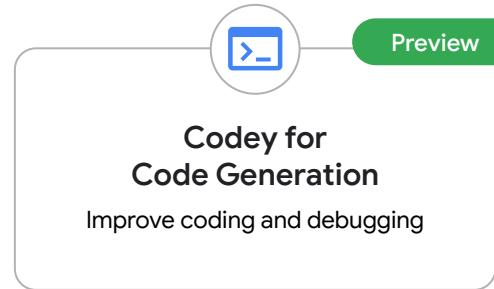
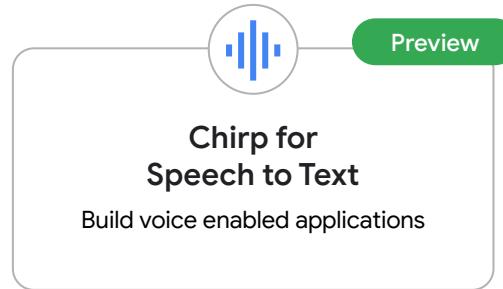
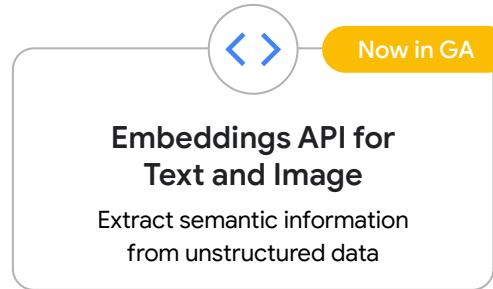
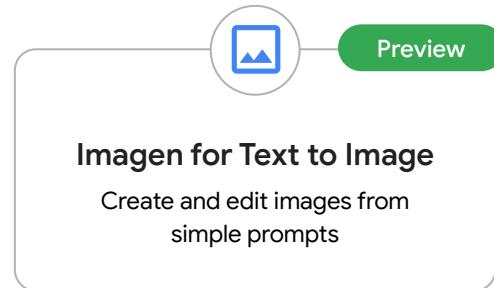
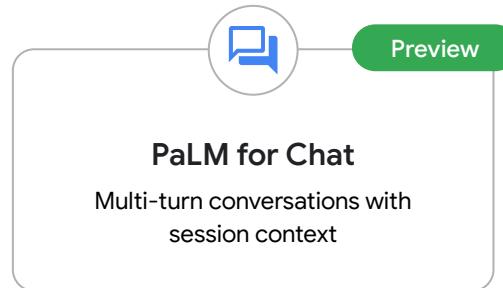
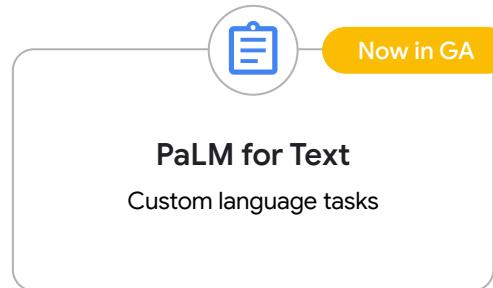
Enterprise-ready open source models

## Third-Party

Coming soon!

# Foundation Models

Across a variety of model sizes to address use cases



# PaLM 2

## Text

Perform language tasks like summarization, question answering, and classification from natural language prompts.

## Chat

Preview

Text tasks that flow like natural, multi-turn conversations such as help agents.

## Embeddings

Retrieve the vector representation for the semantic meaning of input text.

# Codey

Currently US only

## Code Generation

Generate functions,  
tests, and object classes  
from natural language  
descriptions.

Currently US only

## Code Completion

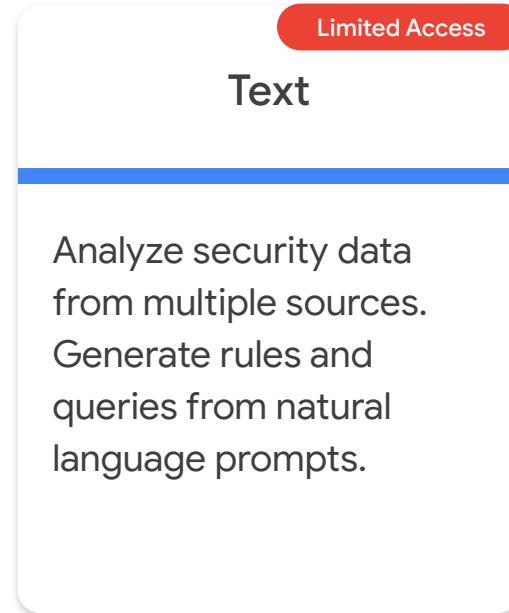
In-line code generation  
based on surrounding  
code.

Currently US only

## Code Chat

Multi-turn reasoning,  
debugging, and  
explanations of code.

# Sec-PaLM2



# Imagen

Limited Access

## Image Generation

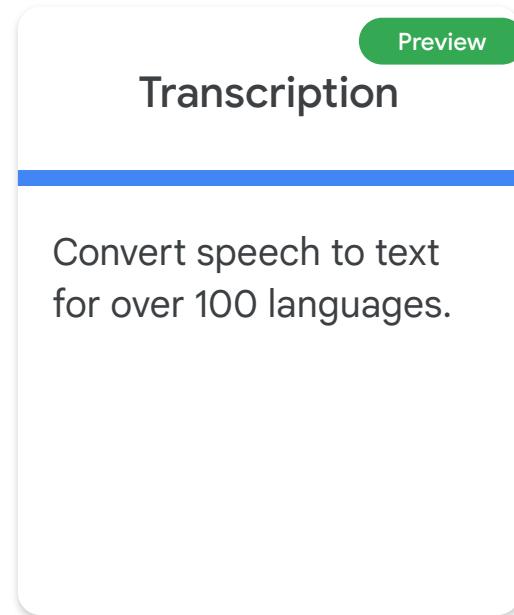
Create images from text prompts.

Limited Access

## Embeddings

Vector representation of an image. Also generates text vectors for querying text-to-image and image-to-text.

# Chirp



Convert speech to text  
for over 100 languages.

# Other foundation models

## Image Generation

- Stable Diffusion
- Stable Diffusion Inpainting
- InstructPix2Pix
- ControlNet

## Image Processing

- PaLI zero-shot
- BLIP
- BLIP 2
- CLIP
- OWL-ViT
- ViT GPT2
- LayoutLM for VQA

## Natural Language

- BERT
- T5-FLAN

# Fine-tunable models

## Foundation

- PaLM (via Gen AI Studio)
- Stable Diffusion Inpainting
- ControlNet

## Image Classification

- EfficientNetV2
- ResNet
- BEiT
- DeiT
- MobileNet
- ... many more!

## Object Detection

- SpineNet
- YOLO
- Faster R-CNN
- Mask R-CNN
- RetinaNet

# Task-specific solutions: Vision

## Detection

- Occupancy
- Person/Vehicle
- PPE
- Objects
- Person blur
- Watermark
- Faces

## Recognition

- Product
- Tag

## Extraction

- Text
- Form Parser
- Document AI
- OCR

## Classification

- Moderation

# Task-specific solutions: Language

## Classification

- Entity analysis
- Sentiment analysis
- Entity sentiment analysis
- Text Moderation

## Translation

- Text Translation

## Extraction

- Syntax analysis

# Task-specific solutions: Tabular

## Classification / Regression

- AutoML E2E
- TabNet

Sometimes, you don't need a model.  
You just need a **prompt**.

# Model Garden

- 01 Introduction to Model Garden
- 02 Available Model Types
- 03 **Working with Gen AI Studio**
- 04 Working with Model Registry
- 05 Intro: Course Use Cases



# Generative AI Studio

## PaLM 2 for Text

Fine-tuned to follow natural language instructions and is suitable for a variety of language tasks, such as: classification, extraction, summarization and content generation.

[OPEN PROMPT DESIGN](#)[VIEW API CODE](#)[OVERVIEW](#)[USE CASES](#)[DOCUMENTATION](#)

### Overview

**text-bison** is the name of the PaLM 2 for text large language model that understands and generates language. It's a foundation model that performs well at a variety of natural language tasks such as sentiment analysis, entity extraction, and content creation. The type of content that **text-bison** can create includes document summaries, answers to questions, and labels that classify content.

The PaLM 2 for text is ideal for tasks that can be completed with one API response, without the need for continuous conversation. For text tasks that require back-and-forth interactions, use the [PaLM 2 for chat](#).

### Use cases

- **Summarization:** Create a shorter version of a document that incorporates pertinent information from the original text. For example, you might want to summarize a chapter from a textbook. Or, you could create a succinct product description from a long paragraph that describes the product in detail.
- **Question answering:** Provide answers to questions in text. For example, you might automate the creation

## Codey for Code Generation

Generates code based on natural language input. Good for writing functions, classes, unit tests, and more.

[OPEN PROMPT DESIGN](#)[VIEW API CODE](#)[OVERVIEW](#)[USE CASES](#)[DOCUMENTATION](#)

### Overview

**code-bison** is the name of the model that supports code generation. It's a foundation model that generates code based on a natural language description. The type of content that **code-bison** can create includes functions, web pages, and unit tests. **code-bison** is supported by the code generation Codey API. The Codey APIs are in the PaLM API family.

The Codey API for code generation is ideal for tasks that can be completed with one API response, without the need for continuous conversation. For code tasks that require back-and-forth interactions, use the [Codey API for code chat](#).

### Use cases

Some common used cases for code generation are:

- **Unit tests:** Use the prompt to request a unit test for a function.
- **Write a function:** Pass a problem to the model to get a function that solves that problem.
- **Create a class:** Use a prompt to describe the purpose of a class and have code that defines the class returned.

### Prompt

Write a prompt and then click Submit



### Response

Markdown

The model will generate a response after you click Submit

Model may display inaccurate or offensive information that doesn't represent Google's view. Google models currently only support English.

**i** We want your feedback.

**Model**

text-bison@001

Temperature **?**

0 1

0.2

Token limit **?**

1 1024

256

Top-K **?**

1 40

40

Top-P **?**

0 1

0.8

Safety filter threshold

Block few

**?**

**SUBMIT**    **RESET PARAMETERS**

**REPORT INAPPROPRIATE RESPONSES**

Content processed through Vertex AI is assessed against a list of safety attributes. Confidence scores for these attributes are returned in API responses [but are not visible in the UI](#).

### Prompt

Write a prompt and then click Submit



### Response

Markdown

The model will generate a response after you click Submit

Model may display inaccurate or offensive information that doesn't represent Google's view. Google models currently only support English.

We want your [feedback](#).

### Model

Filter  Type to filter

PaLM 2

text-bison (latest)

Latest

text-bison@001

Best value

Codey (Palm 2)

code-bison@001

[PREVIEW](#)

Code generation

code-gecko@001

[PREVIEW](#)

### REPORT INAPPROPRIATE RESPONSES

Content processed through Vertex AI is assessed against a list of safety attributes. Confidence scores for these attributes are returned in API responses [but are not visible in the UI](#).

**Prompt**

Write a prompt and then click Submit

**Response**

Markdown

The model will generate a response after you click Submit

Model may display inaccurate or offensive information that doesn't represent Google's view. Google models currently only support English.

**Temperature** ?

0  1 0.2

**Token limit** ?

0  1024 256

**Top-K** ?

1  40 40

**Top-P** ?

0  1 0.8

**Safety filter threshold**

Block few ▼ ?

**SUBMIT** **RESET PARAMETERS**

**REPORT INAPPROPRIATE RESPONSES**

Content processed through Vertex AI is assessed against a list of safety attributes. Confidence scores for these attributes are returned in API responses [but are not visible in the UI](#).

**Context** ?

Write your context here...

**Examples** ?**ADD INPUT COLUMN****INPUT** Add an input prefix**OUTPUT** Add an output prefix

⊕ Write an input to add a new row

⊕ Write an output to add a new row

**Test** ?**ADD TO EXAMPLES****INPUT** Add an input prefix**OUTPUT** Add an output prefix **Markdown**

Write an input



Model responses will appear here

Model may display inaccurate or offensive information that doesn't represent Google's view. Google models currently only support English.

**i** We want your feedback.**Model**

text-bison@001

**Temperature** ?

0 1

0.2

**Token limit** ?

1 1024

256

**Top-K** ?

1 40

40

**Top-P** ?

0 1

0.8

**Safety filter threshold**

Block few

**SUBMIT**    **RESET PARAMETERS****i** REPORT INAPPROPRIATE RESPONSESContent processed through Vertex AI is assessed against a list of safety attributes. Confidence scores for these attributes are returned in API responses [but are not visible in the UI](#).

Context ?

You are a machine learning practitioner training a junior staff member. Provide answers to questions that include practical use cases to help the new employee learn.

Examples ?**ADD INPUT COLUMN**Test ?**ADD TO EXAMPLES**

INPUT Add an input prefix

OUTPUT Add an output prefix

Markdown

Why would I choose precision versus recall as my evaluation metric?

Precision and recall are two important metrics for evaluating the performance of a binary classifier. Precision is the fraction of positive predictions that are actually positive, while recall is the fraction of actual positives that are predicted as positive.

Precision is important when the cost of false positives is high. For example, if you are using a classifier to identify fraudulent transactions, you would want to have a high precision so that you don't accidentally block legitimate transactions.

Recall is important when the cost of false negatives is high. For example, if you are using a classifier to identify cancer, you would want to have a high recall so that you don't miss any cases of cancer.

The choice of which metric to use depends on the specific application. In general, you should choose the metric that is most important for your particular application.

We want your [feedback](#).

## Model

text-bison@001

Temperature ?0  1 0.2Token limit ?1  1024 256Top-K ?1  40 40Top-P ?0  1 0.8

## Safety filter threshold

Block few

**SUBMIT**

RESET PARAMETERS

**REPORT INAPPROPRIATE RESPONSES**

Content processed through Vertex AI is assessed against a list of safety attributes. Confidence scores for these attributes are returned in API responses [but are not visible in the UI](#).

Context ?

You are a machine learning practitioner training a junior staff member. Provide answers to questions that include practical use cases to help the new employee learn.

Examples ?

ADD INPUT COLUMN

Test ?

ADD TO EXAMPLES

INPUT Add an input prefix

OUTPUT Add an output prefix

Markdown

Why would I choose precision versus recall as my evaluation metric?



Precision and recall are two important metrics for evaluating the performance of a binary classifier. Precision is the fraction of positive predictions that are actually positive, while recall is the fraction of actual positives that are predicted as positive.

Precision is important when the cost of false positives is high. For example, if you are using a classifier to identify fraudulent transactions, you would want to have a high precision so that you don't accidentally block legitimate transactions.

Recall is important when the cost of false negatives is high. For example, if you are using a classifier to identify cancer, you would want to have a high recall so that you don't miss any cases of cancer.

The choice of which metric to use depends on the specific application. In general, you should choose the metric that is most important for your particular application.

We want your [feedback](#).

Model

text-bison@001

Temperature ?Token limit ?Top-K ?Top-P ?

Safety filter threshold

Block few

SUBMIT

RESET PARAMETERS

REPORT INAPPROPRIATE RESPONSES

Content processed through Vertex AI is assessed against a list of safety attributes. Confidence scores for these attributes are returned in API responses [but are not visible in the UI](#).

Context ?

You are a machine learning practitioner training a junior staff member. Provide answers to questions that include practical use cases to help the new employee learn.

Examples ?

ADD INPUT COLUMN

Test ?

ADD TO EXAMPLES

INPUT Add an input prefix

OUTPUT Add an output prefix

Markdown

Why would I choose precision versus recall as my evaluation metric?



Precision and recall are two important metrics for evaluating the performance of a binary classifier. Precision is the fraction of positive predictions that are actually positive, while recall is the fraction of actual positives that are predicted as positive.

Precision is important when the cost of false positives is high. For example, if you are using a classifier to identify fraudulent transactions, you would want to have a high precision so that you don't accidentally block legitimate transactions.

Recall is important when the cost of false negatives is high. For example, if you are using a classifier to identify cancer, you would want to have a high recall so that you don't miss any cases of cancer.

The choice of which metric to use depends on the specific application. In general, you should choose the metric that is most important for your particular application.

We want your [feedback](#).

Model

text-bison@001

Temperature ?Token limit ?Top-K ?Top-P ?

Safety filter threshold

Block few

SUBMIT

RESET PARAMETERS

## REPORT INAPPROPRIATE RESPONSES

Content processed through Vertex AI is assessed against a list of safety attributes. Confidence scores for these attributes are returned in API responses [but are not visible in the UI](#).

[← Create a tuned model](#)**1** Tuning dataset**2** Model details

START TUNING

Model tuning creates and deploys a new model from an existing one that's better adapted to your use case. Currently, model tuning occurs in limited regions. If you have an organization policy restricting certain regions, model tuning may fail.

**Tuning dataset**

The tuning dataset is a JSONL file that contains model prompt and responses examples(one per line). It's recommended that you use at least 100-500 samples. You can upload the file or select one that's already on Cloud Storage.

- Upload JSONL file to Cloud Storage  
 Existing JSONL file on Cloud Storage

**Select JSONL file \*** my\_training\_data.jsonl BROWSE

Each line in the file contains one example:

- An `input_text` field containing the prompt
- An `output_text` field containing an example response

For example: `{"input_text": "Create a description for Plantation Palms", "output_text": "Enjoy some fun in the sun at Gulf Shores."}`

**Dataset location \*** gs:// awesometrainingdata BROWSE

The Cloud Storage location where the JSONL file will be stored.

 CONTINUE

## Create a tuned model

ntial

 Tuning dataset

 Model details

**START TUNING**

**Model name \***

Product Generator

The name of the new model. Up to 128 characters.

### Settings

**Base model**

text-bison@001



The base model that will be used to create a new tuned model.

**Train steps \***

100



**Learning Rate \***

3



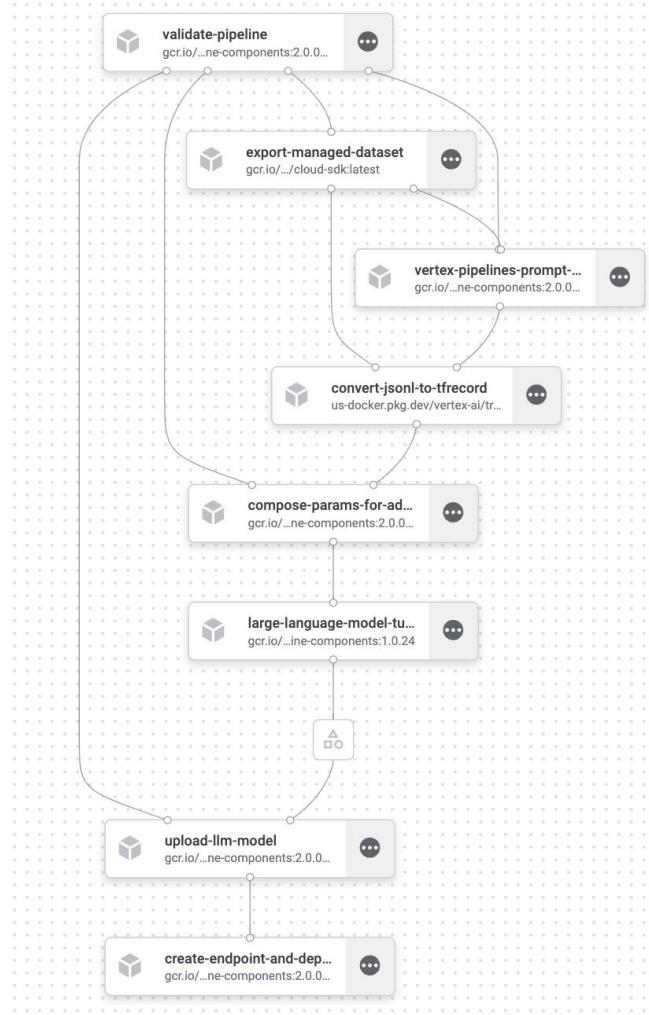
**Working directory \***

 gs:// awesometrainingdata/working/product\_generator/

BROWSE

The Cloud Storage location where the artifacts are stored during the pipeline tuning run.

jud



# Model Garden

- 01 Introduction to Model Garden
- 02 Available Model Types
- 03 Working with Gen AI Studio
- 04 **Working with Model Registry**
- 05 Intro: Course Use Cases



# Model Registry

## Model Registry

[+ CREATE](#)[IMPORT](#)[REFRESH](#)[LEARN](#)

Models are built from your datasets or unmanaged data sources. There are many different types of machine learning models available on Vertex AI, depending on your use case and level of experience with machine learning. [Learn more](#)

Region

us-central1 (Iowa)

**Filter** Enter a property name

<input type="checkbox"/> Name	Deployment status	Description	Default version	Type	Source	Updated ↓	Labels
-------------------------------	-------------------	-------------	-----------------	------	--------	-----------	--------

No rows to display

Models are built from your datasets or unmanaged data sources. Choose the different types of machine learning models available on Vertex AI based on your use case and level of experience with machine learning. [Learn more](#)

## Region

us-central1 (Iowa)

**Filter** Enter a property name

<input type="checkbox"/> Name	Deployment status
No rows to display	

## Train new model

- 1 Training method
- 2 Model details
- 3 Training container
- 4 Hyperparameters (optional)
- 5 Compute and pricing
- 6 Prediction container (optional)

[START TRAINING](#)[CANCEL](#)

Dataset \*

No managed dataset



Annotation set

-



Objective

Custom



Please refer to the pricing guide for more details (and available deployment options) for each method.



AutoML options are only available when you train with a managed dataset.

### Model training method

 AutoML

Train high-quality models with minimal effort and machine learning expertise. Just specify how long you want to train. [Learn more](#)

 AutoML Edge

Train a model that can be exported for on-prem/on-device use. Typically has lower accuracy. [Learn more](#)

 Custom training (advanced)

Run your TensorFlow, scikit-learn, and XGBoost training applications in the cloud. Train with one of Google Cloud's pre-built containers or use your own. [Learn more](#)

[CONTINUE](#)

Models are built from your datasets or unmanaged data sources. There are different types of machine learning models available on Vertex AI, depending on the case and level of experience with machine learning. [Learn more](#)

## Region

us-central1 (Iowa)

**Filter** Enter a property name

<input type="checkbox"/>	Name	Deployment status
--------------------------	------	-------------------

No rows to display

## Import model

- 1 Name and region
- 2 Model settings
- 3 Explainability (optional)

**IMPORT**

CANCEL

You can import model artifacts that have been trained outside of Google Cloud. Once your model has been imported, you can serve it for online or batch predictions and compare it against your other Cloud AI models. [More info](#)

 Import as new model

Creates a new model group and assigns the imported model as version 1

 Import as new version

Imports the model as a version of an existing model

Name \*

Description

Region

us-central1 (Iowa)

**▼ ADVANCED OPTIONS****CONTINUE**

# Model Card

## Stable Diffusion v1-5

Latent text-to-image diffusion model capable of generating photo-realistic images given a text input.

[DEPLOY](#)[OPEN NOTEBOOK](#)[OVERVIEW](#)[USE CASES](#)[DOCUMENTATION](#)[PRICING](#)

### Overview

The Stable Diffusion model was first published in the paper "[High-Resolution Image Synthesis with Latent Diffusion Models](#)" by Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Bjorn Ommer in CVPR 2022. To enable diffusion models training on limited computational resources while retaining their quality and flexibility, the authors applied them in the latent space of powerful pretrained autoencoders. Training diffusion models on such a representation allows for the first time to reach a near-optimal point between complexity reduction and detail preservation, greatly boosting visual fidelity. The latent diffusion models achieved a new state of the art for image inpainting and highly competitive performance on various tasks, including unconditional image generation, semantic scene synthesis, and super-resolution, while significantly reducing computational requirements compared to pixel-based diffusion models.

Supported by Stability AI and LAION, the 1st version of Stable Diffusion was trained on 512x512 images from a subset of the LAION-5B database. The model uses a frozen CLIP ViT-L/14 text encoder to condition the model on text prompts. With its 860M UNet and 123M text encoder, the model is relatively lightweight and runs on a GPU with at least 10GB VRAM.

The Stable Diffusion V1.5 was trained by runwayml which published the modeling implementation in their Github repository [runwayml/stable-diffusion](#).

### Resource ID

runwayml/stable-diffusion-v1-5

### Tags

Task

Generation

# Stable Diffusion v1-5

Latent text-to-image diffusion model capable of generating images given a text input.

[OVERVIEW](#)[USE CASES](#)[DOCUMENTATION](#)

## Save model

To deploy this model, you will need to save a copy to Model Registry in your Cloud project.

Resource ID

runwayml/stable-diffusion-v1-5 (Latest model)



Save as new model

Creates a new model group and assigns this model as version 1

Save as new version

Saves the model as a version of an existing model

Model name \*

stable\_diffusion



Some locations have been restricted due to a policy set by your organization. [Learn more about restricting locations.](#)

Region \*

us-central1 (Iowa)



## Advanced Options



**SAVE**

CANCEL

## Overview

The **Stable Diffusion** model was first published in the paper ["Stable Diffusion: A Latent Space Model for Image Generation and Inpainting"](#) by Robin Rombach, Andreas Blattmann, Dominik Liebenwein, and Philipp Hennig. The authors applied them in the latent space of powerful pre-trained models. This representation allows for the first time to reach a near-optimal trade-off between visual fidelity preservation, greatly boosting visual fidelity. The latent diffusion model achieves state-of-the-art inpainting and highly competitive performance on various tasks such as image generation, scene synthesis, and super-resolution, while significantly reducing the computational cost compared to state-of-the-art diffusion models.

Supported by Stability AI and LAION, the 1st version of Stable Diffusion was trained on a dataset of 400M images from the LAION-5B database. The model uses a frozen CLIP VQGAN encoder to generate images from text prompts. With its 860M UNet and 123M text encoder, the model requires at least 10GB VRAM.

The Stable Diffusion V1.5 was trained by runwayml which provides a public repository [runwayml/stable-diffusion](#).

# Stable Diffusion v1-5

Latent text-to-image diffusion model capable of generating images given a text input.

**OVERVIEW**

**DEPLOY** OPEN NOTEBOOK

OVERVIEW USE CASES DOCUMENTATION

## Overview

The **Stable Diffusion** model was first published in the paper "Stable Diffusion: A Latent Space Model for Image Generation and Inpainting" by Robin Rombach, Andreas Blattmann, Dominik Reichert, and Philipp Hennig. This model enables diffusion models training on limited computational resources. The authors applied them in the latent space of powerful pre-trained representation models. This representation allows for the first time to reach a near-optimal preservation of visual fidelity. The latent diffusion model can be used for inpainting and highly competitive performance on various tasks such as scene synthesis, and super-resolution, while significantly outperforming other diffusion models.

Supported by Stability AI and LAION, the 1st version of Stable Diffusion uses 860M parameters and is trained on 1.2B images from the LAION-5B database. The model uses a frozen CLIP text encoder and a frozen UNet. With its 860M UNet and 123M text encoder, the model requires at least 10GB VRAM.

The Stable Diffusion V1.5 was trained by runwayml which has a GitHub repository [runwayml/stable-diffusion](#).

**Deploy to endpoint**

1 Define your endpoint    2 Model settings    3 Model monitoring

DEPLOY CANCEL

Create new endpoint  Add to existing endpoint

Endpoint name \*

Location

**i** Some locations have been restricted due to a policy set by your organization. [Learn more about restricting locations.](#)

Region us-central1 (Iowa)

Access

Determines how your endpoint can be accessed. By default, endpoints are available for prediction serving through a REST API. Endpoint access can't be changed after the endpoint is created.

Standard  
Makes the endpoint available for prediction serving through a REST API. AutoML and custom-trained models can be added to standard endpoints.

Private  
Create a private connection to this endpoint using a VPC network and [private services access](#). Only custom-trained and tabular models can be added to private endpoints. [Learn more](#)

▼ ADVANCED OPTIONS

CONTINUE

# Model versions

[stable\\_diffusion](#)

[EDIT DETAILS](#)

Model description

—

Region

us-central1 (Iowa)

Model labels

—

## Versions

Filter Enter a property name



<input type="checkbox"/> Version ID ↓	Alias	Status	Description	Endpoints	Created	Labels
<input type="checkbox"/> 1	default	Ready	—	—	Jun 29, 2023, 7:48:55 AM	

# Model Garden

- 01 Introduction to Model Garden
- 02 Available Model Types
- 03 Working with Gen AI Studio
- 04 Working with Model Registry
- 05 **Intro: Course Use Cases**



# Course Use Cases

03

## Task-Specific Models

Problem focus: text classification

Vertex AI provides multiple ways to solve classification tasks. Task-specific models are an easy way to get started.

04

## Foundation Models

Problem focus: language understanding

The foundation models provide a lot of power just as they are. We can use them to solve many tasks.

05

## Fine-tunable models

Problem focus: Productionalization

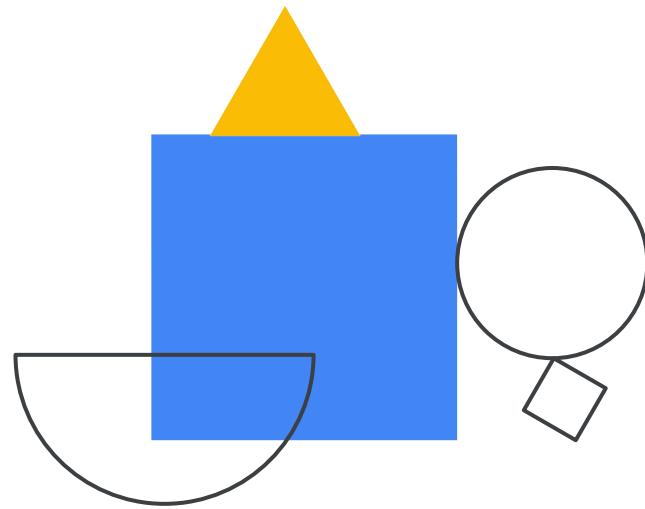
Vertex AI provides us the tools to tune models for our own use cases. We're going to see how to formalize that into a pipeline.

03

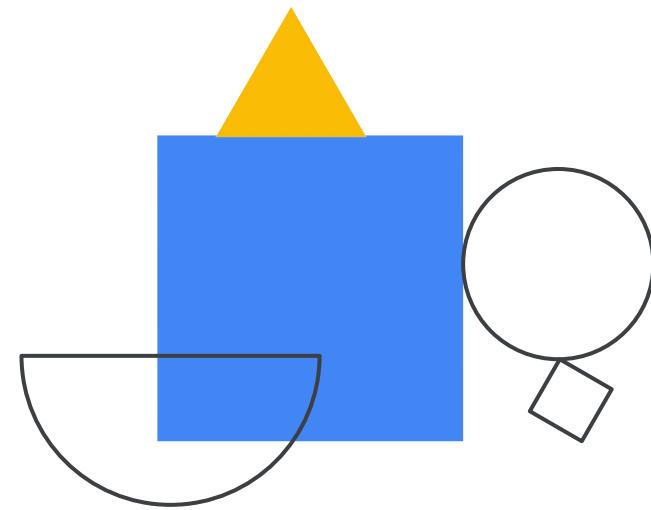


## Task-specific Solutions

# Lab Startup: Text Classification with Model Garden



Complete Task 1.



# Task-specific Solutions

01 **Ways to use task-specific models**

02 Using the Python SDK

03 AutoML

04 Lab: Text Classification with Model Garden



# Task-specific solutions

## Vision

Process images, video, and documents to detect, extract, classify, and enrich.

## Language

Process natural language for sentiment, entities, and translation.

## Tabular

Classification and regression solutions for structured data

# Models-as-a-Service

Vision

Video Intelligence

Natural Language

Translate

Speech-to-Text

Text-to-Speech

Dialogflow

Document AI

Contact Center AI

Product Discovery

Vertex AI Vision

# App Platform

## PPE detector

Identify people and personal protective equipment (PPE).



## Person blur

Mask or blur a person's appearance in video



## Person/vehicle detector

Detects and counts people and vehicles in video.





## Data sinks

**Vision AI Warehouse**

Assets stored on Google Cloud

**BigQuery**

Store structured insights data

**Cloud Storage**

Store results to Cloud Storage

## Pre-trained models

**Person/vehicle detector**

Draw boxes around people and cars

**Person blur**

Anonymize human figures

**Object detector**

Draw boxes around known objects

**Universal input**

1 source

**Person/vehicle detector**

People + vehicles

**Cloud Storage**

my-video-data/processed/

# REST APIs

```
curl "https://language.googleapis.com/v1/documents:analyzeSentiment?key=${API_KEY}" \
-s -X POST \
-H "Content-Type: application/json" \
--data-binary @request.json
```

# Client libraries

```
from google.cloud import language_v1

client = language_v1.LanguageServiceClient()

type_ = language_v1.Document.Type.PLAIN_TEXT
document = {"type_": type_, "content": content}

response = client.analyze_sentiment(request={"document": document})
sentiment = response.document_sentiment
```

# Task-specific Solutions

01 Ways to use task-specific models

02 **Using the Python SDK**

03 AutoML

04 Lab: Text Classification with Model Garden



Filter

- google-cloud-language
  - Overview
  - Changelog
  - Multiprocessing
  - 2.0.0 Migration Guide
  - Language V1
  - Language V1beta2
- google-cloud-life-sciences
- google-cloud-logging
- google-cloud-managed-identities
- google-cloud-media-translation
- google-cloud-memcache
- google-cloud-migrationcenter
- google-cloud-monitoring
- google-cloud-monitoring-dashboards
- google-cloud-monitoring-metrics-scopes
- google-cloud-network-connectivity
- google-cloud-network-management
- google-cloud-network-security
- google-cloud-network-services
- google-cloud-notebooks
- google-cloud-optimization
- google-cloud-orchestration-airflow

Python > Documentation > Reference

Version latest

Was this helpful?

[Send feedback](#)

# Python Client for Natural Language API

[support](#) [stable](#) [pypi v2.10.0](#) [python 3.7 | 3.8 | 3.9 | 3.10 | 3.11](#)

[Natural Language API](#): provides natural language understanding technologies to developers, including sentiment analysis, entity analysis, entity sentiment analysis, content classification, and syntax analysis. This API is part of the larger Cloud Machine Learning API family.

- [Client Library Documentation](#)
- [Product Documentation](#)

## Quick Start

In order to use this library, you first need to go through the following steps:

- [1. Select or create a Cloud Platform project.](#)
- [2. Enable billing for your project.](#)
- [3. Enable the Natural Language API.](#)
- [4. Setup Authentication.](#)

Filter

- google-cloud-language
  - Overview
  - Changelog
  - Multiprocessing
  - 2.0.0 Migration Guide
  - Language V1
  - Language V1beta2
- google-cloud-lite-sciences
- google-cloud-logging
- google-cloud-managed-identities
- google-cloud-media-translation
- google-cloud-memcache
- google-cloud-migrationcenter
- google-cloud-monitoring
- google-cloud-monitoring-dashboards
- google-cloud-monitoring-metrics-scopes
- google-cloud-network-connectivity
- google-cloud-network-management
- google-cloud-network-security
- google-cloud-network-services
- google-cloud-notebooks
- google-cloud-optimization
- google-cloud-orchestration-airflow

Python > Documentation > Reference

Version latest ▾

Was this helpful?

[Send feedback](#)

# Python Client for Natural Language API ▾

[support](#) [stable](#) [pypi v2.10.0](#) [python 3.7 | 3.8 | 3.9 | 3.10 | 3.11](#)

[Natural Language API](#): provides natural language understanding technologies to developers, including sentiment analysis, entity analysis, entity sentiment analysis, content classification, and syntax analysis. This API is part of the larger Cloud Machine Learning API family.

- [Client Library Documentation](#)
- [Product Documentation](#)

## Quick Start

In order to use this library, you first need to go through the following steps:

1. [Select or create a Cloud Platform project.](#)
2. [Enable billing for your project.](#)
3. [Enable the Natural Language API.](#)
4. [Setup Authentication.](#)

Filter

google-cloud-language

- Overview
- Changelog
- Multiprocessing
- 2.0.0 Migration Guide
- Language V1
  - language\_service
    - Overview
    - LanguageServiceAsyncClient
    - LanguageServiceClient
  - types
- Language V1beta2

google-cloud-lite-sciences

google-cloud-logging

google-cloud-managed-identities

google-cloud-media-translation

google-cloud-memcache

google-cloud-migrationcenter

google-cloud-monitoring

google-cloud-monitoring-dashboards

google-cloud-monitoring-metrics-scopes

google-cloud-network-connectivity

google-cloud-network-management

Python > Documentation > Reference

Version latest ▾

Was this helpful?

[Send feedback](#)

# Python Client for Natural Language API

support stable pypi v2.10.0 python 3.7 | 3.8 | 3.9 | 3.10 | 3.11

**Natural Language API:** provides natural language understanding technologies to developers, including sentiment analysis, entity analysis, entity sentiment analysis, content classification, and syntax analysis. This API is part of the larger Cloud Machine Learning API family.

- [Client Library Documentation](#)
- [Product Documentation](#)

## Quick Start

In order to use this library, you first need to go through the following steps:

1. [Select or create a Cloud Platform project.](#)
2. [Enable billing for your project.](#)
3. [Enable the Natural Language API.](#)
4. [Setup Authentication.](#)

Filter

# Analyzing Sentiment

[Send feedback](#)

**Sentiment Analysis** inspects the given text and identifies the prevailing emotional opinion within the text, especially to determine a writer's attitude as positive, negative, or neutral. Sentiment analysis is performed through the `analyzeSentiment` method. For information on which languages are supported by the Natural Language API, see [Language Support](#). For information on how to interpret the `score` and `magnitude` sentiment values included in the analysis, see [Interpreting sentiment analysis values](#).

This section demonstrates a few ways to detect sentiment in a document. For each document, you must submit a separate request.

## Analyzing Sentiment in a String

Here is an example of performing sentiment analysis on a text string sent directly to the Natural Language API:

Protocol    gcloud    Go    Java    Node.js    **Python**    Additional languages

To authenticate to Natural Language, set up Application Default Credentials. For more information, see [Set up authentication for a local development environment](#).

 [Open in Editor](#) [View on GitHub](#) [Feedback](#)

```
from google.cloud import language_v1

def sample_analyze_sentiment(content):
    client = language_v1.LanguageServiceClient()
```

Filter

# Analyzing Sentiment

Send feedback

**Sentiment Analysis** inspects the given text and identifies the prevailing emotional opinion within the text, especially to determine a writer's attitude as positive, negative, or neutral. Sentiment analysis is performed through the `analyzeSentiment` method. For information on which languages are supported by the Natural Language API, see [Language Support](#). For information on how to interpret the `score` and `magnitude` sentiment values included in the analysis, see [Interpreting sentiment analysis values](#).

This section demonstrates a few ways to detect sentiment in a document. For each document, you must submit a separate request.

## Analyzing Sentiment in a String

Here is an example of performing sentiment analysis on a text string sent directly to the Natural Language API:

Protocol    gcloud    Go    Java    Node.js    **Python**    Additional languages

To authenticate to Natural Language, set up Application Default Credentials. For more information, see [Set up authentication for a local development environment](#).

Open in Editor

View on GitHub

Feedback

```
from google.cloud import language_v1  
  
def sample_analyze_sentiment(content):  
    client = language_v1.LanguageServiceClient()
```



# Client

Python > Documentation > Reference

Was this helpful?

[Send feedback](#)

Filter

- google-cloud-language
  - Overview
  - Changelog
  - Multiprocessing
  - 2.0.0 Migration Guide
- Language V1
  - language\_service
    - Overview
    - LanguageServiceAsyncClient
    - LanguageServiceClient**
    - types
  - Language V1beta2
- google-cloud-life-sciences
- google-cloud-logging
- google-cloud-managed-identities
- google-cloud-media-translation
- google-cloud-memcache
- google-cloud-migrationcenter
- google-cloud-monitoring
- google-cloud-monitoring-dashboards
- google-cloud-monitoring-metrics-scopes
- google-cloud-network-connectivity
- google-cloud-network-management

Was this helpful?

On this page

- Properties**
  - transport
- Methods**
  - LanguageServiceClient
  - \_exit\_
  - analyze\_entities
  - analyze\_entity\_sentiment
  - analyze\_sentiment
  - analyze\_syntax
  - annotate\_text
  - classify\_text
  - common\_billing\_account\_path
  - common\_folder\_path
  - common\_location\_path
  - common\_organization\_path
  - common\_project\_path
  - from\_service\_account\_file
  - from\_service\_account\_info
  - from\_service\_account\_json
  - get\_mtls\_endpoint\_and\_cert\_source
  - moderate\_text
  - parse\_common\_billing\_account\_path
  - parse\_common\_folder\_path
  - parse\_common\_location\_path
  - parse\_common\_organization\_path
  - parse\_common\_project\_path

Version latest

```
LanguageServiceClient(*, credentials: Optional[google.auth.credentials.Credentials] = None,
                     transport: Optional[Union[str,
                     google.cloud.language_v1.services.language_service.transports.base.LanguageServiceTransport]] = None,
                     client_options: Optional[Union[google.api_core.client_options.ClientOptions, dict]] = None,
                     client_info: google.api_core.gapic_v1.client_info.ClientInfo =
                     <google.api_core.gapic_v1.client_info.ClientInfo object>)
```

Provides text analysis operations such as sentiment analysis and entity recognition.

## Properties

### transport

Returns the transport used by the client instance.

Returns	
Type	Description
LanguageServiceTransport	The transport used by the client instance.

# analyze\_sentiment



```
analyze_sentiment(request:  
Optional[Union[google.cloud.language_v1.types.language_service.AnalyzeSentimentRequest, dict]] =  
None, *, document: Optional[google.cloud.language_v1.types.language_service.Document] = None,  
encoding_type: Optional[google.cloud.language_v1.types.language_service.EncodingType] = None,  
retry: Union[google.api_core.retry.Retry, google.api_core.gapic_v1.method._MethodDefault] =  
<_MethodDefault._DEFAULT_VALUE: <object object>>, timeout: Union[float, object] =  
<_MethodDefault._DEFAULT_VALUE: <object object>>, metadata: Sequence[Tuple[str, str]] = ())
```

Analyzes the sentiment of the provided text.



```
# This snippet has been automatically generated and should be regarded as a  
# code template only.  
# It will require modifications to work:  
# - It may require correct/in-range values for request initialization.  
# - It may require specifying regional endpoints when creating the service  
#   client as shown in:  
#   https://googleapis.dev/python/google-api-core/latest/client_options.html  
from google.cloud import language_v1  
  
def sample_analyze_sentiment():  
    # Create a client  
    client = language_v1.LanguageServiceClient()  
  
    # Initialize request argument(s)  
    document = language_v1.Document()  
    document.content = "content_value"  
  
    request = language_v1.AnalyzeSentimentRequest(  
        document=document,  
    )
```

## On this page

Properties

transport

Methods

LanguageServiceClient

\_exit\_\_

analyze\_entities

analyze\_entity\_sentiment

analyze\_sentiment

analyze\_syntax

annotate\_text

classify\_text

common\_billing\_account\_path

common\_folder\_path

common\_location\_path

common\_organization\_path

common\_project\_path

from\_service\_account\_file

from\_service\_account\_info

from\_service\_account\_json

get\_mtls\_endpoint\_and\_cert\_source

moderate\_text

parse\_common\_billing\_account\_path

parse\_common\_folder\_path

parse\_common\_location\_path

parse\_common\_organization\_path

parse\_common\_project\_path

# Python client: Analyze sentiment

```
from google.cloud import language_v1

client = language_v1.LanguageServiceClient()

type_ = language_v1.Document.Type.PLAIN_TEXT
document = {"type_": type_, "content": "Google Cloud is the best!"}

response = client.analyze_sentiment(request={"document": document})

sentiment = response.document_sentiment
```

# Python client: Analyze sentiment

```
from google.cloud import language_v1

client = language_v1.LanguageServiceClient()

type_ = language_v1.Document.Type.PLAIN_TEXT
document = {"type_": type_, "content": "Google Cloud is the best!"}

response = client.analyze_sentiment(request={"document": document})

sentiment = response.document_sentiment
```

# Python client: Analyze sentiment

```
from google.cloud import language_v1

client = language_v1.LanguageServiceClient()

type_ = language_v1.Document.Type.PLAIN_TEXT
document = {"type_": type_, "gcs_content_uri": "gs://my-bucket/my-text-object"}
```

```
response = client.analyze_sentiment(request={"document": document})

sentiment = response.document_sentiment
```

# Python client: Analyze sentiment

```
from google.cloud import language_v1

client = language_v1.LanguageServiceClient()

type_ = language_v1.Document.Type.PLAIN_TEXT
document = {"type_": type_, "content": "Google Cloud is the best!"}

response = client.analyze_sentiment(request={"document": document})

sentiment = response.document_sentiment
```

# Python client: Analyze sentiment

```
from google.cloud import language_v1

client = language_v1.LanguageServiceClient()

type_ = language_v1.Document.Type.PLAIN_TEXT
document = {"type_": type_, "content": "Google Cloud is the best!"}

response = client.analyze_sentiment(request={"document": document})

sentiment = response.document_sentiment
```

# Python client: Analyze sentiment

```
from google.cloud import language_v1

client = language_v1.LanguageServiceClient()

type_ = language_v1.Document.Type.PLAIN_TEXT
document = {"type_": type_, "content": "Google Cloud is the best!"}

response = client.analyze_sentiment(request={"document": document})

sentiment = response.document_sentiment
```

# Task-specific Solutions

01 Ways to use task-specific models

02 Using the Python SDK

03 **AutoML**

04 Lab: Text Classification with Model Garden



# Natural Language API: Classify Text

## Content Categories

```
/Adult  
/Arts & Entertainment/Celebrities & Entertainment News  
/Arts & Entertainment/Other  
/Arts & Entertainment/Comics & Animation/Anime & Manga  
/Arts & Entertainment/Comics & Animation/Cartoons  
/Arts & Entertainment/Comics & Animation/Comics  
/Arts & Entertainment/Comics & Animation/Other  
/Arts & Entertainment/Entertainment Industry/Film & TV Industry  
/Arts & Entertainment/Entertainment Industry/Recording Industry  
/Arts & Entertainment/Entertainment Industry/Other  
/Arts & Entertainment/Events & Listings/Bars, Clubs & Nightlife  
/Arts & Entertainment/Events & Listings/Concerts & Music Festivals  
/Arts & Entertainment/Events & Listings/Event Ticket Sales  
/Arts & Entertainment/Events & Listings/Expos & Conventions  
/Arts & Entertainment/Events & Listings/Film Festivals  
/Arts & Entertainment/Events & Listings/Food & Beverage Events  
/Arts & Entertainment/Events & Listings/Live Sporting Events  
/Arts & Entertainment/Events & Listings/Movie Listings & Theater Showtimes  
/Arts & Entertainment/Events & Listings/Other  
/Arts & Entertainment/Fun & Trivia/Flash-Based Entertainment  
/Arts & Entertainment/Fun & Trivia/Fun Tests & Silly Surveys  
/Arts & Entertainment/Fun & Trivia/Other  
/Arts & Entertainment/Humor/Funny Pictures & Videos  
/Arts & Entertainment/Humor/Live Comedy  
/Arts & Entertainment/Humor/Political Humor  
/Arts & Entertainment/Humor/Spoofs & Satire  
/Arts & Entertainment/Humor/Other  
/Arts & Entertainment/Movies/Action & Adventure Films  
/Arts & Entertainment/Movies/Animated Films  
/Arts & Entertainment/Movies/Bollywood & South Asian Films  
/Arts & Entertainment/Movies/Classic Films  
/Arts & Entertainment/Movies/Comedy Films  
/Arts & Entertainment/Movies/Cult & Indie Films
```



# Natural Language API: Classify Text

“

My waiter, Robert, provided an excellent experience.

The lobster eggs benedict was delicious! My compliments to the chef.

I would have liked to see the market price for the lobster before purchase, though.  
Please add that online!

## Natural Language API: Classify Text

Food/Meat & Seafood	0.92
Food/Breakfast Foods	0.84
Hospitality Industry/Food Service	0.78

# AutoML Text

“

My waiter, Robert, provided an excellent experience.

The lobster eggs benedict was delicious! My compliments to the chef.

I would have liked to see the market price for the lobster before purchase, though. Please add that online!

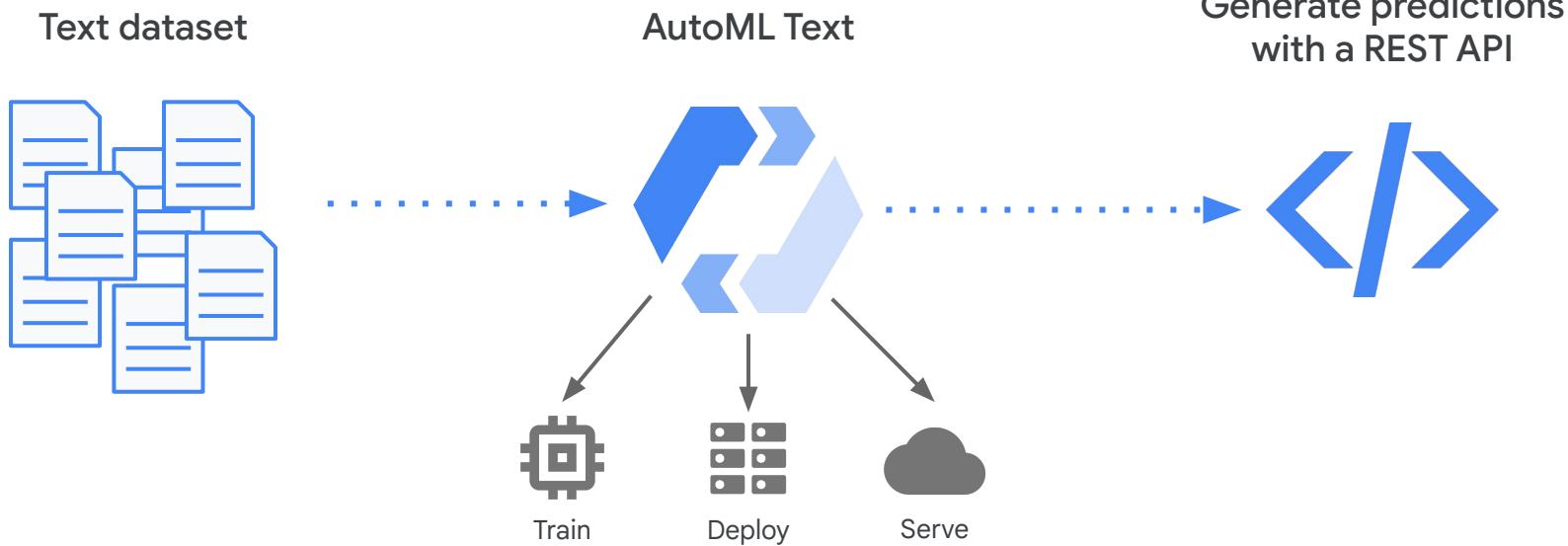
## Natural Language API: Classify Text

Food/Meat & Seafood	0.92
Food/Breakfast Foods	0.84
Hospitality Industry/Food Service	0.78

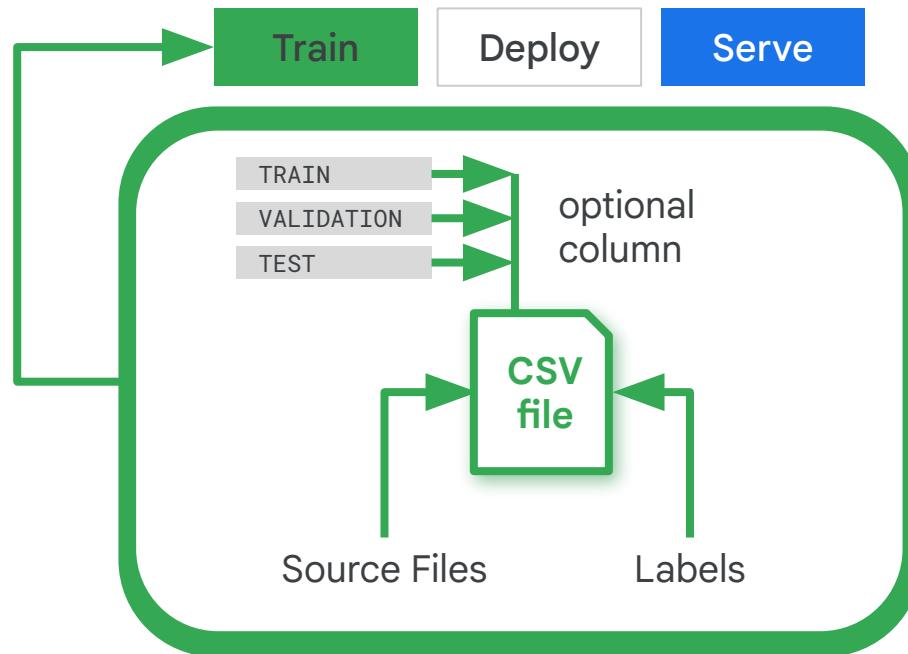
## AutoML Text

Great Service	0.95
Happy Customer	0.92
Suggestion	0.8

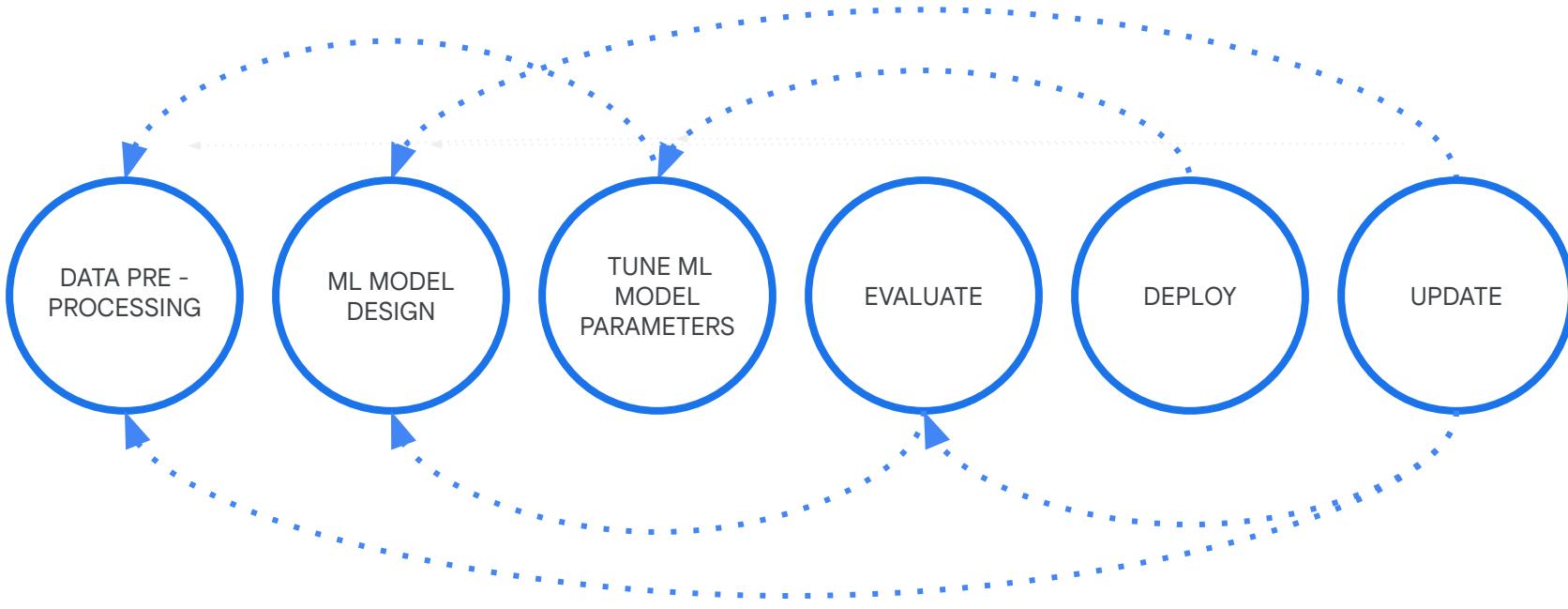
# AutoML



# AutoML Prepared Dataset Format



# Codeless model building with AutoML



# AutoML E2E

Tabular Workflow for End-to-End AutoML is the complete AutoML pipeline for classification and regression tasks.

[VIEW API CODE](#)[OVERVIEW](#)[USE CASES](#)[DOCUMENTATION](#)[PRICING](#)

## Overview

Tabular Workflow for End-to-End AutoML is the complete AutoML pipeline for classification and regression tasks. It is similar to the [AutoML API](#), but allows you to choose what to control and what to automate. Instead of having controls for the whole pipeline, you have controls for every step in the pipeline. These pipeline controls include:

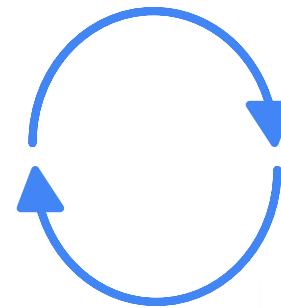
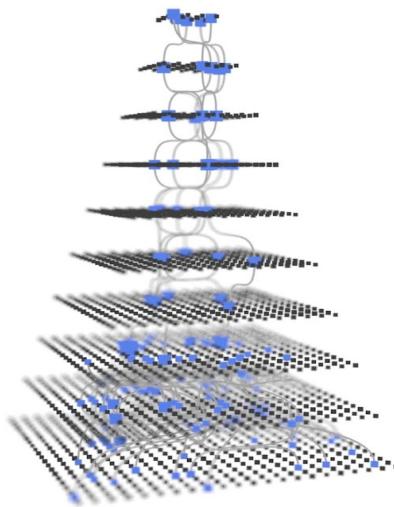
- Data splitting
- Feature engineering
- Architecture search
- Model training
- Model ensembling
- Model distillation

## Algorithm

Our initial efforts of [neural architecture search](#) have enabled breakthroughs in computer vision with [NasNet](#), and evolutionary methods such as [AmoebaNet](#) and hardware-aware mobile vision architecture [MNasNet](#) further show the benefit of these learning-to-learn methods. Recently, we applied a learning-based approach to tabular data, creating a scalable end-to-end AutoML solution.

# AutoML is built with Neural Architecture Search

Controller: proposes ML models



Iterate to  
find the  
most  
accurate  
model

Train & evaluate models



# Task-specific Solutions

01 Ways to use task-specific models

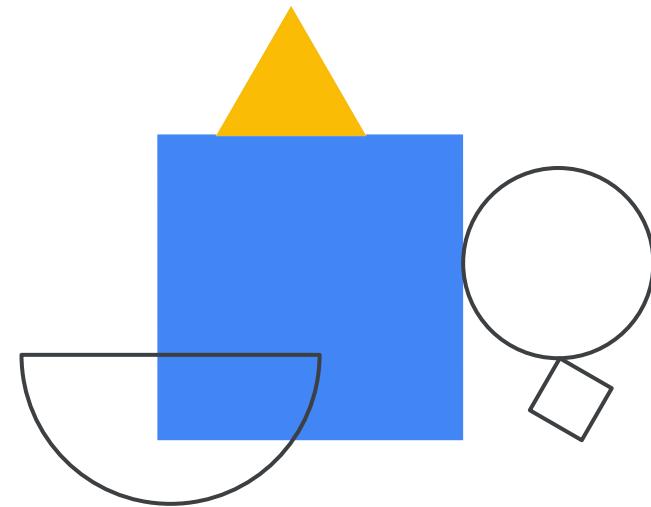
02 Using the Python SDK

03 AutoML

04 **Lab: Text Classification with Model Garden**



**Lab:**  
**Text Classification**  
**with Model Garden**



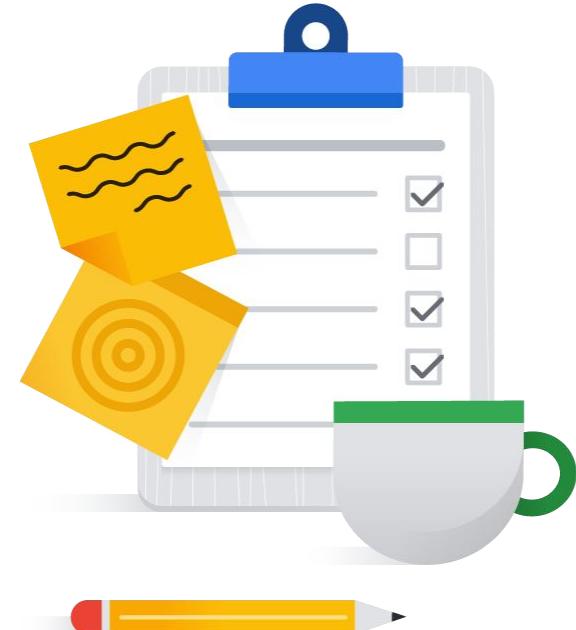




# Foundation Models

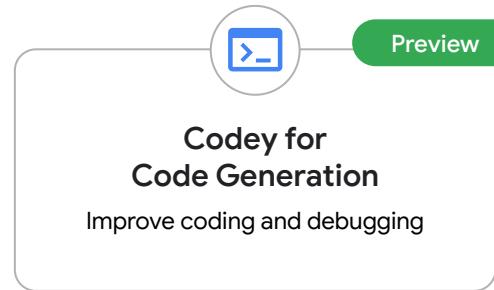
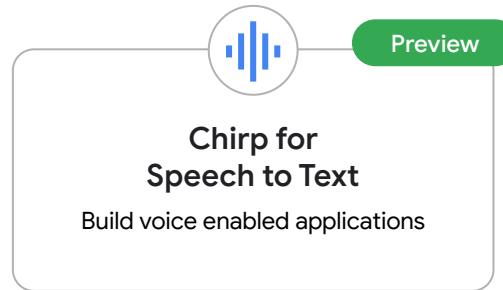
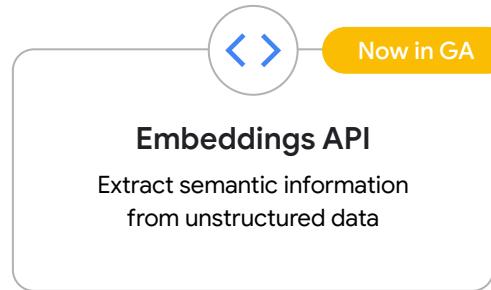
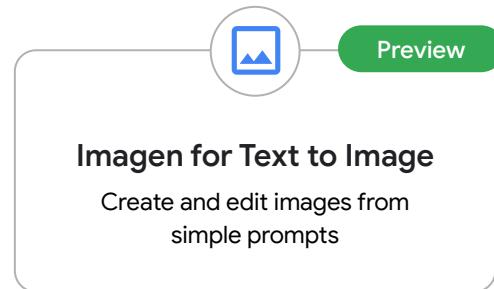
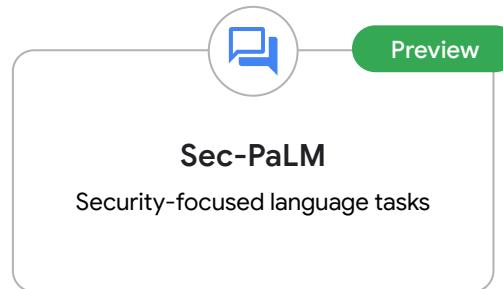
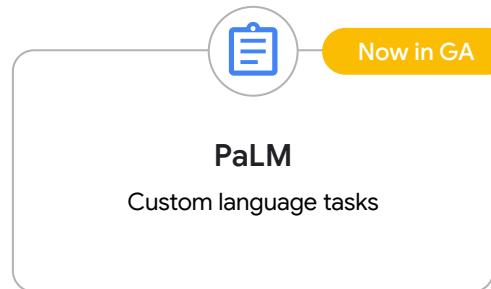
# Model Garden

- 01 **Foundation Models**
- 02 PaLM in Gen AI Studio
- 03 PaLM API
- 04 Embeddings API
- 05 Lab: Use the PaLM API to Cluster Products Based on Descriptions



# Foundation Models

Across a variety of model sizes to address use cases



# Other foundation models

## Image Generation

- Stable Diffusion
- Stable Diffusion Inpainting
- InstructPix2Pix
- ControlNet

## Image Processing

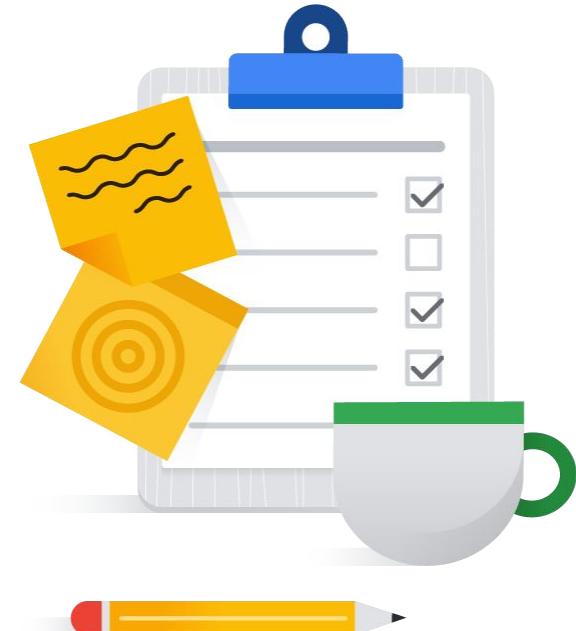
- PaLI zero-shot
- BLIP
- BLIP 2
- CLIP
- OWL-ViT
- ViT GPT2
- LayoutLM for VQA

## Natural Language

- BERT
- T5-FLAN

# Model Garden

- 01 Foundation Models
- 02 **PaLM in Gen AI Studio**
- 03 PaLM API
- 04 Embeddings API
- 05 Lab: Use the PaLM API to Cluster Products Based on Descriptions



# Model Card

## PaLM 2 for Text

Fine-tuned to follow natural language instructions and is suitable for a variety of language tasks, such as: classification, extraction, summarization and content generation.

[OPEN PROMPT DESIGN](#) [VIEW API CODE](#)

[OVERVIEW](#) [USE CASES](#) [DOCUMENTATION](#)

### Overview

**text-bison** is the name of the PaLM 2 for text large language model that understands and generates language. It's a foundation model that performs well at a variety of natural language tasks such as sentiment analysis, entity extraction, and content creation. The type of content that **text-bison** can create includes document summaries, answers to questions, and labels that classify content.

The PaLM 2 for text is ideal for tasks that can be completed with one API response, without the need for continuous conversation. For text tasks that require back-and-forth interactions, use the [PaLM 2 for chat](#).

### Use cases

- **Summarization:** Create a shorter version of a document that incorporates pertinent information from the original text. For example, you might want to summarize a chapter from a textbook. Or, you could create a succinct product description from a long paragraph that describes the product in detail.
- **Question answering:** Provide answers to questions in text. For example, you might automate the creation

# Gen AI Studio

## Generative AI Studio

Generative AI Studio lets you quickly test and customize generative AI models so you can leverage their capabilities in your applications. [Learn more](#)

 [FEEDBACK ON GENERATIVE AI STUDIO](#)

 [VIEW DOCUMENTATION](#)

### Language

Test, tune, and deploy generative AI language models. Access the PaLM API for Chat for code generation, content generation, chat, summarization, and more.

[OPEN](#)

### Speech

Convert speech into text or synthesize speech from text using Google's Universal Speech Model (USM)

[OPEN](#)

# Gen AI Studio - Language

## Language

[GET STARTED](#)[MY PROMPTS](#)[TUNING](#) [PREVIEW](#)

### Get started



#### Design and test your own prompts

Design prompts for tasks relevant to your business use case including code generation. Take a tutorial on [creating effective text prompts](#).

[+ TEXT PROMPT](#)[+ CODE PROMPT](#)

#### Start a conversation

Have a freeform chat with the model, which tracks what was previously said and responds based on context. Take a tutorial on [designing chat prompts](#).

[+ TEXT CHAT](#)[+ CODE CHAT](#)

#### Tune a model

Tune a model so it's better equipped for your use case, then deploy to an endpoint to get predictions or test it in prompt design. Take a tutorial on [creating a tuned model](#).

[+ CREATE TUNED MODEL](#)

# Prompt Examples

**Chat**

Kindergarten  
Your name is Mil  
astronomer who  
about the solar s  
short sentences.

**Summarization**

**Freeform**  
Support rep c  
You are a customer  
manager and would  
see what your team  
are about.

**Classification**

**Freeform**  
Sentiment analy  
You would like to see  
write about certain pe  
articles and would lik  
certain person is writ

**Writing**

**Freeform**  
Marketing g  
You work in Goo  
marketing team  
create marketing  
Pixel 7 Pro. You

**Extraction**

**Structured**  
Pixel Techn  
Generate techni  
from text of a P  
one-shot.

**Ideation**

**Freeform**  
Reading comprehen  
Your child is preparing for SAT verbal  
exam and needs more practice in  
reading comprehension.

**Freeform**  
Generate memes  
Generate memes based on a certain  
topic.

**OPEN**

**OPEN**

**OPEN**

**OPEN**

**OPEN**

**OPEN**

# Gen AI Studio - Language

## Language

[GET STARTED](#)[MY PROMPTS](#)[TUNING](#) [PREVIEW](#)

### Get started



#### Design and test your own prompts

Design prompts for tasks relevant to your business use case including code generation. Take a tutorial on [creating effective text prompts](#).

[+ TEXT PROMPT](#)[+ CODE PROMPT](#)

#### Start a conversation

Have a freeform chat with the model, which tracks what was previously said and responds based on context. Take a tutorial on [designing chat prompts](#).

[+ TEXT CHAT](#)[+ CODE CHAT](#)

#### Tune a model

Tune a model so it's better equipped for your use case, then deploy to an endpoint to get predictions or test it in prompt design. Take a tutorial on [creating a tuned model](#).

[+ CREATE TUNED MODEL](#)

# Gen AI Studio - Freeform Prompt

Untitled prompt

FREEFORM STRUCTURED

SAVE VIEW CODE

Prompt

Write a prompt and then click Submit

Model: text-bison@001

Temperature: 0.2

Token limit: 256

Top-K: 40

Top-P: 0.8

Safety filter threshold: Block few

Response

The model will generate a response after you click Submit

Markdown

Model may display inaccurate or offensive information that doesn't represent Google's view. Google models currently only support English.

REPORT INAPPROPRIATE RESPONSES

Content processed through Vertex AI is assessed against a list of safety attributes. Confidence scores for these attributes are returned in API responses but are not visible in the UI.

▼ **Prompt**

Why would I choose precision versus recall as my evaluation metric?



## Response



Markdown

Precision and recall are two important metrics for evaluating the performance of a classification model. Precision is the fraction of predicted positive instances that are actually positive, while recall is the fraction of actual positive instances that are predicted positive.

Precision is important because it measures how well the model avoids false positives. A false positive is an instance that is predicted to be positive when it is actually negative. False positives can be costly, as they can lead to wasted time and resources.

Recall is important because it measures how well the model identifies positive instances. A missed positive is an instance that is actually positive but is predicted to be negative. Missed positives can be costly, as they can lead to missed opportunities.

The choice of which metric to use depends on the specific application. If the cost of false positives is high, then precision is more important. If the cost of missed positives is high, then recall is more important.

In some cases, it may be important to consider both precision and recall. This can be done by using the F1 score, which is a weighted average of precision and recall. The F1 score is calculated as follows:

$$\text{F1} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

▼ **Prompt**

You are a senior machine learning practitioner training a junior staff member. Provide answers to questions that include practical use cases to help the new employee learn.



Why would I choose precision versus recall as my evaluation metric?

**Response** Markdown

Precision and recall are two important metrics for evaluating the performance of a machine learning model. Precision measures the proportion of positive predictions that are actually correct, while recall measures the proportion of actual positives that are predicted correctly.

In some cases, you may want to prioritize precision over recall. For example, if you are building a model to detect fraud, you may want to make sure that you are not falsely accusing people of fraud. In this case, you would want to have a high precision score, even if it means that you have a lower recall score.

In other cases, you may want to prioritize recall over precision. For example, if you are building a model to diagnose a disease, you may want to make sure that you do not miss any cases of the disease. In this case, you would want to have a high recall score, even if it means that you have a lower precision score.

The choice of which metric to prioritize depends on the specific application of the machine learning model.

**Sources** 

1. [https://github.com/muntazirabidi/machine\\_learning\\_tutorials](https://github.com/muntazirabidi/machine_learning_tutorials) 

## ▼ Prompt

You are a senior machine learning practitioner at a travel company that is training a junior staff member. Provide answers to questions that include practical use cases related to the industry to help the new employee learn.



Why would I choose precision versus recall as my evaluation metric?

## Response



Markdown

Precision and recall are two important metrics for evaluating the performance of a machine learning model. Precision measures the proportion of positive predictions that are actually correct, while recall measures the proportion of actual positives that are predicted correctly.

In the context of travel, precision and recall could be used to evaluate a model that predicts whether a customer will book a hotel room. Precision would measure the proportion of customers who were predicted to book a hotel room who actually did, while recall would measure the proportion of customers who actually booked a hotel room who were predicted to do so.

Precision and recall are often used together to evaluate a model's performance. A common approach is to plot precision against recall, and to choose the point on the curve that represents the best trade-off between precision and recall. This trade-off will depend on the specific application. For example, a travel company might be more interested in maximizing precision, as they would want to minimize the number of customers who are predicted to book a hotel room but who do not actually do so. On the other hand, a hotel might be more interested in maximizing recall, as they would want to ensure that all customers who are likely to book a hotel room are predicted to do so.

In addition to precision and recall,

### Sources

- [https://github.com/muntazirabidi/machine\\_learning\\_tutorials](https://github.com/muntazirabidi/machine_learning_tutorials)

We want your [feedback](#).

**Model** text-bison@001

Temperature 0.2

Token limit 256

Top-K 40

Top-P 0.8

Safety filter threshold Block few

**SUBMIT** **RESET PARAMETERS**

## Response

Markdown

Precision and recall are two important metrics for evaluating the performance of a machine learning model. Precision measures the proportion of positive predictions that are actually correct, while recall measures the proportion of actual positives that are predicted correctly.

In the context of travel, precision and recall could be used to evaluate a model that predicts whether a customer will book a hotel room. Precision would measure the proportion of customers who were predicted to book a hotel room who actually did, while recall would measure the proportion of customers who actually booked a hotel room who were predicted to do so.

Precision and recall are often used together to evaluate a model's performance. A common approach is to plot the precision-recall curve, which shows the trade-off between precision and recall for different thresholds. The area under the precision-recall curve (AUC) is a measure of the overall performance of the model.

There are a few reasons why you might choose precision versus recall as your evaluation metric. First, the choice of metric depends on the specific application. For example, if you are more concerned about false positives than false negatives, then you would want to prioritize precision. On the other hand, if you are more concerned about false negatives than false positives, then you would want to prioritize recall.

### Sources

1. [https://github.com/muntazirabidi/machine\\_learning\\_tutorials](https://github.com/muntazirabidi/machine_learning_tutorials) 

# Gen AI Studio - Freeform Prompt

Untitled prompt

FREEFORM **STRUCTURED**

**Context** ?

You are a senior machine learning practitioner at a travel company that is training a junior staff member. Provide answers to questions that include practical use cases related to the industry to help the new employee learn.

Why would I choose precision versus recall as my evaluation metric?

**Examples** ? **ADD INPUT COLUMN**

**INPUT** Add an input prefix      **OUTPUT** Add an output prefix

(+) Write an input to add a new row      (+) Write an output to add a new row

**Test** ? **ADD TO EXAMPLES**

**INPUT** Add an input prefix      **OUTPUT** Add an output prefix **Markdown**

Write an input      Model responses will appear here

## Context

You are a senior machine learning practitioner at a travel company that is training a junior staff member. Provide answers to questions that include practical use cases related to the industry to help the new employee learn.

## Examples

 ADD INPUT COLUMN

## Test

 ADD TO EXAMPLES

INPUT Add an input prefix

OUTPUT Add an output prefix  Markdown

Why would I choose precision versus  
recall as my evaluation metric?



Model responses will appear here

## ▼ Test ?

[ADD TO EXAMPLES](#)

INPUT Add an input prefix

Why would I choose precision versus recall as my evaluation metric?

OUTPUT Add an output prefix



Markdown

Precision and recall are two important metrics for evaluating the performance of a machine learning model. Precision is the fraction of positive predictions that are actually correct, while recall is the fraction of actual positives that are predicted correctly.

In the context of travel, precision and recall could be used to evaluate a model that predicts whether a customer will book a hotel room. Precision would measure the fraction of customers who were predicted to book a room who actually did, while recall would measure the fraction of customers who actually booked a room who were predicted to do so.

Precision and recall are often used together in a trade-off, where a higher value of one metric is achieved at the expense of a lower value of the other. For example, a model with high precision might only predict that a small number of customers will book a room, but those customers would be very likely to actually do so. A model with high recall, on the other hand, might predict that a large number of customers will book a room, but some of those customers would not actually do so.

The choice of which metric to prioritize depends on the specific application. In some cases, it may be more important to have a high precision, while in other cases it may be more important to have a high

**Examples**  ADD INPUT COLUMN**INPUT** Add an input prefix

Why would I choose precision versus recall as my evaluation metric?

**OUTPUT** Add an output prefix

Precision and recall are two important metrics for evaluating the performance of a machine learning model. Precision measures the proportion of positive predictions that are actually correct, while recall measures the proportion of actual positives that are predicted correctly.

In the context of travel, precision and recall could be used to evaluate a model that predicts whether a customer will book a hotel room. Precision would measure the proportion of customers who were predicted to book a hotel room who actually did, while recall would measure the proportion of customers who actually booked a hotel room who were predicted to do so.

Precision and recall are often used together to evaluate a model's performance. A common approach is to plot precision against recall, and to choose the point on the curve that represents the best trade-off between precision and recall. This trade-off will depend on the specific application. For example, a travel company might be more interested in maximizing precision, as they would want to minimize the number of customers who are predicted to book a hotel room but who do not actually do so. On the other hand, a hotel might be more interested in maximizing recall, as they would want to ensure that all customers who are likely to book a hotel room are predicted to do so.

## ▼ Test ?

[ADD TO EXAMPLES](#)

INPUT Add an input prefix

When should you use mean squared error as a loss function rather than mean absolute error?



OUTPUT Add an output prefix



Markdown

Mean squared error (MSE) and mean absolute error (MAE) are two common loss functions used in machine learning. MSE is the average of the squared differences between the predicted values and the actual values. MAE is the average of the absolute differences between the predicted values and the actual values.

MSE is more sensitive to outliers than MAE. This means that a single large error can have a big impact on the MSE. MAE is less sensitive to outliers, so it is less likely to be affected by a single large error.

MSE is a good choice when the data is normally distributed. MAE is a good choice when the data is not normally distributed.

In the context of travel, MSE might be a good choice for a model that predicts the price of a hotel room. MAE might be a good choice for a model that predicts the number of days a customer will stay at a hotel.

The best way to choose between MSE and MAE is to experiment with both loss functions and see which one gives better results on your data.

[Untitled prompt](#)
FREEFORM
STRUCTURED
[SAVE](#)
[VIEW CODE](#)

**Context** [?](#)

You are a senior machine learning practitioner at a travel company that is training a junior staff member. Provide answers to questions that include practical use cases related to the industry to help the new employee learn.

**Model** [text-bison@001](#)

**Temperature** [0.2](#)

**Token limit** [256](#)

**Top-K** [40](#)

**Top-P** [0.8](#)

**Safety filter threshold** [Block few](#)

[ADD INPUT COLUMN](#)
[ADD TO EXAMPLES](#)

**INPUT** Add an input prefix
**OUTPUT** Add an output prefix
[Markdown](#)

When should you use mean squared error as a loss function rather than mean absolute error?

Mean squared error (MSE) and mean absolute error (MAE) are two common loss functions used in machine learning. MSE is the average of the squared differences between the predicted values and the actual values. MAE is the average of the absolute differences between the predicted values and the actual values.

MSE is more sensitive to outliers than MAE. This means that a single large error can have a big impact on the MSE. MAE is less sensitive to outliers, so it is less likely to be affected by a single large error.

[SUBMIT](#)
[RESET PARAMETERS](#)

[REPORT INAPPROPRIATE RESPONSES](#)

Content processed through Vertex AI is assessed

Language

GET STARTED **MY PROMPTS** TUNING PREVIEW

My Prompts [+ CREATE PROMPT](#) [+ CREATE CHAT PROMPT](#)

**Filter** Filter prompts

Prompt name

[Senior ML Practitioner](#)

[Senior ML Practitioner](#)

FREEFORM

STRUCTURED

## Context

You are a senior machine learning practitioner at a travel company that is training a junior staff member. Provide answers to questions that include practical use cases related to the industry to help the new employee learn.

## Examples

ADD INPUT COLUMN

INPUT Add an input prefix

OUTPUT Add an output prefix

Why would I choose precision versus recall as my evaluation metric?

Precision and recall are two important metrics for evaluating the performance of a machine learning model. Precision measures the proportion of positive predictions that are actually correct, while recall measures the proportion of actual positives that are predicted correctly.



In the context of travel, precision and recall could be

[←](#) Senior ML Practitioner 
FREEFORM STRUCTURED
SAVE

**Context**

You are a senior machine learning practitioner at a travel company that is training a junior staff member. Provide answers to questions that include practical use cases related to the industry to help the new employee learn.

We want your [feedback](#).

**Model** `text-bison@001`

**Temperature** 0.2

**Token limit** 1024 256

**Top-K** 40

**Top-P** 0.8

**Safety filter threshold** Block few

**Examples** 
 ADD INPUT COLUMN

INPUT Add an input prefix
OUTPUT Add an output prefix

Why would I choose precision versus recall as my evaluation metric?

Precision and recall are two important metrics for evaluating the performance of a machine learning model. Precision measures the proportion of positive predictions that are actually correct, while recall measures the proportion of actual positives that are predicted correctly.

In the context of travel, precision and recall could be used to evaluate a model that predicts whether a customer will book a hotel room. Precision would

SUBMIT
 RESET PARAMETERS

[Senior ML Practitioner](#)[Context](#)

You are a senior machine learning practitioner at a travel company that is training a junior machine learning model. You have access to a dataset of travel reviews and want to use it to train a model that can predict the sentiment of new reviews. You also have some sample reviews that you want to use to test the model's performance.

[Examples](#)

INPUT Add an input prefix

Why would I choose precision versus recall as my evaluation metric?

[View code](#)[PYTHON](#)[PYTHON COLAB](#)[CURL](#)

Use this script to request a model response in your application.

1. Set up the [Vertex AI SDK for Python](#)
2. Use the following code in your application to request a model response

```
import vertexai
from vertexai.language_models import TextGenerationModel

vertexai.init(project="qwiklabs-gcp-00-6860773f99f8", location="us-central1")
parameters = {
    "temperature": 0.2,
    "max_output_tokens": 256,
    "top_p": 0.8,
    "top_k": 40
}
model = TextGenerationModel.from_pretrained("text-bison@001")
response = model.predict(
    """
    You are a senior machine learning practitioner at a travel company that is training a junior machine learning model. You have access to a dataset of travel reviews and want to use it to train a model that can predict the sentiment of new reviews. You also have some sample reviews that you want to use to test the model's performance.
    """
)
print(f"Response from Model: {response.text}")
```

# Model Garden

- 01 Foundation Models
- 02 PaLM in Gen AI Studio
- 03 **PaLM API**
- 04 Embeddings API
- 05 Lab: Use the PaLM API to Cluster Products Based on Descriptions



# PaLM Python API

```
import vertexai
from vertexai.language_models import TextGenerationModel

vertexai.init(project="YOUR-PROJECT-ID", location="us-central1")

parameters = {
    "temperature": 0.2,
    "max_output_tokens": 256,
    "top_p": 0.8,
    "top_k": 40
}

...
```

# PaLM Python API

```
import vertexai
from vertexai.language_models import TextGenerationModel

vertexai.init(project="YOUR-PROJECT-ID", location="us-central1")

parameters = {
    "temperature": 0.2,
    "max_output_tokens": 256,
    "top_p": 0.8,
    "top_k": 40
}

...
```

# Model (Hyper)Parameters

```
import vertexai
from vertexai.language_models import TextGenerationModel

vertexai.init(project="YOUR-PROJECT-ID", location="us-central1")

parameters = {
    "temperature": 0.2,
    "max_output_tokens": 256,
    "top_p": 0.8,
    "top_k": 40
}

...
```

# Model

```
...  
  
model = TextGenerationModel.from_pretrained("text-bison@001")  
  
response = model.predict(  
    """You are a senior machine learning practitioner at a travel company that is  
    training a junior staff member. Provide answers to questions that include practical use  
    cases related to the industry to help the new employee learn.  
  
    Why would I choose precision versus recall as my evaluation metric?  
    """,  
    **parameters  
)  
  
print(f"Response from Model: {response.text}")
```

# Prediction

```
...  
  
model = TextGenerationModel.from_pretrained("text-bison@001")  
  
response = model.predict(  
    """You are a senior machine learning practitioner at a travel company that is  
    training a junior staff member. Provide answers to questions that include practical use  
    cases related to the industry to help the new employee learn.  
  
    Why would I choose precision versus recall as my evaluation metric?  
    """,  
    **parameters  
)  
  
print(f"Response from Model: {response.text}")
```

# Response

```
...  
  
model = TextGenerationModel.from_pretrained("text-bison@001")  
  
response = model.predict(  
    """You are a senior machine learning practitioner at a travel company that is  
    training a junior staff member. Provide answers to questions that include practical use  
    cases related to the industry to help the new employee learn.  
  
    Why would I choose precision versus recall as my evaluation metric?  
    """,  
    **parameters  
)  
  
print(f"Response from Model: {response.text}")
```

# Custom Examples

```
...  
response = model.predict()  
"""You are a senior machine learning practitioner at a travel company that is  
training a junior staff member. Provide answers to questions that include practical use  
cases related to the industry to help the new employee learn.
```

input: Why would I choose precision versus recall as my evaluation metric?

output: Precision and recall are two important metrics for evaluating the performance of a machine learning model. Precision measures the proportion of positive predictions that are actually correct, while recall measures the proportion of actual positives that are predicted correctly. ...

input: When should you use mean squared error as a loss function rather than mean absolute error?

output:

```
"""  
    **parameters  
)
```

# Custom Examples

```
...  
response = model.predict()  
"""You are a senior machine learning practitioner at a travel company that is  
training a junior staff member. Provide answers to questions that include practical use  
cases related to the industry to help the new employee learn.
```

input: Why would I choose precision versus recall as my evaluation metric?  
output: Precision and recall are two important metrics for evaluating the performance of  
a machine learning model. Precision measures the proportion of positive predictions that  
are actually correct, while recall measures the proportion of actual positives that are  
predicted correctly. ...

input: When should you use mean squared error as a loss function rather than mean  
absolute error?

output:

```
"""  
    **parameters  
)
```

# Model Garden

- 01 Foundation Models
- 02 PaLM in Gen AI Studio
- 03 PaLM API
- 04 **Embeddings API**
- 05 Lab: Use the PaLM API to Cluster Products Based on Descriptions



# Embeddings Model

## Embeddings for text

Text embedding is an important NLP technique that converts textual data into numerical vectors that can be processed by machine learning algorithms, especially large models. These vector representations are designed to capture the semantic meaning and context of the words they represent.

[VIEW API CODE](#)[OVERVIEW](#)[USE CASES](#)[DOCUMENTATION](#)[PRICING](#)

### Overview

Text embedding is a NLP technique that converts textual data into numerical vectors that can be processed by machine learning algorithms, especially large models. These vector representations are designed to capture the semantic meaning and context of the words they represent.

### Use cases

- **Semantic Search:** Text embeddings can be used to represent both the user's query and the universe of documents in a high-dimensional vector space. Documents that are more semantically similar to the user's query will have a shorter distance in the vector space, and can be ranked higher in the search results.
- **Text Classification:** Training a model that maps the text embeddings to the correct category labels (e.g., cat vs. dog, spam vs. not spam). Once the model is trained, it can be used to classify new text inputs into one or more categories based on their embeddings.
- And use cases such as clustering, anomaly detection, sentiment analysis, and more.

### Resource ID

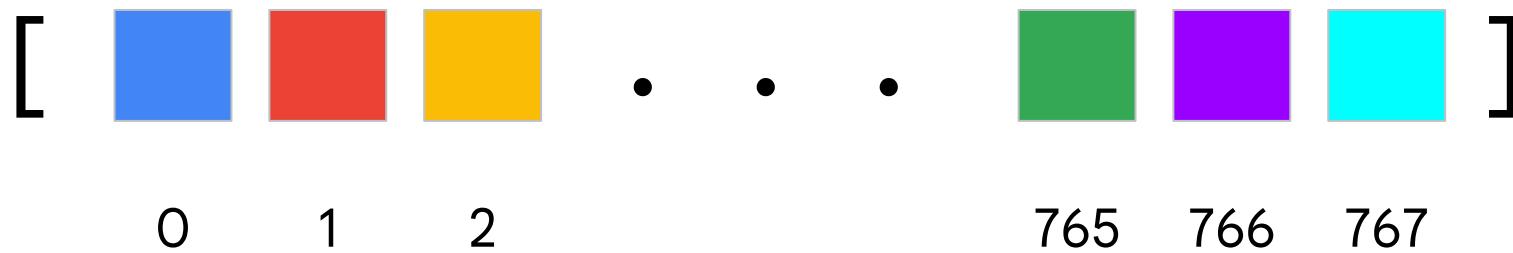
textembedding-gecko@001

### Tags

Task

Embedding

# Embeddings Vector



# Embeddings Python API

```
import vertexai
from vertexai.language_models import TextEmbeddingModel

vertexai.init(project="YOUR-PROJECT-ID", location="us-central1")

model = TextEmbeddingModel.from_pretrained("textembedding-gecko")

embeddings = model.get_embeddings([
    "Gen AI is changing the way we create.",
    "Self-driving cars would be fantastic.",
    "I was promised jetpacks."]
)

for embedding in embeddings:
    vector = embedding.values
    print(vector)
```

# Embeddings Python API

```
import vertexai
from vertexai.language_models import TextEmbeddingModel

vertexai.init(project="YOUR-PROJECT-ID", location="us-central1")

model = TextEmbeddingModel.from_pretrained("textembedding-gecko")

embeddings = model.get_embeddings([
    "Gen AI is changing the way we create.",
    "Self-driving cars would be fantastic.",
    "I was promised jetpacks."]
)

for embedding in embeddings:
    vector = embedding.values
    print(vector)
```

# Embeddings Python API

```
import vertexai
from vertexai.language_models import TextEmbeddingModel

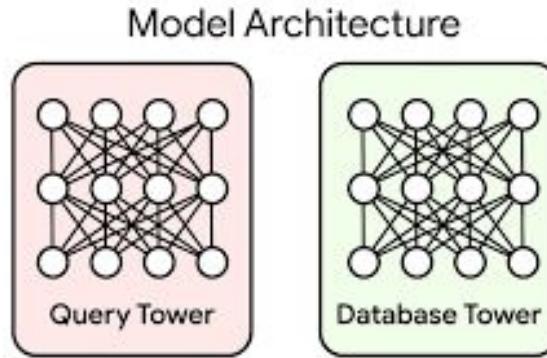
vertexai.init(project="YOUR-PROJECT-ID", location="us-central1")

model = TextEmbeddingModel.from_pretrained("textembedding-gecko")

embeddings = model.get_embeddings([
    "Gen AI is changing the way we create.",
    "Self-driving cars would be fantastic.",
    "I was promised jetpacks."]
)

for embedding in embeddings:
    vector = embedding.values
    print(vector)
```

# Vertex AI Matching Engine

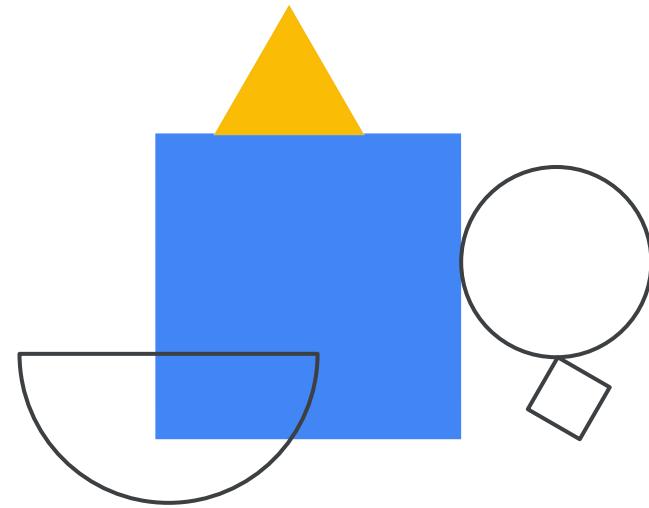


# Model Garden

- 01 Foundation Models
- 02 PaLM in Gen AI Studio
- 03 PaLM API
- 04 Embeddings API
- 05 **Lab: Use the PaLM API to Cluster Products Based on Descriptions**



**Lab:**  
**Use the PaLM API to**  
**Cluster Products Based**  
**on Descriptions**







## Fine-tunable Models

# Fine-tunable Models

01 Fine-tunable Models Review

02 Tuning PaLM

03 Vertex AI Pipelines

04 Lab: Running Custom Model Training on  
Vertex AI Pipelines



# Fine-tunable models

## Foundation

- PaLM (via Gen AI Studio)
- Stable Diffusion Inpainting
- ControlNet

## Image Classification

- EfficientNetV2
- ResNet
- BEiT
- DeiT
- MobileNet
- ... many more!

## Object Detection

- SpineNet
- YOLO
- Faster R-CNN
- Mask R-CNN
- RetinaNet

# Stable Diffusion Inpainting



-



+

"A unicorn  
sitting on a  
bench"

# Stable Diffusion Inpainting



# EfficientNetV2



## Classification

---

Mountain: 0.16

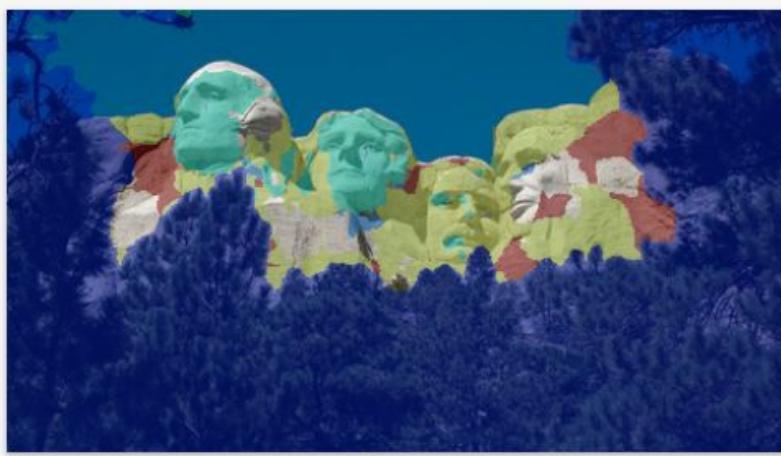
Pinyon: 0.074

Geological\_forma<sub>n</sub>tion: 0.073

National Monument: 0.068

Stone\_pine: 0.056

# DeepLabv3+



sidewalk

building

wall

pole

polegroup

traffic light

traffic sign

vegetation

person

The models are generally good,  
but not good at **your task.**

# Fine-tunable Models

01 Fine-tunable Models Review

02 **Tuning PaLM**

03 Vertex AI Pipelines

04 Lab: Running Custom Model Training on  
Vertex AI Pipelines



Let's build a recipe application that is aware  
of a **food preference** and a **food allergy**.

“

Create a vegan recipe for the requested dish that does not contain nightshade plants. Format the recipe as JSON.

Dish: <User Input Here>

# Pasta

```
{  
  "title": "Vegan Pasta with Roasted Vegetables",  
  "ingredients": [  
    "1 pound gluten-free pasta",  
    "1 tablespoon olive oil",  
    "1 yellow onion, diced",  
    "2 cloves garlic, minced",  
    "1 red bell pepper, diced",  
    "1 green bell pepper, diced",  
    "1 zucchini, diced",  
    "1 carrot, diced",  
    "1 (15 ounce) can chickpeas, rinsed and  
    drained",  
    "1 (15 ounce) can corn, drained",  
    "1/2 cup vegetable broth",  
    "1/4 cup chopped fresh basil",  
    "1/4 cup chopped fresh parsley",  
    "Salt and pepper to taste"  
  ],  
}
```



```
  "steps": [  
    "Preheat oven to 400 degrees F (200 degrees C). Toss  
    the vegetables with olive oil and salt and pepper to  
    taste. Spread them out on a baking sheet and roast for  
    20-25 minutes, or until tender and slightly browned.",  
    "While the vegetables are roasting, cook the pasta  
    according to the package directions. Drain and set  
    aside.",  
    "In a large pot or skillet, heat the olive oil over  
    medium heat. Add the onion and cook until softened, about  
    5 minutes. Add the garlic and cook for 1 minute more.",  
    "Add the roasted vegetables, chickpeas, corn,  
    vegetable broth, basil, and parsley to the pot. Bring to  
    a boil, then reduce heat and simmer for 10 minutes, or  
    until the vegetables are heated through.",  
    "Add the cooked pasta to the pot and stir to combine.  
    Serve immediately."  
  ],  
  "servings": 4  
}
```

“

Create a vegan recipe for the requested dish that does not contain nightshade plants like tomatoes, potatoes, eggplants, and peppers. Format the recipe as JSON.

Dish: <User Input Here>

# Pasta

```
{  
  "name": "Vegan Pasta with Sun-Dried Tomatoes  
and Walnuts",  
  "ingredients": [  
    "1 pound whole wheat pasta",  
    "1/2 cup sun-dried tomatoes, chopped",  
    "1/2 cup walnuts, chopped",  
    "1/4 cup olive oil",  
    "1/4 cup chopped fresh basil",  
    "1/4 cup chopped fresh parsley",  
    "1/4 cup grated Parmesan cheese  
(optional)",  
    "Salt and pepper to taste",  
  ],
```



```
  "instructions": [  
    "Bring a large pot of salted water to a boil.  
Add the pasta and cook according to the package  
directions.",  
    "While the pasta is cooking, heat the olive oil  
in a large skillet over medium heat. Add the  
sun-dried tomatoes and walnuts and cook for 2-3  
minutes, or until the walnuts are toasted and the  
tomatoes are softened.",  
    "Drain the pasta and add it to the skillet with  
the sun-dried tomatoes and walnuts. Stir to  
combine.",  
    "Add the basil, parsley, Parmesan cheese (if  
using), salt, and pepper to taste. Stir to  
combine.",  
    "Serve immediately.",  
  ],  
  "yield": "4 servings"  
}
```

“

Create a vegan recipe for the requested dish that does not contain nightshade plants like tomatoes, potatoes, eggplants, peppers, chilis, cayenne, and paprika. Replace animal products like milk and eggs with plant-based equivalents like almond milk and flax eggs. Use the following JSON format for the recipe:

```
{ "title": "", "ingredients": [ { "measurement": "", "ingredient": "" } ], "steps": [],  
"yield": "" }
```

Dish: <User Input Here>

Where do we get lots of examples?  
**PaLM!**

# Fine-tuning dataset

```
{"input_text": "PromptExample1", "output_text": "OutputExample1"}  
 {"input_text": "PromptExample2", "output_text": "OutputExample2"}  
 {"input_text": "PromptExample3", "output_text": "OutputExample3"}
```

## input\_text

- Should contain instructions
- Should match production user inputs

## output\_text

- Should contain content you expect
- Should be in the format you want

# Dataset JSON

```
{"input_text": 'Create a vegan recipe for the requested dish that does not contain nightshade plants like tomatoes, potatoes, eggplants, peppers, chilis, cayenne, and paprika. Replace animal products like milk and eggs with plant-based equivalents like almond milk and flax eggs. Use the following JSON format for the recipe:\n{\n"title": "",\n"ingredients": [ {\n"measurement": "",\n"ingredient": "" } ],\n"steps": [ "" ],\n"yield": ""\n}\ndish: pasta', "output_text": '{\n"title": "Hearty Mediterranean Pasta",\n"ingredients": [\n{\n"measurement": "1/2 cup",\n"ingredient": "cooked chickpeas"},\n{\n"measurement": "1/2 cup",\n"ingredient": "cooked lentils"},\n{\n"measurement": "1/4 cup",\n"ingredient": "olive oil"},\n{\n"measurement": "1/4 cup",\n"ingredient": "balsamic vinegar"},\n{\n"measurement": "1/4 cup",\n"ingredient": "Italian seasoning"},\n{\n"measurement": "1/4 cup",\n"ingredient": "fresh parsley, chopped"},\n{\n"measurement": "1/4 cup",\n"ingredient": "vegan parmesan cheese"},\n{\n"measurement": "1/2 pound",\n"ingredient": "whole wheat pasta"}],\n"steps": [\n"In a large bowl, combine the chickpeas, lentils, olive oil, balsamic vinegar, Italian seasoning, parsley, and vegan parmesan cheese.",\n"Cook the pasta according to the package directions.",\n"Drain the pasta and add it to the bowl with the chickpea mixture.",\n"Toss to combine.",\n"Serve immediately."],\n"yield": "4 servings"}'}
```

## Create a tuned model

### 1 Tuning dataset

### 2 Model details

START TUNING

Model tuning creates and deploys a new model from an existing one that's better adapted to your use case. Currently, model tuning occurs in limited regions. If you have an organization policy restricting certain regions, model tuning may fail.

#### Tuning dataset

The tuning dataset is a JSONL file that contains model prompt and responses examples (one per line). It's recommended that you use at least [100-500 samples](#).  
[Learn more about dataset formatting](#)

You can upload the file or select one that's already on Cloud Storage.

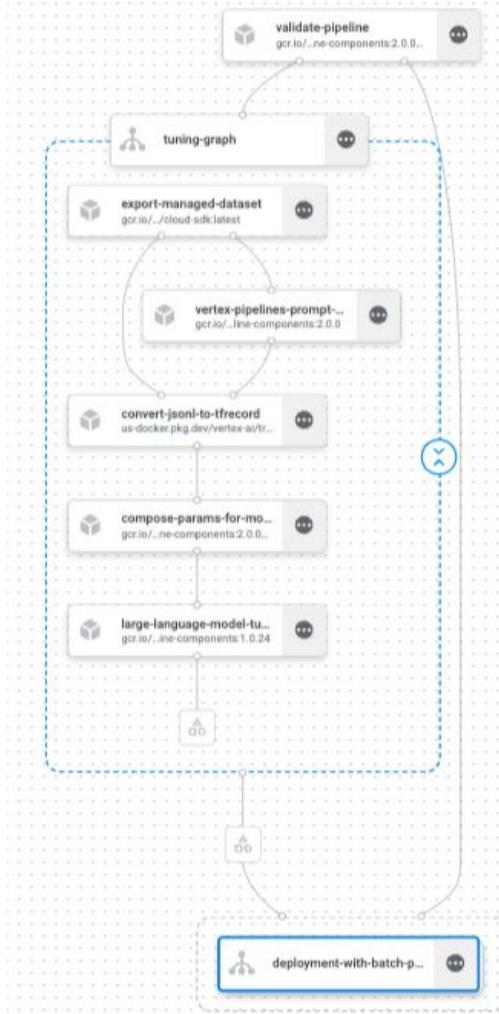
- Upload JSONL file to Cloud Storage
- Existing JSONL file on Cloud Storage

GCS file path \*

gs:// custom-recipe-app/recipes.jsonl

BROWSE

**CONTINUE**



# Serverless process management!

# Use Fine-tuned Model

```
import vertexai
from vertexai.language_models import TextGenerationModel

vertexai.init(project="YOUR-PROJECT-ID", location="us-central1")

model = TextGenerationModel.from_pretrained("YOUR_MODEL")  
  
prompt = """Create a vegan recipe ...
Dish: """
  
  
user_input = "pasta"
  
  
response = model.predict(prompt + user_input)
  
  
print(f"Response from Model: {response.text}")
```

# Fine-tunable Models

01 Fine-tunable Models Review

02 Tuning PaLM

03 **Vertex AI Pipelines**

04 Lab: Running Custom Model Training on  
Vertex AI Pipelines

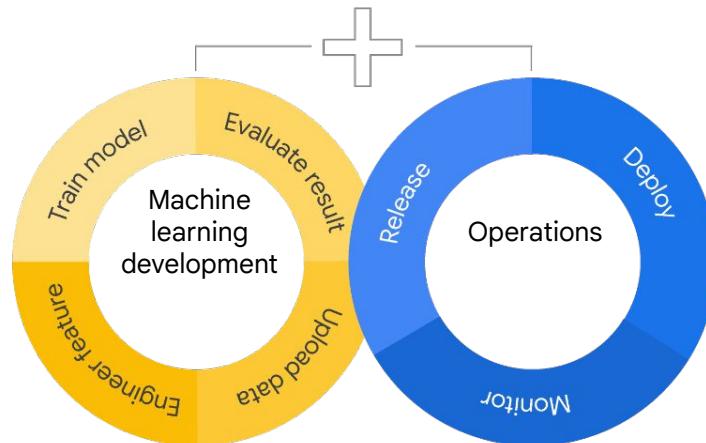


# Improve models with MLOps

Models need to evolve over time, which requires a process for managing change



MLOps phases:



Continuous integration (CI)

Continuous training (CT)

Continuous delivery (CD)

# MLOps

## Continuous Integration

Allow the user to tell us when the model is incorrect.

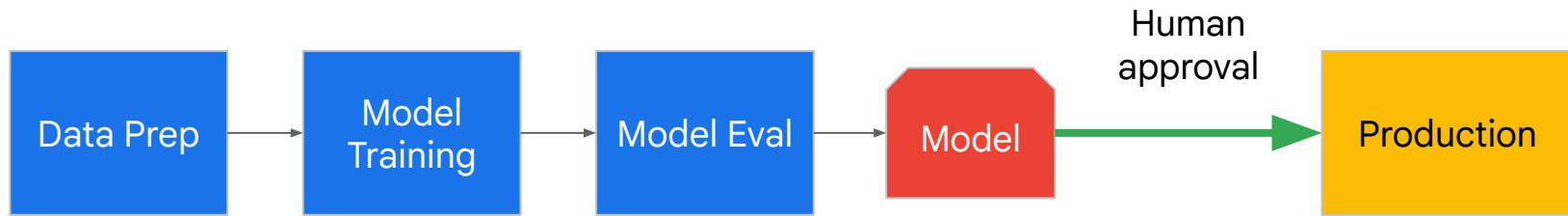
## Continuous Training

Train a new version when you've accumulated enough new training examples.

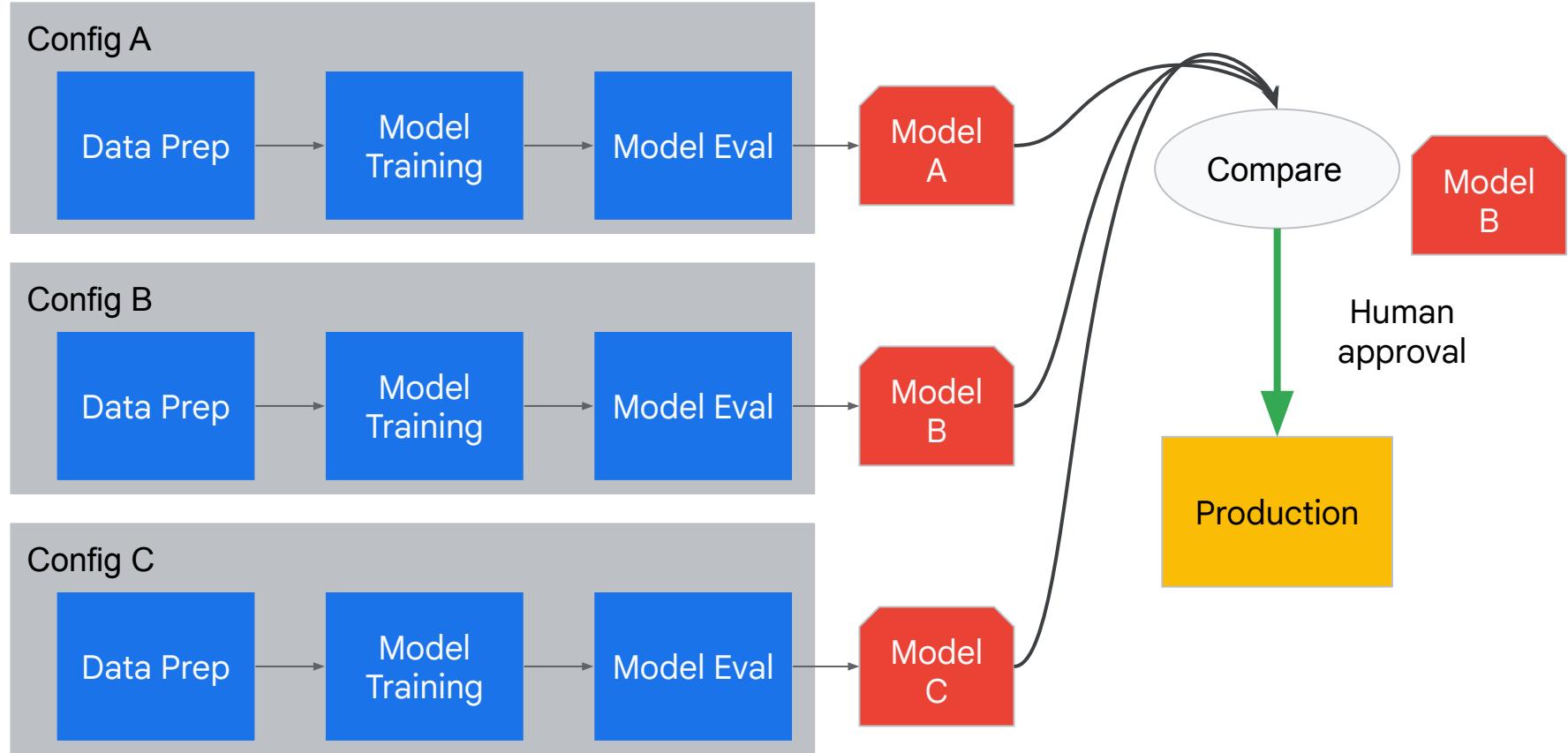
## Continuous Delivery

This process usually results in new versions, which are easy to deploy

# MLOps



# Experiment!





## Kubeflow

---

Open-source platform for  
machine learning  
pipelines on Kubernetes



## TensorFlow Extended

## TFX

---

Flexible, open-source  
framework that helps you  
build, deploy, and  
manage pipelines.

# Kubeflow

## Portable

Built on top of  
Kubernetes, Kubeflow  
runs anywhere!

## Reusable

Components are  
functions you can use on  
your data. Pass in  
parameters, get outputs.

## Scalable

All pipeline steps are  
containers. Handle 1 data  
point or 1 million with the  
same process.

# Vertex AI Pipelines

## Serverless

No need to build or manage a K8s cluster!

## Experiments

No pipeline run is entirely independent. Keep track of pipeline runs for the same purpose with Experiments.

## Ecosystem

Core cloud services like logging, security, and data governance work together with the other Vertex AI tools to provide a complete platform!

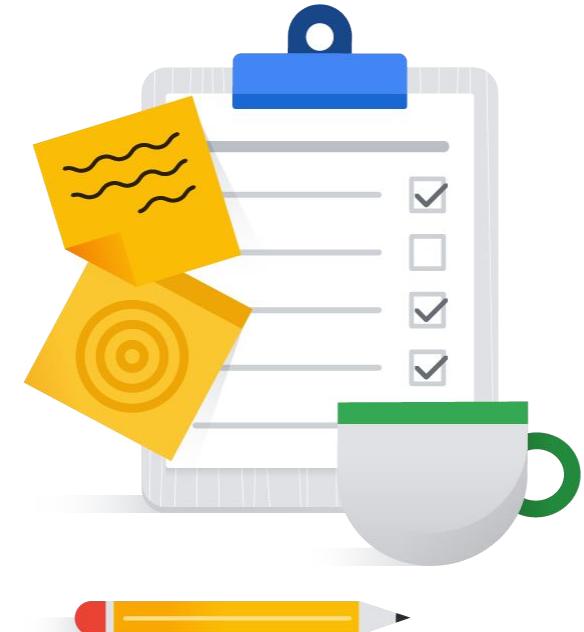
# Fine-tunable Models

01 Fine-tunable Models Review

02 Tuning PaLM

03 Vertex AI Pipelines

04 **Lab: Running Custom Model Training on  
Vertex AI Pipelines**



# Take a structured approach to GenAI transformation



## Ramp your skills

Continue your learning journey and complete our [Generative AI Learning Path](#), free of charge, on Google Cloud Skills Boost



## Organizational readiness

Assess your organization's current status and business needs for generative & traditional AI capabilities



## Identify use cases

Select from one of our [Jumpstart GenAI offers](#), and work with Google Cloud to develop a technical design doc and sample code to solve the use case



## Test and scale

Purchase and implement generative AI solutions. Not all AI is built equal. POC often and fail fast to identify what works for your business.

Contact your Google Cloud Representative to learn more

# 4 GenAI Jumpstart offers - \$25k & 2 weeks per use case



## CREATE

Bring your thoughts and visions to life

### Use cases:

- Images from text
- Product descriptions from images
- Blog post from content\*
- Email from content\*
- Release notes from content\*
- Report from content\*
- Press releases from content\*
- Personalized ads\*



## SUMMARIZE

Condense and summarize your knowledge base into a simple format

### Use cases:

- Content/video summarization
- Intra-knowledge Q&A
- Explanations of code content\*
- External chatbot using internal data\*
- External chatbot using website data\*



## DISCOVER

Help your customers and employees find what they need at the right time

### Use cases:

- Search for a document
- Machine-generated event monitoring
- File organization based on content\*
- Exam questions from content\*



## AUTOMATE

Automate your customer service across multiple channels

### Use cases:

- Contract information extraction
- Feedback classification and ticket creation
- Sentiment analysis\*
- Content translation\*
- Structured data extraction from file\*
- Media tagging\*
- Product tagging\*
- Content moderation \*

\*may require Responsible AI Review

**Lab:**

**Running Custom Model**

**Training on Vertex AI Pipelines**

