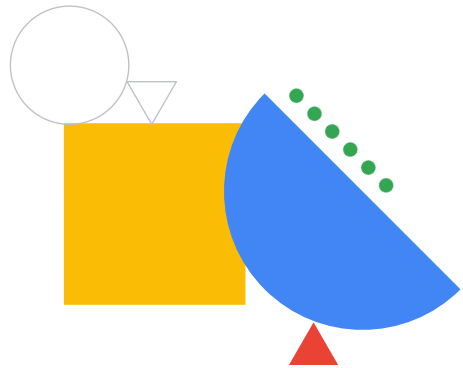


# The Machine Learning Workflow with Vertex AI

## Module 5

Google Cloud Big Data and Machine Learning Fundamentals





# Introduction

01 Big Data and Machine Learning on Google Cloud ☐

02 Data Engineering for Streaming Data ☐

03 Big Data with BigQuery ☐

04 Machine Learning Options on Google Cloud ☐

05 The Machine Learning Workflow with Vertex AI ☒

In the previous module of this course, you explored the machine learning options available on Google Cloud. Now let's switch our focus to the machine learning Workflow with Vertex AI.

# Agenda



- ✓ The three stages of the ML workflow
  - Data preparation
  - Model training
    - Model training
    - Model evaluation
  - Model serving
- ✓ Hands-on lab

Vertex AI, Google's AI platform, provides developers and data scientists one unified environment to build custom ML models. In this module, you'll explore three major stages of the ML workflow from data preparation, to model training, and finally, model deployment. This process is actually not too different from serving food in a restaurant—starting with preparing raw ingredients through to serving dishes on the table.

After that, you'll get hands-on practice building a machine-learning model end-to-end using AutoML, the no-code solution on Vertex AI.

# Traditional programming vs. machine learning

## Traditional programming



Data + rules → answers

$$1 + 1 = 2$$

## Machine learning



Data + answers → rules

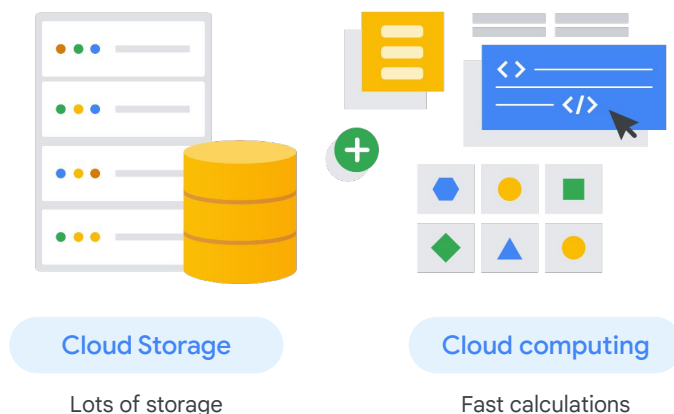
1	+	1	=	2	1 + 1 = 2
2	+	3	=	5	2 + 3 = 5

But before we get into the details, let's look at the basic differences between machine learning and traditional programming.

In traditional programming, simply put, one plus one equals two ( $1+1=2$ ). Data, plus rules—otherwise known as algorithms—lead to answers. And with traditional programming, a computer can only follow the algorithms that a human has set up.

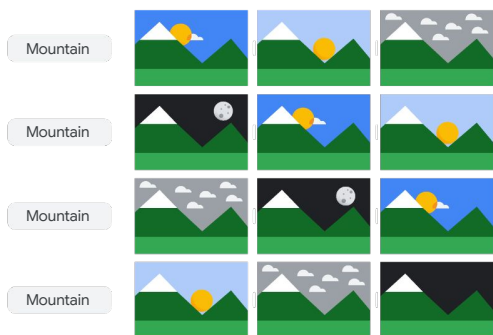
But what if we're just too lazy to figure out the algorithms? Or what if the algorithms are too complex to figure out? This is where machine learning comes in. With machine learning, we feed a machine a large amount of data, along with answers that we would expect a model to conclude from that data. Then, we show the machine a learning method by selecting a machine learning model. From there, we expect the machine to learn from the provided data and examples to solve the puzzle on its own. So, instead of telling a machine how to do addition, we give it pairs of numbers and the answers. For example, 1, 1, and 2, and 2, 3, and 5. We then ask it to figure out how to do addition on its own.

## ML needs storage and computing



But how is it possible that a machine can actually learn to solve puzzles? For machine learning to be successful, you'll need lots of storage, like what's available with **Cloud Storage**, and the ability to make fast calculations, like with **cloud computing**.

## ML example: Google Photos



- ✓ Pictures (data)
- ✓ Tags (answers)
- ✓ Associations between pictures and tags (rules)
- ✓ Data + answers = rules
- ✓ Applications: search features, albums

There are many practical examples of this capability. For example, by feeding Google Photos lots of pictures with tags (imagining tags are answers associates with data), we can teach the software to associate and then automatically attach tags to new pictures (finding rules). Tags can then be used for search functionality, or even to automatically create photo albums.

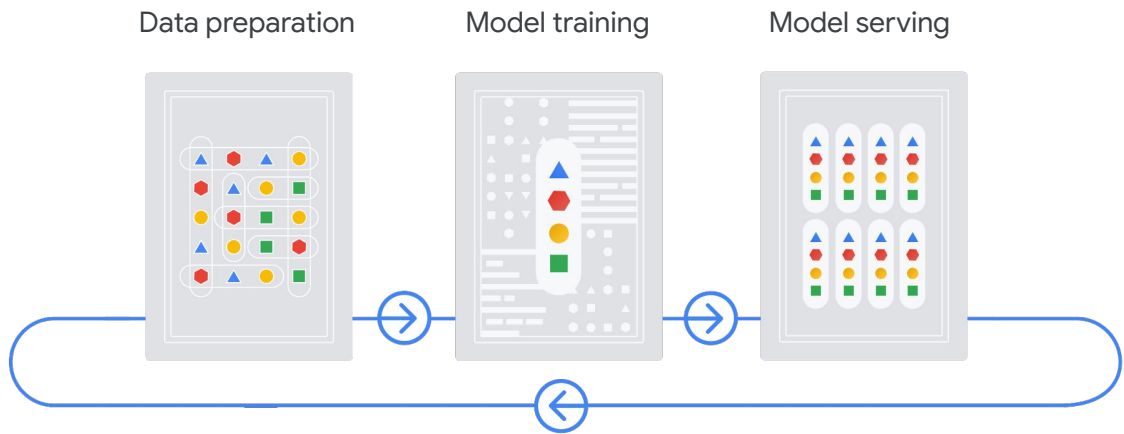
Can you come up with any  
other **ML examples**?

Data + answers → rules

Can you come up with any other examples to apply machine learning capabilities (data+answers→rules)? Take a moment to think about it.

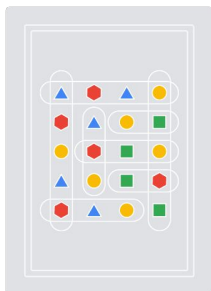


# The machine learning workflow



There are three key stages to this learning process.

## Stage 1: Data preparation



A model needs a large amount of data to learn from.

- ✓ Data uploading
- ✓ Feature engineering

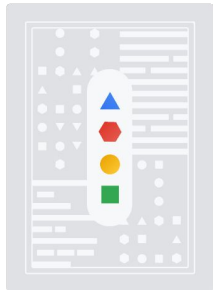
Data types

- ✓ Streaming vs. batch data
- ✓ Structured vs. unstructured data

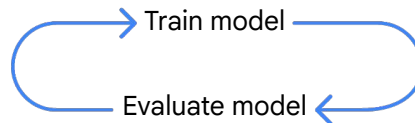
The first stage is data preparation, which includes two steps: data uploading and feature engineering. You will be introduced to feature engineering in the next lesson.

A model needs a large amount of data to learn from. Data used in machine learning can be either real-time streaming data or batch data, and it can be either structured, which is numbers and text normally saved in tables, or unstructured, which is data that can't be put into tables, like images and videos.

## Stage 2: Model training

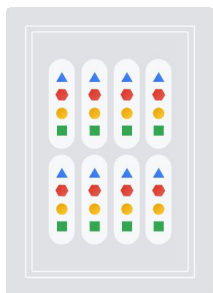


A model needs a tremendous amount of iterative training.



The second stage is model training. A model needs a tremendous amount of iterative training. This is when training and evaluation form a cycle to train model, then evaluate the model, and then train the data some more.

## Stage 3: Model serving

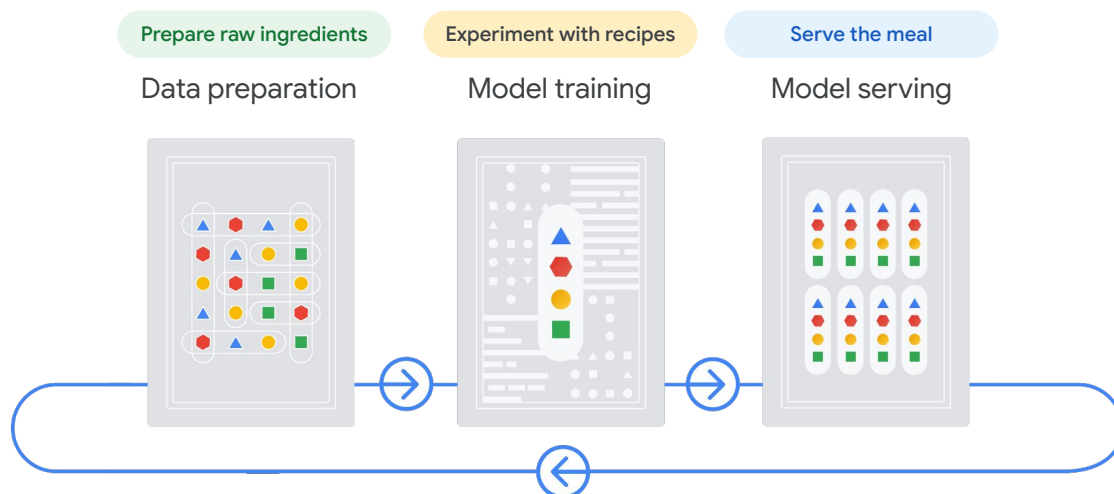


A model needs to actually be used in order to predict results.

- ✓ Deployed
- ✓ Monitored
- ✓ Managed

The third and final stage is model serving. A model needs to actually be used in order to predict results. This is when the machine learning model is deployed, monitored, and managed. If you don't move an ML model into production, it has no use and remains only a theoretical model.

## It's similar to serving food in restaurant



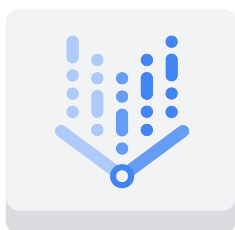
Google Cloud

We mentioned at the start that the machine learning workflow on Vertex AI is not too different from serving food in a restaurant. So, if you compare these steps to running a restaurant,

- Data preparation is when you prepare the raw ingredients,
- Model training is when you experiment with different recipes, and
- Model serving is when you finalize the menu to serve the meal to lots of hungry customers.

Now it's important to note that an ML workflow isn't linear, it's iterative. For example, during model training, you may need to return to dig into the raw data and generate more useful features to feed the model. When monitoring the model during model serving, you might find data drifting, or the accuracy of your prediction might suddenly drop. You might need to check the data sources and adjust the model parameters. Fortunately, these steps can be automated with machine learning operations, or MLOps. We'll go into more detail on this soon.

# Vertex AI is Google's unified AI platform

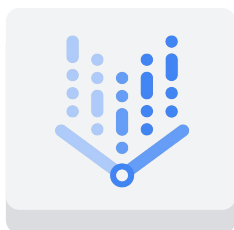


Vertex AI

- 1 AutoML: No-code solution
- 2 Custom training: Code-based solution

So how does Vertex AI support this workflow? You'll recall that Vertex AI provides two options to build machine learning models: AutoML, which is a codeless solution, and Custom Training, which is a code-based solution.

## Vertex AI features



Vertex AI

### Feature Store

A centralized repository for organizing, storing, and serving features to feed to training models

### Vizier

Helps tune hyperparameters in complex machine learning models

### Explainable AI

Helps with things like interpreting training performance

### Pipelines

Help monitor the ML production line

Vertex AI provides many features to support the ML workflow, all of which are accessible through either AutoML or Vertex AI workbench. Examples include:

- **Feature Store** provides a centralized repository for organizing, storing, and serving features to feed to training models,
- **Vizier** helps you tune hyperparameters in complex machine learning models,
- **Explainable AI** helps with things like interpreting training performance, and
- **Pipelines** help you monitor the ML production line. Specifically,
  - It helps you *automate, monitor, and govern* your ML workflow in a *serverless manner*.
  - The ML workflow starts from *data preparation, to train and evaluate the ML model, and finally to deploy and monitor the ML model*.
  - Additionally, it helps *store your workflow's artifacts* using Vertex ML Metadata.
    - By storing the artifacts of your ML workflow in Vertex ML Metadata, you can analyze the lineage of your workflow's artifacts — for example, an ML model's lineage may include the training data, hyperparameters, and code that were used to create the model.



## Data preparation

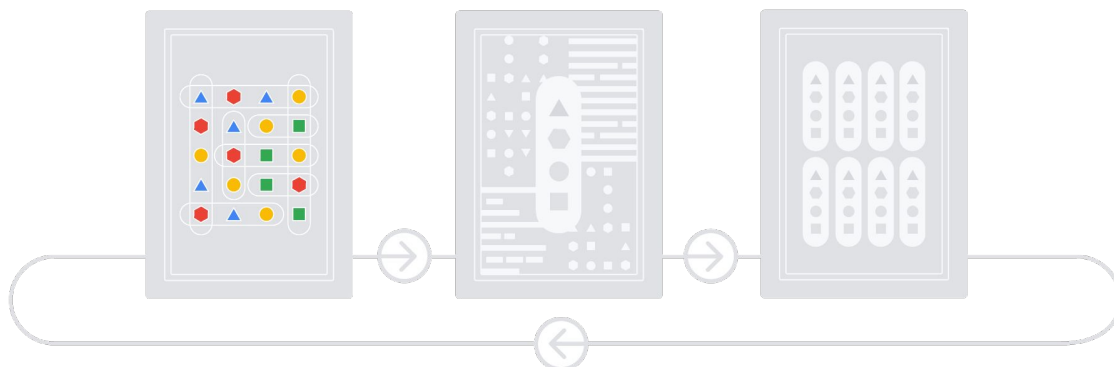


## Data preparation

Prepare raw ingredients

01 Upload data

02 Feature engineering



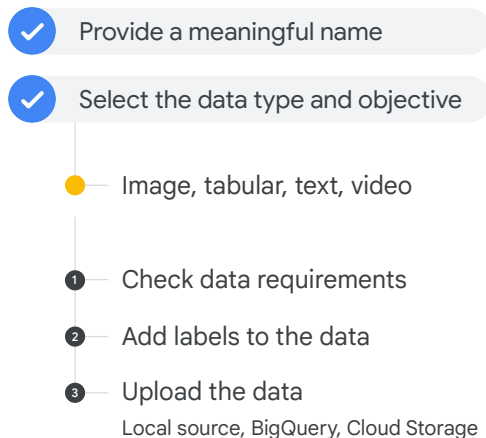
Google Cloud

Now let's look closer at an AutoML workflow. The first stage of the AutoML workflow is **data preparation**. During this stage, you must **upload data** and then prepare the data for model training with **feature engineering**.

# Upload data

01 Upload data

02 Feature engineering

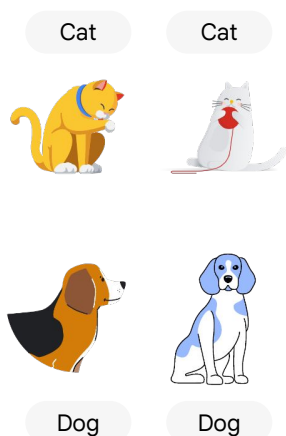


When you upload a dataset in the Vertex AI user interface, you'll need to provide a meaningful name for the data and then select the data type and objective. AutoML allows four types of data: image, tabular, text, and video.

To select the correct data type and objective, you should:

- Start by checking data requirements. We've included a link to these requirements in the resources section of this course.
- Next, you'll need to add labels to the data if you haven't already.
- The final step is to upload the data. Data can be uploaded from a local source, BigQuery, or Cloud Storage. You will practice these steps in the lab.

## A label is a training target



A label can be created:

- ✓ Manually
- ✓ Using Google's paid label service

A label is a training target. So, if you want a model to distinguish a cat from a dog, you must first provide sample images that are tagged—or *labeled*—either cat or dog. A label can be manually added, or it can be added by using Google's paid label service via the Vertex console. These human labellers will manually generate accurate labels for you.

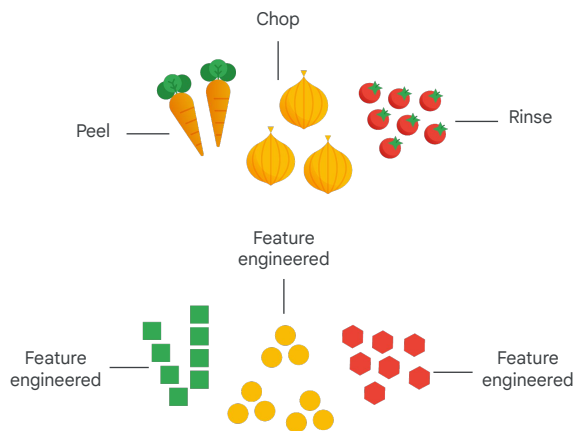
# Feature engineering

01 Upload data

02 Feature engineering

A feature is a factor that may contribute to the prediction.

- An independent variable in statistics
- A column in a table



Google Cloud

After your data is uploaded to AutoML, the next step is preparing the data for model training with **feature engineering**. A feature, as we discussed in the BigQuery module, refers to a factor that contributes to the prediction. It's an independent variable in statistics or a column in a table.

Imagine you're in the kitchen preparing a meal. Your data is like your ingredients, such as carrots, onions, and tomatoes. Before you start cooking, you'll need to peel the carrots, chop the onions, and rinse the tomatoes. This is what feature engineering is like: the data must be processed before the model starts training.

# Vertex AI Feature Store

## A centralized repository

- ✓ Organize, store, serve features.
- ✓ Aggregate features from different sources.
- ✓ Shop/use features available in store.

## Feature store benefits

- ✓ Features are sharable.
- ✓ Features are reusable.
- ✓ Features are scalable.
- ✓ Features are easy to use.

Preparing features can be both challenging and tedious. To help, Vertex AI has a function called *Feature Store*.

Feature Store is a centralized repository to organize, store, and serve machine learning features. It aggregates all the different features from different sources and updates them to make them available from a central repository. Then, when engineers need to model something, they can use the features available in the Feature Store dictionary to build a dataset.

Vertex AI automates the feature aggregation to scale the process.

So what are the benefits of Vertex AI Feature Store?

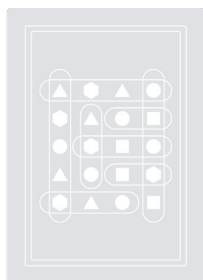
- First, **features are shareable** for training or serving tasks. Features are managed and served from a central repository, which helps maintain consistency across your organization.
- Second, **features are reusable**. This helps save time and reduces duplicative efforts, especially for high-value features.
- Third, **features are scalable**. Features automatically scale to provide low-latency serving, so you can focus on developing the logic to create the features without worrying about deployment.
- And fourth, **features are easy to use**. Feature Store is built on an easy-to-navigate user interface.



## Model training

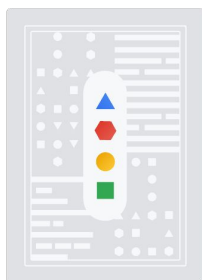
# Model training

## Data preparation



Prepare raw ingredients

## Model training



Experiment with recipes

01 Model training

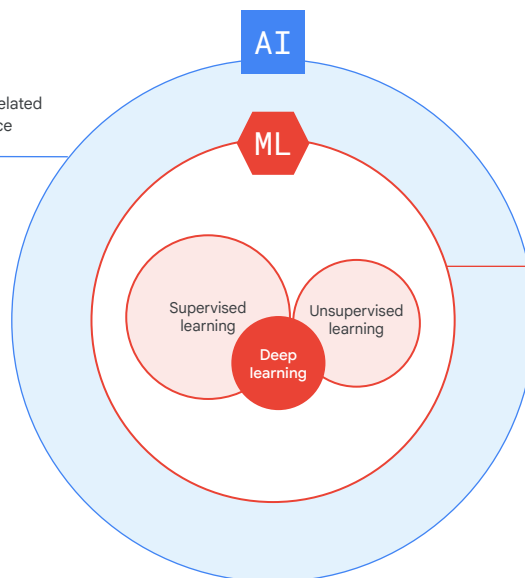
02 Model evaluation

Now that our data is ready, which, if we return to the cooking analogy, is our ingredients, it's time to train the model. This is like experimenting with some recipes. This stage involves two steps: **model training**, which would be like cooking the recipe, and **model evaluation**, which is when we taste how good the meal is. This process might be iterative.

# AI vs. ML

## Artificial intelligence

An umbrella term that includes anything related to computers mimicking human intelligence



## Machine learning

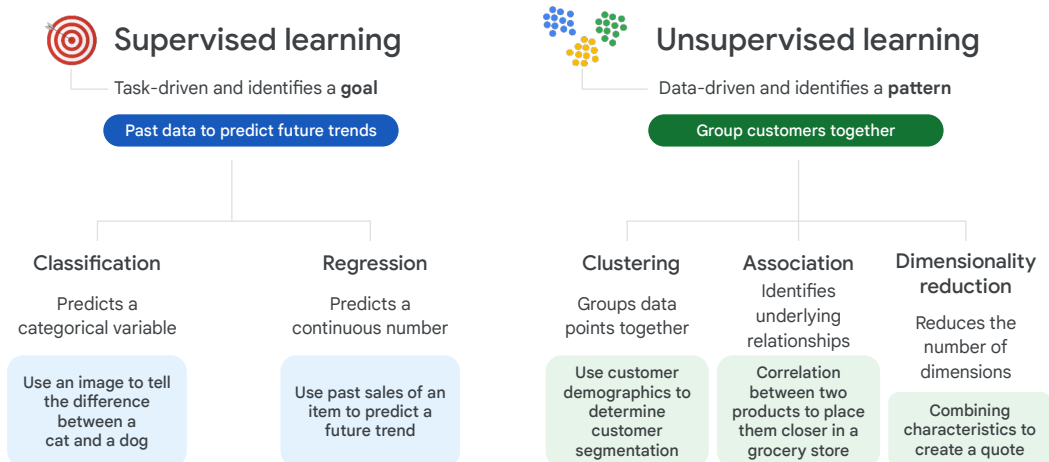
A subset of AI that mainly refers to supervised and unsupervised learning

Before we get into more details about this stage, let's pause to clarify two terms: artificial intelligence and machine learning.

- *Artificial intelligence*, or AI, is an umbrella term that includes anything related to computers mimicking human intelligence. For example, in an online word processor, robots perform human actions all the way down to spell check.
- *Machine learning* is a subset of AI that mainly refers to supervised and unsupervised learning.
- You might also hear the term deep learning, or deep neural networks. It's a subset of machine learning that adds layers in between input data and output results to make a machine learn at more depth.



# ML models



Google Cloud

So, what's the difference between supervised and unsupervised learning?

- **Supervised learning** is task-driven and identifies a goal.
- **Unsupervised learning**, however, is data-driven and identifies a pattern.

An easy way to distinguish between the two is that **supervised** learning provides each data point with a **label**, or an answer, while unsupervised does not.

For example, if we were given sales data from an online retailer, we could use supervised learning to predict the sales trend for the next couple of months and use unsupervised learning to group customers together based on common characteristics.

There are two major types of **supervised learning**:

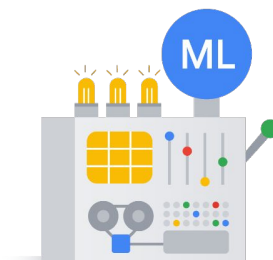
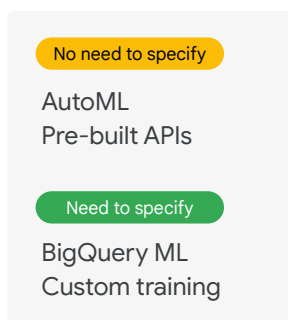
- The first is **classification**, which predicts a categorical variable, like using an image to tell the difference between a cat and a dog.
- The second type is a **regression** model, which predicts a continuous number, like using past sales of an item to predict a future trend.

And then there are three major types of **unsupervised learning**:

- The first is **clustering**, which groups together data points with similar characteristics and assigns them to "clusters", like using customer

- demographics, to determine customer segmentation.
- The second is **association**, which identifies underlying relationships, like a correlation between two products to place them closer together in a grocery store for a promotion.
- And the third is **dimensionality reduction**, which reduces the number of dimensions, or features, in a dataset to improve the efficiency of a model. For example, combining customer characteristics like age, driving violation history, or car type, to create an insurance quote. If too many dimensions are included, it can consume too many compute resources, which might make the model inefficient.

## Which ML options need to specify ML models?



Although Google Cloud provides four machine learning options, with AutoML and pre-built APIs you don't need to specify a machine learning model. Instead, you'll define your objective, such as text translation or image detection. Then on the backend, Google will select the best model to meet your business goal.

With the other two options, BigQuery ML and custom training, you'll need to specify which model you want to train your data on and assign something called hyperparameters. You can think of hyperparameters as user-defined knobs in a machine that helps guide the machine learning process. For example, one parameter is a learning rate, which is how fast you want the machine to learn.

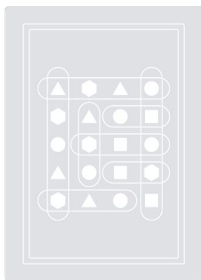
With AutoML, you don't need to worry about adjusting these hyperparameter knobs because the tuning happens automatically on the back end. This is largely done by a *neural architect search*, which finds the best fit model by comparing the performance against thousands of other models.



## Model evaluation

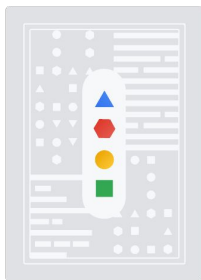
# Model evaluation

Data preparation



Prepare raw ingredients

Model training



Experiment with recipes



01 Model training

02 Model evaluation

While we are experimenting with a recipe, we need to keep tasting it constantly to make sure it meets our expectations. This is the **model evaluation** portion of the model training stage.

# Evaluation metrics



Vertex AI

## Evaluation metrics

Confusion matrix

Recall





Precision

Feature importance

Vertex AI provides extensive evaluation metrics to help determine a model's performance. Among the metrics are two sets of measurements.

- The first is based on the confusion matrix, for example recall and precision.
- The second is based on feature importance, which we'll explore later in this section of the course.

## Confusion matrix

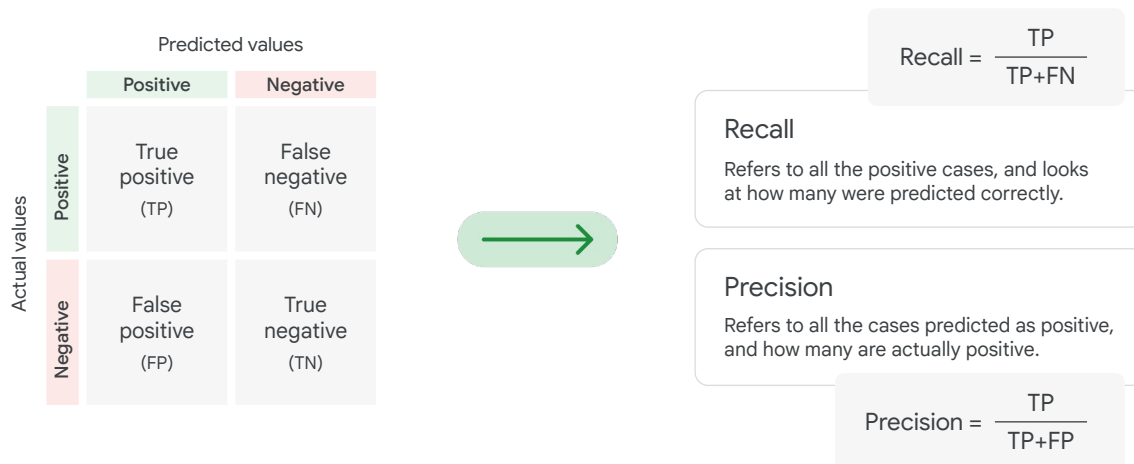
		Predicted values	
		Positive (cat)	Negative (dog)
Actual values	Positive (cat)	<b>True positive</b>  This is a cat.	<b>False negative</b>  This is a dog. Type 2 error
	Negative (dog)	<b>False positive</b>  This is a cat. Type 1 error	<b>True negative</b>  This is not a cat.

A **confusion matrix** is a specific performance measurement for machine learning classification problems. It's a table with combinations of **predicted** and **actual** values. To keep things simple we assume the output includes only two classes.

Let's explore an example of a confusion matrix.

- The first is a **true positive** combination, which can be interpreted as, "The model predicted positive, and that's true." The model predicted that this **is** an image of a cat, and it actually is.
- The opposite of that is a **true negative** combination, which can be interpreted as, "The model predicted negative, and that's true." The model predicted that a dog is **not** a cat, and it actually isn't.
- Then there is a **false positive** combination, otherwise known as a Type 1 Error, which can be interpreted as, "The model predicted positive, and that's false." The model predicted that a dog is a cat but it actually **isn't**.
- Finally, there is the **false negative** combination, otherwise known as a Type 2 Error, which can be interpreted as, "The model predicted negative, and that's false." The model predicted that a cat is **not** a cat, but it actually is.

# Recall and precision

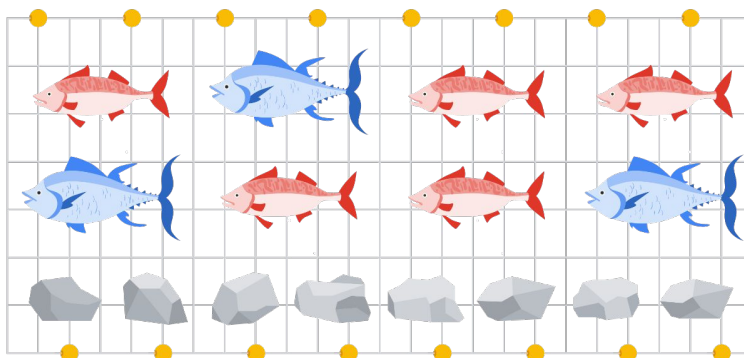
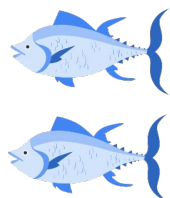


A confusion matrix is the foundation for many other metrics used to evaluate the performance of a machine learning model. Let's take a look at the two popular metrics, recall and precision, that you'll encounter in the lab.

- **Recall** refers to all the positive cases, and looks at how many were predicted correctly. This means that recall is equal to the *true positives*, divided by the sum of the *true positives* and *false negatives*.
- **Precision** refers to all the cases predicted as positive, and how many are actually positive. This means that precision is equal to the *true positives*, divided by the sum of the *true positives* and *false positives*.



## Example: Fishing with a wide net



Wide net

Total fish in the lake: 100  
Caught: 80 fish + 80 rocks

Recall: 80%

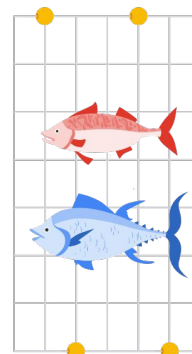
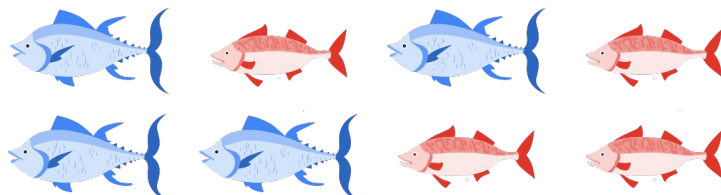
Precision: 50%

Google Cloud

Imagine you're fishing with a net. Using a **wide net**, you caught both fish and rocks: 80 fish out of 100 total fish in the lake, plus 80 rocks.

- The **recall** in this case is 80%, which is calculated by the number of fish caught, 80, divided by the total number of fish in the lake, 100.
- The **precision** is 50%, which is calculated by taking the number of fish caught, 80, and dividing it by the number of fish and rocks collected, 160.

## Example: Fishing with a smaller net



Smaller net

Total fish in the lake: 100  
Caught: 20 fish

Recall: 20%

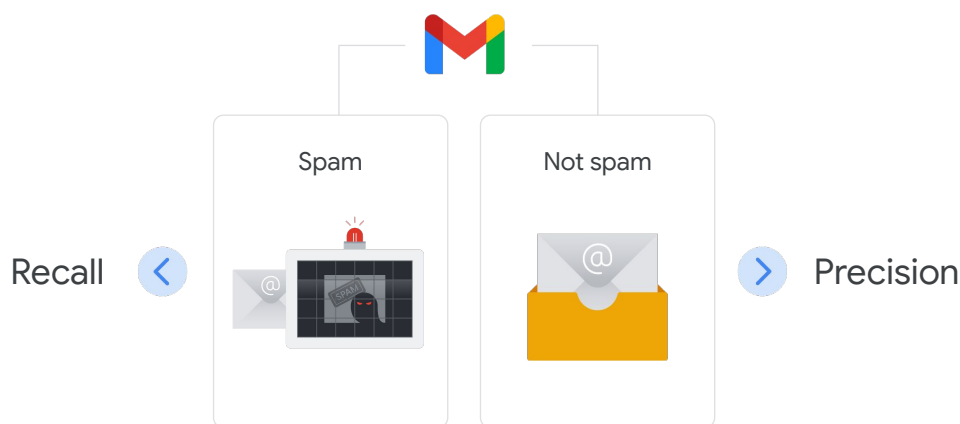
Precision: 100%

Google Cloud

Let's say you wanted to improve the precision, so you switched to a **smaller net**. This time you caught 20 fish and 0 rocks.

- The recall becomes 20% (20 out of 100 fish collected), and
- The precision becomes 100% (20 out of 20 total fish and rocks collected).

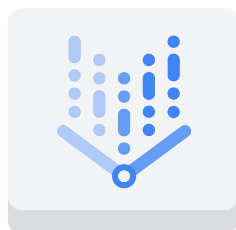
## The trade-off between recall and precision



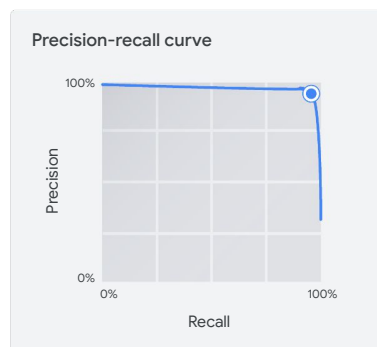
Recall and precision are often a trade-off. Depending on your use case, you may need to optimize for one or the other.

Consider a classification model where Gmail separates emails into two categories: spam and not-spam. If the goal is to catch as many potential spam emails as possible, Gmail may want to prioritize recall. In contrast, if the goal is to only catch the messages that are definitely spam without blocking other emails, Gmail may want to prioritize precision.

## The precision-recall curve

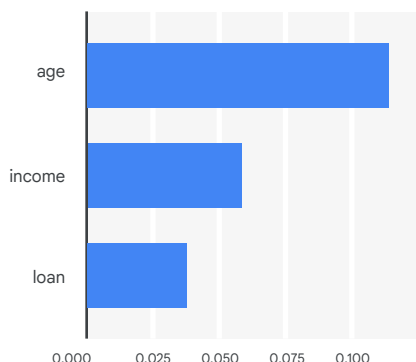


Vertex AI

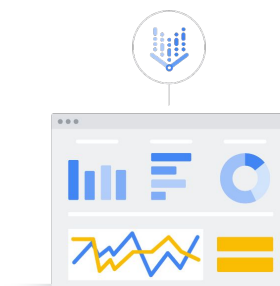


In Vertex AI, the platform visualizes the precision and the recall curve so they can be adjusted based on the problem that needs solving. You'll get the opportunity to practice adjusting precision and recall in the AutoML lab.

## Feature importance



Feature importance identifies how each feature contributes to a prediction.



Explainable AI

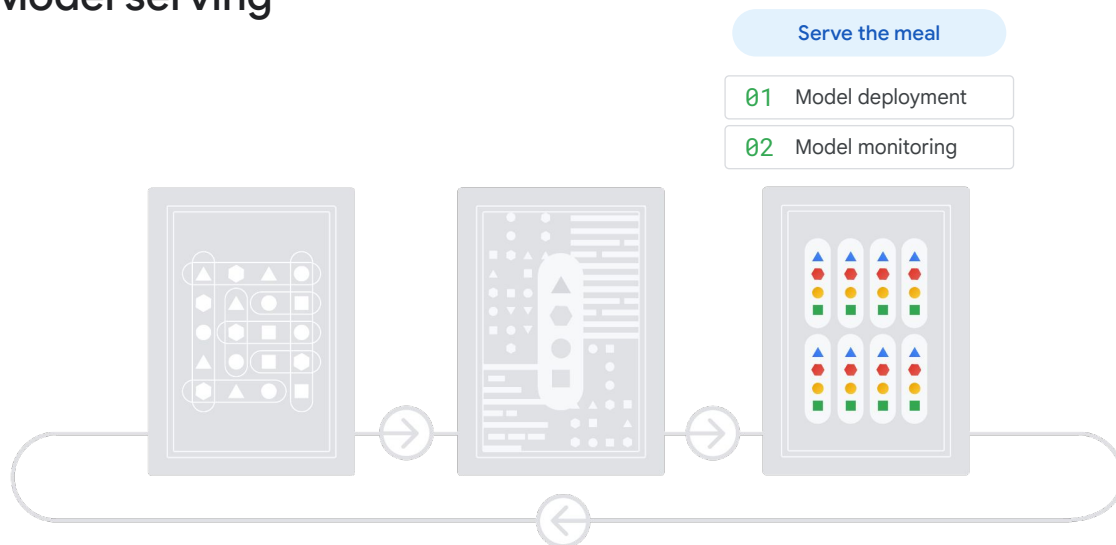
In addition to the confusion matrix and the metrics generated to measure model effectiveness, such as recall and precision, the other useful measurement is **feature importance**. In Vertex AI, feature importance is displayed through a bar chart to illustrate how each feature contributes to a prediction. The longer the bar, or the larger the numerical value associated with a feature, the more important it is. This information helps decide which features are included in a machine learning model to predict the goal. You'll observe the feature importance chart in the lab as well.

Feature importance is just one example of Vertex AI's comprehensive machine learning functionality called **Explainable AI**. Explainable AI is a set of tools and frameworks to help understand and interpret predictions made by machine learning models.



## Model deployment and monitoring

# Model serving



Google Cloud

The recipes are ready and now it's time to serve the meal! This represents the final stage of the machine learning workflow, **model serving**.

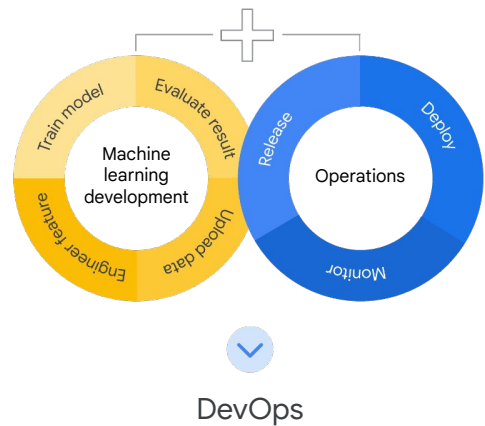
Model serving consists of two steps:

- First, **model deployment**, which we can compare to serving the meal to a hungry customer, and
- Second, **model monitoring**, which we can compare to checking with the waitstaff to ensure that the restaurant is operating efficiently.

It's important to note that model management exists throughout this whole workflow to manage the underlying machine learning infrastructure. This lets data scientists focus on what to do, rather than how to do it.

# Machine learning operations (MLOps)

- ✓ Solve production challenges
- ! Building an integrated ML system
- ! Operating it in production



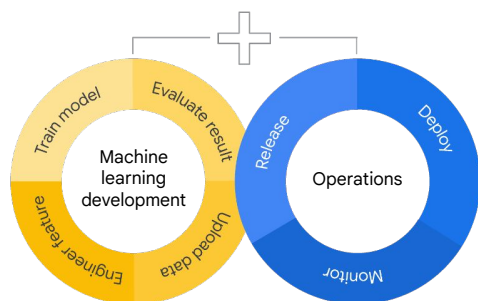
Machine learning operations, or MLOps, play a big role. MLOps combines machine learning development with operations, and applies similar principles from DevOps to machine learning models, which is short for development and operations.

MLOps aims to solve production challenges related to machine learning. In this case, this refers to building an integrated machine learning system and operating it in production. These are considered to be some of the biggest pain points by the ML practitioners' community, because both data and code are constantly evolving in machine learning.



# Practicing MLOps

Advocating for automation and monitoring at each step of the ML system construction.



Adopting a process to enable:

Continuous integration (CI)

Continuous training (CT)

Continuous delivery (CD)

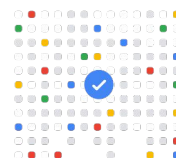
Practicing MLOps means advocating for automation and monitoring at each step of the ML system construction. This means adopting a process to enable

- continuous integration,
- continuous training, and
- continuous delivery.

# Model deployment is when the model is implemented

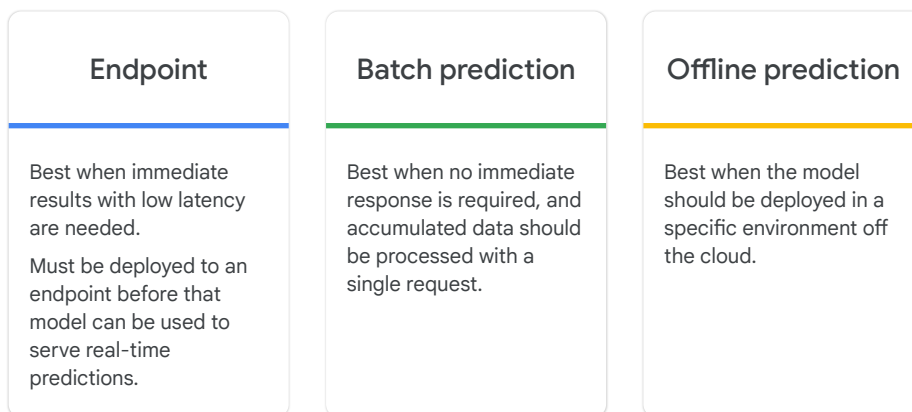
01

Model  
deployment



So, what does MLOps have to do with model serving? Well, let's start with **model deployment**, which is the exciting time when a model is implemented. In our restaurant analogy, it's when the food is put on the table for the customer to eat! MLOps provides a set of best practices on the backend to **automate** this process.

## Three ML deployment options



There are three options to deploy a machine learning model.

- The first is to deploy to an **endpoint**. This option is best when immediate results with low latency are needed, such as making instant recommendations based on a user's browsing habits whenever they're online. A model must be deployed to an endpoint before that model can be used to serve real-time predictions.
- The second option is to deploy using **batch prediction**. This option is best when no immediate response is required, and accumulated data should be processed with a single request. For example, sending out new ads every other week based on the user's recent purchasing behavior and what's currently popular on the market.
- And the final option is to deploy using **offline prediction**. This option is best when the model should be deployed in a specific environment off the cloud. In the lab, you'll practice predicting with an endpoint.

# Model monitoring

02

Model  
monitoring

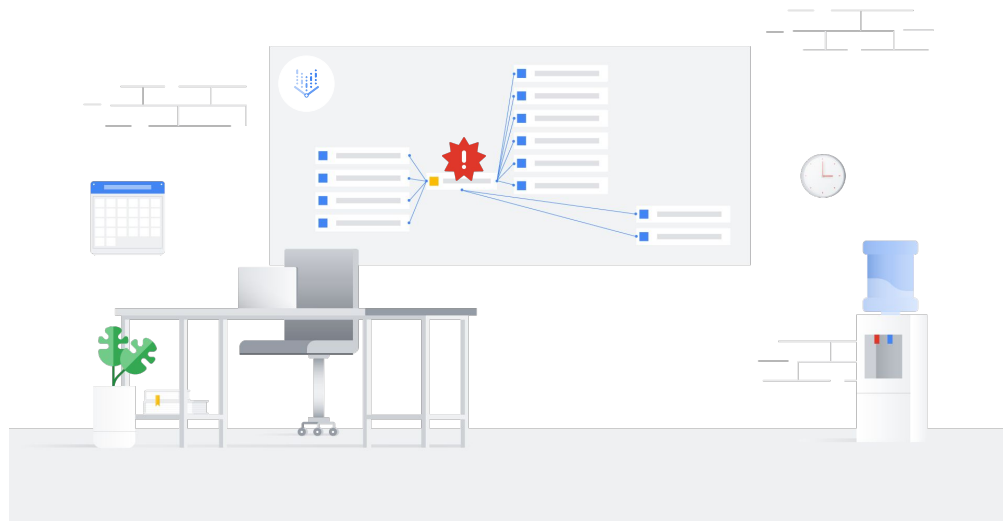


## Vertex AI Pipelines

- ✓ Automate
- ✓ Monitor
- ✓ Govern

Now let's shift our focus to **model monitoring**. The backbone of MLOps on Vertex AI is a tool called **Vertex AI Pipelines**. It automates, monitors, and governs machine learning systems by orchestrating the workflow in a serverless manner.

## Vertex AI Pipelines is similar to a production line



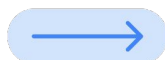
Google Cloud

Imagine you're in a production control room, and Vertex AI Pipelines is displaying the production data onscreen. If something goes wrong, it automatically triggers warnings based on a predefined threshold.

## You can define pipelines using Vertex AI Workbench



Vertex AI  
Workbench



Define your own pipeline  
with prebuilt pipeline  
components

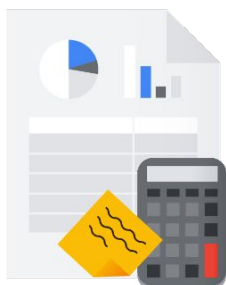
With Vertex AI Workbench, which is a notebook tool, you can define your own pipeline. You can do this with prebuilt pipeline components, which means that you primarily need to specify how the pipeline is put together using components as building blocks.

And it's with these final two steps, model deployment and model monitoring, that we complete our exploration of the machine learning workflow. The restaurant is open and operating smoothly – Bon appetit!



## Lab: Predicting Loan Risk with AutoML

## Lab introduction



Use AutoML to build a machine learning model to predict loan risk.

> Loans from a financial institution

> 2,050 data points

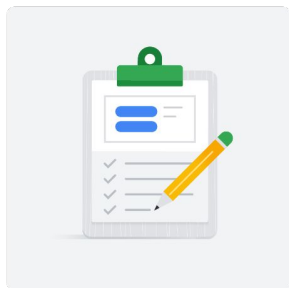
AutoML requires at least 1,000 data points

Let's put what you've just learned into practice with a hands-on lab.

In this lab, you'll use AutoML, a codeless tool, to build a machine learning model to predict loan risk. The dataset used in the lab relates to loans from a financial institution and has 2,050 data points. AutoML requires at least 1,000 data points in a dataset.



## Lab objectives



AutoML lab

You'll practice working through the three phases of the ML workflow:

- 1 Data preparation
- 2 Model training
- 3 Model serving

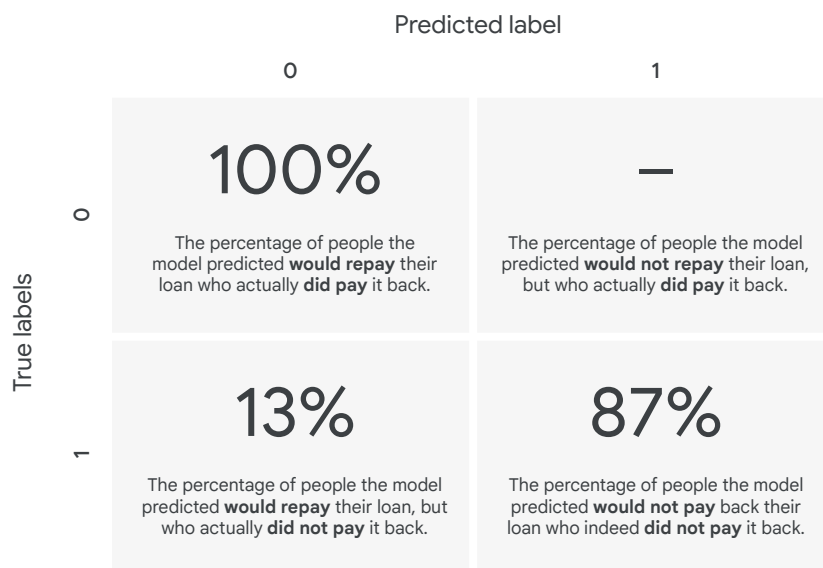
The goal is to practice working through the three phases of the machine learning workflow: data preparation, model training, and model serving.

Let's get started!



## Lab recap: Predicting Loan Risk with AutoML

## Confusion matrix result



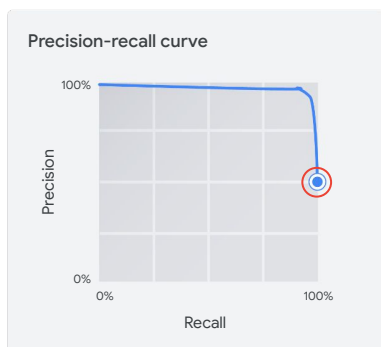
Well done on completing the AutoML lab! You've now had a chance to use Vertex AI to build a machine learning model without writing a single line of code.

Let's take a few moments to review the results of the lab, starting with the confusion matrix. But before that begins, please pause to consider the matrix results for yourself.

- The true positives were 100%. This represents the percentage of people the model predicted would repay their loan who actually did pay it back.
- The true negatives were 87%. This represents the percentage of people the model predicted would not repay their loan who indeed did not pay it back.
- The false negatives were 0%. This represents the percentage of people the model predicted would not repay their loan, but who actually did pay it back.
- And finally, the false positives were 13%. This represents the percentage of people the model predicted would repay their loan, but who actually did not pay it back.

As a general principle, it's good to have high true positives and true negatives, and low false positives and false negatives. However, how high or low they need to be really depends on the business goals you're looking to achieve. There are different ways to improve the performance of a model, which might include using a more accurate data source, using a larger dataset, choosing a different type of ML model, or tuning the hyperparameters.

## Confidence threshold of 0



Determines how a machine learning model counts the positive cases.

- + Higher threshold Increases precision, decrease recall
- Lower threshold decreases precision, increases recall
- 0 Zero threshold produces the highest recall of 100%, and the lowest precision of 50%

The model predicts that 100% of loan applicants will be able to repay a loan they take out. However, in actuality, only 50% of people were able to repay the loan.

Let's also review the precision-recall curve from the AutoML lab.

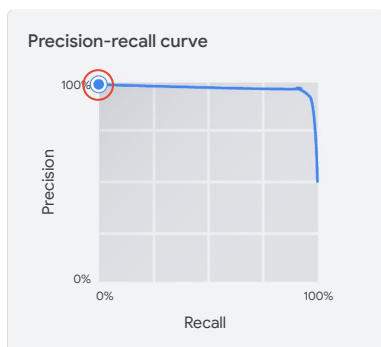
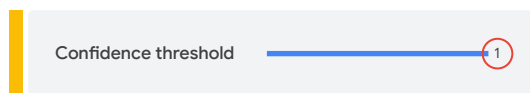
The **confidence threshold** determines how a machine learning model counts the positive cases.

- A higher threshold increases the precision, but decreases recall.
- A lower threshold decreases the precision, but increases recall.

Moving the threshold to zero produces the highest recall of 100%, and the lowest precision of 50%. So, what does that mean?

That means the model *predicts* that 100% of loan applicants will be able to repay a loan they take out. However, in *actuality*, only 50% of people were able to repay the loan. Using this threshold to identify the default cases in this example can be risky, because it means that you're only likely to get half of the loan investment back.

# Confidence threshold of 1



Determines how a machine learning model counts the positive cases.

- + Higher threshold Increases precision, decrease recall
- Lower threshold decreases precision, increases recall
- 1 A threshold of 1 produces the highest precision of 100%, with the lowest recall of 1%.

Of all the people who were predicted to repay the loan, 100% of them actually did. However, you rejected 99% of loan applicants by only offering loans to 1% of them.

Now let's move to the other extreme by moving the threshold to 1. This will produce the highest precision of 100% with the lowest recall of 1%.

So, what does that mean?

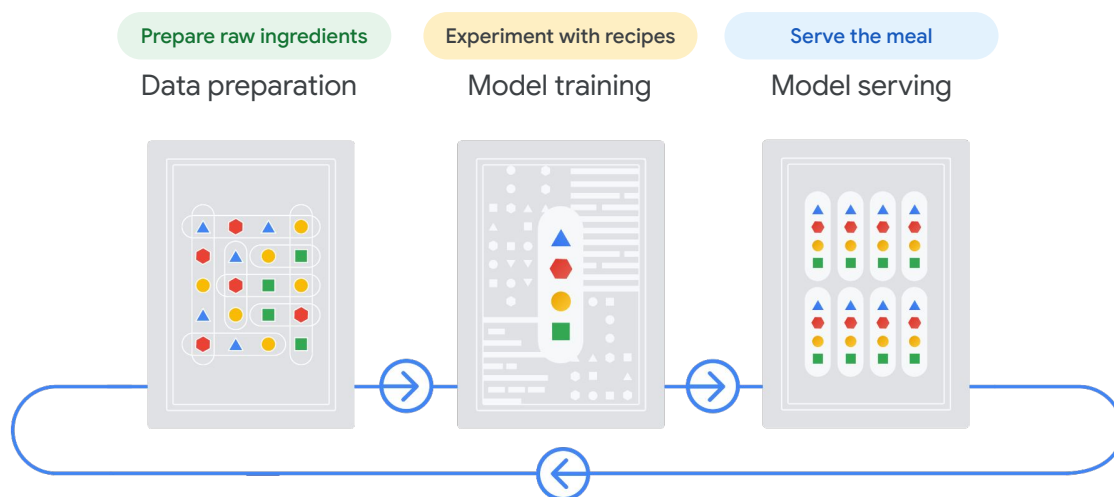
It means that of all the people who were predicted to repay the loan, 100% of them actually did. However, you rejected 99% of loan applicants by only offering loans to 1% of them. That's a pretty big loss of business for your company.

These are both extreme examples, but it's important that you always try to set an appropriate threshold for your model.



## Summary

## The three stages of the ML workflow



Before we finish this module of the course, let's quickly review the three stages of the machine learning workflow—with the help of our restaurant analogy.

# Summary



1

## Data preparation

- Data upload
- Feature engineering



2

## Model training

- Model training
- Model evaluation



3

## Model serving

- Model deployment
- Model monitoring



- In stage one, data preparation, we uploaded data and applied feature engineering. This translated to gathering our ingredients and then chopping and prepping them in the kitchen.
- In stage two, model training, the model was trained and evaluated. This is where we experimented with our recipes and tasted the meal to make sure it turned out as expected.
- And in the final stage, model serving, the model was deployed and monitored. This translates to serving the meal to the hungry customers and adjusting the menu as more people tried the dish.