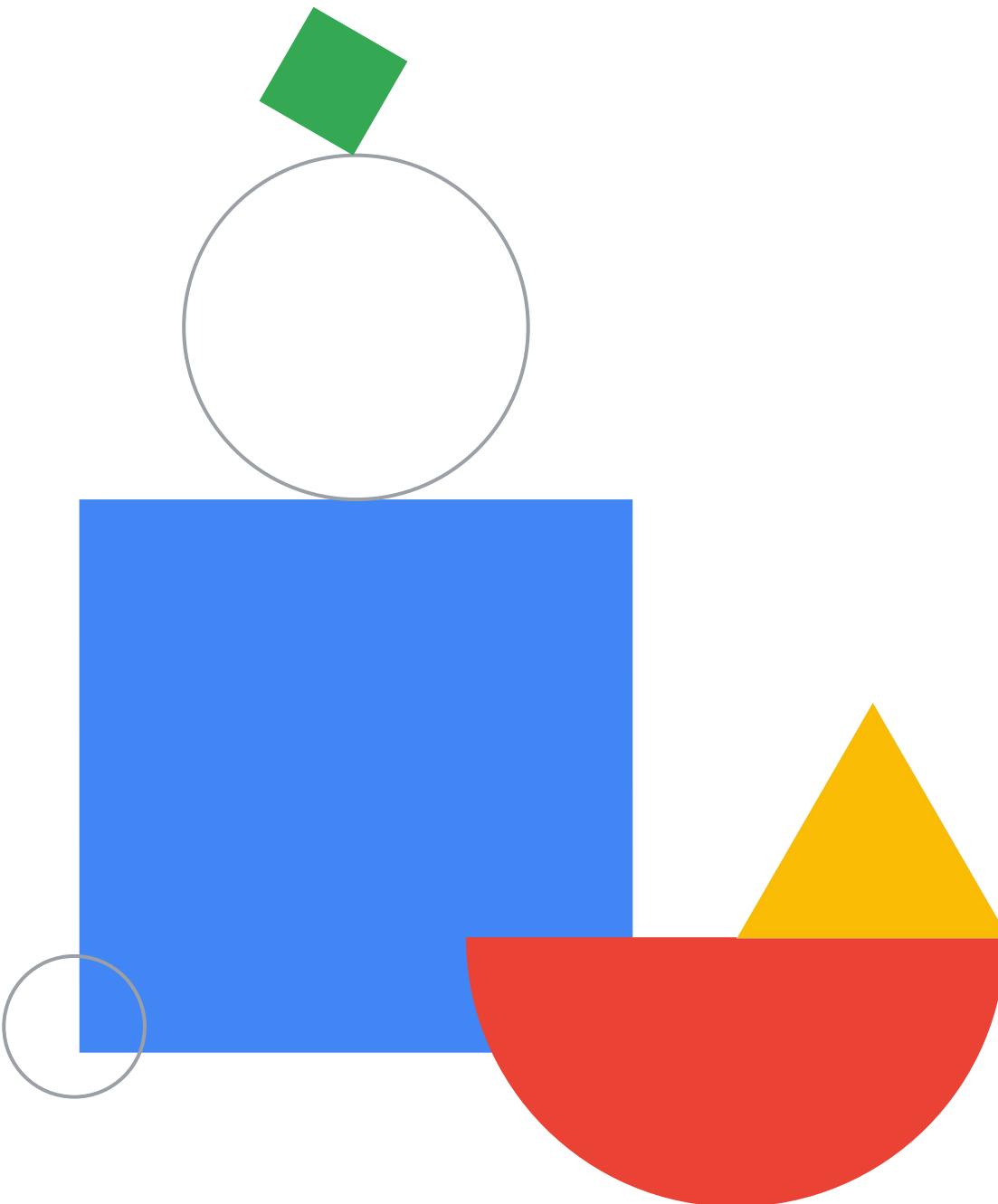


Launching into Machine Learning

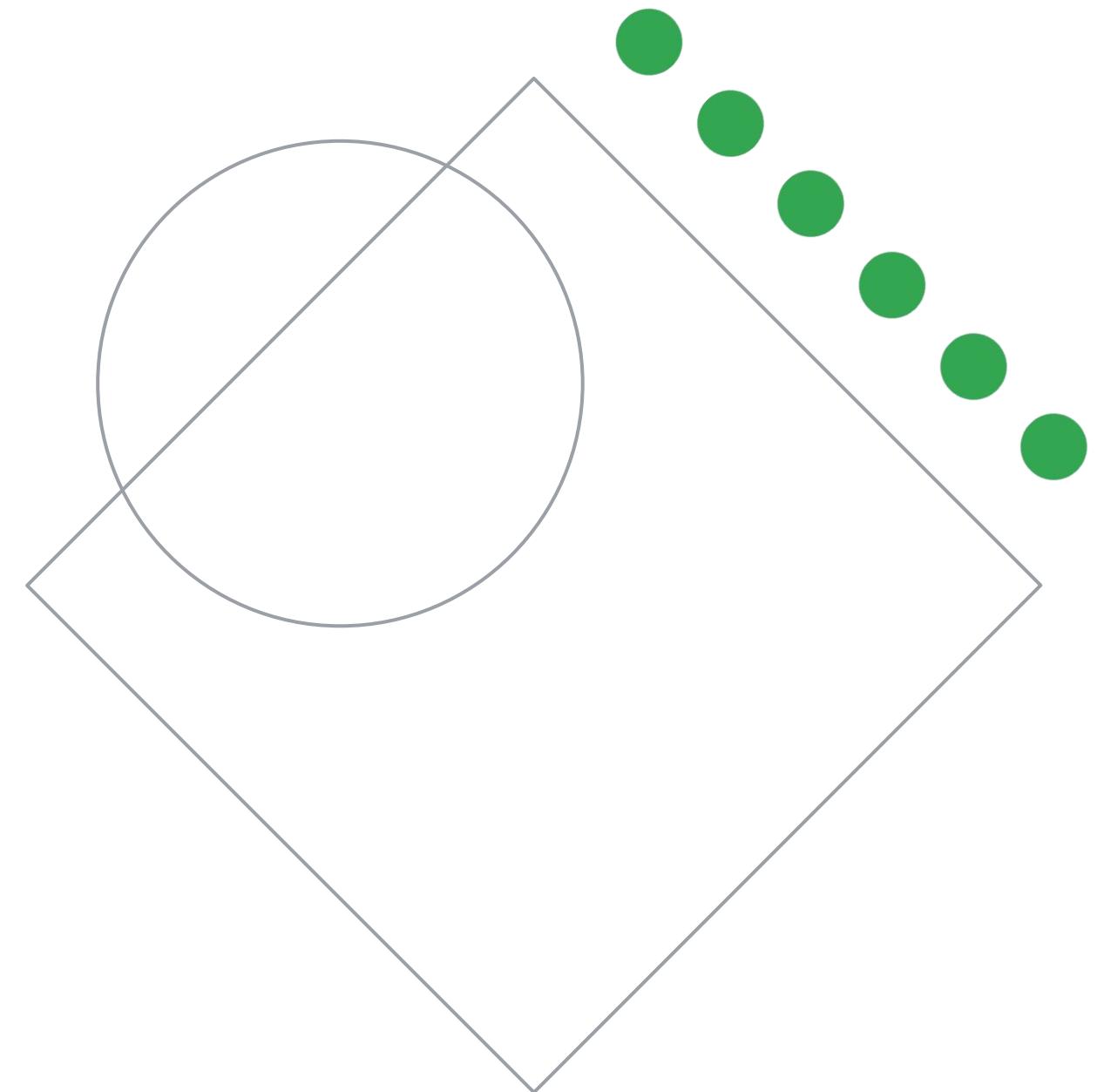


In this course, you learn to ...

- 01 Describe how to improve data quality
- 02 Perform exploratory data analysis
- 03 Build and train AutoML Models using Vertex AI
- 04 Build and train AutoML Models using BigQuery ML
- 05 Optimize and evaluate models using loss functions and performance metrics
- 06 Create repeatable and scalable training, evaluation, and test datasets



**Get to know your data:
Improve data through
Exploratory Data Analysis**

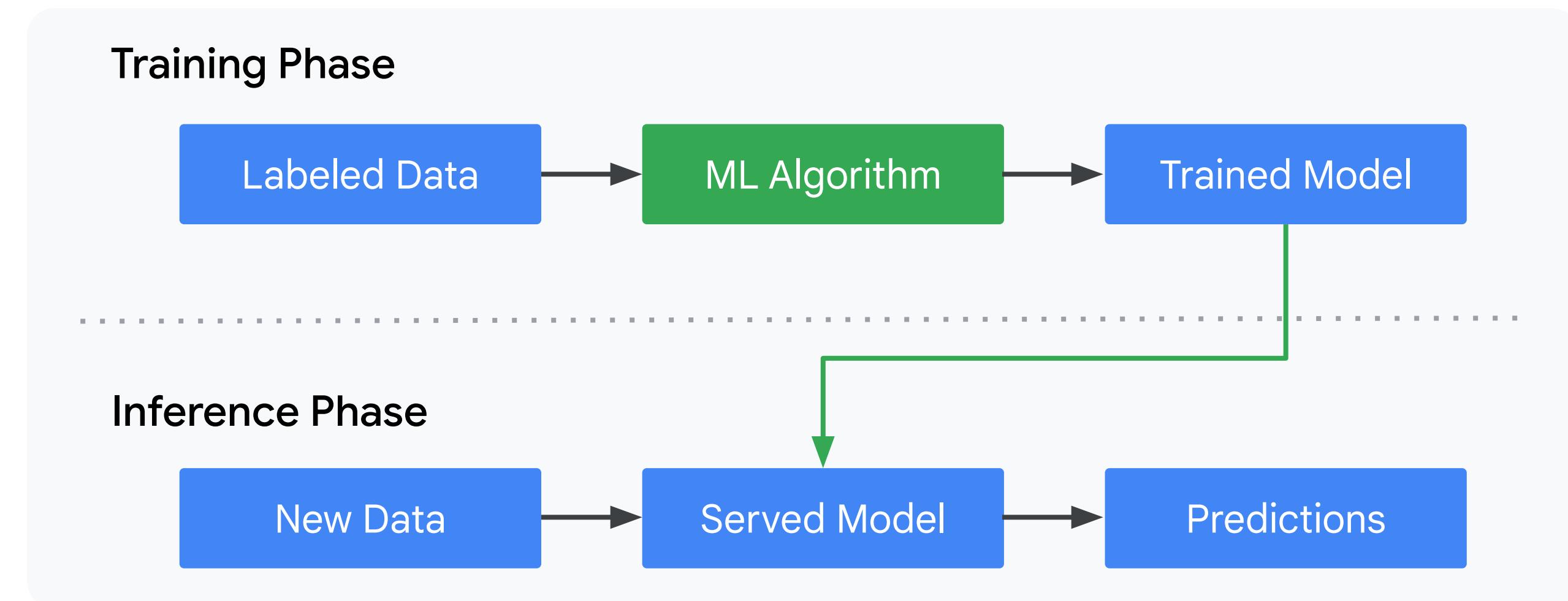


In this module, you learn to ...

- 01 Perform Exploratory Data Analysis
- 02 Categorize missing data strategies
- 03 Improve data quality

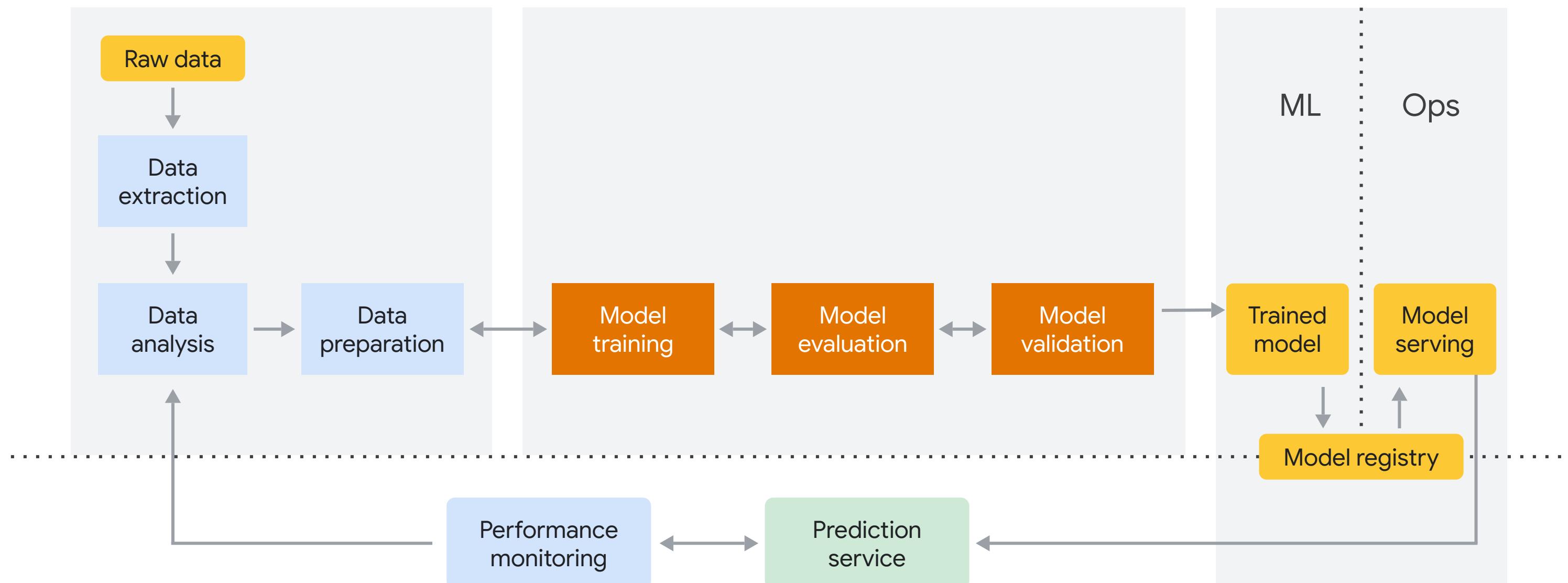


Machine learning phases



An ML pipeline recap

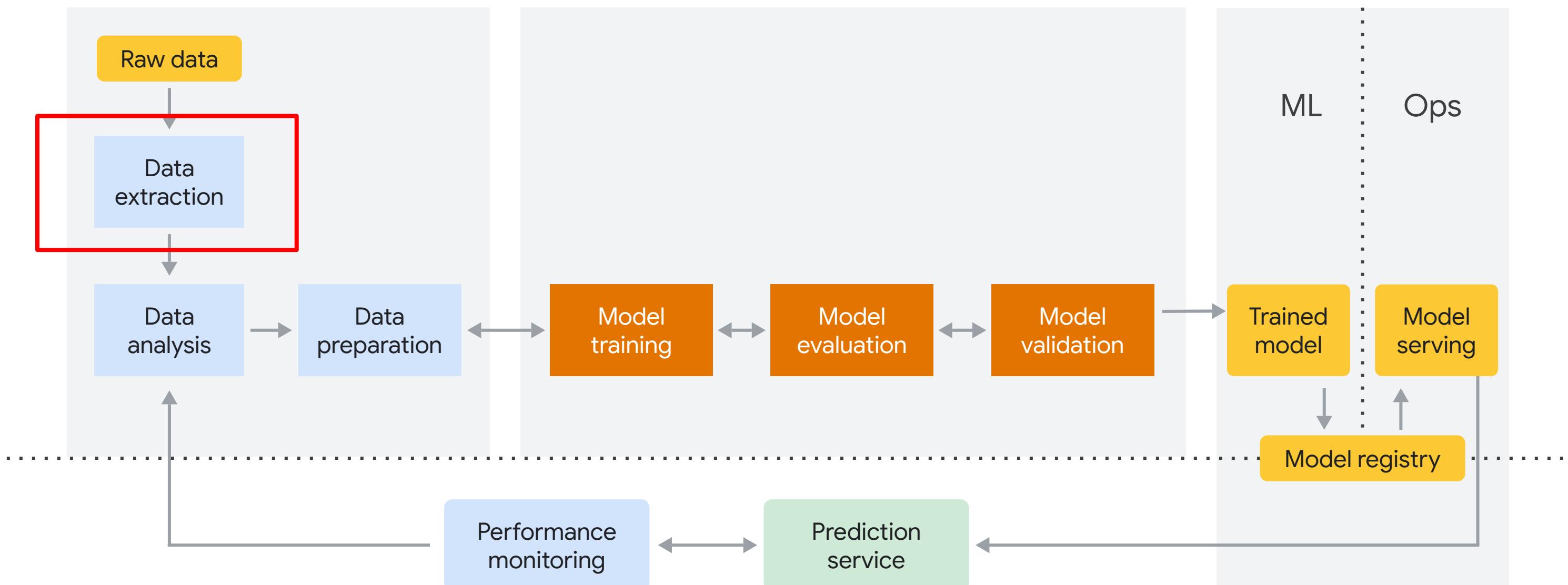
Staging/pre-production/production environments



Experimentation/development/test environments

An ML pipeline recap

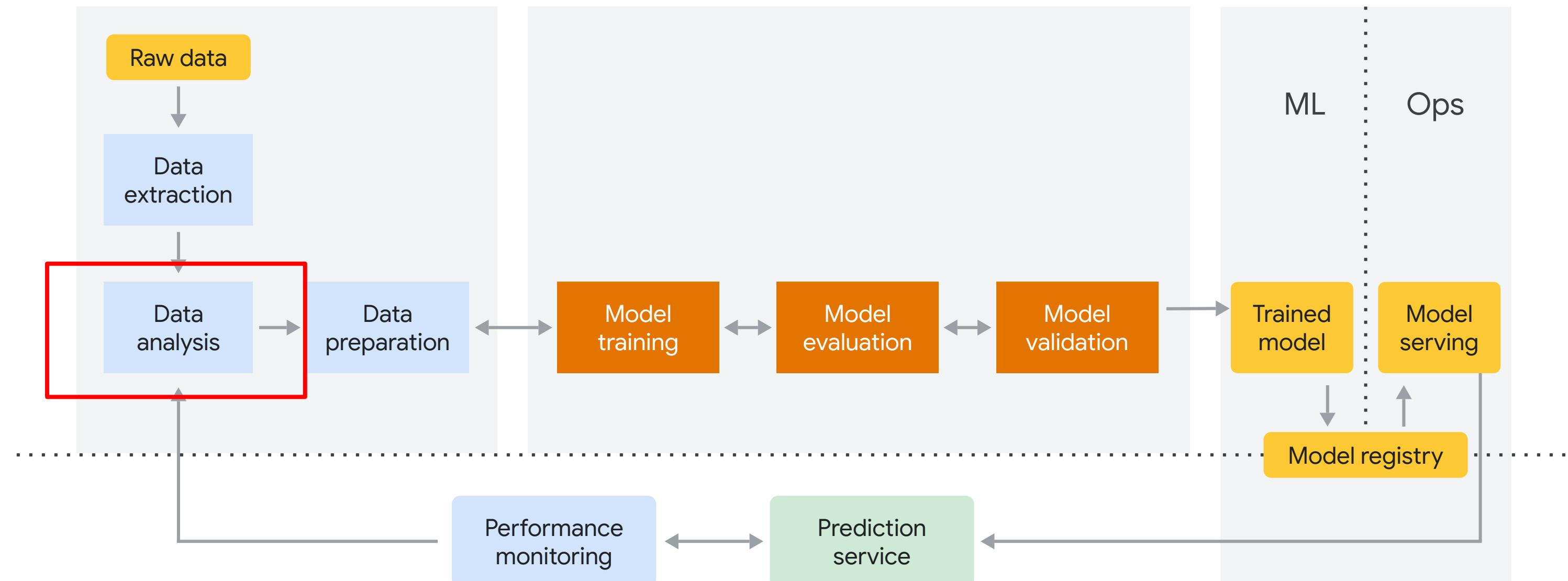
Staging/pre-production/production environments



Experimentation/development/test environments

An ML pipeline recap

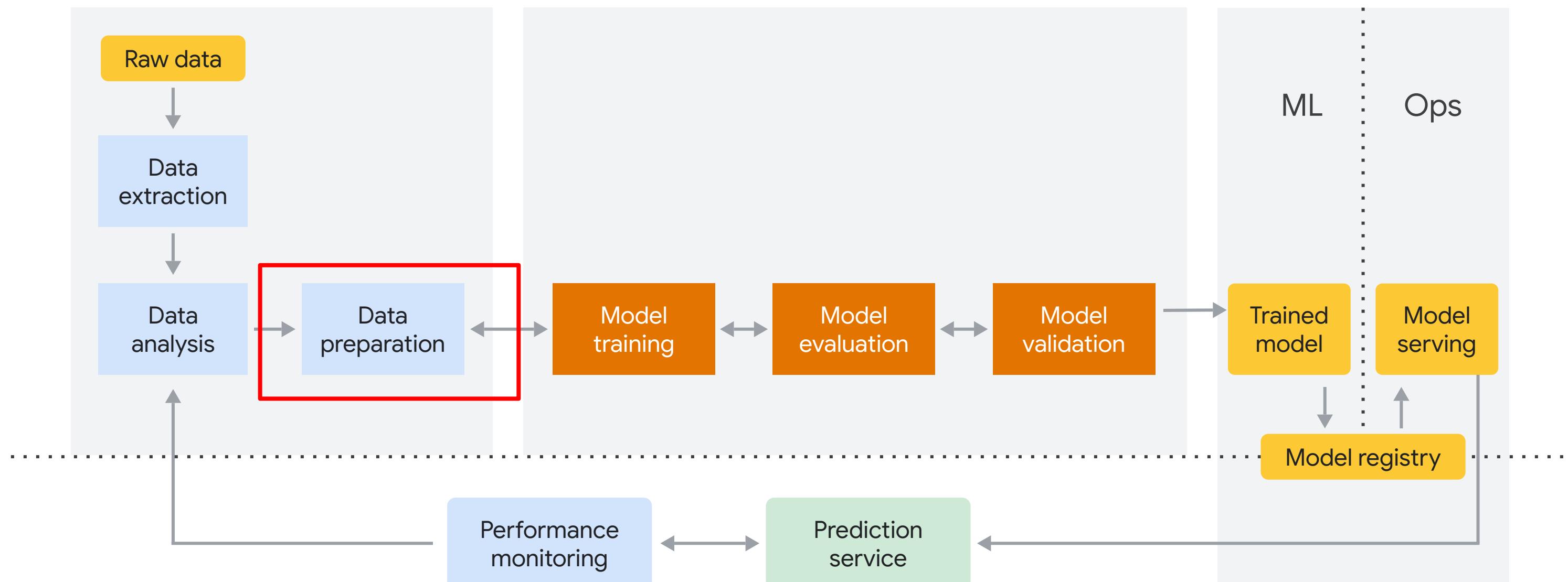
Staging/pre-production/production environments



Experimentation/development/test environments

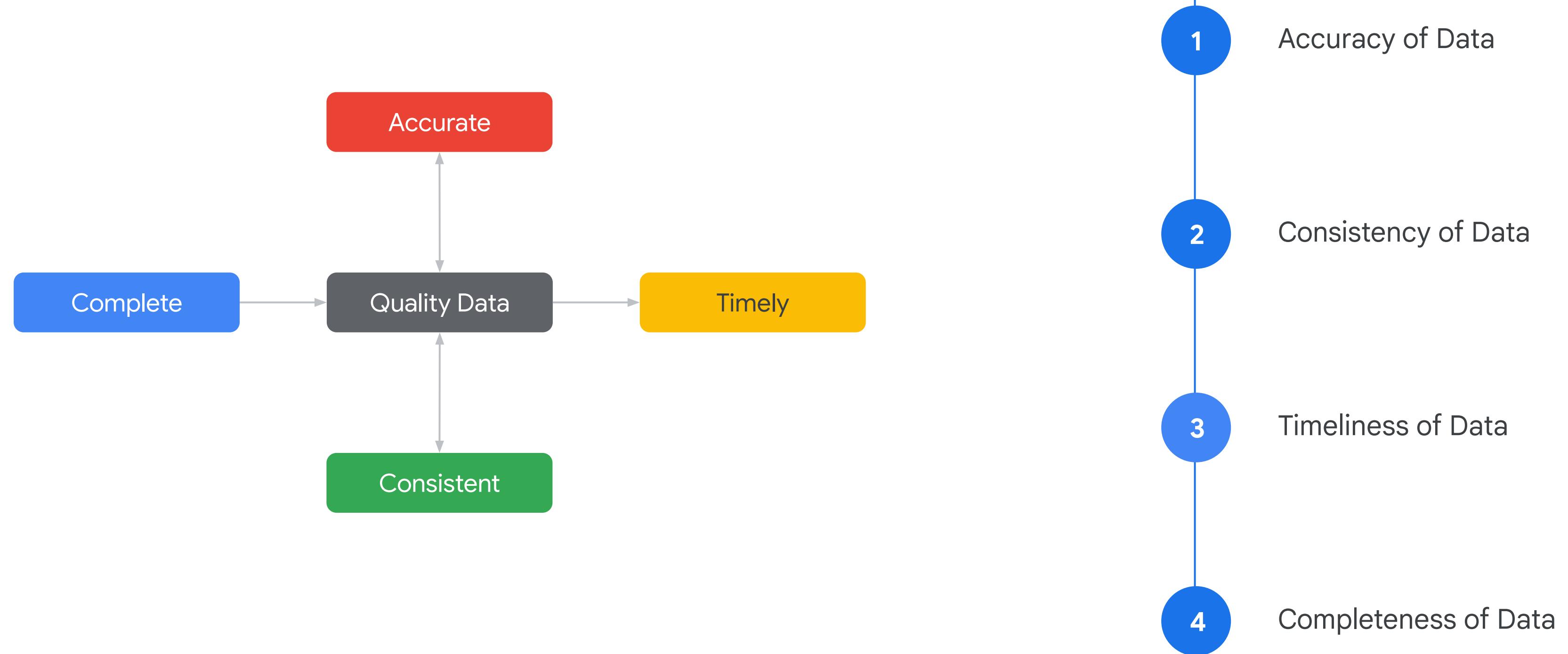
An ML pipeline recap

Staging/pre-production/production environments

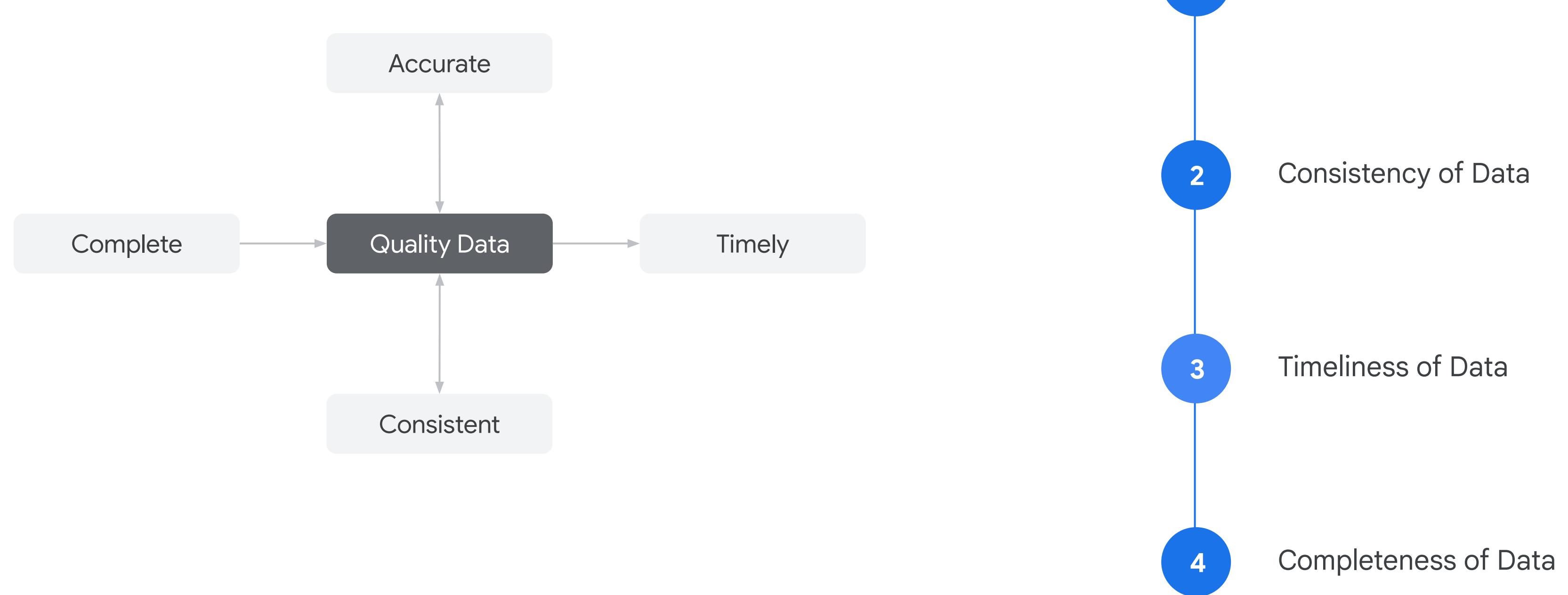


Experimentation/development/test environments

Attributes related to the data quality



Attributes related to the data quality



Attributes related to the data quality



Attributes related to the data quality



Attributes related to the data quality



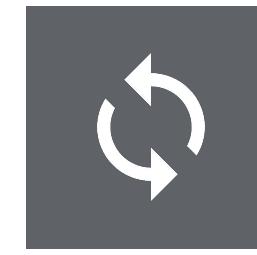
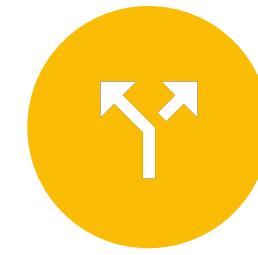
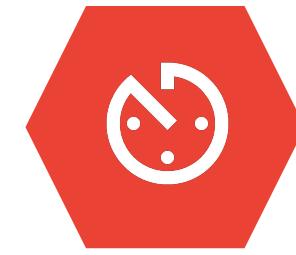
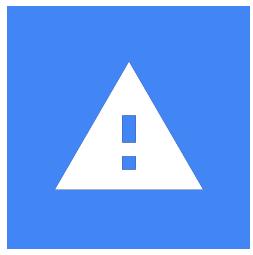
Attributes related to the data quality



Attributes related to the data quality



Ways to improve data quality



— 1 —

Resolve Missing
Values

— 2 —

Convert the
Date feature
column to
Datetime
Format

— 3 —

Parse date/time
features

— 4 —

Remove
unwanted values

— 5 —

Convert
categorical
columns to
“one-hot
encodings”

Missing values

Let's show the null values for all features in the DataFrame.

```
In [10]: df_transport.isnull().sum()  
Out[10]: Date           2  
          Zip Code       2  
          Model Year     2  
          Fuel            3  
          Make            3  
          Light_Duty      3  
          Vehicles         3  
          dtype: int64
```

Missing values

Let's show the null values for all features in the DataFrame.

```
In [10]: df_transport.isnull().sum()  
Out[10]: Date           2  
          Zip Code       2  
          Model Year     2  
          Fuel            3  
          Make            3  
          Light_Duty      3  
          Vehicles         3  
          dtype: int64
```

Missing values

```
In [11]: print (df_transport ['Date'])
          print (df_transport ['Date'].isnull())
          0      10/1/2018
          1      10/1/2018
          2          NaN
          3      10/1/2018
          4      10/1/2018
          ...
          994     6/7/2019
          995     6/8/2019
          996     6/10/2019
          997     6/11/2019
          998
          Name: Date, Length: 999, dtype: object
          0    False
          1    False
          2     True
          3    False
```

Missing values

```
In [14]: print ( "Rows      : " ,df_transport.shape [0])
          print ( "Columns   : " ,df_transport.shape [1])
          print ( "\nFeatures : \n" ,df_transport.columns.tolist())
          print ( "\nUnique values : \n" ,df_transport.nunique())
          print ( "\nMissing values : " , df_transport.isnull().sum().values.sum())
          Rows      : 999
          Columns   : 7
```

Features :

```
[`Date`, `Zip Code`, `Model Year`, `Fuel`, `Make`, `Light_Duty`, `Vehicles`]
```

Unique values :

```
    Date        248
    Zip Code     6
    Model Year    15
    Fuel         8
    Make        43
    Light_Duty    3
    Vehicles     210
    dtype: int64
```

Missing values : 18

Missing values

```
In [14]: print ("Rows      : " ,df_transport.shape [0])
          print ("Columns   : " ,df_transport.shape [1])
          print ("\nFeatures : \n" ,df_transport.columns.tolist())
          print ("\nUnique values : \n",df_trasport.nunique())
          print ("\nMissing values : " , df_transport.isnull().sum().values.sum())
          Rows      : 999
          Columns   : 7
```

Features :

```
[`Date`, `Zip Code`, `Model Year`, `Fuel`, `Make`, `Light_Duty`, `Vehicles`]
```

Unique values :

Date	248
Zip Code	6
Model Year	15
Fuel	8
Make	43
Light_Duty	3
Vehicles	210

dtype: int64

Missing values : 18

Missing values = “Messy data”

```
[5]: df_transport = pd.read_csv(`../data/transport/untidy_vehicle_data.csv`)  
df_transport.head() # Output the first five rows.
```

	Date	Zip Code	Model Year	Fuel	Make	Light_Duty	Vehicles
0	10/1/2018	90000	2006	Gasoline	OTHER/UNK	NaN	1.0
1	10/1/2018	NaN	2014	Gasoline	NaN	Yes	1.0
2	NaN	90000	NaN	Gasoline	OTHER/UNK	Yes	Nan
3	10/1/2018	90000	2017	Gasoline	OTHER/UNK	Yes	1.0
4	10/1/2018	90000	<2006	Diesel and Diesel Hybrid	OTHER/UNK	No	55.0

Date and time

```
In [6]: df_transport.info()
<class `pandas.core.frame.DataFrame`>
RangeIndex: 999 entries, 0 to 998
Data columns (total 7 columns):
Date                997 non-null object
Zip Code             997 non-null object
Model Year           997 non-null object
Fuel                 996 non-null object
Make                 996 non-null object
Light_Duty            996 non-null object
Vehicles              996 non-null float64
dtype: float64(1), object(6)
memory usage: 54.8+ KB
```

Convert the Date feature column to a **datetime** **format**

We can convert it to the
datetime with the
`to_datetime()` function in
Pandas.

```
In [19]: # TODO 2a
         df_transport[‘Date’] = pd.to_datetime(df_transport[‘Date’],
                                                 format= %m/%d/%Y)

In [20]: # TODO 2b
         df_transport.info() # Date is now converted
<class ‘pandas.core.frame.DataFrame’>
RangeIndex: 999 entries, 0 to 998
Data columns (total 7 columns):
 Date           999 non-null datetime64[ns]
 Zip Code       999 non-null object
 Model Year    999 non-null object
 Fuel           999 non-null object
 Make           999 non-null object
 Light_Duty     999 non-null object
 Vehicles       999 non-null float64
 dtype: datetime64[ns](1), float64(1), object(5)
memory usage: 54.8+ KB
```

Parse date

Let's parse Date into three columns (e.g., year, month, and day.)

```
In [21]: df_transport['year'] = df_transport['Date'].dt.year  
df_transport['month'] = df_transport['Date'].dt.month  
df_transport['day'] = df_transport['Date'].dt.day  
#df['hour'] = df['date'].dt.hour - you could use if your  
#df['minute'] = df['date'].dt.minute - you could use this if  
df_transport.info()  
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 999 entries, 0 to 998  
Data columns (total 10 columns):  
 Date          999 non-null datetime64[ns]  
 Zip Code      999 non-null object  
 Model Year    999 non-null object  
 Fuel          999 non-null object  
 Make          999 non-null object  
 Light_Duty    999 non-null object  
 Vehicles      999 non-null float64  
 Year          999 non-null int64  
 Month         999 non-null int64  
 Day           999 non-null int64  
 dtypes: datetime64[ns](1), float64(1), int64(3), objects(5)  
 memory usage: 78.2+ KB
```

Unwanted characters - Model Year

Let's investigate a bit more of our data by using the `.groupby()` function.

```
In [9]: grouped_data = df_transport.groupby(['Zip Code', 'Model Year', 'Fuel', 'Make', 'Light_Duty', 'Vehicles'])  
df_transport.groupby('Fuel').first() # Get the first entry for each month.
```

```
Out [9]:
```

Fuel	Date	Zip Code	Model Year	Make	Light_Duty	Vehicles
Battery Electric	10/1/2018	90000	<2006	OTHER/UNK	No	4.0
Diesel and Diesel Hybrid	10/1/2018	90000	<2006	OTHER/UNK	No	55.0
Flex-Fuel	10/14/2018	90001	2007	Type_A	Yes	78.0
Gasoline	10/1/2018	90000	2006	OTHER/UNK	Yes	1.0
Hybrid Gasoline	10/24/2018	90001	2009	OTHER/UNK	Yes	18.0
Natural Gas	10/25/2018	90001	2009	OTHER/UNK	No	2.0
Other	10/8/2018	90000	<2006	OTHER/UNK	Yes	6.0
Plug-in Hybrid	11/2/2018	90001	2012	OTHER/UNK	Yes	1.0

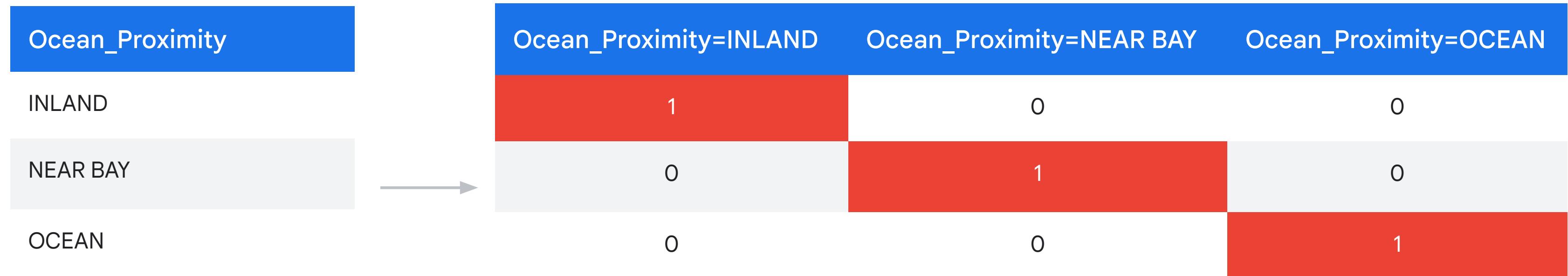
Categorical data - “Yes/No”

```
df_transport = pd.read_csv(`../data/transport/untidy_vehicle_data.csv`)
df_transport.head() # Output the first five rows.
```

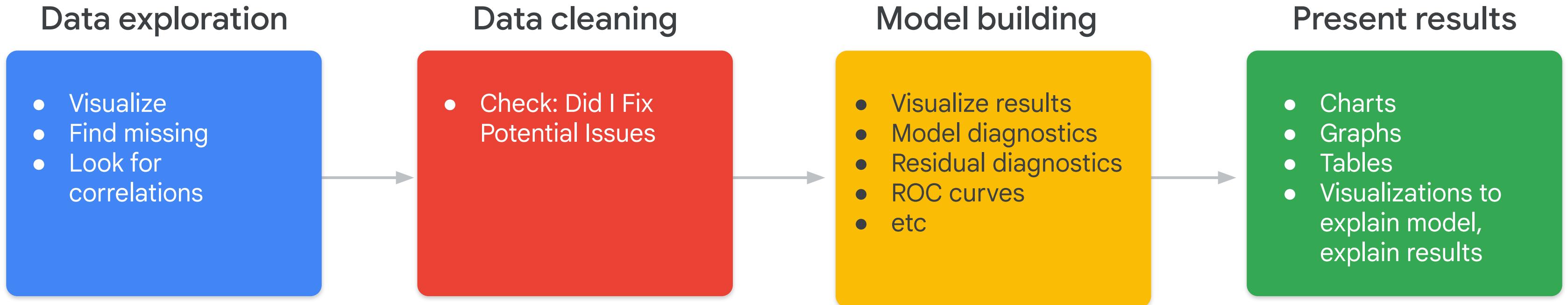
	Date	Zip Code	Model Year	Fuel	Make	Light_Duty	Vehicles
0	10/1/2018	90000	2006	Gasoline	OTHER/UNK	NaN	1.0
1	10/1/2018	NaN	2014	Gasoline	NaN	Yes	1.0
2	NaN	90000	NaN	Gasoline	OTHER/UNK	Yes	Nan
3	10/1/2018	90000	2017	Gasoline	OTHER/UNK	Yes	1.0
4	10/1/2018	90000	<2006	Diesel and Diesel Hybrid	OTHER/UNK	No	55.0

Categorical features

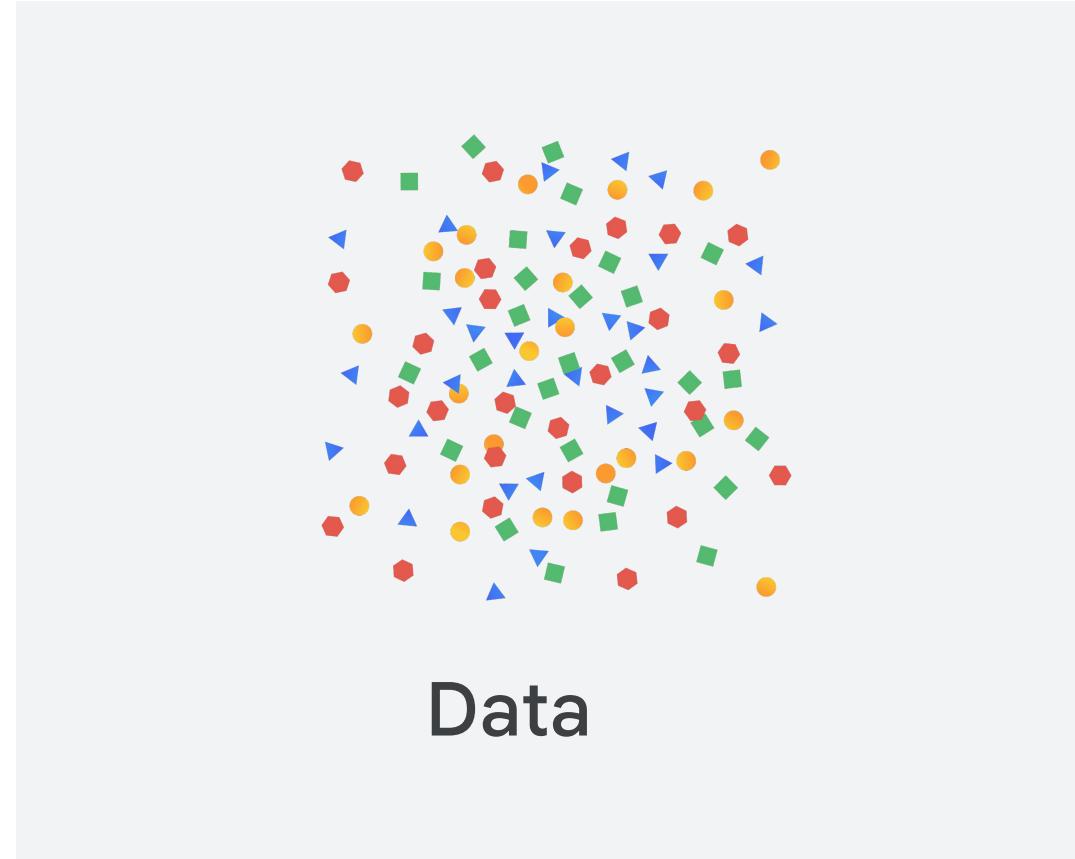
One-hot encoding



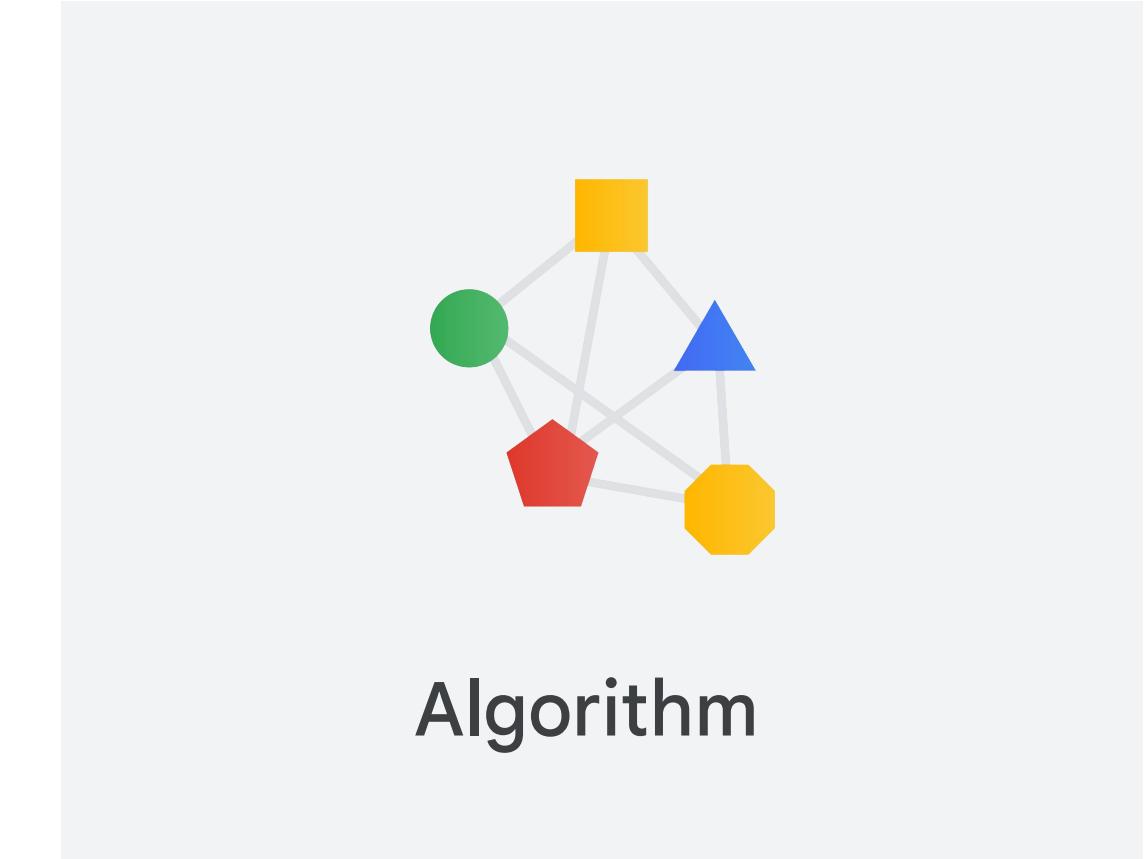
Data quality



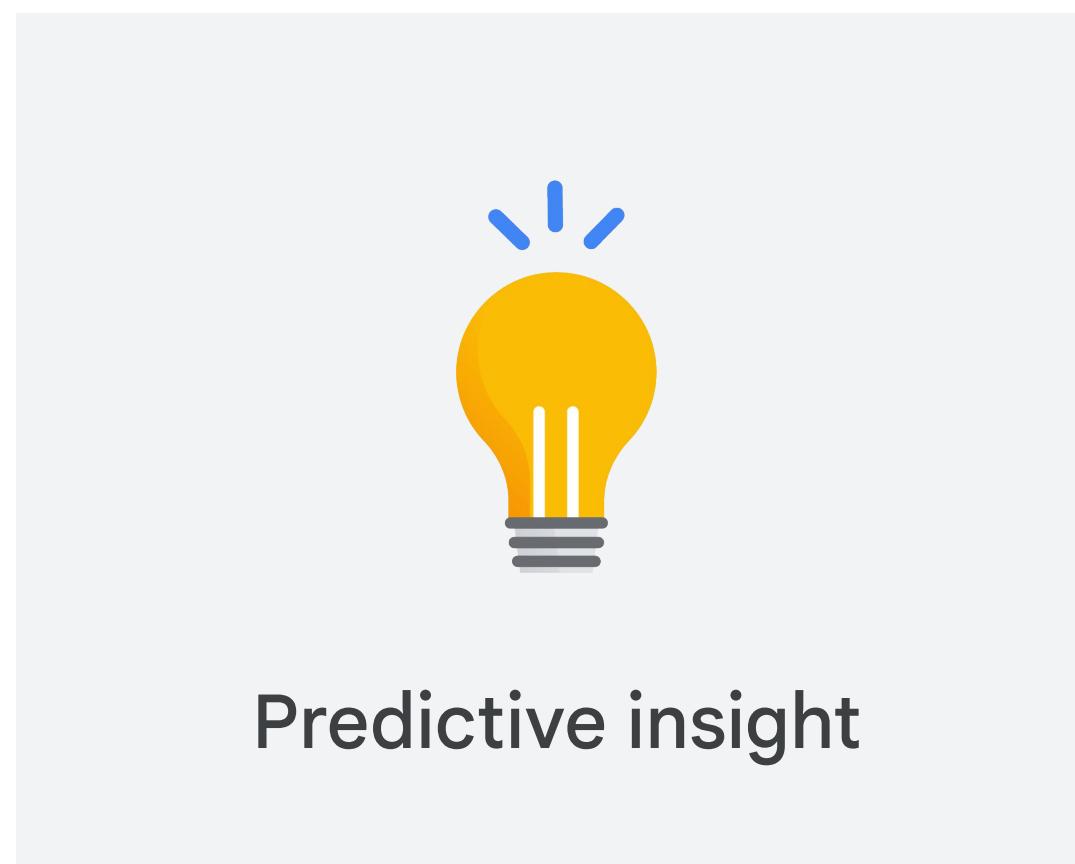
Machine learning is a way to use standard algorithms to derive predictive insights from data and make repeated decisions.



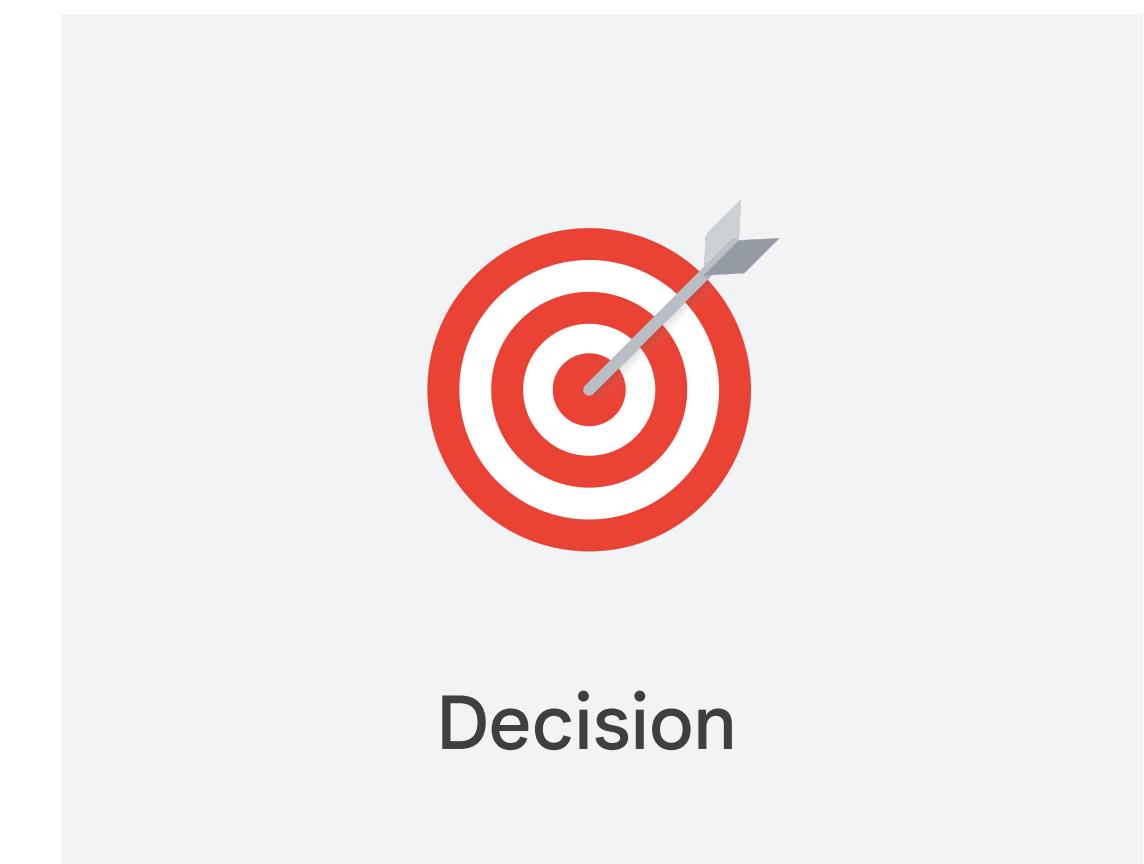
Data



Algorithm

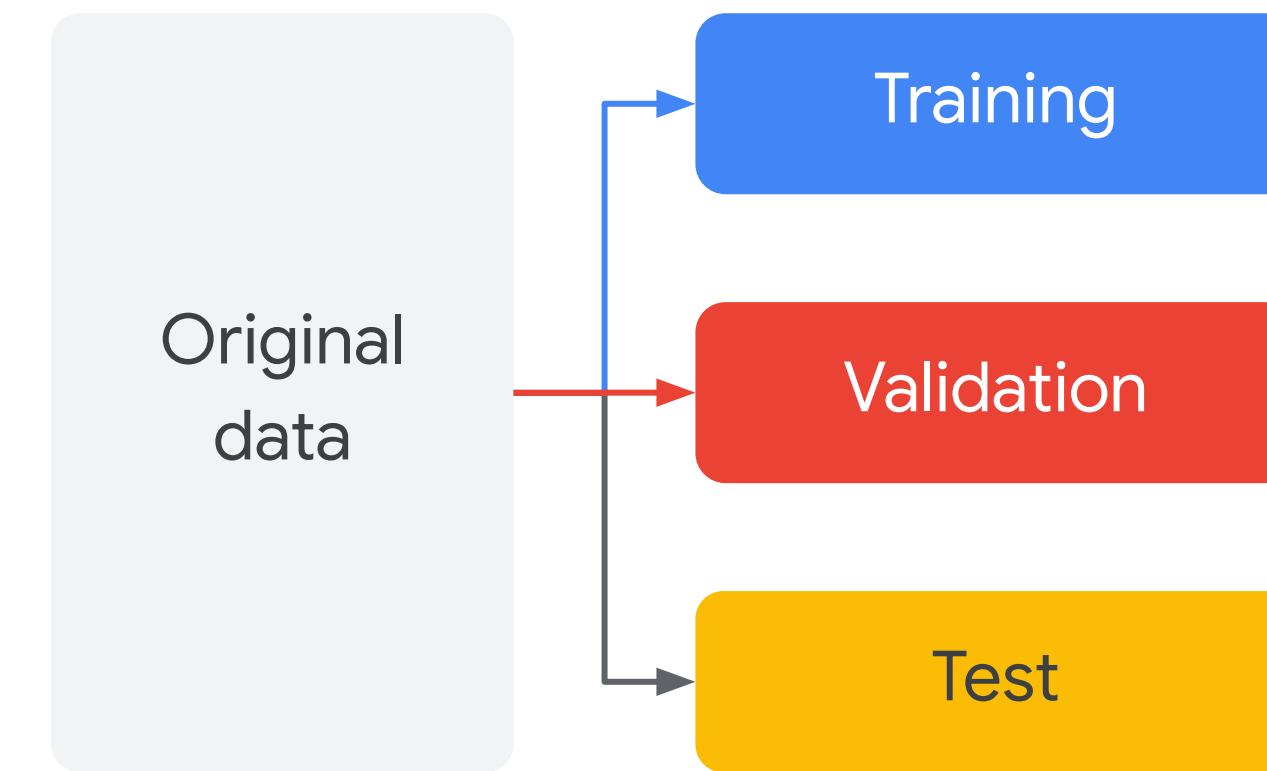
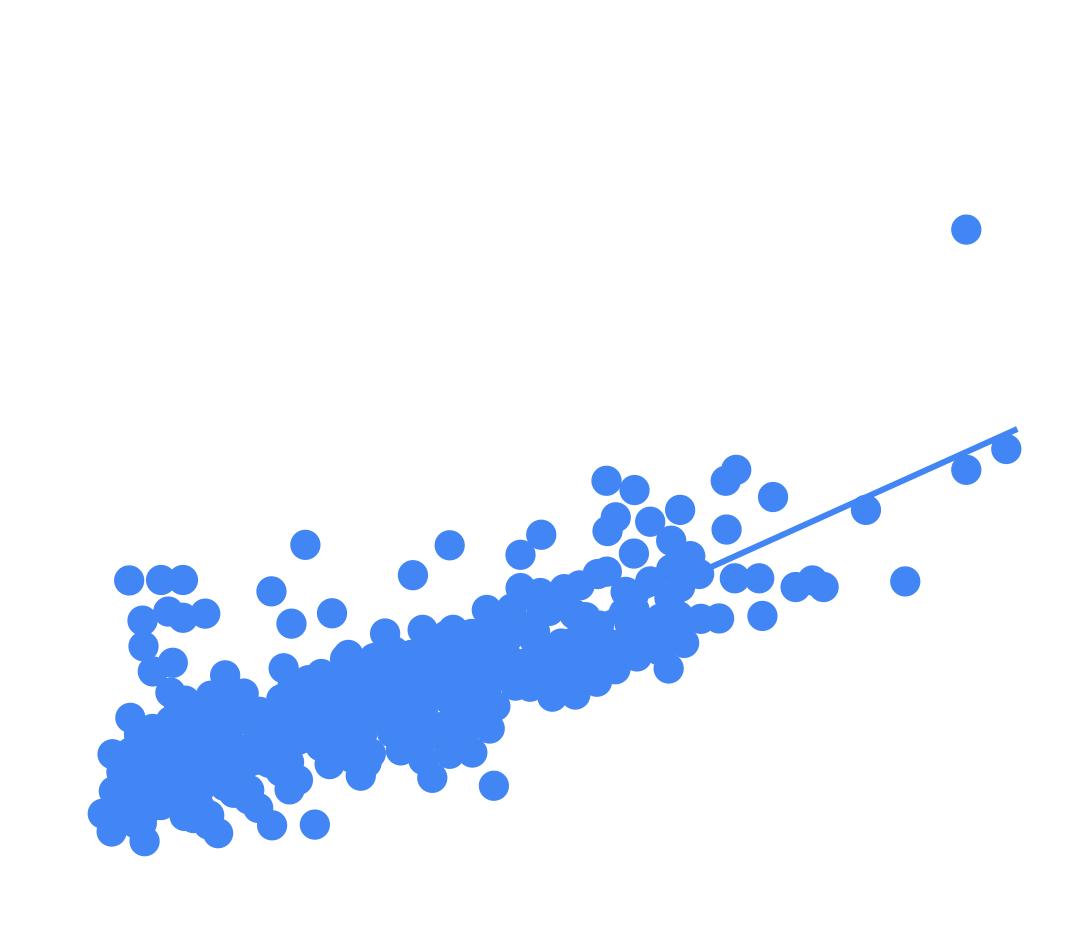


Predictive insight

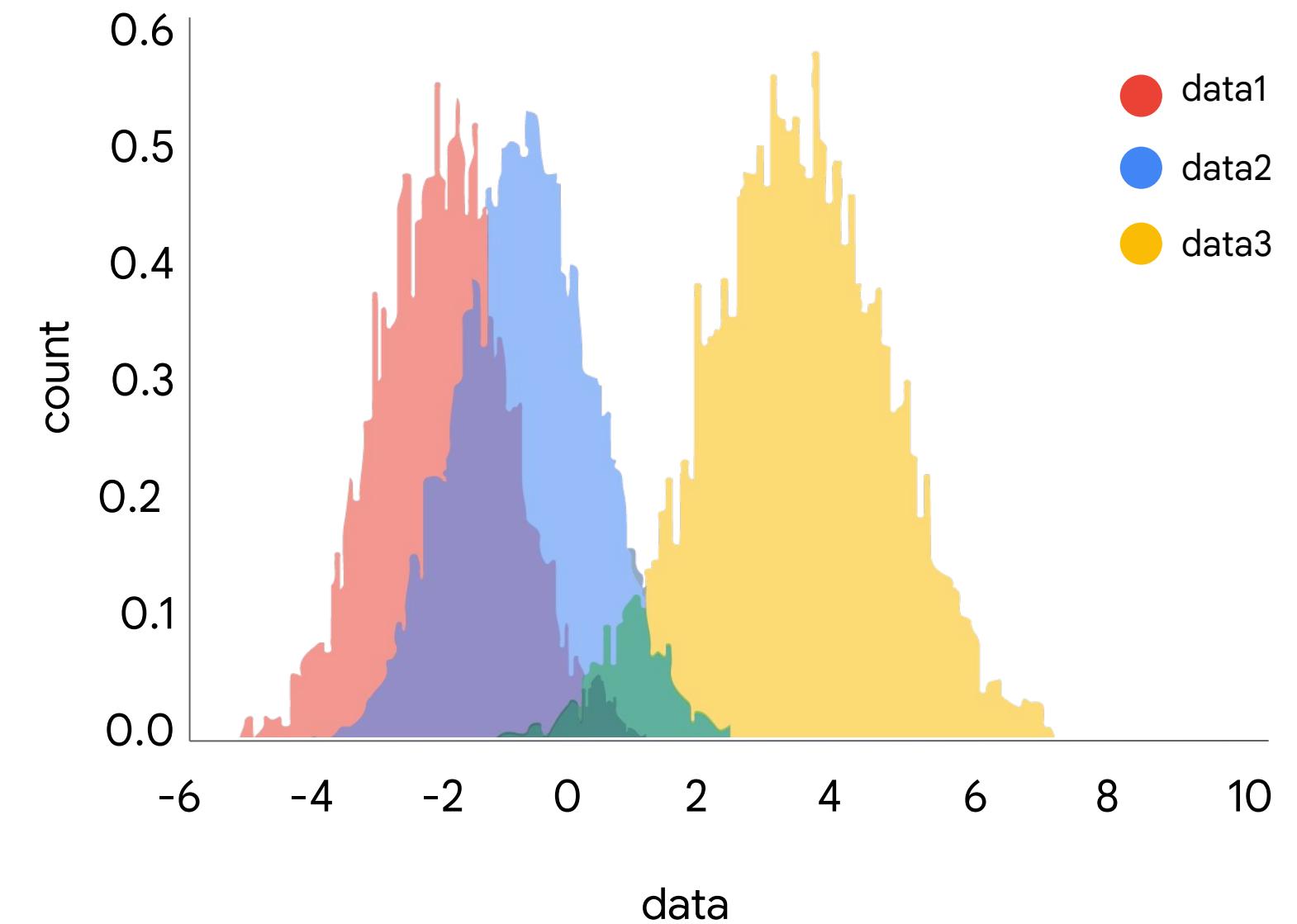
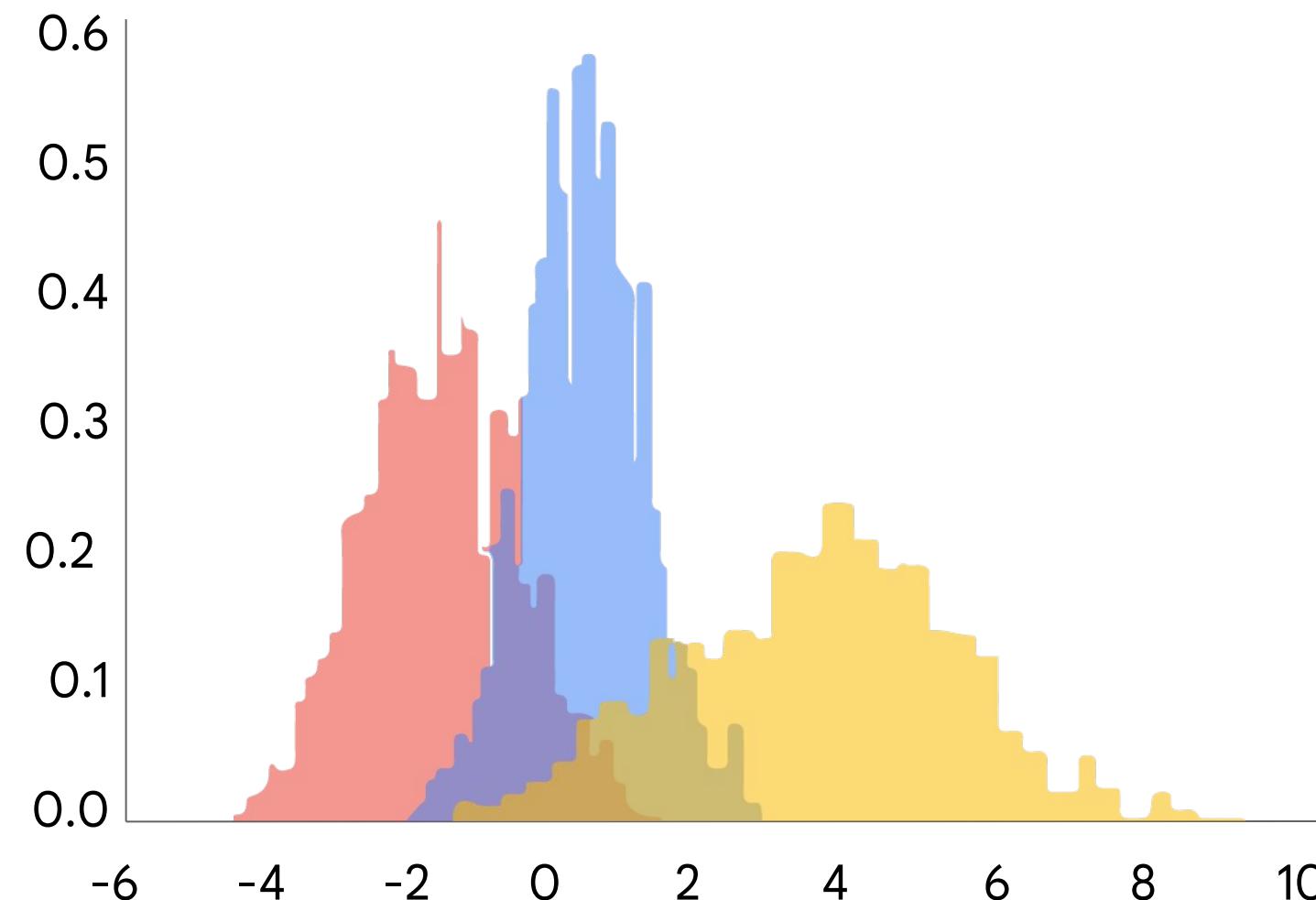


Decision

The importance of data quality



Exploratory data analysis (EDA)



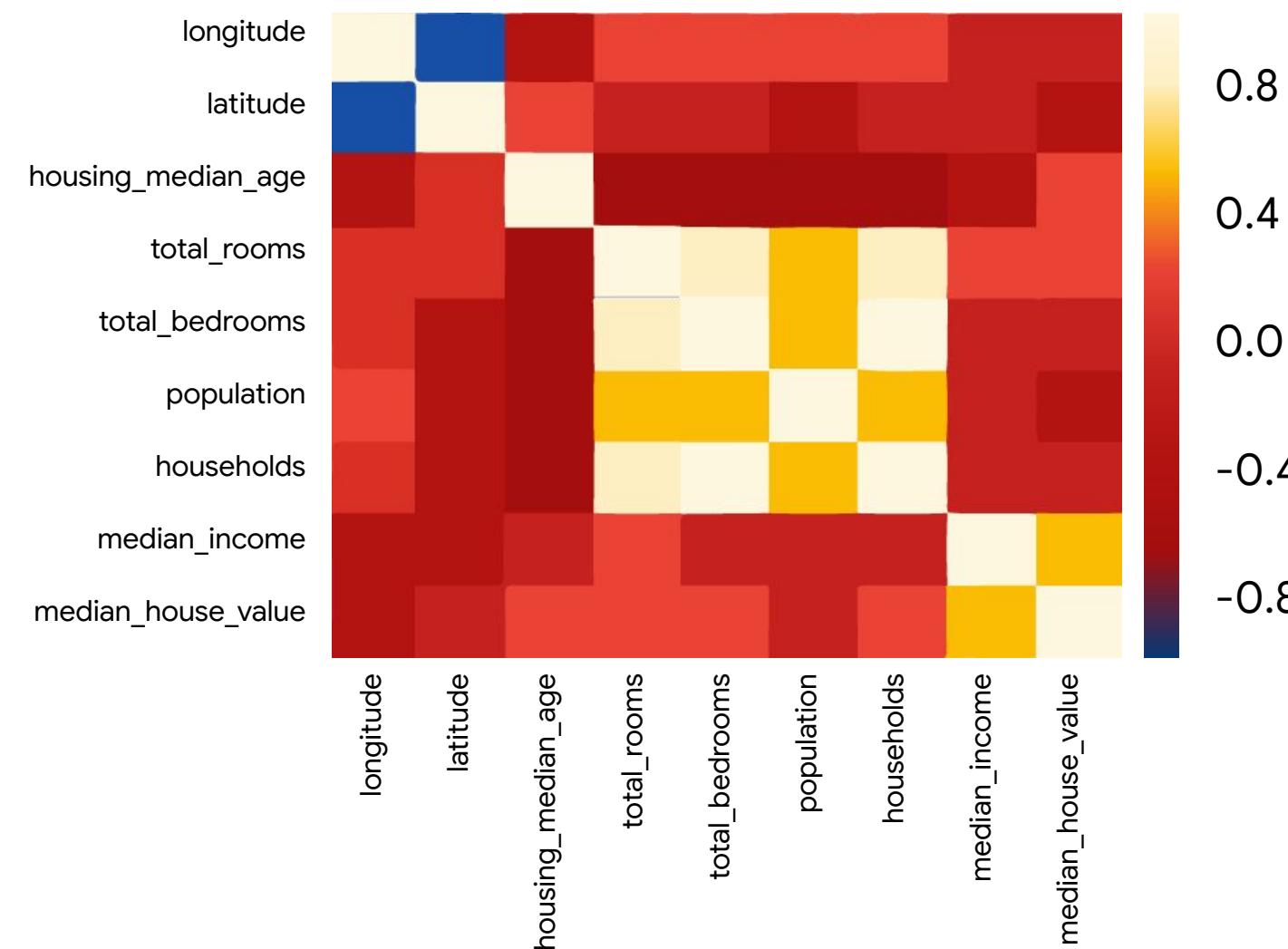
Exploratory data analysis (EDA)

[12]:

```
sns.heatmap(df_USAhousing.corr())
```

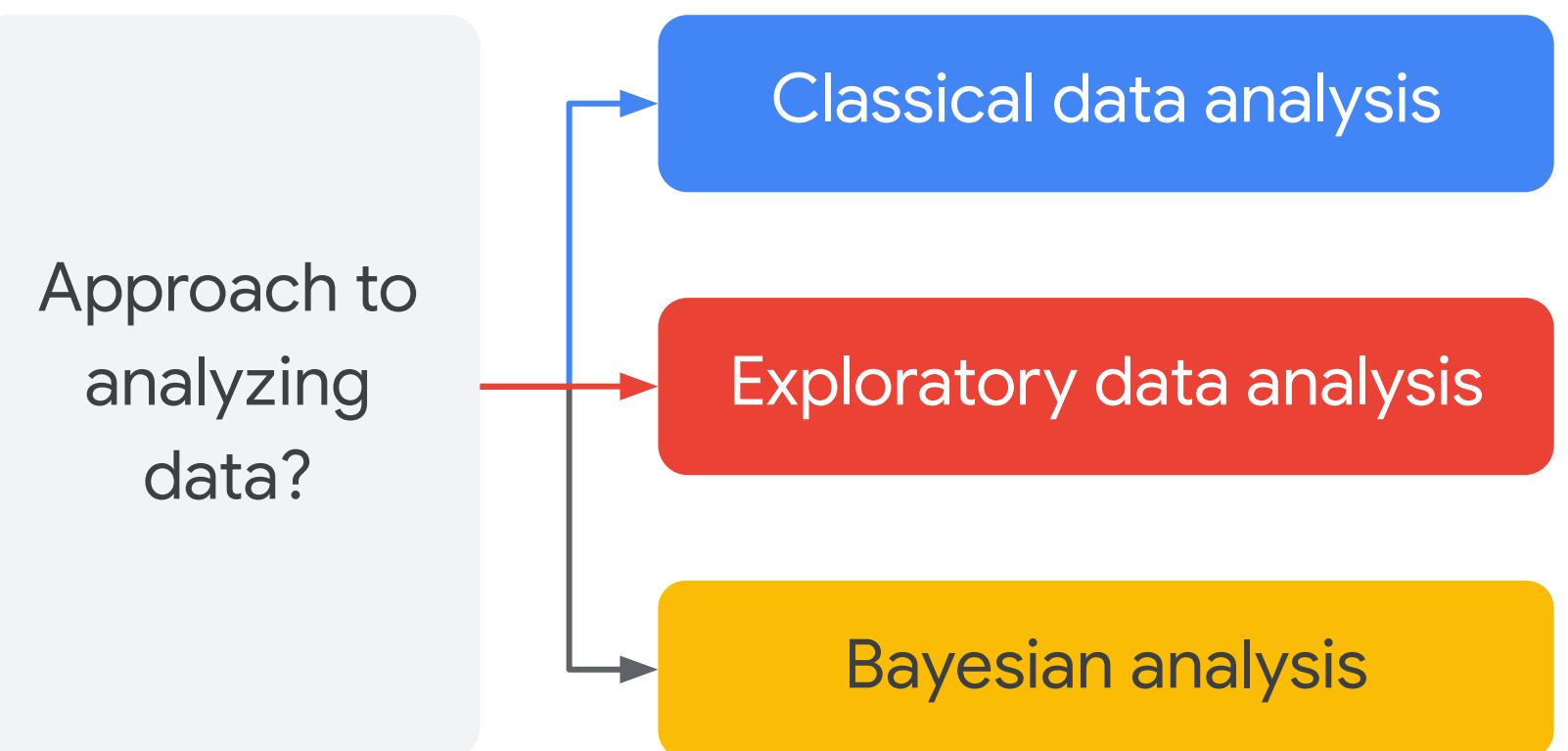
[12]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f25ba7c1dd8>
```



CDA vs. EDA vs. Bayesian?

What other approaches exist and how
does exploratory data analysis differ from
these other approaches?



Problem => Data => Model => Analysis => Conclusions

CDA vs. EDA vs. Bayesian?

What other approaches exist and how
does exploratory data analysis differ from
these other approaches?

Approach to
analyzing
data?

Classical data analysis

Problem => Data => Analysis => Model => Conclusions

CDA vs. EDA vs. Bayesian?

What other approaches exist and how
does exploratory data analysis differ from
these other approaches?

Approach to
analyzing
data?



Exploratory data analysis

Problem => Data => Model => Prior Distribution =>
Analysis => Conclusions

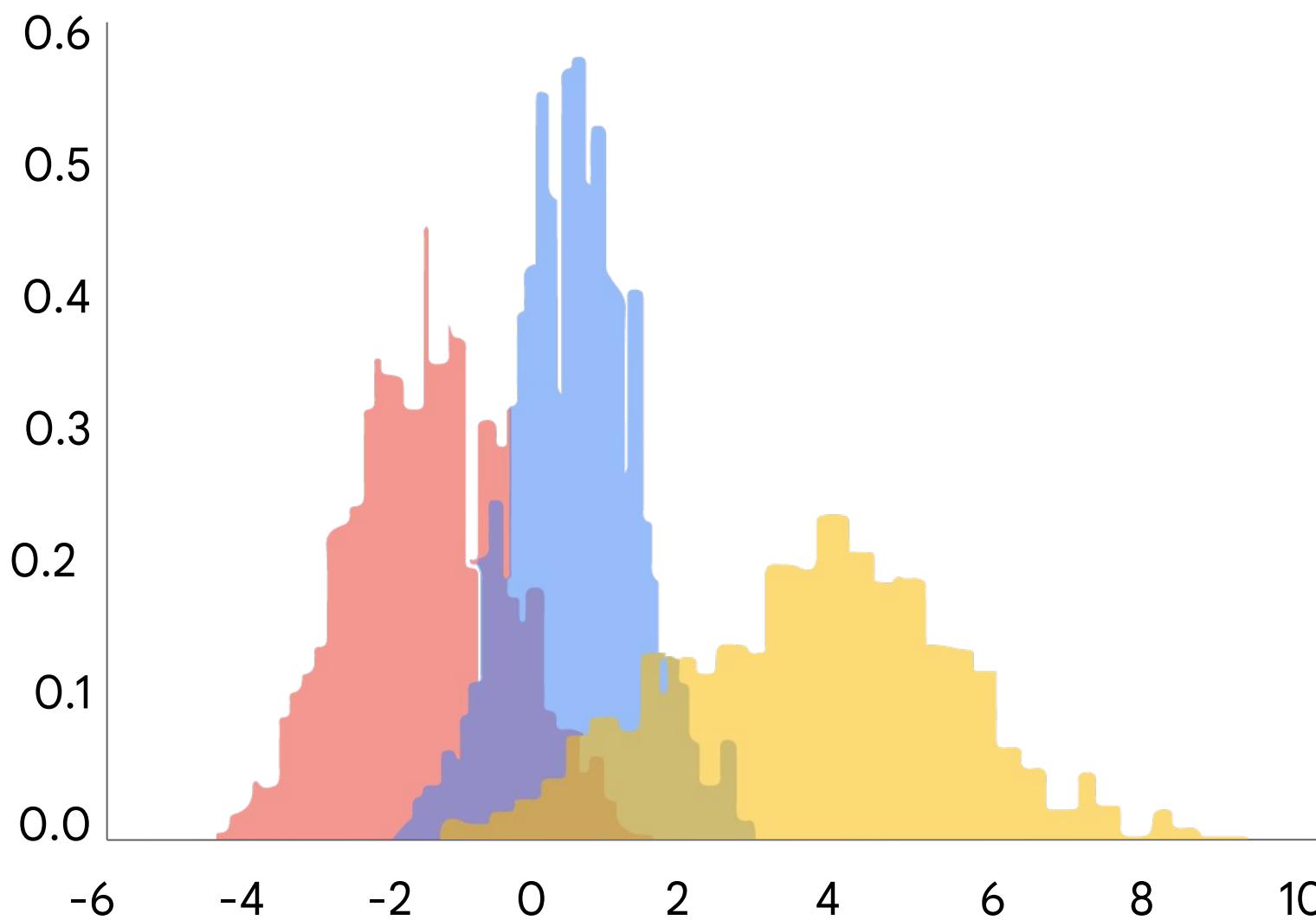
CDA vs. EDA vs. Bayesian?

What other approaches exist and how
does exploratory data analysis differ from
these other approaches?

Approach to
analyzing
data?

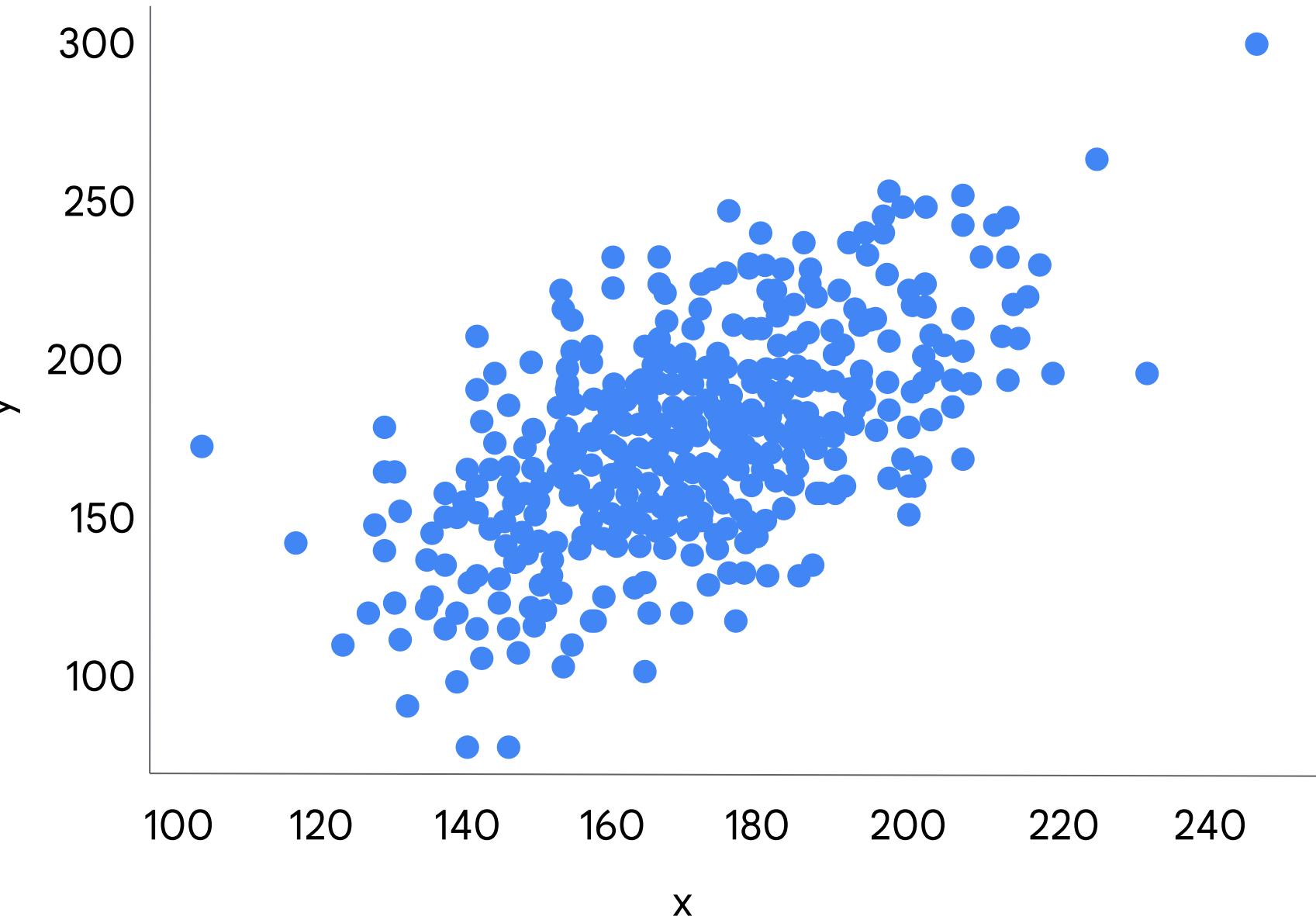
Bayesian analysis

How is EDA used in machine learning?

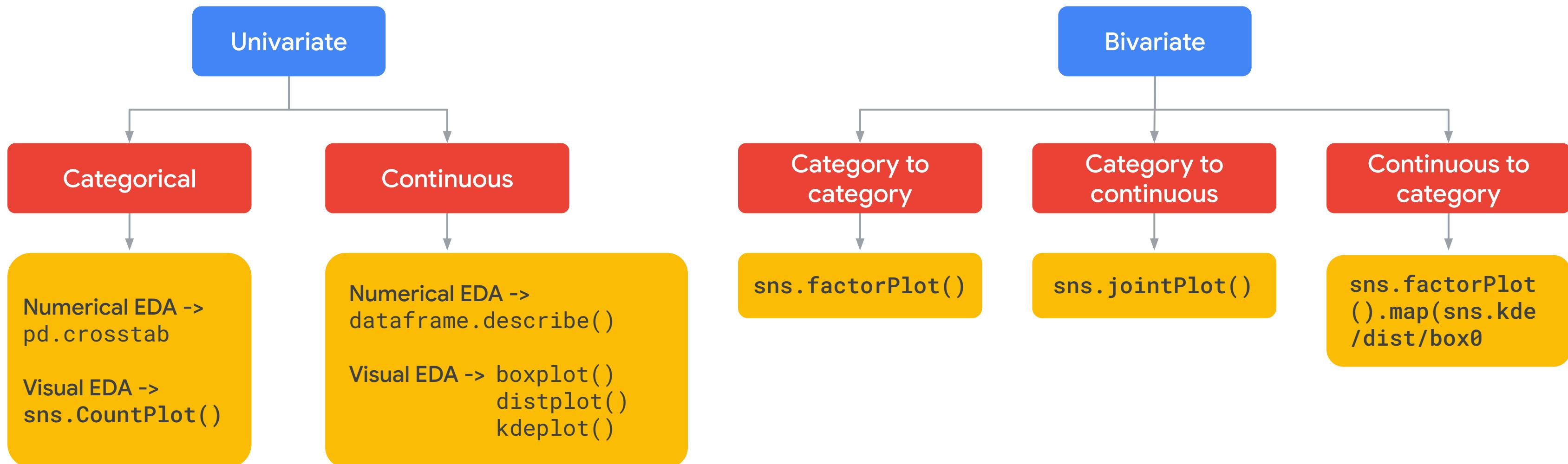


How is EDA used in machine learning?

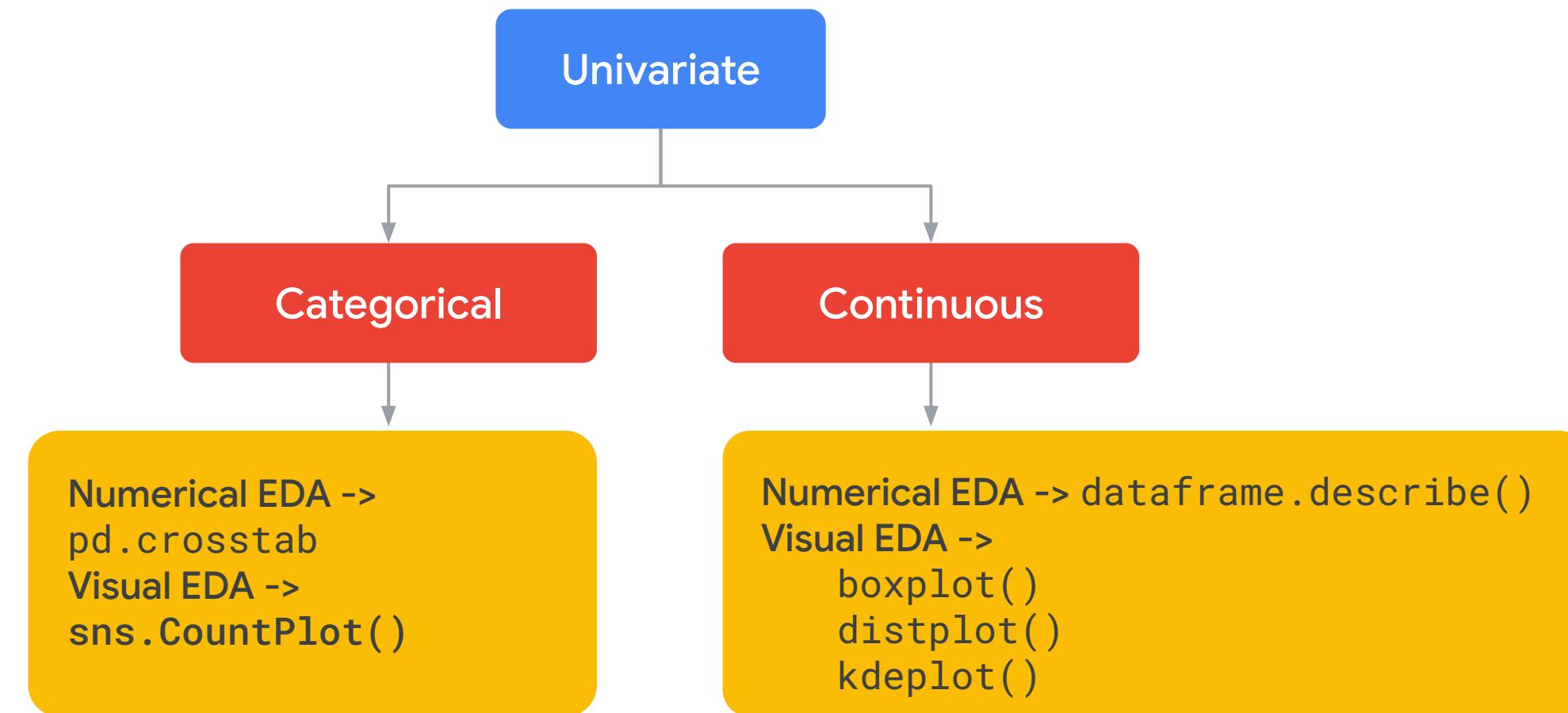
Scatter plot with Seaborn



Exploratory data analysis

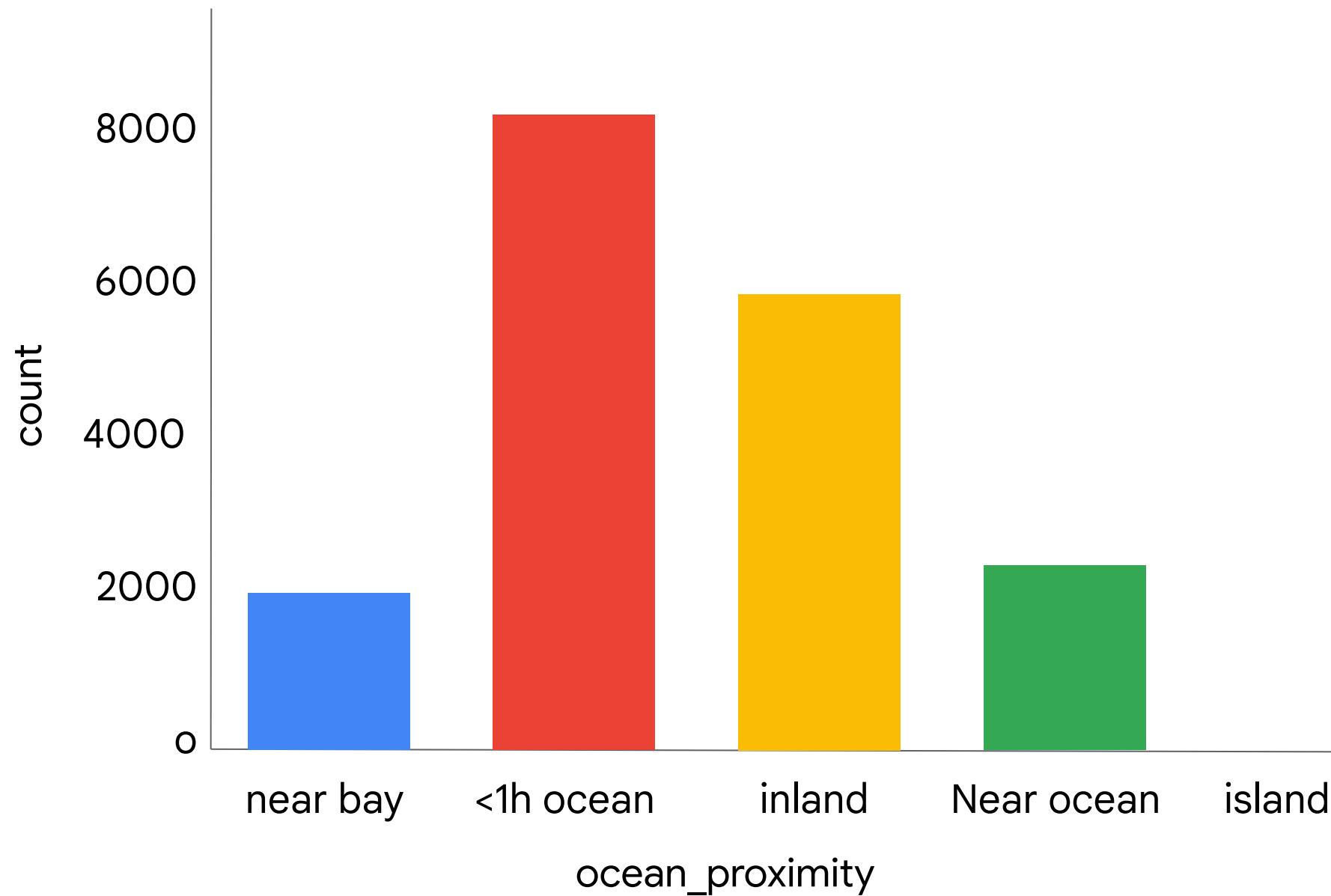


Exploratory data analysis

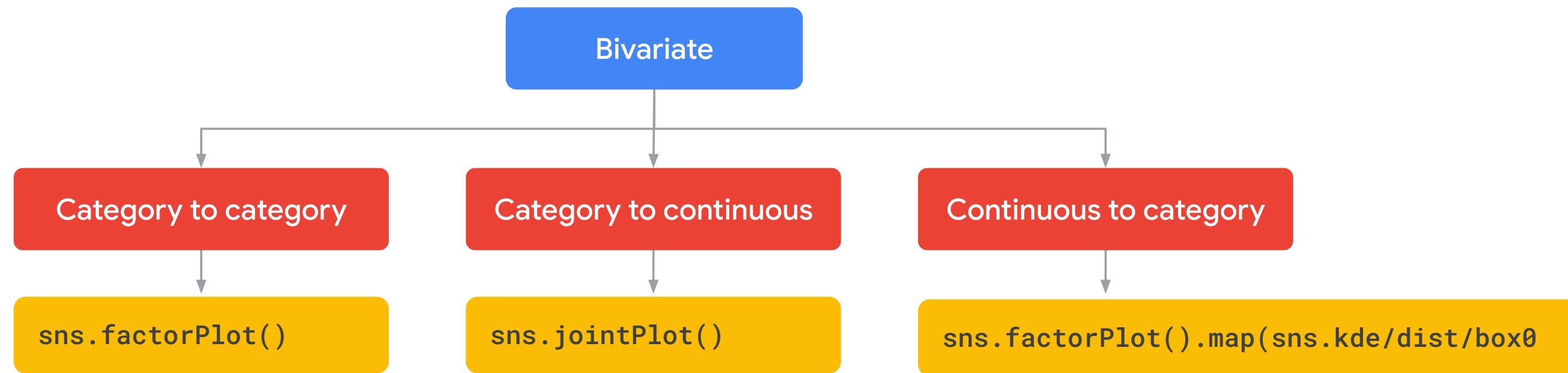


```
[17]: sns.countplot(x = `ocean_proximity`, data=df_USAhousing)  
[17]: <matplotlib.axes._subplots.AxesSubplot at 0x7f25ba4ef400>
```

Univariate data

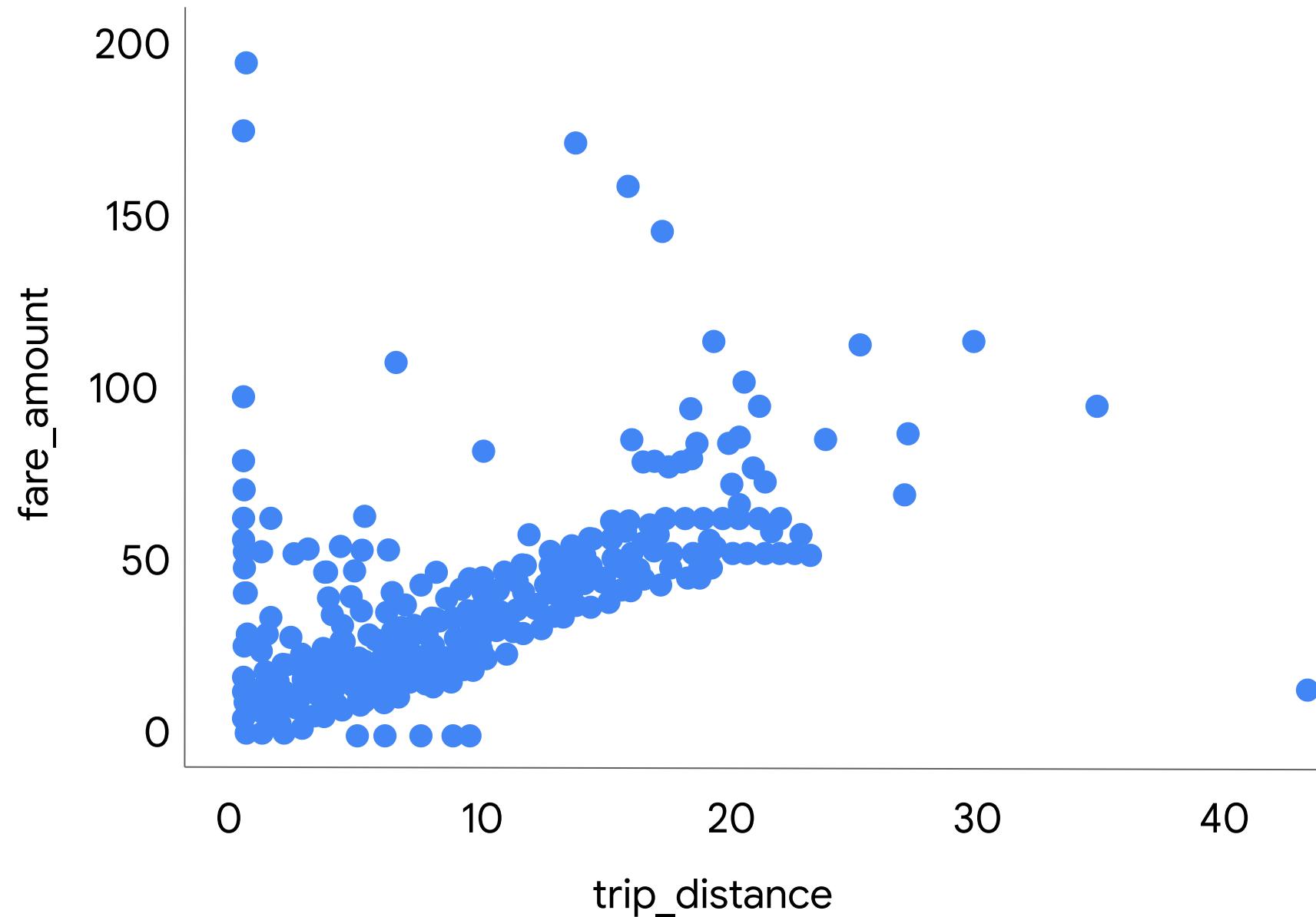


Exploratory data analysis

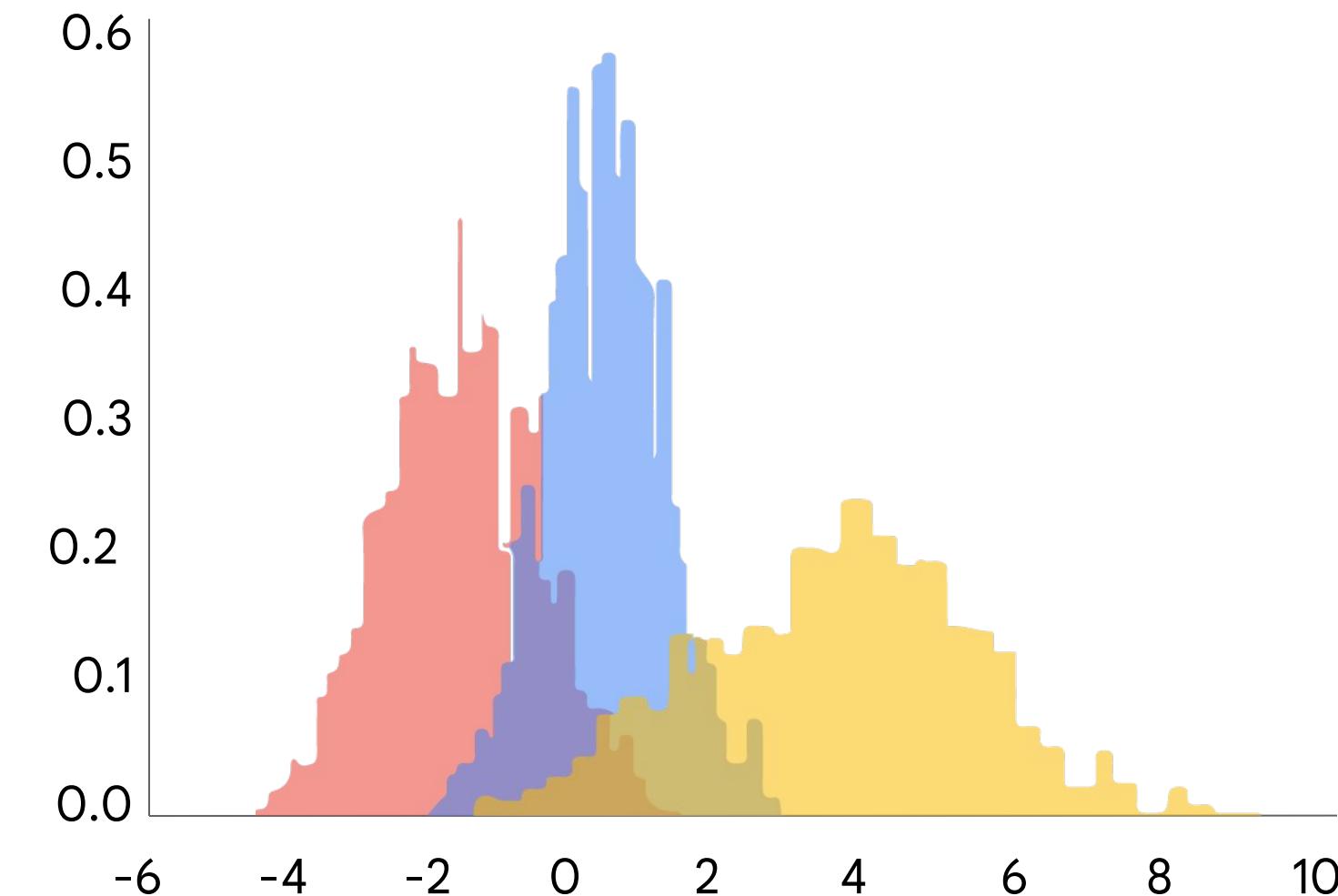


Bivariate data

```
# TODO 2  
ax = sns.regplot(  
    x="trip_distance", y="fare_amount",  
    fit_reg=False, ci=None, truncate=True, data=trips)  
ax.figure.set_size_inches(10, 8)
```



Data analysis and visualization



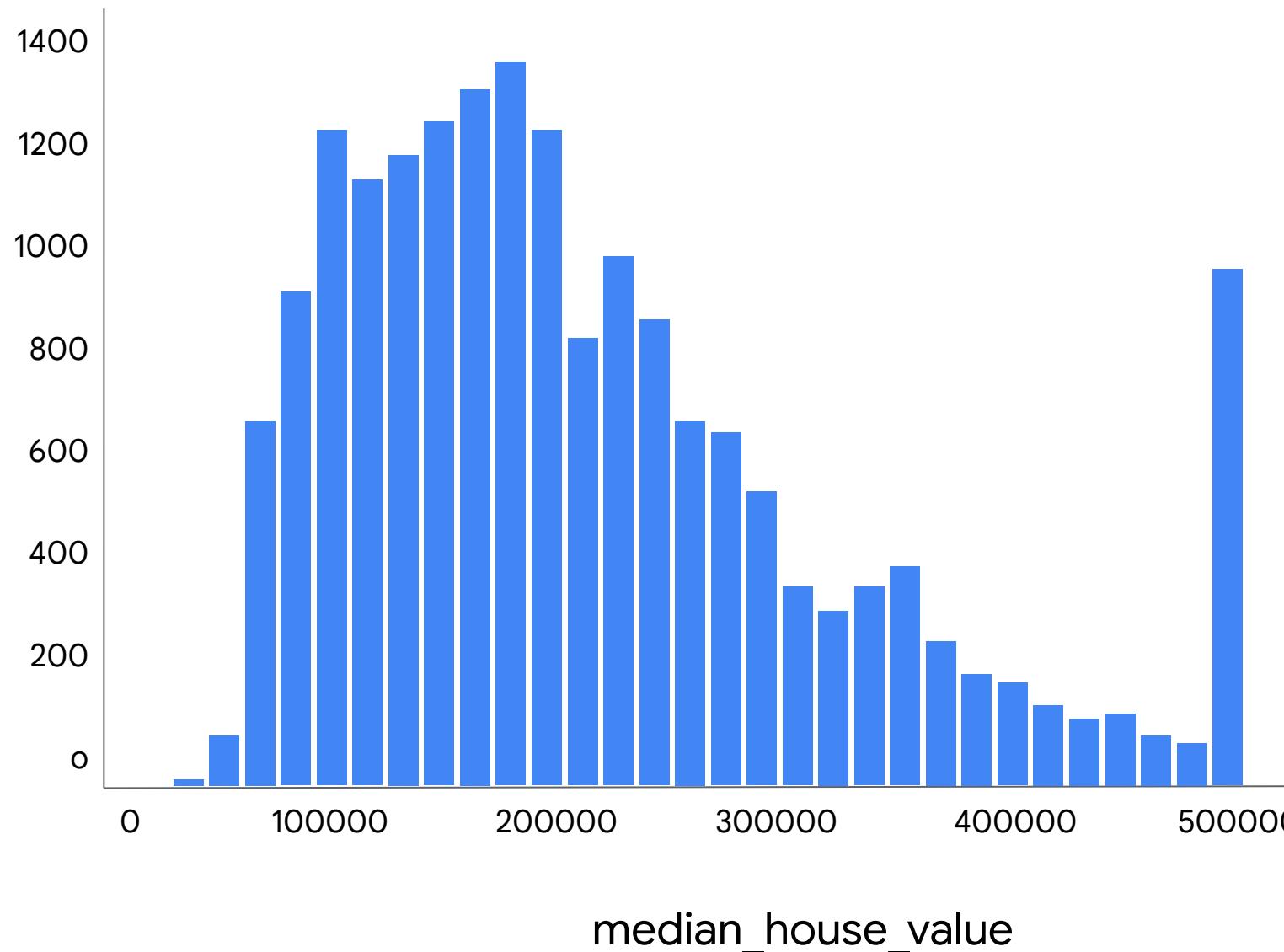
[14]:

```
sns.set_style('whitegrid')
df_USAhousing[‘median_house_value’].hist(bins=30)
plt.xlabel(‘median_house_value’)
```

[14]:

```
Text(0.5, 0, ‘median_house_value’)
```

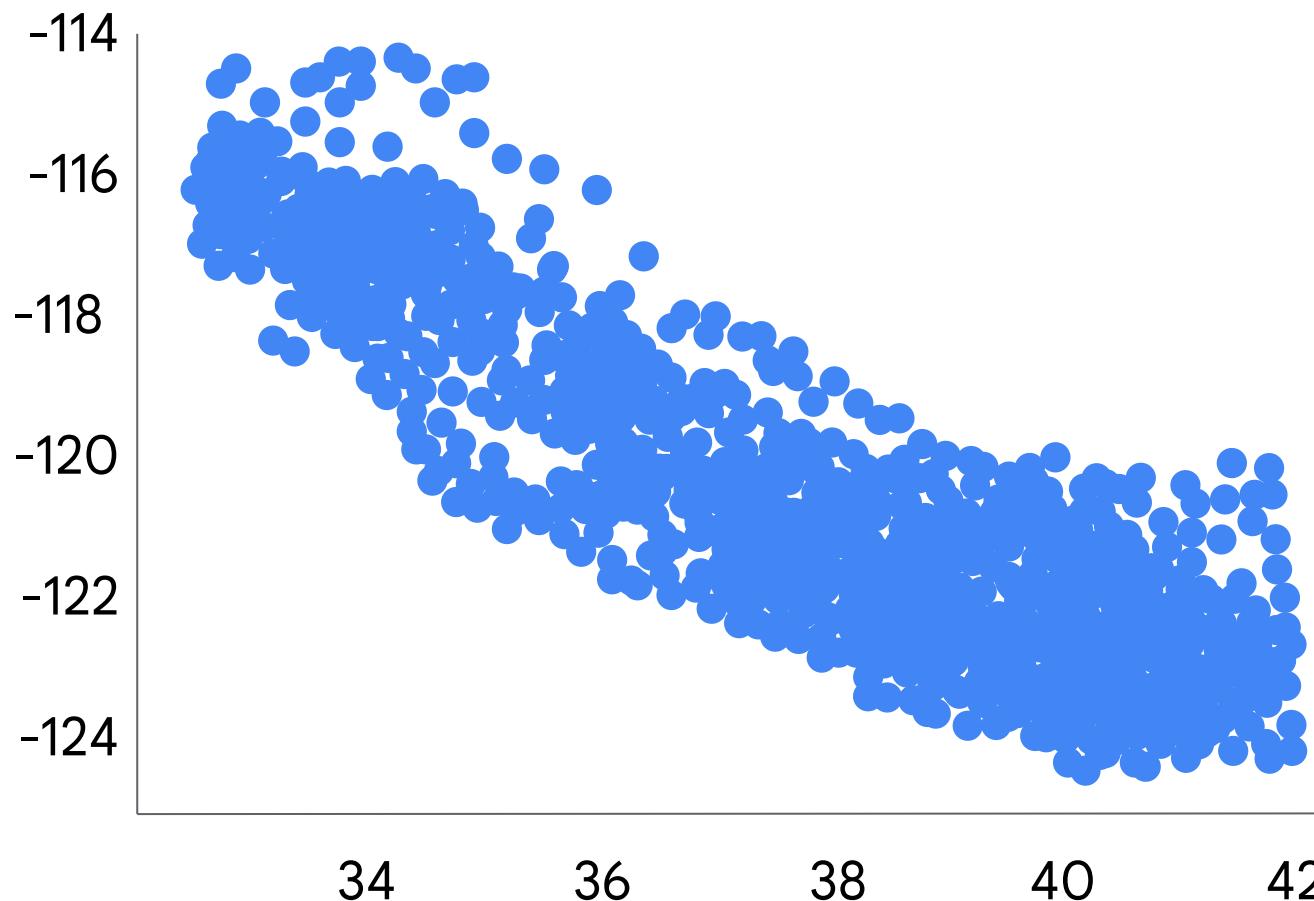
Histogram



Scatter plot

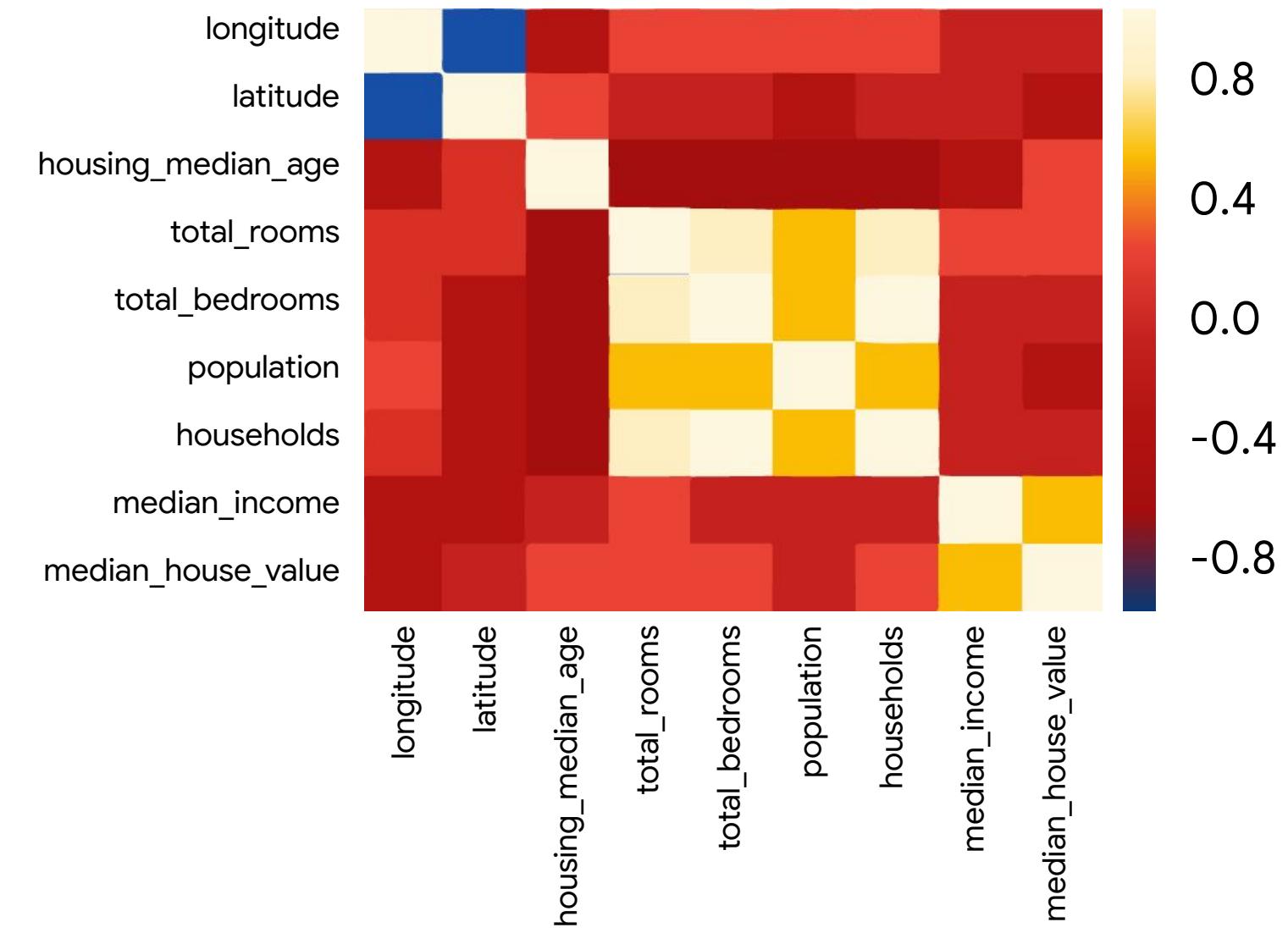
```
X = df_USAhousing[‘latitude’]  
Y = df_USAhousing[‘longitude’]
```

```
plt.scatter(x,y)  
plt.show()
```



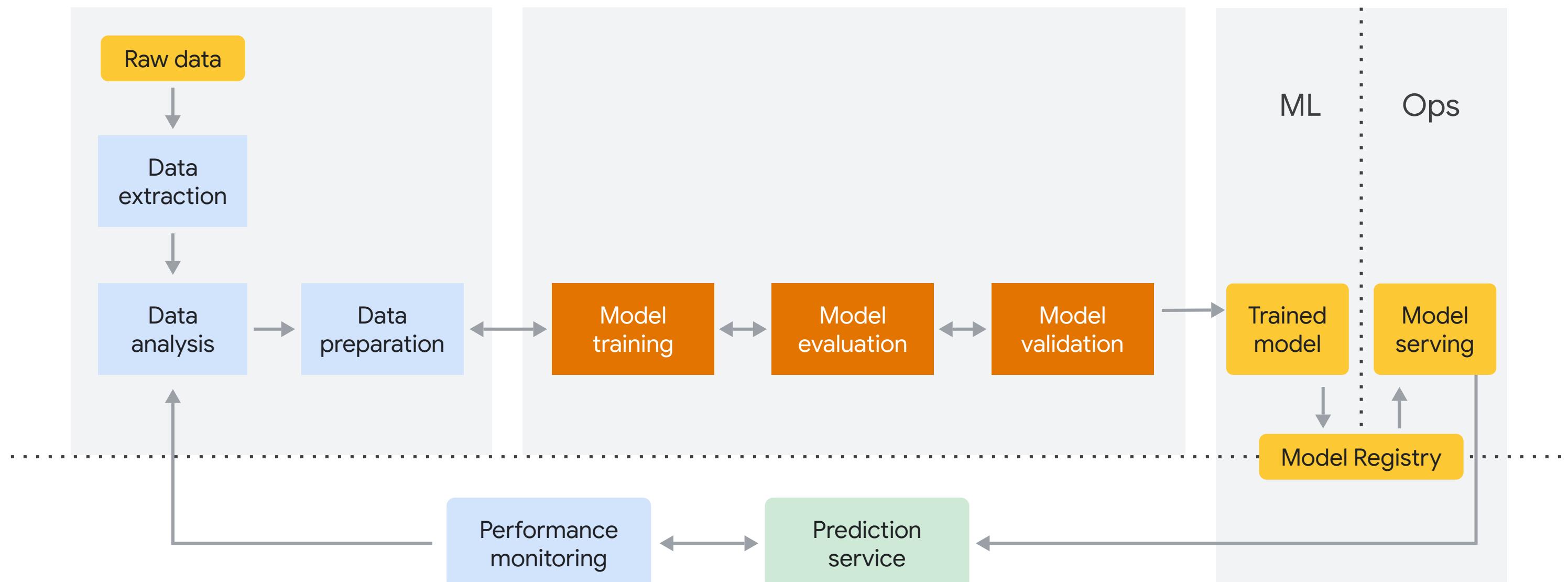
Correlations - multivariate

```
[12]: sns.heatmap(df_USAhousing.corr())  
[12]: <matplotlib.axes._subplots.AxesSubplot at 0x7f25ba7c1dd8>
```



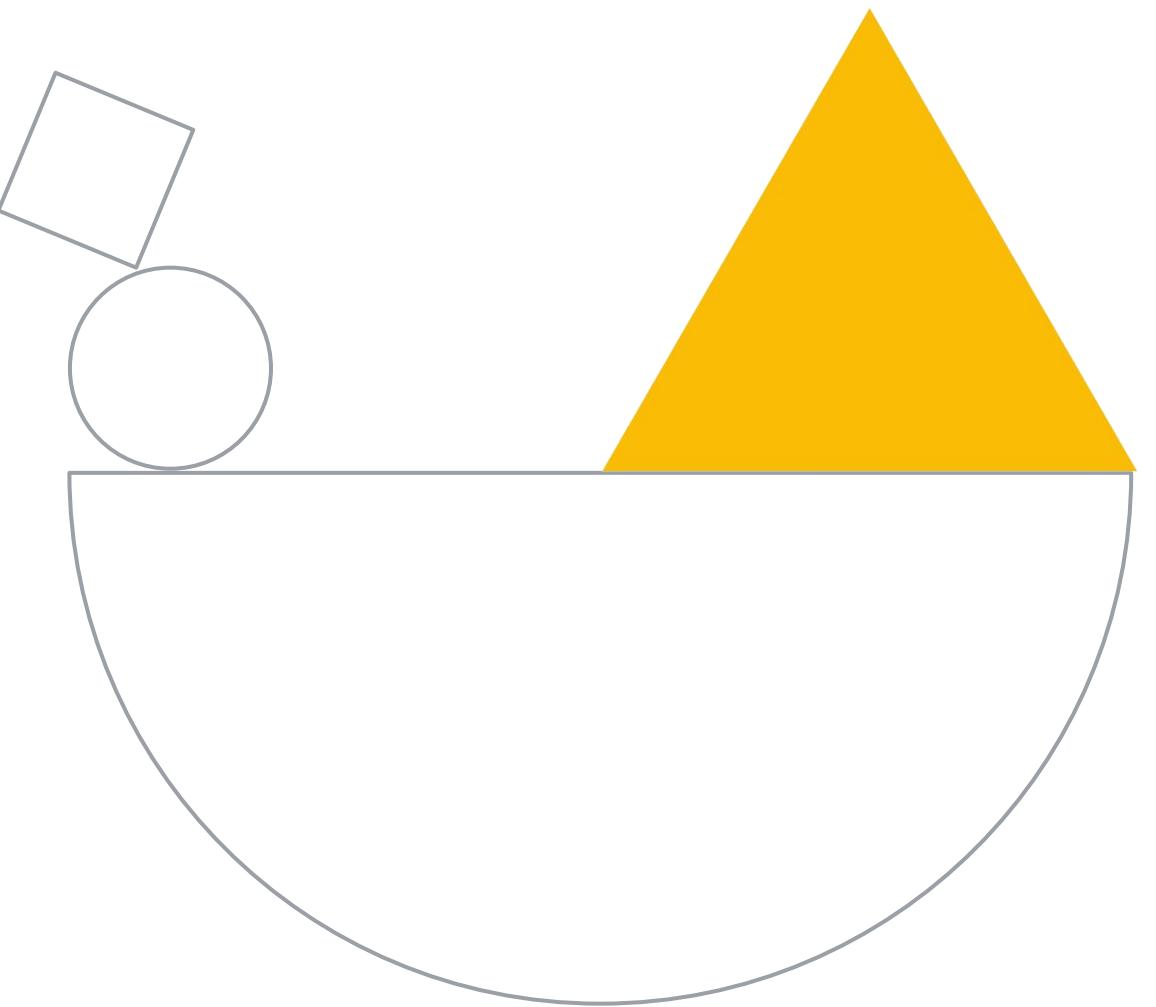
An ML pipeline recap

Staging/pre-production/production environments



Experimentation/development/test environments

Machine Learning in Practice



In this module, you learn to ...

01

Differentiate between supervised and unsupervised learning

02

Use Scikit Learn to perform Linear Regression

03

Understand Regression/Classification



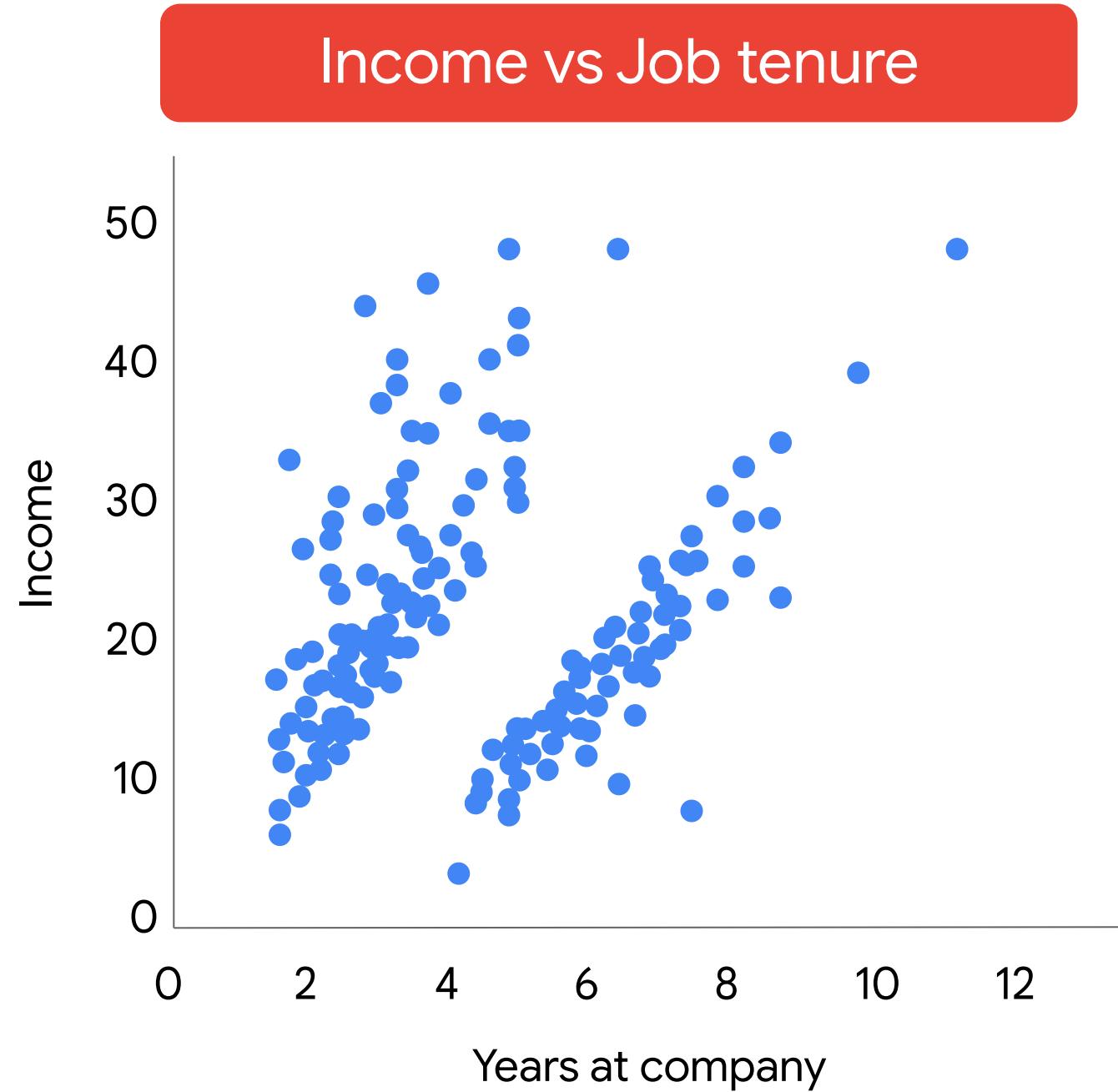
Supervised learning

**Unsupervised and
supervised learning
are the two types of
ML algorithms**

In unsupervised learning,
data is not labeled

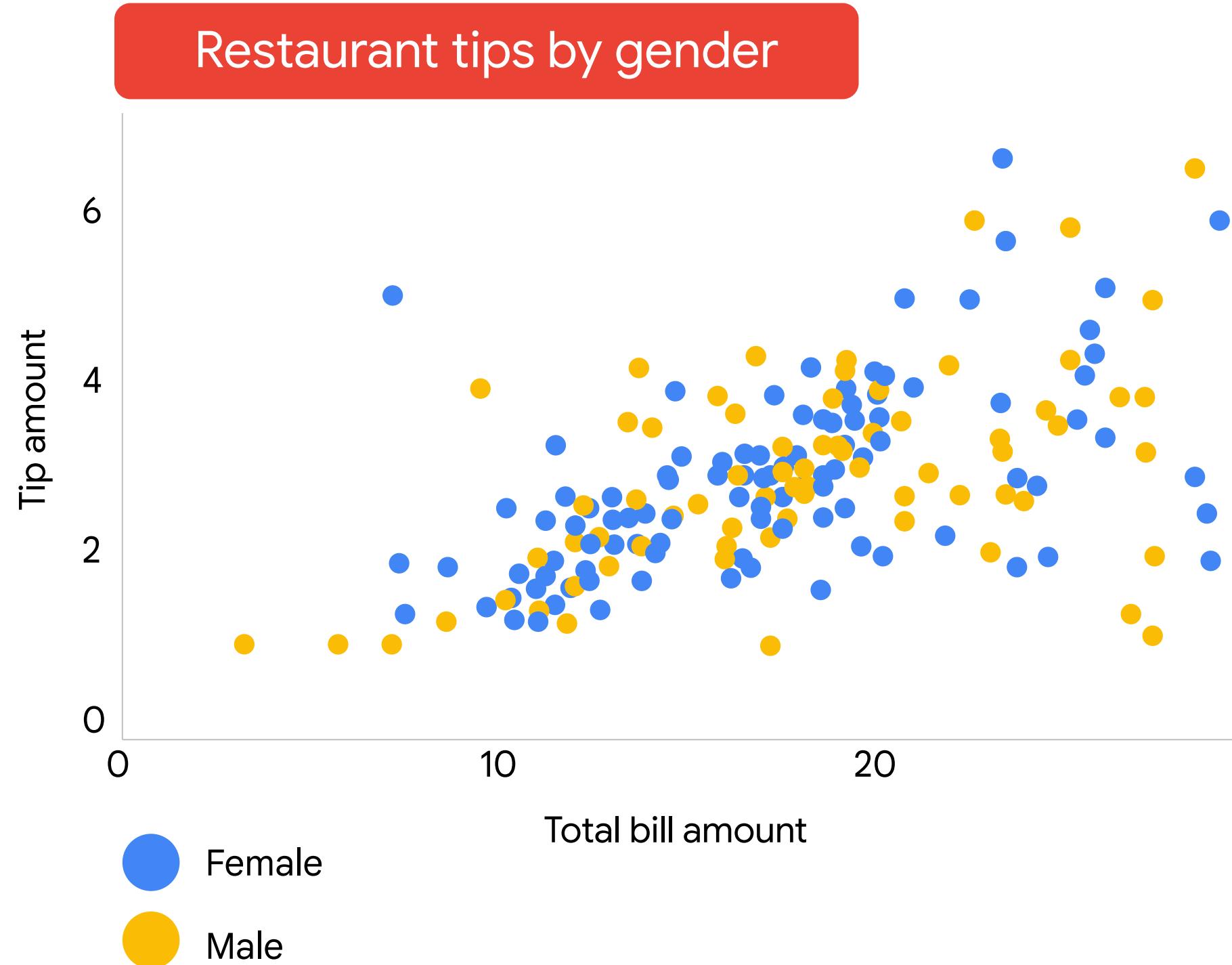
Example Model: Clustering

Is this employee on the “fast-track” or not?



**Supervised learning
implies the data is
already labeled**

In supervised learning we are
learning from past examples to
predict future values



Supervised ML model types:

Regression and classification

	total_bill	tip	sex	smoker	day	time
1	16.99	1.01	Female	No	Sun	Dinner
2	10.34	1.66	Male	No	Sun	Dinner
3	21.01	3.5	Male	No	Sun	Dinner
4	23.68	3.31	Male	No	Sun	Dinner
5	24.59	3.61	Female	No	Sun	Dinner
6	25.29	4.71	Male	No	Sun	Dinner
7	8.77	2	Male	No	Sun	Dinner
8	26.88	3.12	Male	No	Sun	Dinner

Supervised ML model types:

Regression and classification

1	total_bill	tip	sex	smoker	day	time
2	16.99	1.01	Female	No	Sun	Dinner
3	10.34	1.66	Male	No	Sun	Dinner
4	21.01	3.5	Male	No	Sun	Dinner
5	23.68	3.31	Male	No	Sun	Dinner
6	24.59	3.61	Female	No	Sun	Dinner
7	25.29	4.71	Male	No	Sun	Dinner
8	8.77	2	Male	No	Sun	Dinner
9	26.88	3.12	Male	No	Sun	Dinner

Option 1
Regression Model
Predict the tip amount

Supervised ML model types:

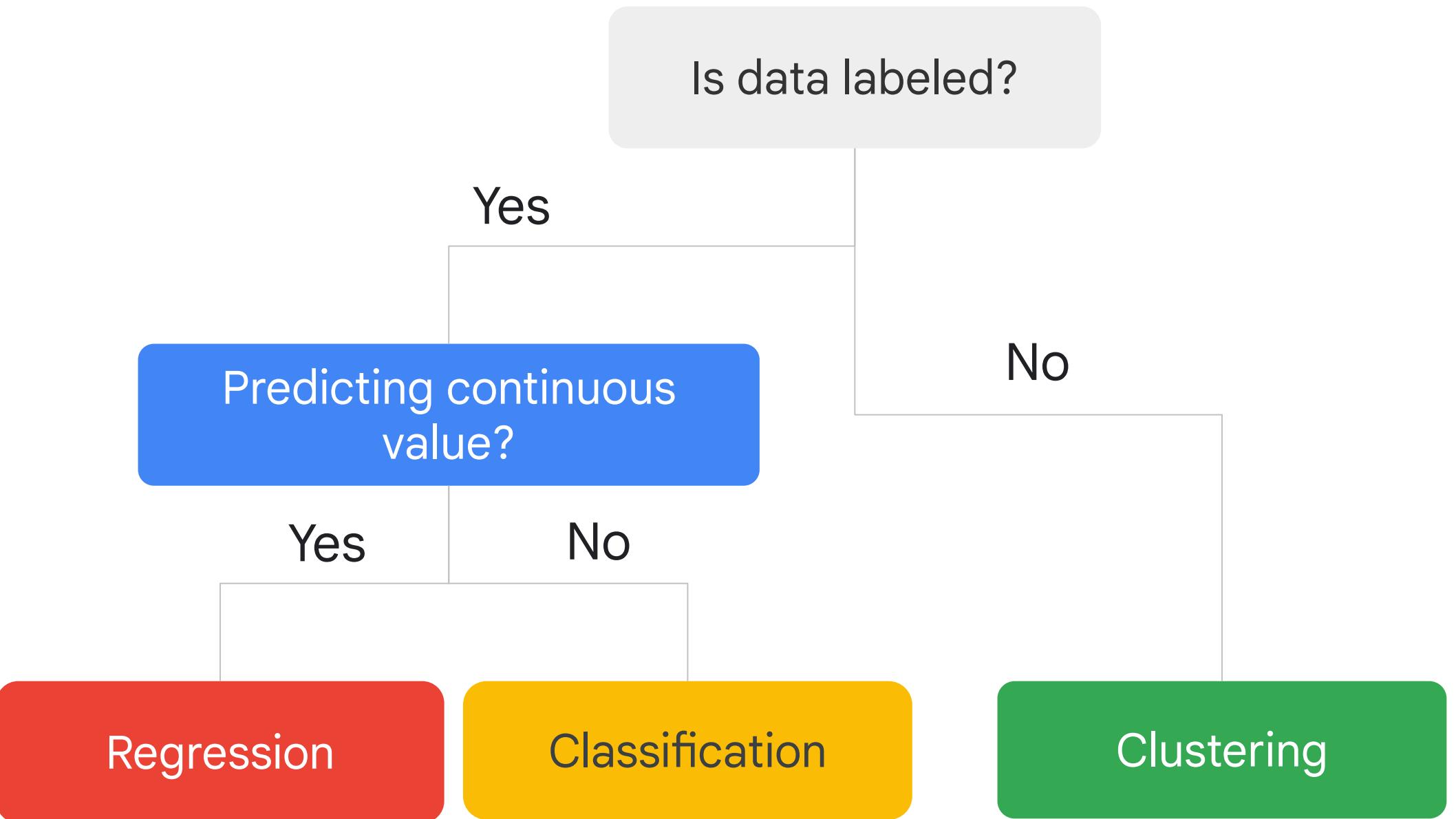
Regression and classification

1	total_bill	tip	sex	smoker	day	time
2	16.99	1.01	Female	No	Sun	Dinner
3	10.34	1.66	Male	No	Sun	Dinner
4	21.01	3.5	Male	No	Sun	Dinner
5	23.68	3.31	Male	No	Sun	Dinner
6	24.59	3.61	Female	No	Sun	Dinner
7	25.29	4.71	Male	No	Sun	Dinner
8	8.77	2	Male	No	Sun	Dinner
9	26.88	3.12	Male	No	Sun	Dinner

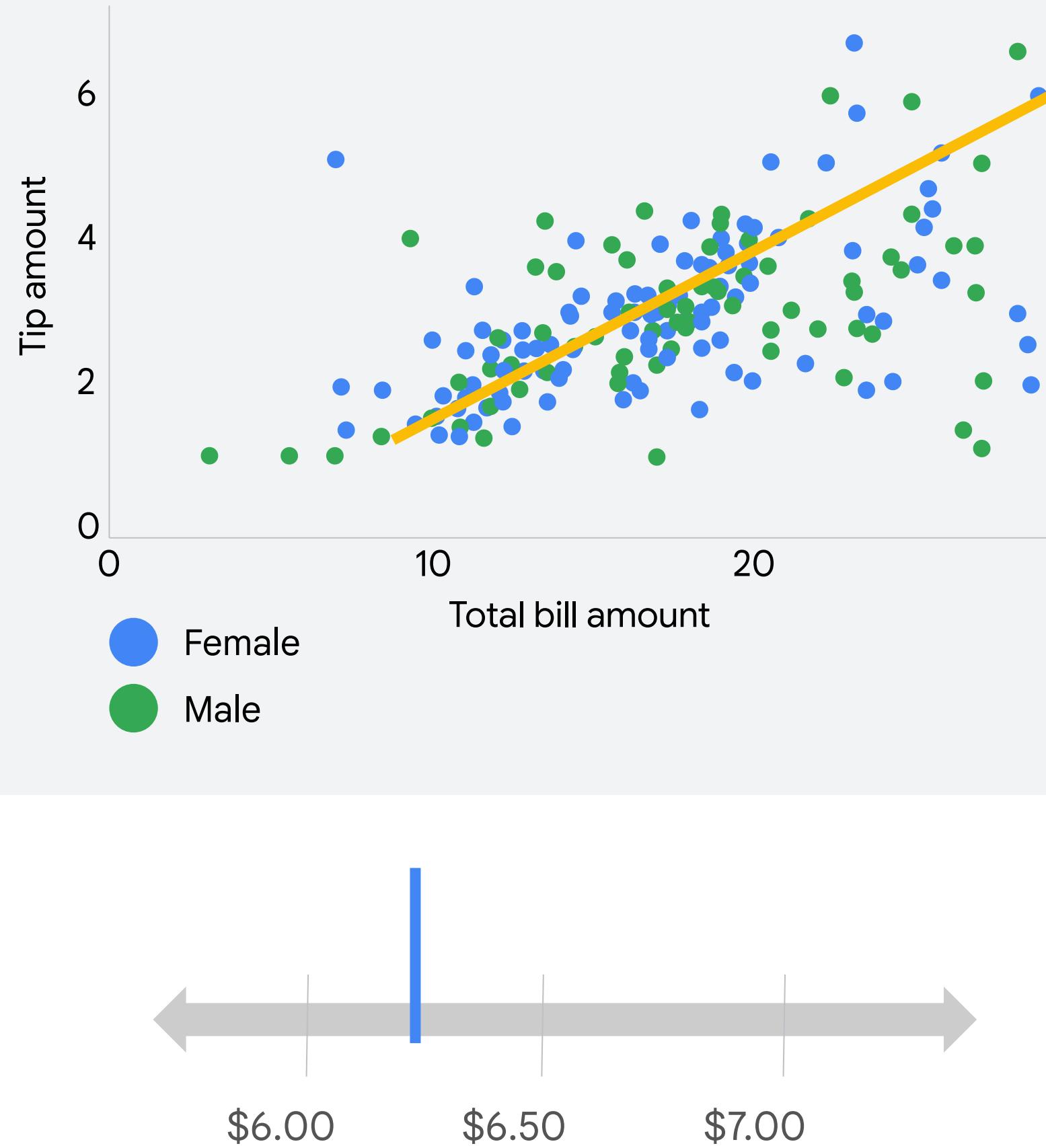
**Option 1
Regression Model**
Predict the tip amount

**Option 2
Classification Model**
Predict the sex of the customer

The type of ML
problem depends
on whether or not
you have **labeled**
data and what
you are interested
in **predicting**

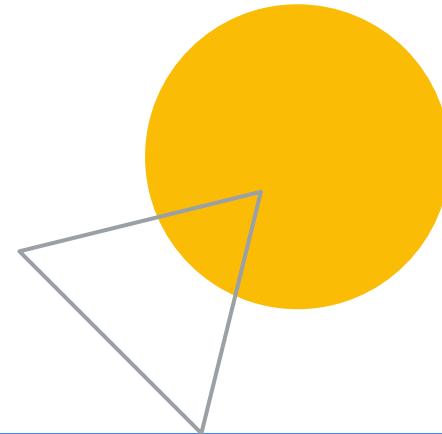


Restaurant tips by gender



Use regression for
predicting continuous
label values

Regression model (linear) predicts
tip amount of \$6.25.



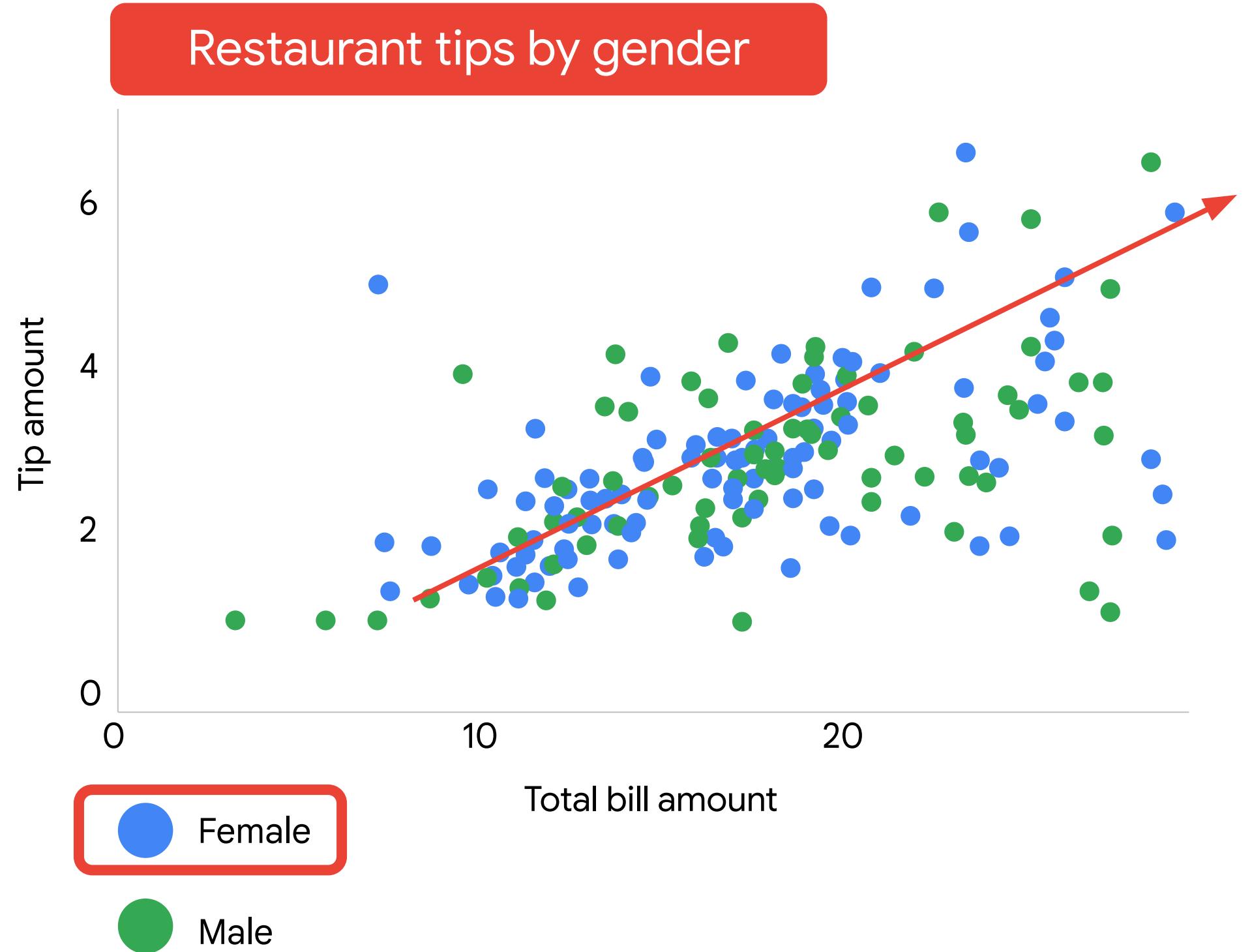
Use classification for predicting categorical label values

Classification model (binary) predicts category of female.



Use classification for predicting categorical label values

Classification model (binary) predicts category of female.



Use classification for predicting categorical label values

Classification model (binary) predicts category of female.

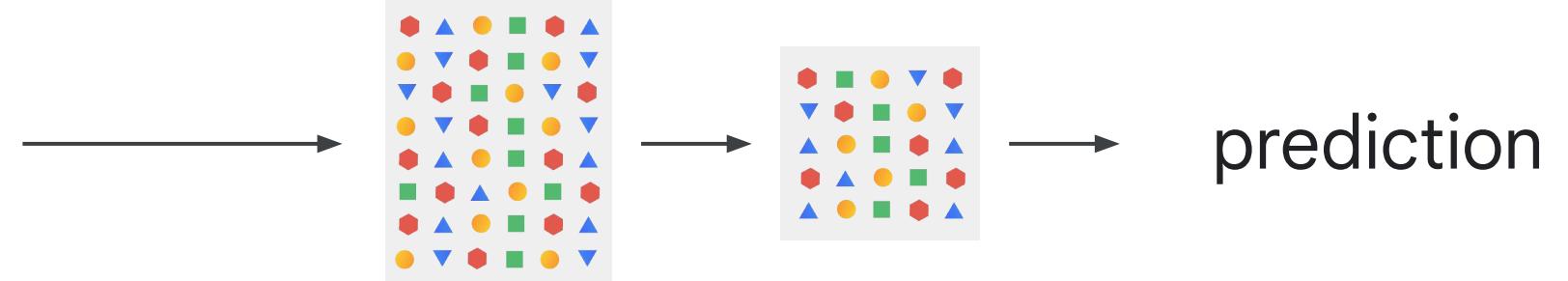


A data warehouse can be a source of structured data training examples for your ML model

```
SELECT  
    gestation_weeks,  
    mother_age,  
    cigarette_use,  
    alcohol_use,  
    weight_gain_pounds  
FROM  
    `bigquery-public-data.samples.natality`  
WHERE cigarette_use is not null AND alcohol
```

Data on births is sourced from our BigQuery data warehouse using SQL.

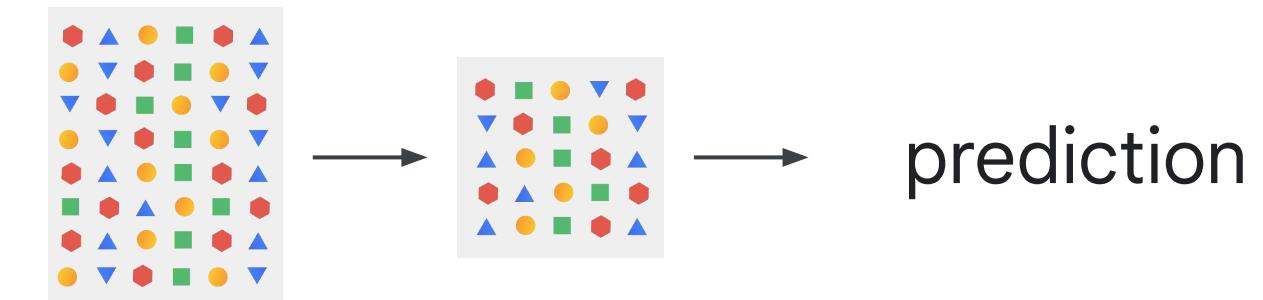
weight	year	mother_age	gestation_weeks	cigarette_use	alcohol_use
7.86	2003	25	39	false	false
7.5	2003	21	39	false	false
8.06	2004	29	40	false	false
7.56	2004	38	37	false	false
7.06	2003	22	38	false	false



Since baby weight is a continuous value, use regression to predict

weight	year	mother_age	gestation_weeks	cigarette_use	alcohol_use
7.86	2003	25	39	false	false
7.5	2003	21	39	false	false
8.06	2004	29	40	false	false
7.56	2004	38	37	false	false
7.06	2003	22	38	false	false

Weight is stored as a floating point number, representing a continuous (real) value.

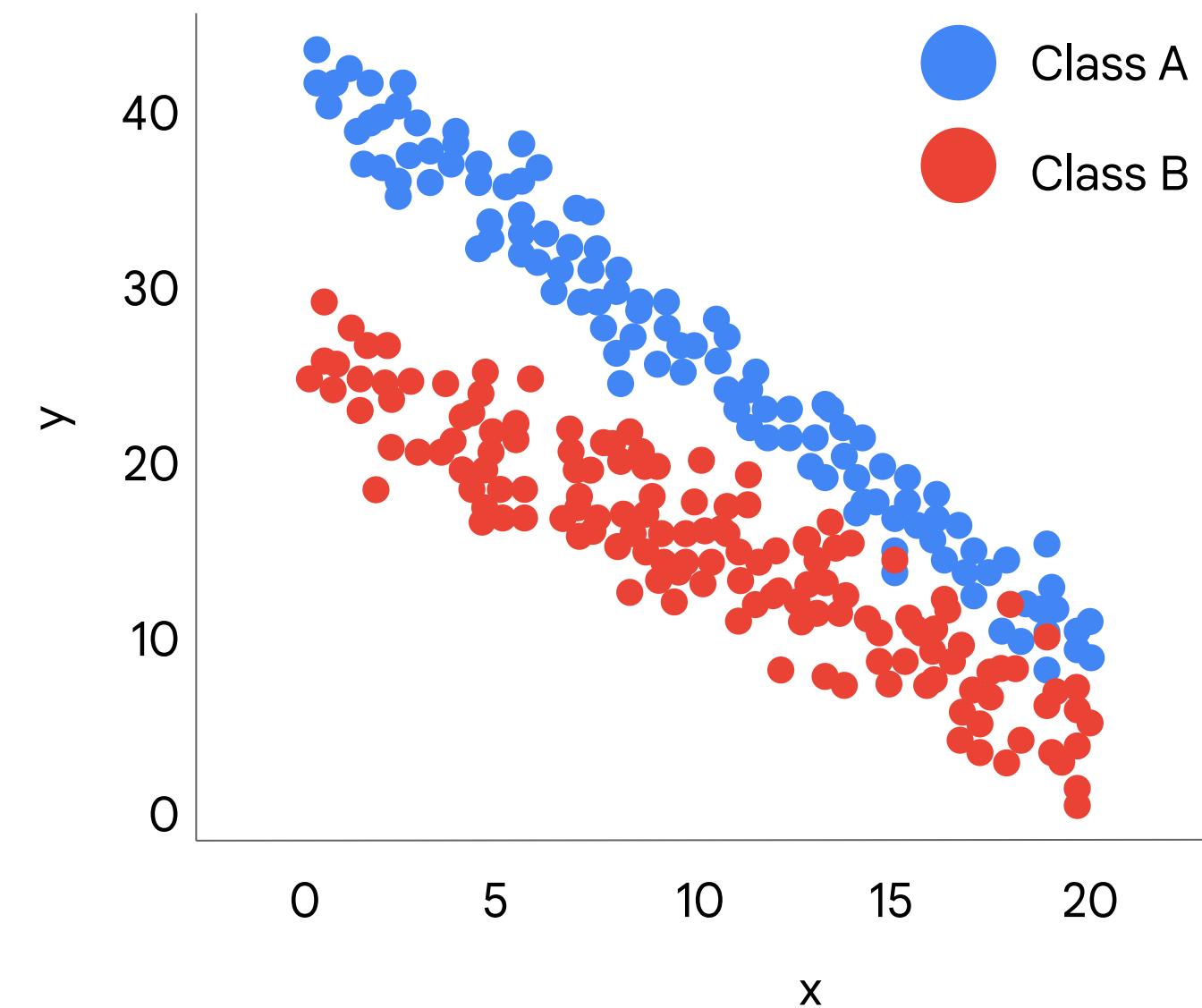


Regression DNN model

Quiz: Regression/Classification

Is this dataset a good candidate for linear regression and/or linear classification?

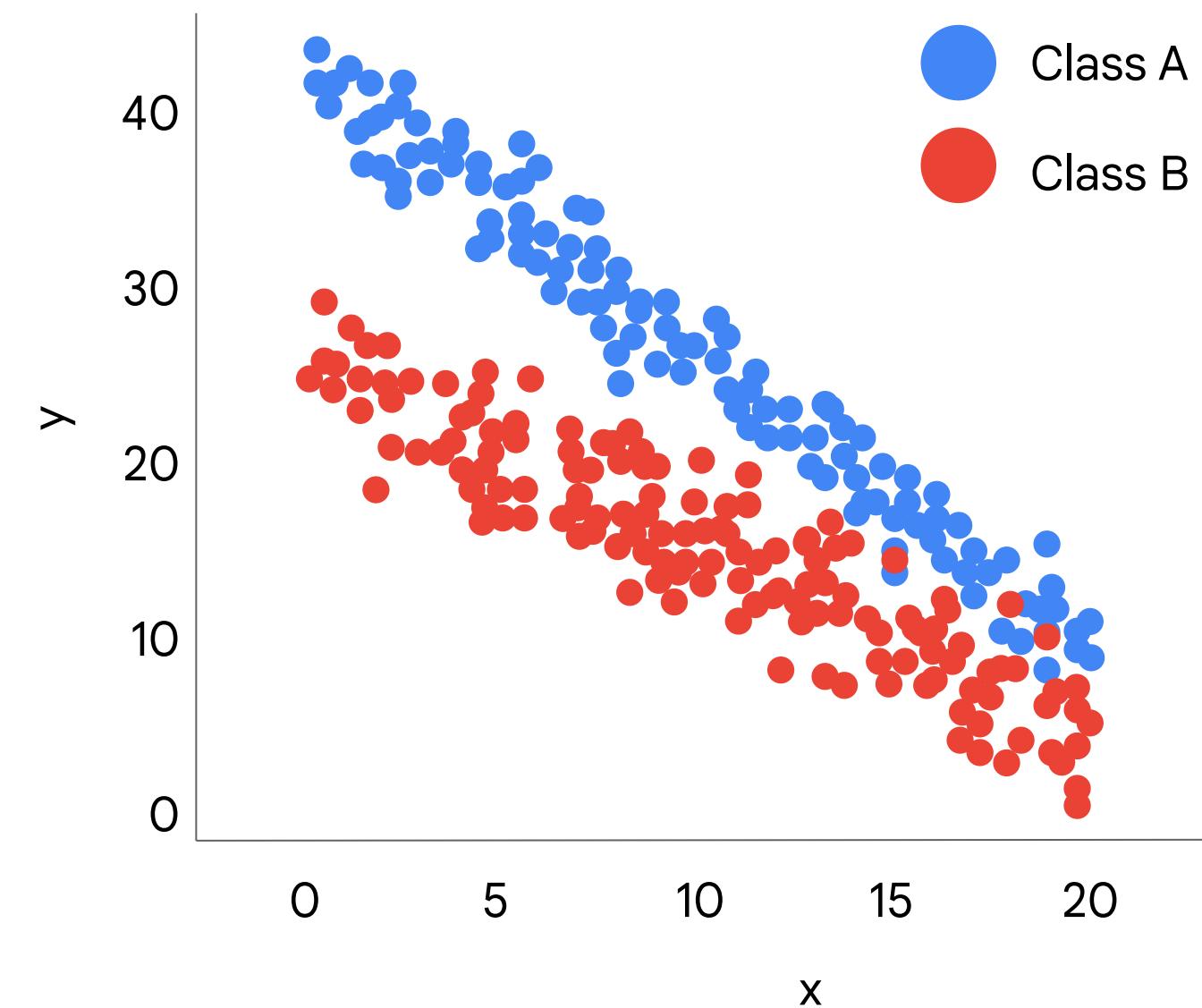
- A. Linear classification
- B. Both
- C. None of the above



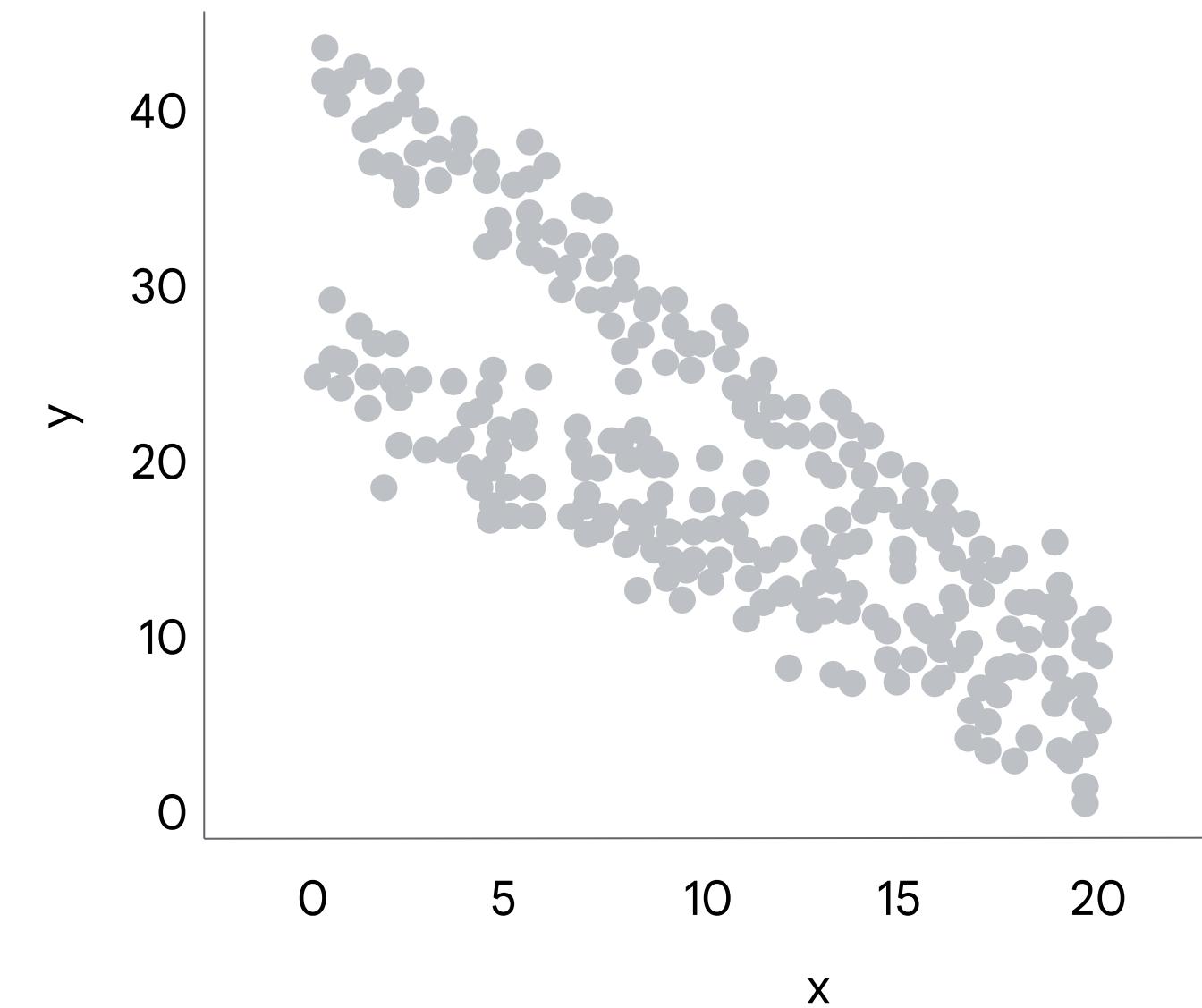
Quiz: Regression/Classification

Is this dataset a good candidate for linear regression and/or linear classification?

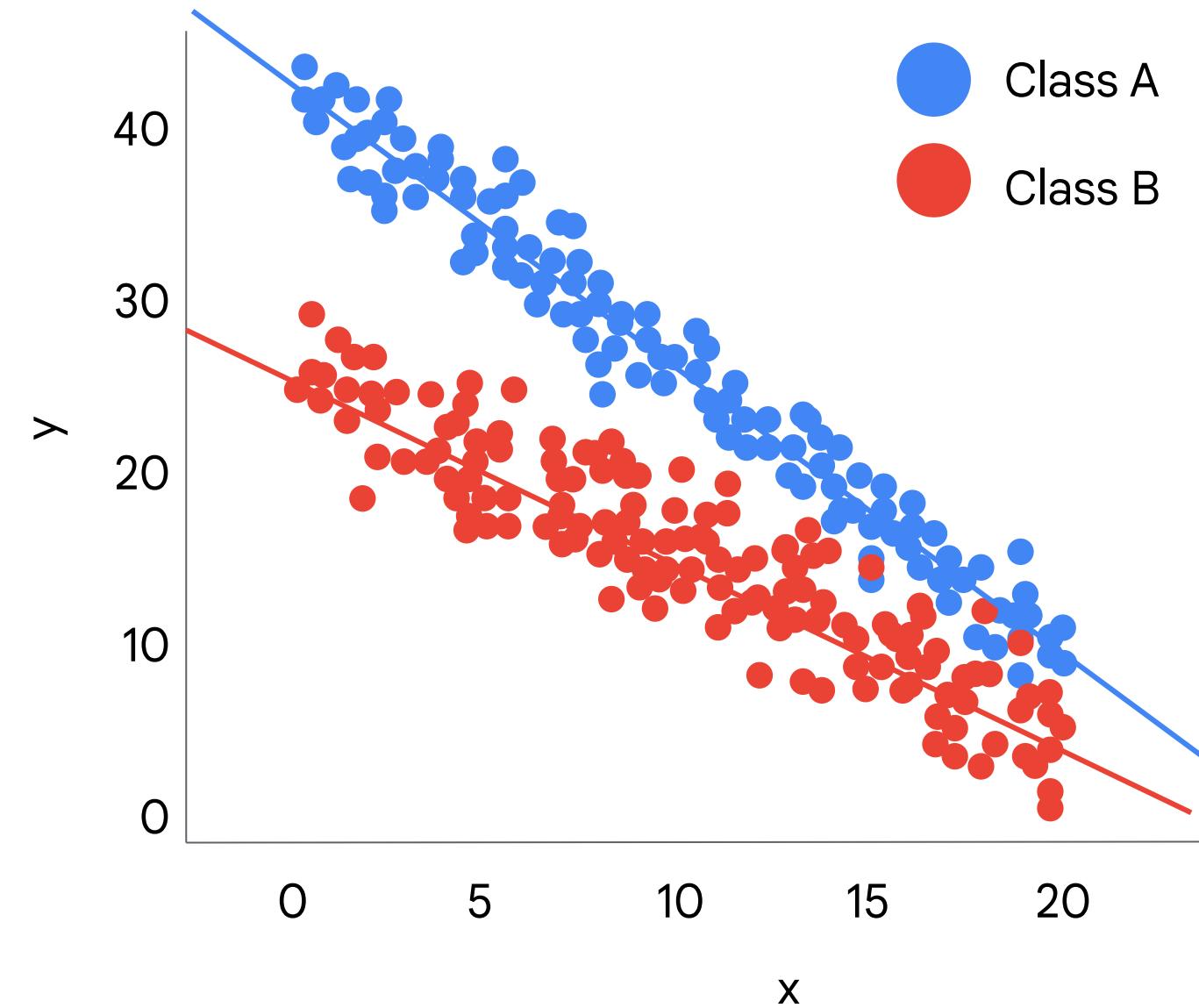
- A. Linear classification
- B. Both
- C. None of the above



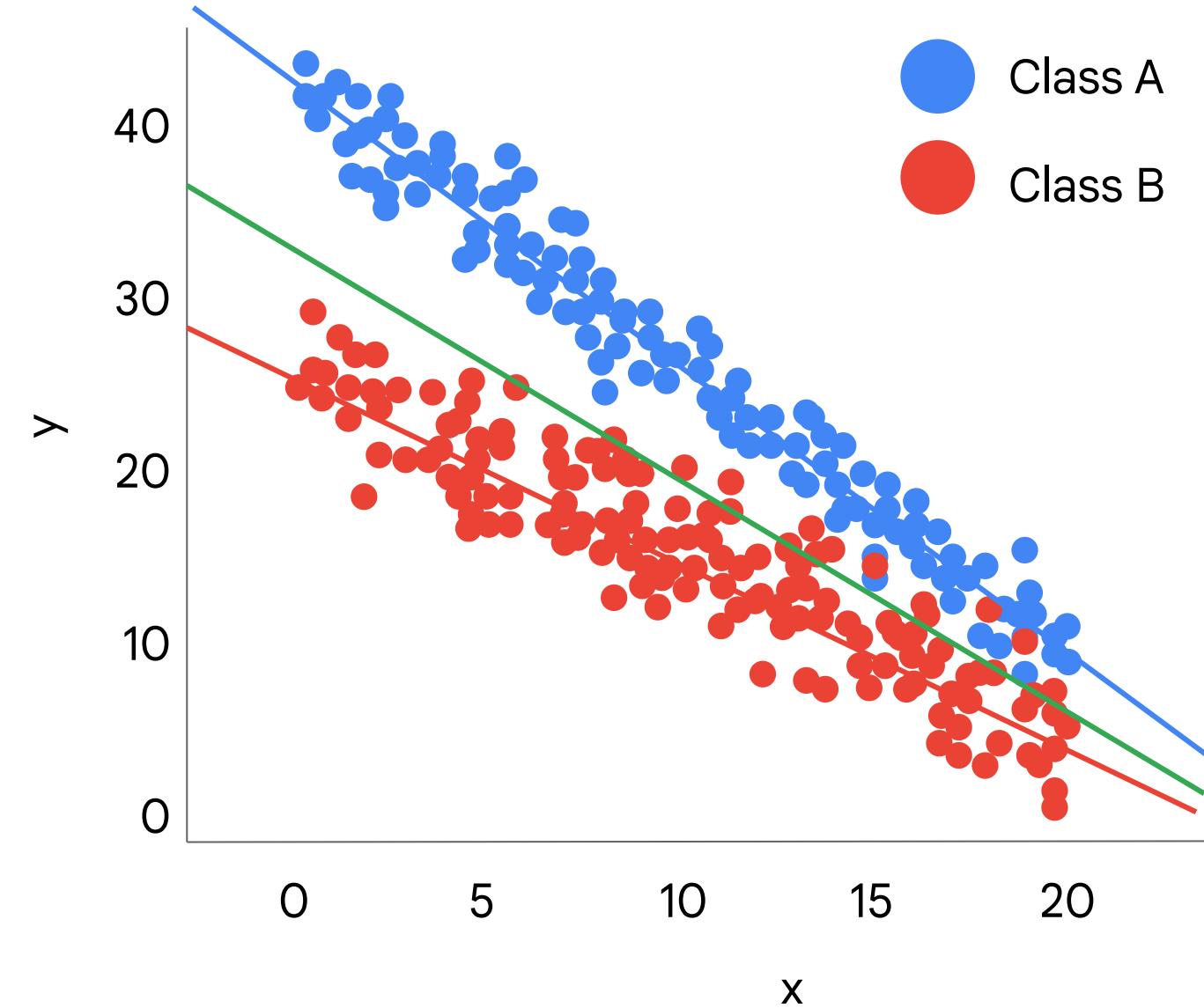
**Is this dataset a good
candidate for linear
regression and/or linear
classification?**



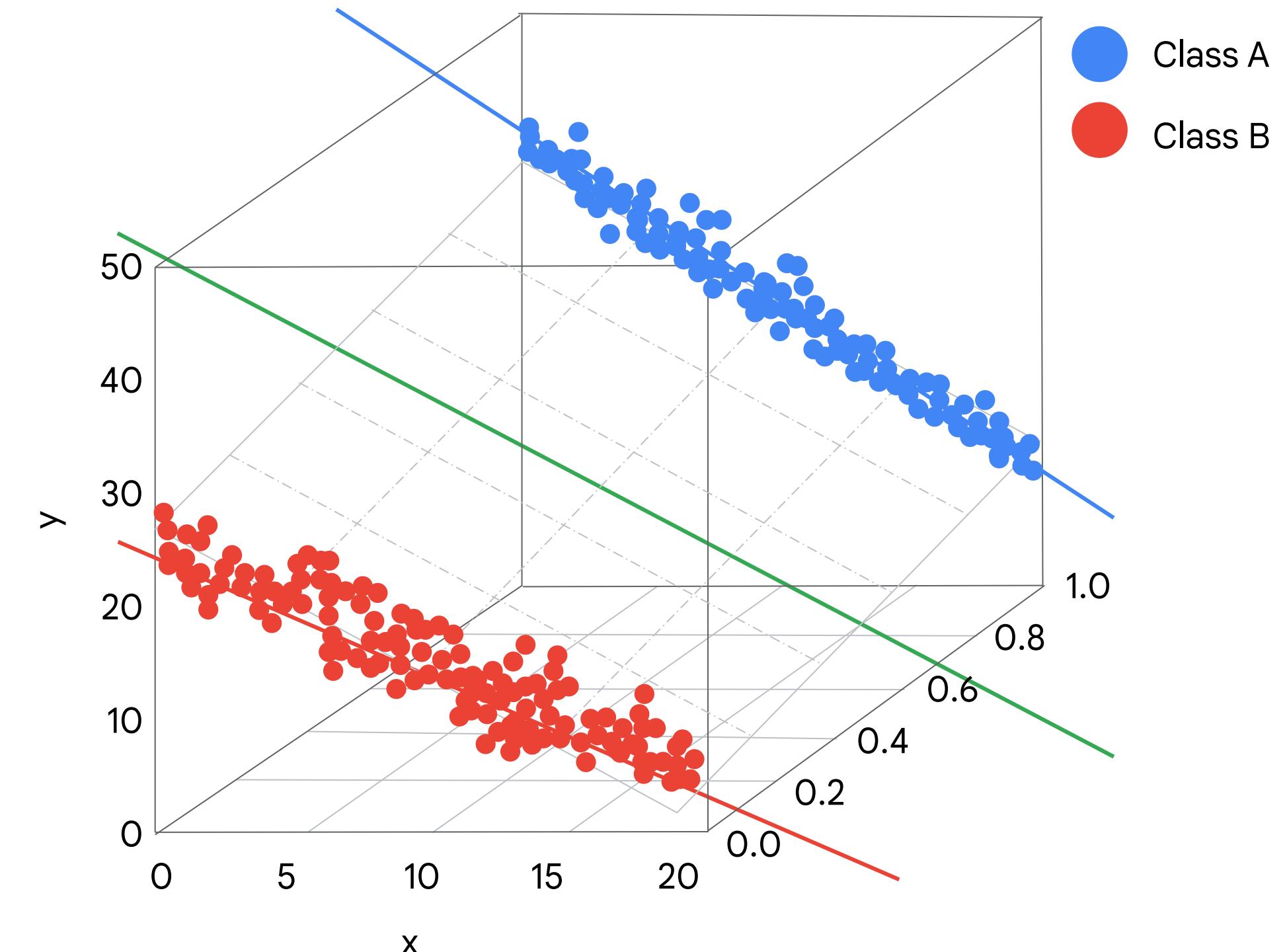
Is this dataset a good candidate for linear regression and/or linear classification?



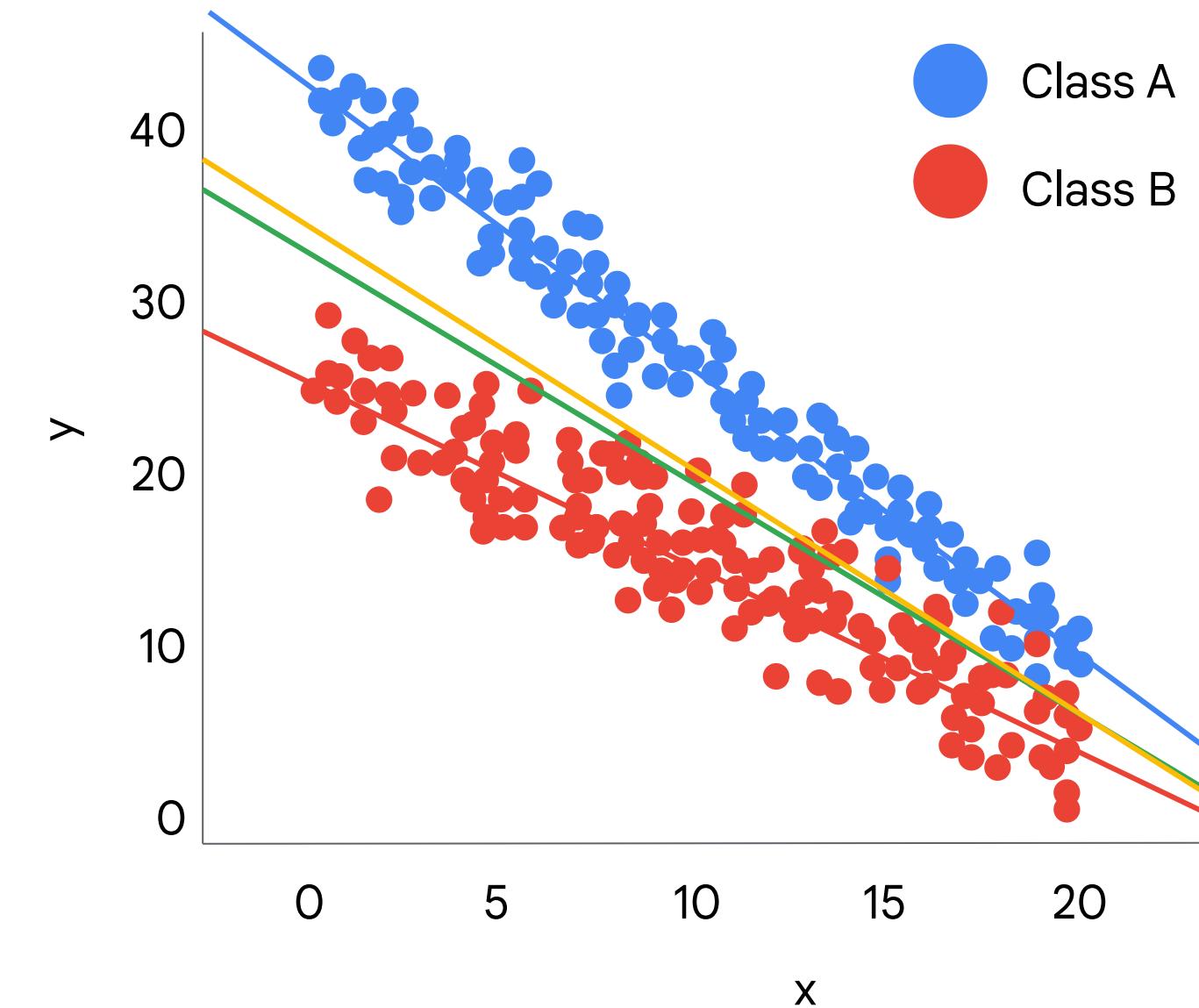
Is this dataset a good
candidate for linear
regression and/or linear
classification?



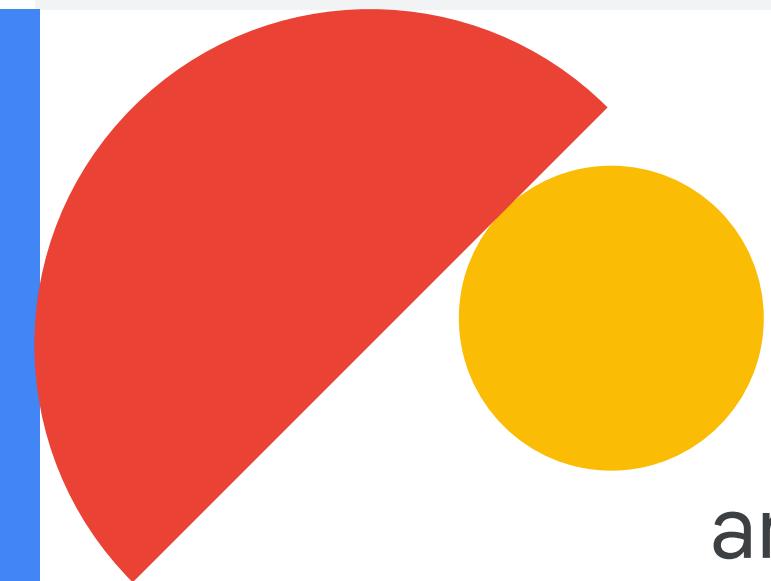
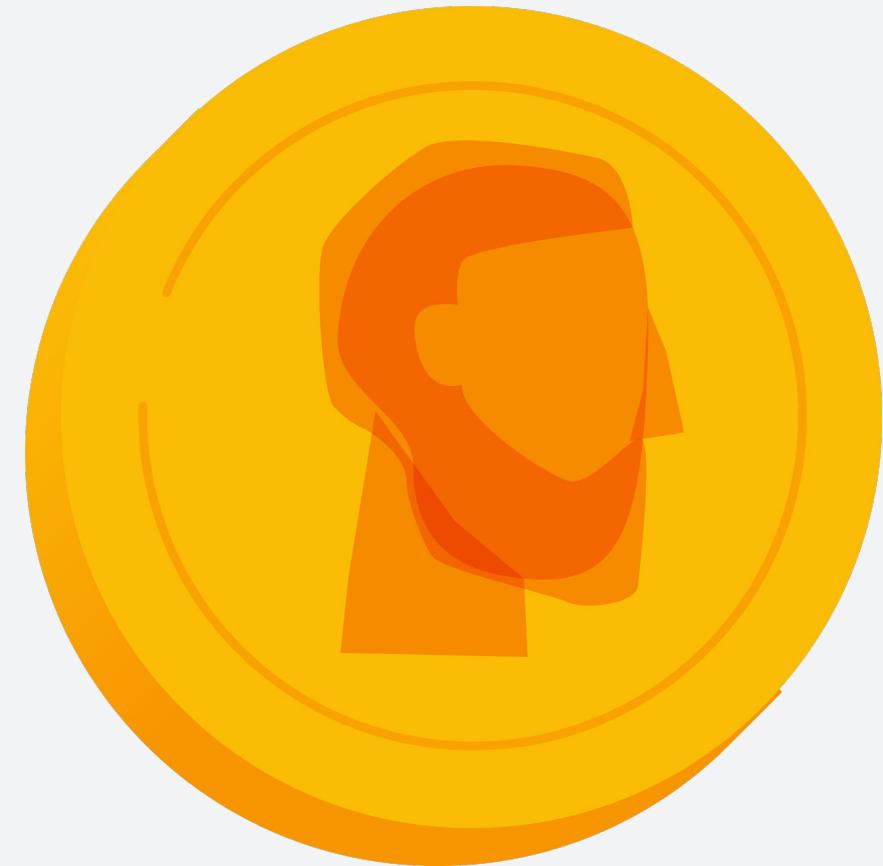
Is this dataset a good candidate for linear regression and/or linear classification?



Is this dataset a good
candidate for linear
regression and/or linear
classification?

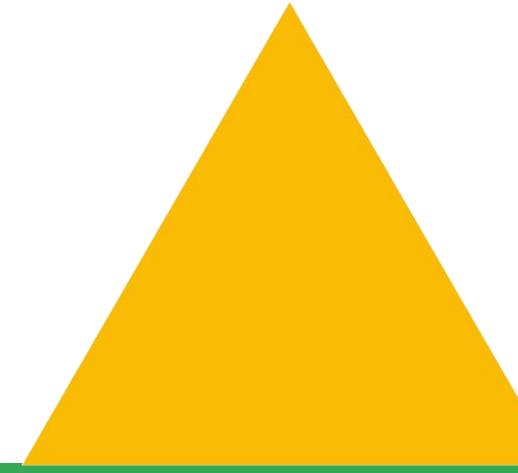


Suppose you use linear regression to predict coin flips



You might use features like angle of bend, coin mass, etc.

Suppose you use linear
regression to predict
coin flips



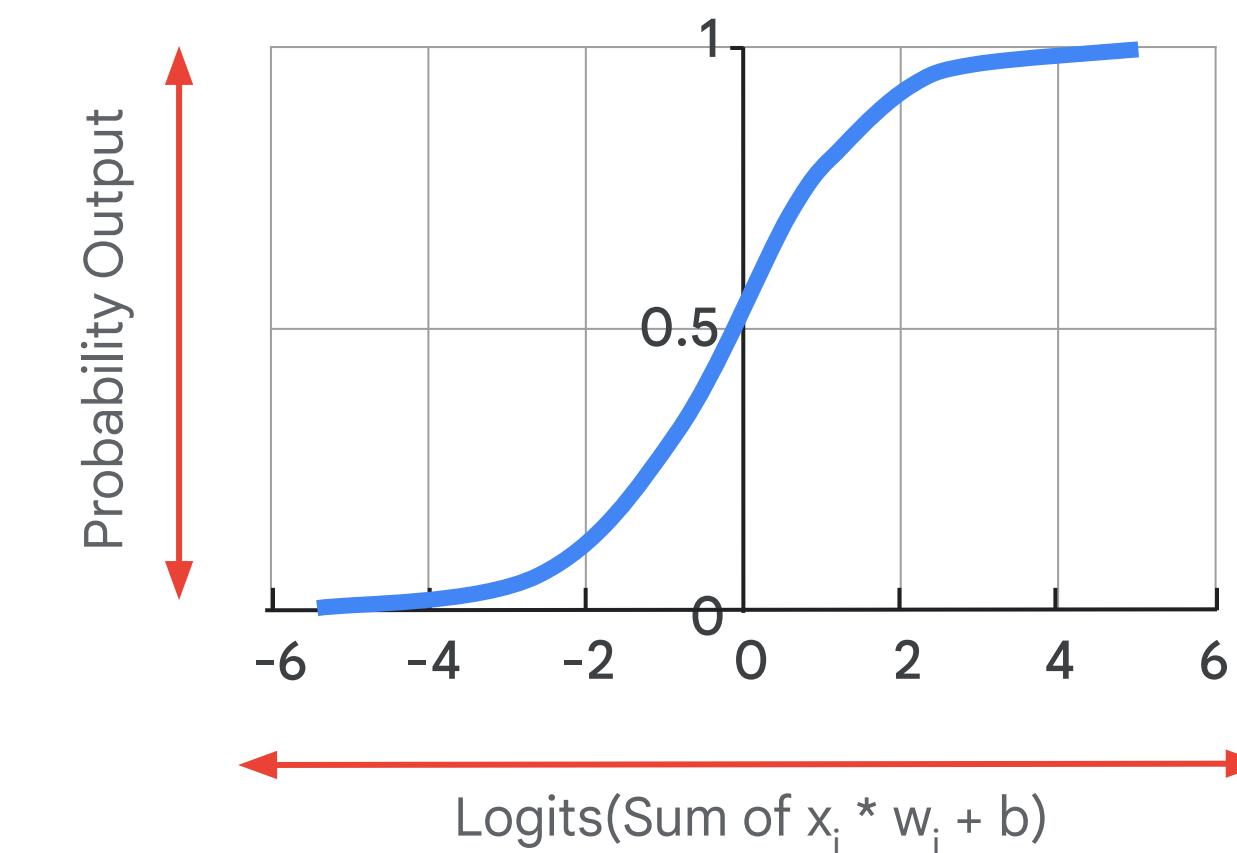
What could go wrong?

Logistic Regression: transform linear regression by a sigmoid activation function

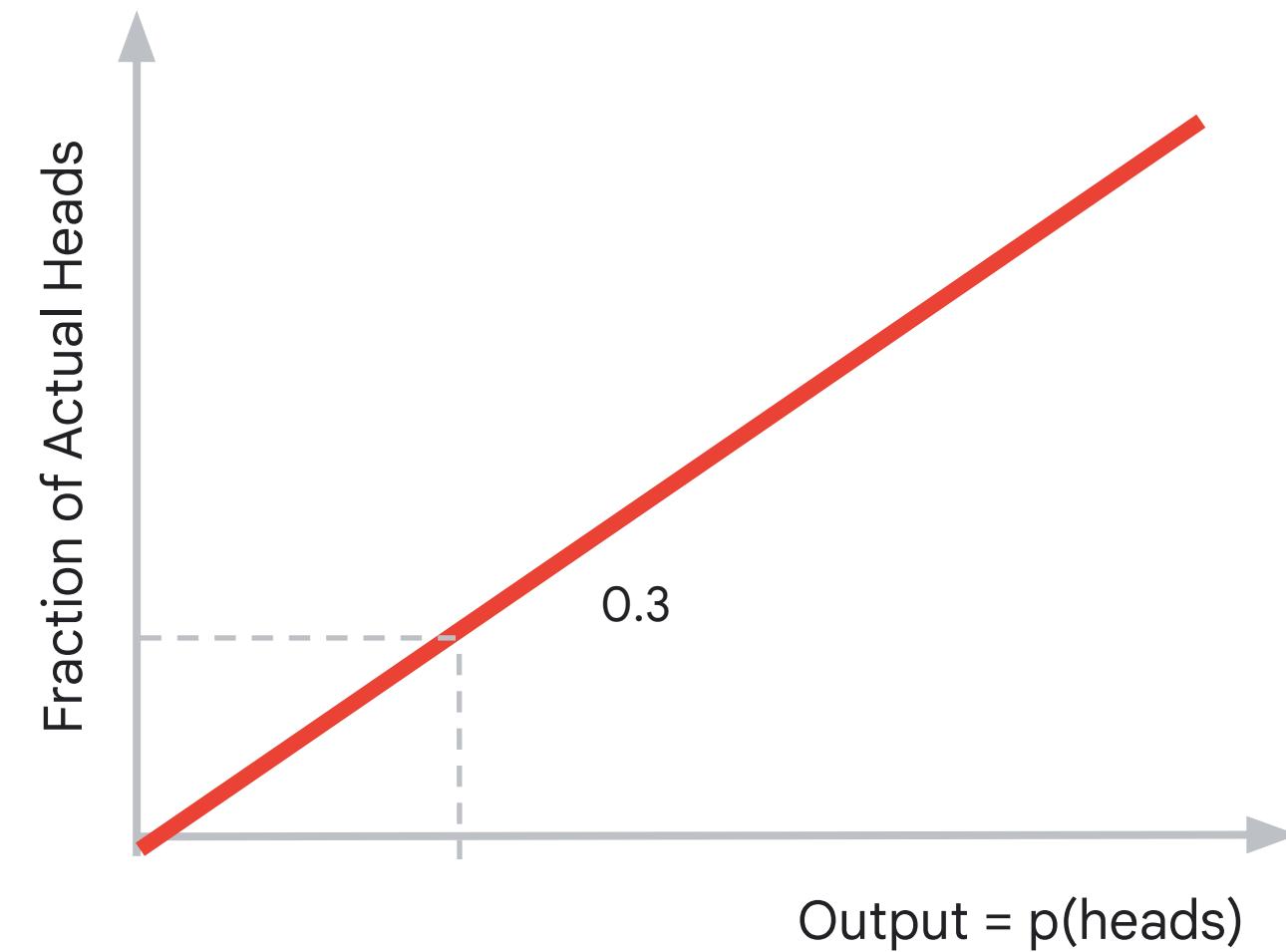
linear model

$$\hat{y} = \frac{1}{1 + e^{-(w^T x + b)}}$$

squish through a sigmoid



The output of Logistic Regression is a calibrated probability estimate



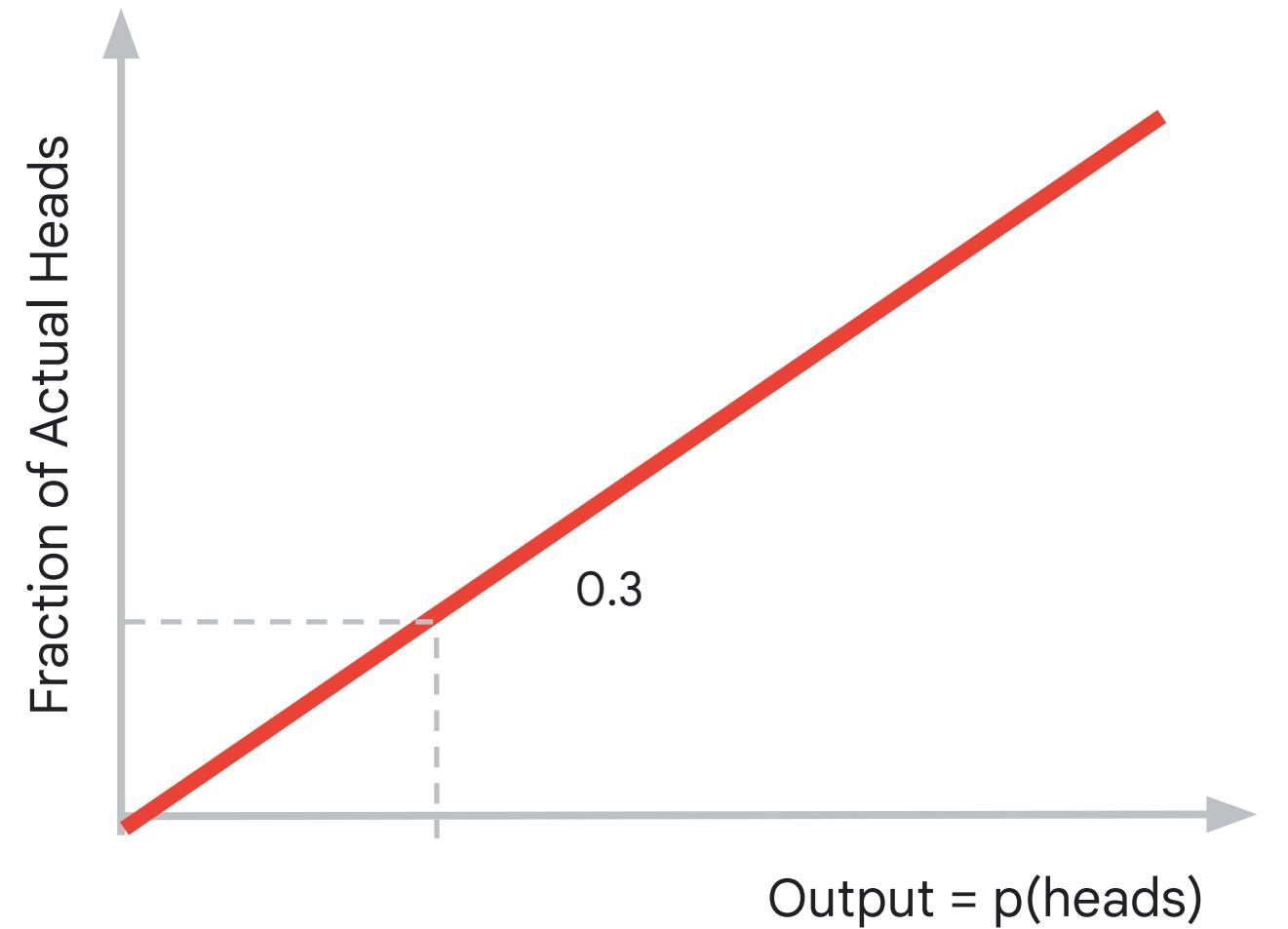
The output of Logistic Regression is a calibrated probability estimate

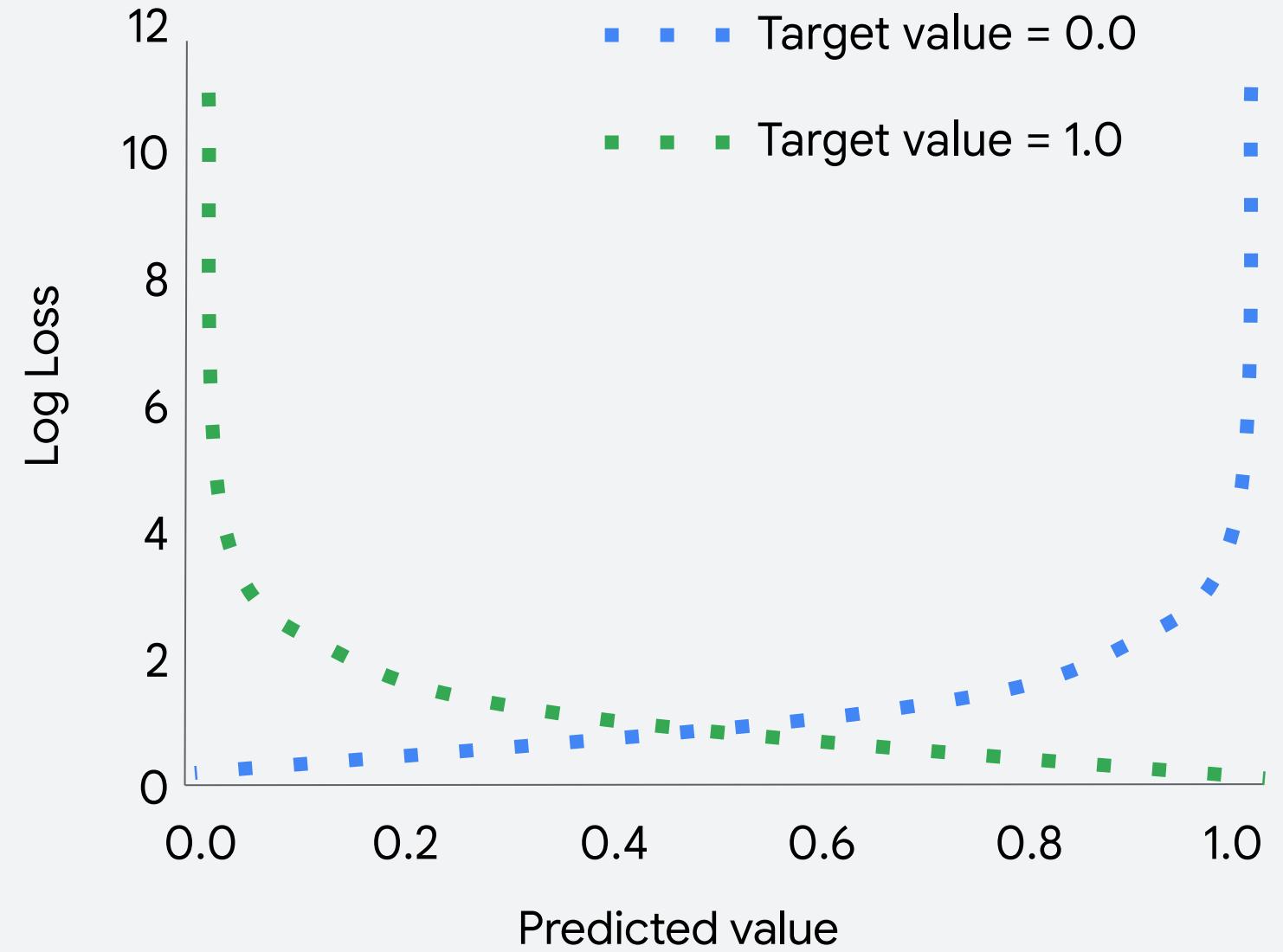
Useful because we can cast binary classification problems into probabilistic problems:

Will customer buy item?

becomes

Predict the probability that customer buys item





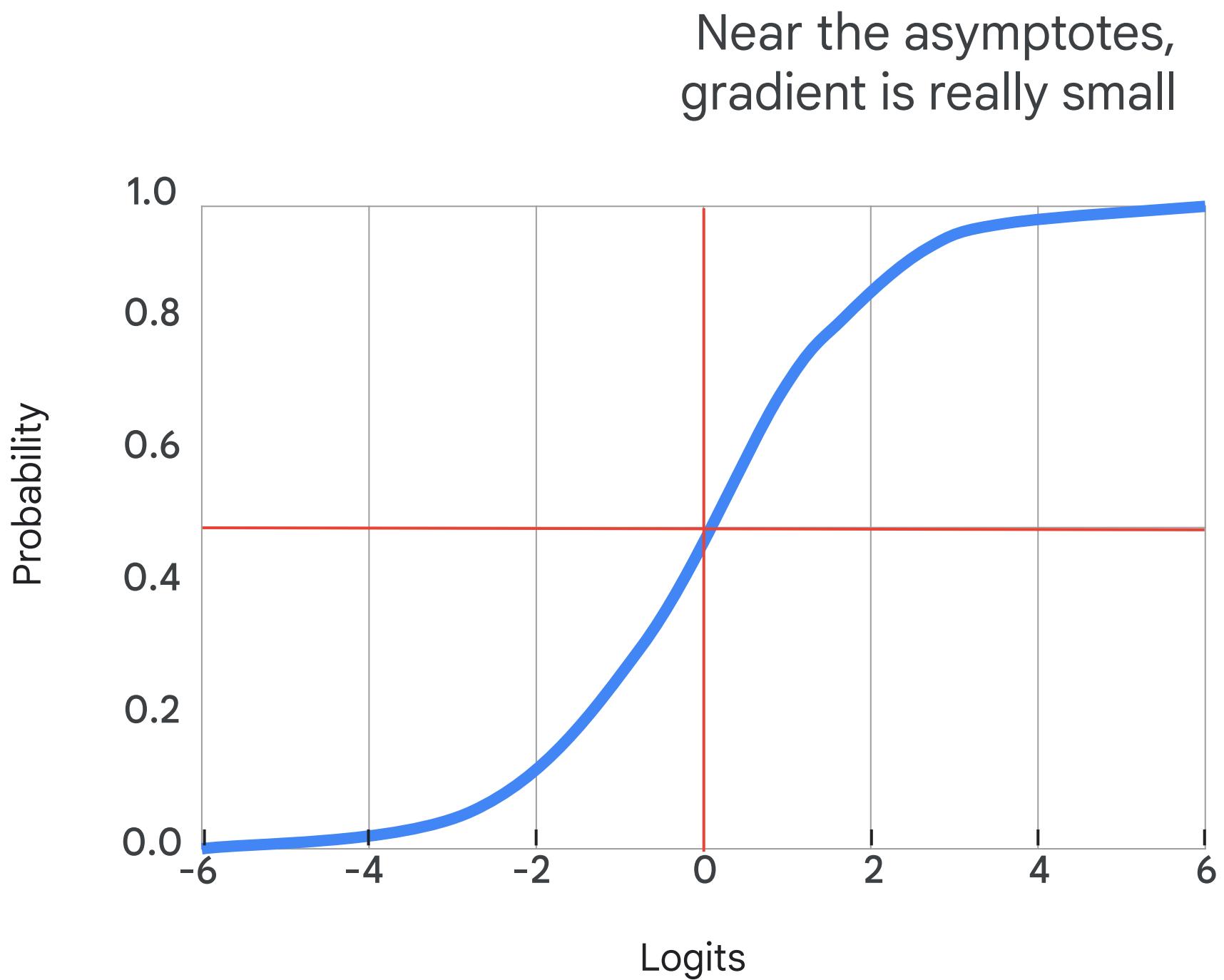
Typically, use cross-entropy
(related to Shannon's
information theory) as the
error metric

Less emphasis on errors where the output is
relatively close to the label.



$$\text{LogLoss} = \sum_{(x,y) \in D} -y\log(\hat{y}) - (1 - y)\log(1 - \hat{y})$$

**Regularization is
important in logistic
regression because
driving the loss to
zero is difficult
and dangerous**



Weights will be driven to $-\infty$
and $+\infty$ the longer we train

Quiz: Logistic regression regularization

Why is it important to add regularization to logistic regression?

- A. Helps stops weights being driven to +/- infinity.
- B. Helps logits stay away from asymptotes which can halt training
- C. Transforms outputs into a calibrated probability estimate
- D. Both A & B
- E. Both A & C



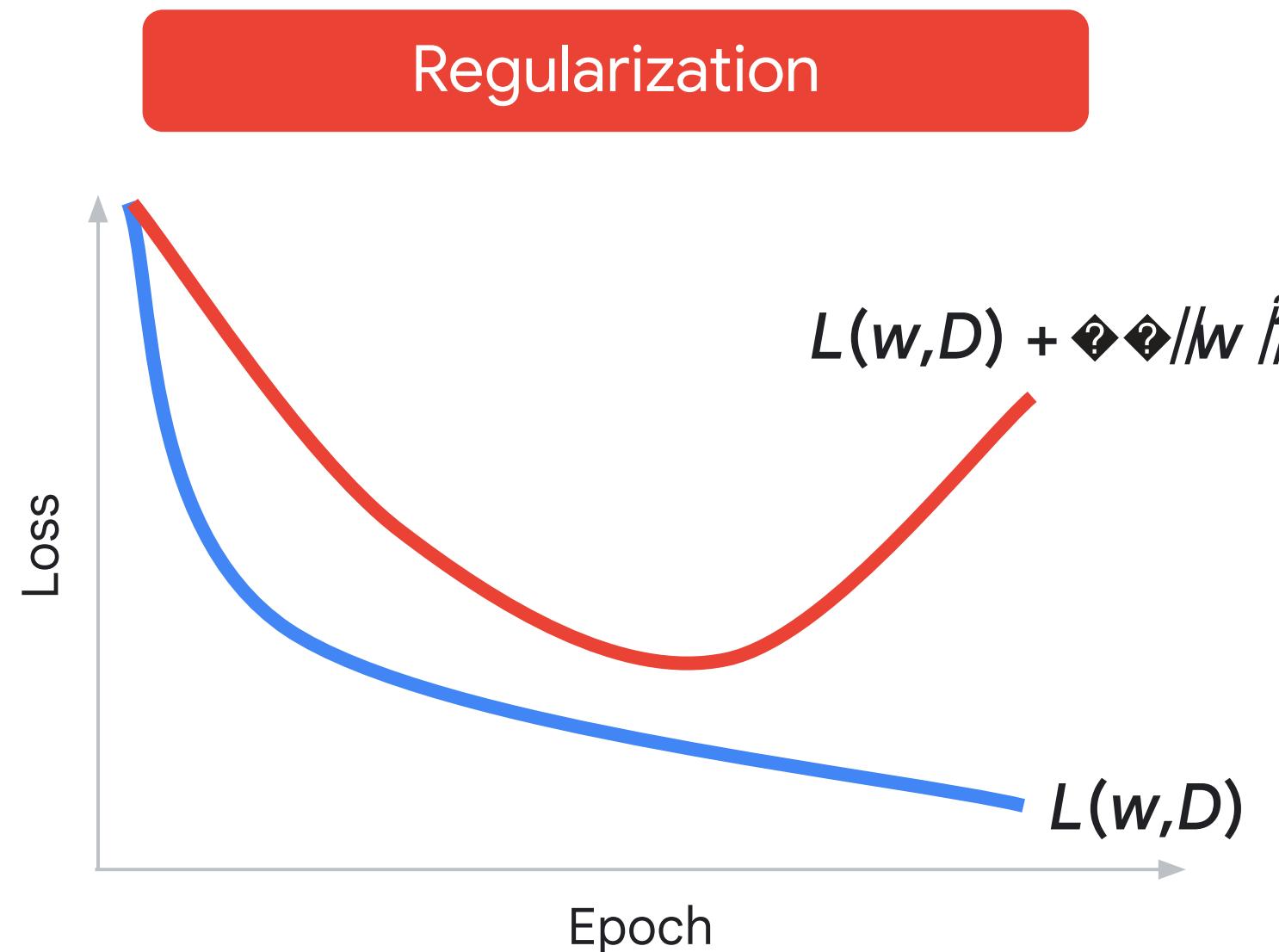
Quiz: Logistic regression regularization

Why is it important to add regularization to logistic regression?

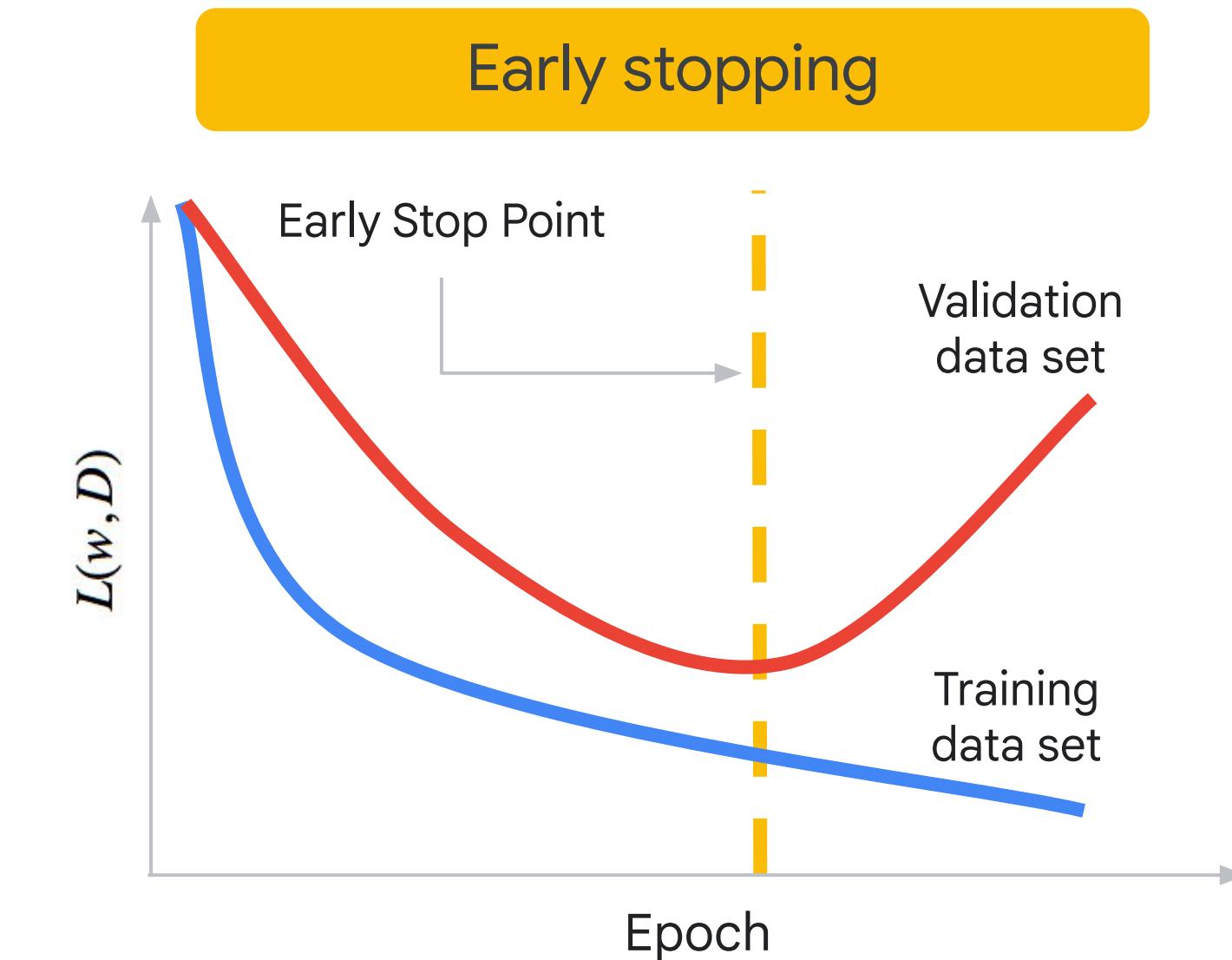
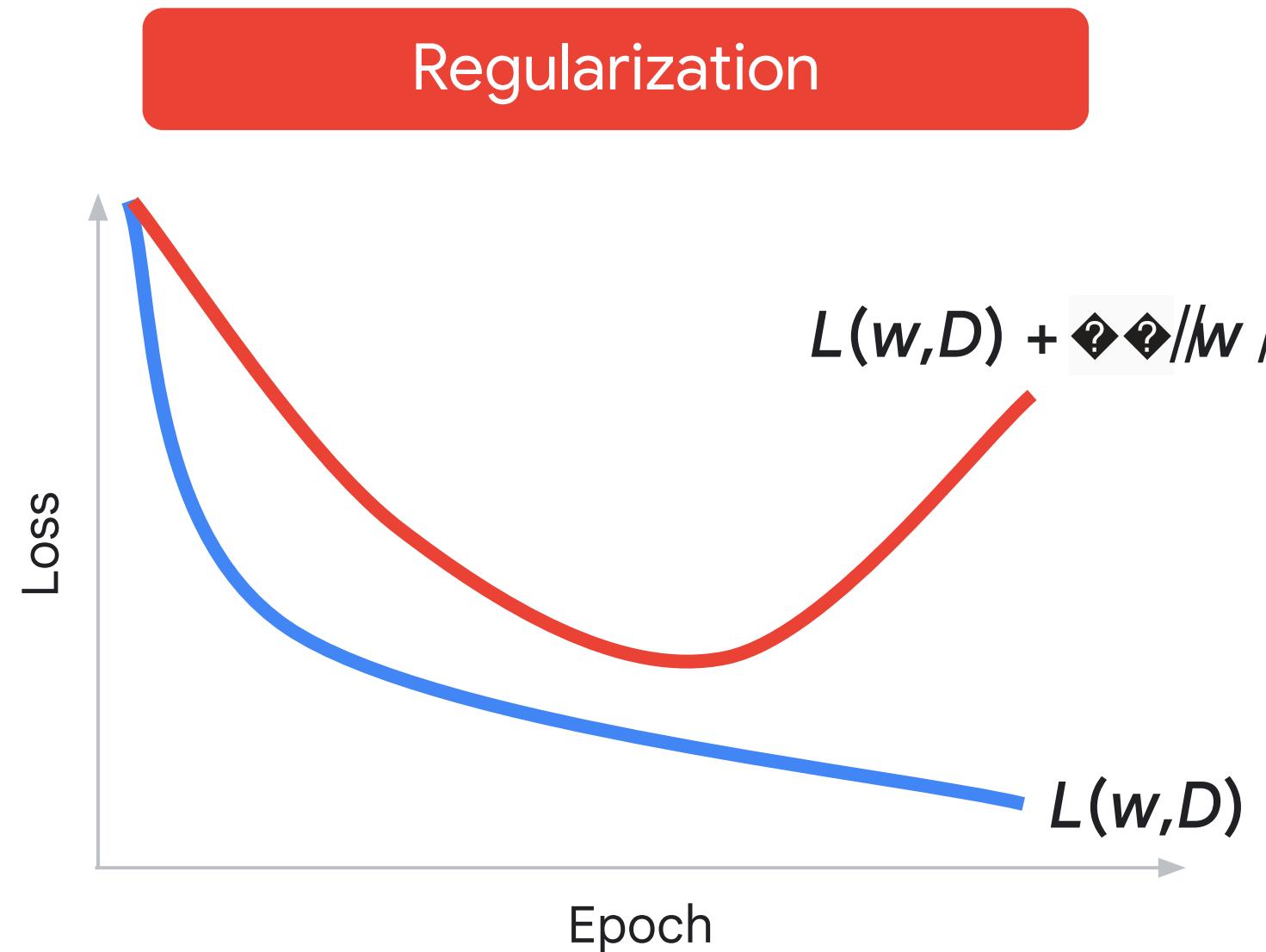
- A. Helps stops weights being driven to +/- infinity.
- B. Helps logits stay away from asymptotes which can halt training
- C. Transforms outputs into a calibrated probability estimate
- D. Both A & B**
- E. Both A & C



Often we do both regularization and early stopping to counteract overfitting



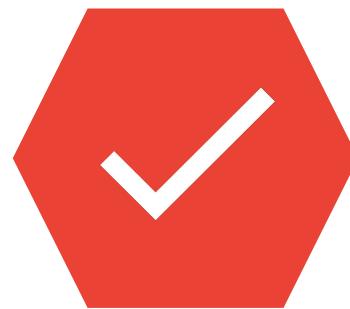
Often we do both regularization and early stopping to counteract overfitting



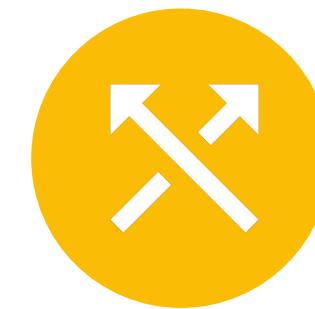
In many real-world problems, the probability is not enough; we need to make a binary decision



Send the mail to
spam folder or not?



Approve the loan
or not?



Which road should we
route the user through?

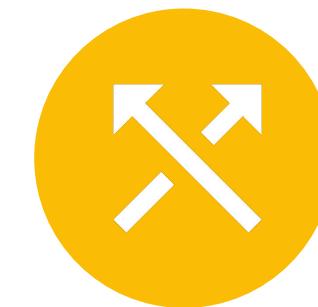
In many real-world problems, the probability is not enough; we need to make a binary decision



Send the mail to
spam folder or not?



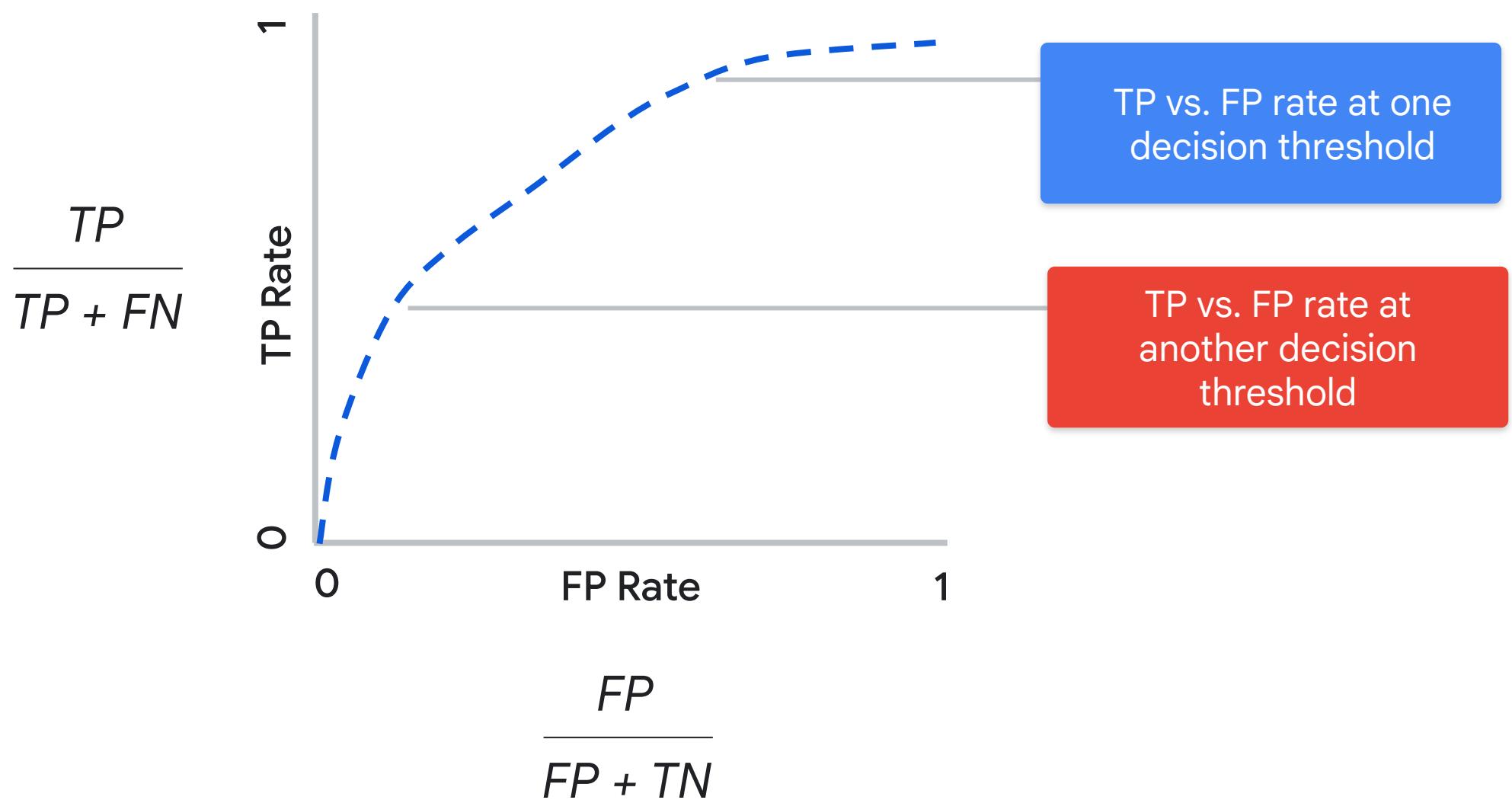
Approve the loan
or not?



Which road should we
route the user through?

Choice of threshold is important and can be tuned

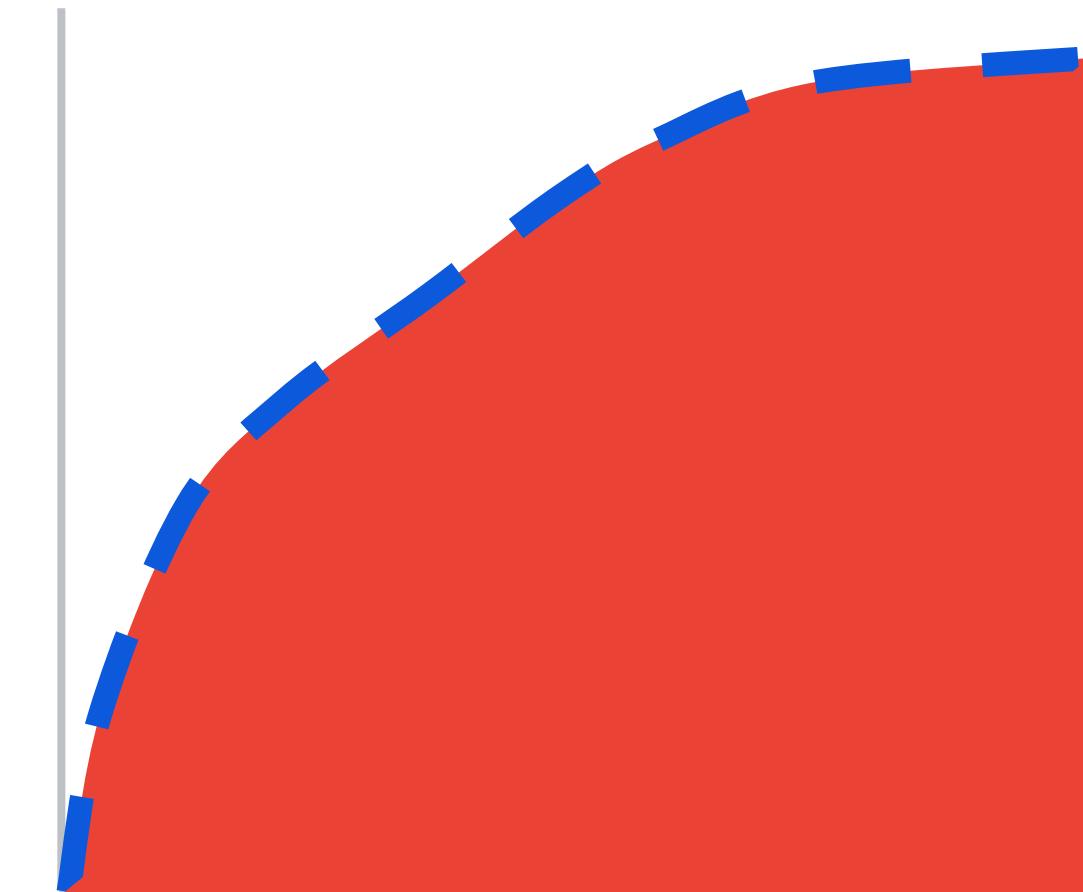
**Use the ROC curve
to choose the
decision threshold
based on decision
criteria**



The Area-Under-Curve (AUC) provides an aggregate measure of performance across all possible classification thresholds

AUC helps you choose between models when you don't know what decision threshold is going to be ultimately used.

"If we pick a random positive and a random negative, what's the probability my model scores them in the correct relative order?"



Logistic regression predictions should be unbiased

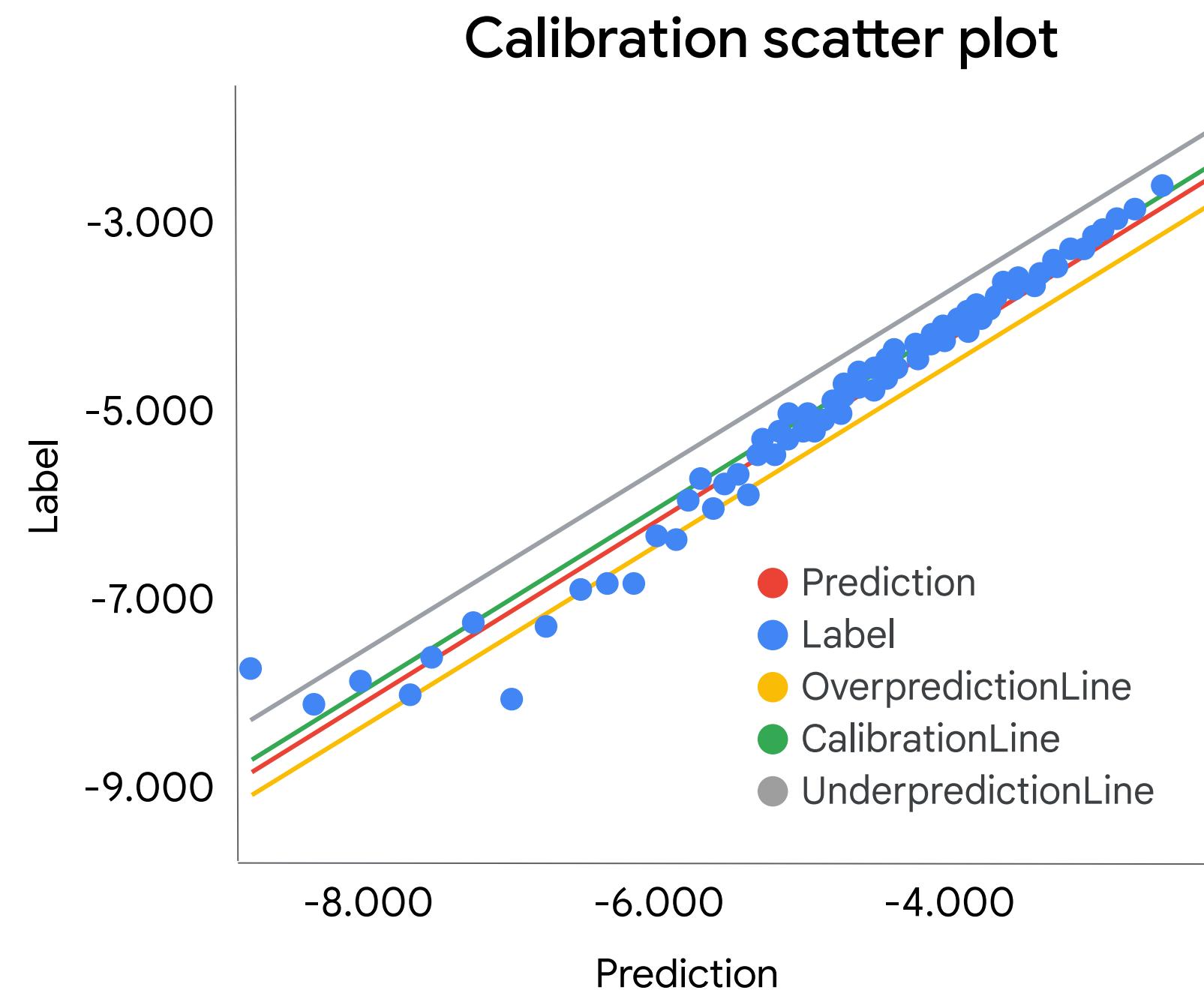
average of predictions == average of observations



Look for bias in slices of data, this can guide improvements.

**Use calibration
plots of bucketed
bias to find slices
where your model
performs poorly**

Each dot represents many examples in
the same bucketed prediction range



Quiz: Logistic regression

Which of these is important when performing logistic regression?

- A. Adding regularization
- B. Choosing a tuned threshold
- C. Checking for bias
- D. All of the above



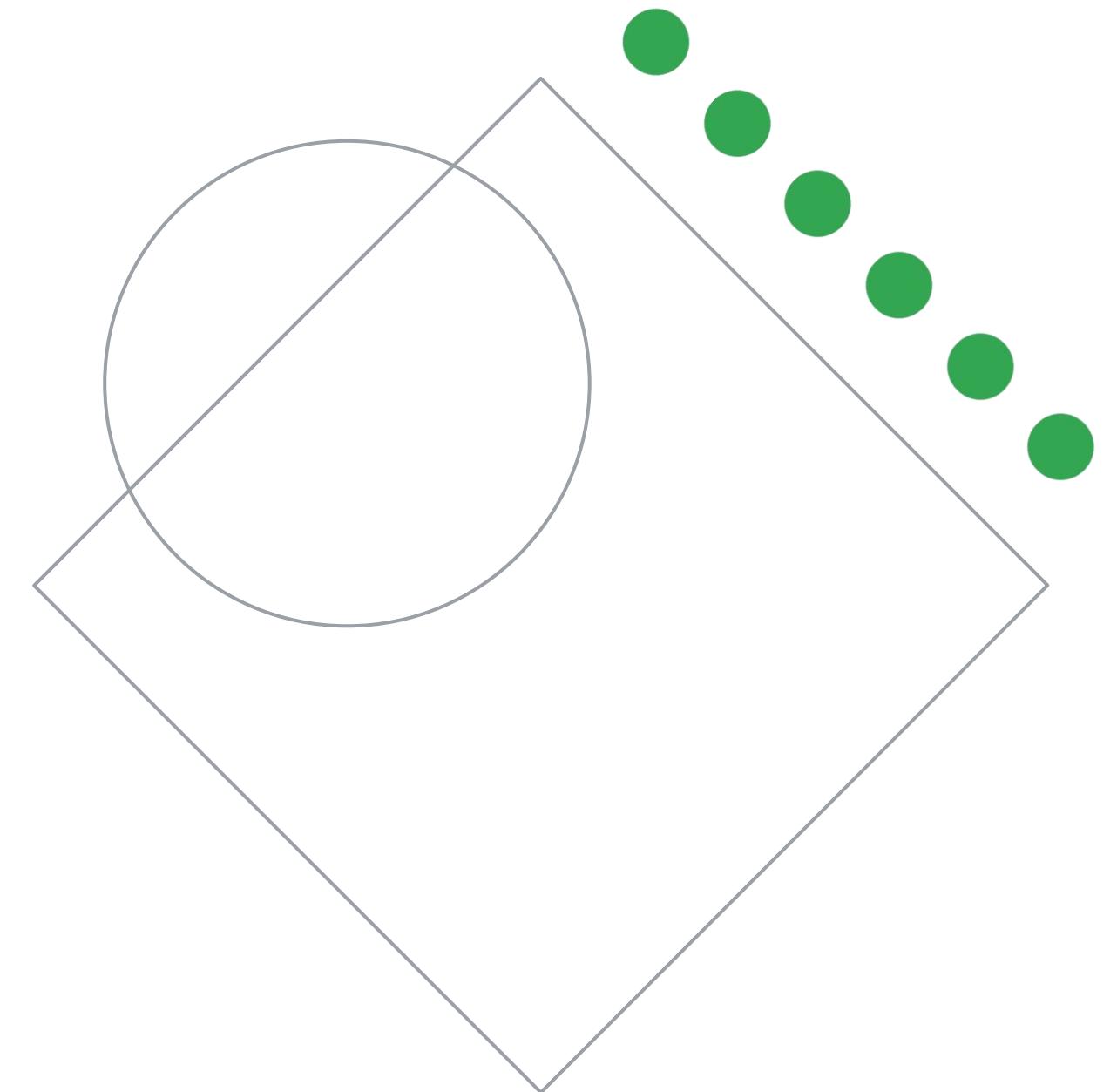
Quiz: Logistic regression

Which of these is important when performing logistic regression?

- A. Adding regularization
- B. Choosing a tuned threshold
- C. Checking for bias
- D. All of the above**



Introduction to automated machine learning (AutoML) using Vertex AI



In this module, you learn to ...

01

Define automated machine learning

02

Describe how to train a Vertex AI AutoML regression model

03

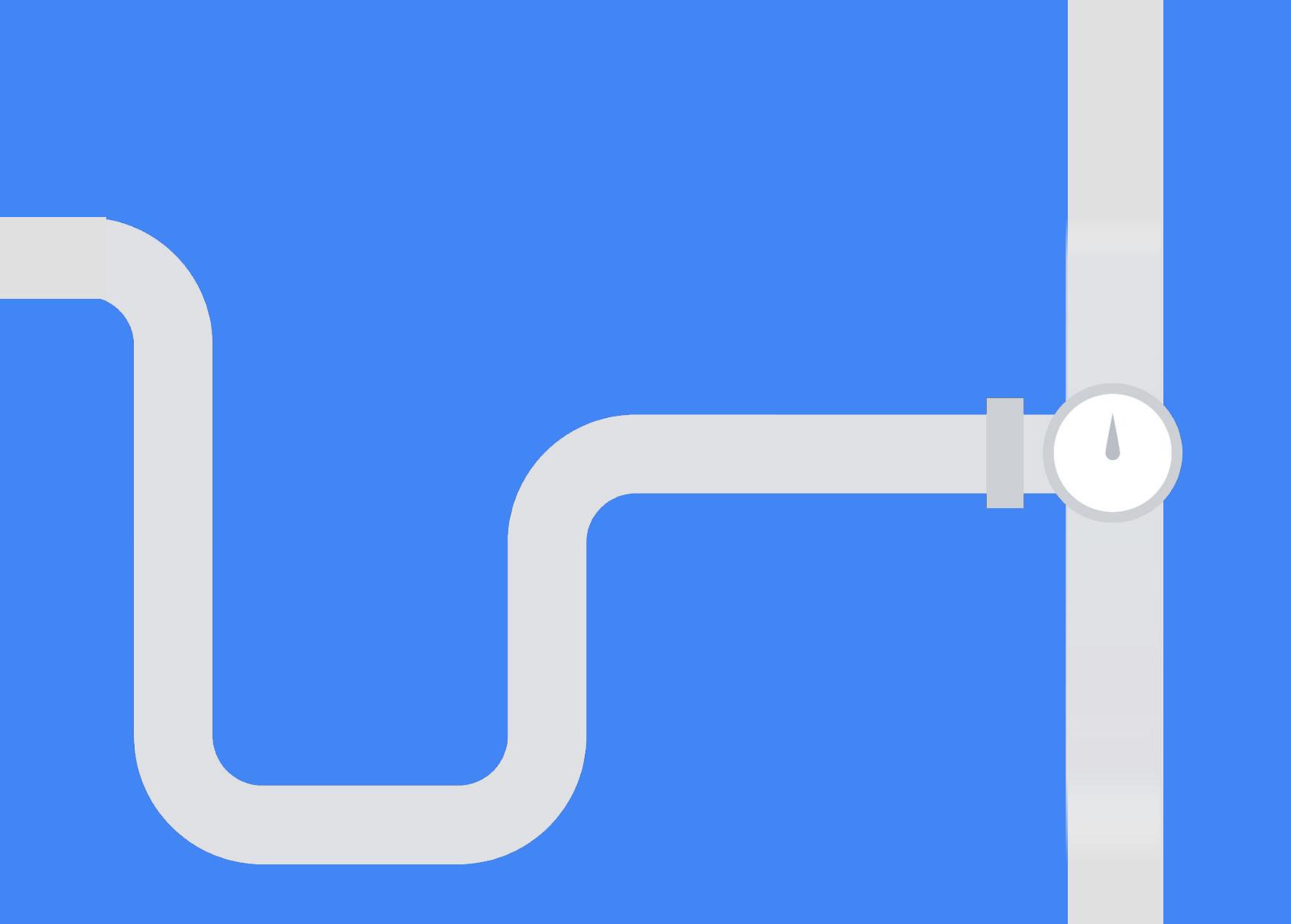
Explain how to evaluate Vertex AI AutoML models



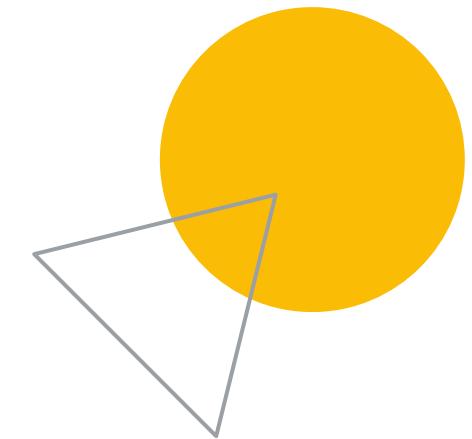
Machine learning vs. deep learning

All machine learning starts with a
business requirement or problem
you are **trying to solve**.

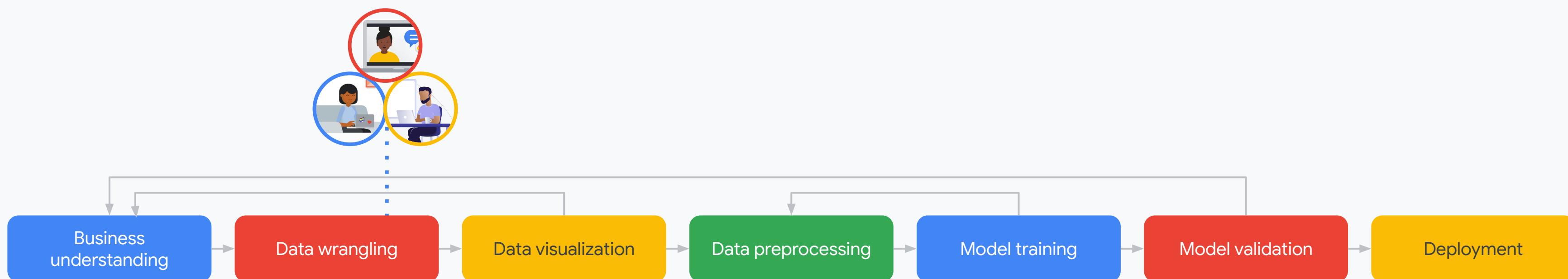




The machine learning pipeline



Pipelines are used to automate data preprocessing and model training for easy model upgrading.



Business use case

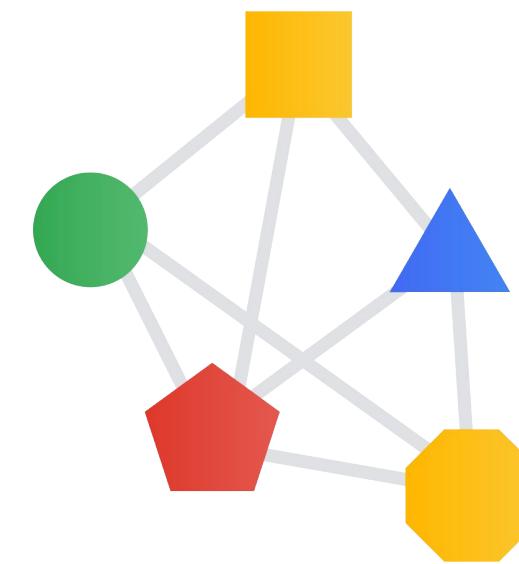
A team at XYZ company has just:



Defined business
use case



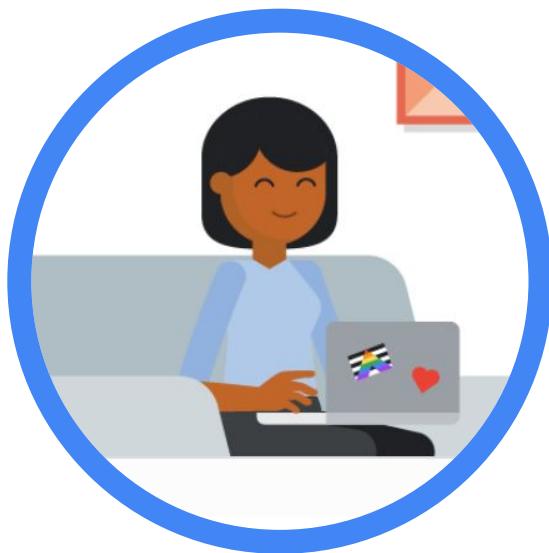
Established
success criteria



Delivered ML model
to production

Business use case

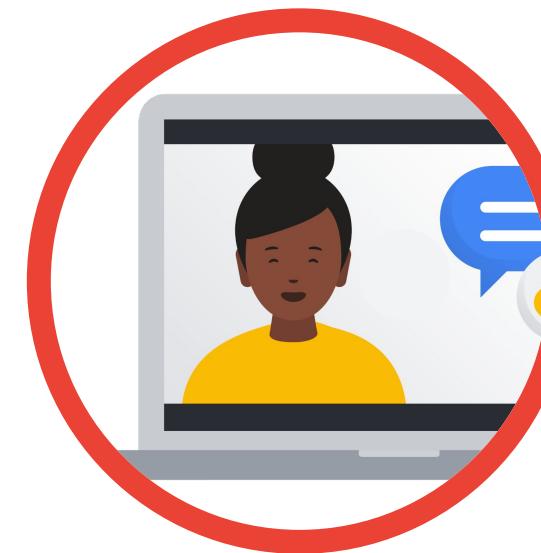
The team consists of:



Software Developer



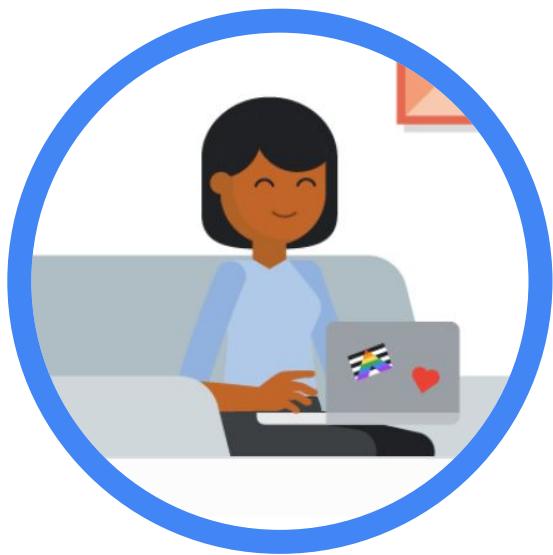
Data Analyst



Data Scientist

The goal is to predict consumer spending score.
The dataset contains structured data, so no images or videos.

What should they do?



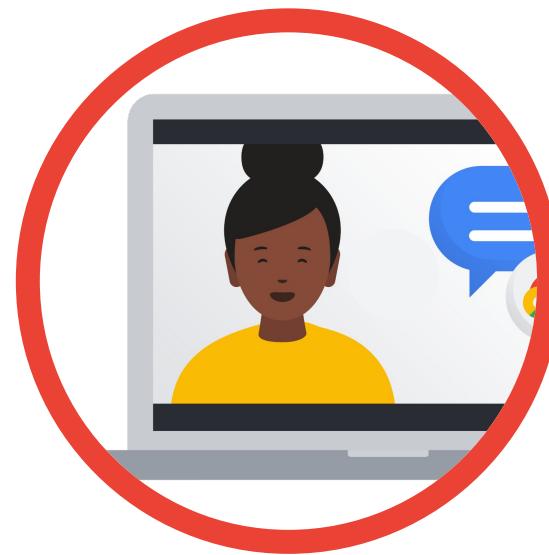
Software Developer

Knows Java; no ML experience.



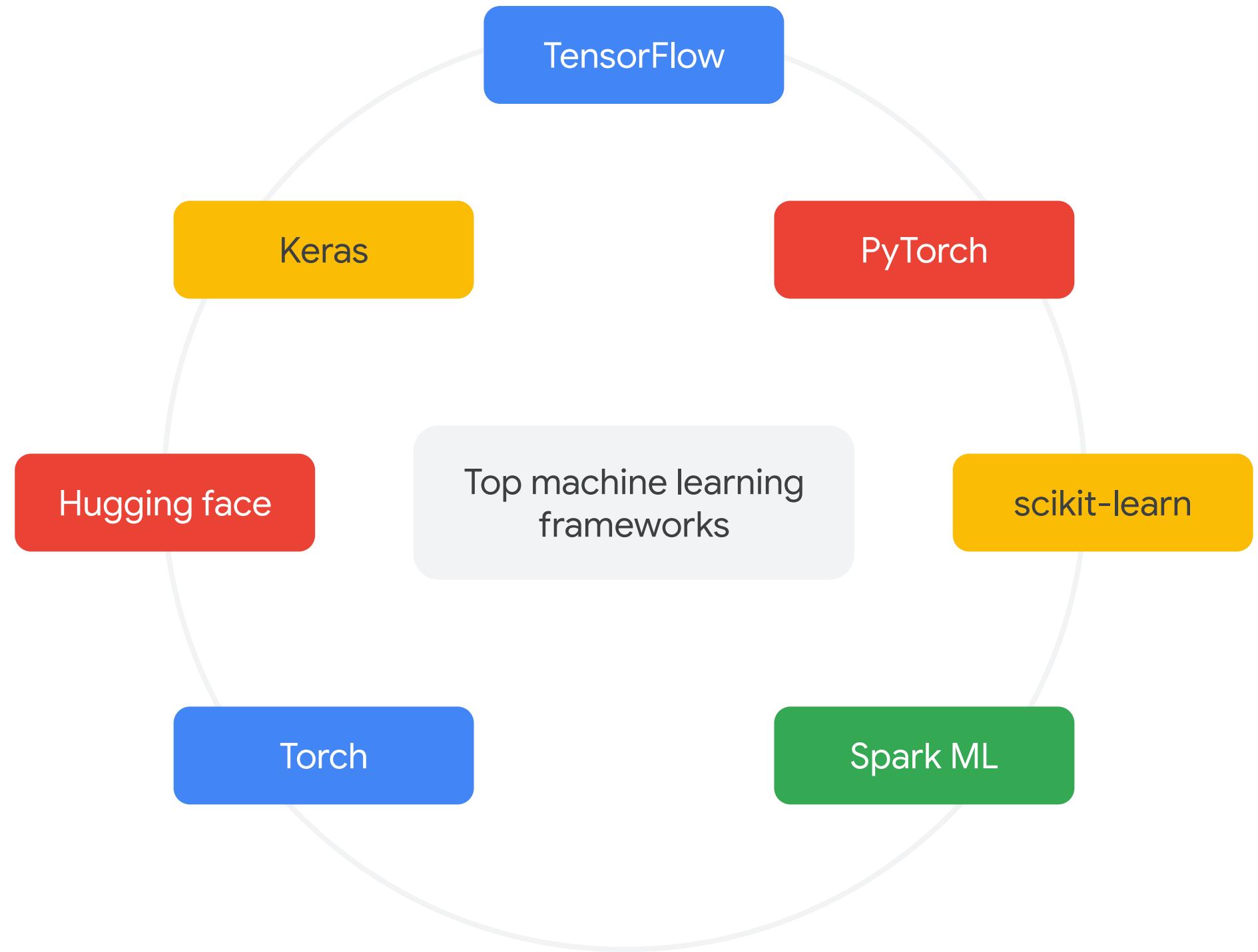
Data Analyst

Knows SQL but no ML,
and wants a
“point and click” solution.

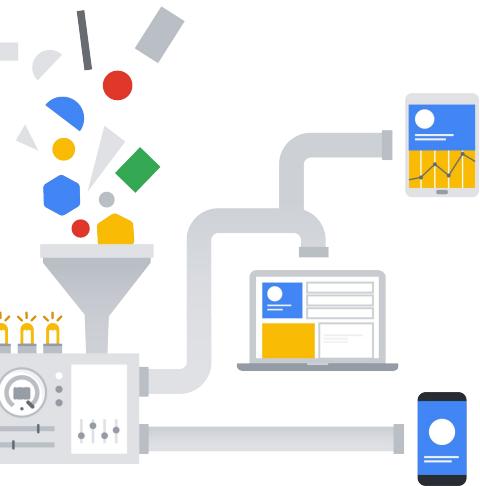


Data Scientist

Domain knowledge.
Limited experience
putting a model
into production.



Machine learning versus **statistics**



Machine learning

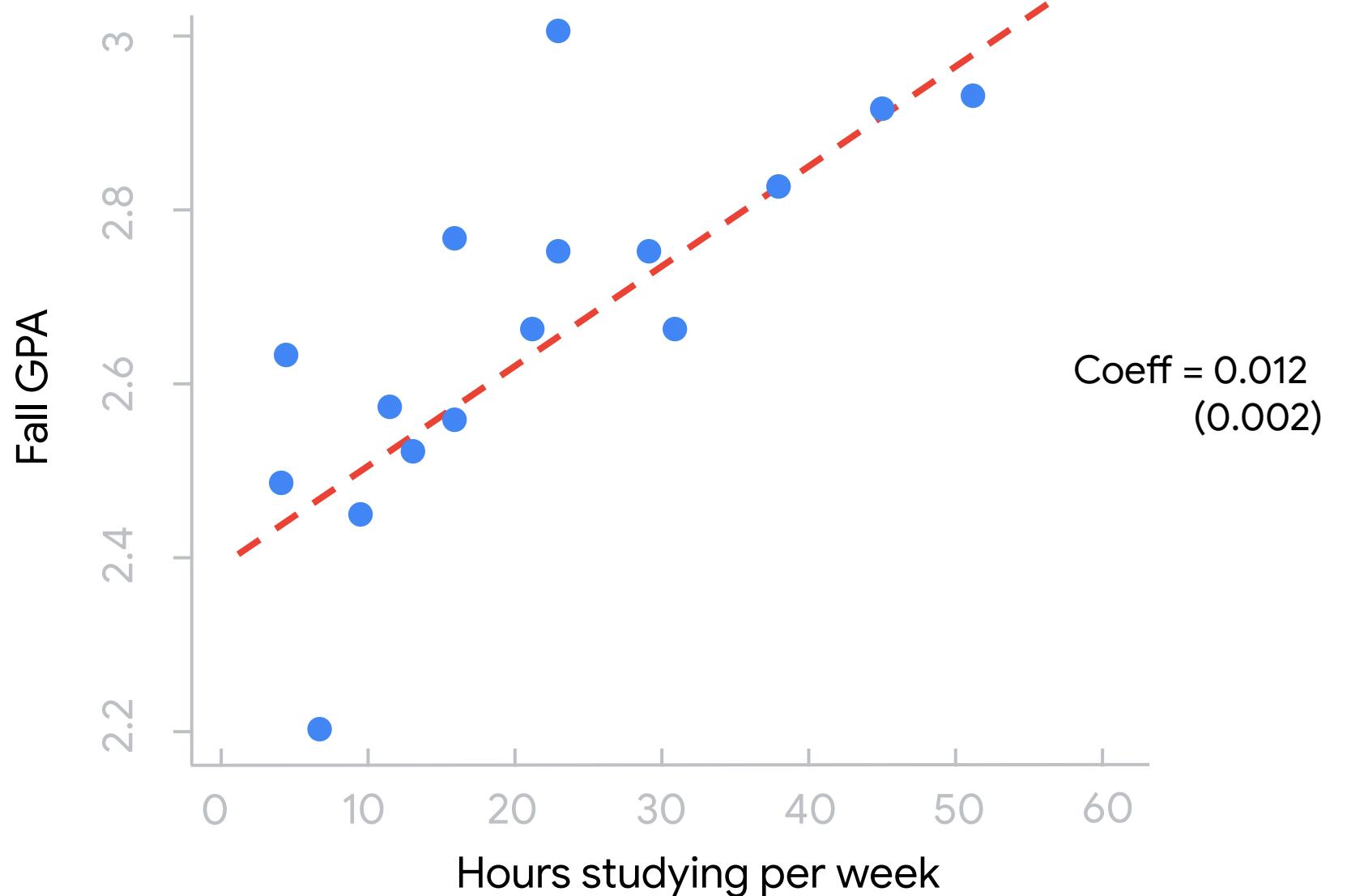
Lots of data. Keep outliers
and build models for them.



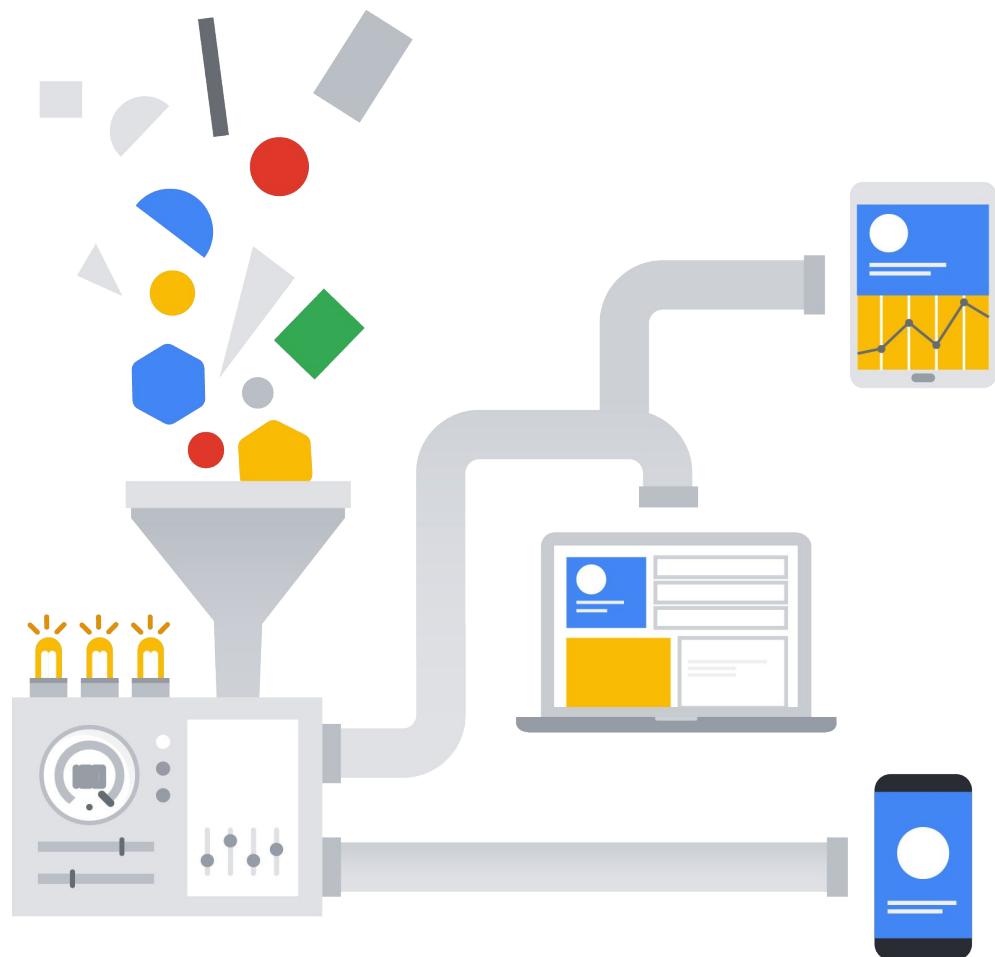
Statistics

"I've got all the data I'll
ever get." Throw away
outliers.

Machine learning versus statistics



Machine learning



Statistics



Machine learning versus statistics

	Machine learning	Standard statistics (linear/logistic regressions)
Data preparation?	Doesn't require explicit commands to find patterns in data	Need to know variables and parameters beforehand
Hypothesis	No hypothesis needed	Need hypothesis to test
Type of data?	Multi-dimensional data that can be non-linear in nature	Linear data
Training?	Needs to be "trained"	No training
Goal?	Generally better for predictions	Generally better for inferences/hypothesis testing
Scientific question?	What will happen?	How/why it happened?

Machine learning versus statistics

	Machine learning	Standard statistics (linear/logistic regressions)
Data preparation?	Doesn't require explicit commands to find patterns in data	Need to know variables and parameters beforehand
Hypothesis	No hypothesis needed	Need hypothesis to test
Type of data?	Multi-dimensional data that can be non-linear in nature	Linear data
Training?	Needs to be "trained"	No training
Goal?	Generally better for predictions	Generally better for inferences/hypothesis testing
Scientific question?	What will happen?	How/why it happened?

Machine learning versus statistics

Machine learning	Standard statistics (linear/logistic regressions)
Data preparation?	Doesn't require explicit commands to find patterns in data
Hypothesis	No hypothesis needed
Type of data?	Multi-dimensional data that can be non-linear in nature
Training?	Needs to be "trained"
Goal?	Generally better for predictions
Scientific question?	What will happen? How/why it happened?

Machine learning versus statistics

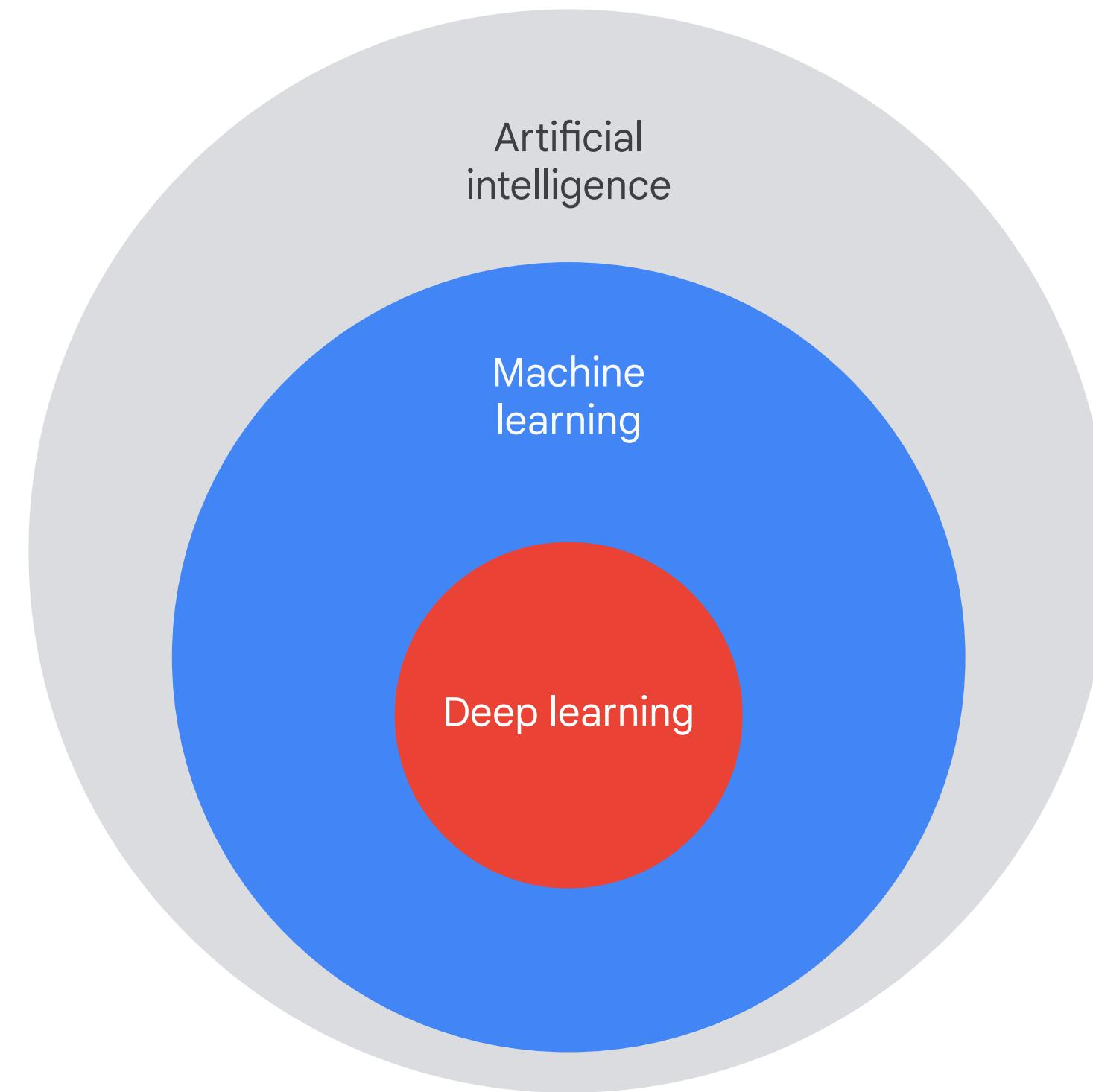
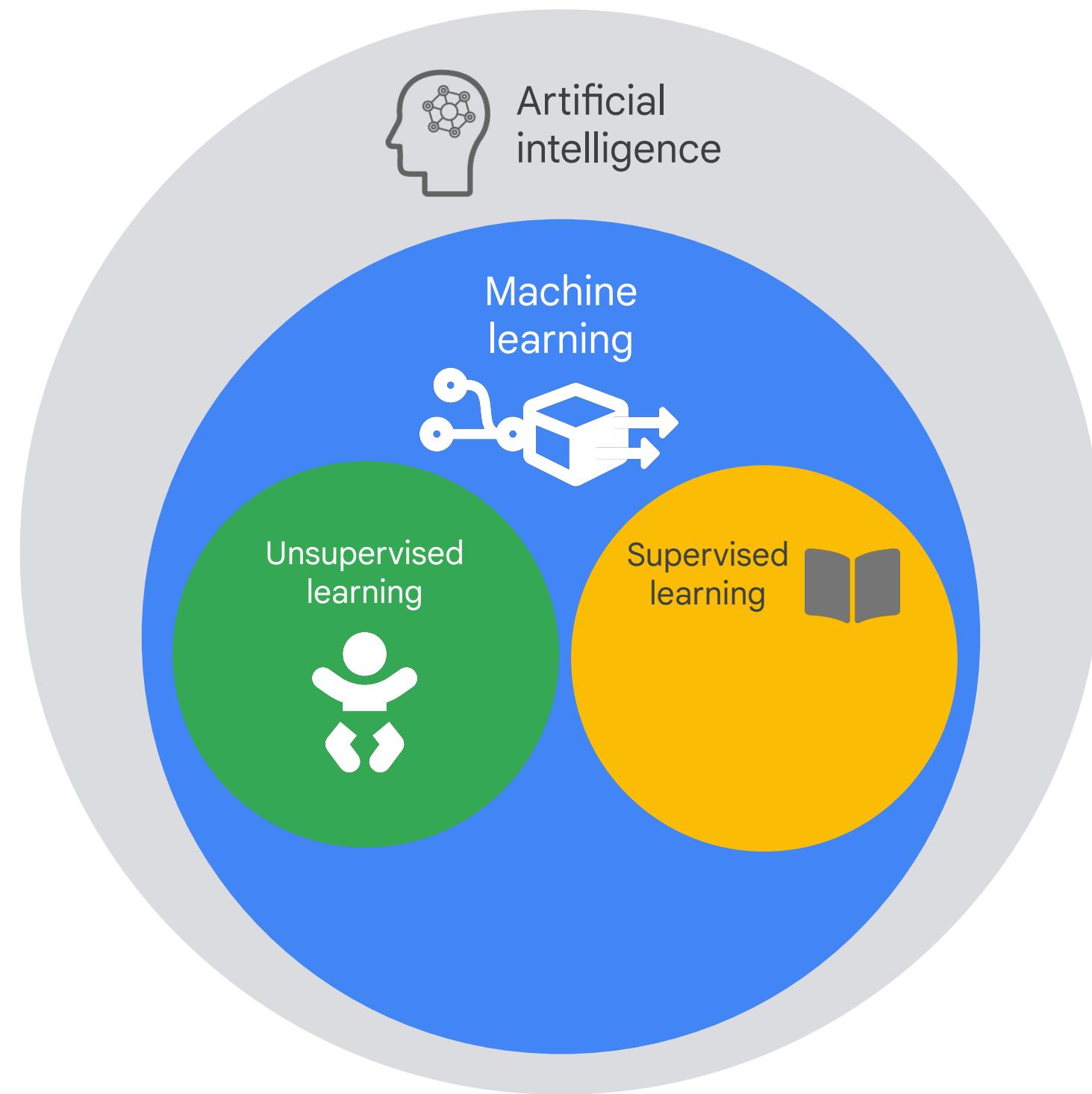
Machine learning	Standard statistics (linear/logistic regressions)
Data preparation?	Doesn't require explicit commands to find patterns in data
Hypothesis	No hypothesis needed
Type of data?	Multi-dimensional data that can be non-linear in nature
Training?	Needs to be "trained"
Goal?	Generally better for predictions
Scientific question?	What will happen? How/why it happened?

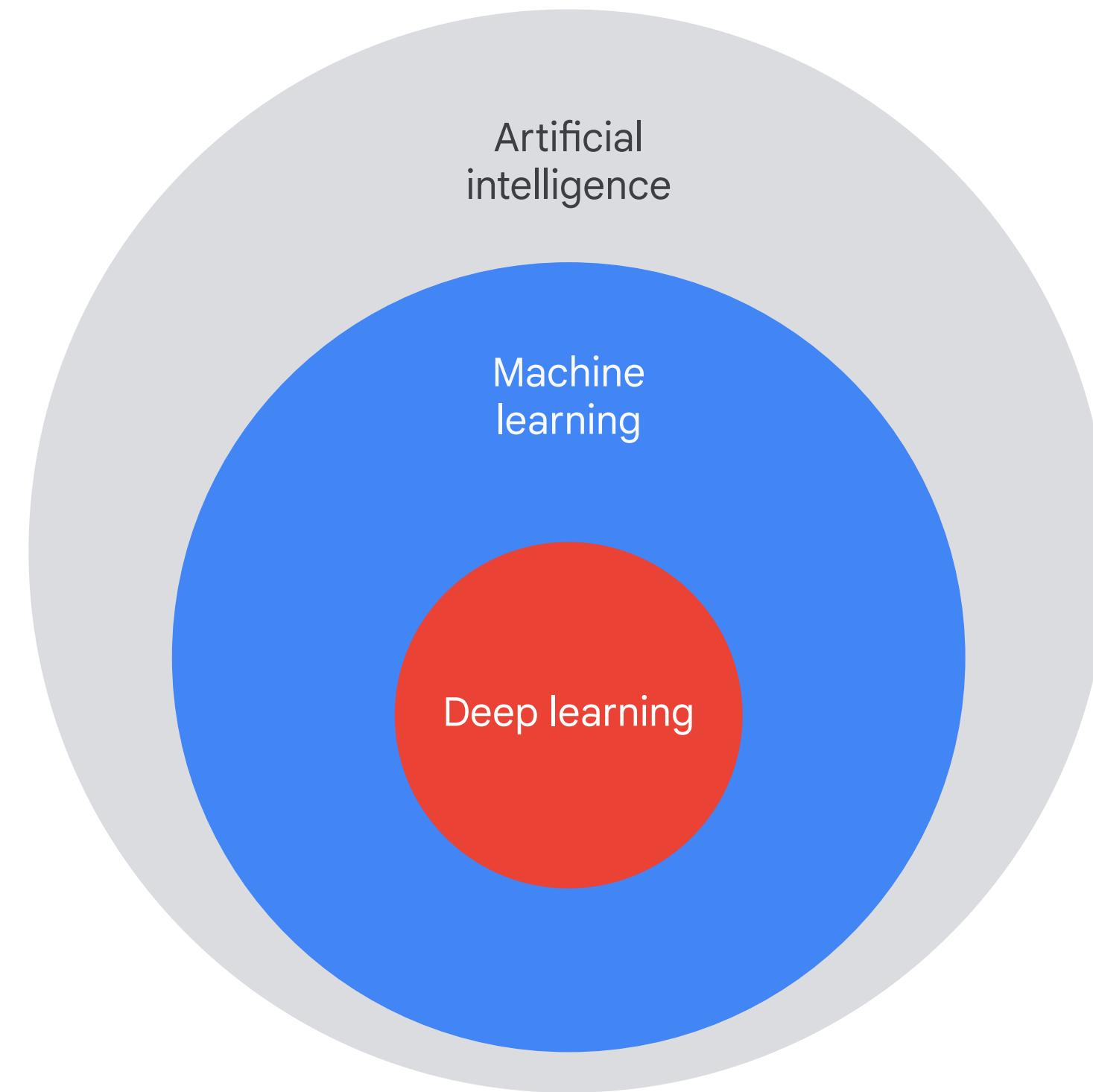
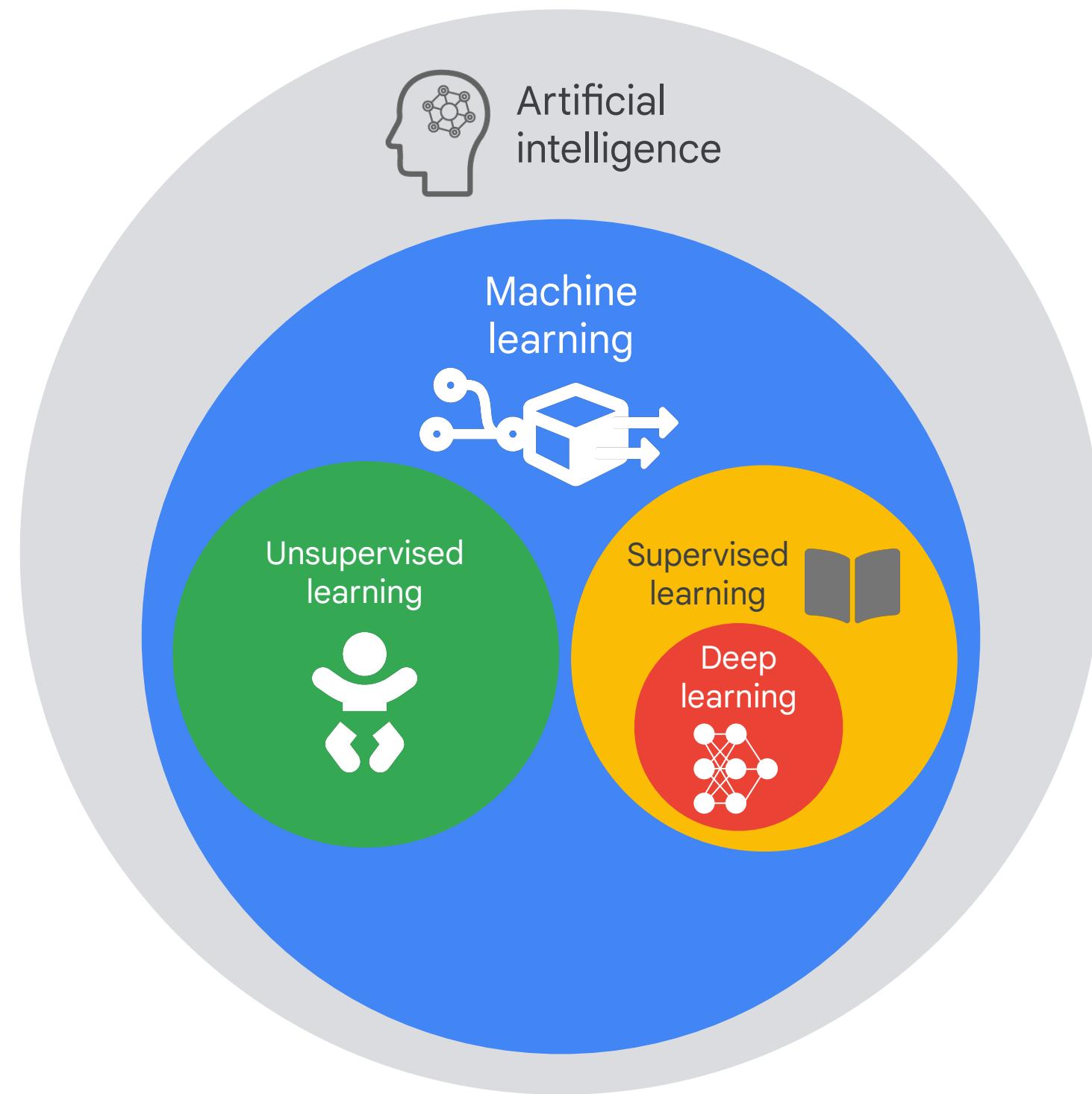
Machine learning versus statistics

Machine learning	Standard statistics (linear/logistic regressions)
Data preparation?	Doesn't require explicit commands to find patterns in data
Hypothesis	No hypothesis needed
Type of data?	Multi-dimensional data that can be non-linear in nature
Training?	Needs to be "trained"
Goal?	Generally better for predictions
Scientific question?	What will happen? How/why it happened?

Machine learning versus statistics

Machine learning	Standard statistics (linear/logistic regressions)
Data preparation?	Doesn't require explicit commands to find patterns in data
Hypothesis	No hypothesis needed
Type of data?	Multi-dimensional data that can be non-linear in nature
Training?	Needs to be "trained"
Goal?	Generally better for predictions
Scientific question?	What will happen? How/why it happened?







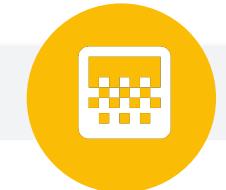
Factors

- Data requirement
- Accuracy
- Training time
- Hardware dependency
- Hyperparameter tuning



Deep learning

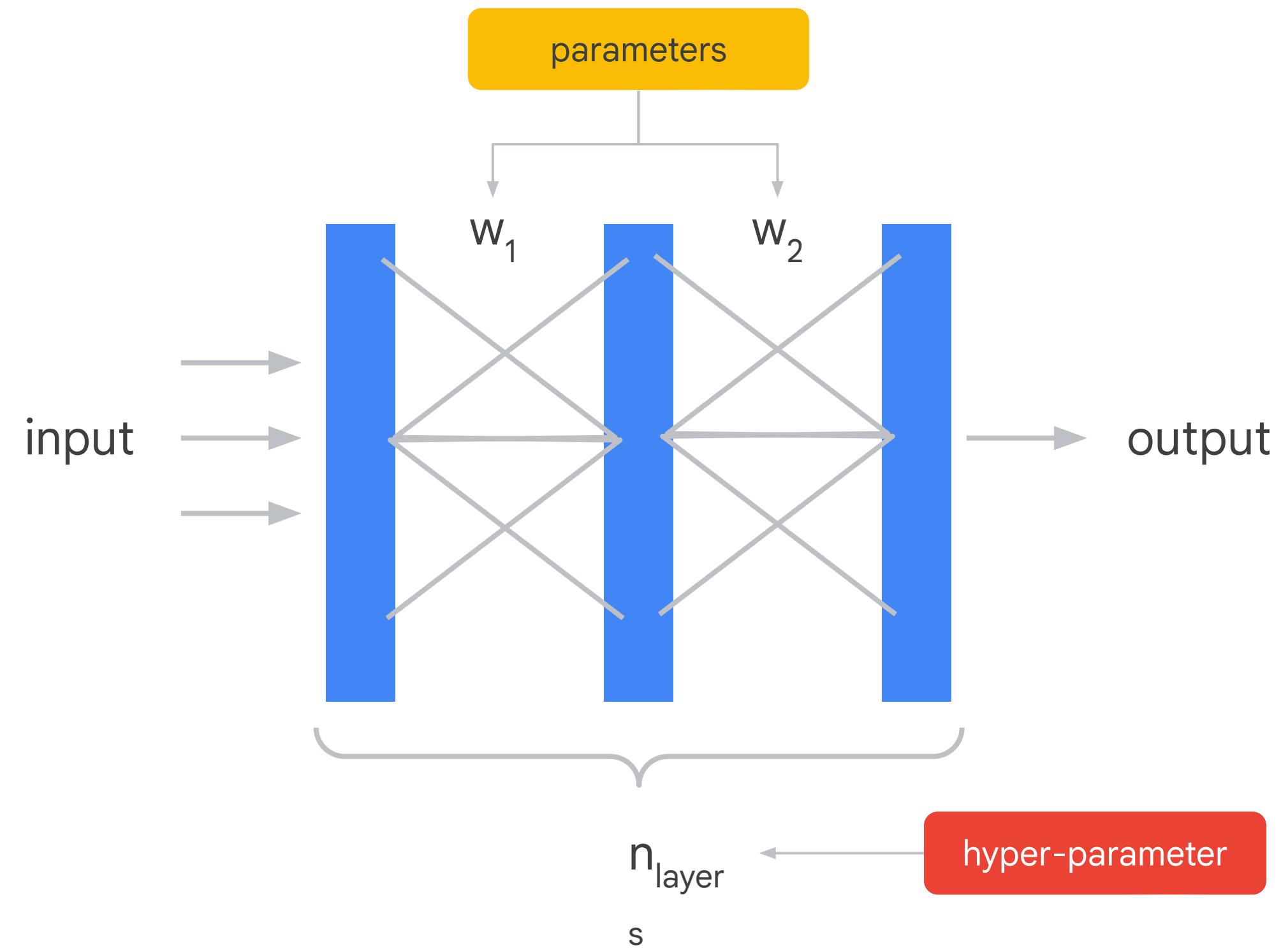
- Requires large data
- Provides high accuracy
- Takes longer to train
- Requires GPU to train properly
- Can be tuned in various ways



Machine learning

- Can train on lesser data
- Gives lesser accuracy
- Takes less time to train
- Trains on CPU
- Limited tuning capabilities

Deep learning



Summary

01

Machine learning is a subfield of artificial intelligence.

02

The goal is to make computers learn from your data.

03

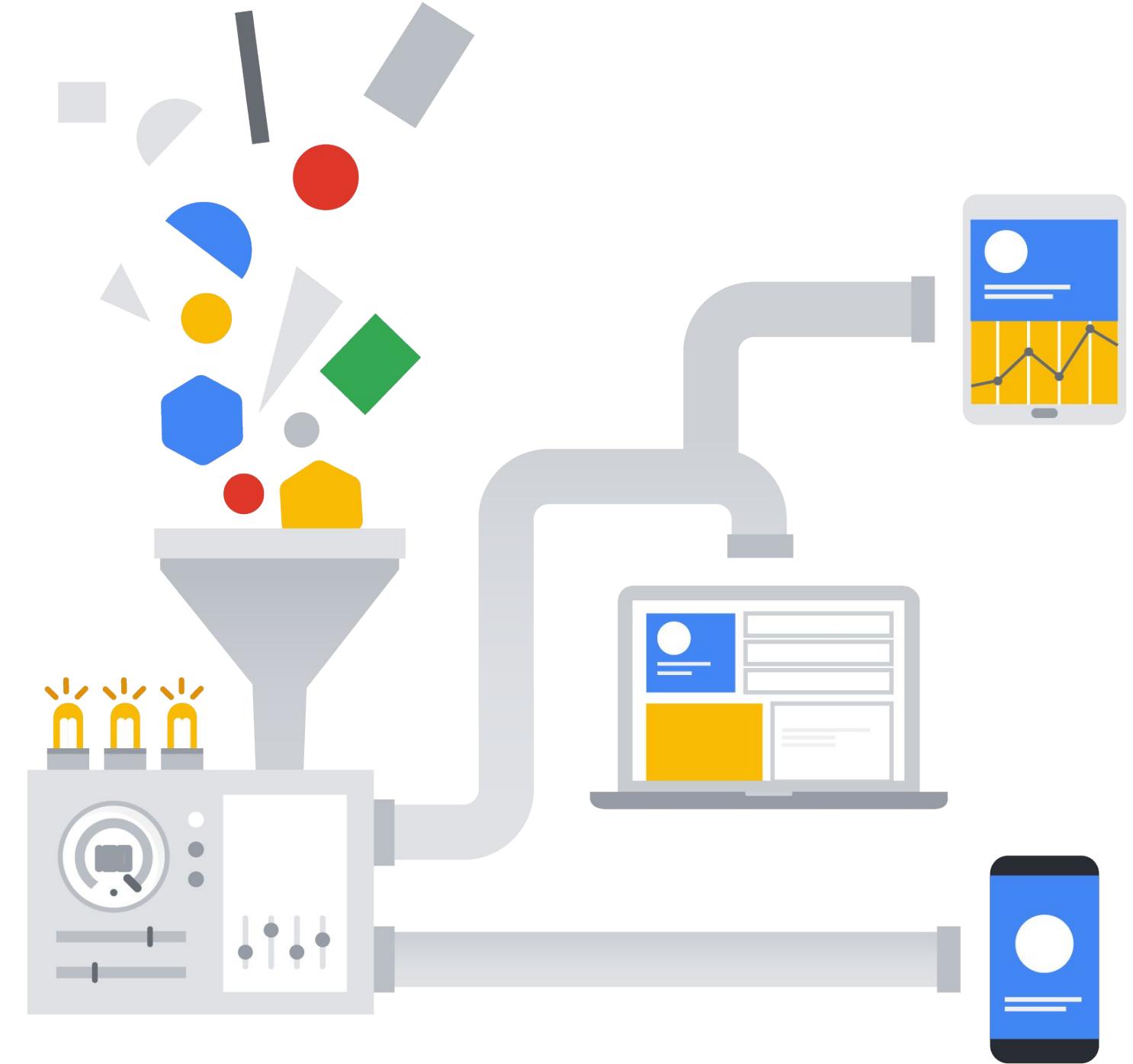
Your code provides an algorithm that adapts.

04

The result (the algorithm and the learned parameters) is your trained model.

What is AutoML?

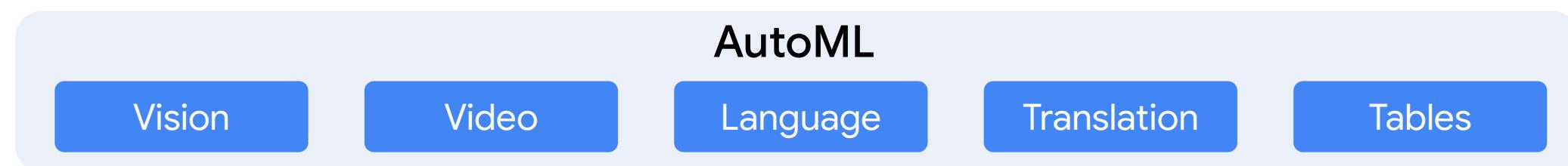
Automate the process
of applying ML to
real-world problems



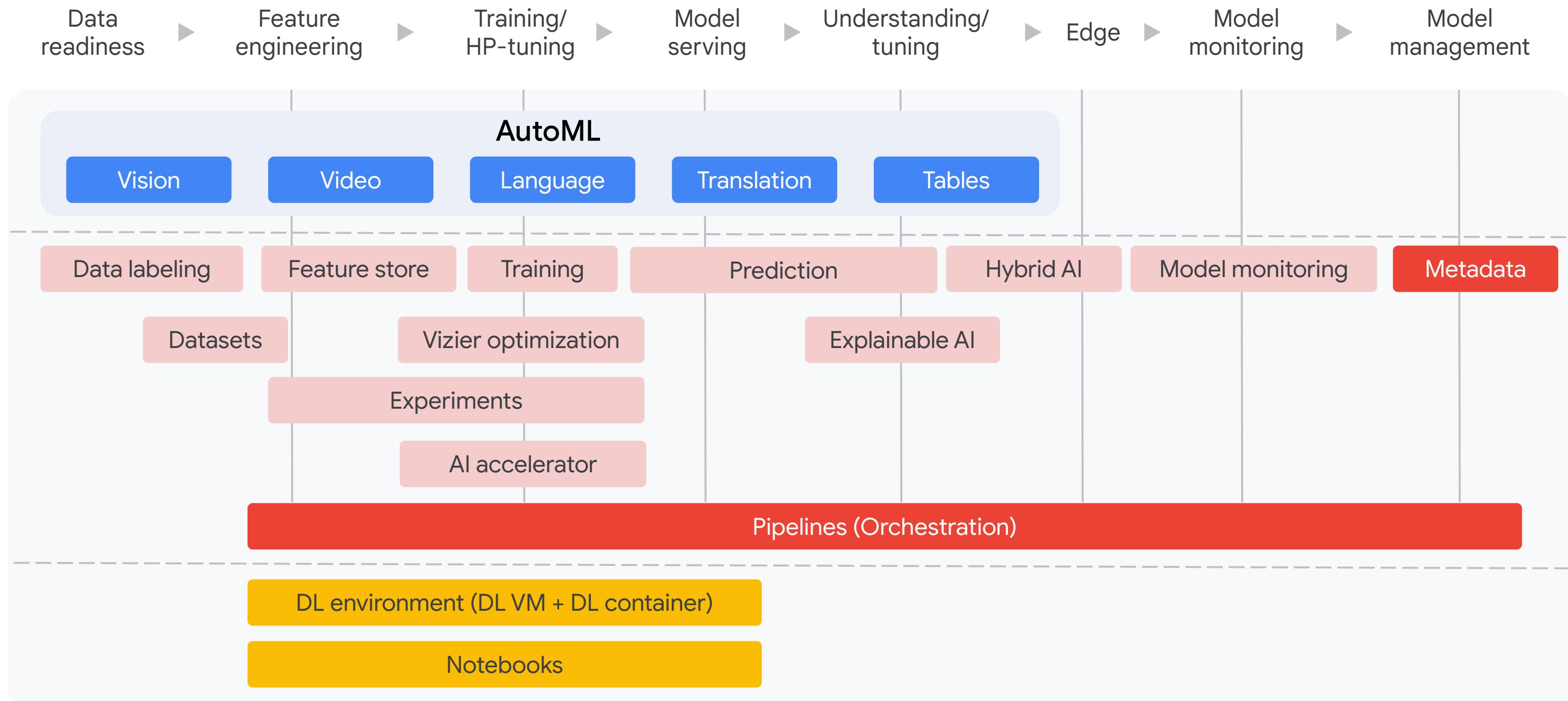
What is “automated” in AutoML?

Components of an ML pipeline

Data readiness ➔ Feature engineering ➔ Training/ HP-tuning ➔ Model serving ➔ Understanding/ tuning ➔ Edge ➔ Model monitoring ➔ Model management



What's included in Vertex AI?



Vertex AI AutoML

Business use case

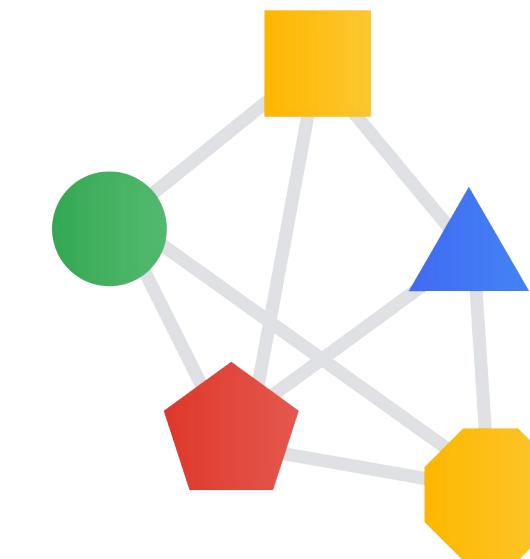
A team at XYZ company has just defined a business use case, established the success criteria, and wants to deliver an ML model to production. It will be their first ML project.



Define business use case



Establish success criteria



Deliver ML model
to production

Which machine learning framework should they use?



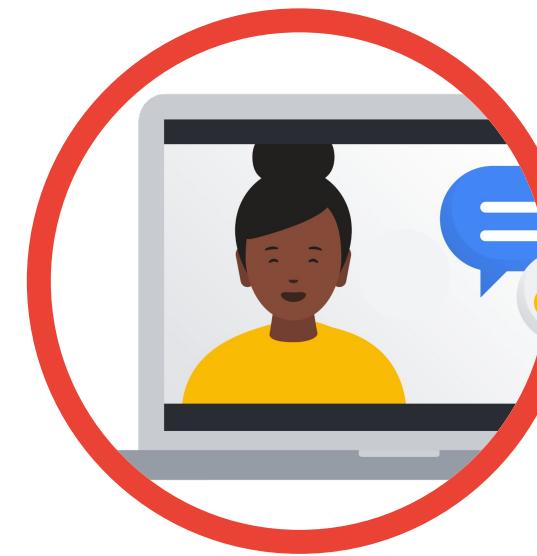
Software Developer

Knows Java; no ML experience.



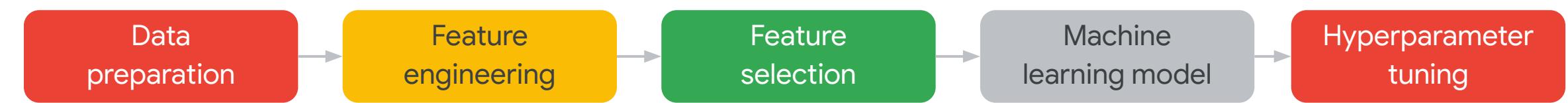
Data Analyst

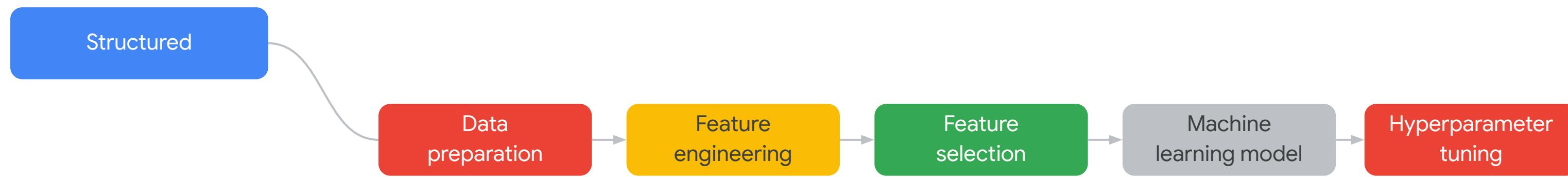
Knows SQL but no ML,
and wants a
“point and click” solution.

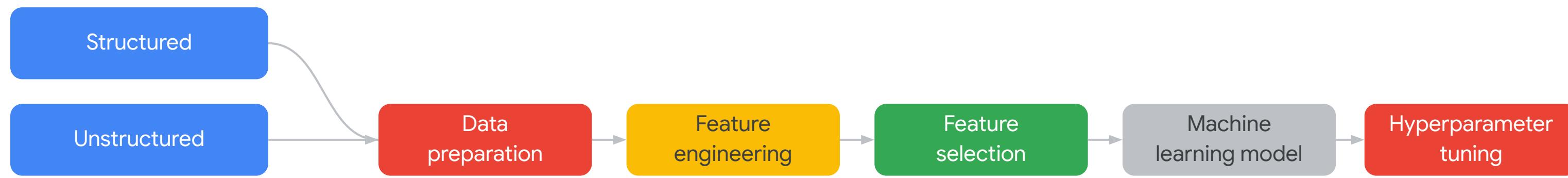


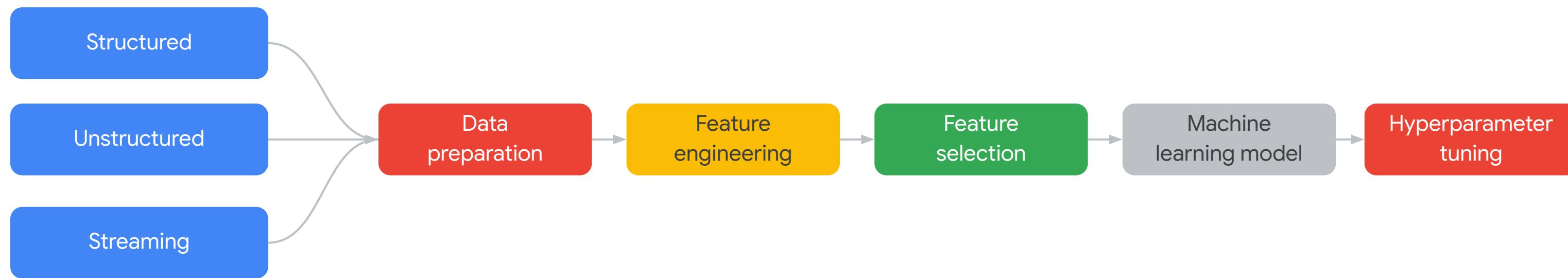
Data Scientist

Domain knowledge.
Limited experience
putting a model
into production.









Dataset

```
✓ [1] import pandas as pd
0s

✓ ⏎ df = pd.read_csv('consumer_spend.csv')
0s      data.head()

   ↗   Graduated   Profession Work_Experience Family_Size Spending_Score
  0     No        Healthcare          1.0           4.0            Low
  1    Yes       Engineer           NaN           3.0          Average
  2    Yes       Engineer          1.0           1.0            Low
  3    Yes        Lawyer           0.0           2.0            High
  4    Yes Entertainment         NaN           6.0            High
```

```
%>%%writefile train/model_definition.py
# Here we'll import data processing Libraries Like Numpy and TensorFlow
import tensorflow as tf
import numpy as np

# Get data

(x_train, y_train), (x_test, y_test) = tf.keras.datasets.fashion_mnist.load_data()

# add empty color dimension
x_train = np.expand_dims(x_train, -1)
x_test = np.expand_dims(x_test, -1)

def create_model():
    # The `tf.keras.Sequential` method will sequential groups a Linear stack of Layers into a tf.keras.Model.
    model = tf.keras.models.Sequential()
    # The `Flatten()` method will flattens the input and it does not affect the batch size.
    model.add(tf.keras.layers.Flatten(input_shape=x_train.shape[1:]))
    # The `Dense()` method is just your regular densely-connected NN Layer.
    model.add(tf.keras.layers.Dense(1028))
    # The `Activation()` method applies an activation function to an output.
    model.add(tf.keras.layers.Activation('relu'))
    # The `Dropout()` method applies dropout to the input.
    model.add(tf.keras.layers.Dropout(0.5))
    # The `Dense()` method is just your regular densely-connected NN Layer.
    model.add(tf.keras.layers.Dense(512))
    # The `Activation()` method applies an activation function to an output.
    model.add(tf.keras.layers.Activation('relu'))
    # The `Dropout()` method applies dropout to the input.
    model.add(tf.keras.layers.Dropout(0.5))
    # The `Dense()` method is just your regular densely-connected NN Layer.
    model.add(tf.keras.layers.Dense(256))
    # The `Activation()` method applies an activation function to an output.
    model.add(tf.keras.layers.Activation('relu'))
    # The `Dropout()` method applies dropout to the input.
    model.add(tf.keras.layers.Dropout(0.5))
    # The `Dense()` method is just your regular densely-connected NN Layer.
    model.add(tf.keras.layers.Dense(10))
```

Vertex AI's AutoML

Vertex AI

Create dataset

Select a data type and objective

First select the type of data your dataset will contain. Then select an objective, which is the outcome that you want to achieve with the trained model. [Learn more about model types](#)

IMAGE TABULAR TEXT VIDEO

Regression/classification
Predict a target column's value.
Supports tables with hundreds of columns and millions of rows.

Forecasting PREVIEW
Predict the likelihood of certain events or demand.

This screenshot shows the Vertex AI web interface for creating a dataset. On the left, there's a sidebar with various options like Dashboard, Datasets (which is selected and highlighted in blue), Features, Labeling tasks, Notebooks, Pipelines, Training, Experiments, Models, and Marketplace. The main area is titled 'Create dataset' and has a sub-section 'Select a data type and objective'. It says to first choose the data type (IMAGE, TABULAR, TEXT, VIDEO) and then an objective (Regression/classification or Forecasting). Below these are two cards: one for 'Regression/classification' (selected) showing a scatter plot and a monitor, and one for 'Forecasting' (Preview) showing a line graph with an AI icon. A note at the top right says 'Learn more about model types'.

Select a data type and objective

First select the type of data your dataset will contain. Then select an objective, which is the outcome that you want to achieve with the trained model. [Learn more about model types](#)

IMAGE

TABULAR

TEXT

VIDEO



Regression/classification

Predict a target column's value.
Supports tables with hundreds of columns and millions of rows.



Forecasting PREVIEW

Predict the likelihood of certain events or demand.

Select a data type and objective

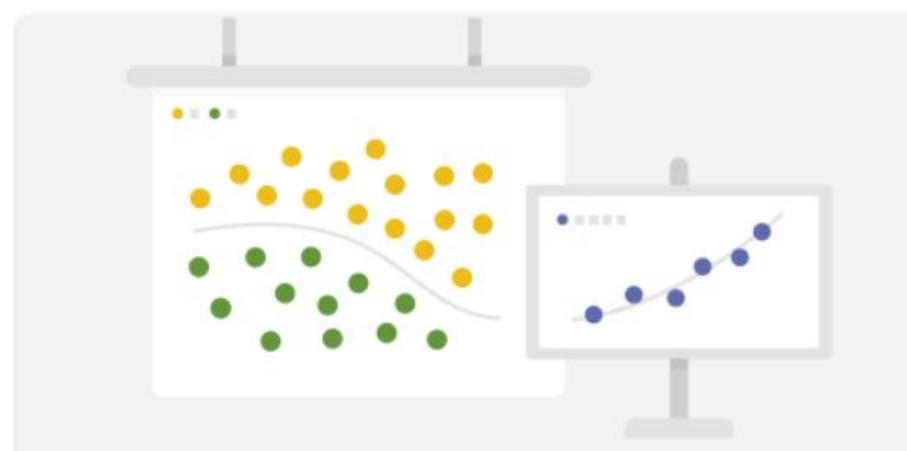
First select the type of data your dataset will contain. Then select an objective, which is the outcome that you want to achieve with the trained model. [Learn more about model types](#)

IMAGE

TABULAR

TEXT

VIDEO



Regression/classification

Predict a target column's value.
Supports tables with hundreds of columns and millions of rows.



Forecasting PREVIEW

Predict the likelihood of certain events or demand.

Select a data type and objective

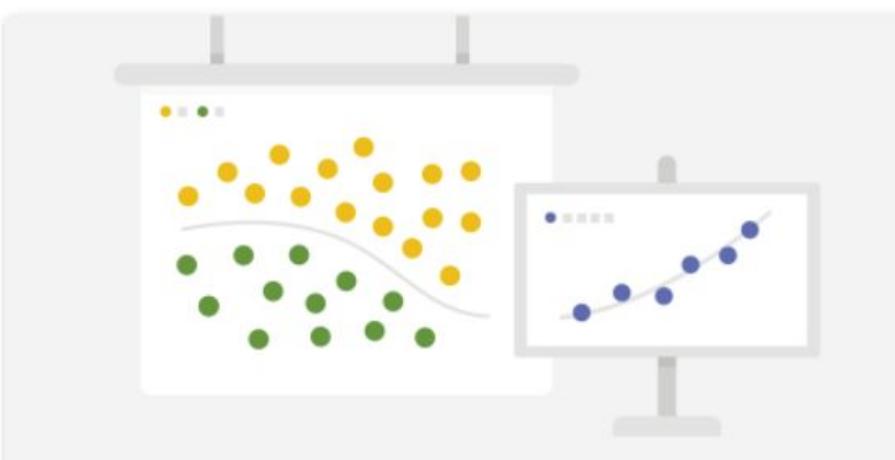
First select the type of data your dataset will contain. Then select an objective, which is the outcome that you want to achieve with the trained model. [Learn more about model types](#)

IMAGE

TABULAR

TEXT

VIDEO



Regression/classification

Predict a target column's value.
Supports tables with hundreds of columns and millions of rows.

Forecasting PREVIEW

Predict the likelihood of certain events or demand.

Datasets

Select an objective

An objective is an outcome you want to achieve with a trained model. Don't worry, you can use this dataset for other image-based objectives later.

IMAGE TABULAR TEXT VIDEO



Image classification (Single-label)
Predict the one correct label that you want assigned to an image.



Image classification (Multi-label)
Predict all the correct labels that you want assigned to an image.

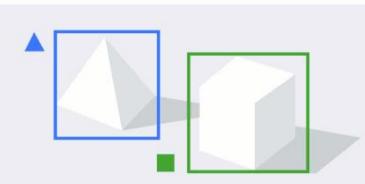


Image object detection
Predict all the locations of objects that you're interested in.

Select an objective

An objective is an outcome you want to achieve with a trained model.

IMAGE **TABULAR** TEXT VIDEO



Regression/classification
Predict a target column's value. Supports tables with hundreds of columns and millions of rows.

Select an objective

An objective is an outcome you want to achieve with a trained model. Don't worry, you can use this dataset for other text-based objectives later.

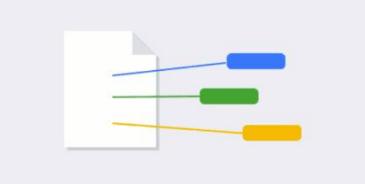
IMAGE TABULAR **TEXT** VIDEO



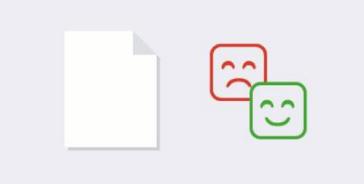
Text classification (Single-label)
Predict the one correct label that you want assigned to a document.



Text classification (Multi-label)
Predict all the correct labels that you want assigned to a document.



Text entity extraction
Identify entities within your text items.

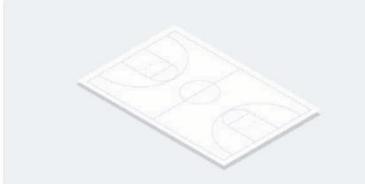


Text sentiment analysis
Understand the overall sentiment expressed in a block of text.

Select an objective

An objective is an outcome you want to achieve with a trained model. Don't worry, you can use this dataset for other video-based objectives later.

IMAGE TABULAR TEXT **VIDEO**



Video classification
Get label predictions for entire videos, shots, and frames.



Video action recognition
Identify the action moments in your videos.



Video object tracking
Get labels, tracks, and timestamps for objects you want to track in a video.

Step 1: Create a dataset

Vertex AI

← Create dataset

Dataset name * credit_risk
Can use up to 128 characters.

Select a data type and objective

First select the type of data your dataset will contain. Then select an objective, which is the outcome that you want to achieve with the trained model. [Learn more about model types](#)

IMAGE TABULAR TEXT VIDEO

Regression/classification
Predict a target column's value.
Supports tables with hundreds of columns and millions of rows.

Forecasting PREVIEW
Predict the likelihood of certain events or demand.

Dashboard Datasets Features Labeling tasks Notebooks Pipelines Training Experiments Models Endpoints Batch predictions Metadata

Step 2: Select a datatype and objective

Vertex AI

Create dataset

Select a data type and objective

First select the type of data your dataset will contain. Then select an objective, which is the outcome that you want to achieve with the trained model. [Learn more about model types](#)

IMAGE TABULAR TEXT VIDEO

Regression/classification

Predict a target column's value.
Supports tables with hundreds of columns and millions of rows.

Forecasting PREVIEW

Predict the likelihood of certain events or demand.

Dashboard

Datasets

Features

Labeling tasks

Notebooks

Pipelines

Training

Experiments

Models

Marketplace

Step 3: Upload the data

Vertex AI credit_risk SOURCE ANALYZE

Add data to your dataset

Before you begin, read the [data guide](#) to learn how to prepare your data. Then choose a data source.

Select a data source

- CSV file: Can be uploaded from your computer or on Cloud Storage. [Learn more](#)
- BigQuery: Select a table or view from BigQuery. [Learn more](#)

Upload CSV files from your computer

Select CSV files from Cloud Storage

Select a table or view from BigQuery

Upload CSV files from your computer

Add up to 500 CSV files per upload. The files will be stored in a new Cloud Storage bucket ([charges apply](#)). Data from multiple files will be referenced as one dataset.

[SELECT FILES](#)



You can build two model types with tabular data. The model type is automatically chosen based on the data type of your target column.

- Regression models predict a numeric value. For example, predicting home prices or consumer spending.
- Classification models predict a category from a fixed number of categories. Examples include predicting whether an email is spam or not, or classes a student might be interested in attending.

Step 3: Upload the data

Vertex AI credit_risk SOURCE ANALYZE

Add data to your dataset

Before you begin, read the [data guide](#) to learn how to prepare your data. Then choose a data source.

Select a data source

- CSV file: Can be uploaded from your computer or on Cloud Storage. [Learn more](#)
- BigQuery: Select a table or view from BigQuery. [Learn more](#)

Upload CSV files from your computer

Select CSV files from Cloud Storage

Select a table or view from BigQuery

Upload CSV files from your computer

Add up to 500 CSV files per upload. The files will be stored in a new Cloud Storage bucket ([charges apply](#)). Data from multiple files will be referenced as one dataset.

[SELECT FILES](#)



You can build two model types with tabular data. The model type is automatically chosen based on the data type of your target column.

- Regression models predict a numeric value. For example, predicting home prices or consumer spending.
- Classification models predict a category from a fixed number of categories. Examples include predicting whether an email is spam or not, or classes a student might be interested in attending.

Google Cloud Platform cloud-training-demos

Vertex AI consumer_spend_9.28.2021

SOURCE ANALYZE

Datasets

Features

Labeling tasks

Notebooks

Pipelines

Training

Experiments

Models

Endpoints

Batch predictions

Metadata

Dataset Info

Created: Sep 28, 2021 10:13 PM

Dataset format: CSV

Dataset location: gs://cloud-training...consumer_spend.csv

Summary

Total columns: 5

Total rows: 8,068

Filter Enter property name or value

Column name	Missing % (count)	Distinct values
Family_Size	-	10
Graduated	-	3
Profession	-	10
Spending_Score	-	3
Work_Experience	-	16

```
✓[17] print(df['Spending_Score'].value_counts())
```

Low	4878
Average	1974
High	1216

Name: Spending_Score, dtype: int64

Step 4: Train a new model

Training jobs and models

Use this dataset and annotation set to train a new machine learning model with AutoML or custom code

TRAIN NEW MODEL

Vertex AI finished training model "credit_risk_202182831655"



Inbox



Vertex AI

to me ▾

Hello Vertex AI Customer,

Vertex AI finished training model "credit_risk_202182831655".

Additional Details:

Operation State: Succeeded

Resource Name:

[projects/663413318684/locations/us-central1/trainingPipelines/4923316201041428480](https://pantheon.corp.google.com/vertex-ai/models?project=cloud-training-demos)

To continue your progress, go back to your training pipeline using

<https://pantheon.corp.google.com/vertex-ai/models?project=cloud-training-demos>

Sincerely,

The Google Cloud AI Team

 Vertex AI

Models  

	Dashboard	Models	ID	Data
	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 

Filter Enter a property name

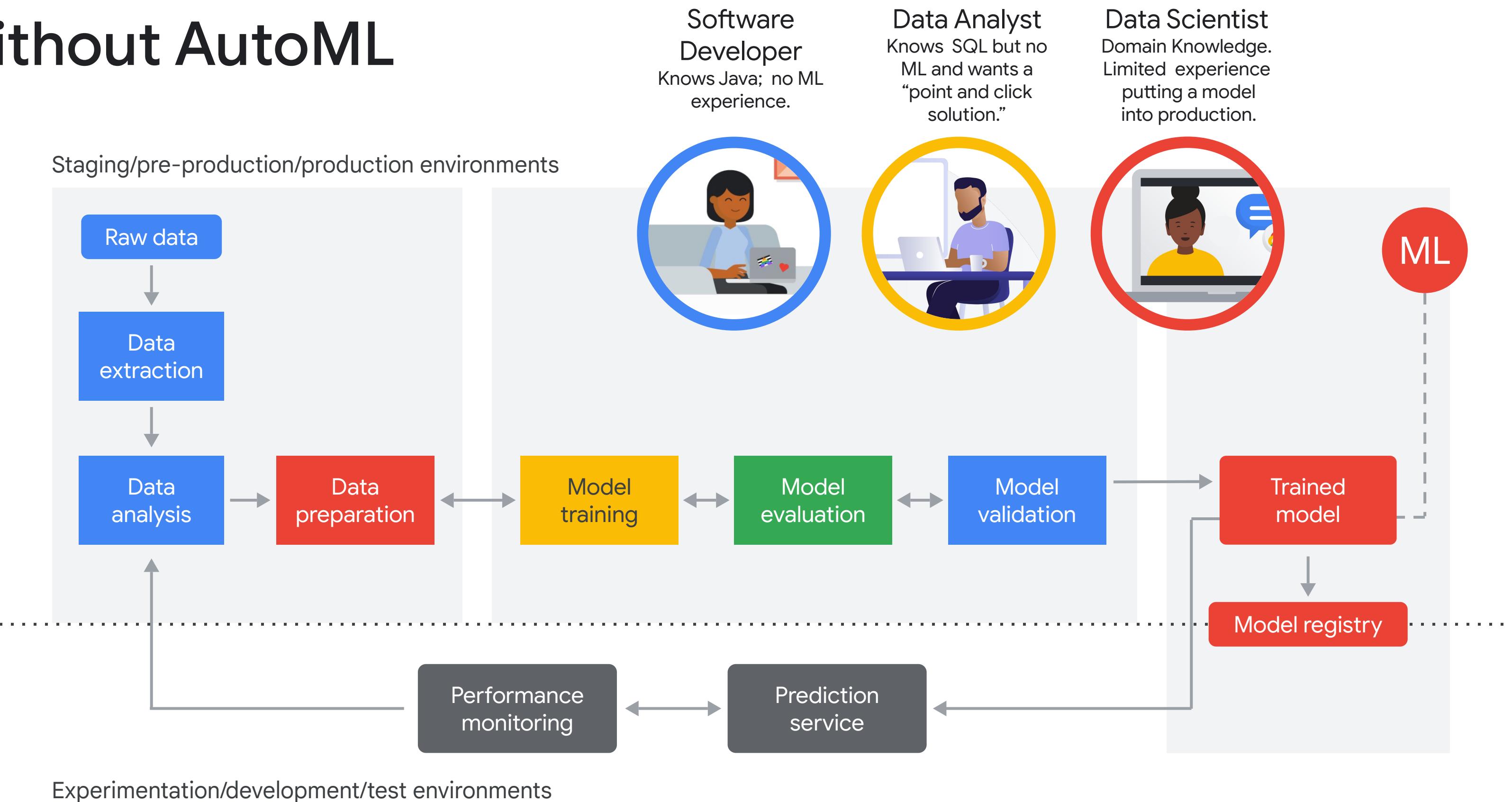
Name	ID	Data
credit_risk_202182831655	2430035046594248704	credit_risk

From uploading to training

The screenshot shows the Vertex AI interface. On the left, a sidebar lists various options: Dashboard, Datasets, Features, Labeling tasks, Notebooks, Pipelines, Training, Experiments, Models (which is highlighted with a red box and a red arrow pointing down), Endpoints, Batch predictions, and Metadata. The main area is titled 'Models' and contains a 'CREATE' button and an 'IMPORT' button. Below this is a descriptive text about building models from datasets or unmanaged data sources, mentioning 'us-central1 (Iowa)' as the region. A 'Filter' input field is present. A table lists five models with columns for Name, ID, and a checkmark icon:

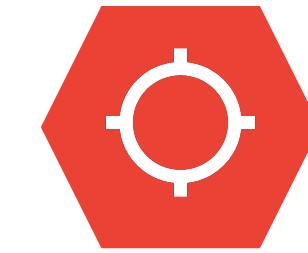
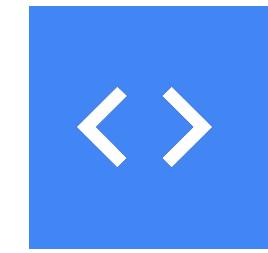
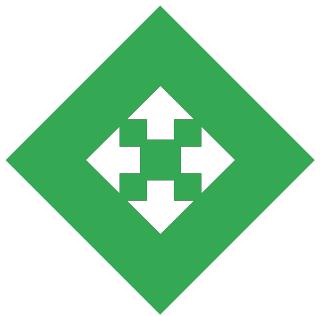
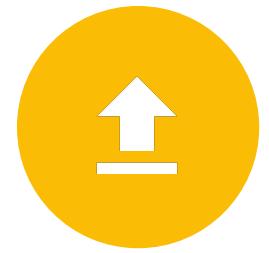
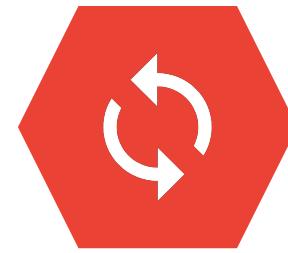
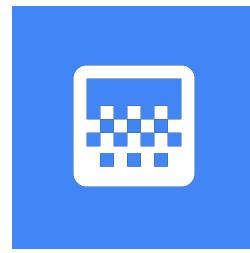
Name	ID
credit_risk_202182831655	2430035046594248704
consumer_spending_score_202182732546	2541499137371668480
chicago-taxi-tips-classifier-v01	6897412362899292160
imagedataset_1623500071212_202161213847	80510639432269824

Without AutoML



Where Vertex AI fits in the ML workflow

You can use Vertex AI to manage the following stages in the ML workflow:



Create a dataset and upload data.

Train an ML model on your data:

- Train the model
- Evaluate model accuracy
- Tune hyperparameters (custom training only)

Upload and store your model in Vertex AI.

Deploy your trained model to an endpoint for serving predictions.

Send prediction requests to your endpoint.

Specify a prediction traffic split in your endpoint.

Manage your models and endpoints.

If you want to code, you still can

Vertex AI

Create dataset

IMAGE TABULAR TEXT VIDEO

Image classification (Single-label)
Predict the one correct label that you want assigned to an image.

Image classification (Multi-label)
Predict all the correct labels that you want assigned to an image.

Image object detection
Predict all the locations of objects that you're interested in.

Image segmentation
Predict per-pixel areas of an image with a label.

Dashboard Datasets Features Labeling tasks Notebooks Pipelines Training Experiments Models Endpoints Batch predictions Marketplace

Choose a training method

Auto ML

- Create and train a model with minimal technical effort.
- Quickly prototype models or explore datasets before developing in a custom training application.

Custom training

- Create a training application optimized for your targeted outcome.
- Maintain complete control over training application functionality.
 - Target any objective, use any algorithms, develop your own loss functions or metrics, or other customizations.

When to use AutoML and when to use custom training

	AutoML	Custom training
Data science expertise needed	No.	Yes, to develop the training application and also to do some of the data preparation like feature engineering.
Programming ability needed	No, AutoML is codeless.	Yes, to develop the training application.
Time to trained model	Lower. Less data preparation is required, and no development is needed.	Higher. More data preparation is required, and training application development is needed.
Limits on machine learning objectives	Yes. You must target one of AutoML's predefined objectives.	No.
Can manually optimize model performance with hyperparameter tuning	No. AutoML does some automated hyperparameter tuning, but you can't modify the values used.	Yes. You can tune the model during each training run for experimentation and comparison.

When to use AutoML and when to use custom training

	AutoML	Custom training
Data science expertise needed	No.	Yes, to develop the training application and also to do some of the data preparation like feature engineering.
Programming ability needed	No, AutoML is codeless.	Yes, to develop the training application.
Time to trained model	Lower. Less data preparation is required, and no development is needed.	Higher. More data preparation is required, and training application development is needed.
Limits on machine learning objectives	Yes. You must target one of AutoML's predefined objectives.	No.
Can manually optimize model performance with hyperparameter tuning	No. AutoML does some automated hyperparameter tuning, but you can't modify the values used.	Yes. You can tune the model during each training run for experimentation and comparison.

When to use AutoML and when to use custom training

	AutoML	Custom training
Data science expertise needed	No.	Yes, to develop the training application and also to do some of the data preparation like feature engineering.
Programming ability needed	No, AutoML is codeless.	Yes, to develop the training application.
Time to trained model	Lower. Less data preparation is required, and no development is needed.	Higher. More data preparation is required, and training application development is needed.
Limits on machine learning objectives	Yes. You must target one of AutoML's predefined objectives.	No.
Can manually optimize model performance with hyperparameter tuning	No. AutoML does some automated hyperparameter tuning, but you can't modify the values used.	Yes. You can tune the model during each training run for experimentation and comparison.

When to use AutoML and when to use custom training

	AutoML	Custom training
Data science expertise needed	No.	Yes, to develop the training application and also to do some of the data preparation like feature engineering.
Programming ability needed	No, AutoML is codeless.	Yes, to develop the training application.
Time to trained model	Lower. Less data preparation is required, and no development is needed.	Higher. More data preparation is required, and training application development is needed.
Limits on machine learning objectives	Yes. You must target one of AutoML's predefined objectives.	No.
Can manually optimize model performance with hyperparameter tuning	No. AutoML does some automated hyperparameter tuning, but you can't modify the values used.	Yes. You can tune the model during each training run for experimentation and comparison.

When to use AutoML and when to use custom training

	AutoML	Custom training
Data science expertise needed	No.	Yes, to develop the training application and also to do some of the data preparation like feature engineering.
Programming ability needed	No, AutoML is codeless.	Yes, to develop the training application.
Time to trained model	Lower. Less data preparation is required, and no development is needed.	Higher. More data preparation is required, and training application development is needed.
Limits on machine learning objectives	Yes. You must target one of AutoML's predefined objectives.	No.
Can manually optimize model performance with hyperparameter tuning	No. AutoML does some automated hyperparameter tuning, but you can't modify the values used.	Yes. You can tune the model during each training run for experimentation and comparison.

When to use AutoML and when to use custom training

	AutoML	Custom training
Data science expertise needed	No.	Yes, to develop the training application and also to do some of the data preparation like feature engineering.
Programming ability needed	No, AutoML is codeless.	Yes, to develop the training application.
Time to trained model	Lower. Less data preparation is required, and no development is needed.	Higher. More data preparation is required, and training application development is needed.
Limits on machine learning objectives	Yes. You must target one of AutoML's predefined objectives.	No.
Can manually optimize model performance with hyperparameter tuning	No. AutoML does some automated hyperparameter tuning, but you can't modify the values used.	Yes. You can tune the model during each training run for experimentation and comparison.

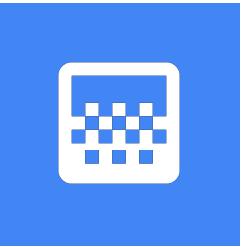
When to use AutoML and when to use custom training

	AutoML	Custom training
Can control aspects of the training environment	Limited. For image and tabular datasets, you can specify the number of node hours to train for, and whether to allow early stopping of training.	Yes. You can specify aspects of the environment such as Compute Engine machine type, disk size, machine learning framework, and number of nodes.
Limits on data size	Yes. AutoML uses managed datasets; data size limitations vary depending on the type of dataset.	For unmanaged datasets, no. Managed datasets have the same limits as Vertex AI datasets that are used to train AutoML models.

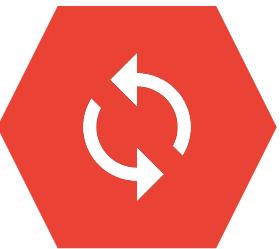
When to use AutoML and when to use custom training

	AutoML	Custom training
Can control aspects of the training environment	Limited. For image and tabular datasets, you can specify the number of node hours to train for, and whether to allow early stopping of training.	Yes. You can specify aspects of the environment such as Compute Engine machine type, disk size, machine learning framework, and number of nodes.
Limits on data size	Yes. AutoML uses managed datasets; data size limitations vary depending on the type of dataset.	For unmanaged datasets, no. Managed datasets have the same limits as Vertex AI datasets that are used to train AutoML models.

Vertex AI

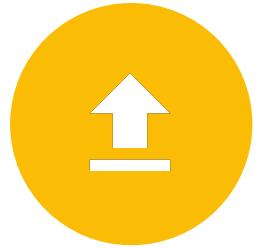


Create a dataset and upload data.

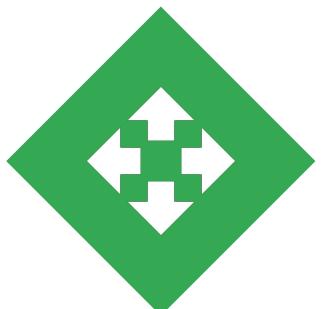


Train an ML model on your data:

- Train the model
- Evaluate model accuracy
- Tune hyperparameters (custom training only)



Upload and store your model in Vertex AI.



Deploy your trained model to an endpoint for serving predictions.

...and more!

AutoML models

To learn about evaluation metrics for unstructured data such as image, text, or video, please see the Vertex AI documentation.

Vertex AI documentation

Overview

Training and tutorials

Use cases

Code samples

Vertex AI brings AutoML and AI Platform together into a unified API, client library, and user interface. With Vertex AI, both AutoML training and *custom training* are available options. Whichever option you choose for training, you can save models, deploy models and request predictions with Vertex AI. [Learn more](#).

Tutorials	Get started	Resources
Hello image data	Introduction to Vertex AI	Pricing
Hello text data	Vertex AI for AI Platform users	Release notes
Hello video data	Vertex AI for AutoML users	Get support
Hello tabular data	Migrating to Vertex AI	
Hello custom training		
More →	More →	More →

Model evaluation metrics provide quantitative measurements of how your model performed on the test set.

How AutoML Tables uses your dataset

Data split

Random assignment

80% of your data is randomly assigned for training, 10% for validation and 10% for testing.



Manual

You assign each data row for training, validation, and testing. [Learn more](#)

Chronological assignment

The earliest 80% of your data is assigned to training, the next 10% for validation and the latest 10% for testing. This option requires a Time column in your dataset. [Learn more](#)



Training 80%



Validation 10%



Testing 10%



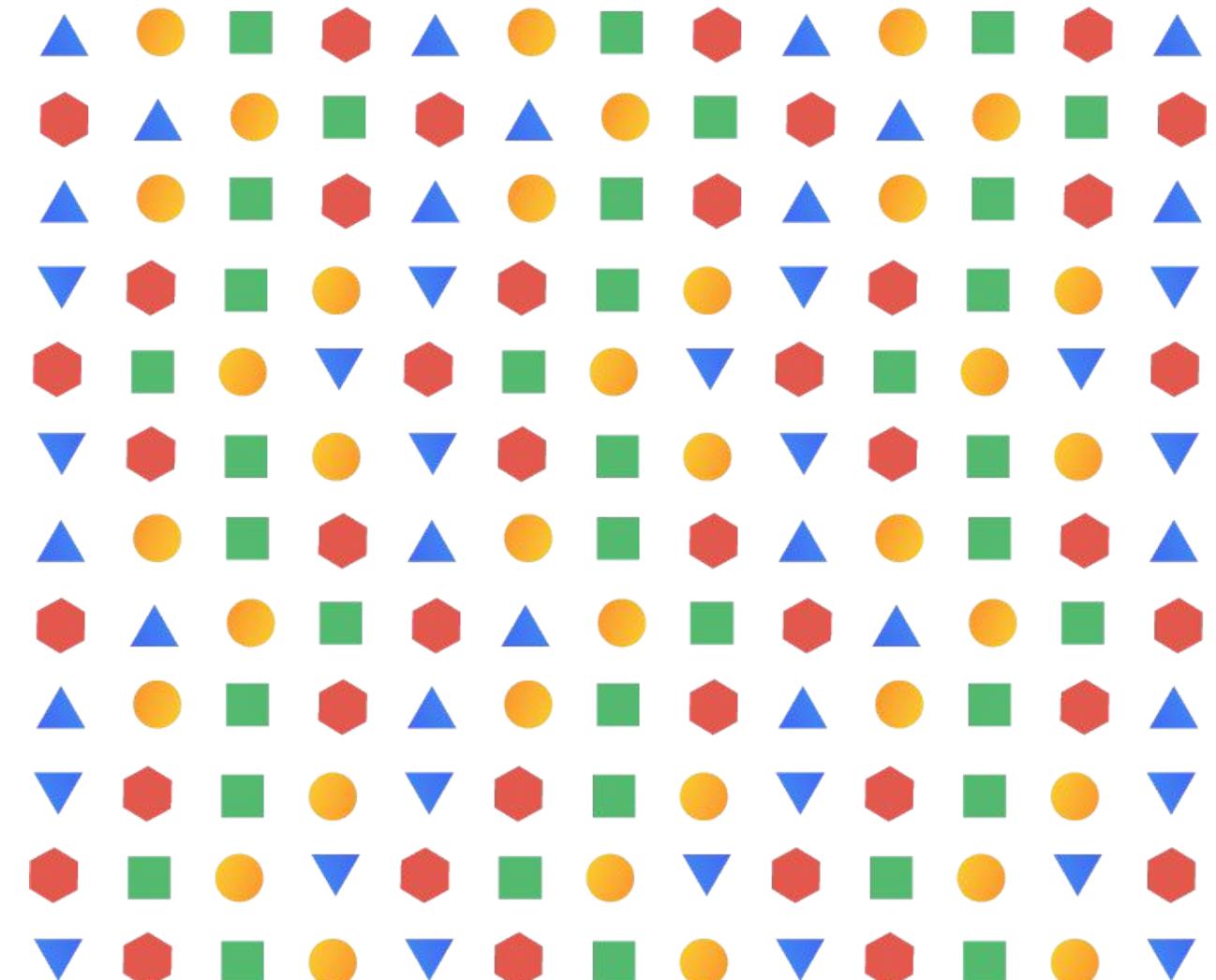
Training set

- The vast majority of your data should be here.
- This is the data your model “sees.”
- The model uses it to learn its parameters.



Validation set

- The validation set is sometimes referred to as the “dev” set, and is also used in the training process.
- It is used to tune the hyperparameters (variables that specify the model’s structure) after the model uses the training data in each iteration of the training process.
- Using the validation set to tune the hyperparameters means the model will generalize better.

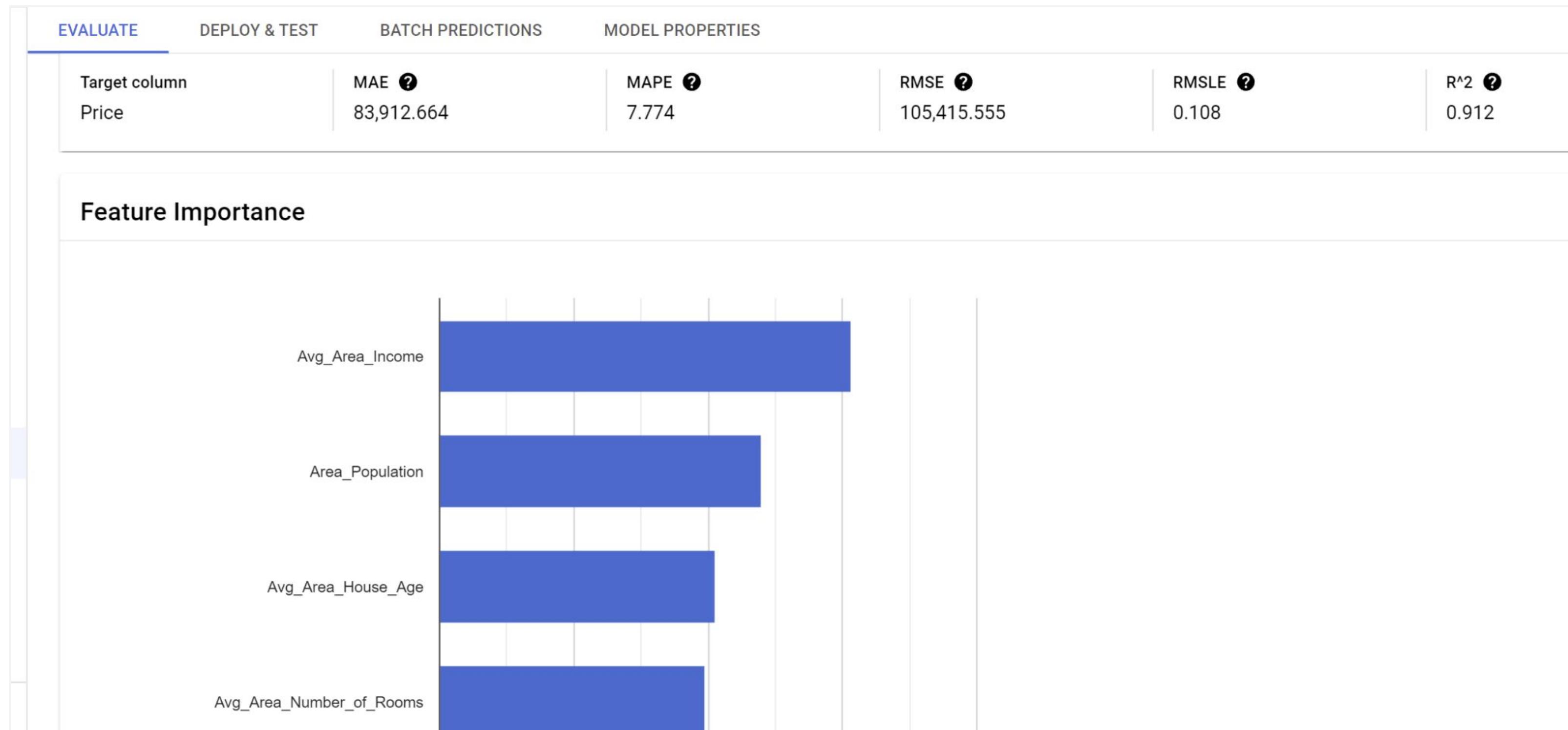


Test set

- The test set is used after the training process is complete.
- AutoML Tables uses it as a new challenge for the model to get a strong indication of how the model will perform on real-world data.



Linear regression models



Mean absolute error (MAE)

EVALUATE	DEPLOY & TEST	BATCH PREDICTIONS	MODEL PROPERTIES		
Target column sales		MAE ? 0.505	MAPE ? 3.424	RMSE ? 0.98	RMSLE ? 0.053

MAE is the average magnitude of errors: the differences between a target and predicted value.

Uses absolute value, so it doesn't indicate over- or under-performance.

A low value indicates a higher-quality model, where 0 means the model made no errors.

Mean absolute percentage error (MAPE)

EVALUATE	DEPLOY & TEST	BATCH PREDICTIONS	MODEL PROPERTIES		
Target column sales	MAE ? 0.505	MAPE ? 3.424	RMSE ? 0.98	RMSLE ? 0.053	R ² ? 0.976

MAPE is the average absolute percentage difference between labels and predicted values.

This metric ranges between zero and infinity where a lower value indicates a higher-quality model.

If the target column contains 0 values, MAPE is undefined.

Root mean square error (RMSE)

EVALUATE	DEPLOY & TEST	BATCH PREDICTIONS	MODEL PROPERTIES		
Target column sales	MAE ? 0.505	MAPE ? 3.424	RMSE ? 0.98	RMSLE ? 0.053	R ² ? 0.976

RMSE is the root of squared differences between observed and predicted values.

More sensitive to outliers than MAE, so it is more useful if you're concerned about large errors.

Like MAE, a lower value indicates a higher-quality model.

Root mean squared logarithmic error metric (RMSLE)

EVALUATE	DEPLOY & TEST	BATCH PREDICTIONS	MODEL PROPERTIES		
Target column sales	MAE ? 0.505	MAPE ? 3.424	RMSE ? 0.98	RMSLE ? 0.053	R ² ? 0.976

RMSLE penalizes under-prediction more than over-prediction.

Is a good metric to avoid penalizing differences for large prediction values more heavily than for small prediction values.

A lower value indicates a higher-quality model.

Metrics is returned only if all values are non-negative.

R squared (R^2)

EVALUATE	DEPLOY & TEST	BATCH PREDICTIONS	MODEL PROPERTIES	
Target column sales	MAE ? 0.505	MAPE ? 3.424	RMSE ? 0.98	RMSLE ? 0.053

R squared (R^2) is the square of the Pearson correlation coefficient between the observed and predicted values.

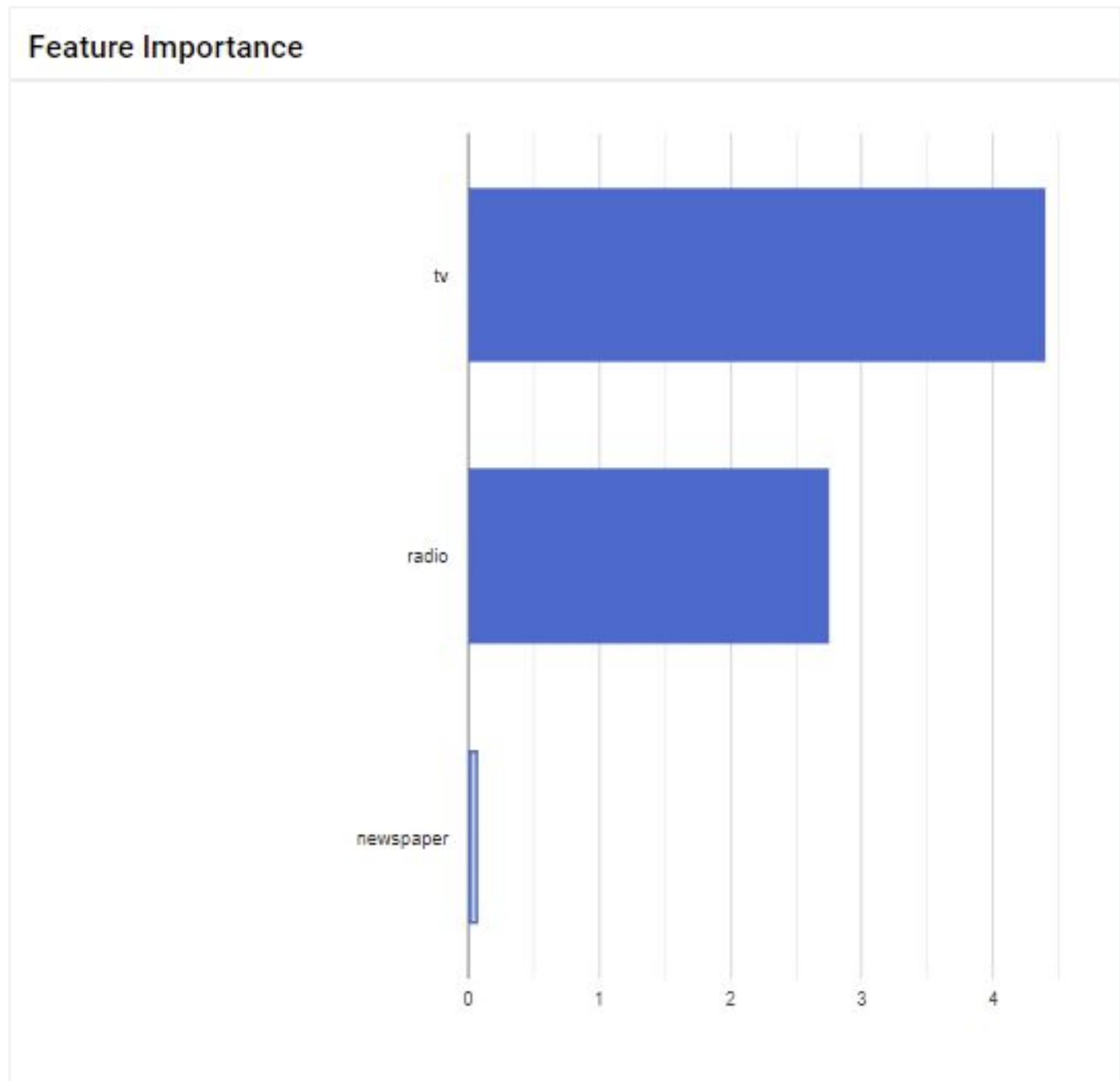
The (R^2) value ranges from 0 to 1.

A higher value indicates a higher-quality model.

Feature importance

Vertex AI shows you how much each feature impacts a model.

Values are provided as a percentage for each feature where the higher the percentage, the more strongly that feature impacted the model training.



Deploy and test

EVALUATE **DEPLOY & TEST** BATCH PREDICTIONS MODEL PROPERTIES

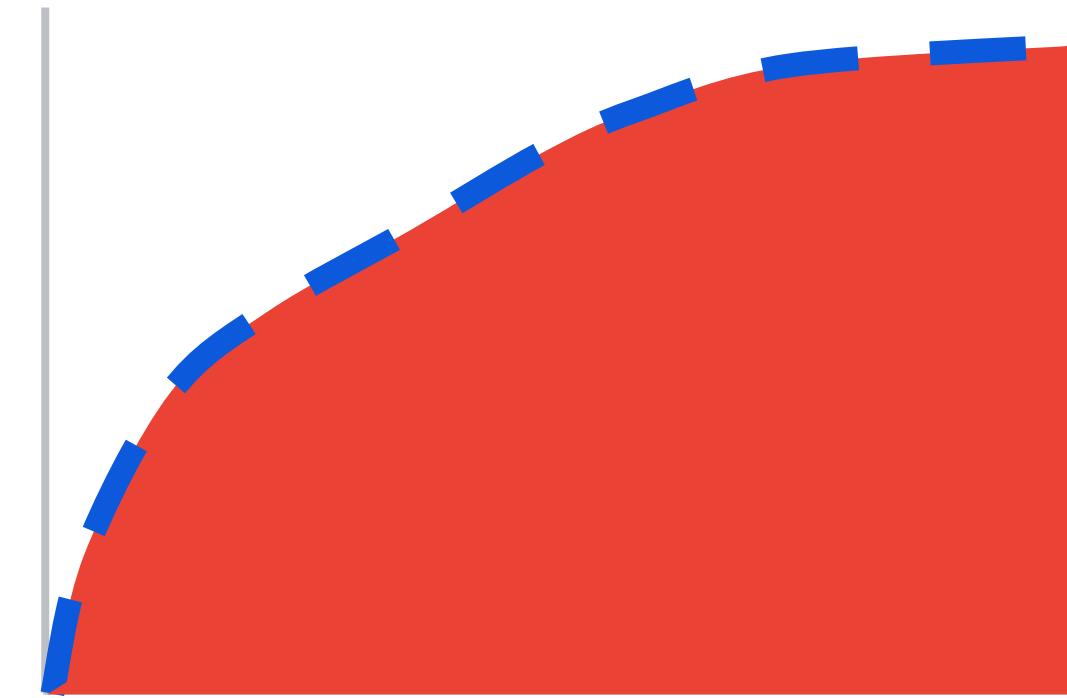
Test your model [PREVIEW](#)

Feature column name	Type	Required or optional	Value	Local feature importance	Predicted column not yet known
age	Text	Required	40.707912	--	Prediction result
clientid	Text	Required	993.000000	--	--
income	Text	Required	45165.925955	--	--
loan	Text	Required	3872.402468	--	--

PREDICT **RESET**

AUC

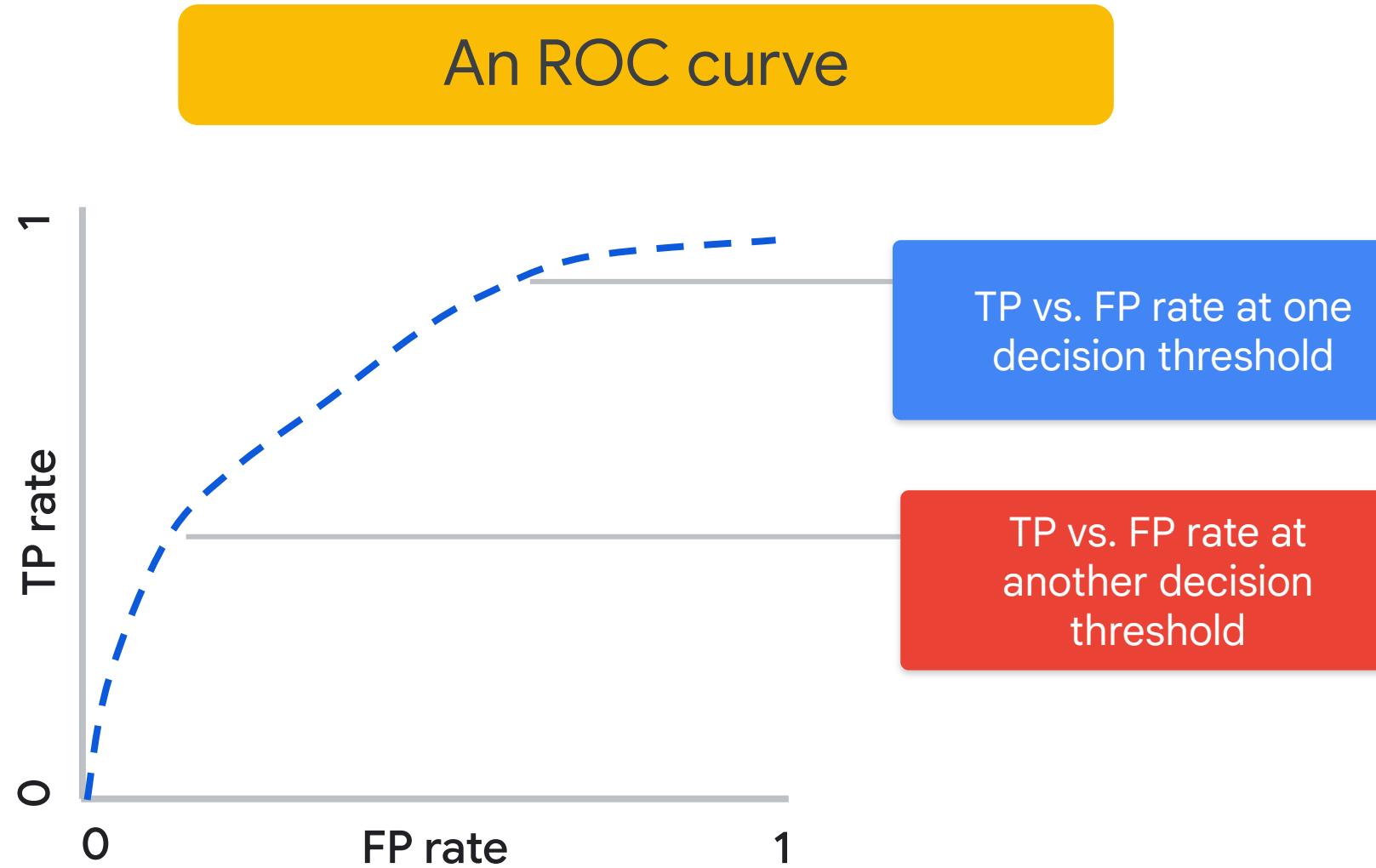
- AUC = area under the ROC curve.
- Interpretation: If we pick a random positive and a random negative, what's the probability that my model scores them in the correct relative order?
- Intuition: Gives an aggregate measure of performance aggregated across all possible classification thresholds.



Confidence threshold 0.5

All labels	
PR AUC	0.696
ROC AUC	0.81
Log loss	0.815
F1 score	0.5686126
Precision	60.4%
Recall	53.7%
Created	Sep 29, 2021 12:52:20 AM

ROC curve



Confidence threshold

All labels

PR AUC 0.696

ROC AUC 0.81

Log loss 0.815

F1 score 0.5686126

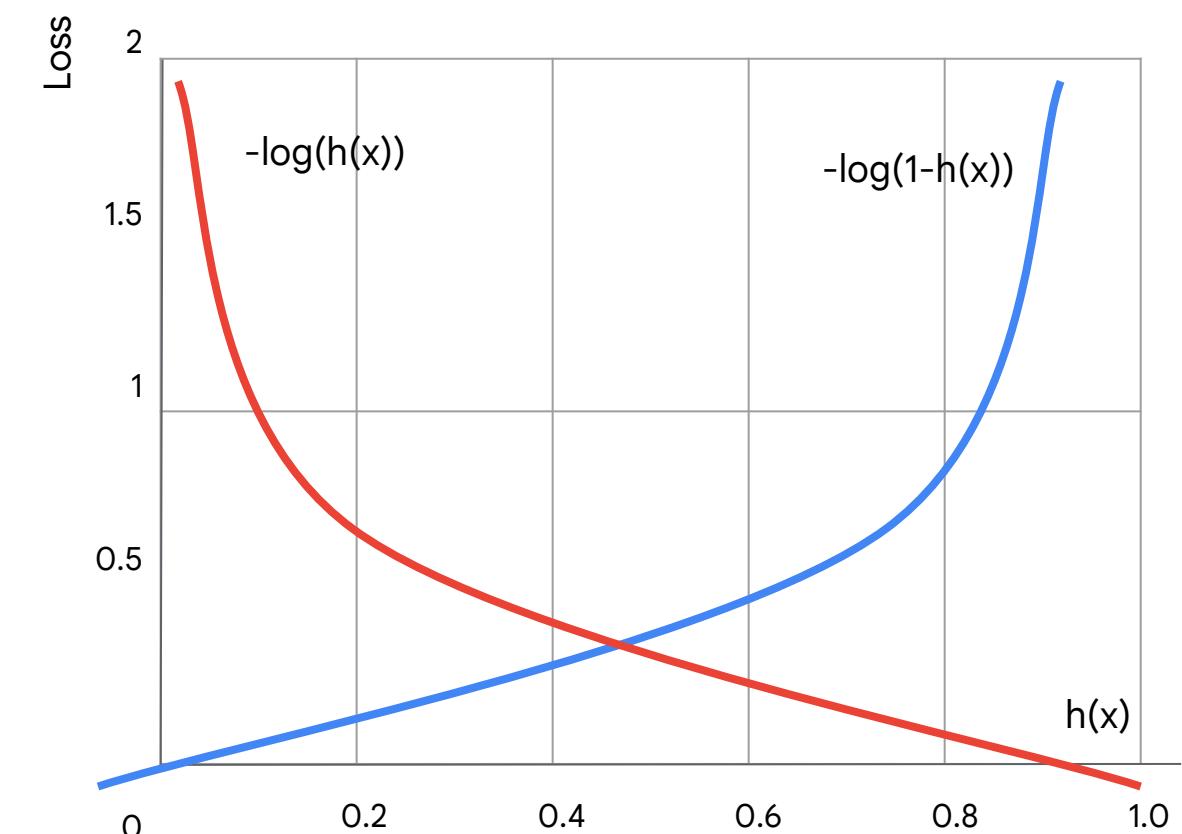
Precision 60.4%

Recall 53.7%

Created Sep 29, 2021 12:52:20 AM

Log loss

- The red line represents 1 class.
 - When the predicted probability (x axis) is close to 1, the loss is less.
 - When the predicted probability is close to 0, loss approaches infinity.
- The blue line represents 0 class.
 - When the predicted probability (x-axis), is close to 0, the loss is less.
 - When the predicted probability is close to 1, loss approaches infinity.

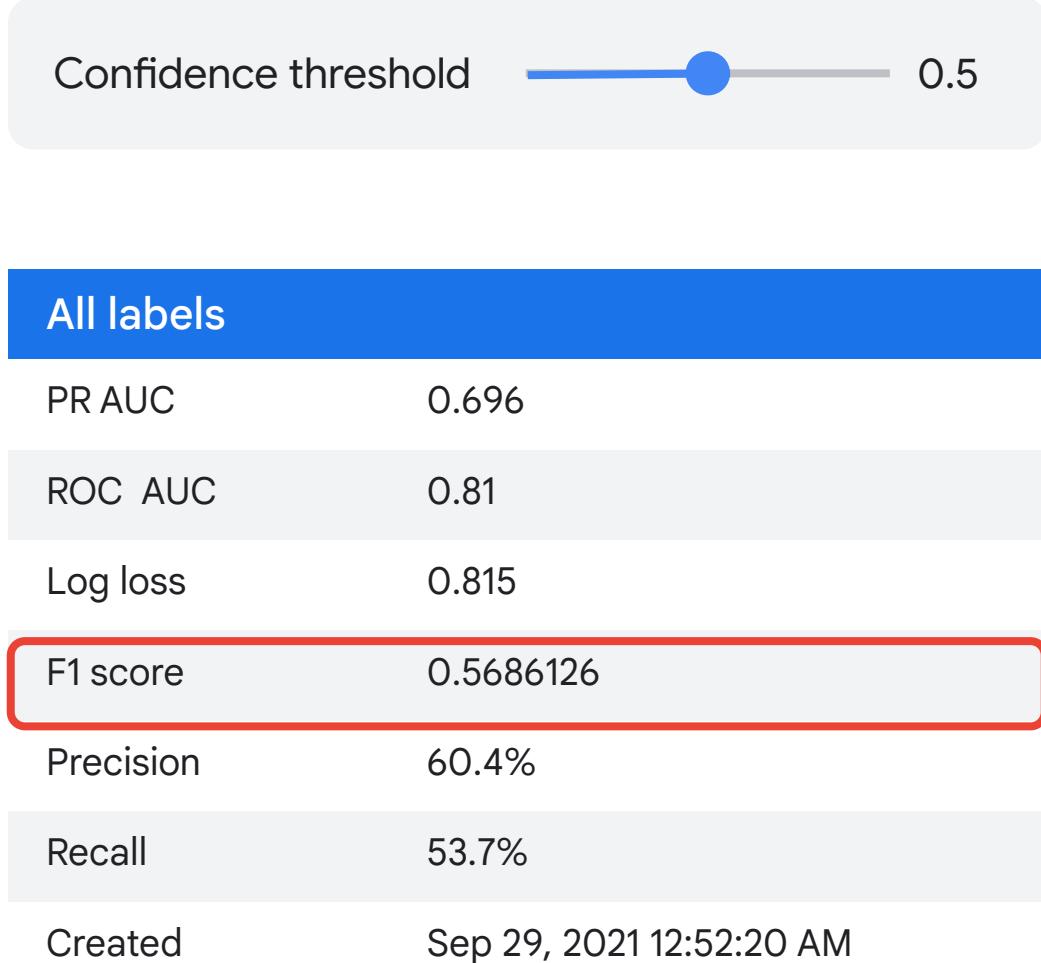


Confidence threshold 0.5

All labels	
PR AUC	0.696
ROC AUC	0.81
Log loss	0.815
F1 score	0.5686126
Precision	60.4%
Recall	53.7%
Created	Sep 29, 2021 12:52:20 AM

F1 score

- Precision answers the question:
 - Out of the equipment classified “will fail,” what fraction was correct?
- Recall answers the question:
 - Out of the equipment that actually failed, what fraction did the classifier pick up?
- F1 score is the harmonic mean of precision and recall.



Classification evaluation metrics: Precision

Precision attempts to answer the following question:

What proportion of positive identifications was actually correct?

Precision is defined as follows:

$$\text{Precision} = \frac{TP}{TP + FP}$$

Confidence threshold  0.5

All labels	
PR AUC	0.696
ROC AUC	0.81
Log loss	0.815
F1 score	0.5686126
Precision	60.4%
Recall	53.7%
Created	Sep 29, 2021 12:52:20 AM

Classification evaluation metrics: Recall

Recall attempts to answer the following question:

What proportion of actual positives was identified correctly?

Mathematically, recall is defined as follows:

$$\text{Recall} = \frac{TP}{TP + FN}$$

Confidence threshold  0.5

All labels	
PR AUC	0.696
ROC AUC	0.81
Log loss	0.815
F1 score	0.5686126
Precision	60.4%
Recall	53.7%
Created	Sep 29, 2021 12:52:20 AM

Note: A model that produces no false negative has a recall of 1.0

Confusion matrix

	Billing issues	Deposit issues	Disconnection issues	Service issues
Billing issues	73%	9%	12%	6%
Deposit issues	-	100%	-	-
Disconnection issues	6%	2%	82%	10%
Service issues	21%	14%	2%	63%

When the values on the diagonal are high, the classifier is correctly identifying categories to the extent indicated.

	Billing issues	Deposit issues	Disconnection issues	Service issues
Billing issues	12%	9%	72%	6%
Deposit issues	-	100%	-	-
Disconnection issues	82%	2%	6%	10%
Service issues	21%	63%	2%	14%

When the values outside the diagonal are higher, the classifier is misidentifying categories to the extent indicated.

List model evaluations

Once you have trained a model, you can list evaluation metrics for that model.



Note: You can also filter when you list resources, operations, and metrics. For information on filtering, see [Filtering when listing](#).

Web UI REST & CMD LINE Go Java Node.js Python Additional languages

1. Open the [AutoML Vision UI](#) and click the **Models** tab (with lightbulb icon) in the left navigation bar to display the available models.
To view the models for a different project, select the project from the drop-down list in the upper right of the title bar.
2. Click the row for the model you want to evaluate.
3. If necessary, click the **Evaluate** tab just below the title bar.
If training has been completed for the model, AutoML Vision shows its evaluation metrics.

IMPORT IMAGES TRAIN EVALUATE TEST & USE Single-Label Classification

Model: cloud_model Confidence threshold: 0.5

All labels

label	Count
daisy	3,299
dandelion	367
roses	96.99%
sunflowers	96.46%
tulips	

All labels

Total images: 3,299
Test items: 367
Precision: 96.99%
Recall: 96.46%

Use the slider to see which confidence threshold works best for your model on the precision-recall tradeoff curve.
[Learn more about these metrics and graphs.](#)

Precision

Recall

Confidence

Recall Precision

Confusion matrix

[Web UI](#)[REST & CMD LINE](#)[Go](#)[Java](#)[Node.js](#)[Python](#)[Additional languages](#)

Before using any of the request data, make the following replacements:

- ***project-id***: your GCP project ID.
- ***model-id***: the ID of your model, from the response when you created the model. The ID is the last element of the name of your model. For example:
 - model name: `projects/project-id/locations/Location-id/models/IOD4412217016962778756`
 - model id: `IOD4412217016962778756`
- ***model-evaluation-id***: the ID value of the model evaluation. You can get model evaluation IDs from the `list` model evaluations operation.

HTTP method and URL:

```
GET https://automl.googleapis.com/v1/projects/project-id/locations/us-central1/models/model
```



[Web UI](#)[REST & CMD LINE](#)[Go](#)[Java](#)[Node.js](#)[Python](#)[Additional languages](#)

Before trying this sample, follow the setup instructions for this language on the [Client Libraries](#) page.

[View on GitHub](#)[Feedback](#)

```
from google.cloud import automl

# TODO(developer): Uncomment and set the following variables
# project_id = "YOUR_PROJECT_ID"
# model_id = "YOUR_MODEL_ID"

client = automl.AutoMlClient()
# Get the full path of the model.
model_full_id = client.model_path(project_id, "us-central1", model_id)

print("List of model evaluations:")
for evaluation in client.list_model_evaluations(parent=model_full_id, filter=""):
    print("Model evaluation name: {}".format(evaluation.name))
    print("Model annotation spec id: {}".format(evaluation.annotation_spec_id))
    print("Create Time: {}".format(evaluation.create_time))
    print("Evaluation example count: {}".format(evaluation.evaluated_example_count))
    print(
        "Classification model evaluation metrics: {}".format(
            evaluation.classification_evaluation_metrics
```

Test your model

- After evaluating your model metrics, you can test your model with new data.
- See if the model's predictions match your expectations.
- If not, you may need to continue improving your model's performance.



Deploy your model

Deploy your model

Endpoints are machine learning models made available for online prediction requests. Endpoints are useful for timely predictions from many users (for example, in response to an application request). You can also request batch predictions if you don't need immediate results.

DEPLOY TO ENDPOINT

Name	ID	Models	Region	Monitoring	Most recent monitoring job
<input checked="" type="checkbox"/> my_usahousing_10.02.2021	563495311188688896	1	us-central1	Disabled	—

Test your model PREVIEW

Feature column name	Type	Required or optional	Value	Local feature importance
Avg_Area_Income	Text	Required	68814.92560741428	--
Avg_Area_House_Age	Text	Required	5.973219004488523	--
Avg_Area_Number_of_Rooms	Text	Required	7.003715444800074	--
Avg_Area_Number_of_Bedrooms	Text	Required	4.05	--
Area_Population	Text	Required	36205.14862834159	--
Address	Text	Required	Suite	--

PREDICT

RESET

Deploy your model and make online predictions

Batch prediction

- Allows you to make many prediction requests at once.
- Is asynchronous (the model won't return a CSV file or BigQuery Table until it processes all prediction requests).

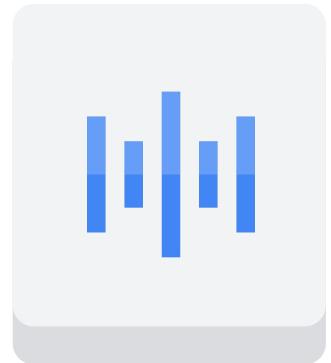
Online prediction

- Deploy your model to make it available for prediction requests using a REST API.
- Is synchronous (the model will quickly return a prediction, but only accepts one prediction request per API call).
- This is useful if parts of your system are dependent on a quick prediction turnaround.

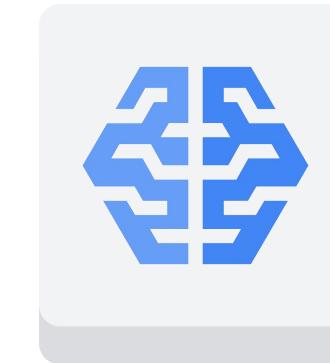
ML/AI solutions



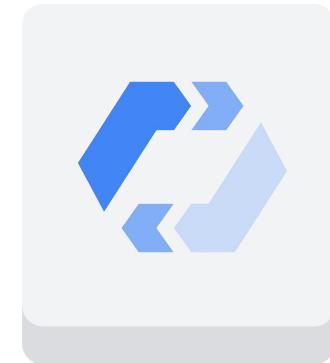
Vision
API



Speech
API



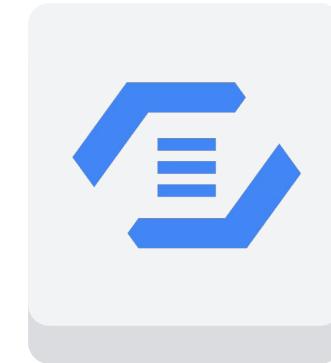
Vertex AI



AutoML
(Fast)



Translation
API

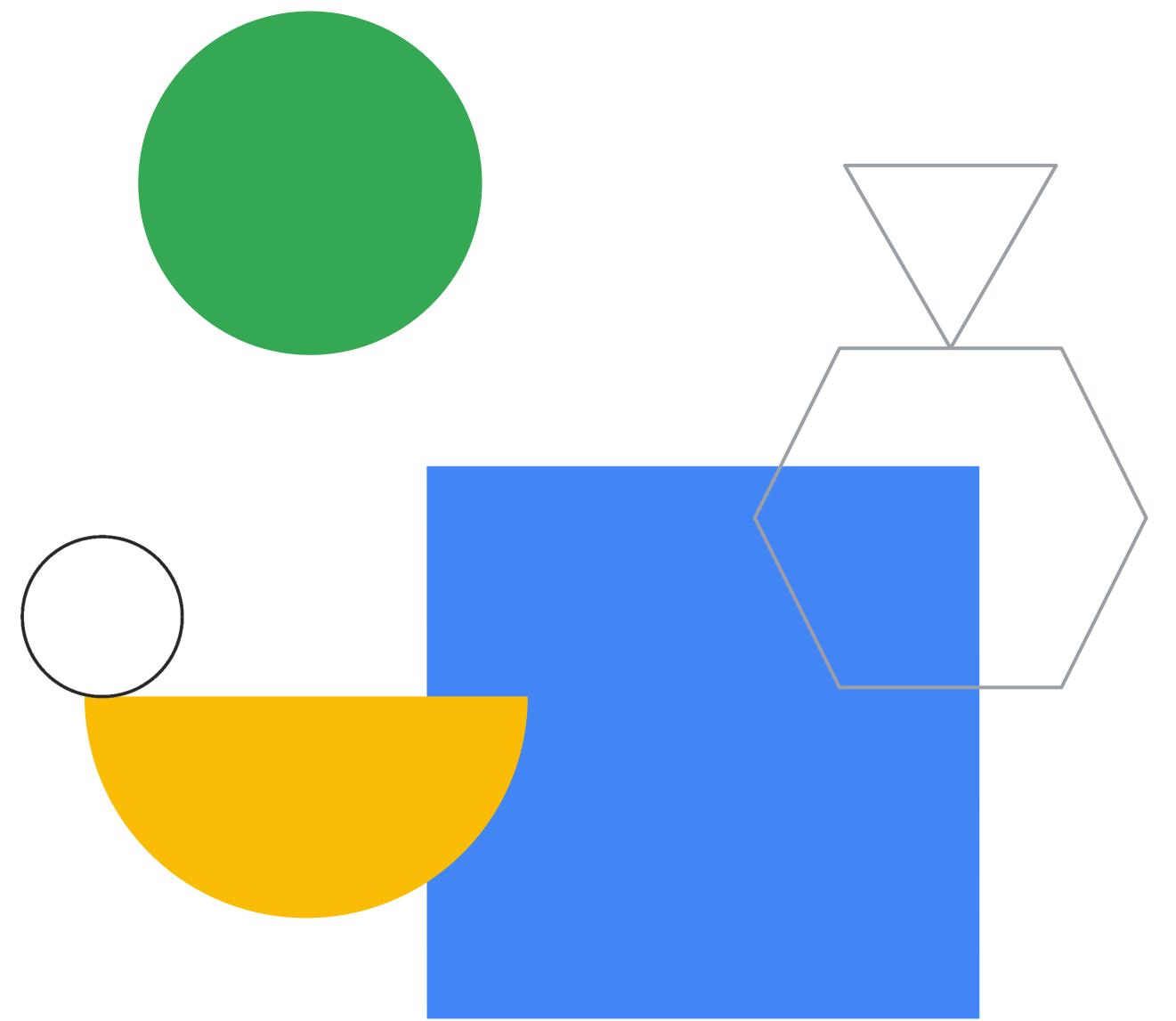


Natural
Language
API

Pre-trained models

It is very common to enrich your data with pre-trained models, to take advantage of unstructured data.

...and more!



**BigQuery Machine
Learning: Develop ML
models where your
data lives**

In this module, you learn to ...

01

Describe BigQuery ML

02

Understand how BigQuery ML supports machine learning models

03

Describe BigQuery ML hyperparameter tuning

04

Explain how to build a recommendation system with BigQuery ML



Business use case

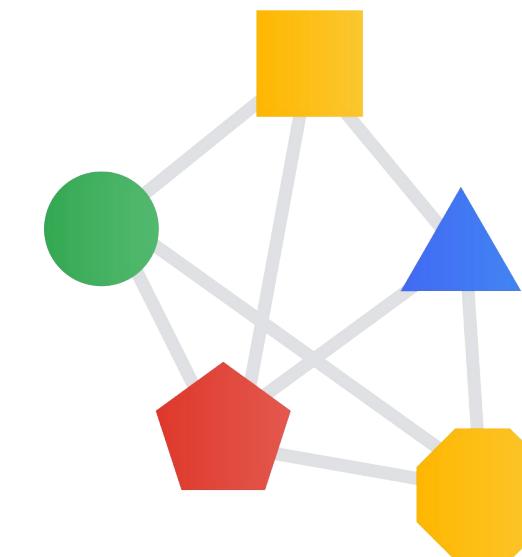
Our team at XYZ company has just defined a business use case. They have established the success criteria and want to deliver a ML model to production. It will be their second project.



Define business use case



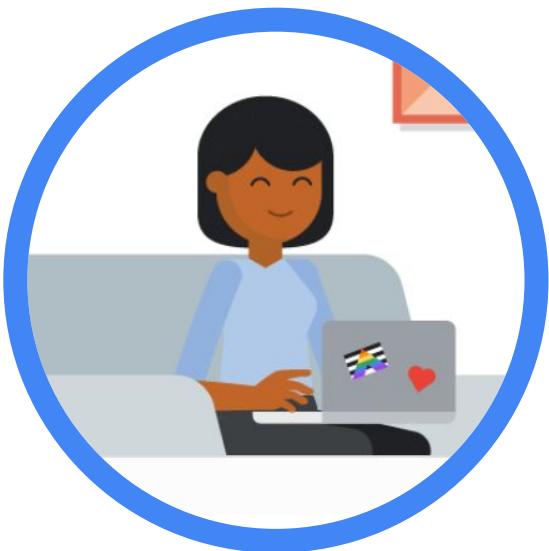
Establish success criteria



Deliver ML model
to production

Business use case

Goal is to predict visitor purchases.



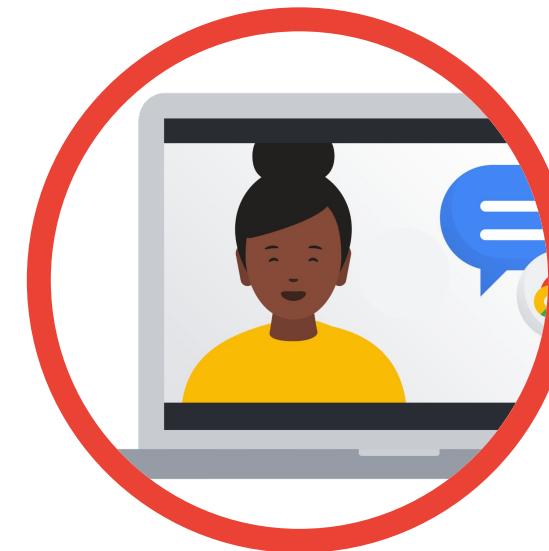
Software Developer

Knows Java



Data Analyst

Knows SQL but has no ML knowledge



Data Scientist

Has domain knowledge but has limited experience putting a machine learning model into production

Business use case

Vertex AI

Create dataset

Dataset name * my_managed_dataset

Can use up to 128 characters.

Select a data type and objective

First select the type of data your dataset will contain. Then select an objective, which is the outcome that you want to achieve with the trained model. [Learn more about model types](#)

IMAGE TABULAR TEXT VIDEO

Regression/classification

Predict a target column's value. Supports tables with hundreds of columns and millions of rows.

Forecasting **PREVIEW**

Predict the likelihood of certain events or demand.

Region us-central1 (Iowa) ?

ADVANCED OPTIONS

CREATE CANCEL

Marketplace

Business use case

The team wants the flexibility of a custom model but not too much coding.

Vertex AI

← Create dataset

Select a data type and objective

First select the type of data your dataset will contain. Then select an objective, which is the outcome that you want to achieve with the trained model. [Learn more about model types](#)

IMAGE TABULAR TEXT VIDEO

Notebooks

Pipelines

Training

Experiments

Models

Marketplace

Datasets

Regression/classification

Predict a target column's value. Supports tables with hundreds of columns and millions of rows.

Forecasting PREVIEW

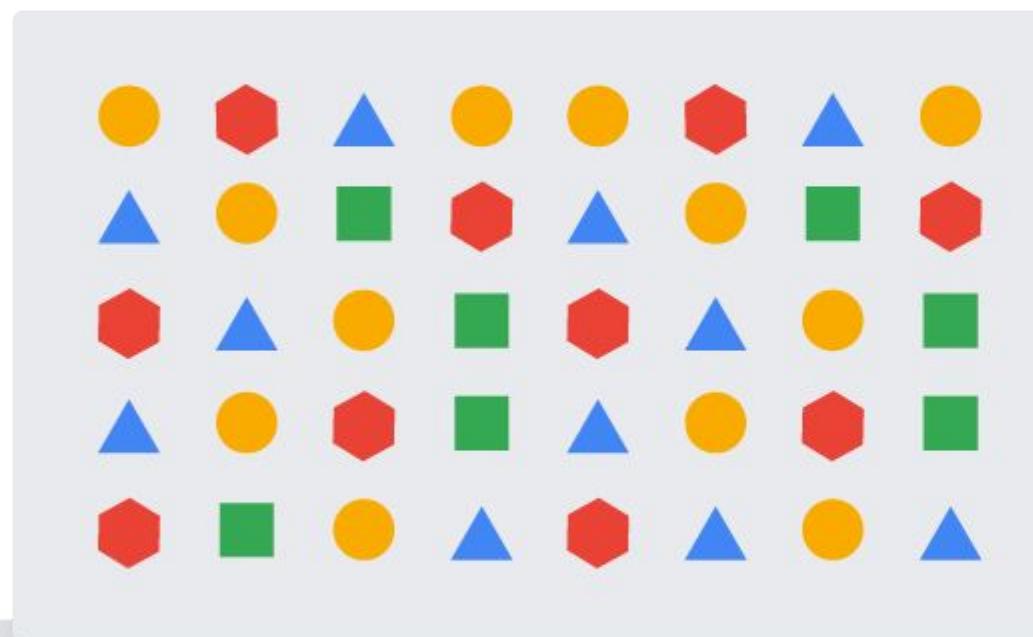
Predict the likelihood of certain events or demand.

Which framework should they use?

AutoML or BigQuery ML

XYZ team's data requirements

XYZ team data



Structured

>100 GB

900 rows

Vertex AI AutoML tabular data requirements

Required

- Columns: 2 (target and input feature)
- Target column must be categorical (2-500 values) or numerical
- Rows: 1,000

Maximum

- Dataset size: 100 GB
- Columns: 1,000
- Rows: 100 million

Vertex AI AutoML tabular data requirements

Required

- Columns: 2 (target and input feature)
- Target column must be categorical (2-500 values) or numerical
- Rows: 1,000

Maximum

- Dataset size: **100 GB**
- Columns: 1,000
- Rows: 100 million

Vertex AI AutoML tabular data requirements

Required

- Columns: 2 (target and input feature)
- Target column must be categorical (2-500 values) or numerical
- Rows: 1,000

Maximum

- Dataset size: 100 GB
- Columns: 1,000
- Rows: 100 million

Vertex AI AutoML tabular data requirements

Required

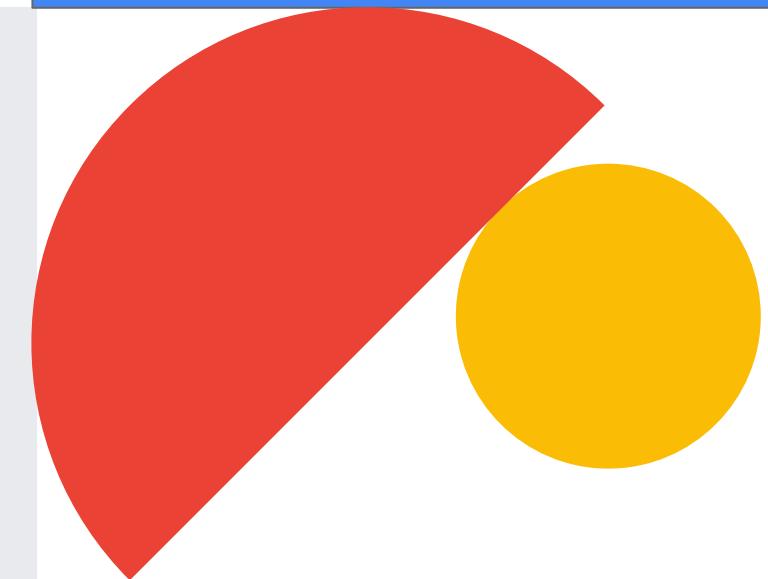
- Columns: 2 (target and input feature)
- Target column must be **categorical** (2-500 values) **or numerical**
- Rows: 1,000

Maximum

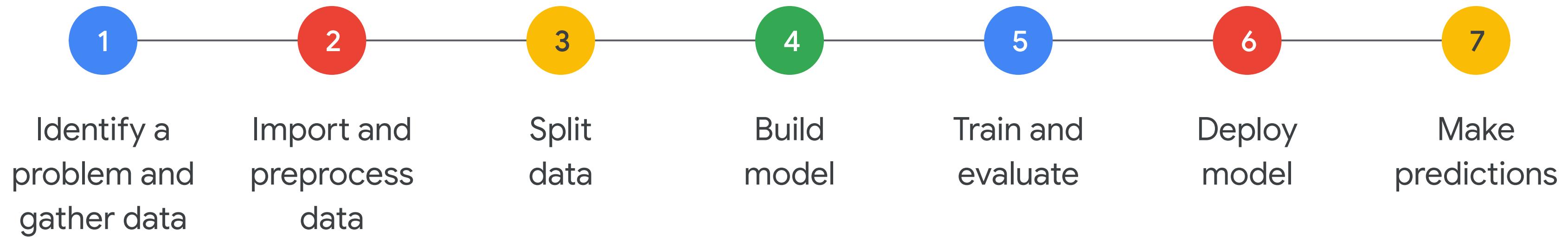
- Dataset size: 100 GB
- Columns: 1,000
- Rows: 100 million

Using BigQuery ML

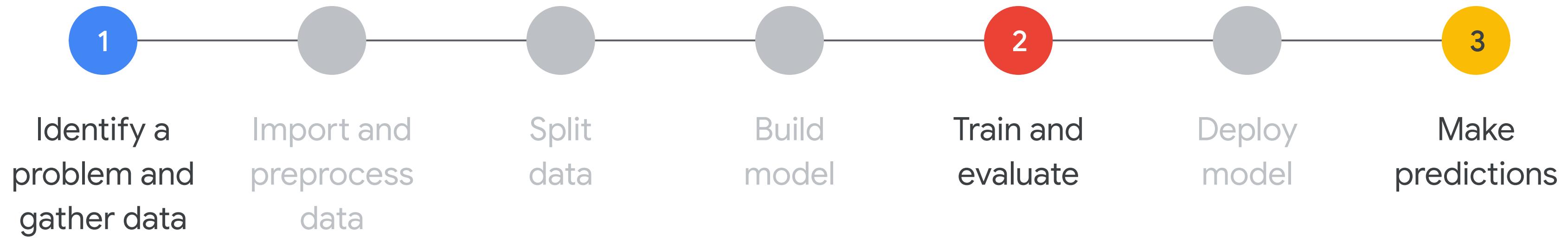
- Allows you to use SQL to invoke machine learning models on structured data.
- Can provide decision-making guidance through predictive analytics by using its machine learning tool, BigQuery ML.
- Doesn't require exporting data out of BigQuery to create and train a model.



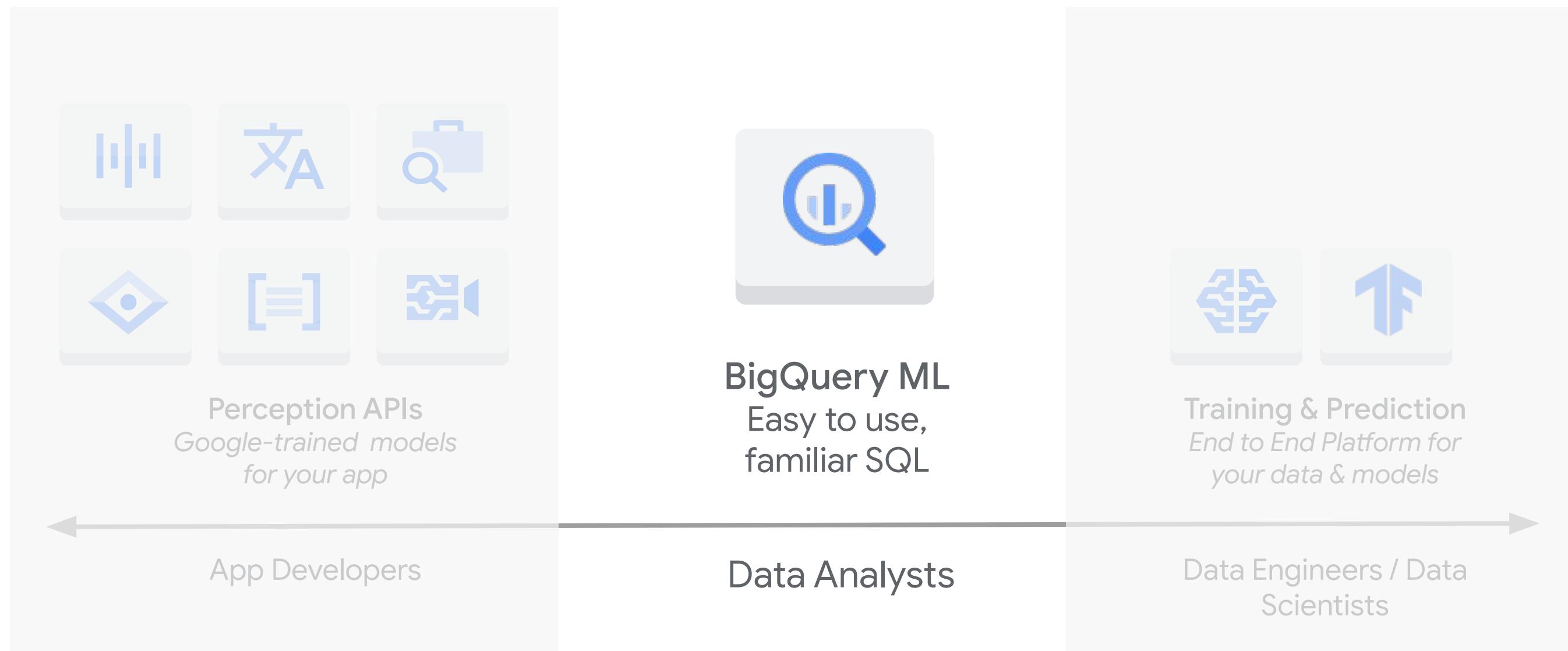
Custom model **without** BigQuery ML



Custom model with BigQuery ML



BigQuery ML is a way to easily build machine learning models



Months to create and deploy an ML model

Tasks

- Suboptimal models on small data in notebooks
- ETL for model training
- Infrastructure management for production models
- ETL for batch predictions
- Extensive data science training

Security & compliance

Set up and maintain data access policies for laptops, notebooks, VM clusters

- But I already defined these for my data
- Has anyone managed to do this successfully?

Working with BigQuery ML



01

Dataset

02

Create/ train

03

Evaluate

04

Predict/ classify

```
FROM  
ML.EVALUATE(MODEL `BigQuery  
ML_tutorial.sample_model`,  
TABLE eval_table)
```

```
CREATE MODEL `BigQuery  
ML_tutorial.sample_model`  
OPTIONS(model_type='logistic_reg') AS  
SELECT
```

```
FROM  
ML.PREDICT(MODEL `BigQuery  
ML_tutorial.sample_model`,  
table game_to_predict) ) AS  
predict
```

BigQuery ML

- 
- 1 Execute ML initiatives without moving data from BigQuery
 - 2 Iterate on models in SQL in BigQuery to increase development speed
 - 3 Automate common ML tasks and hyperparameter tuning

<https://cloud.google.com/bigquery-ml/docs/tutorials>

Example use case

Table info

Table ID	nyc-tlc:yellow.trips
Table Size	129.72 GB
Long-term storage size	129.72 GB
Number of rows	1,108,779,463

pickup_datetime	dropoff_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	rate_code	passenger_count
2010-03-04 00:35:16 UTC	2010-03-04 00:35:16 UTC	-74.035201	40.721548	-74.035201	40.721548	1	1
2010-03-15 17:18:34 UTC	2010-03-15 17:18:35 UTC	0.0	0.0	0.0	0.0	1	1
2015-03-18 01:07:02 UTC	2015-03-18 01:07:07 UTC	0.0	0.0	0.0	0.0	1	5
2015-03-09 01:07:02 UTC	2015-03-09 18:25:37 UTC	-73.93724822998047	40.758201599121094	-73.93726348876953	40.7581901550293	1	1
2010-03-06 06:33:41 UTC	2015-03-06 06:36:06 UTC	-73.785514	40.6454	-73.784564	40.648681	1	2
2013-08-07 00:42:45 UTC	2013-08-07 00:58:43 UTC	-74.025817	40.763044	74.046752	40.78324	5	1
2015-04-26 02:56:37 UTC	2015-04-26 03:00:01 UTC	-73.98765563964844	40.77165603637695	-73.98755645751953	40.771751403808594	1	1
2015-04-29 18:45:03 UTC	2015-04-29 18:49:01 UTC	0.00	0.00	0.00	0.00	1	1
2010-03-11 21:24:48 UTC	2010-03-11 21:46:51 UTC	-74.571511	40.9108	-74.628928	40.964321	1	1
2013-08-24 01:58:23 UTC	2013-08-24 01:58:23 UTC	-73.972171	40.759439	0.00	0.00	5	4

SQL query to extract data

```
SELECT  
    fare_amount,  
    pickup_longitude,  
    pickup_latitude,  
    dropoff_longitude,  
    dropoff_latitude,  
    passenger_count  
  
FROM  
    `nyc-tlc.yellow.trips`
```

```
CREATE OR REPLACE MODEL  
mydataset.model_linreg
```

```
OPTIONS(  
    input_label_cols=['fare_amount'],  
    model_type='linear_reg') AS
```

```
SELECT  
    fare_amount,  
    pickup_longitude,  
    pickup_latitude,  
    dropoff_longitude,  
    dropoff_latitude,  
    passenger_count  
  
FROM  
    `nyc-tlc.yellow.trips`
```

Build and train with **CREATE MODEL**

Build and train with **CREATE MODEL**

```
CREATE OR REPLACE MODEL  
mydataset.model_linreg
```

```
OPTIONS(  
    input_label_cols=['fare_amount'],  
    model_type='linear_reg') AS
```

```
SELECT  
    fare_amount,  
    pickup_longitude,  
    pickup_latitude,  
    dropoff_longitude,  
    dropoff_latitude,  
    passenger_count  
  
FROM  
    `nyc-tlc.yellow.trips`
```

Evaluate with **ML.EVALUATE**

```
SELECT
  *
FROM
  ML.EVALUATE(
    MODEL mydataset.model_linreg
  )
```

Use the model with **ML.PREDICT**

```
SELECT
  *
FROM
  ML.PREDICT(MODEL mydataset.model_linreg,
  (
    SELECT
      fare_amount,
      pickup_longitude,
      pickup_latitude,
      dropoff_longitude,
      dropoff_latitude,
      passenger_count
    FROM
      `nyc-tlc.yellow.trips`
  ))
```

Sample of BigQuery ML Training Metrics



BigQuery ML supported models and features

Classification

- Logistic regression
- DNN classifier (TensorFlow)
- XGBoost
- AutoML Tables
- Wide and Deep NNs^{Preview, GA H1'21}

Other Models

- k-means clustering
- Time series forecasting
- Recommendation: Matrix factorization
- Time series anomaly detection^{Preview Q2'21, GA H2'21}

Regression

- Linear regression
- DNN regressor (TensorFlow)
- XGBoost
- AutoML Tables
- Wide and Deep NNs^{Preview, GA H1'21}

Model ops and explainability

- Import/export TensorFlow models for batch and online prediction
- **hyperparameter tuning using Cloud AI Vizier**^{Preview H1'21, GA H2'21}
- **Model explainability using Cloud AI**^{Preview H1'21, GA H2'21}
- **Managed Kubernetes and TFX pipelines**^{Preview H2'21, GA 2022}
- **List models for comparison and online deployment in Cloud AI**^{Preview H2'21, GA 2022}
- Model versioning, continuous monitoring^{future}

BigQuery ML supported models and features

Classification

- Logistic regression
- DNN classifier (TensorFlow)
- XGBoost
- AutoML Tables
- Wide and Deep NNs^{Preview, GA H1'21}

Other Models

- k-means clustering
- Time series forecasting
- Recommendation: Matrix factorization
- Time series anomaly detection^{Preview Q2'21, GA H2'21}

Regression

- Linear regression
- DNN regressor (TensorFlow)
- XGBoost
- AutoML Tables
- Wide and Deep NNs^{Preview, GA H1'21}

Model ops and explainability

- Import/export TensorFlow models for batch and online prediction
- **hyperparameter tuning using Cloud AI Vizier**^{Preview H1'21, GA H2'21}
- **Model explainability using Cloud AI**^{Preview H1'21, GA H2'21}
- **Managed Kubernetes and TFX pipelines**^{Preview H2'21, GA 2022}
- **List models for comparison and online deployment in Cloud AI**^{Preview H2'21, GA 2022}
- Model versioning, continuous monitoring^{future}

BigQuery ML supported models and features

Classification

- Logistic regression
- DNN classifier (TensorFlow)
- XGBoost
- AutoML Tables
- Wide and Deep NNs^{Preview, GA H1'21}

Regression

- Linear regression
- DNN regressor (TensorFlow)
- XGBoost
- AutoML Tables
- Wide and Deep NNs^{Preview, GA H1'21}

Other Models

- k-means clustering
- Time series forecasting
- Recommendation: Matrix factorization
- Time series anomaly detection^{Preview Q2'21, GA H2'21}

Model ops and explainability

- Import/export TensorFlow models for batch and online prediction
- **hyperparameter tuning using Cloud AI Vizier**^{Preview H1'21, GA H2'21}
- **Model explainability using Cloud AI**^{Preview H1'21, GA H2'21}
- **Managed Kubernetes and TFX pipelines**^{Preview H2'21, GA 2022}
- **List models for comparison and online deployment in Cloud AI**^{Preview H2'21, GA 2022}
- Model versioning, continuous monitoring^{future}

BigQuery ML supported models and features

Classification

- Logistic regression
- DNN classifier (TensorFlow)
- XGBoost
- AutoML Tables
- Wide and Deep NNs^{Preview, GA H1'21}

Other Models

- k-means clustering
- Time series forecasting
- Recommendation: Matrix factorization
- Time series anomaly detection^{Preview Q2'21, GA H2'21}

Regression

- Linear regression
- DNN regressor (TensorFlow)
- XGBoost
- AutoML Tables
- Wide and Deep NNs^{Preview, GA H1'21}

Model ops and explainability

- Import/export TensorFlow models for batch and online prediction
- **hyperparameter tuning using Cloud AI Vizier**^{Preview H1'21, GA H2'21}
- **Model explainability using Cloud AI**^{Preview H1'21, GA H2'21}
- **Managed Kubernetes and TFX pipelines**^{Preview H2'21, GA 2022}
- **List models for comparison and online deployment in Cloud AI**^{Preview H2'21, GA 2022}
- Model versioning, continuous monitoring^{future}

BigQuery ML supported models and features

Classification

- Logistic regression
- DNN classifier (TensorFlow)
- XGBoost
- AutoML Tables
- Wide and Deep NNs^{Preview, GA H1'21}

Other Models

- k-means clustering
- Time series forecasting
- Recommendation: Matrix factorization
- Time series anomaly detection^{Preview Q2'21, GA H2'21}

Regression

- Linear regression
- DNN regressor (TensorFlow)
- XGBoost
- AutoML Tables
- Wide and Deep NNs^{Preview, GA H1'21}

Model ops and explainability

- Import/export TensorFlow models for batch and online prediction
- **hyperparameter tuning using Cloud AI Vizier**^{Preview H1'21, GA H2'21}
- **Model explainability using Cloud AI**^{Preview H1'21, GA H2'21}
- **Managed Kubernetes and TFX pipelines**^{Preview H2'21, GA 2022}
- **List models for comparison and online deployment in Cloud AI**^{Preview H2'21, GA 2022}
- Model versioning, continuous monitoring^{future}

BigQuery ML supported models and features

Classification

- Logistic regression
- DNN classifier (TensorFlow)
- XGBoost
- AutoML Tables
- Wide and Deep NNs^{Preview, GA H1'21}

Other Models

- k-means clustering
- Time series forecasting
- Recommendation: Matrix factorization
- Time series anomaly detection^{Preview Q2'21, GA H2'21}

Regression

- Linear regression
- DNN regressor (TensorFlow)
- XGBoost
- AutoML Tables
- Wide and Deep NNs^{Preview, GA H1'21}

Model ops and explainability

- Import/export TensorFlow models for batch and online prediction
- **hyperparameter tuning using Cloud AI Vizier**^{Preview H1'21, GA H2'21}
- **Model explainability using Cloud AI**^{Preview H1'21, GA H2'21}
- **Managed Kubernetes and TFX pipelines**^{Preview H2'21, GA 2022}
- **List models for comparison and online deployment in Cloud AI**^{Preview H2'21, GA 2022}
- Model versioning, continuous monitoring^{future}

BigQuery ML supported models and features

Classification

- Logistic regression
- DNN classifier (TensorFlow)
- XGBoost
- AutoML Tables
- Wide and Deep NNs^{Preview, GA H1'21}

Other Models

- k-means clustering
- Time series forecasting
- Recommendation: Matrix factorization
- Time series anomaly detection^{Preview Q2'21, GA H2'21}

Regression

- Linear regression
- DNN regressor (TensorFlow)
- XGBoost
- AutoML Tables
- Wide and Deep NNs^{Preview, GA H1'21}

Model ops and explainability

- Import/export TensorFlow models for batch and online prediction
- **hyperparameter tuning using Cloud AI Vizier**^{Preview H1'21, GA H2'21}
- **Model explainability using Cloud AI**^{Preview H1'21, GA H2'21}
- **Managed Kubernetes and TFX pipelines**^{Preview H2'21, GA 2022}
- **List models for comparison and online deployment in Cloud AI**^{Preview H2'21, GA 2022}
- Model versioning, continuous monitoring^{future}

BigQuery ML supported models and features

Classification

- Logistic regression
- DNN classifier (TensorFlow)
- XGBoost
- AutoML Tables
- Wide and Deep NNs^{Preview, GA H1'21}

Other Models

- k-means clustering
- Time series forecasting
- Recommendation: Matrix factorization
- Time series anomaly detection^{Preview Q2'21, GA H2'21}

Regression

- Linear regression
- DNN regressor (TensorFlow)
- XGBoost
- AutoML Tables
- Wide and Deep NNs^{Preview, GA H1'21}

Model ops and explainability

- Import/export TensorFlow models for batch and online prediction
- **hyperparameter tuning using Cloud AI Vizier**^{Preview H1'21, GA H2'21}
- **Model explainability using Cloud AI**^{Preview H1'21, GA H2'21}
- **Managed Kubernetes and TFX pipelines**^{Preview H2'21, GA 2022}
- **List models for comparison and online deployment in Cloud AI**^{Preview H2'21, GA 2022}
- Model versioning, continuous monitoring^{future}

BigQuery ML supported models and features

Classification

- Logistic regression
- DNN classifier (TensorFlow)
- XGBoost
- AutoML Tables
- Wide and Deep NNs^{Preview, GA H1'21}

Other Models

- k-means clustering
- Time series forecasting
- Recommendation: Matrix factorization
- Time series anomaly detection^{Preview Q2'21, GA H2'21}

Regression

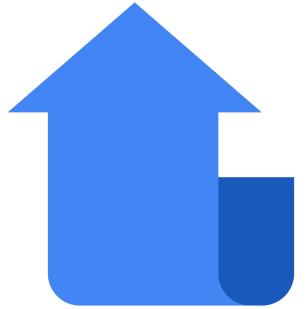
- Linear regression
- DNN regressor (TensorFlow)
- XGBoost
- AutoML Tables
- Wide and Deep NNs^{Preview, GA H1'21}

Model ops and explainability

- Import/export TensorFlow models for batch and online prediction
- **hyperparameter tuning using Cloud AI Vizier**^{Preview H1'21, GA H2'21}
- **Model explainability using Cloud AI**^{Preview H1'21, GA H2'21}
- **Managed Kubernetes and TFX pipelines**^{Preview H2'21, GA 2022}
- **List models for comparison and online deployment in Cloud AI**^{Preview H2'21, GA 2022}
- Model versioning, continuous monitoring^{future}

Hyperparameter tuning

A **hyperparameter** is a model argument whose value is set before the learning process begins.



Less time manually iterating hyperparameters



More time focusing on exploring insights from data

Hyperparameter tuning with BigQuery ML

BigQuery ML supports hyperparameter tuning when training ML models using `CREATE MODEL` statements.

Hyperparameter tuning is commonly used to improve model performance by searching for the optimal hyperparameters.

Hyperparameter tuning supports the



- LINEAR_REG
- LOGISTIC_REG
- KMEANS
- MATRIX_FACTORIZATION
- BOOSTED_TREE_CLASSIFIER
- BOOSTED_TREE_REGRESSOR
- DNN_CLASSIFIER
- DNN_REGRESSOR

Hyperparameter tuning

A Simple DNN

```
CREATE OR REPLACE MODEL `BigQuery ML.natality_dnn_hp`  
OPTIONS (MODEL_TYPE = 'DNN_REGRESSOR',  
        INPUT_LABEL_COLS = ['weight_pounds'] ,  
        DATA_SPLIT_METHOD = 'RANDOM' ,  
        EARLY_STOP=TRUE,  
        HIDDEN_UNITS = [30,50]  
) AS  
SELECT *  
FROM `BigQuery ML.natality`
```

Hyperparameter tuning

A Simple DNN

```
CREATE OR REPLACE MODEL `BigQuery
ML.natality_dnn_hp`
OPTIONS (MODEL_TYPE = 'DNN_REGRESSOR',
INPUT_LABEL_COLS = ['weight_pounds'] ,
DATA_SPLIT_METHOD = 'RANDOM',
EARLY_STOP=TRUE,
HIDDEN_UNITS = [30, 50]
) AS
SELECT *
FROM `BigQuery ML.natality`
```

DNN with hyperparameter tuning

```
CREATE OR REPLACE MODEL `BigQuery ML.natality_dnn_hp`
OPTIONS (MODEL_TYPE = 'DNN_REGRESSOR',
INPUT_LABEL_COLS = ['weight_pounds'] ,
DATA_SPLIT_METHOD = 'RANDOM',
EARLY_STOP=TRUE,
HIDDEN_UNITS = [30, 50]
--
-- HP tuning options
NUM_TRIALS = 10,
MAX_PARALLEL_TRIALS = 2,
DROPOUT=HPARAM_RANGE(0, 0.2),
OPTIMIZER = HPARAM_CANDIDATES(['adam', 'adagrad']),
LEARN_RATE=HPARAM_CANDIDATES([.001, .01])
) AS
SELECT *
FROM `BigQuery ML.natality`
```

Deep Neural Network (DNN) models **without** hyperparameter tuning

```
CREATE or replace MODEL BigQuery  
ML.wine_dnn  
OPTIONS(MODEL_TYPE='DNN_CLASSIFIER',  
       INPUT_LABEL_COLS = ['Quality'])  
as select *  
from BigQuery ML.ml_wine_quality;
```

Aggregate metrics	
Threshold	0.0000
Precision	0.2888
Recall	0.2558
Accuracy	0.5691
F1 score	0.2572
Log loss	2.8729
ROC AUC	0.5351

Deep Neural Network (DNN) models [with](#) hyperparameter tuning

```
CREATE or replace MODEL BigQuery  
ML.wine_dnn_learnrate_005  
OPTIONS(MODEL_TYPE='DNN_CLASSIFIER',  
        DROPOUT = 0.1,  
        EARLY_STOP = TRUE,  
        INPUT_LABEL_COLS = ['Quality'],  
        LEARN_RATE=0.015,  
        MAX_ITERATIONS = 20)  
as select *  
from BigQuery ML.ml_wine_quality;
```

Aggregate metrics		
Threshold	0.0000	
Precision	0.2891	
Recall	0.2471	
Accuracy	0.5884	
F1 score	0.23478	
Log loss	2.3478	
ROC AUC	0.7989	

Deep Neural Network (DNN) models

```
CREATE or replace MODEL BigQuery ML.wine_dnn
OPTIONS(MODEL_TYPE='DNN_CLASSIFIER',
        INPUT_LABEL_COLS = ['Quality'])
as select *
from BigQuery ML.ml_wine_quality;
```

Aggregate metrics

Threshold	0.0000
Precision	0.2888
Recall	0.2558
Accuracy	0.5691
F1 score	0.2572
Log loss	2.8729
ROC AUC	0.5351

Defaults:

- Activation_FN = 'RELU'
- Auto_Class_Weights = FALSE
- Batch_Size = 32 (or n)
- Dropout = 0.0
- Early_Stop = True
- Hidden_Units = [128*]
- Learn_Rate = 0.01
- Max_Iterations = 20
- Min_Rel_Progress = 0.1
(Early_Stop = True)
- Optimizer = 'ADAM'
- Warm_Start = False
- Data_Split :
 - Method = 'AUTO_SPLIT'
 - EVAL_FRACTION
 - COL

```
CREATE or replace MODEL
BigQuery ML.wine_dnn_learnrate_005
OPTIONS(MODEL_TYPE='DNN_CLASSIFIER',
        DROPOUT = 0.1,
        EARLY_STOP = TRUE,
        INPUT_LABEL_COLS = ['Quality'],
        LEARN_RATE=0.015,
        MAX_ITERATIONS = 20)
as select *
from BigQuery ML.ml_wine_quality;
```

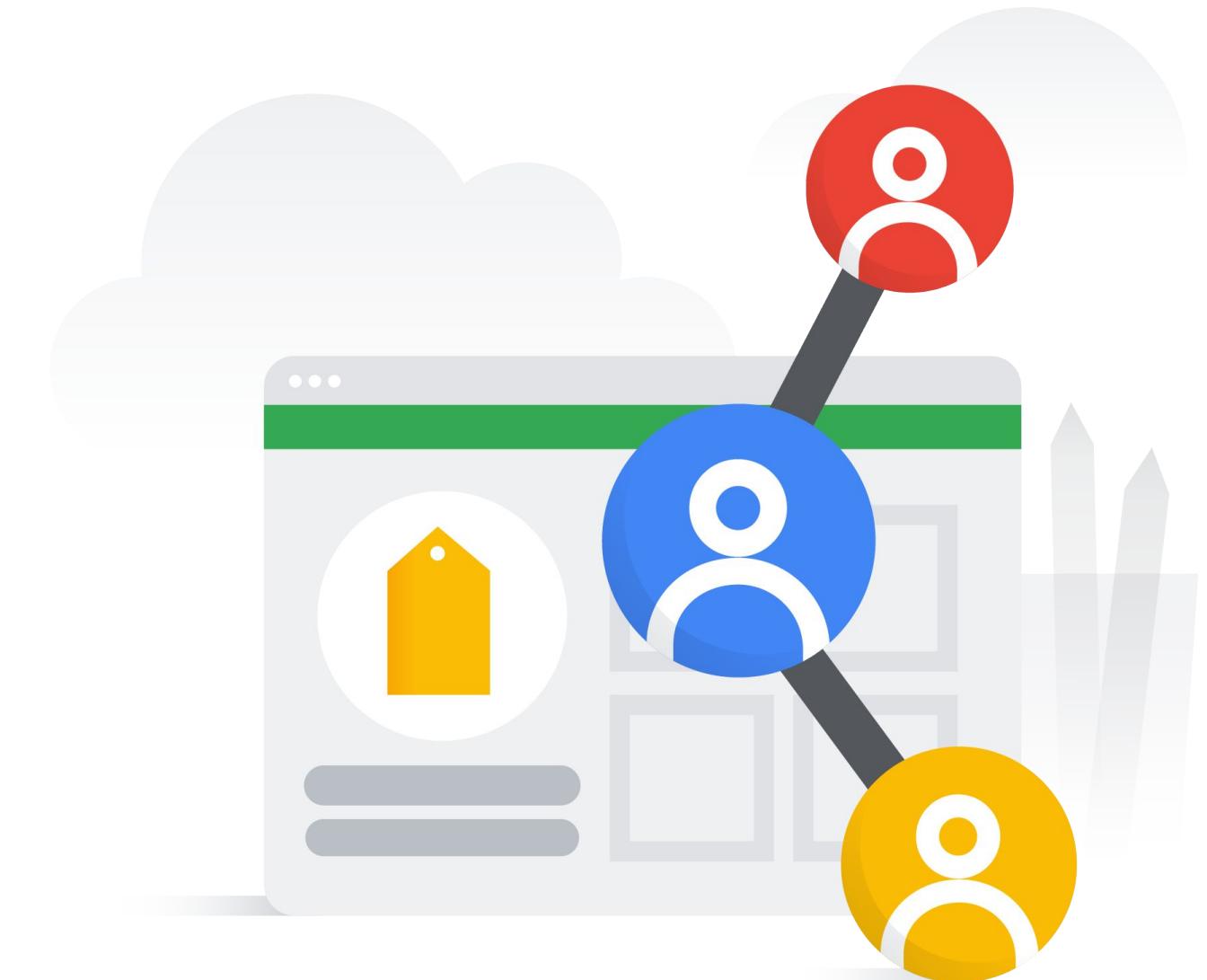
Aggregate metrics

Threshold	0.0000
Precision	0.2891
Recall	0.2471
Accuracy	0.5884
F1 score	0.2394
Log loss	2.3478
ROC AUC	0.7989

**Build a recommendation system with
BigQuery Machine Learning**

Businesses need to build recommendation systems to improve conversions and increase clickthrough rates.

And build customer loyalty.

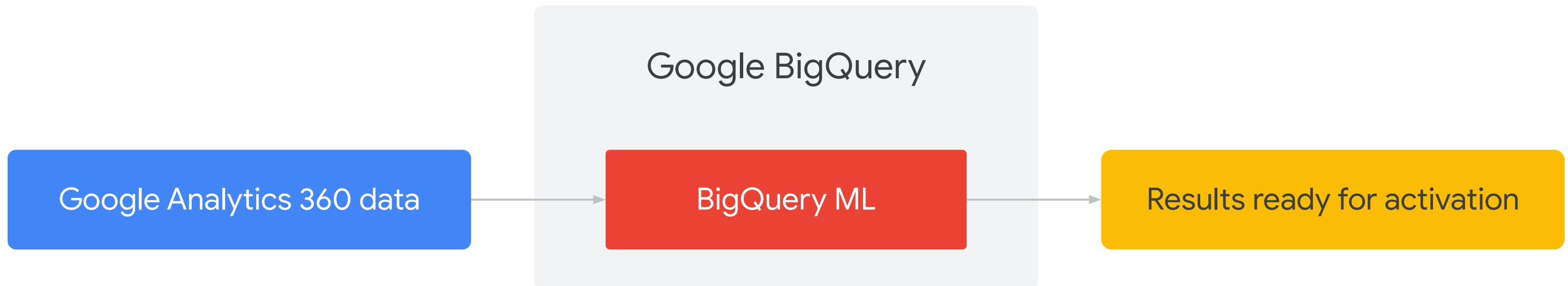


Common use case

- Prepare your training data in BigQuery
- Train a recommendation system with BigQuery ML
- Use the predicted recommendations in production

Common use case

- Prepare your training data in BigQuery
- Train a recommendation system with BigQuery ML
- Use the predicted recommendations in production



01

Preparing the training data



How do you know if a user likes a product?

item

service

movie

Through
feedback



Explicit feedback

HANDS-ON-LAB

Popular

Predict Housing Prices with
Tensorflow and AI Platform

Advanced



HANDS-ON-LAB

Highly Rated

Migrating a Monolithic
Website to Microservices on
Google Kubernetes Engine

Advanced



HANDS-ON-LAB

Highly Rated

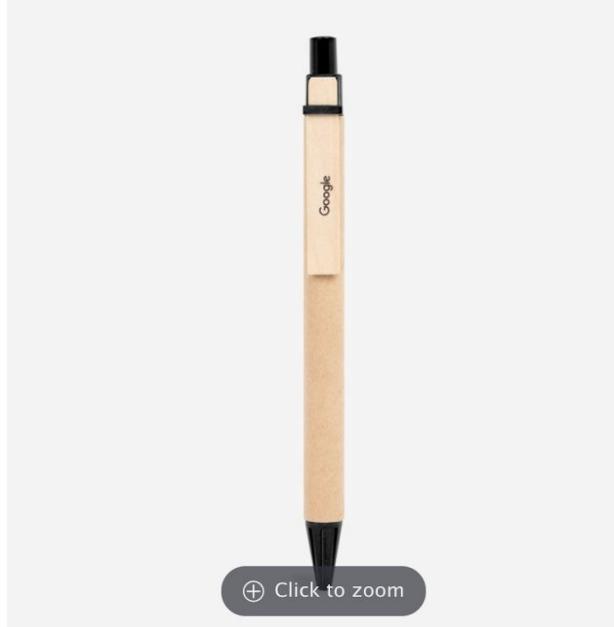
Learning TensorFlow: the
Hello World of Machine
Learning

Advanced



Implicit feedback

For example, amount of time spent viewing a product



A sustainable pen with a no-frills, recycled barrel. The plain kraft look inspires others to think green.

Google Recycled Pen Black
\$2.00

Available 28

Quantity

[ADD TO CART](#) [Add to Wishlist](#)



Google

official merchandise store



New Apparel ▾ Lifestyle ▾ Stationery ▾ Eco-Friendly Shop by Brand ▾ Sale Campus Collection

Brand of Item ▾

Sort By ▾

Hats



YouTube Twill Sandwich Cap Black

\$13.00



Google Leather Strap Hat Black

\$17.00



Google Leather Strap Hat Blue

\$17.00



YouTube Leather Strap Hat Black

\$17.00 \$11.90

Sample Google Analytics 360 data from the Google Merchandise Store

Pre-processed training data

	visitorId	itemID	session_duration
0	5468761641774363851-1	GGOEGAAX0031	59
1	7186762836575002506-1	GGOEGAAX0031	5787
2	1827763305655925232-3	GGOEGAAX0031	6423
3	3863873694540855771-4	GGOEGAAX0031	6168
4	2046797444872582027-2	GGOEGAAX0031	6891
5	5885261202339404877-2	GGOEGAAX0031	85256
6	0097216552247524030-5	GGOEGAAX0031	39980
7	860497822069663220-1	GGOEGAAX0031	2486
8	8960897138106393989-1	GGOEGAAX0031	11641
9	9681217015581152227-1	GGOEGAAX0031	6398

<https://console.cloud.google.com/marketplace/details/obfuscated-ga360-data/obfuscated-ga360-data?filter=solution-type:dataset>

Available on
BigQuery Public Datasets

Selecting your training data

Training data

	visitorId	itemID	session_duration
0	5468761641774363851-1	GGOEGAAX0031	59
1	7186762836575002506-1	GGOEGAAX0031	5787
2	1827763305655925232-3	GGOEGAAX0031	6423
3	3863873694540855771-4	GGOEGAAX0031	6168
4	2046797444872582027-2	GGOEGAAX0031	6891
5	5885261202339404877-2	GGOEGAAX0031	85256
6	0097216552247524030-5	GGOEGAAX0031	39980
7	860497822069663220-1	GGOEGAAX0031	2486
8	8960897138106393989-1	GGOEGAAX0031	11641
9	9681217015581152227-1	GGOEGAAX0031	6398

```
SELECT
  *
FROM
  BigQuery ML.aggregate_web_stats
```

02

Training a recommendation system with [BigQuery ML](#)



Build and train with CREATE MODEL

Training data

	visitorId	itemID	session_duration
0	5468761641774363851-1	GGOEGAAX0031	59
1	7186762836575002506-1	GGOEGAAX0031	5787
2	1827763305655925232-3	GGOEGAAX0031	6423
3	3863873694540855771-4	GGOEGAAX0031	6168
4	2046797444872582027-2	GGOEGAAX0031	6891
5	5885261202339404877-2	GGOEGAAX0031	85256
6	0097216552247524030-5	GGOEGAAX0031	39980
7	860497822069663220-1	GGOEGAAX0031	2486
8	8960897138106393989-1	GGOEGAAX0031	11641
9	9681217015581152227-1	GGOEGAAX0031	6398

```
CREATE OR REPLACE MODEL  
BigQuery ML.retail_recommender
```

```
OPTIONS(  
    model_type='matrix_factorization',  
    user_col='visitorId',  
    item_col='itemID',  
    rating_col='session_duration',  
    feedback_type='implicit'  
) AS
```

```
SELECT  
*  
FROM  
BigQuery ML.aggregate_web_stats
```

Evaluate the model with ML.EVALUATE

```
SELECT  
  *  
  
FROM  
  ML.EVALUATE(  
    MODEL BigQuery ML.retail_recommender  
)
```

CC

Row	mean_average_precision	mean_squared_error	normalized_discounted_cumulative_gain	average_rank
1	0.011463546060150412	1.0999974250389957E8-1	18.68240758482596	0.28084605495250364

Predict for a single user with ML.RECOMMEND

```
DECLARE MY_VISITORID STRING DEFAULT  
"6499749315992064304-2";  
  
SELECT  
  *  
FROM  
  ML.RECOMMEND(  
MODEL BigQuery  
ML.retail_recommender,  
(SELECT MY_VISITORID as visitorID)  
 )  
ORDER BY predicted_session_duration_  
confidence DESC  
LIMIT 5
```

Row	predicted_session_dura tion_confidence	visitorId	session_duration
1	35344.75190048335	0824461277962362623-1	GGOEGAAX0074
2	28775.356706353305	0824461277962362623-1	GGOEGBJC076099
3	26283729992603585	0824461277962362623-1	GGOEAFKQ020499
4	24583.28072228624	0824461277962362623-1	GGOEGAAX0289
5	23462.16560910448	0824461277962362623-1	GGOEGOAB021699

Batch predictions for all users with ML.RECOMMEND

Row	visitorId	itemId	predicted_session_duration_confidence
1	0000010278554503158-1	GGOEGBJC076099	33266.00563956515
2	0000010278554503158-1	GGOEADHB014799	23798.823558455344
3	0000010278554503158-1	GGOEYHPB072210	23317.03648749228
4	0000010278554503158-1	GGOEGEVA022399	23045.588017967602
5	0000010278554503158-1	GGOEGAAX0074	22231.48418727726
6	0000020424342248747-1	GGOEGBJC076099	3122.75144362899
7	0000020424342248747-1	GGOEYHPB072210	29845.991045227882
8	0000020424342248747-1	GGOEGODR017799	28248.607012765326
9	0000020424342248747-1	GGOEGEVA022399	28086.01190871777
10	0000020424342248747-1	GGOEGOBG023599	26348.173682193796

03

How to use the predicted
recommendations
in production



Export recommendations for ad retargeting campaigns

For product campaigns:

- create a new column for “likelihood to purchase”
- based on the predicted recommendations,
- then import into Google Analytics to create new campaigns.

clientId	LikelyToBuyProductA
123	0.70
345	0.90

Connect with your CRM for personalized emails

“Hi Conchita, you might be
interested in {A}, {B}, {C}”



Using BigQuery ML to create a recommendation system involves three steps:

01

[Prepare](#) a training data in BigQuery.

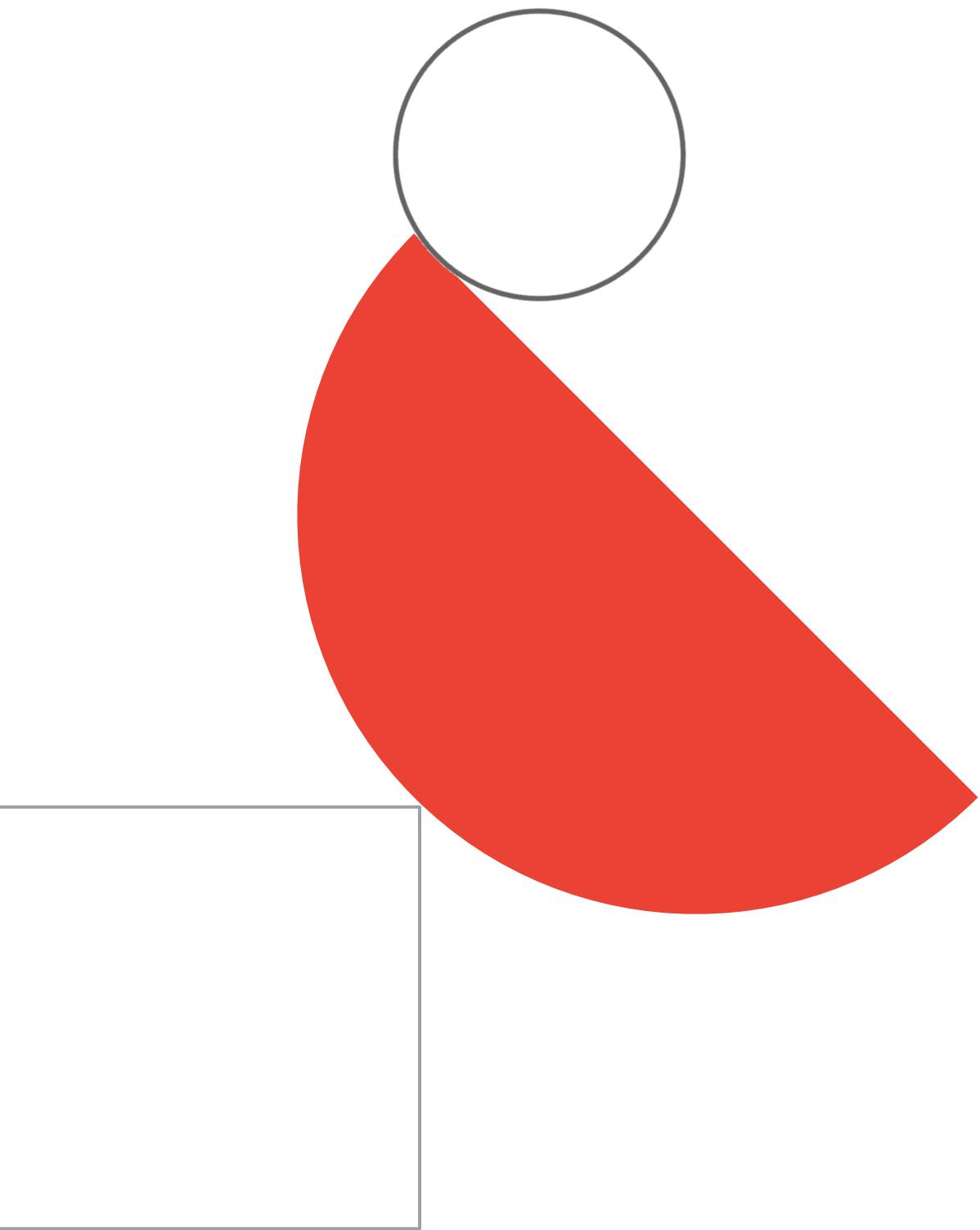
02

[Train](#) a recommendation system with BigQuery ML.

03

Use the predicted recommendations [in production](#).

Optimization



In this module, you learn to ...

01

Measure model performance objectively using loss functions

02

Use loss functions as the basis for an algorithm called gradient descent

03

Optimize gradient descent to be as efficient as possible

04

Use performance metrics to make business decisions

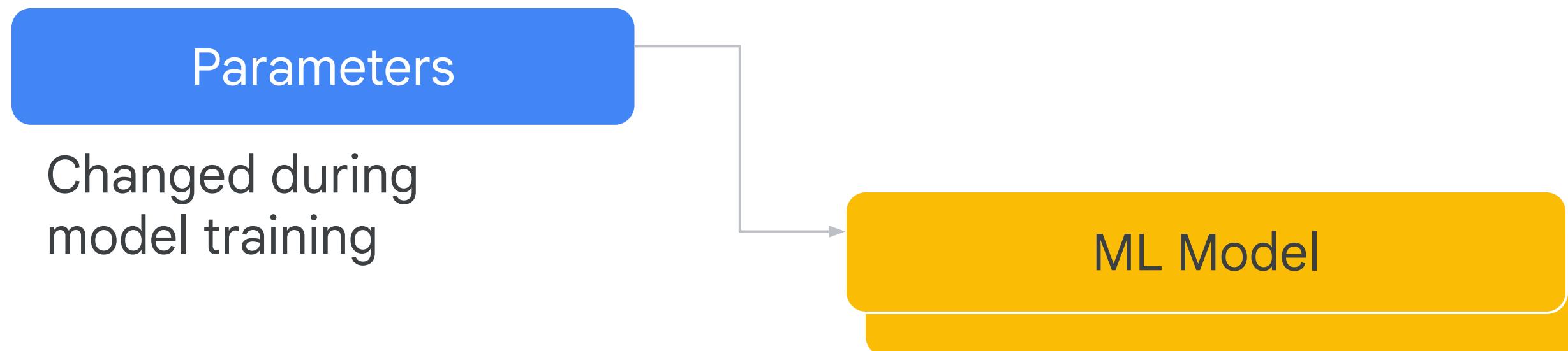


Machine learning models and parameters

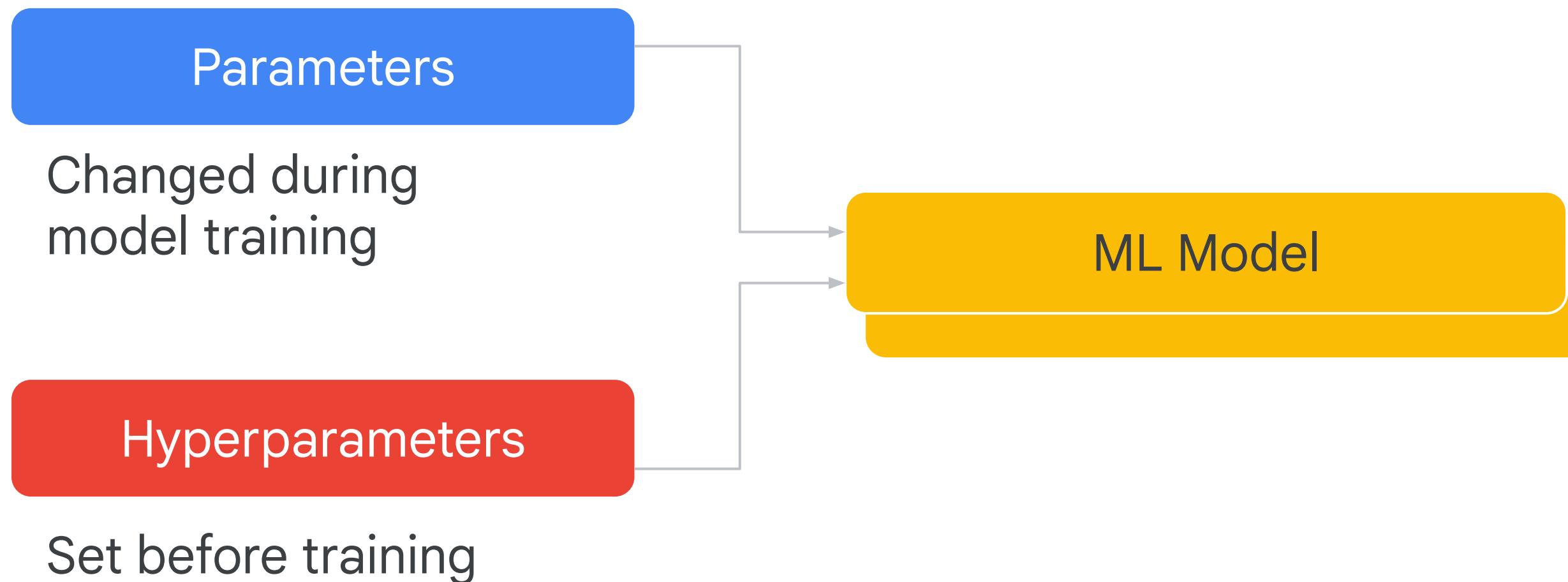
**ML models are mathematical functions with
parameters and hyper-parameters**

ML Model

ML models are mathematical functions with parameters and hyper-parameters

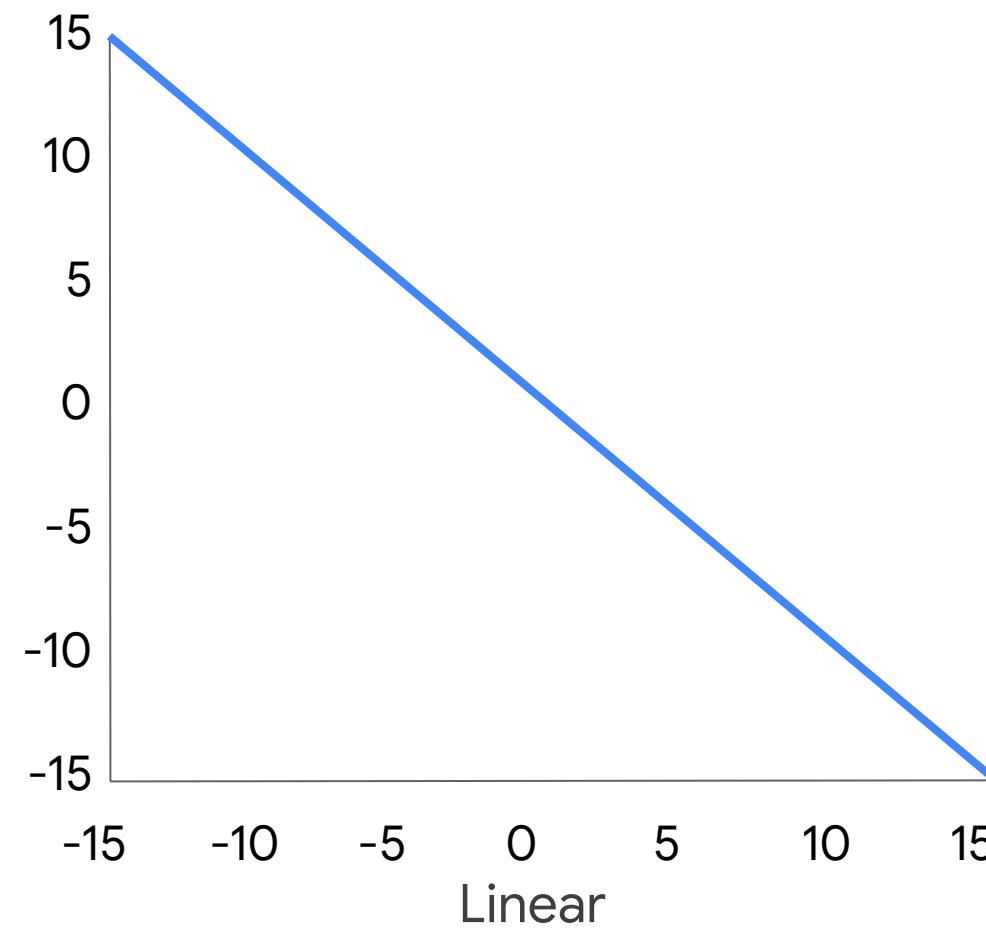


ML models are mathematical functions with parameters and hyper-parameters



Linear models have two types of parameters:

Bias and weight



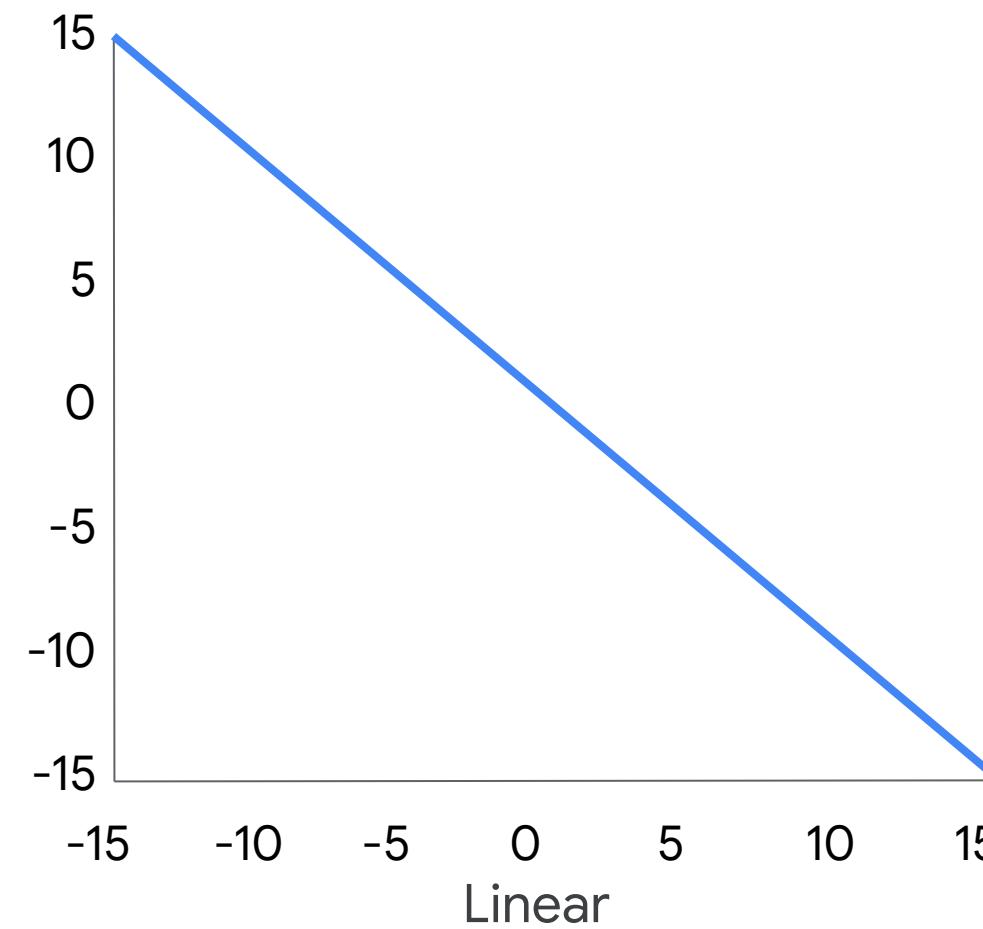
Output Bias Term Input Weight

$$\leftarrow y = \boxed{b} + x \times \boxed{m}$$

Model parameters

Linear models have two types of parameters:

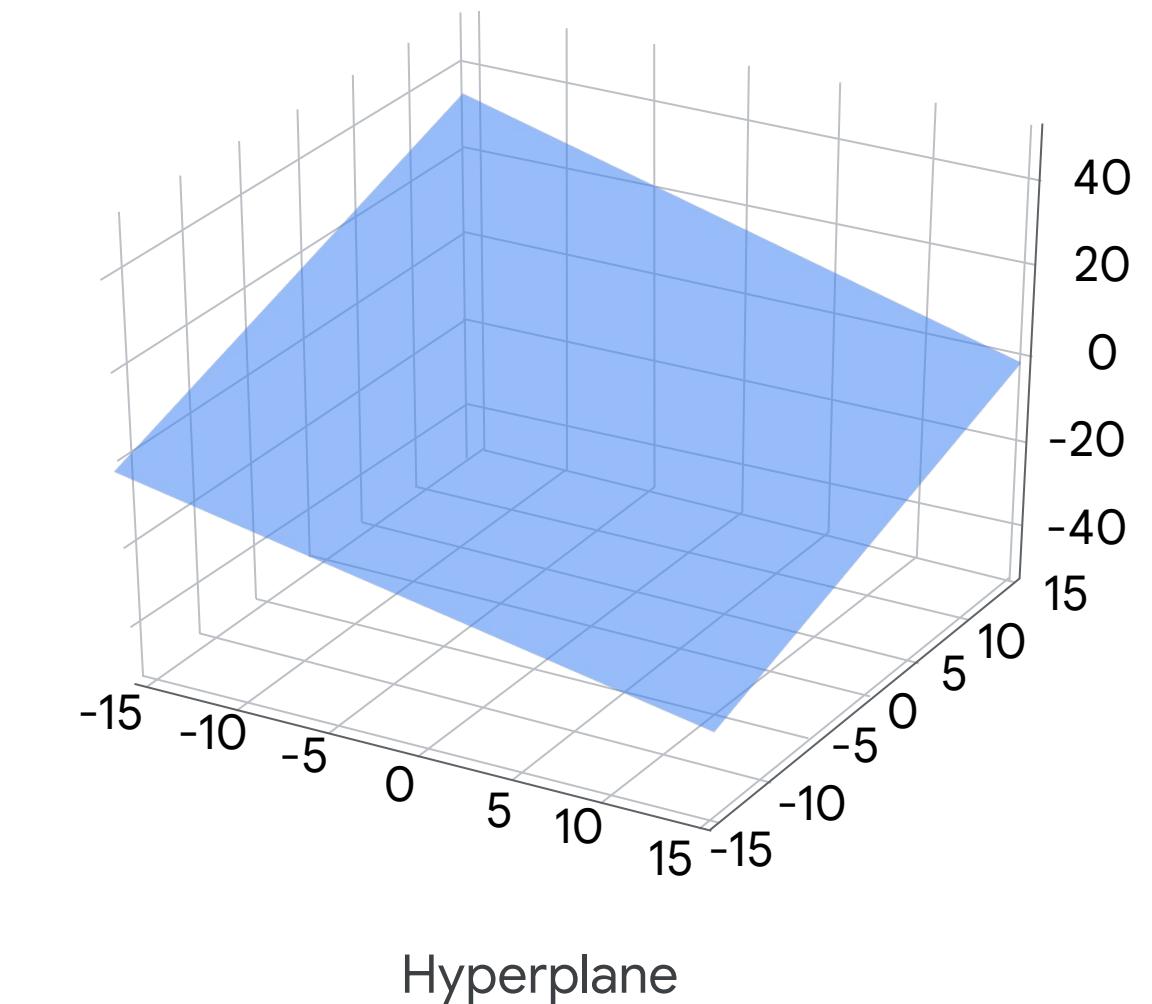
Bias and weight



Output Bias Term Input Weight

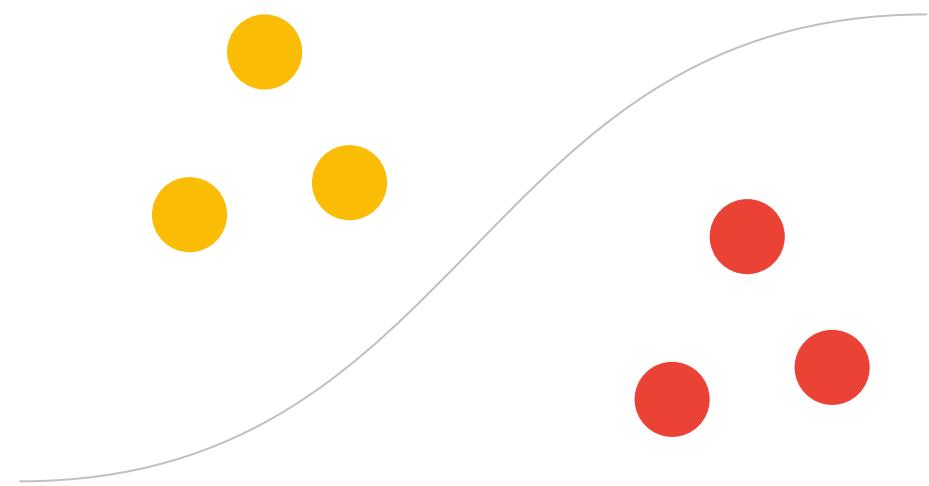
$$\leftarrow y = b + x \times m$$
$$y = b + X \times W \rightarrow$$

Model parameters

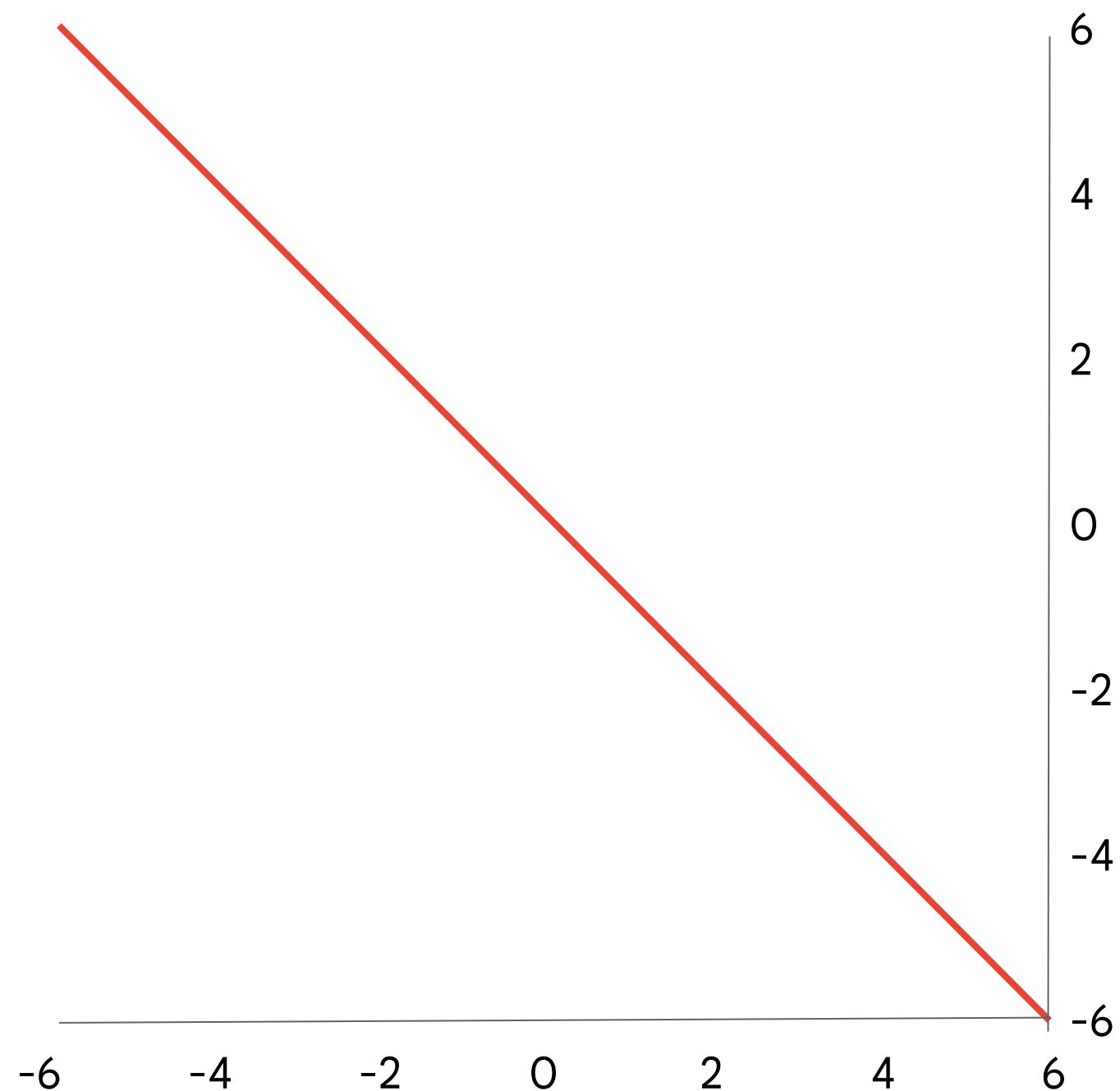


$$b + m \times x = \textcolor{blue}{y}$$

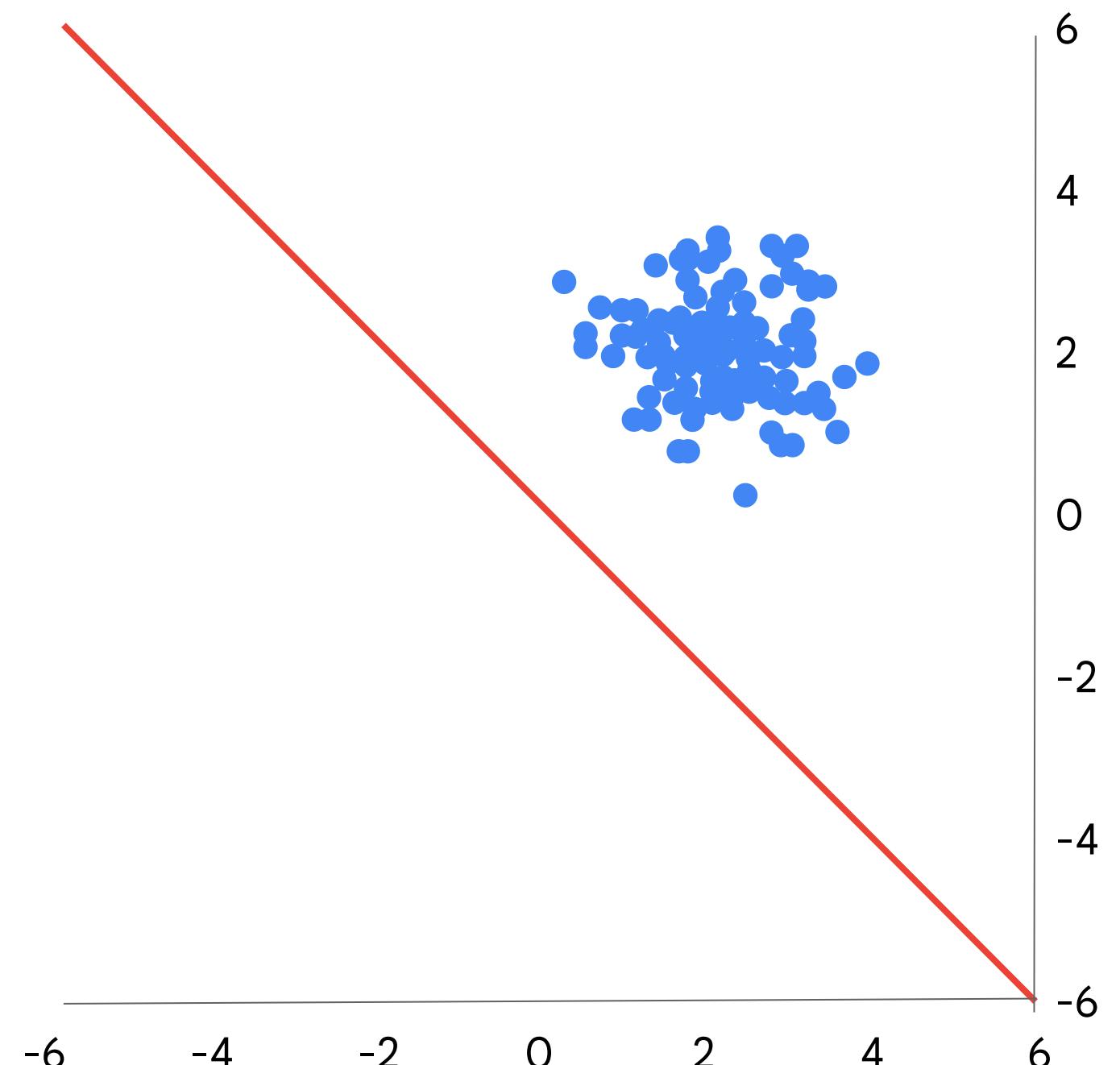
How is class membership encoded?



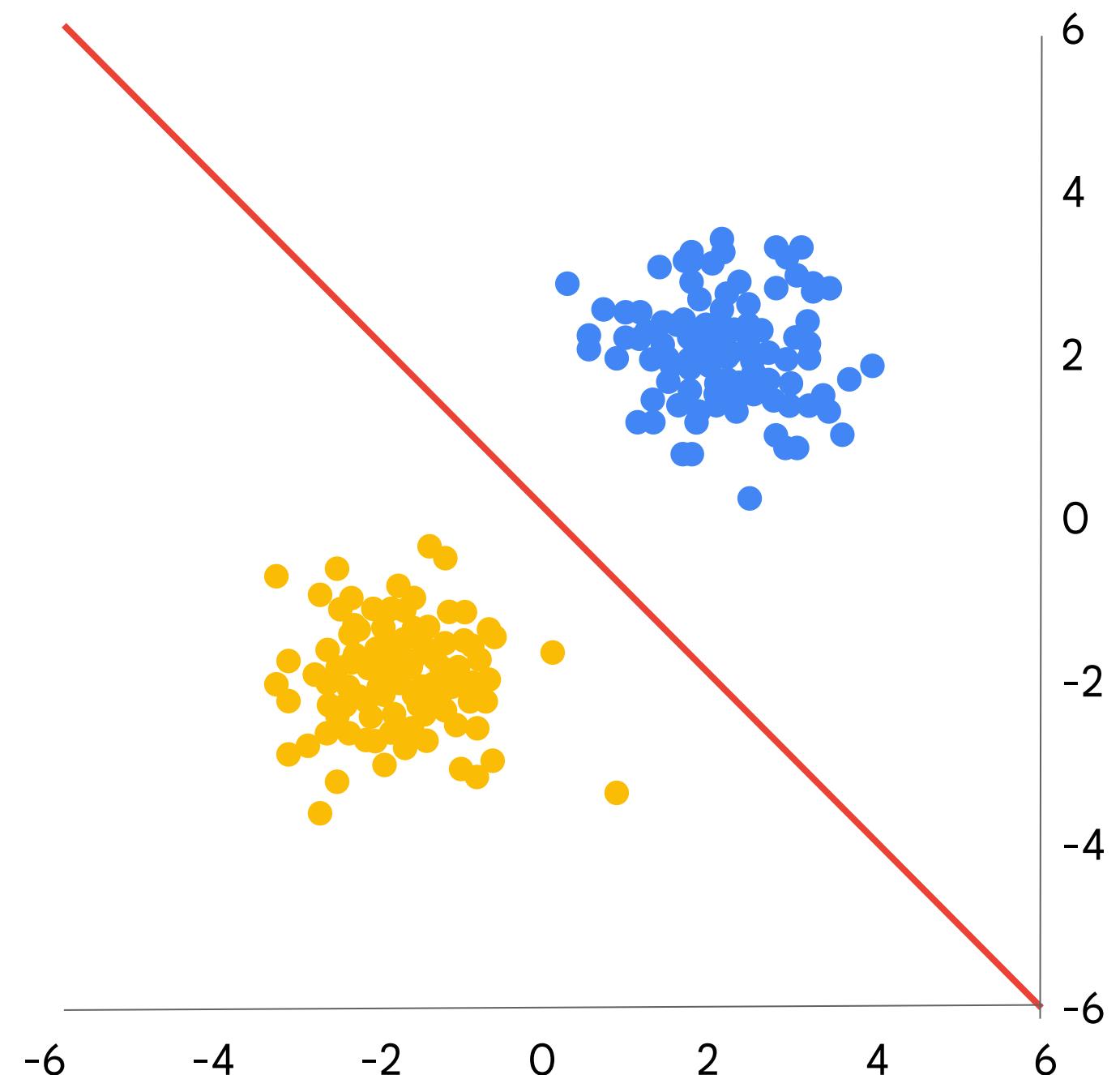
How can linear models classify data?



How can linear models classify data?



How can linear models classify data?



Problem statement:

Not all babies get the care that they need



How do we predict a baby's health *before* they are born?

Which of these could be a
feature in your model?

- A. Mother's Age
- B. Birth Time
- C. Baby Weight

How do we predict a baby's health *before* they are born?

Which of these could be a
feature in your model?

- A. Mother's Age
- B. Birth Time
- C. Baby Weight

Which could be a **label**?

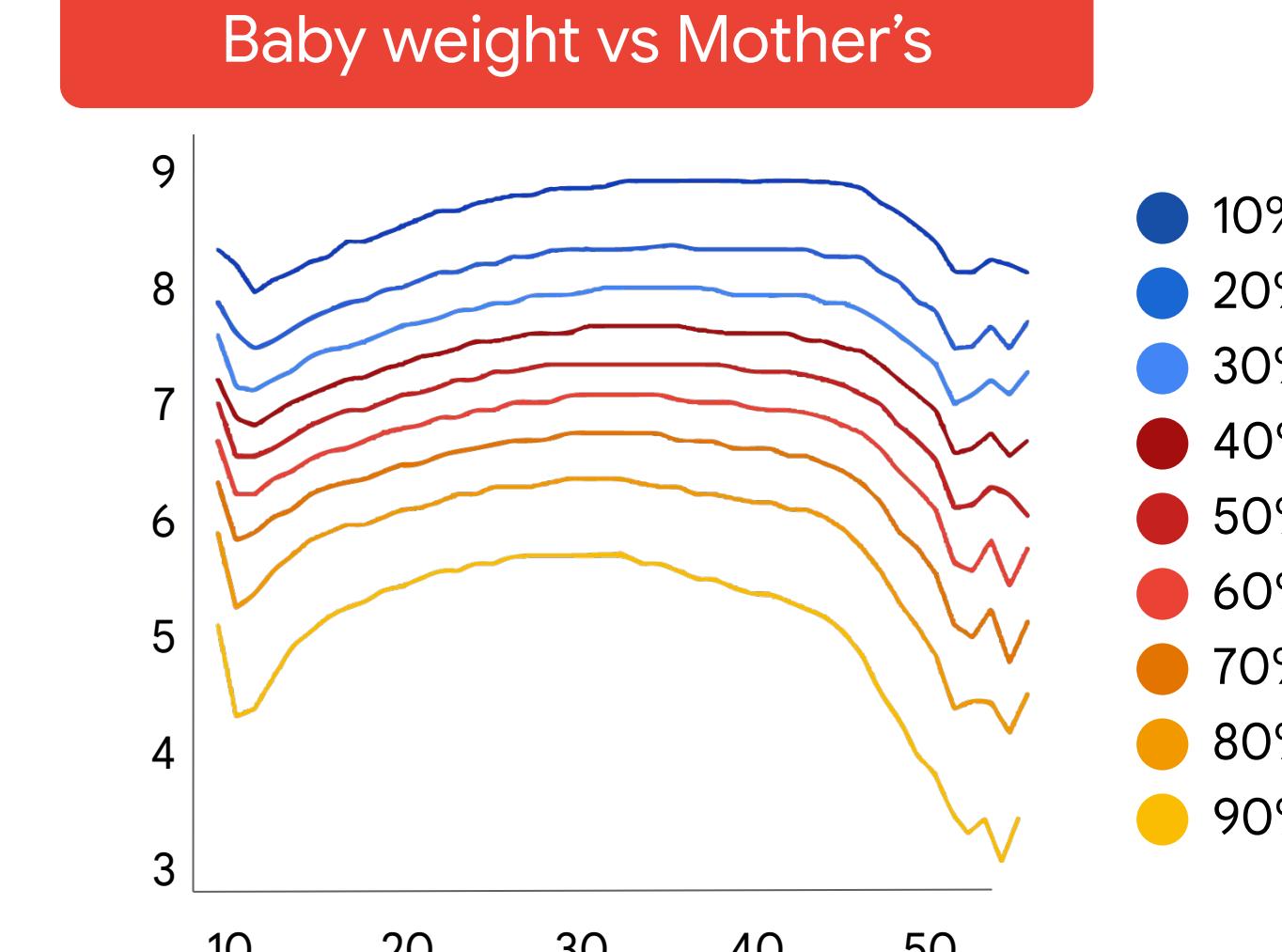
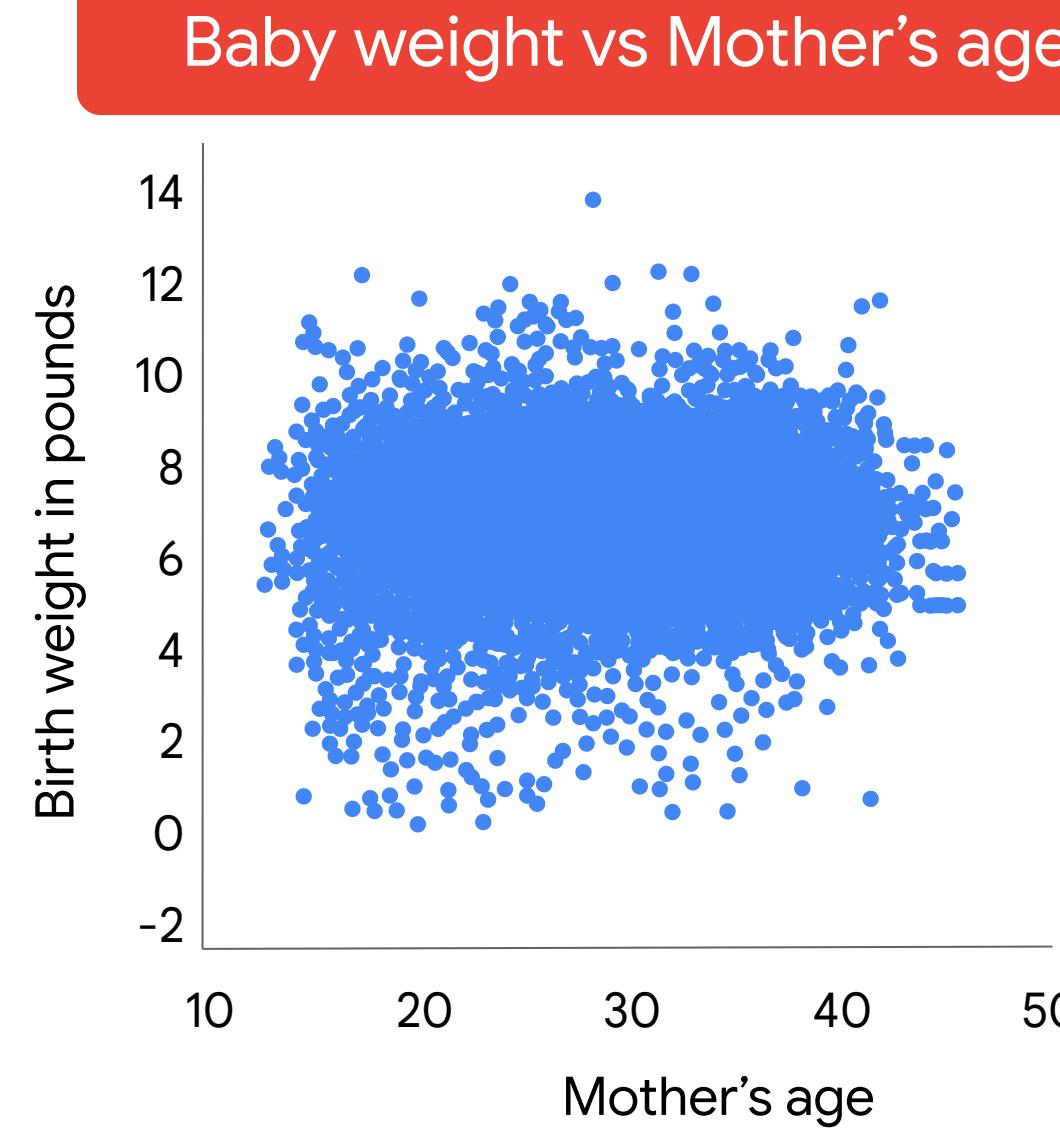
How do we predict a baby's health *before* they are born?

Which of these could be a
feature in your model?

- A. Mother's Age
- B. Birth Time
- C. Baby Weight

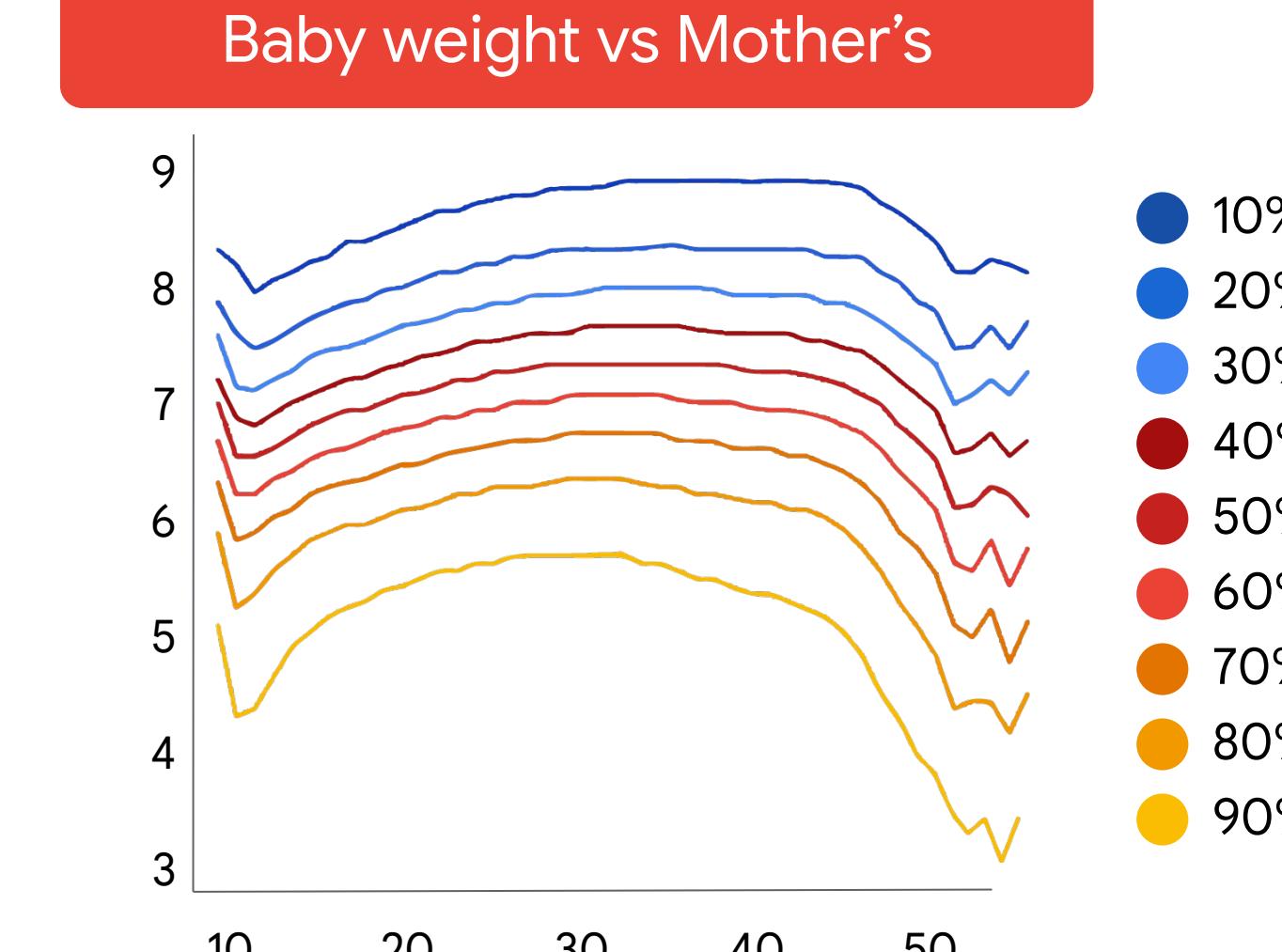
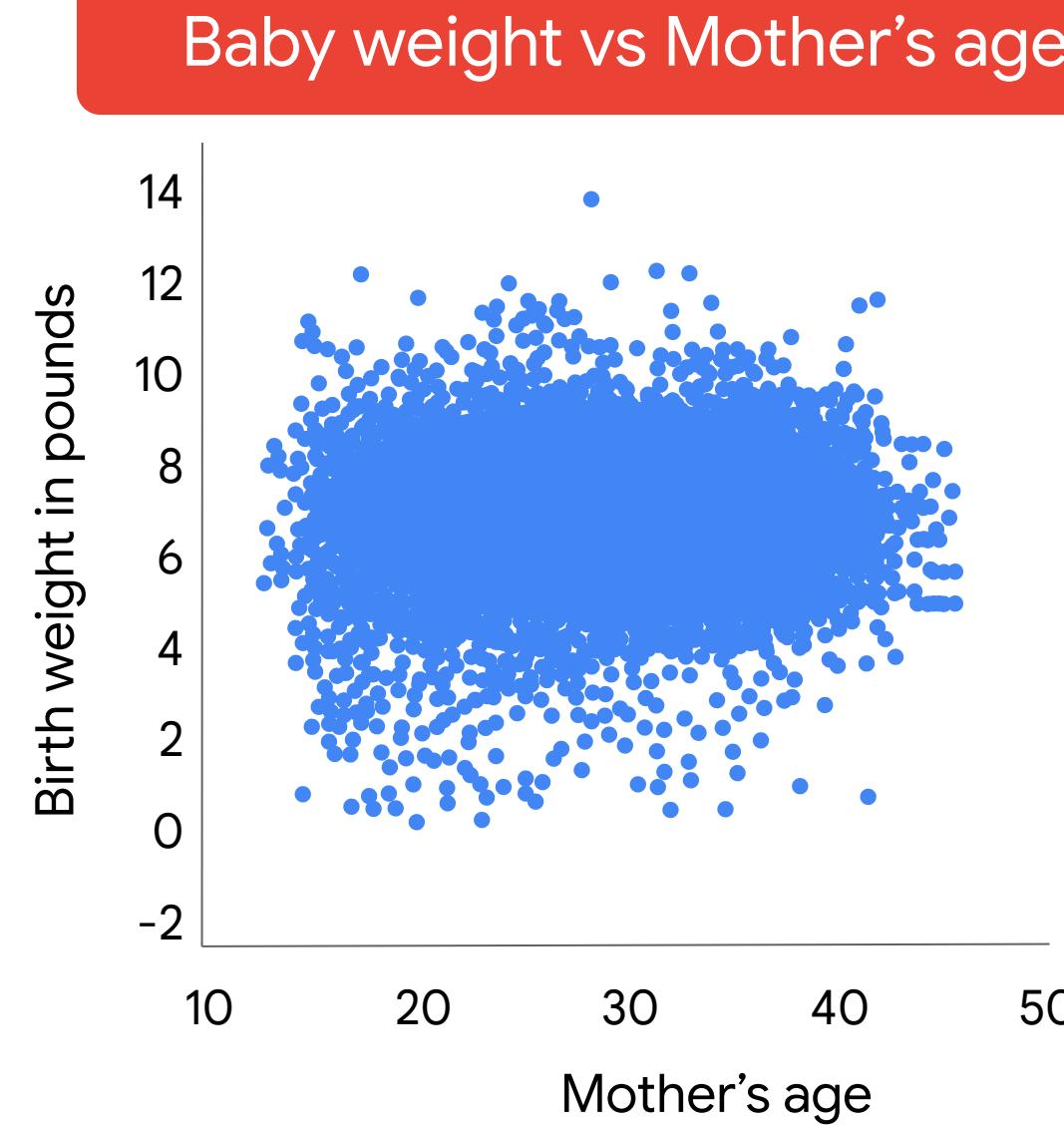
Which could be a **label**?

Exploring the data visually



Exploring the data visually

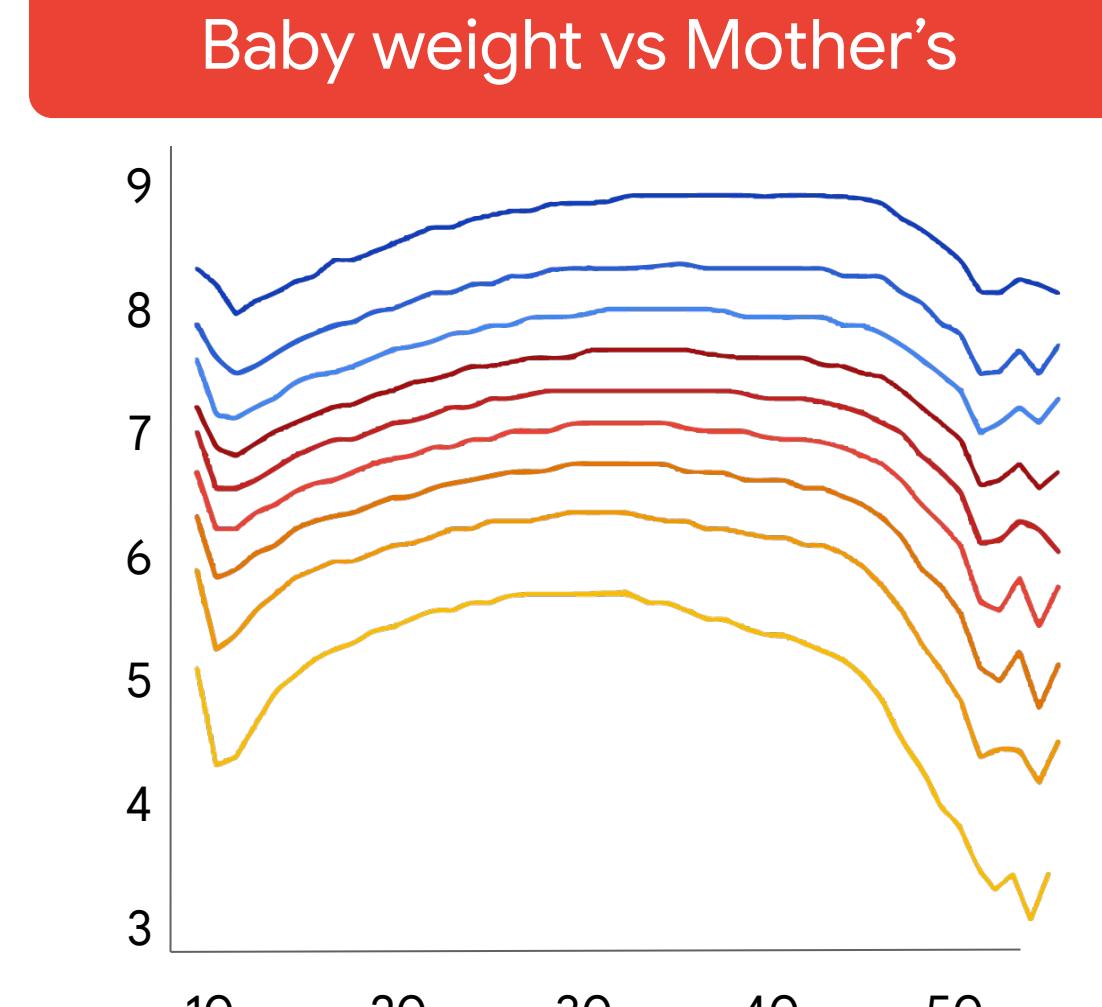
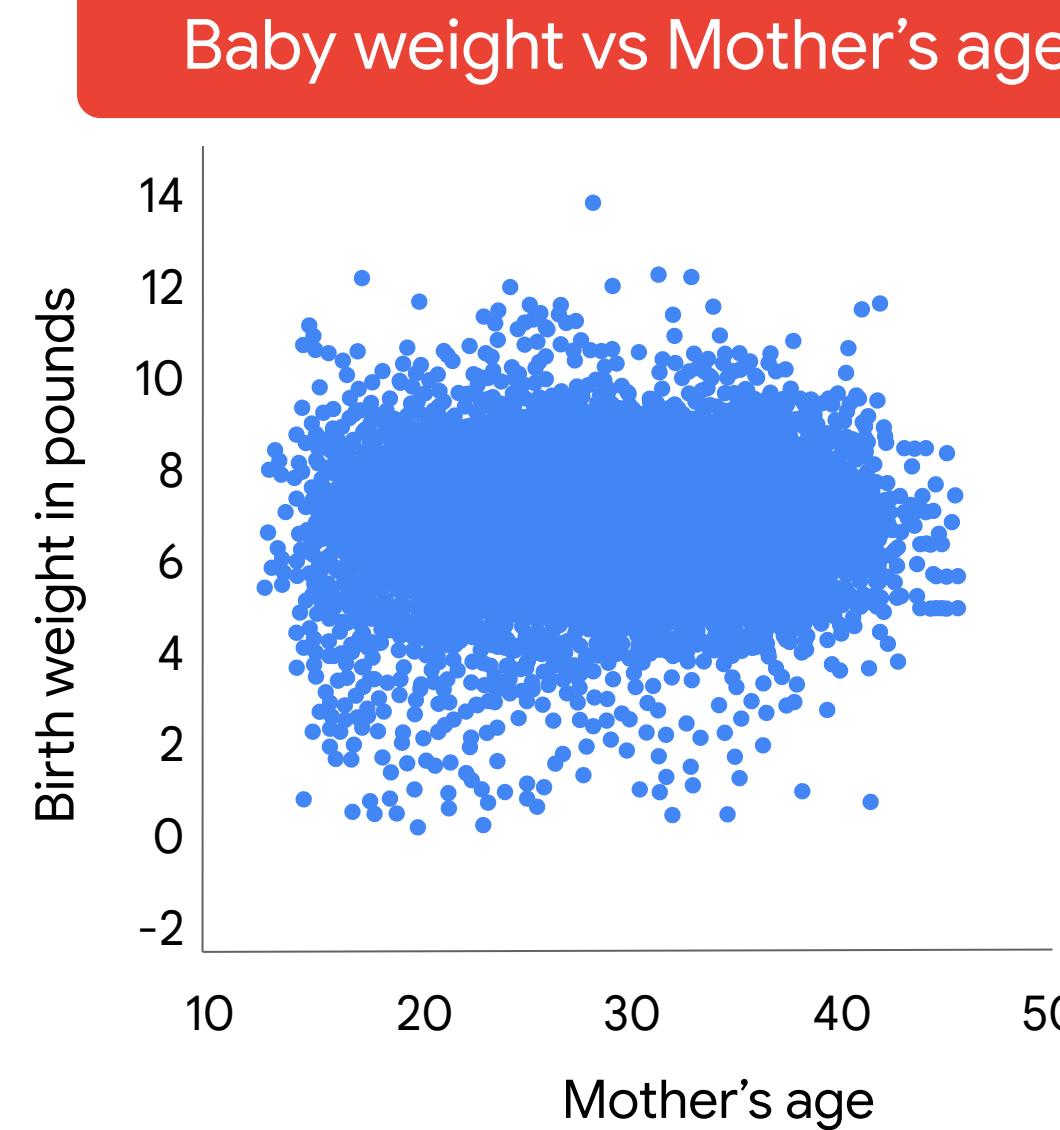
Scatterplots are made from samples of large datasets rather than from the whole dataset.



Exploring the data visually

Scatterplots are made from samples of large datasets rather than from the whole dataset.

Graph representing groups of data, specifically, quantiles.



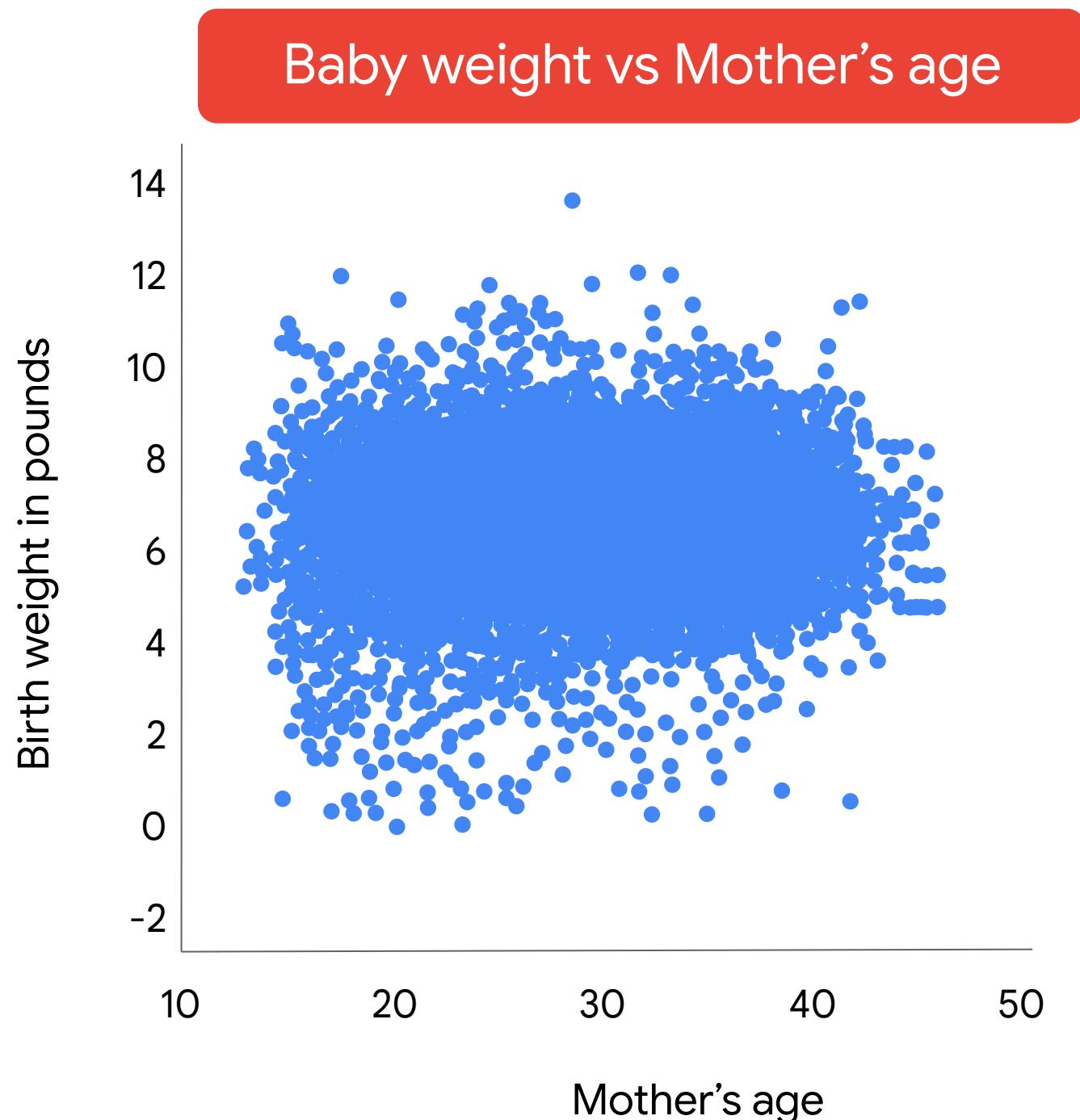
Equation for a linear model tying mother's age and baby weight

The slope of the line is given by w1.

$$y = w_1 x_1 + b$$

x_1 is the feature (e.g. mother's age)

w_1 is the weight for x_1



Equation for a linear model tying mother's age and baby weight

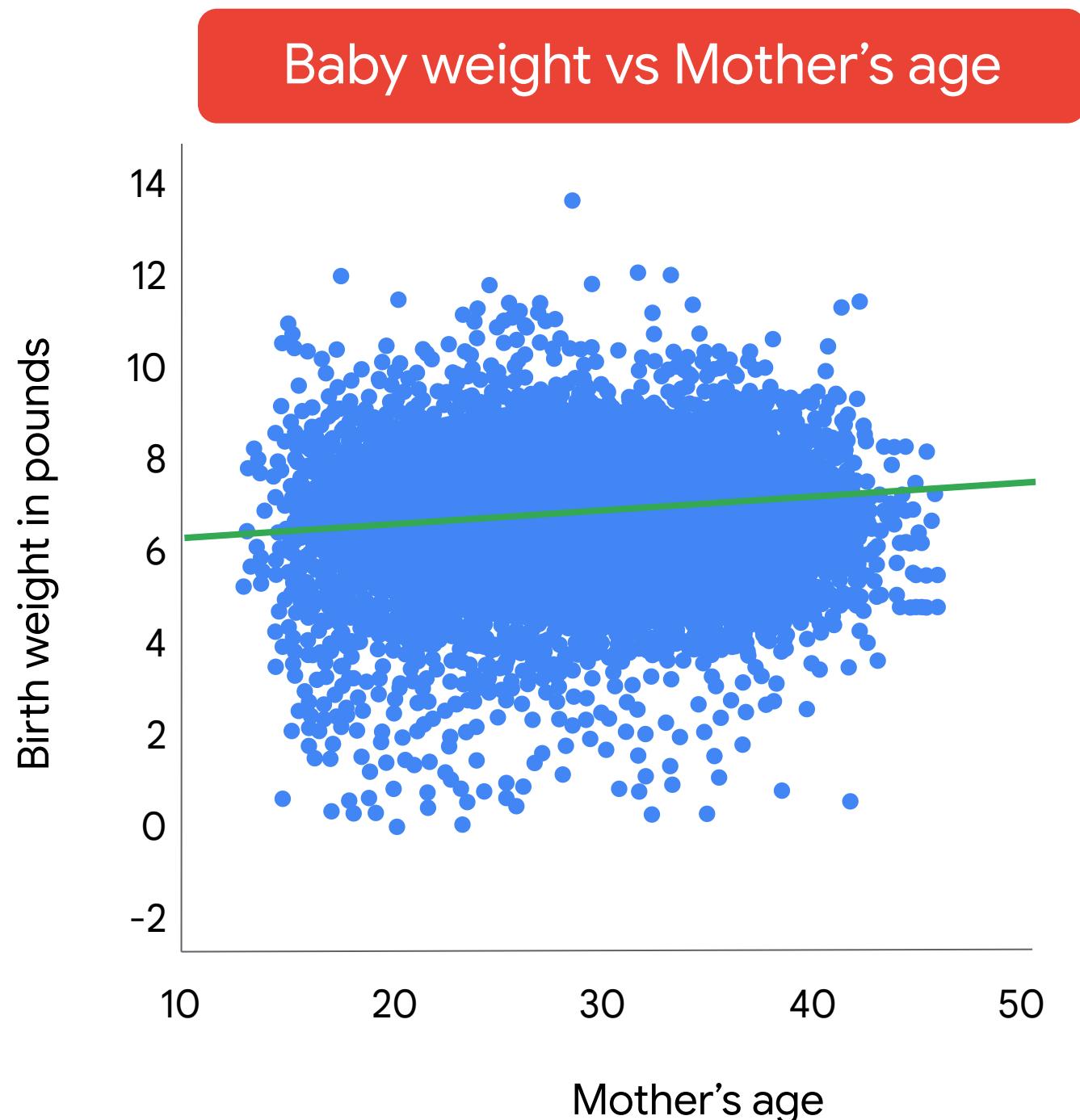
The slope of the line is given by w1.

$$y = w_1 x_1 + b$$

x_1 is the feature (e.g. mother's age)

w_1 is the weight for x_1

● $y = .02x + 6.83$



Equation for a linear model tying mother's age and baby weight

The slope of the line is given by w_1 .

$$y = w_1 x_1 + b$$

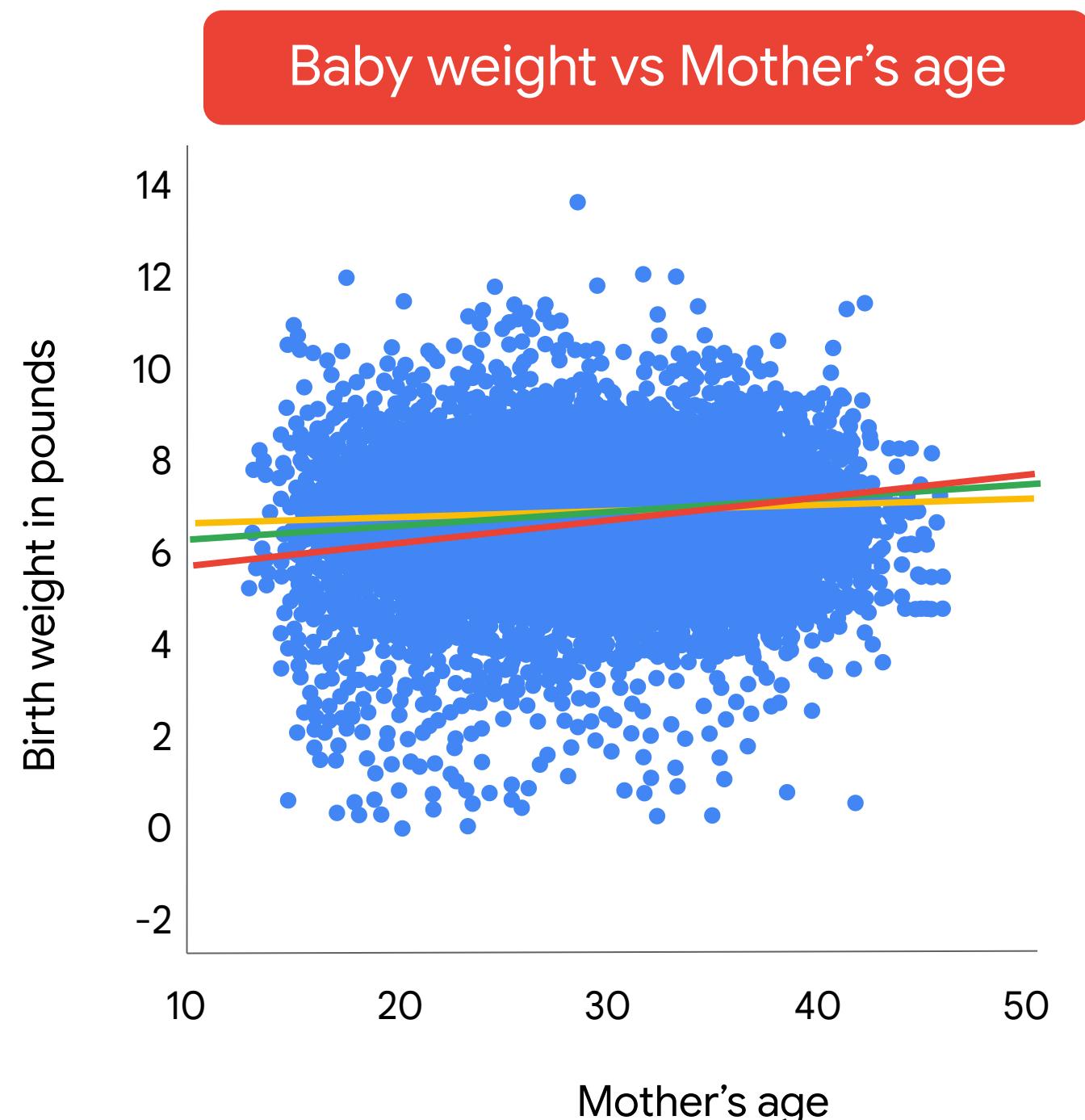
x_1 is the feature (e.g. mother's age)

w_1 is the weight for x_1

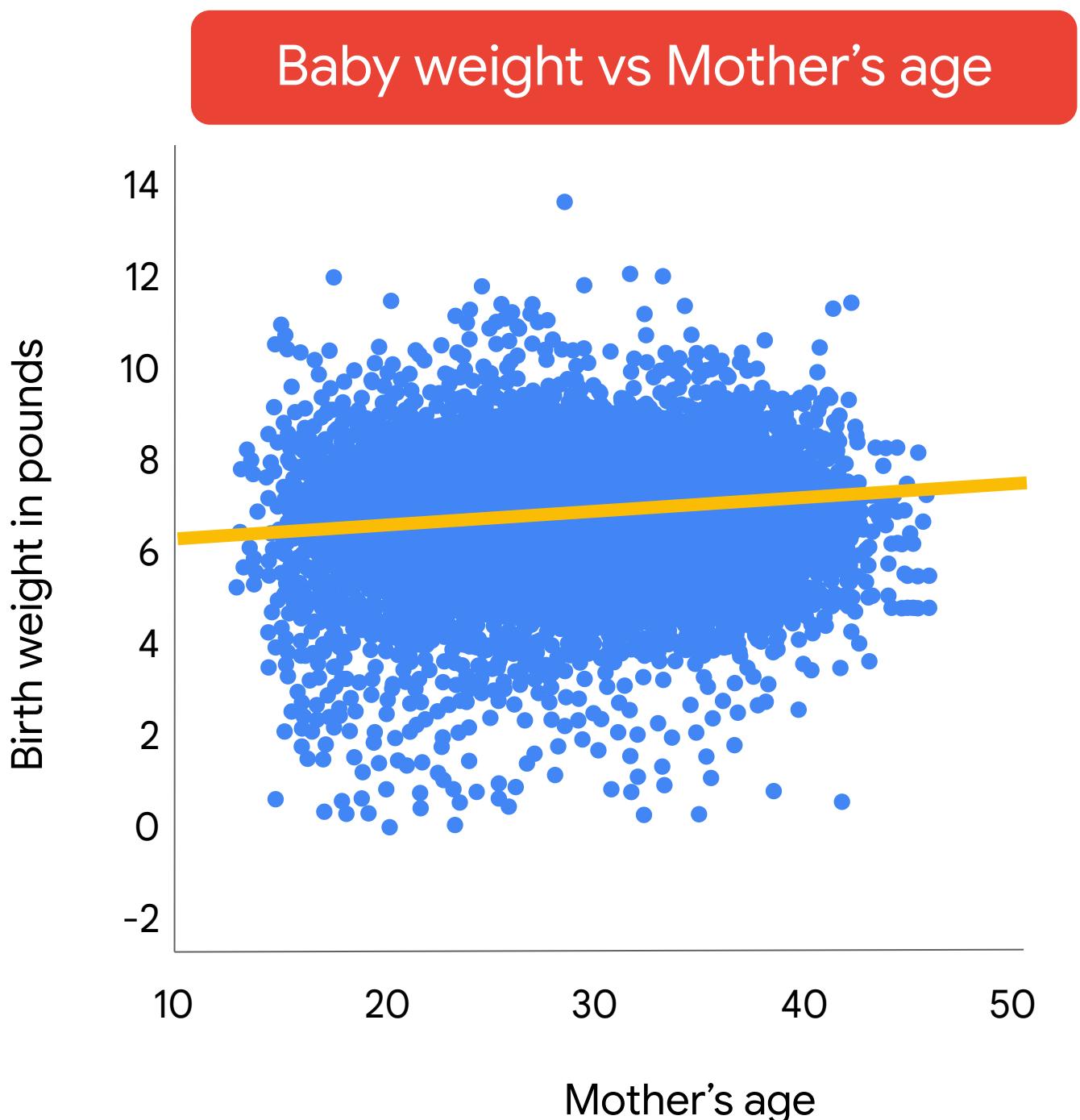
● $y = .02x + 6.83$

● $y = .03x + 6.49$

● $y = .01x + 7.14$

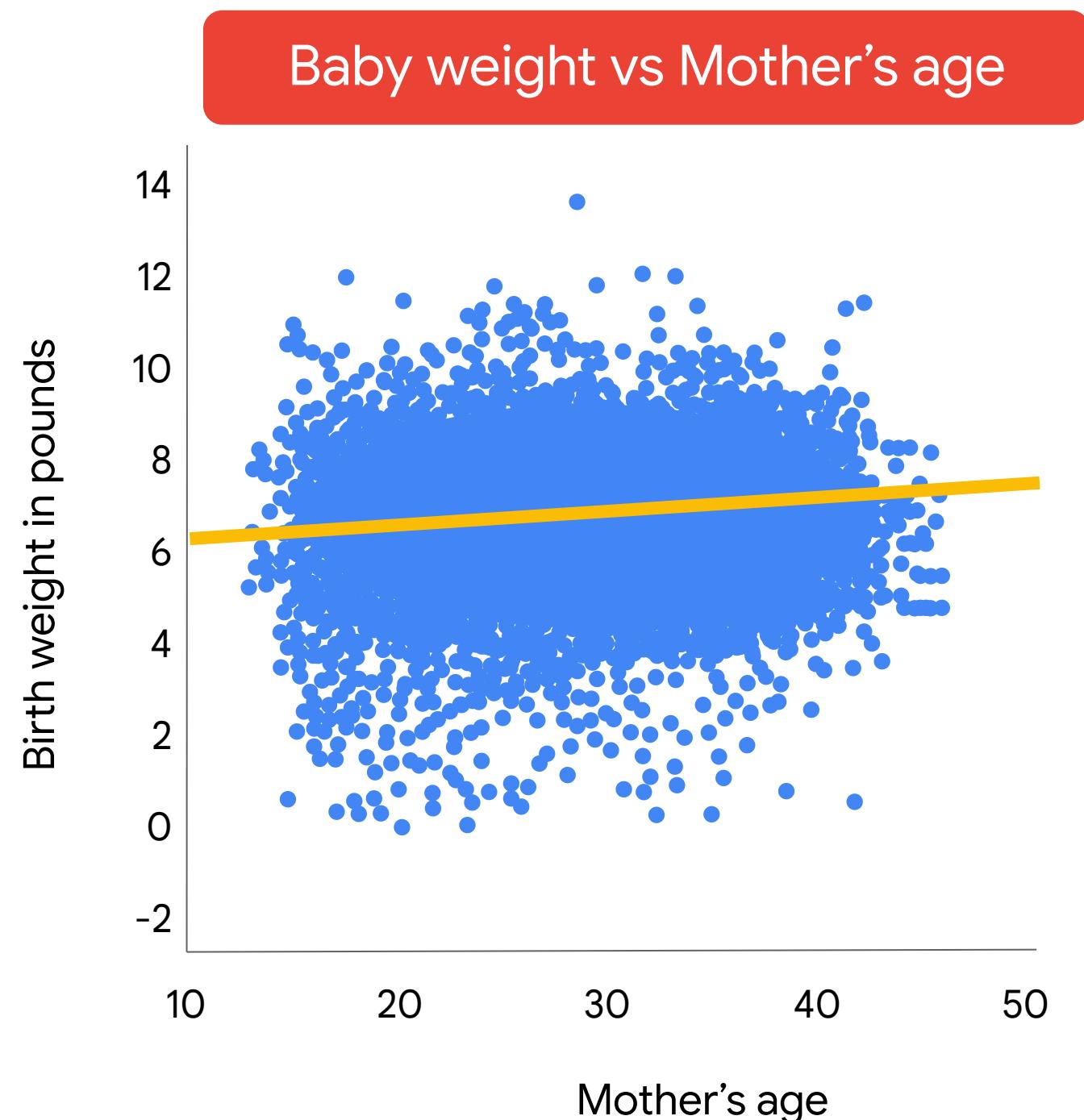


Can't we just solve the equation using all the data?

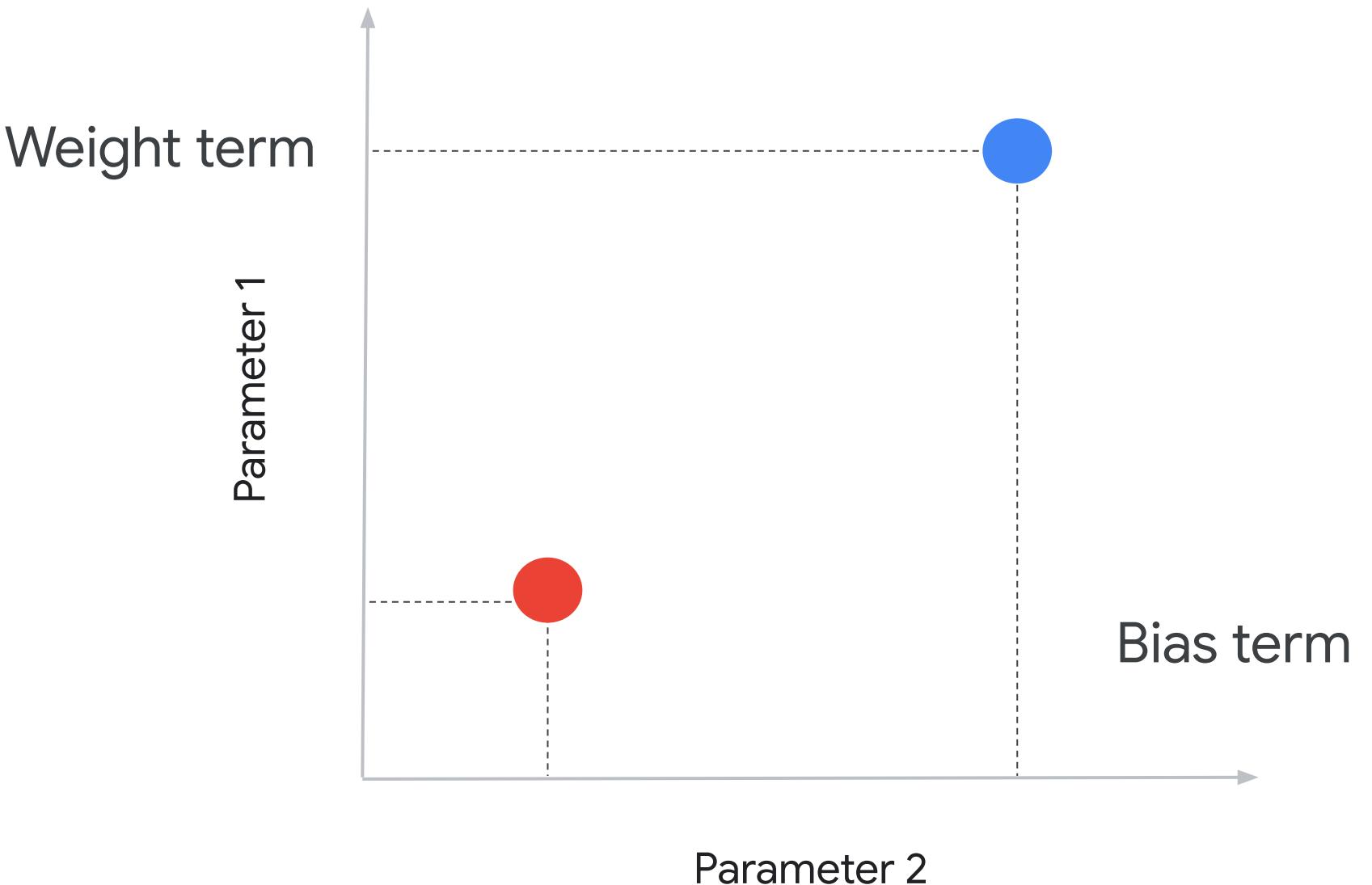


Can't we just solve the equation using all the data?

When an analytical solution is no longer an option, you use gradient descent.



Searching in parameter-space



- Models are defined as mathematical functions using **parameters** and **hyperparameters**.
- Analytical methods for finding the best set of **model parameters** don't scale.
- Think of optimizing your **parameters** as searching through parameter-space.

Loss functions

Compose a loss function by calculating errors

Each error makes sense. How about all the errors added together?

Error = actual (true) - predicted value

Compute the errors:

+0.70

+1.10

+0.65

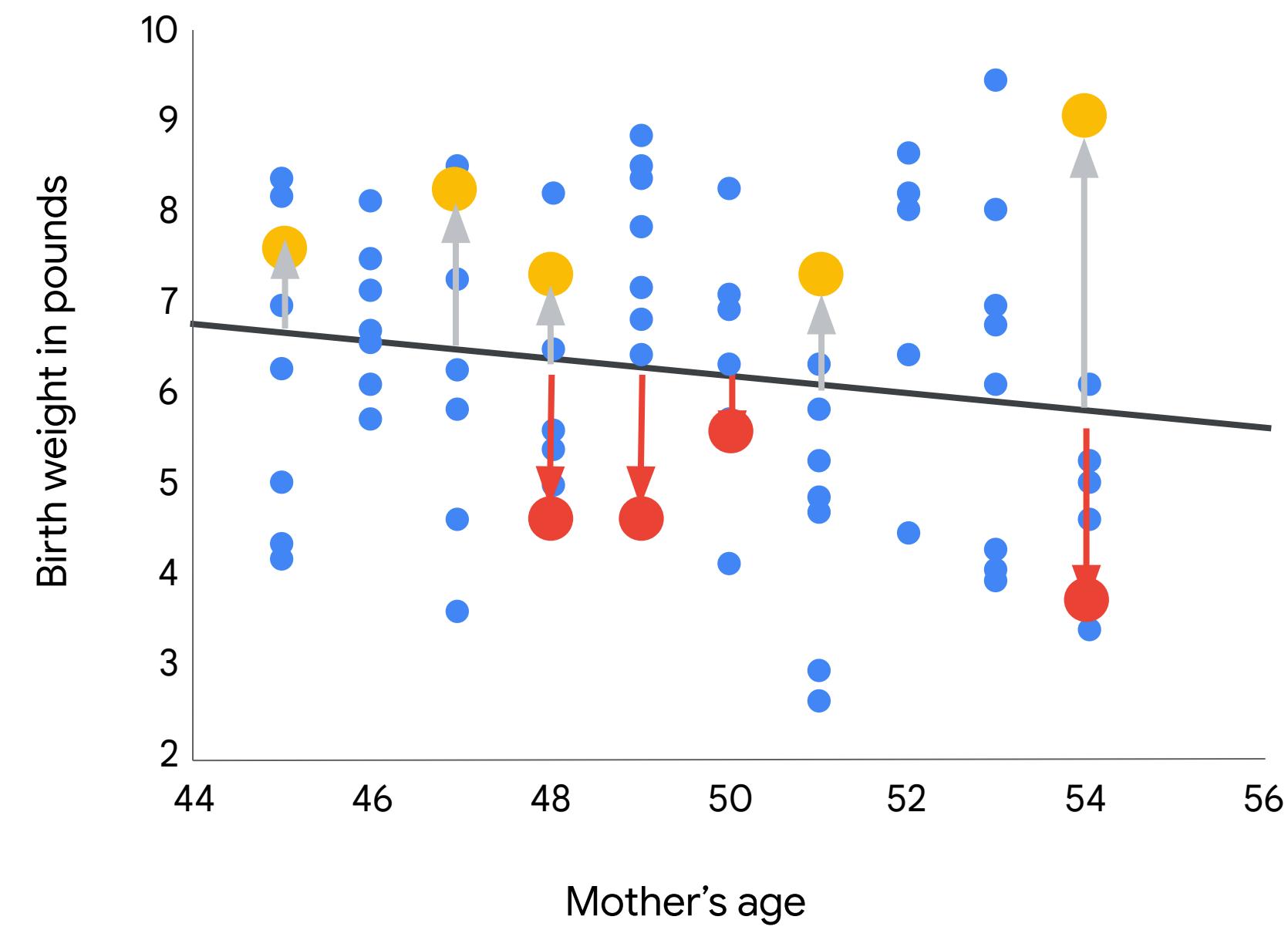
-1.20

-1.15

+1.10

+3.09

-2.10



One loss function metric is

Root Mean Squared Error (RMSE)

01 Get the errors for the training examples.

+0.70
+1.10
+0.65
-1.20
-1.15
+1.10
+3.09
-2.10

02 Compute the squares of the error values.

0.49
1.21
0.42
1.44
1.32
1.21
9.55
4.41

03 Compute the mean of the squared error values.
2.51

One loss function metric is

Root Mean Squared Error (RMSE)

01 Get the errors for the training examples.

+0.70
+1.10
+0.65
-1.20
-1.15
+1.10
+3.09
-2.10

02 Compute the squares of the error values.

0.49
1.21
0.42
1.44
1.32
1.21
9.55
4.41

03 Compute the mean of the squared error values.

2.51

04 Take a square root of the mean.

1.58

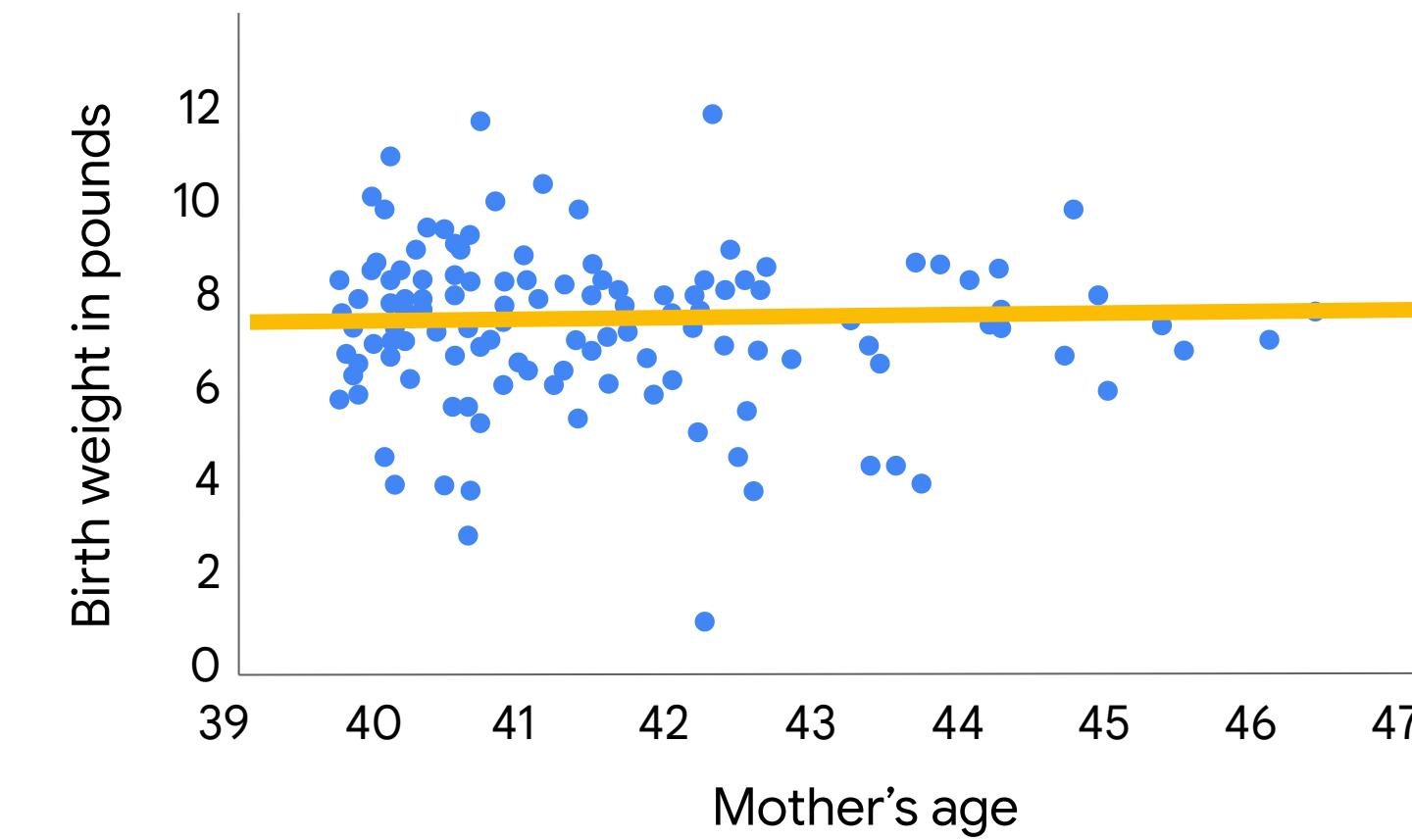
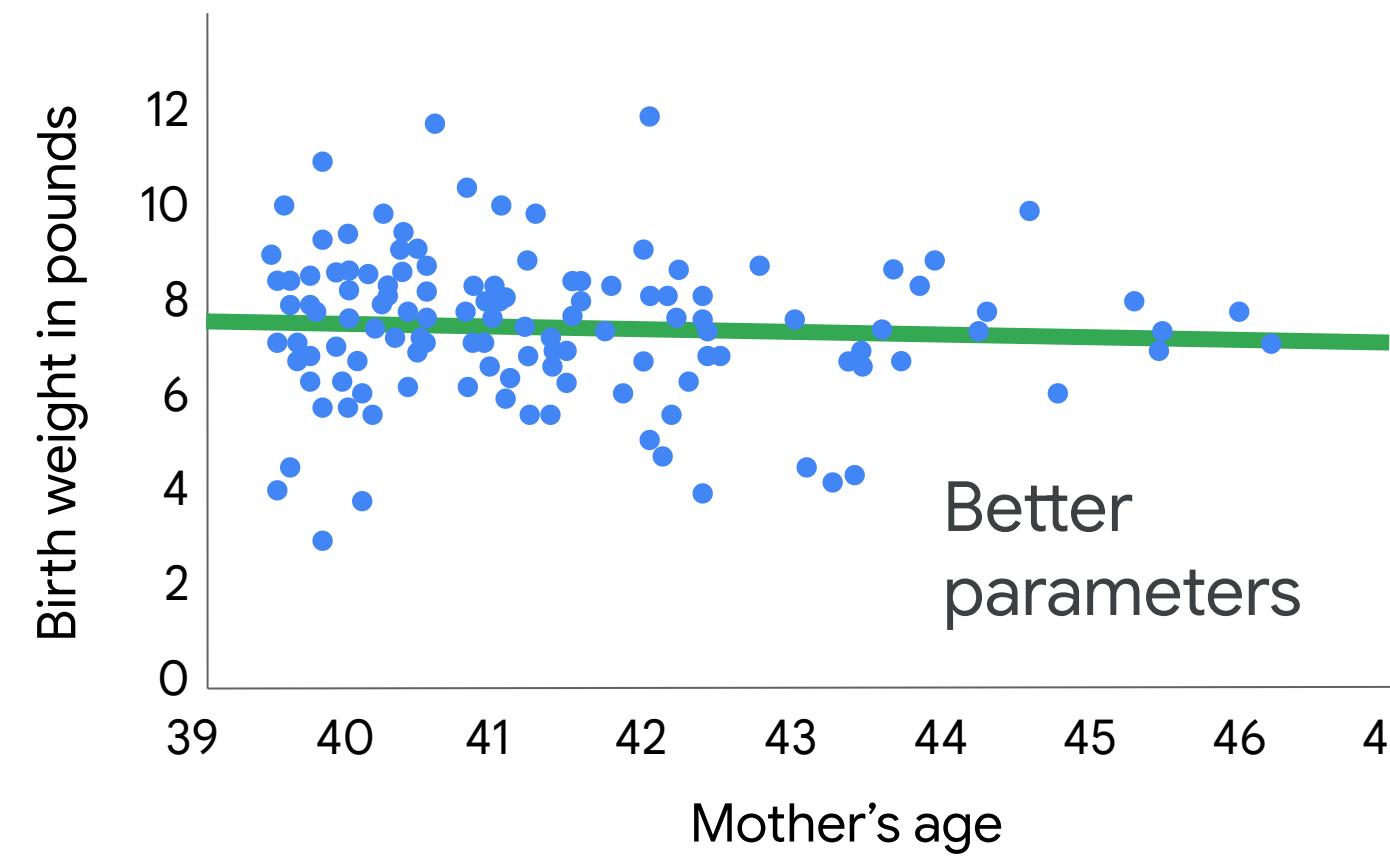
$$\sqrt{\frac{1}{n} * \sum_{i=1}^n (\hat{Y}_i - Y_i)^2}$$

\hat{Y}_i predicted value

\hat{Y}_i labeled value

Lower RMSE indicates a better performing model

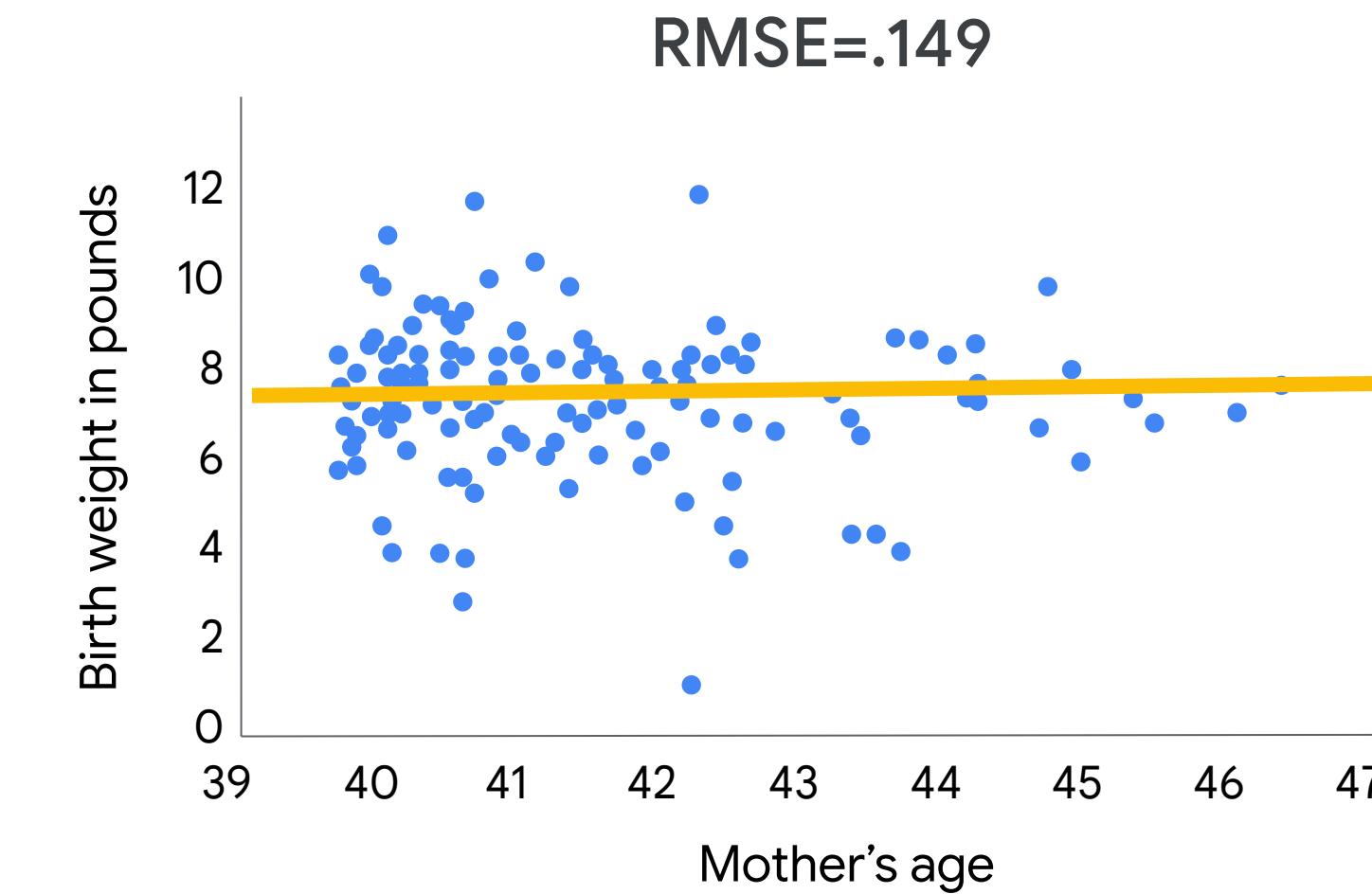
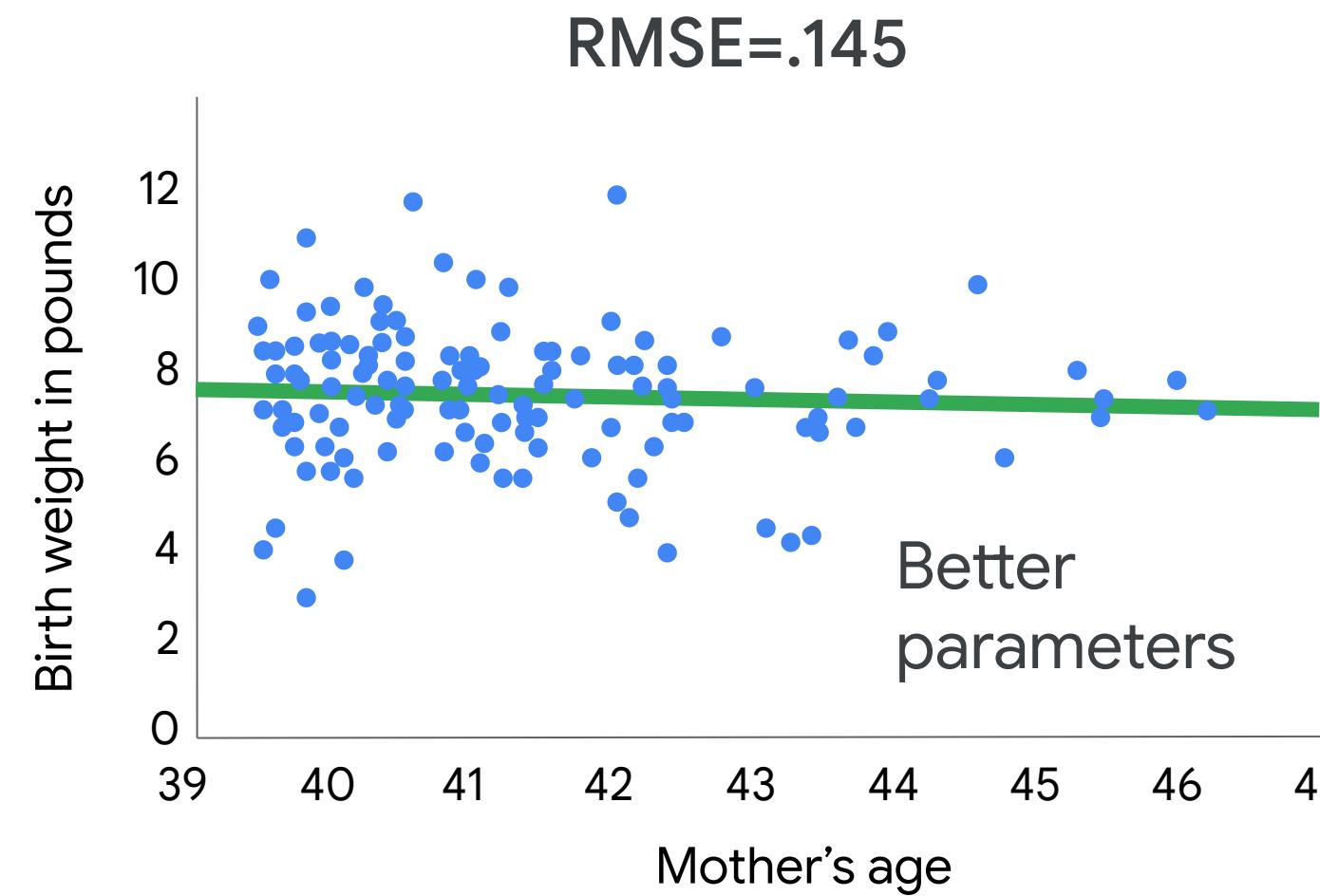
Baby weight vs Mother's age



We need a way to find the best values for weight and bias.

Lower RMSE indicates a better performing model

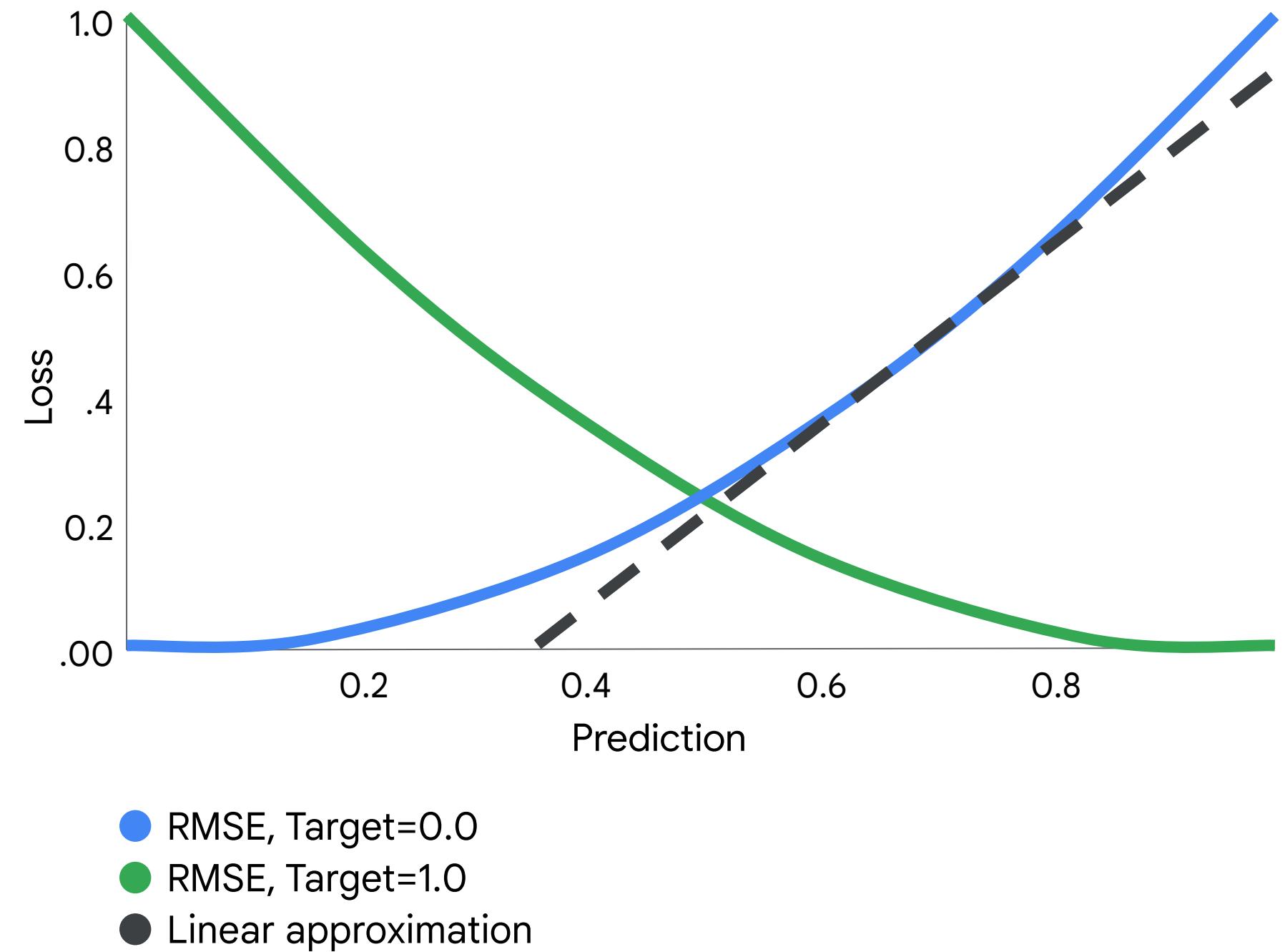
Baby weight vs Mother's age



We need a way to find the best values for weight and bias.

Problem: RMSE doesn't work as well for classification

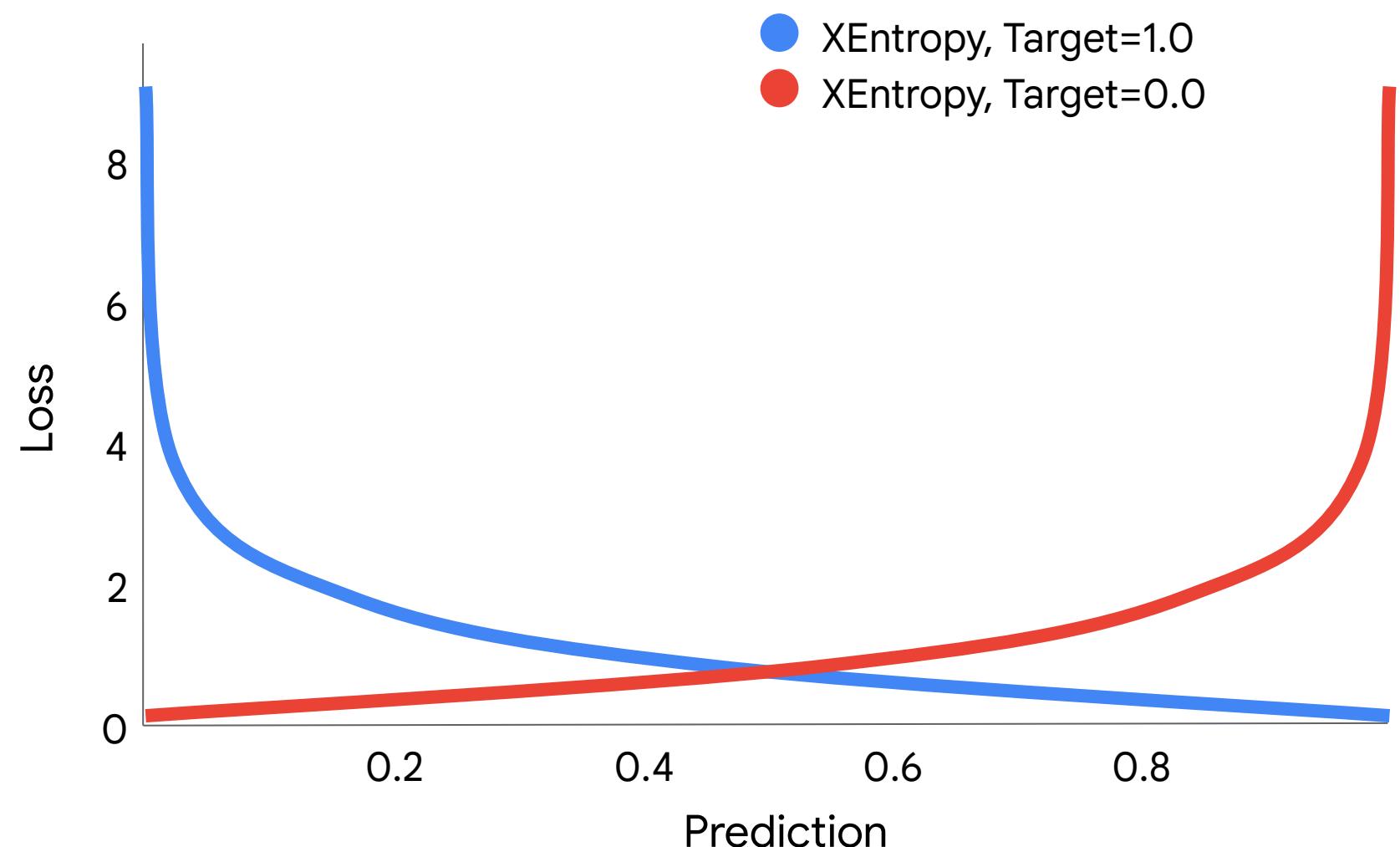
RMSE doesn't penalize bad
classifications appropriately.



$$\frac{-1}{N} \times \sum_{1}^{N} y_i \times \log(\hat{y}_i) + (1 - y_i) \times \log(1 - \hat{y}_i)$$

Problem: RMSE
doesn't work as well
for classification

Bad classifications are penalized
appropriately.



$$-\frac{1}{N} \times \sum_{i=1}^N y_i \times \log(\hat{y}_i) + (1 - y_i) \times \log(1 - \hat{y}_i)$$

Positive term Negative term

Computing
cross-entropy
loss

$$-\frac{1}{N} \times \sum_{i=1}^N y_i \times \log(\hat{y}_i) + (1 - y_i) \times \log(1 - \hat{y}_i)$$

Positive term Negative term

Computing cross-entropy loss

X	Y_i	\hat{Y}_i
	1	.7
	0	.2

$$\left. \begin{aligned} & (1.0 * \log(.7) + (1-1.0) * \log(1-.7)) \\ & + \\ & (0.0 * \log(.2) + (1-0.0) * \log(1-.2)) \end{aligned} \right\} * (-\frac{1}{2}) = .13$$

$$-\frac{1}{N} \times \sum_{i=1}^N y_i \times \log(\hat{y}_i) + (1 - y_i) \times \log(1 - \hat{y}_i)$$

Positive term Negative term

Computing cross-entropy loss

X	Y_i	\hat{Y}_i
	1	.7
	0	.2

$$\left. \begin{aligned} & (1.0 * \log(.7) + (1-1.0) * \log(1-.7)) \\ & + \\ & (0.0 * \log(.2) + (1-0.0) * \log(1-.2)) \end{aligned} \right\} * (-\frac{1}{2}) = .13$$

$$-\frac{1}{N} \times \sum_{i=1}^N y_i \times \log(\hat{y}_i) + (1 - y_i) \times \log(1 - \hat{y}_i)$$

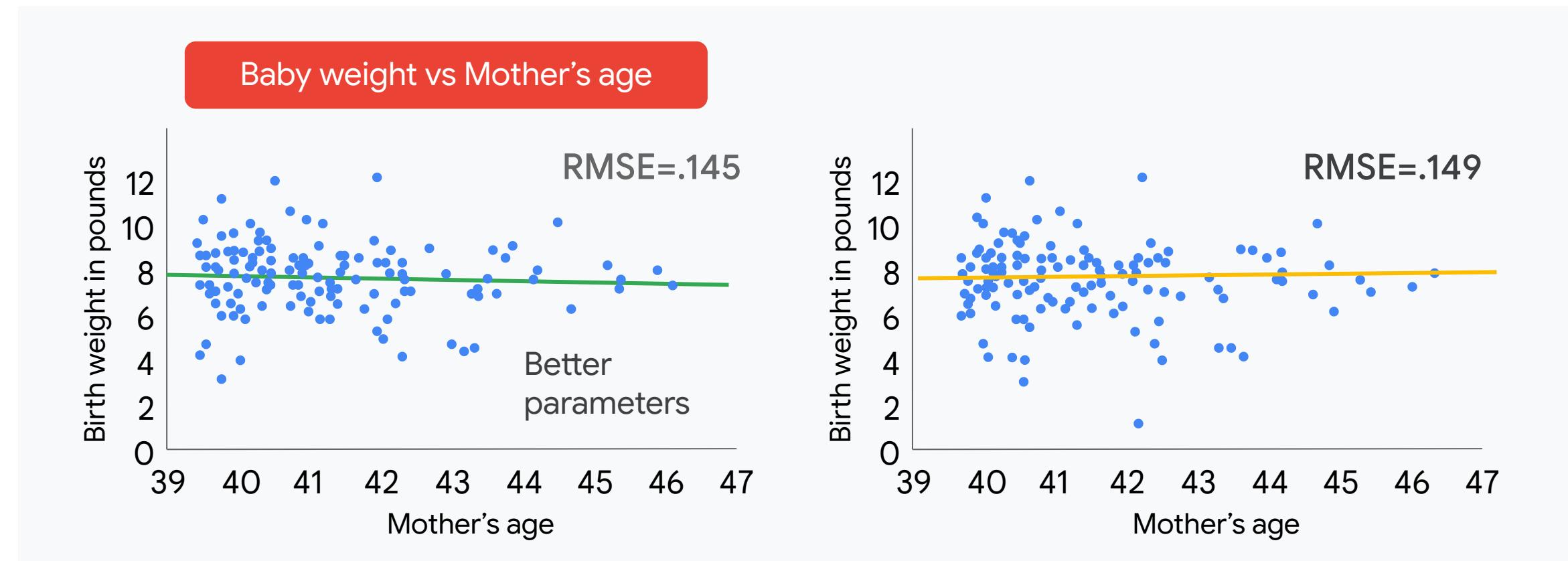
Positive term Negative term

Computing cross-entropy loss

X	Y_i	\hat{Y}_i
	1	.7
	0	.2

$$\left\{ \begin{array}{l} (1.0 * \log(.7) + (1-1.0) * \log(1-.7)) \\ + \\ (0.0 * \log(.2) + (1-0.0) * \log(1-.2)) \end{array} \right\} * (-\frac{1}{2}) = .13$$

From loss functions to gradient descent



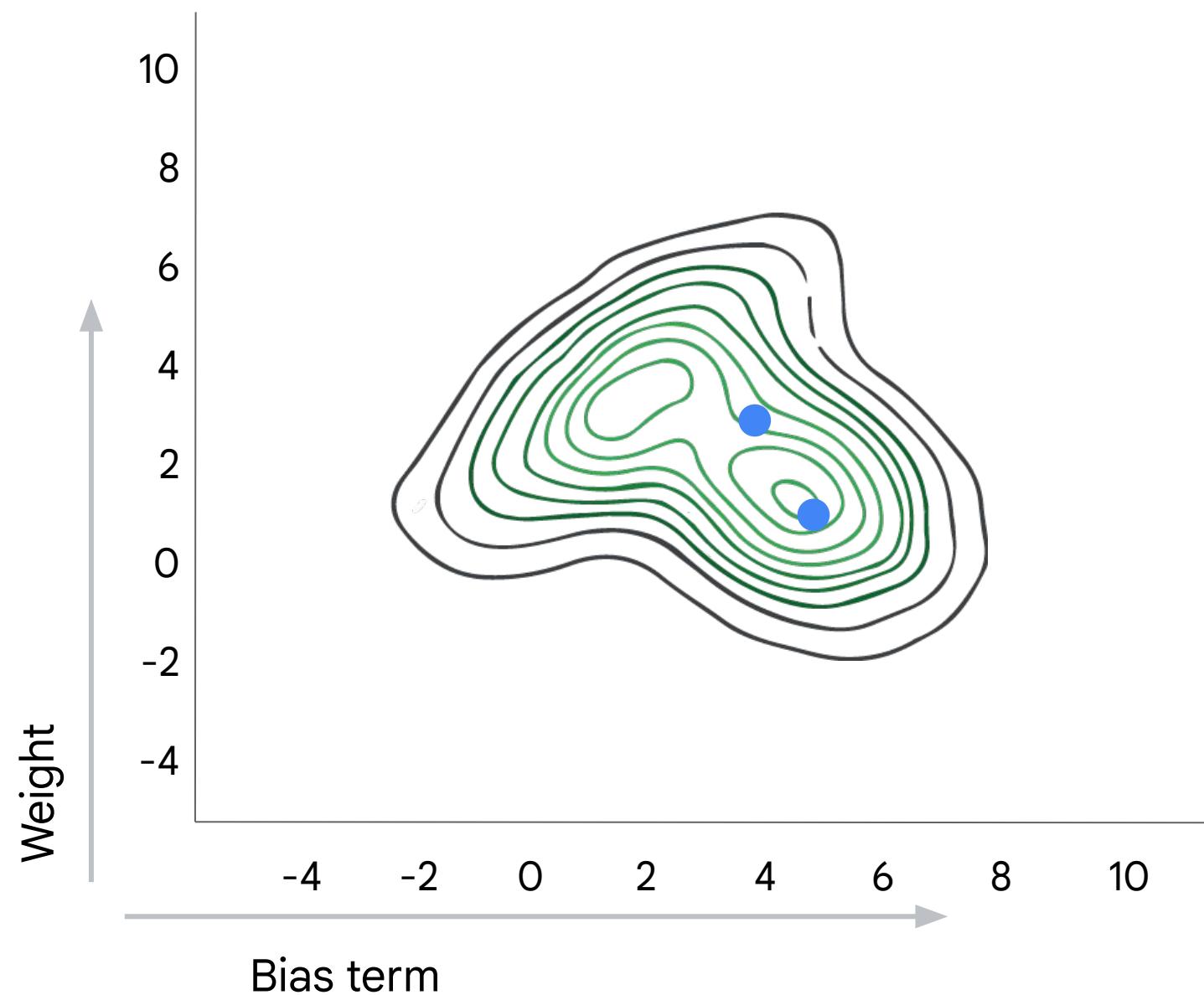
X	Y_i	\hat{Y}_i
	1	.7
	0	.2

$$\frac{-1}{N} \times \sum_{1}^{N} \underbrace{y_i \times \log(\hat{y}_i)}_{\text{Positive term}} + \underbrace{(1 - y_i) \times \log(1 - \hat{y}_i)}_{\text{Negative term}}$$
$$\left((1.0 * \log(.7) + (1-1.0) * \log(1-.7)) + (0.0 * \log(.2) + (1-0.0) * \log(1-.2)) \right) * (-\frac{1}{2}) = .13$$

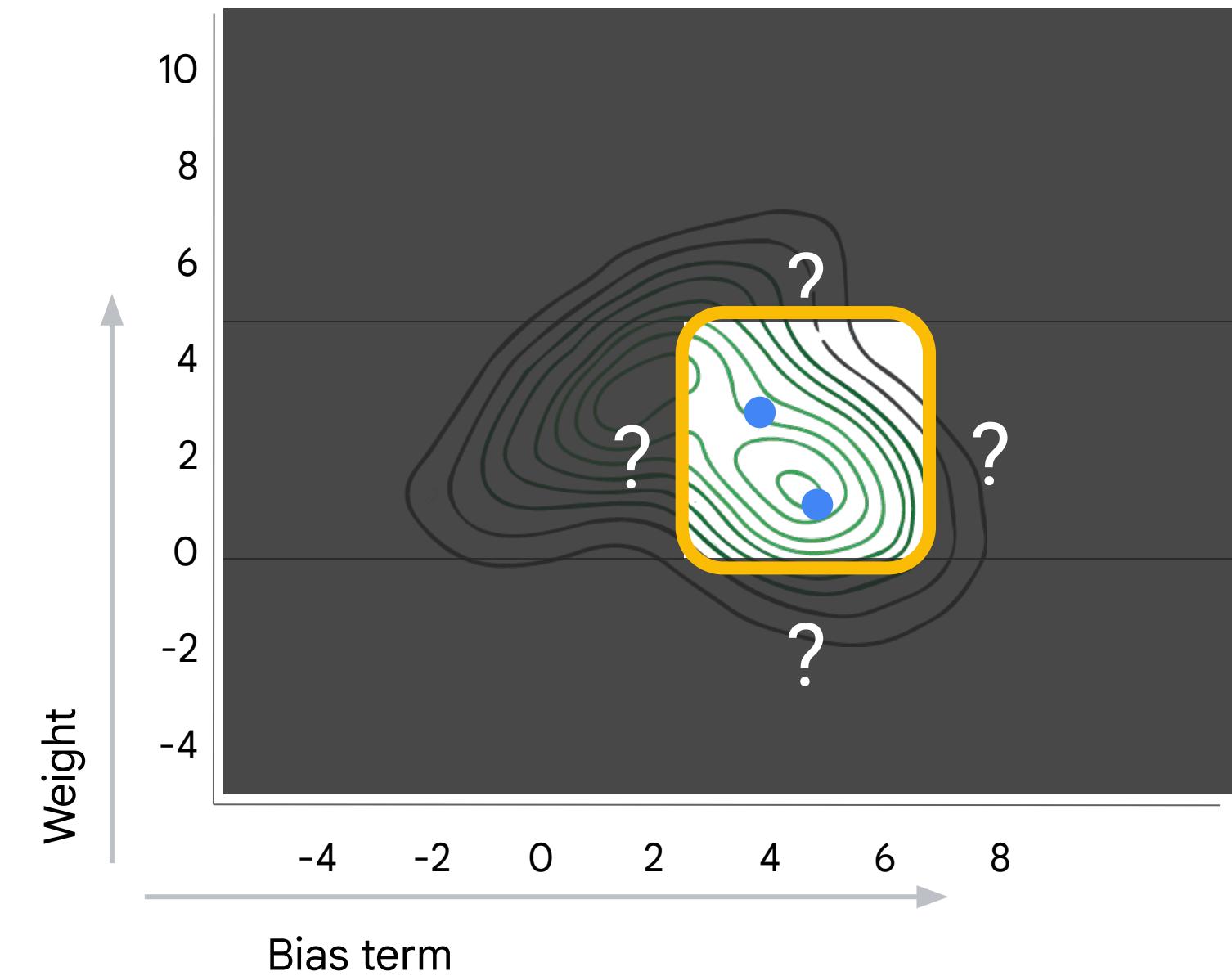
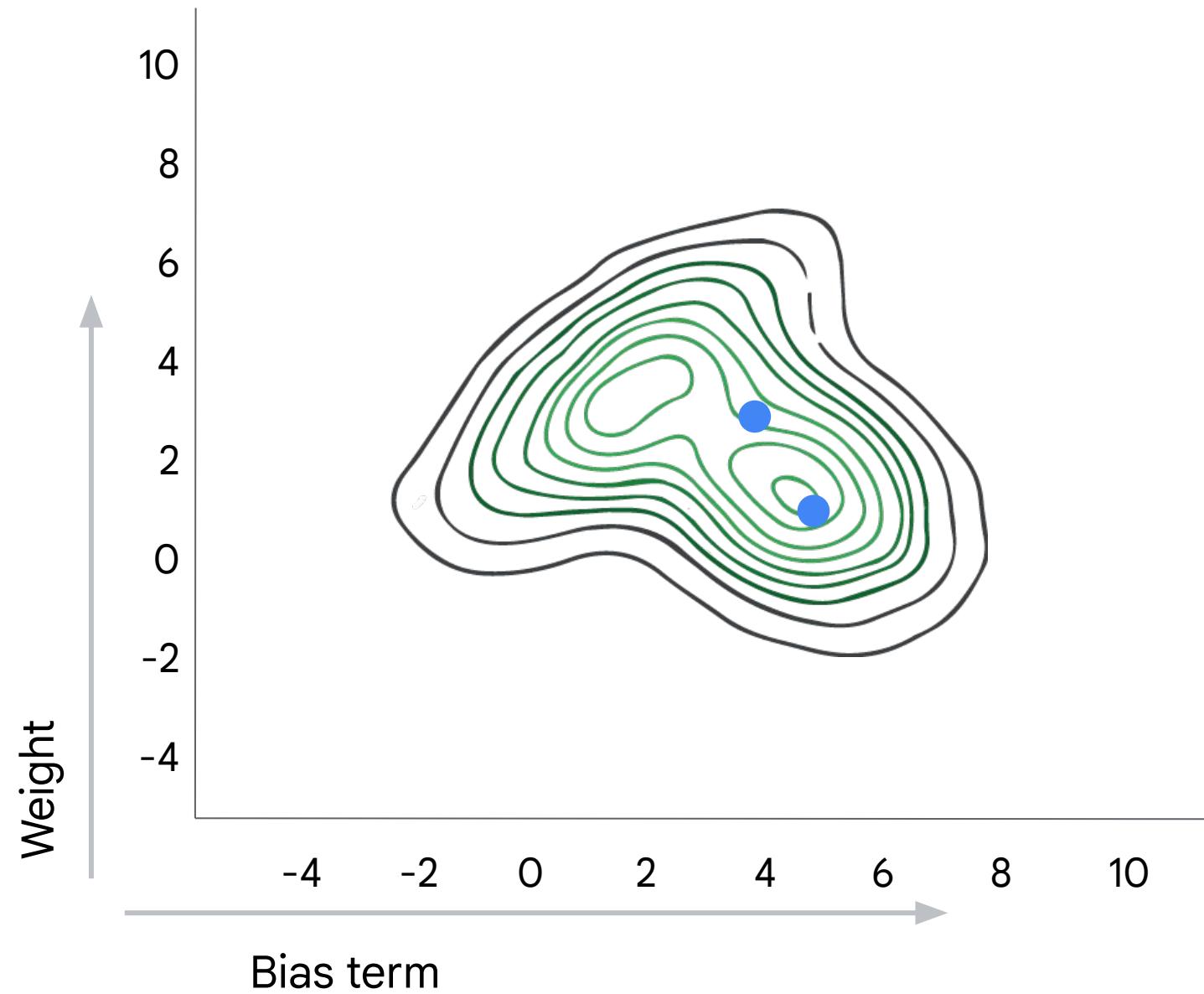
- Optimization was framed as a search in parameter-space.
- Loss functions were introduced as a way to compare these points.

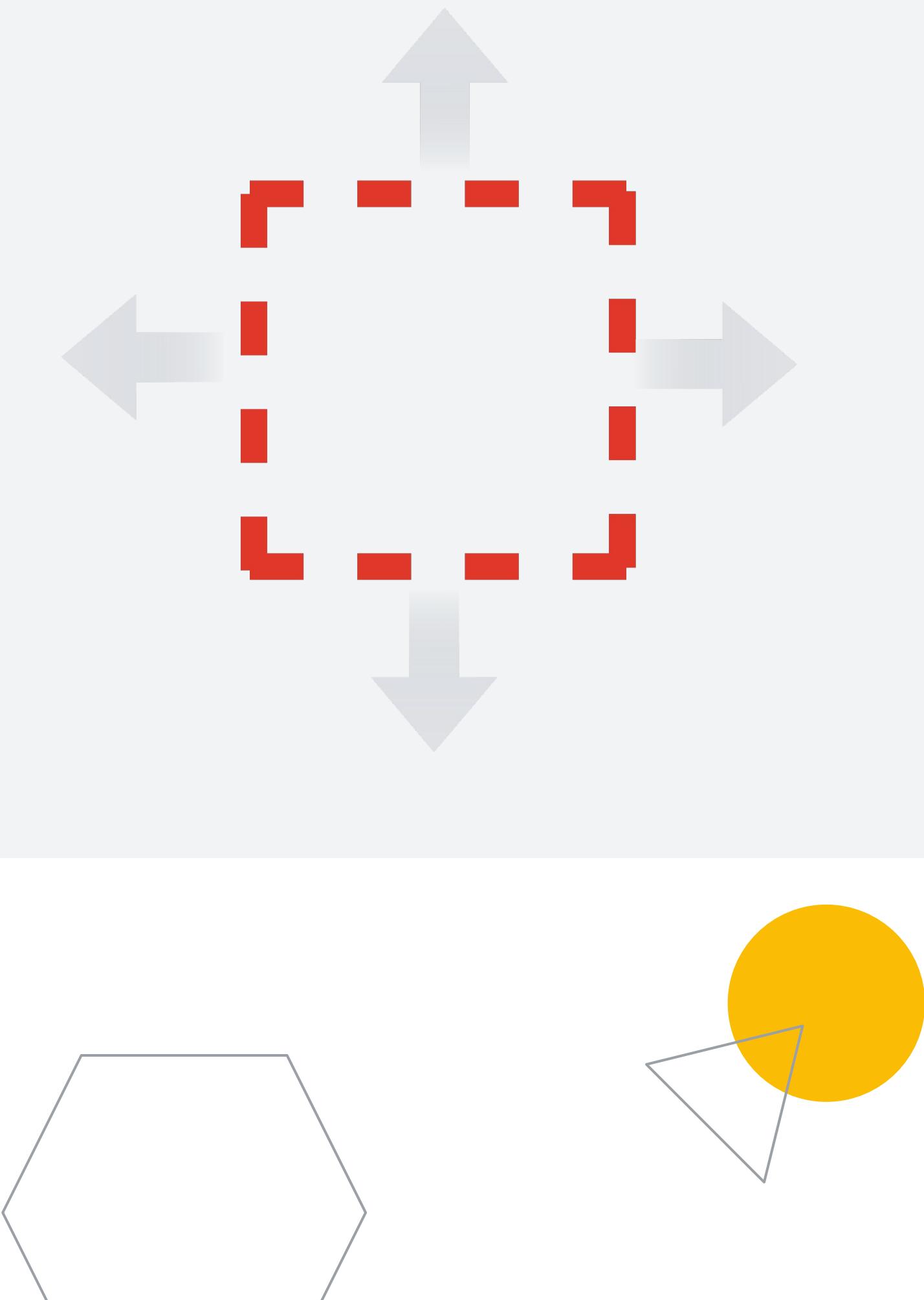
Gradient descent

Loss functions lead to loss surfaces



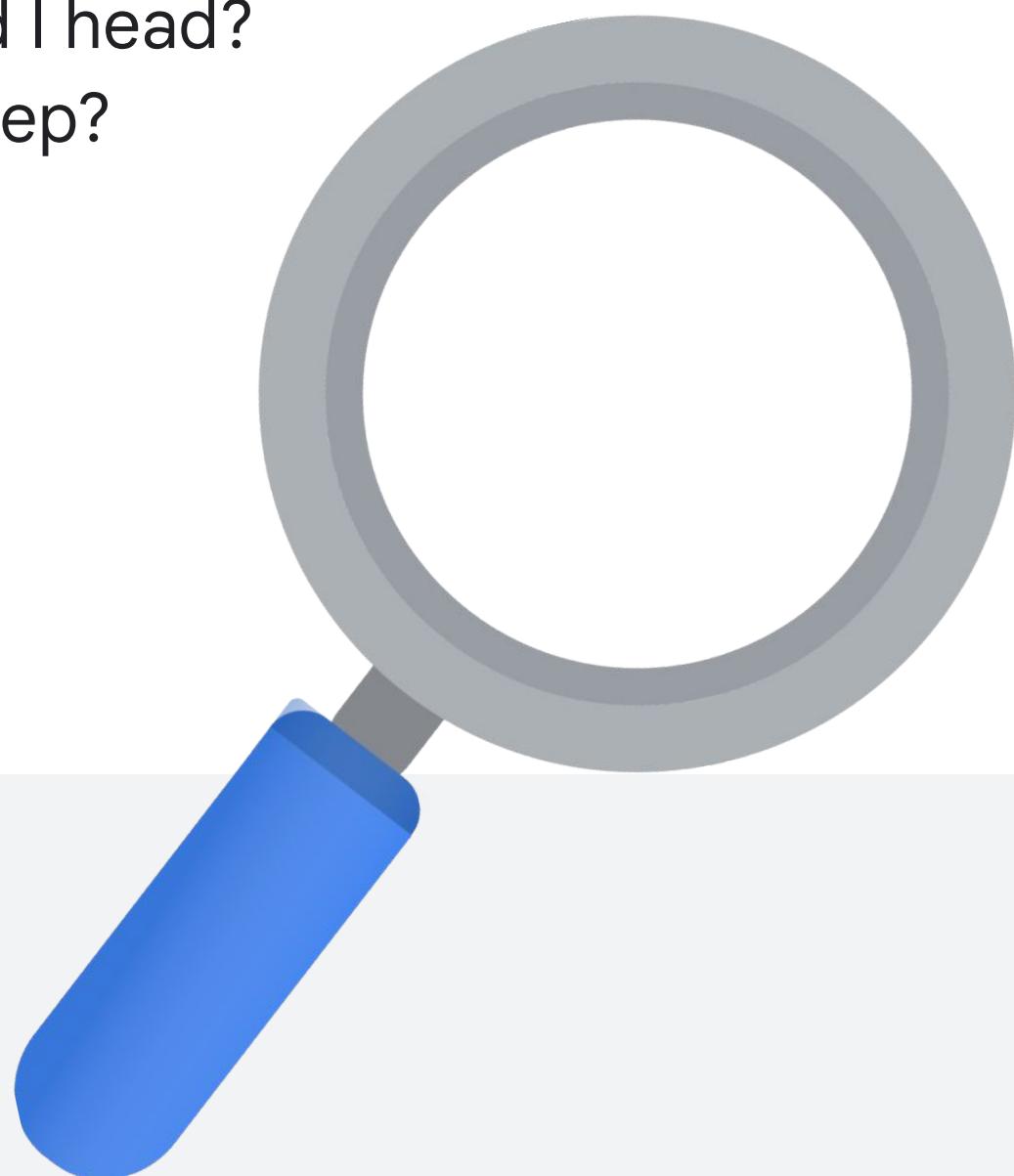
Loss functions lead to loss surfaces





Finding the bottom

Which direction should I head?
How large or small a step?

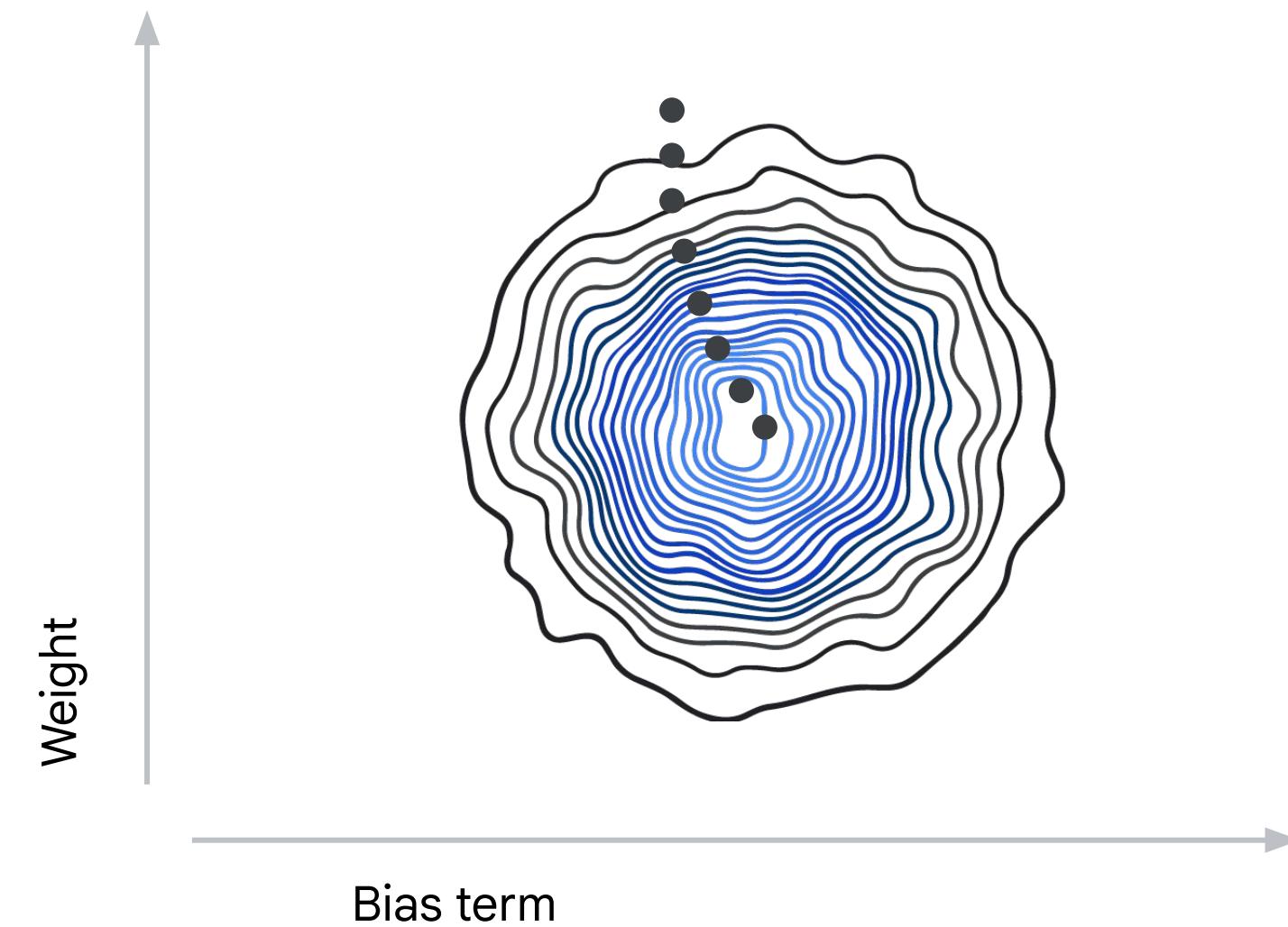


A simple algorithm to find the minimum

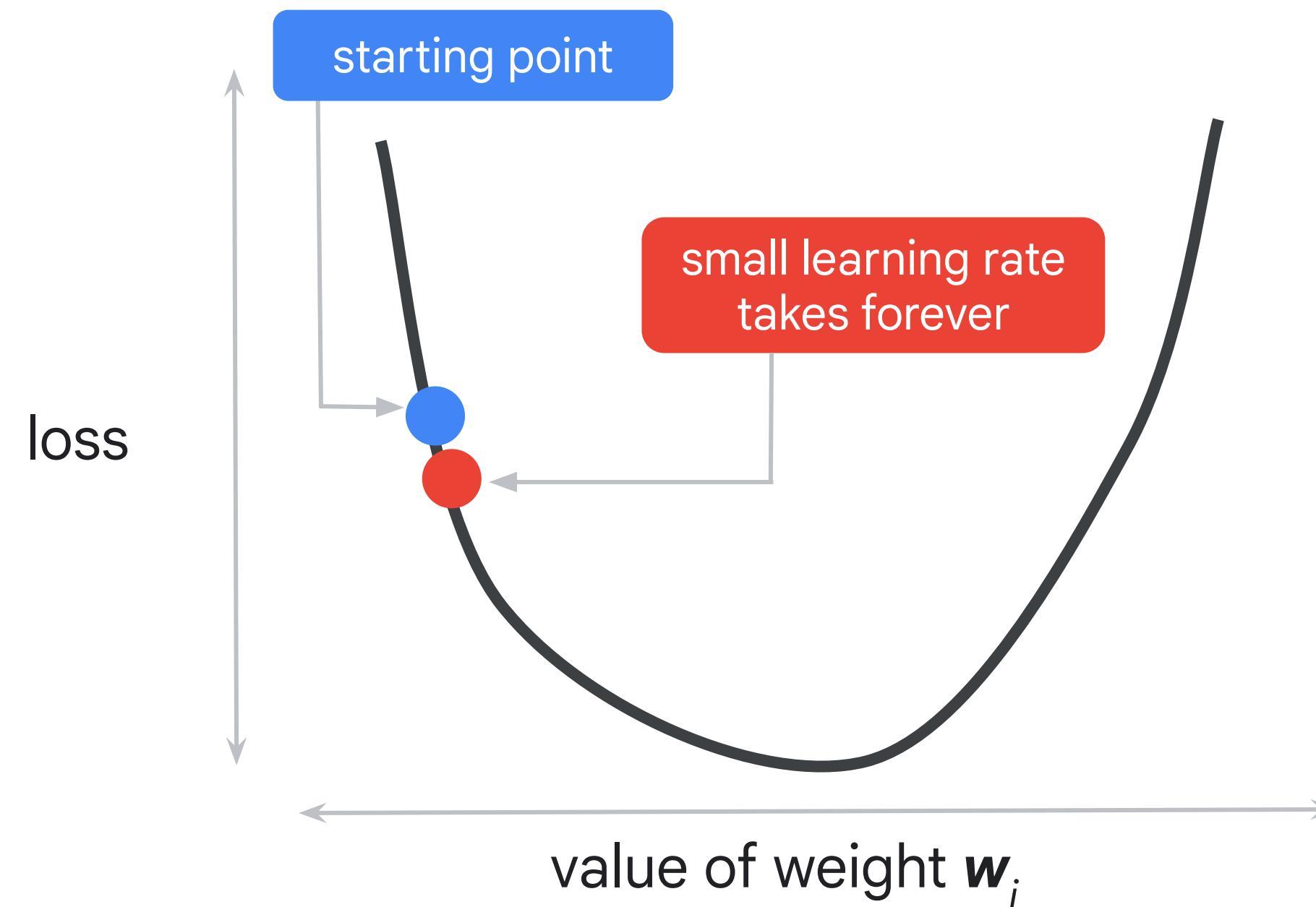
```
while loss is > Epsilon:  
    direction = computeDirection()  
    for i in range(self.params):  
        self.params[i] = //  
        self.params[i] //  
        + stepSize * direction[i]  
    loss = computeLoss()
```

Epsilon = A tiny constant

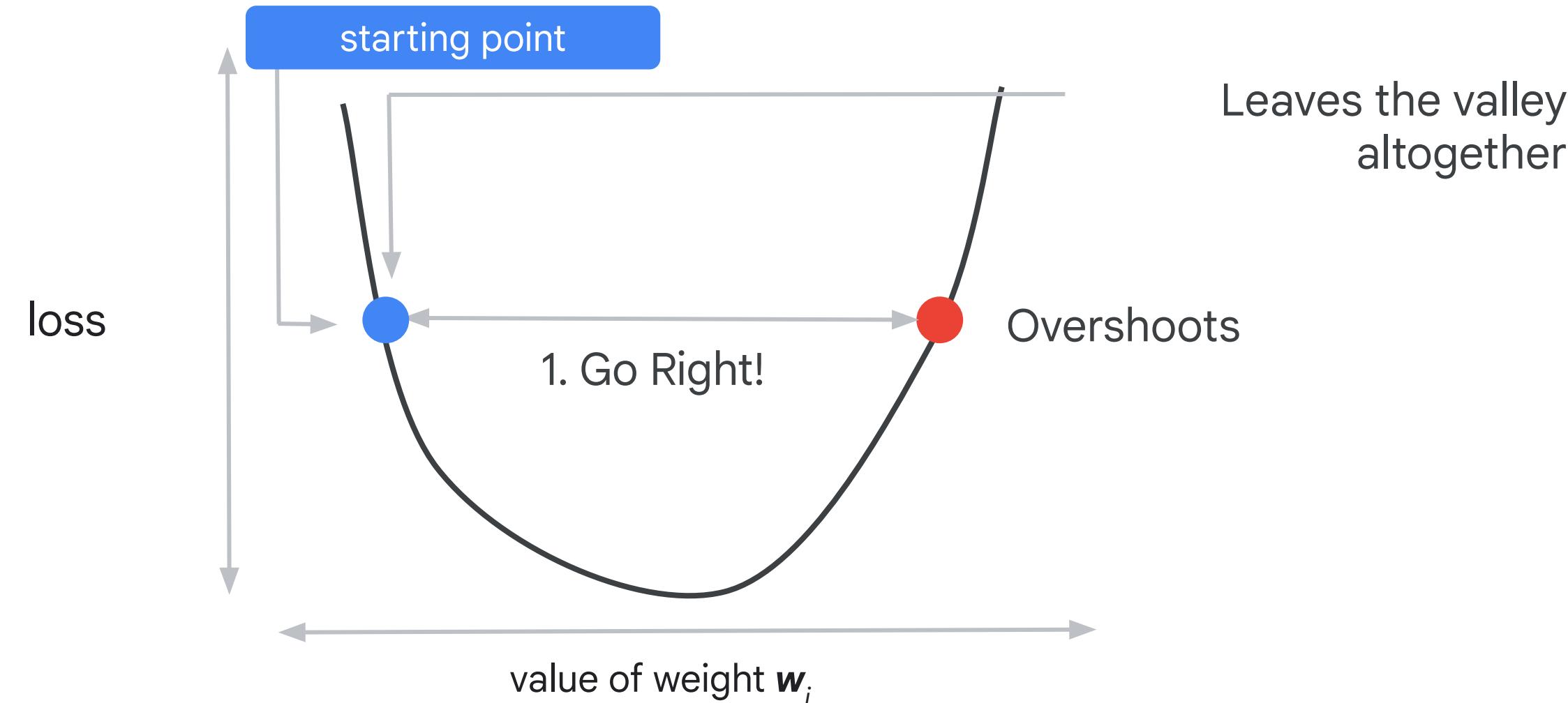
Search for a minima by
descending the **gradient**



Small step sizes can take a very long time to converge

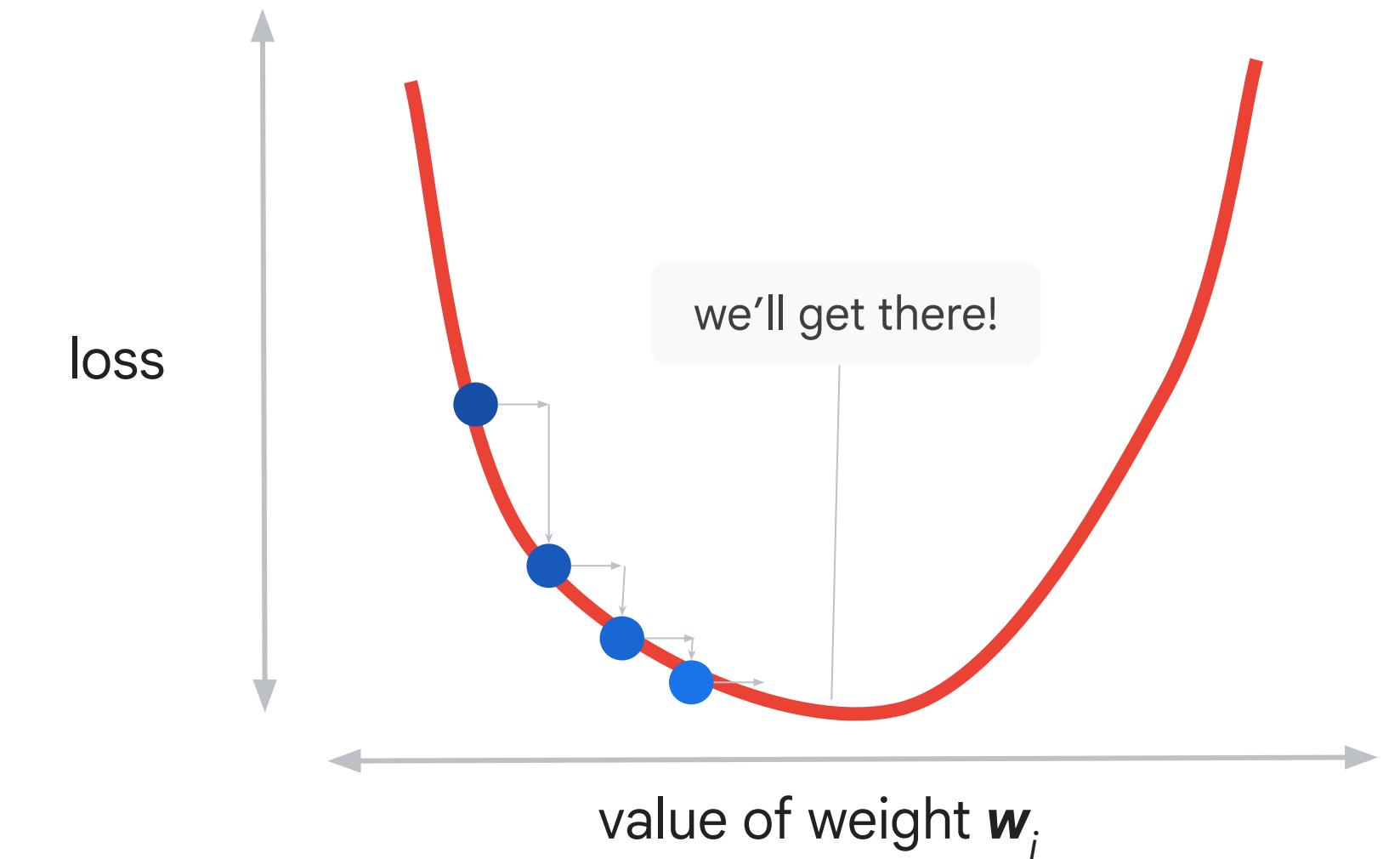


Large step sizes may never converge to the true minimum

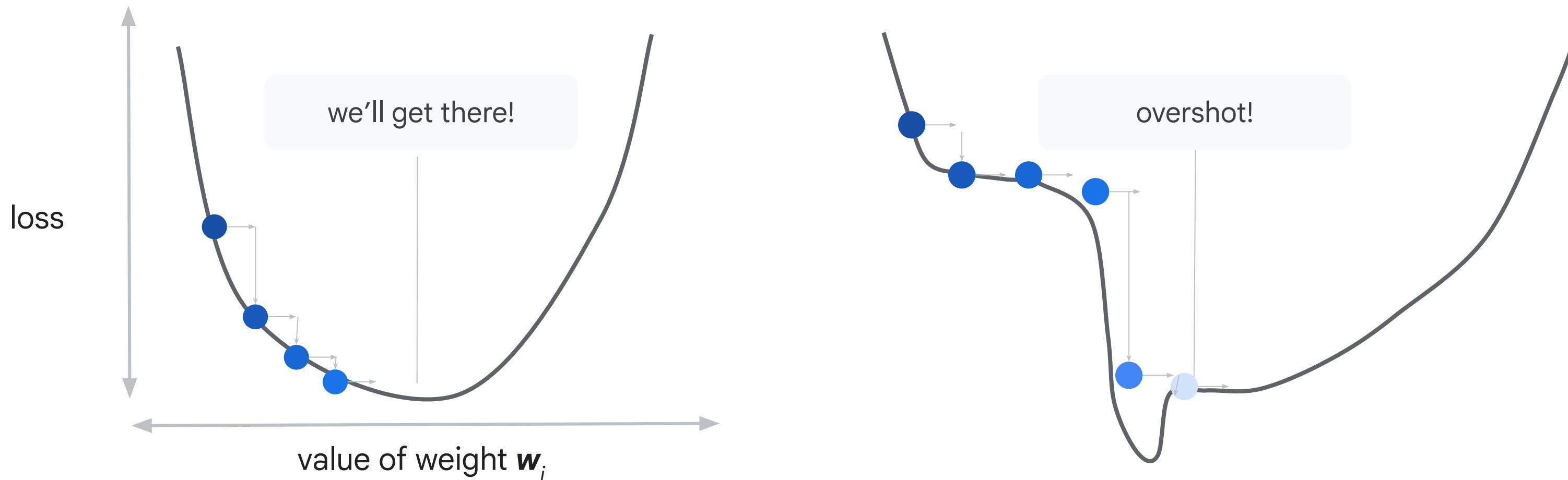


A correct and constant step size can be difficult to find

Step size or “learning rate” is a hyper-parameter which is set before training.



A correct and constant step size can be difficult to find



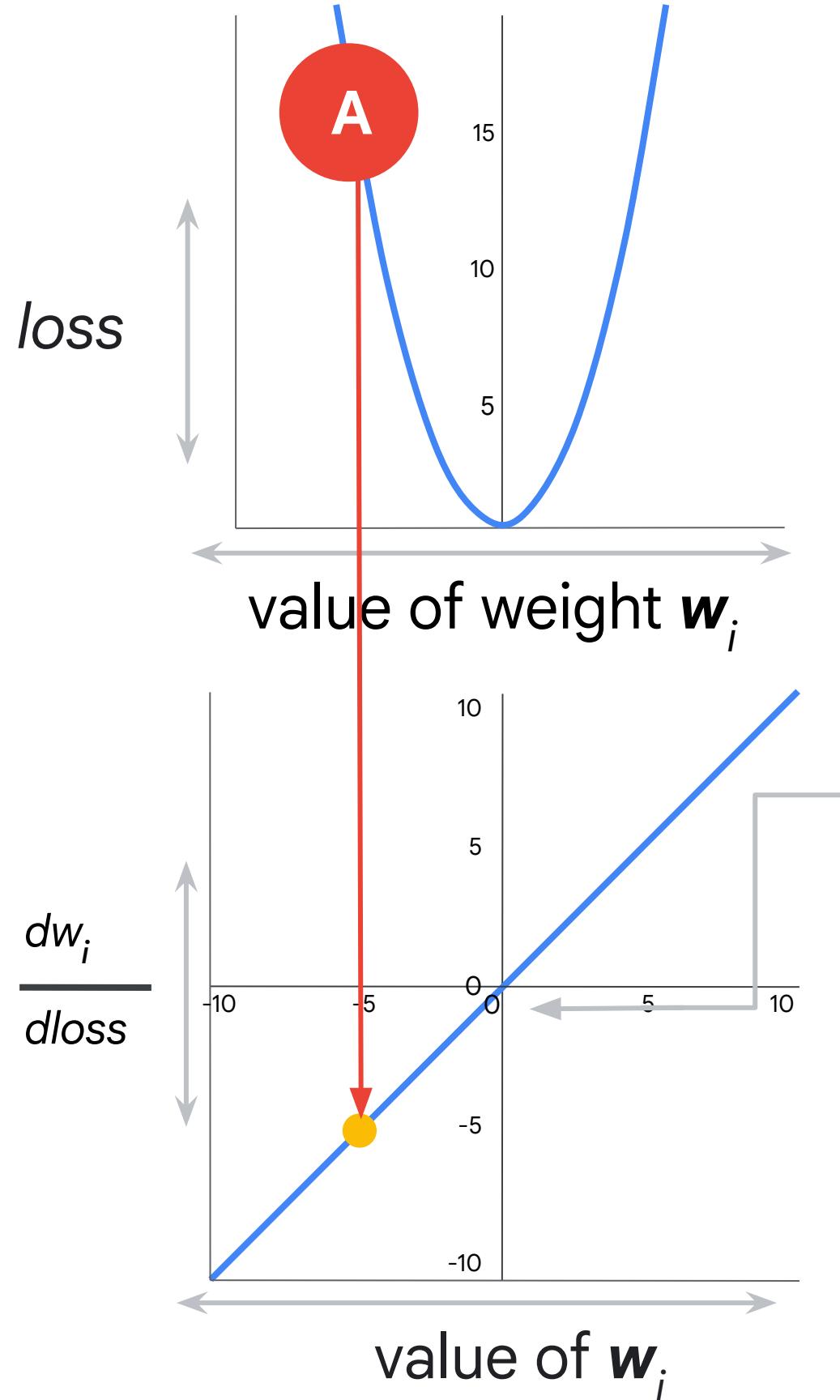
Step size or “learning rate” is a hyper-parameter which is set before training.

One size does not fit all models.

The loss function slope provides direction and step size in your search

Slope is negative
Direction: Go right!

Magnitude is (-5)
Step Size: Big



Loss function
(e.g., RMSE)

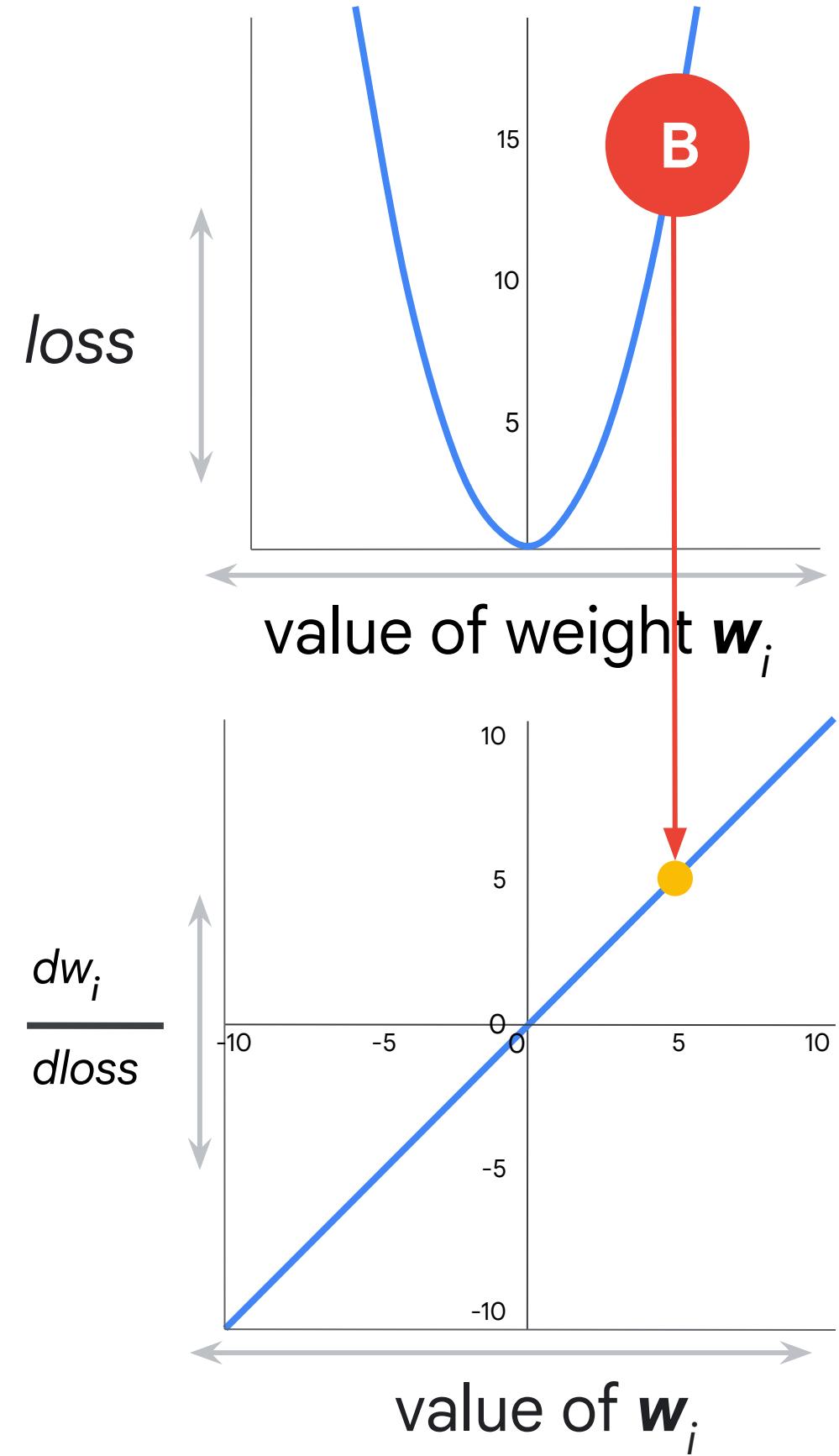
Remember, your goal is to find the minimum loss which is where the Loss Function slope is 0.

Loss function
slope
(derivative)

The loss function slope provides direction and step size in your search

Slope is positive
Direction: Go left!

Magnitude is 5
Step Size: Big



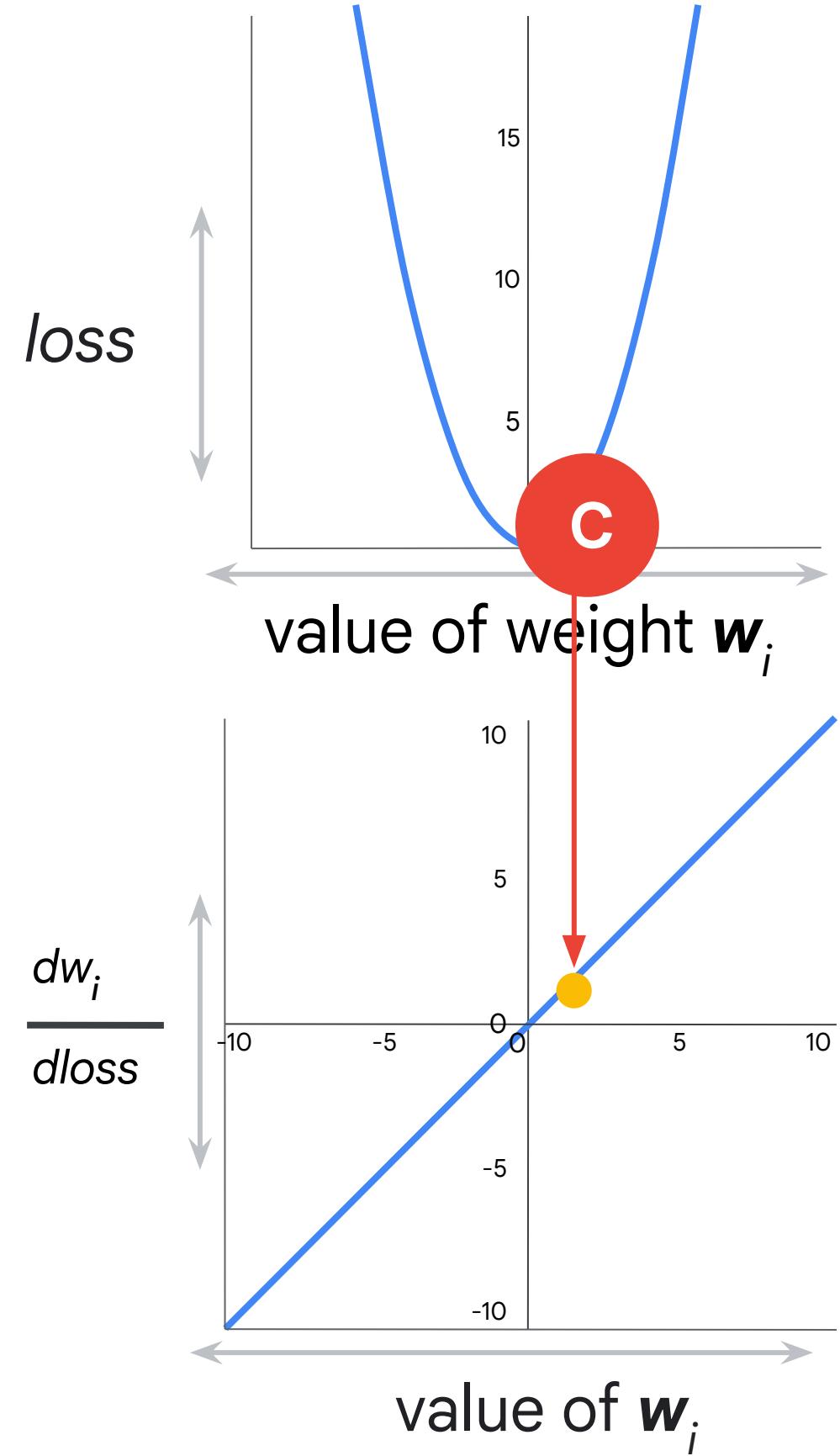
Loss function (e.g., RMSE)

Loss function slope (derivative)

The loss function slope provides direction and step size in your search

Slope is positive direction:
Go left!

Magnitude is 2
Step size

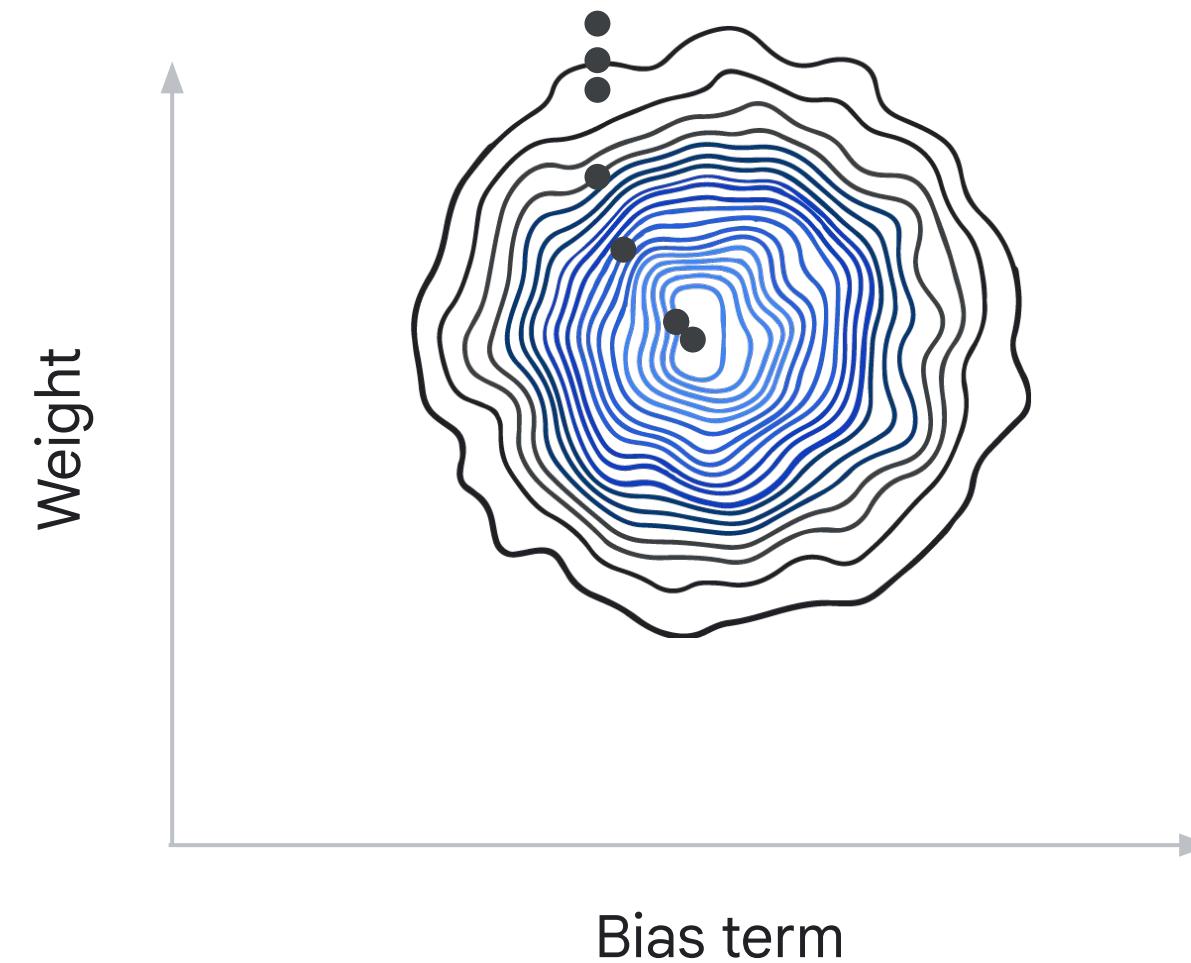


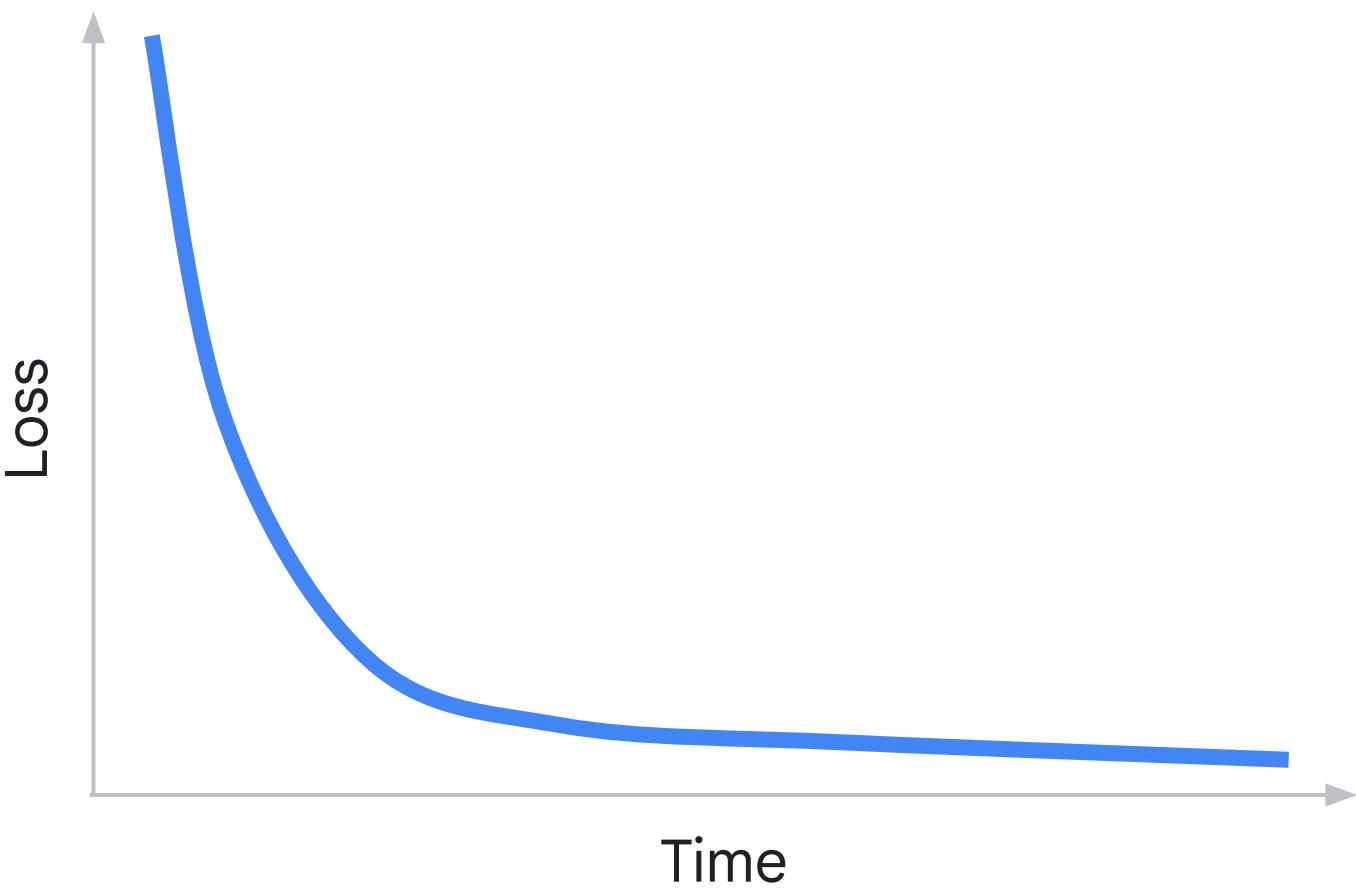
Loss function
(e.g., RMSE)

Loss function
slope
(derivative)

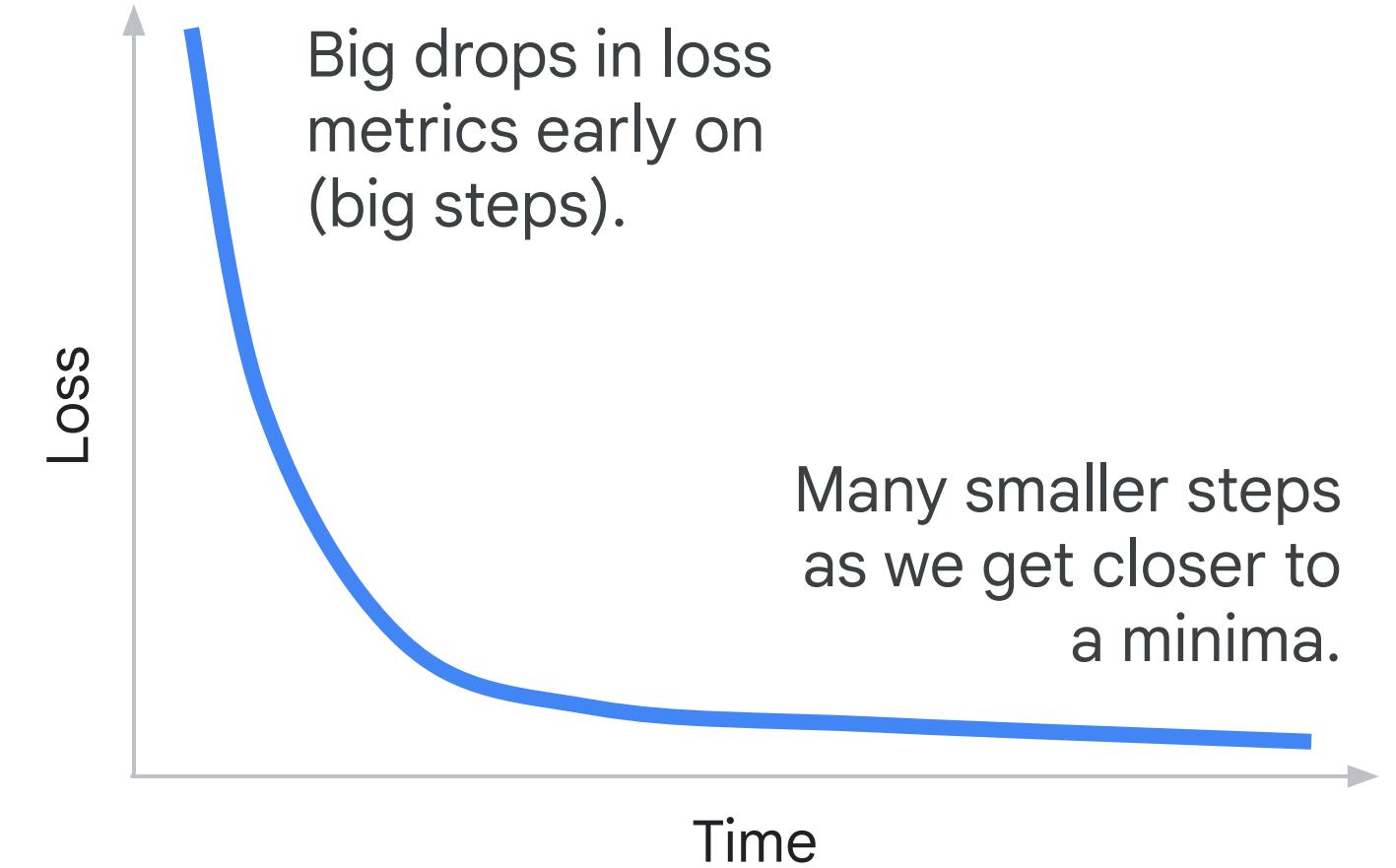
Are you done yet?

```
while loss is > Epsilon:  
    derivative = computeDerivative()  
    for i in range(self.params):  
        self.params[i] = //  
        self.params[i] //  
        - derivative[i]  
        loss = computeLoss()
```

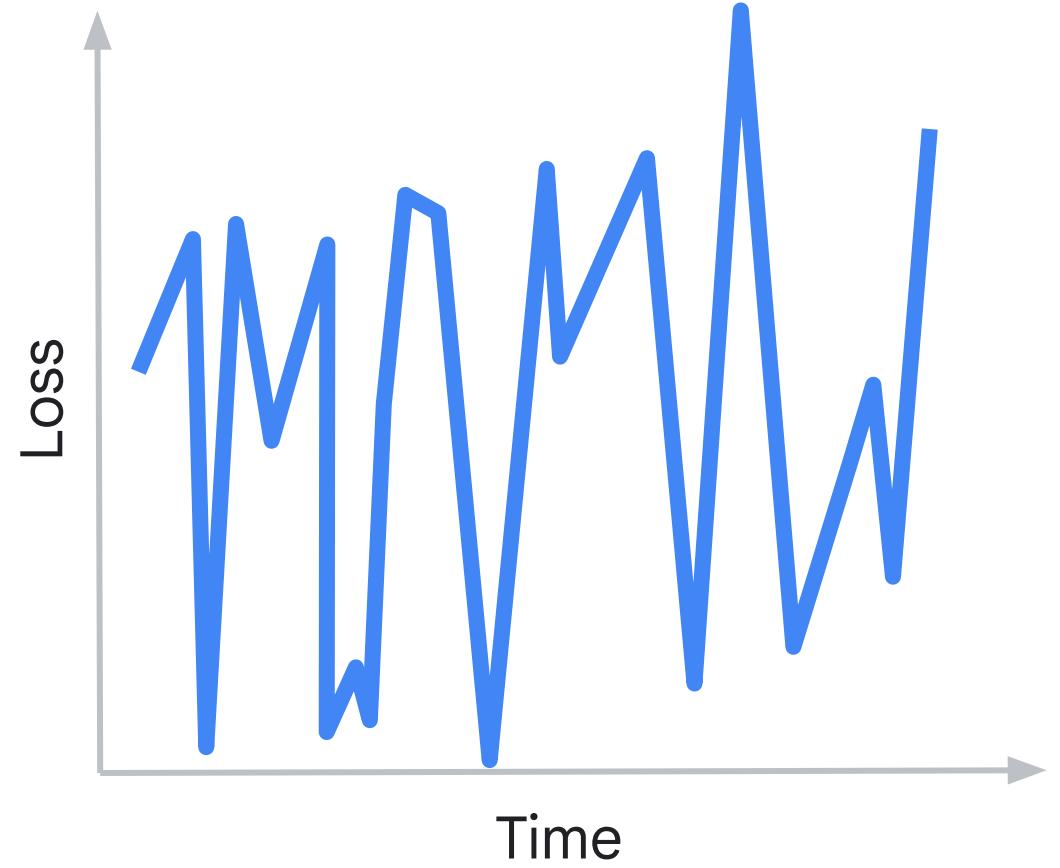




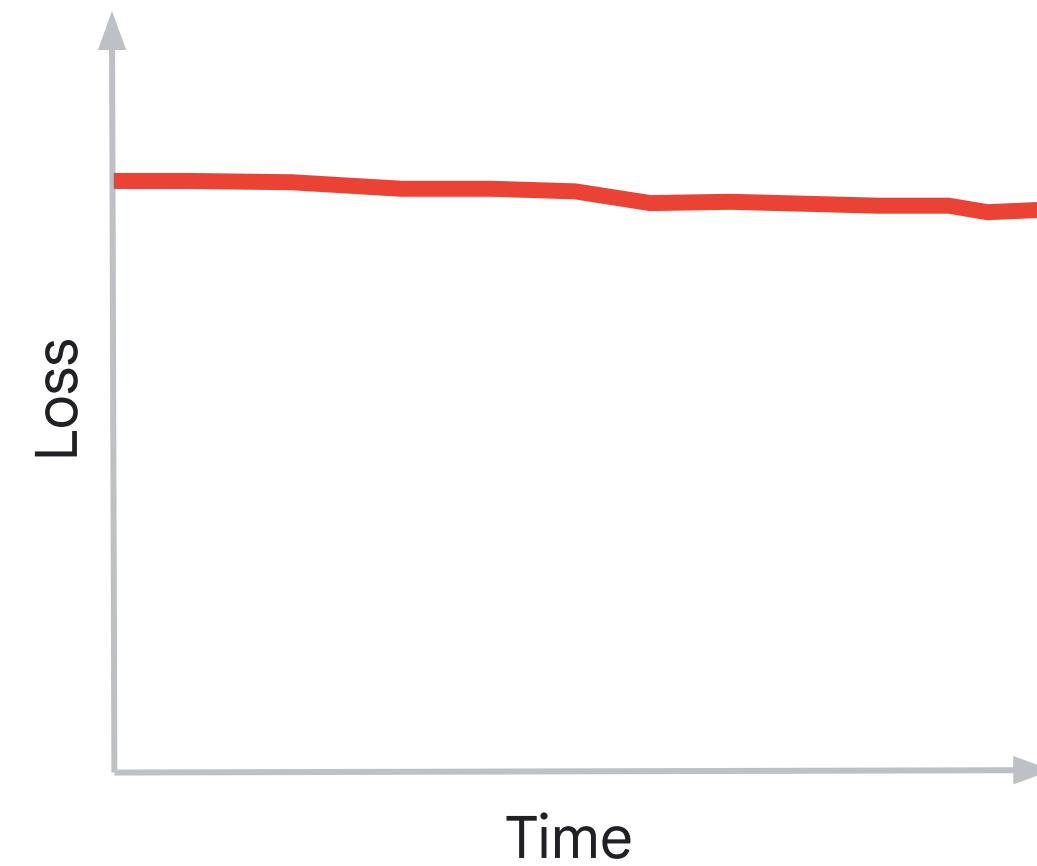
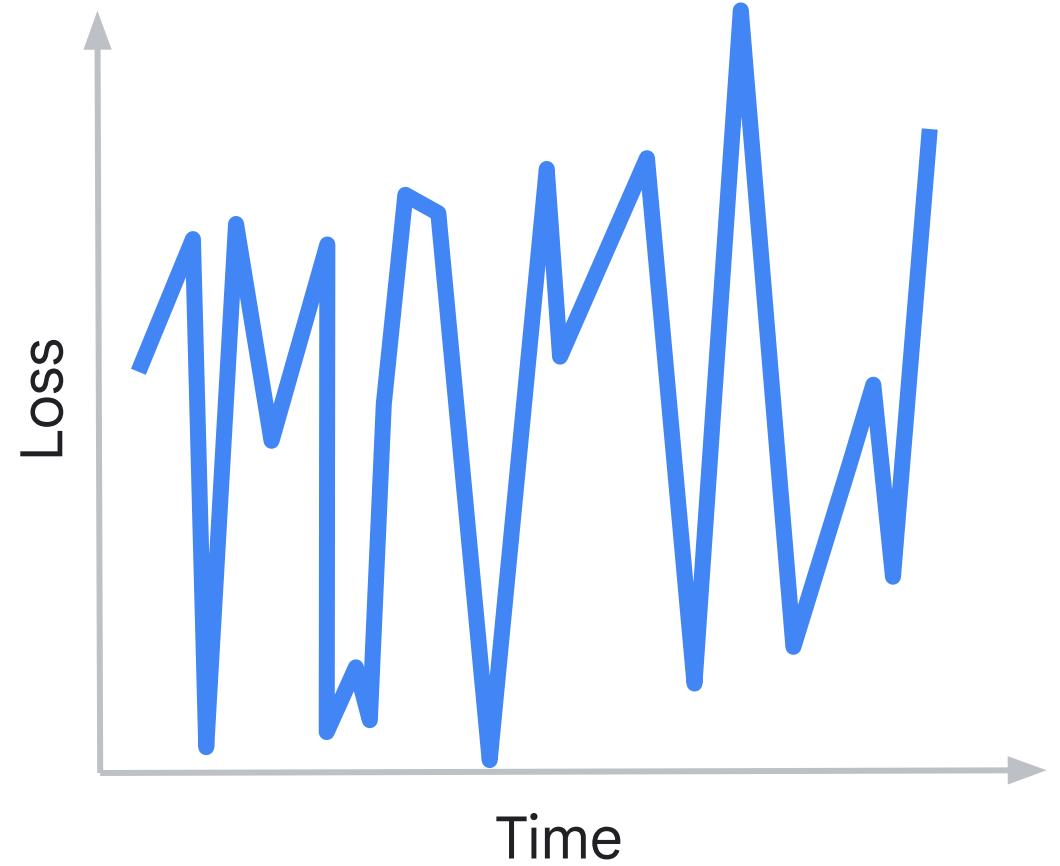
A typical loss curve



Troubleshooting a loss curve



Troubleshooting a loss curve

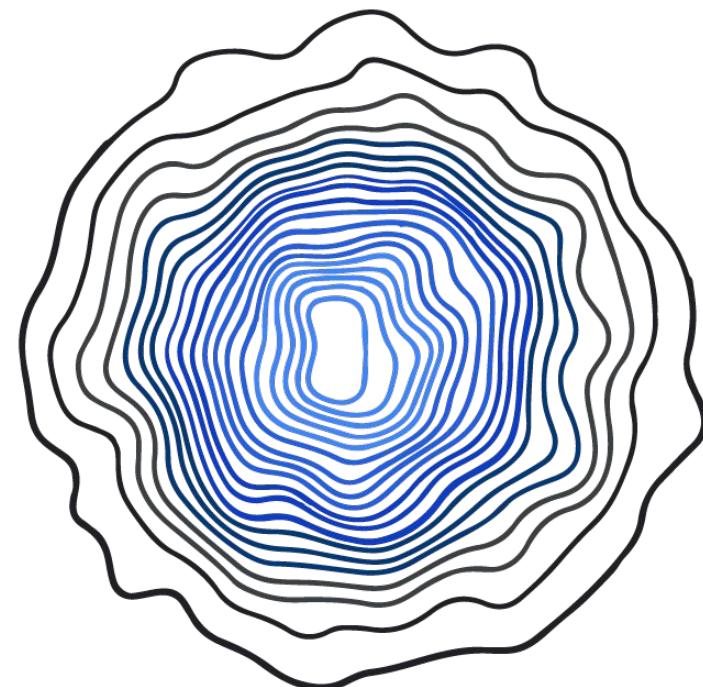


Adding a scaling hyperparameter

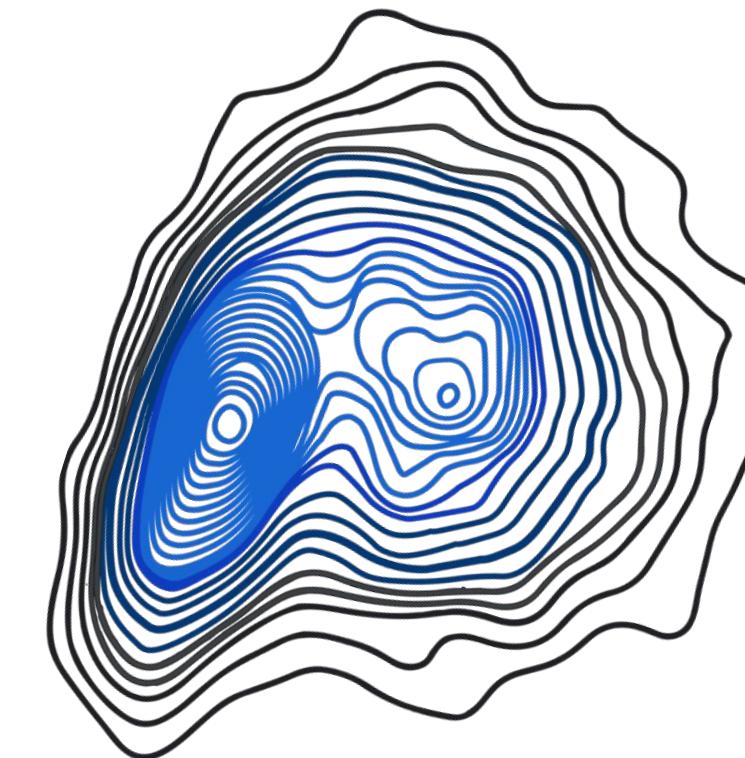
```
while loss is > Epsilon:  
    derivative = computeDerivative()  
    for i in range(self.params):  
        self.params[i] = //  
            self.params[i] //  
                - learning_rate //  
                * derivative[i]  
    loss = computeLoss()
```

Problem: My model changes every time I retrain it

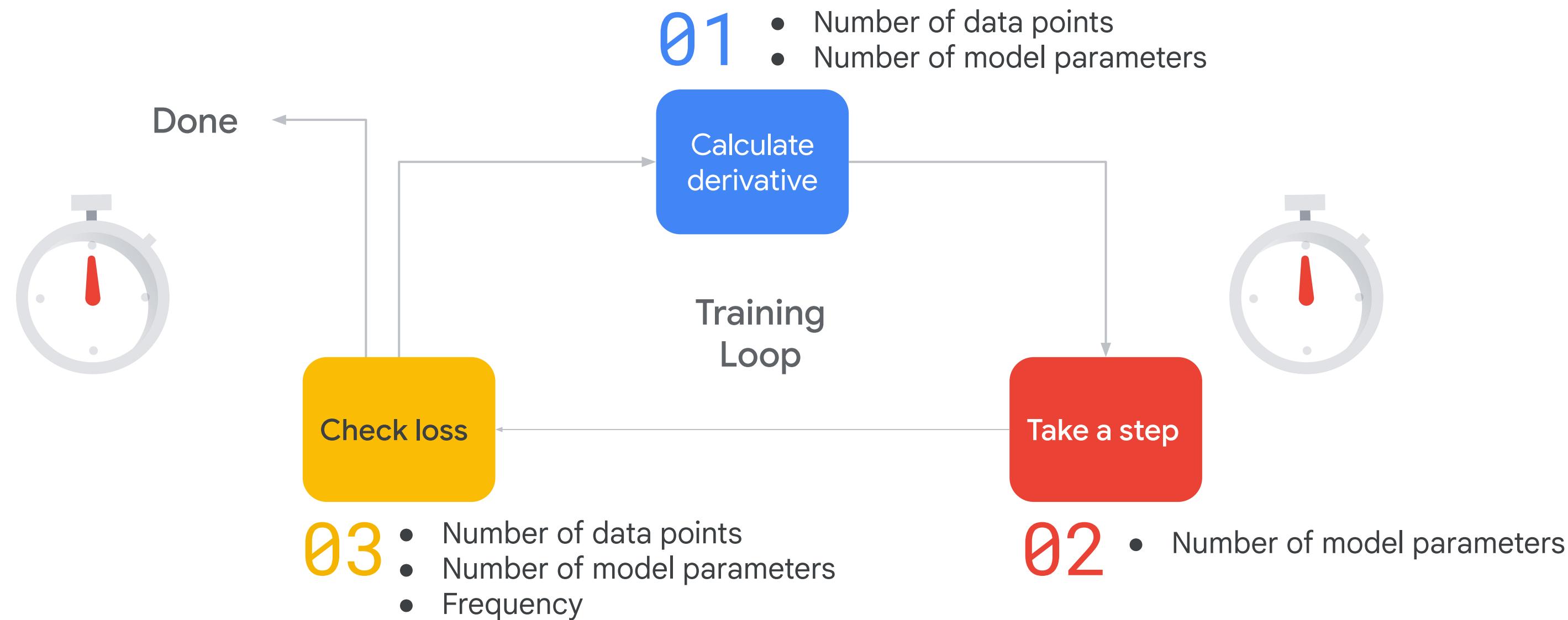
Loss surface with a
global minimum



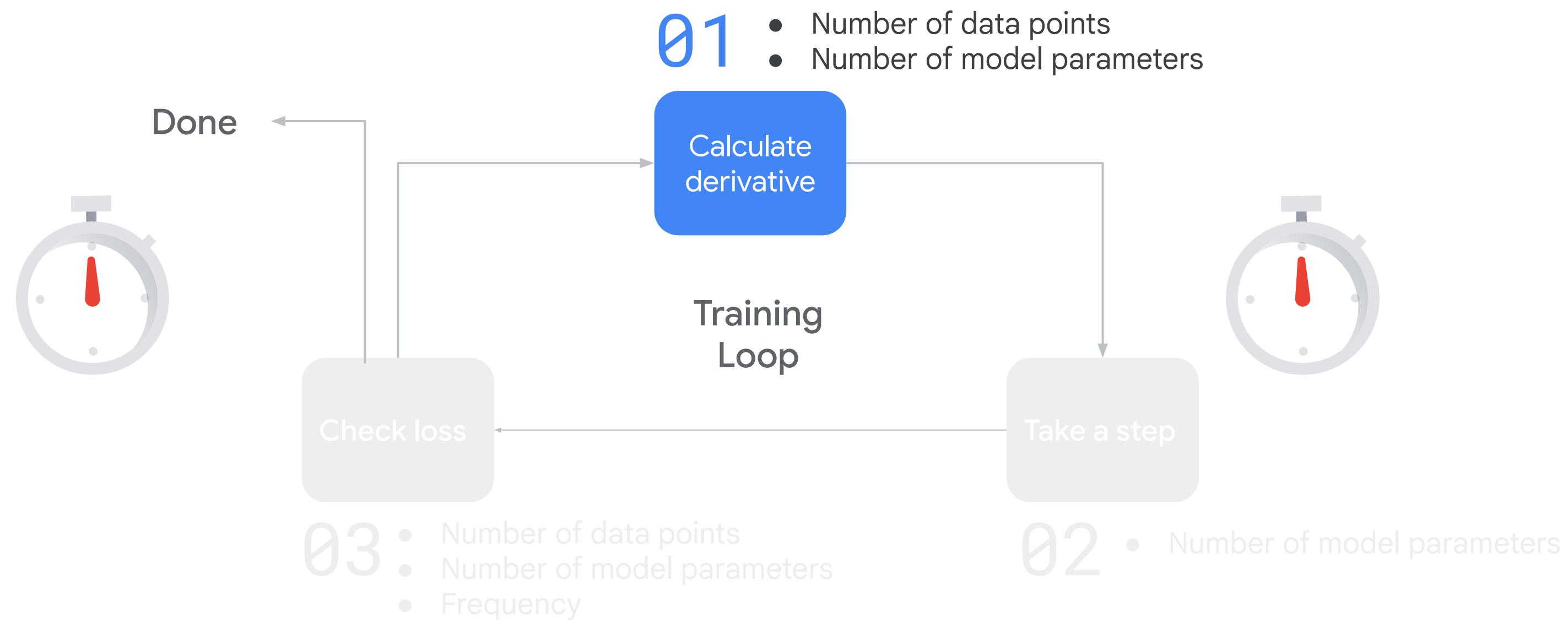
Loss surface with more
than one minima



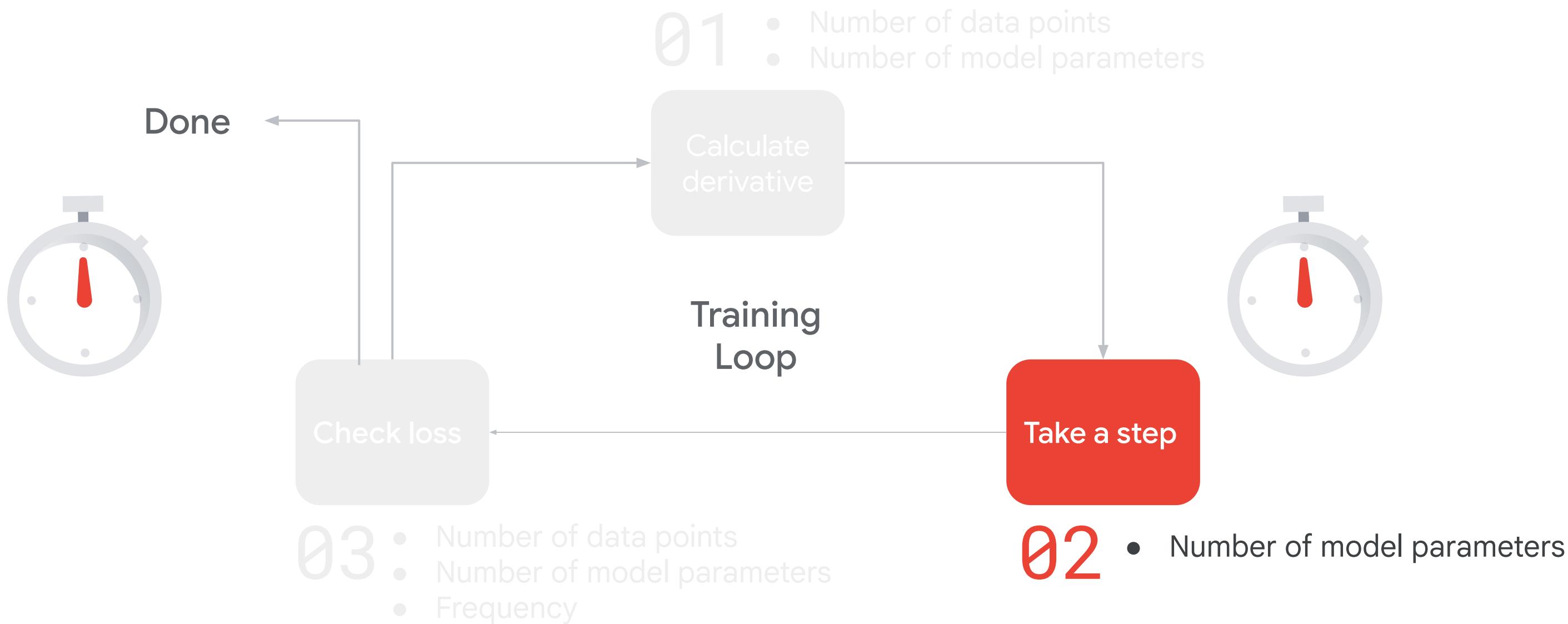
Problem: Model training is still too slow



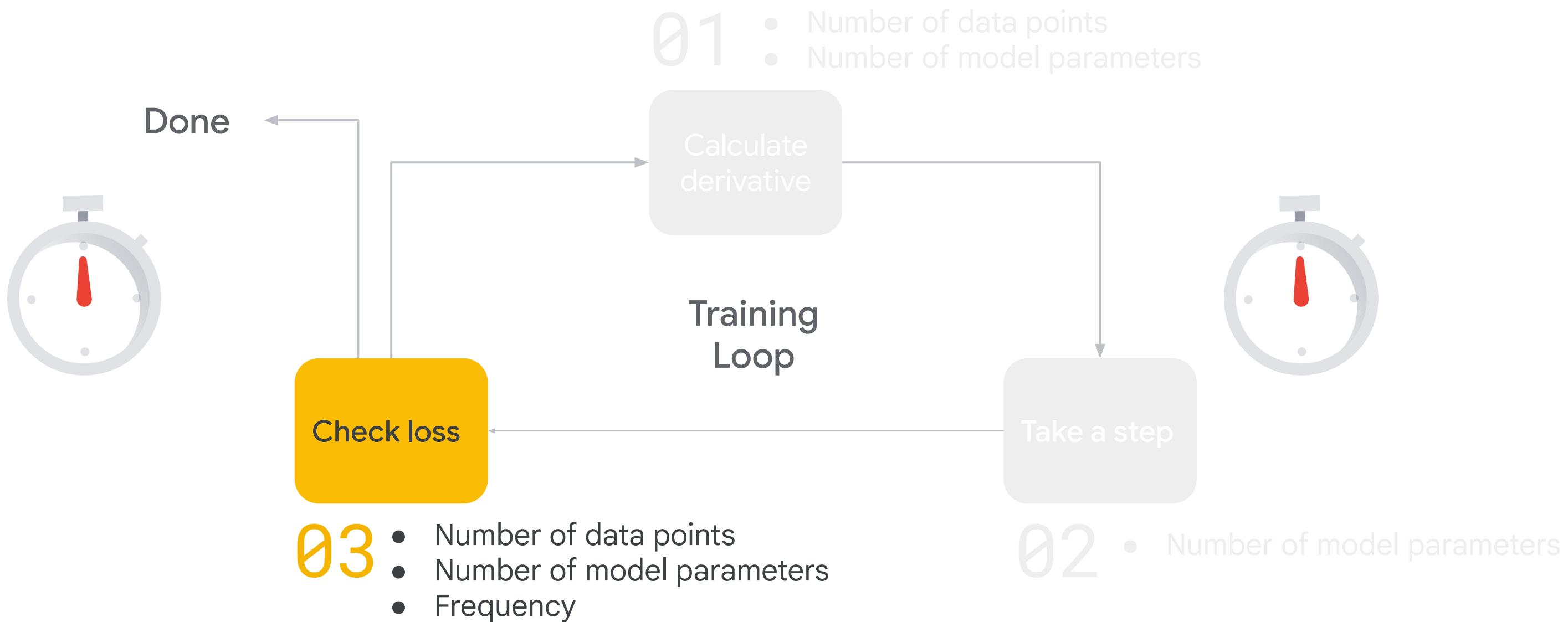
Problem: Model training is still too slow



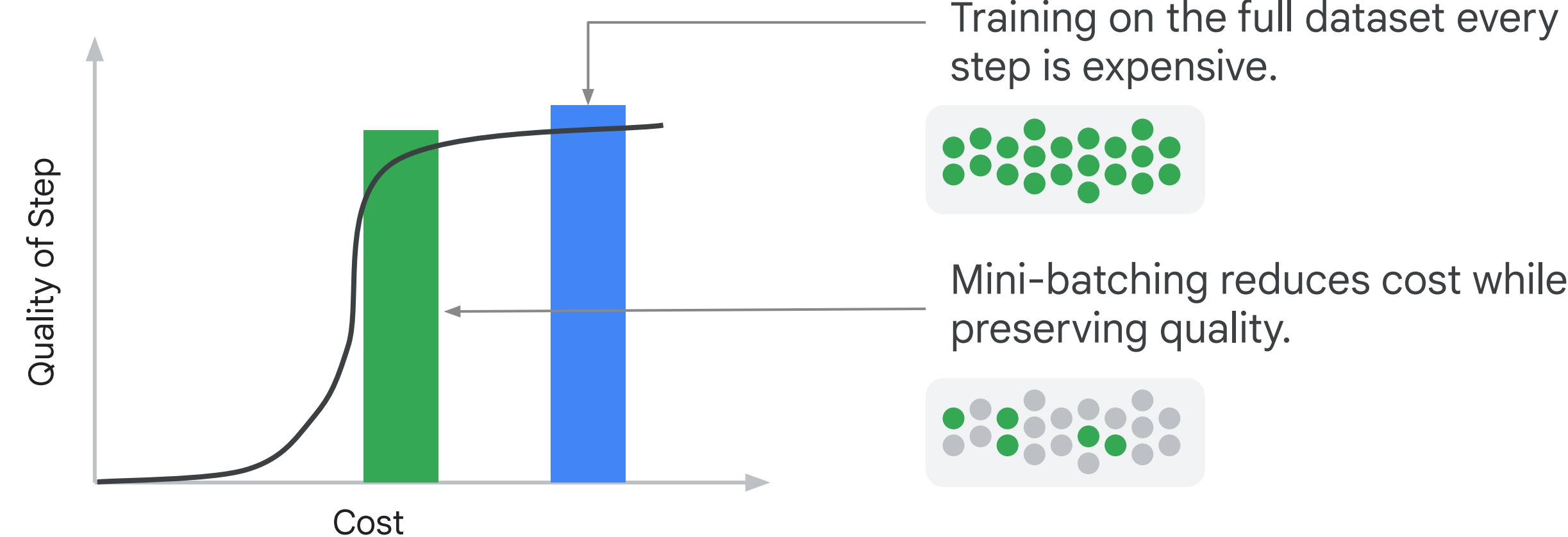
Problem: Model training is still too slow



Problem: Model training is still too slow



Calculating the derivative on fewer data points



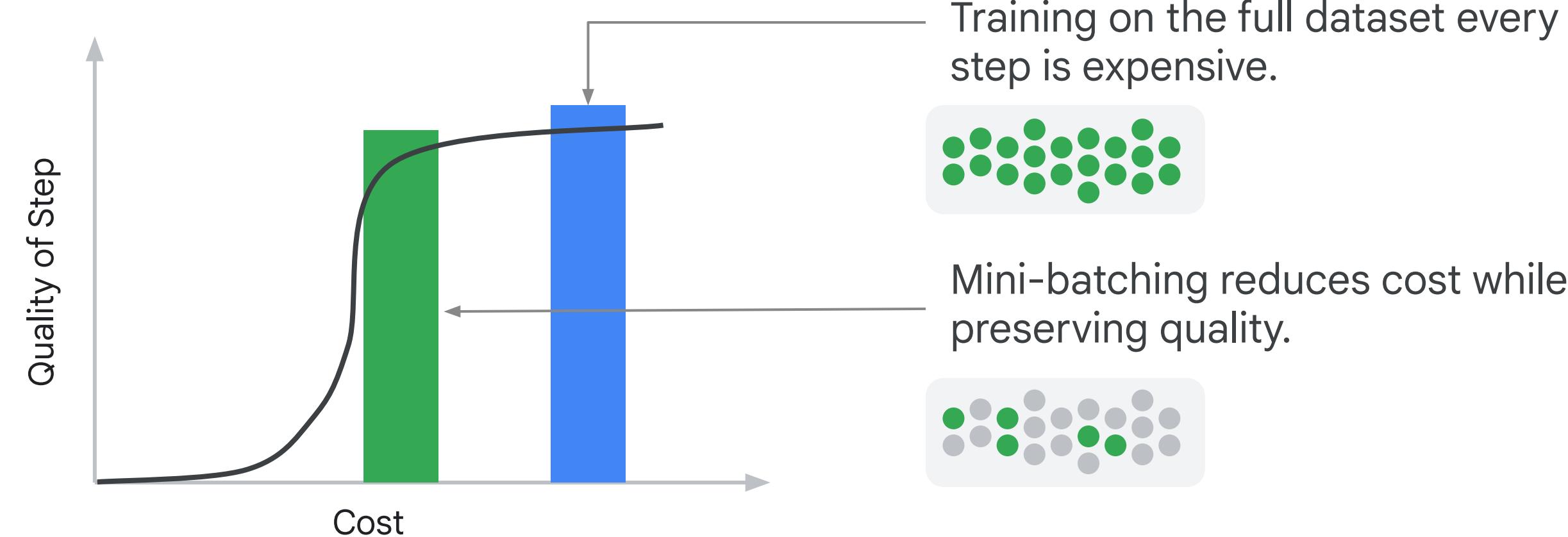
Typical values for batch size: 10 - 1000 examples.

“**Batch** gradient descent”

≠ mini-batch gradient descent

Mini-batch size = batch-size

Calculating the derivative on fewer data points



Typical values for batch size: 10 - 1000 examples.

Checking loss with reduced frequency

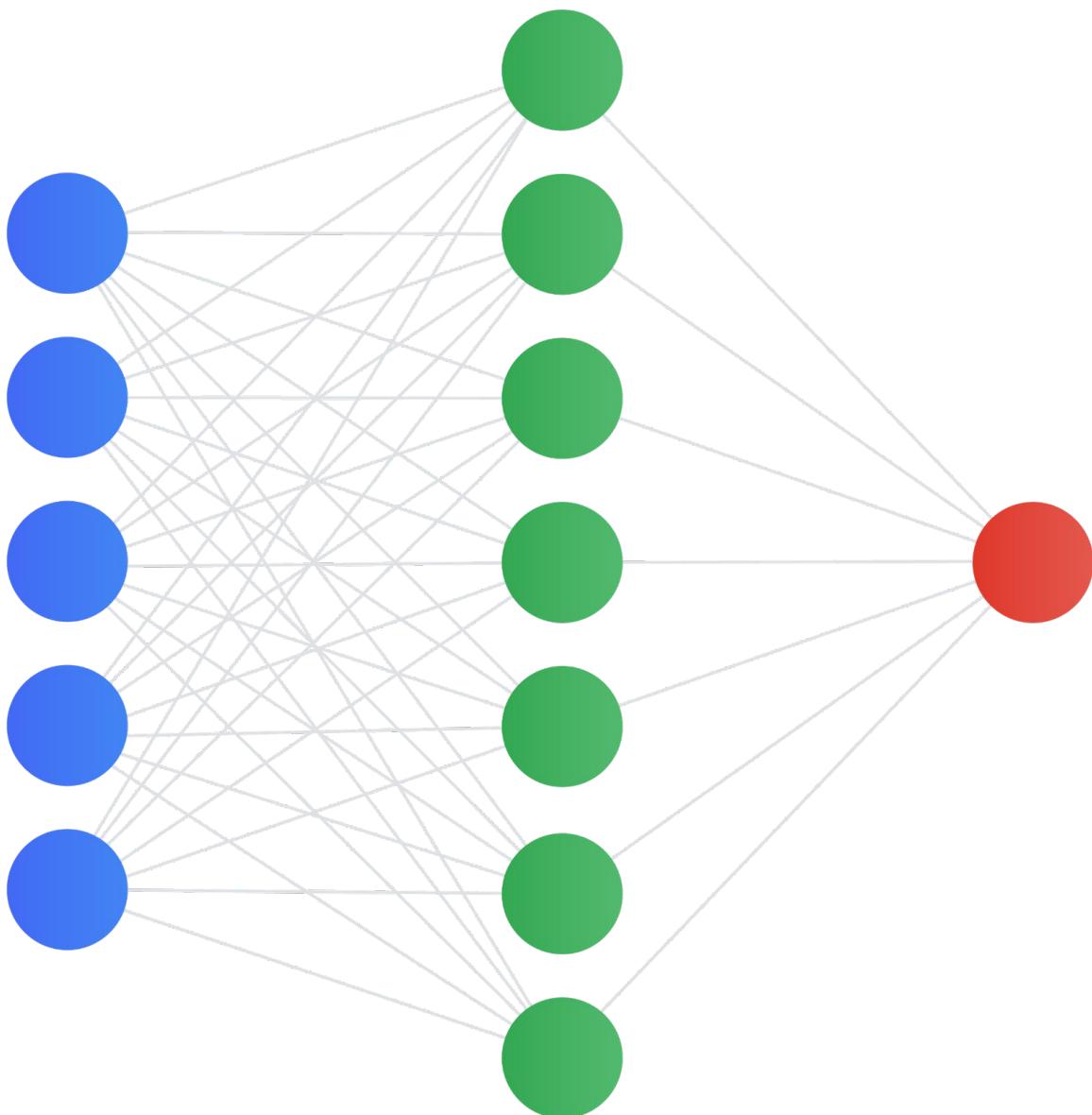
```
while loss is > Epsilon:  
    derivative = computeDerivative()  
    for i in range(self.params):  
        self.params[i] = //  
        self.params[i] //  
        - learning_rate //  
        * derivative[i]  
        if readyToUpdateLoss():  
            loss = updateLoss()
```

Popular implementations for readyToUpdateLoss():

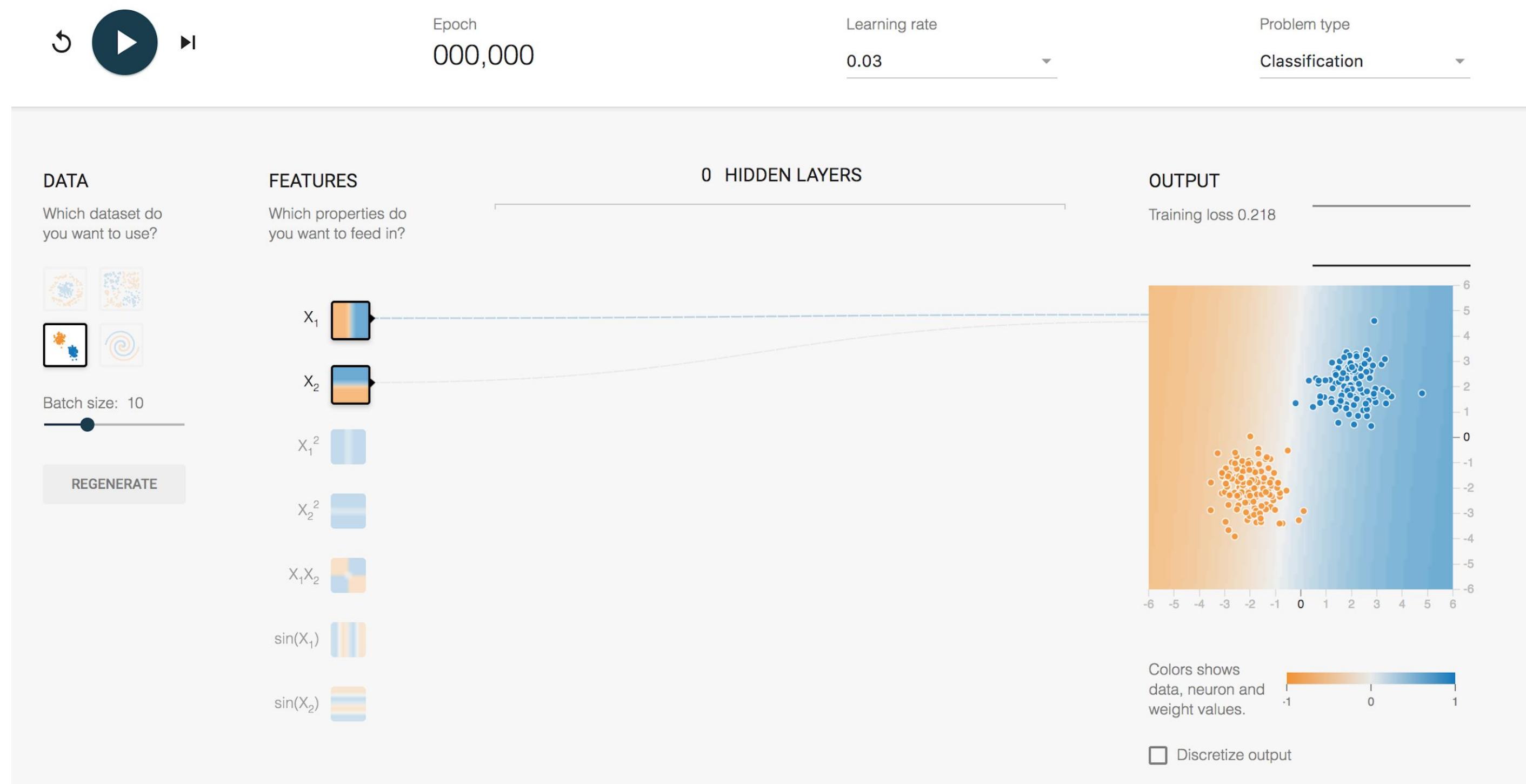
- Time-based (e.g., every hour)
- Step-based (e.g., every 1000 steps)

TensorFlow Playground

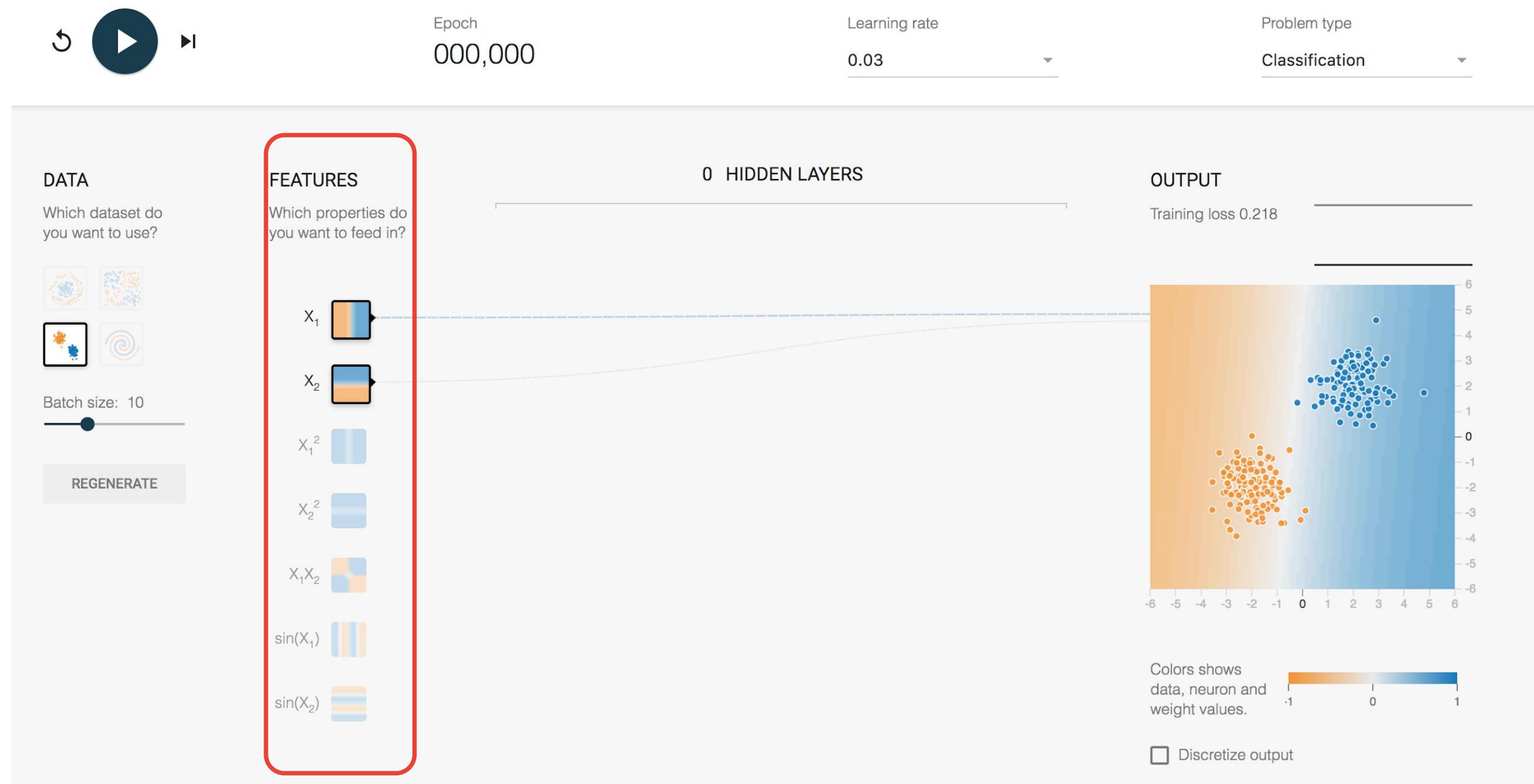
How neural networks work



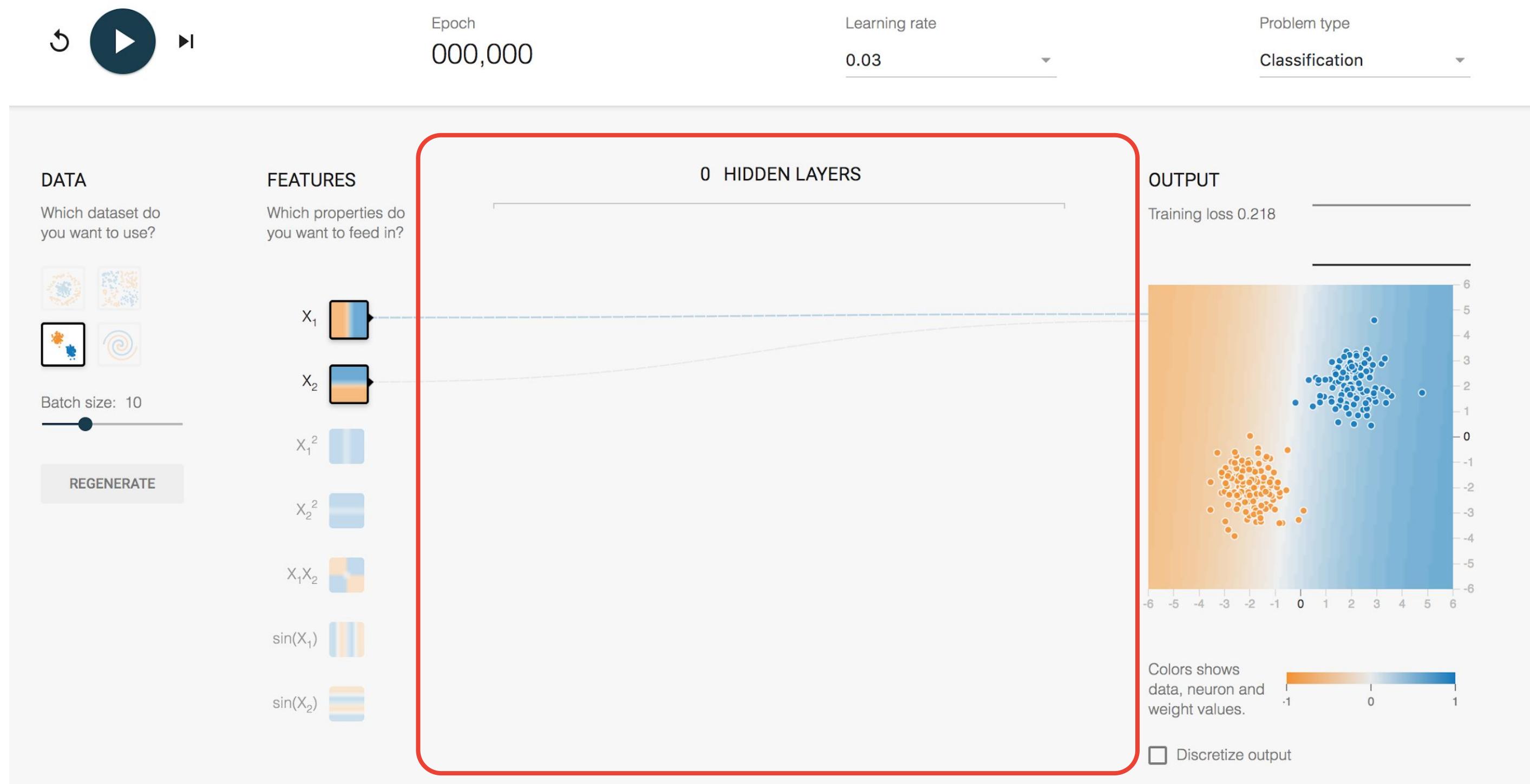
TensorFlow Playground interface



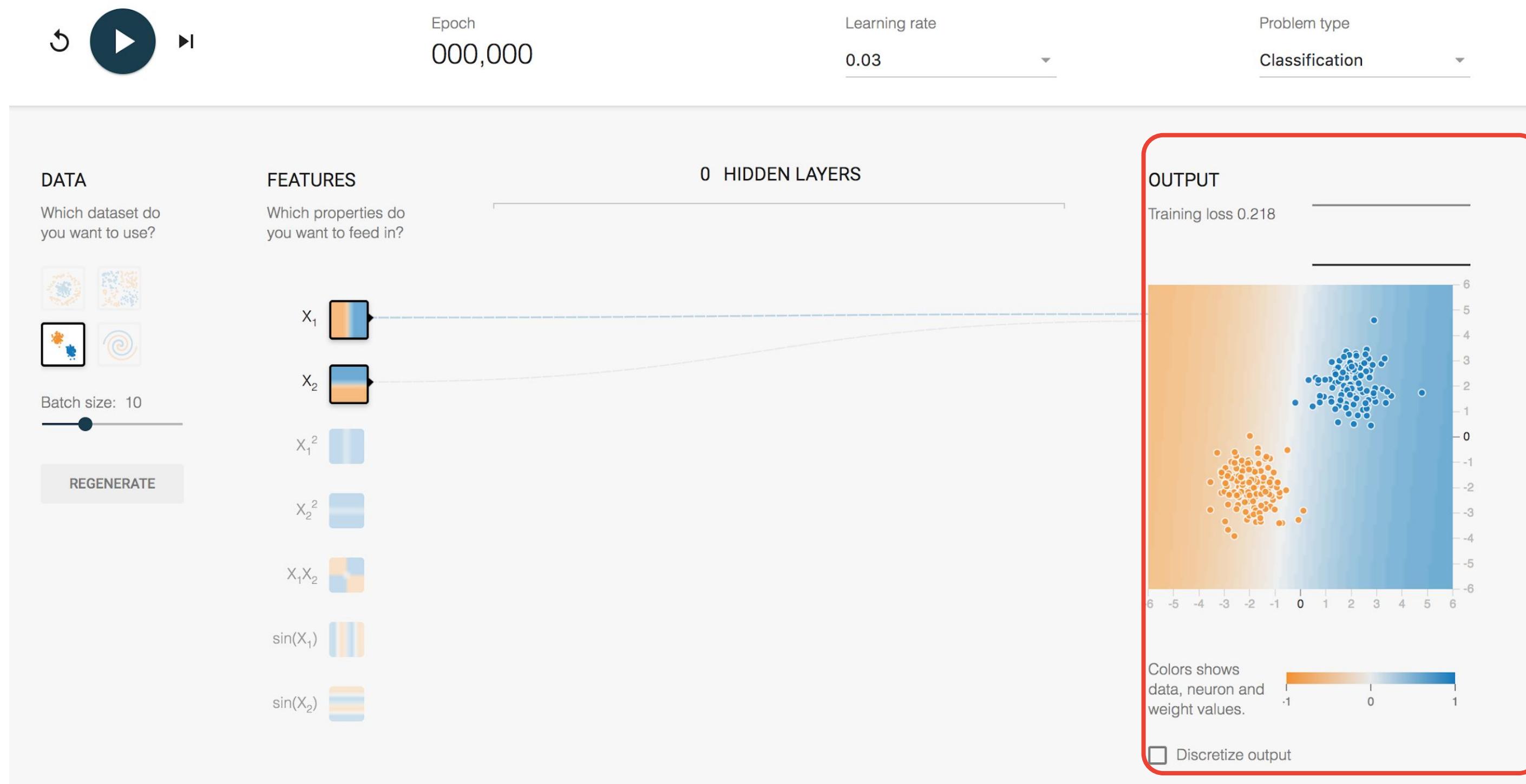
TensorFlow Playground interface



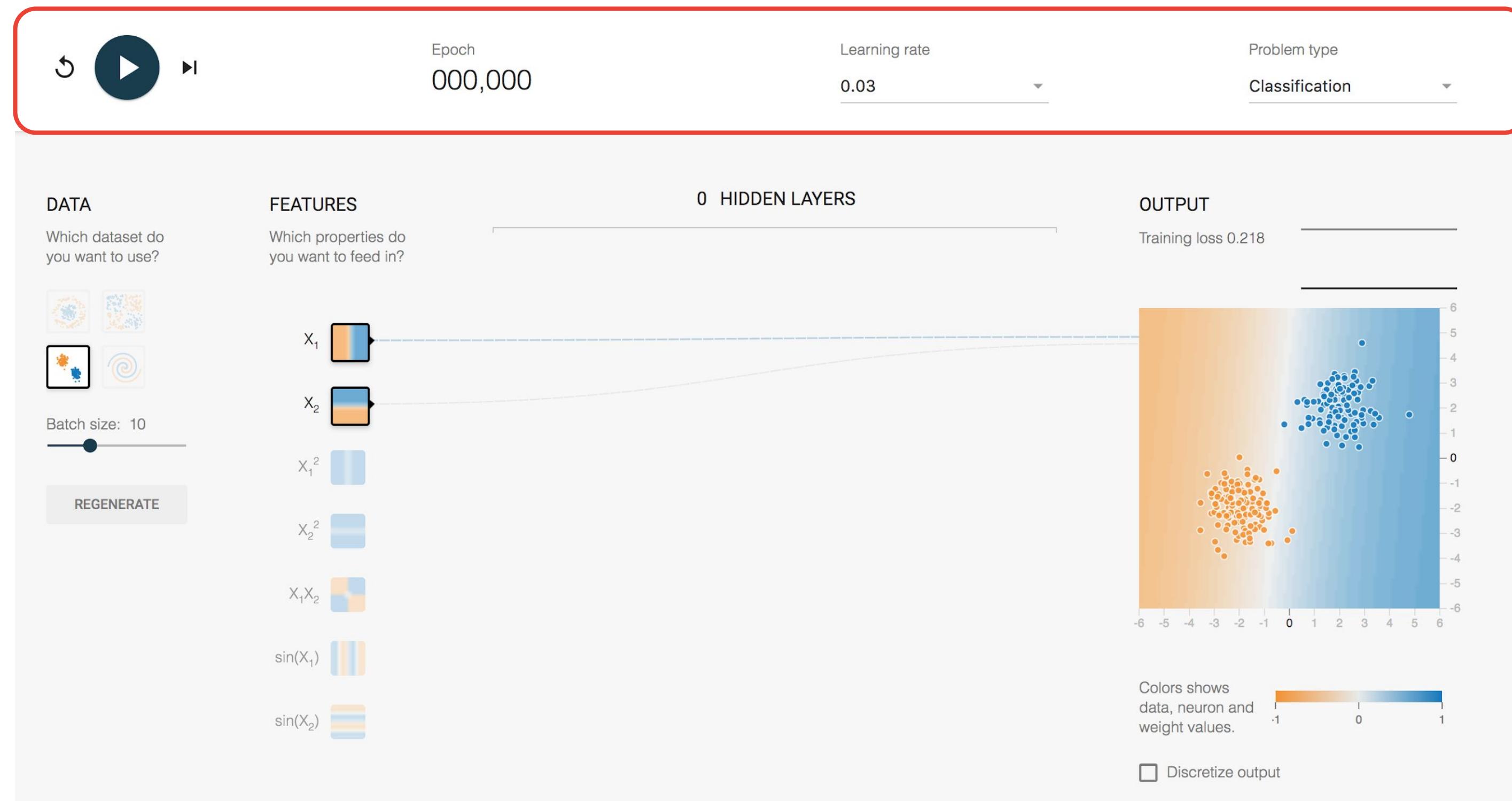
TensorFlow Playground interface



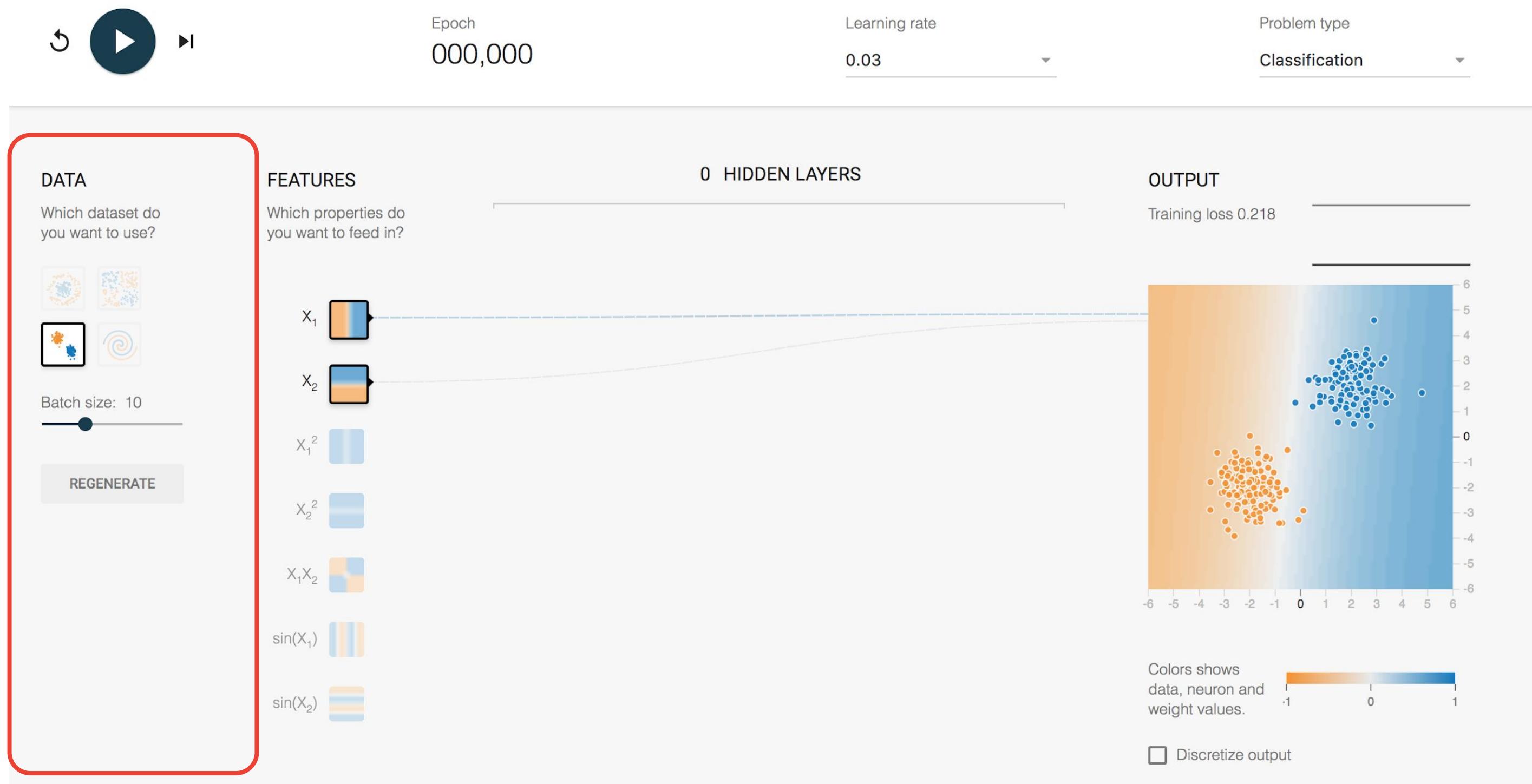
TensorFlow Playground interface



TensorFlow Playground interface



TensorFlow Playground interface

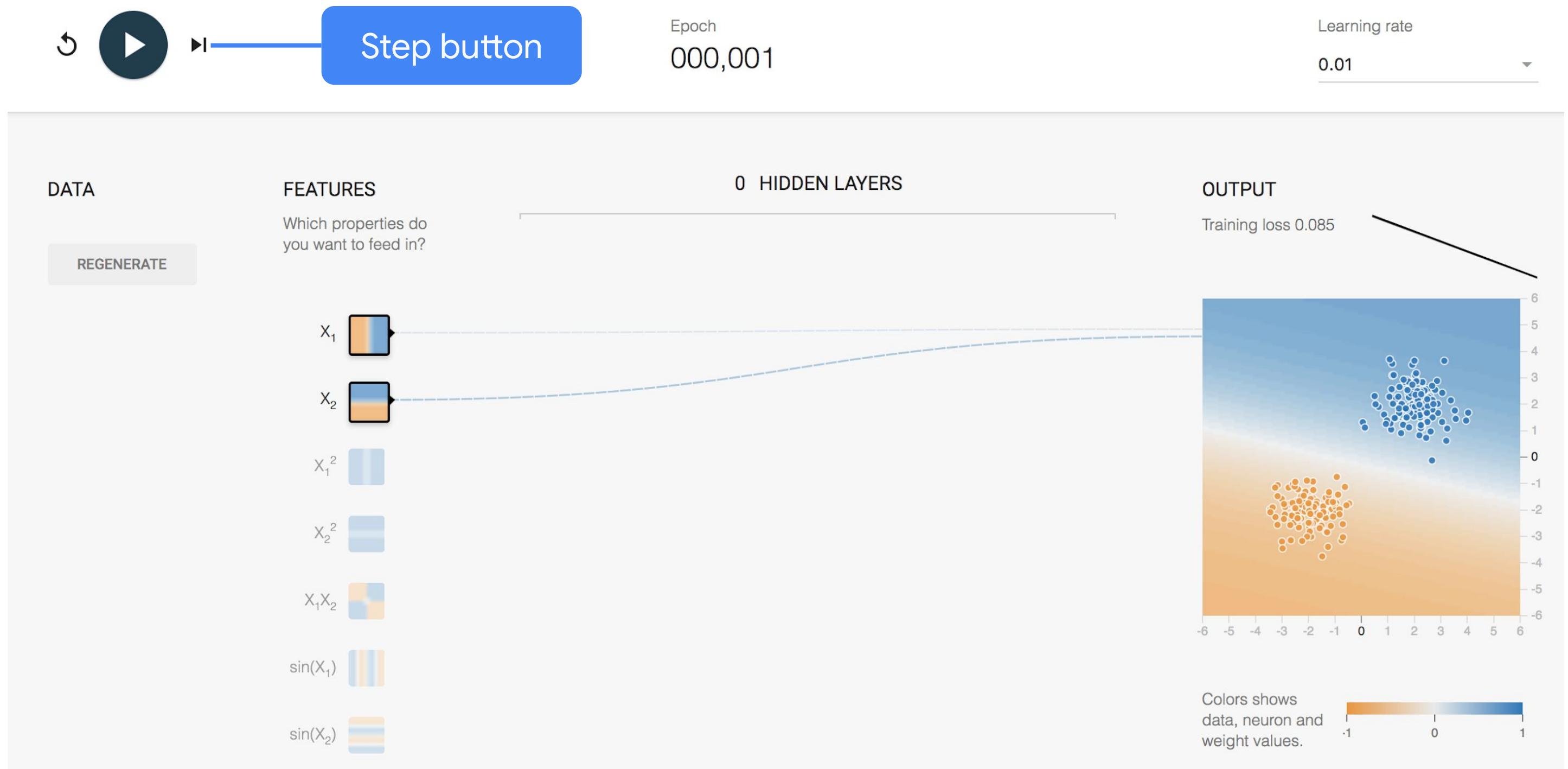


Train your first model



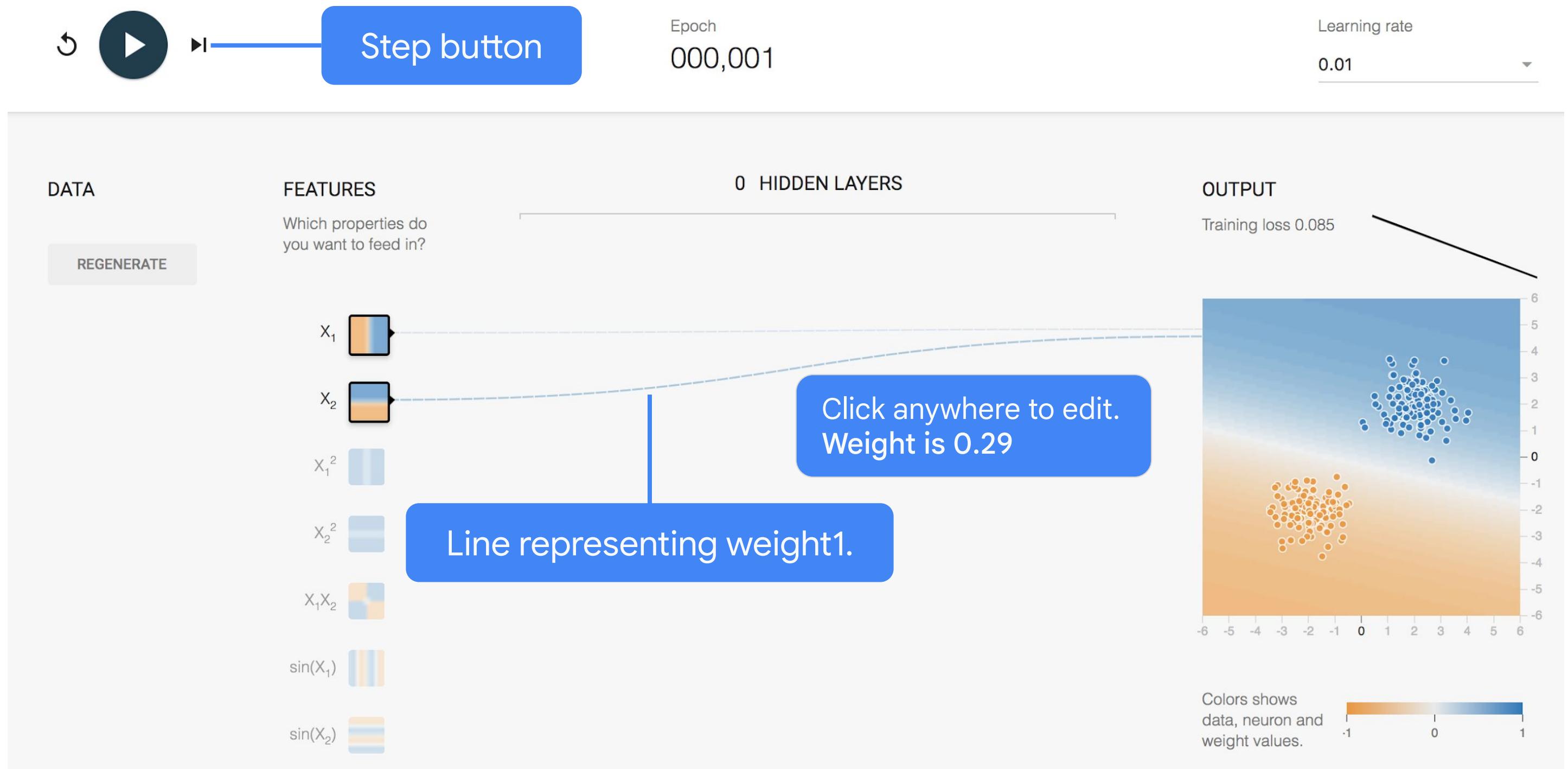
Open this link: <https://goo.gl/EEuEGp>

Train your first model



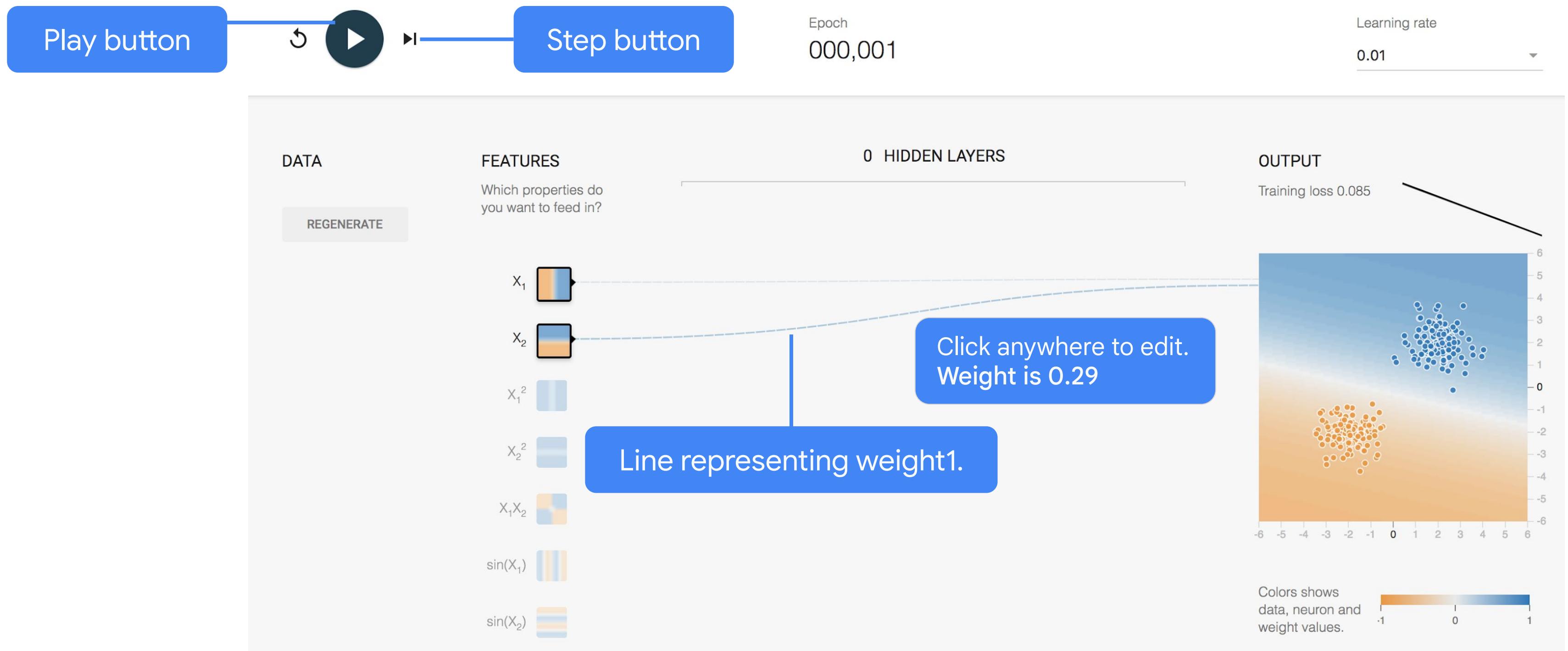
Open this link: <https://goo.gl/EEuEGp>

Train your first model



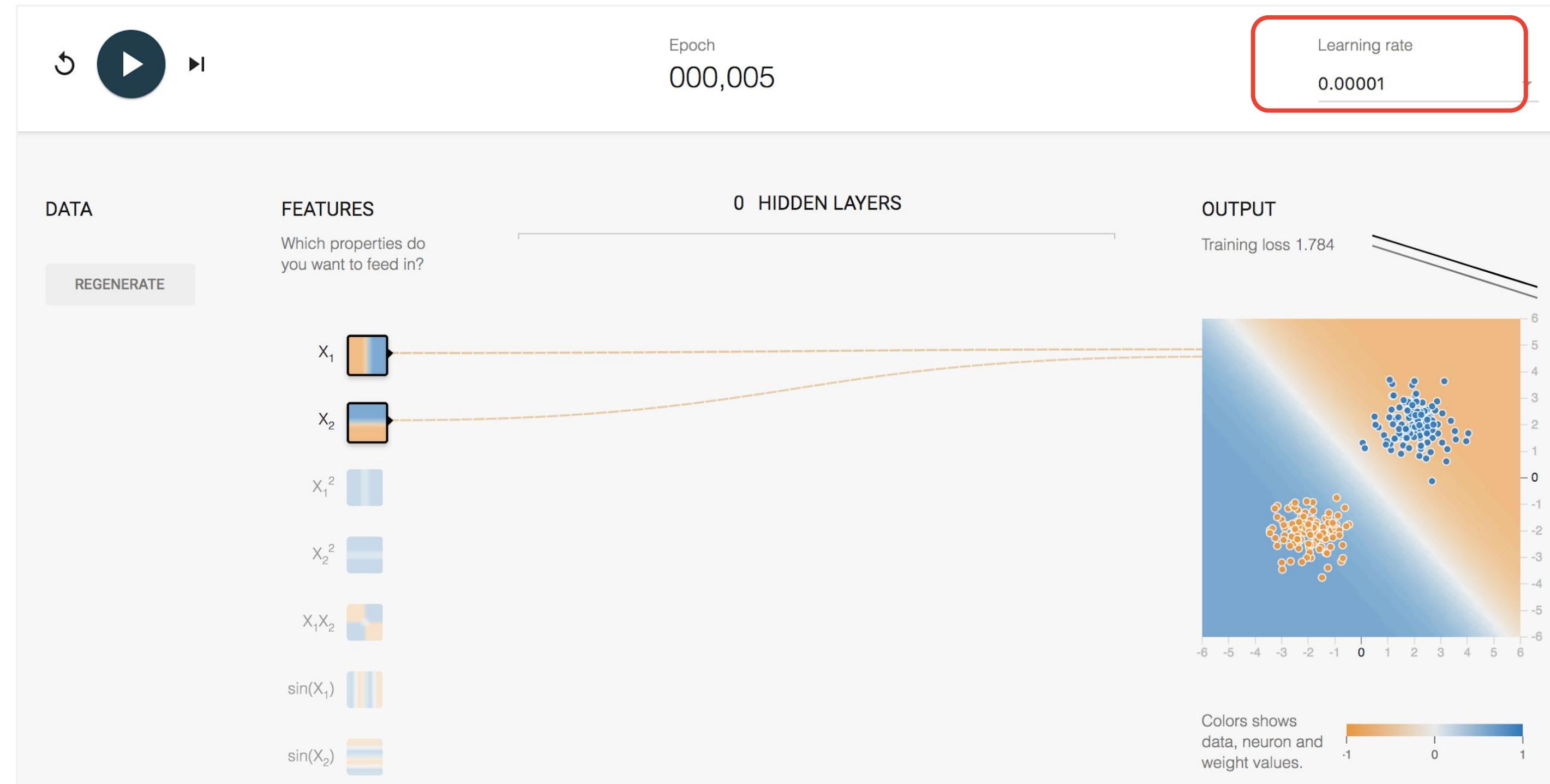
Open this link: <https://goo.gl/EEuEGp>

Train your first model



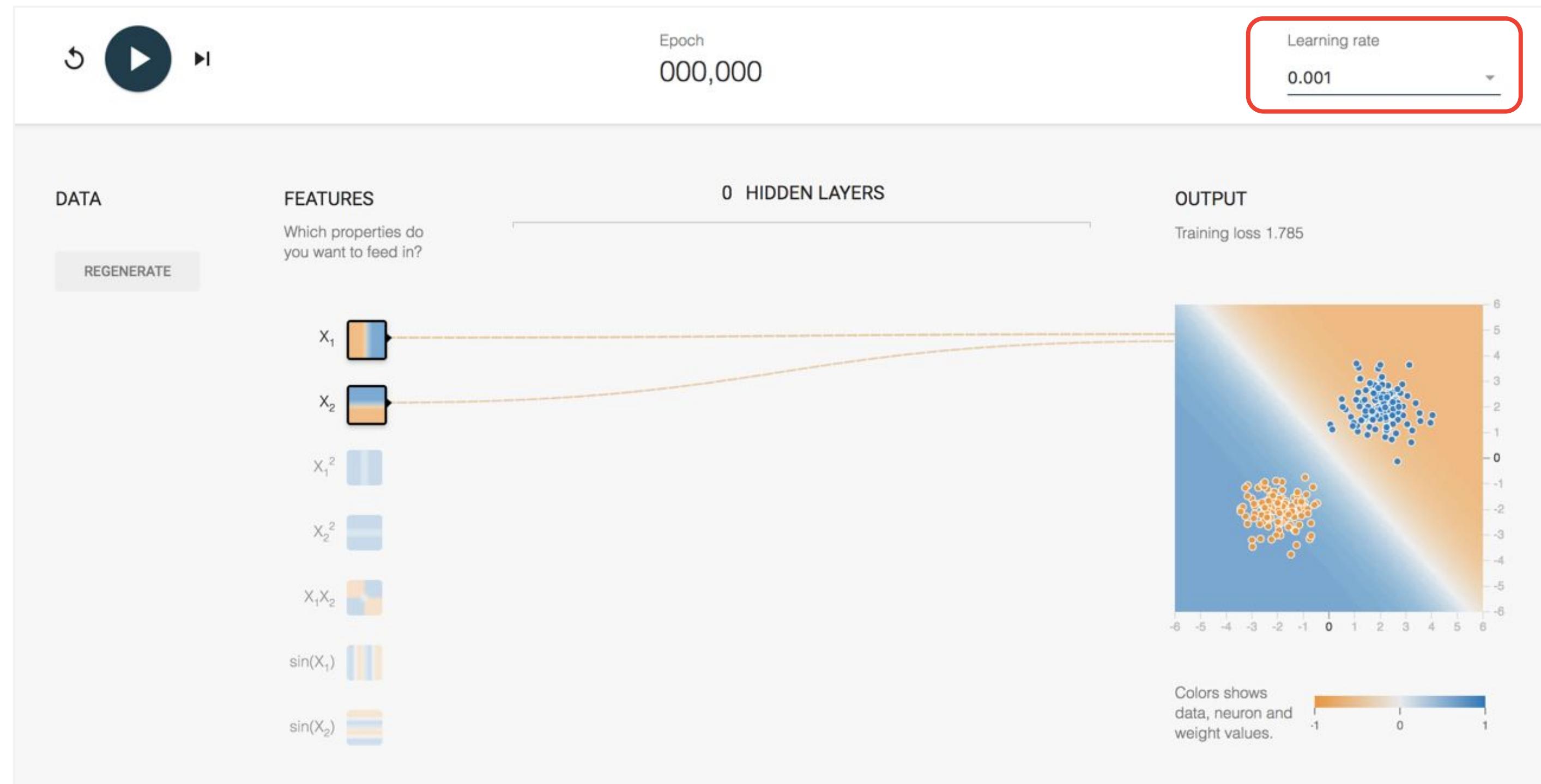
Open this link: <https://goo.gl/EEuEGp>

Experiment with learning rate: 0.00001 (tiny)



Open this link: <https://goo.gl/3pmeKj>

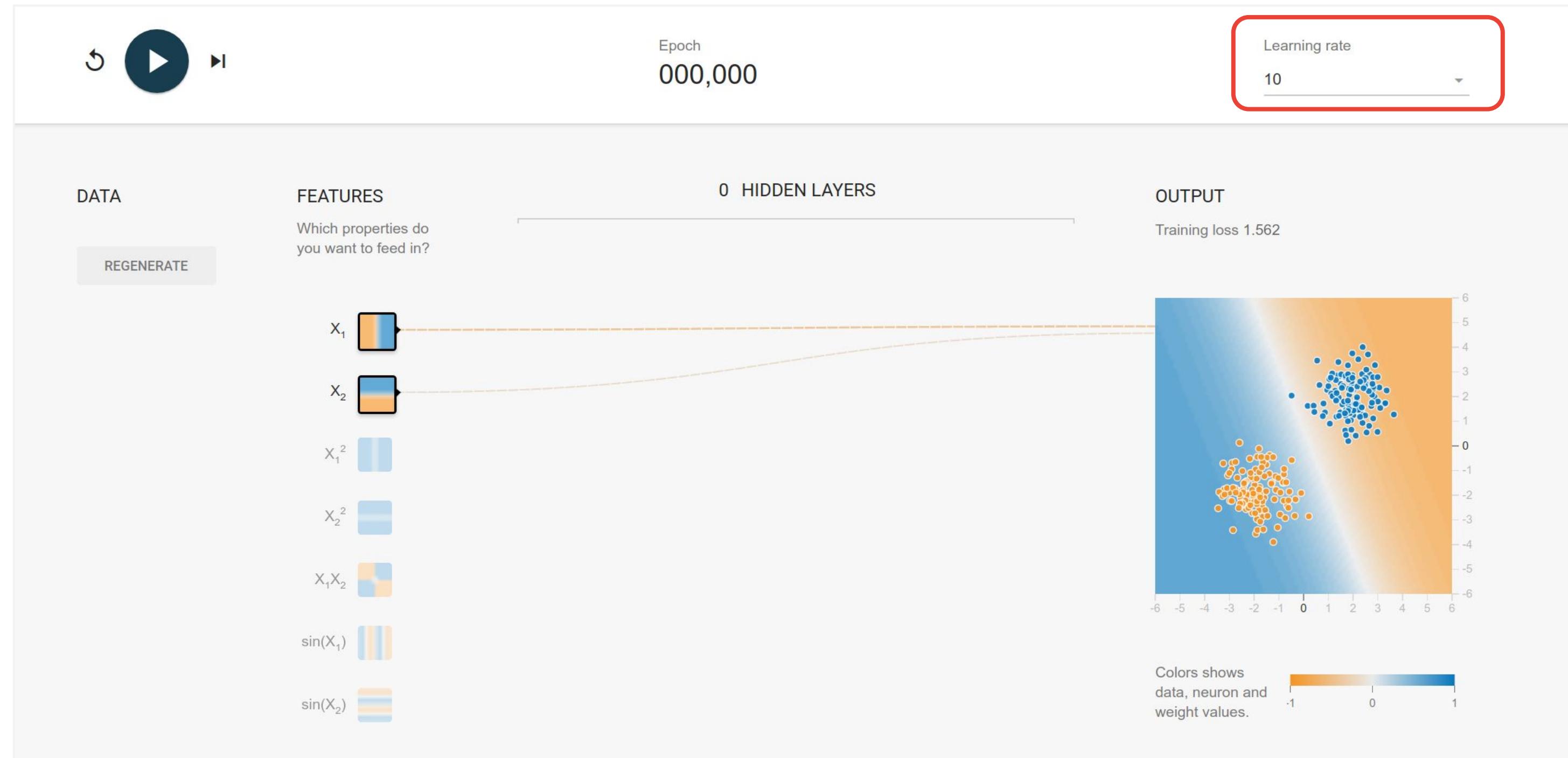
Experiment with learning rate: 0.001 (small)



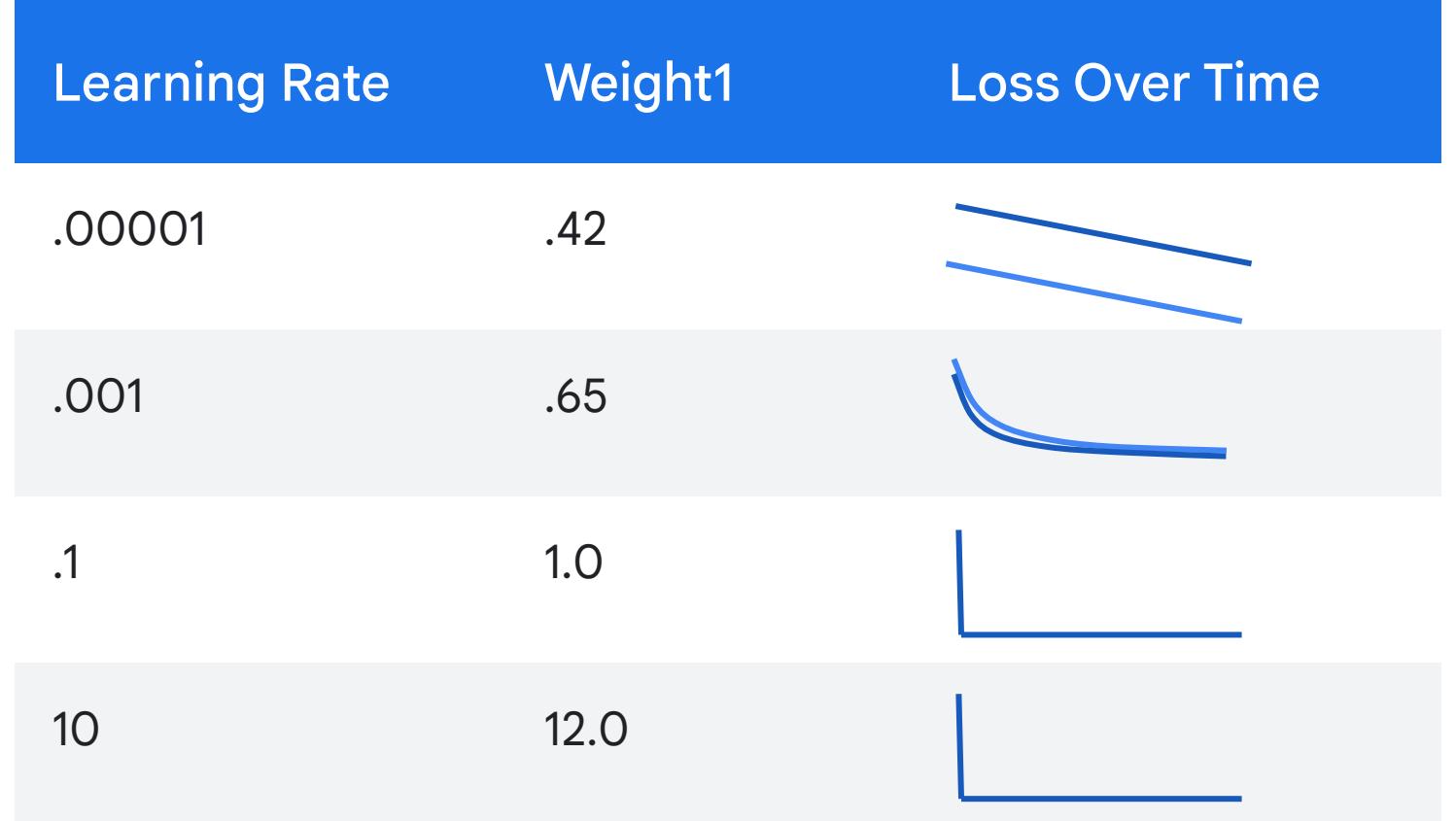
Experiment with learning rate: 0.1 (moderate)



Experiment with learning rate: 10 (huge)

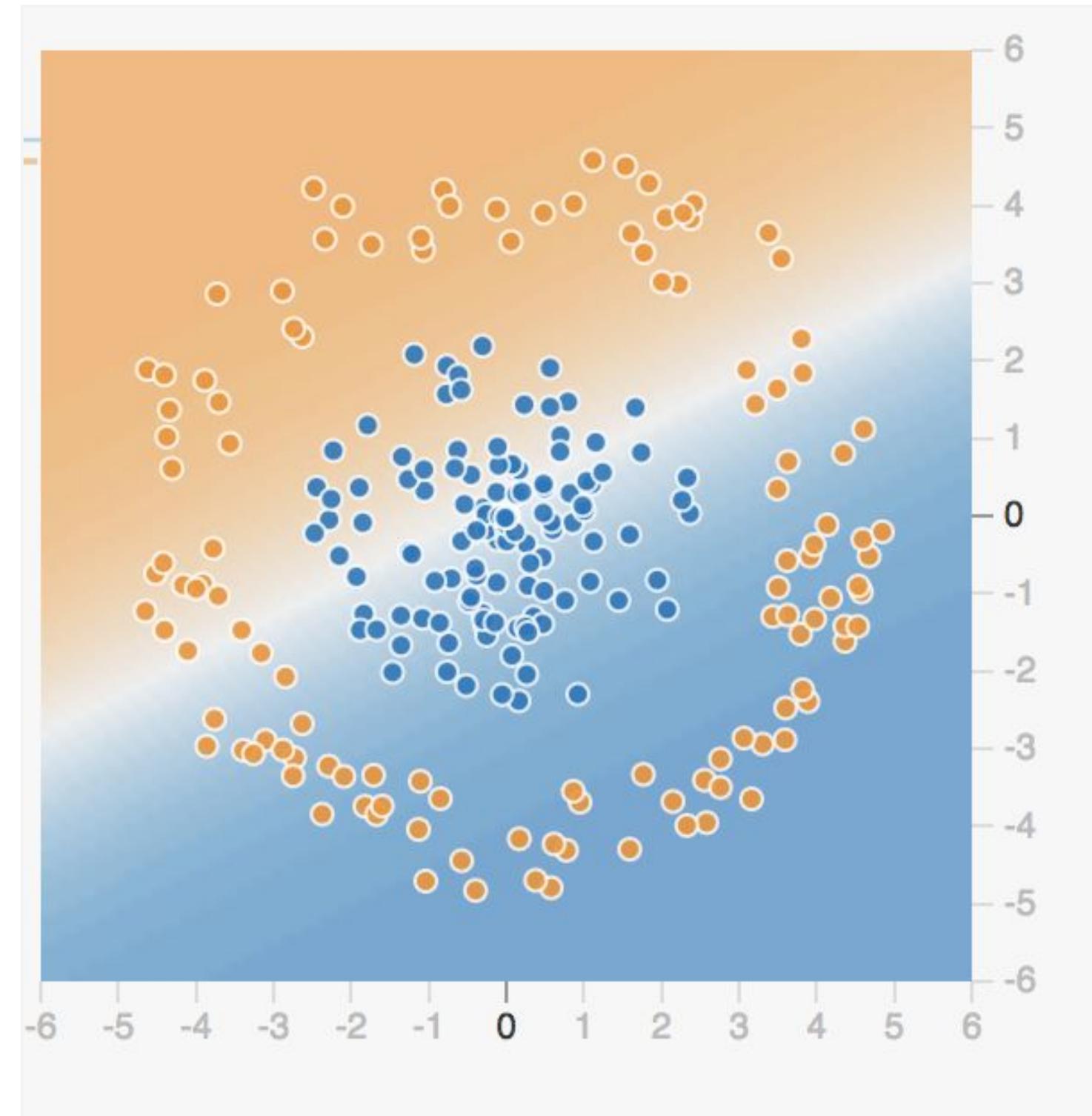


Experimenting with learning rate: **Observations**



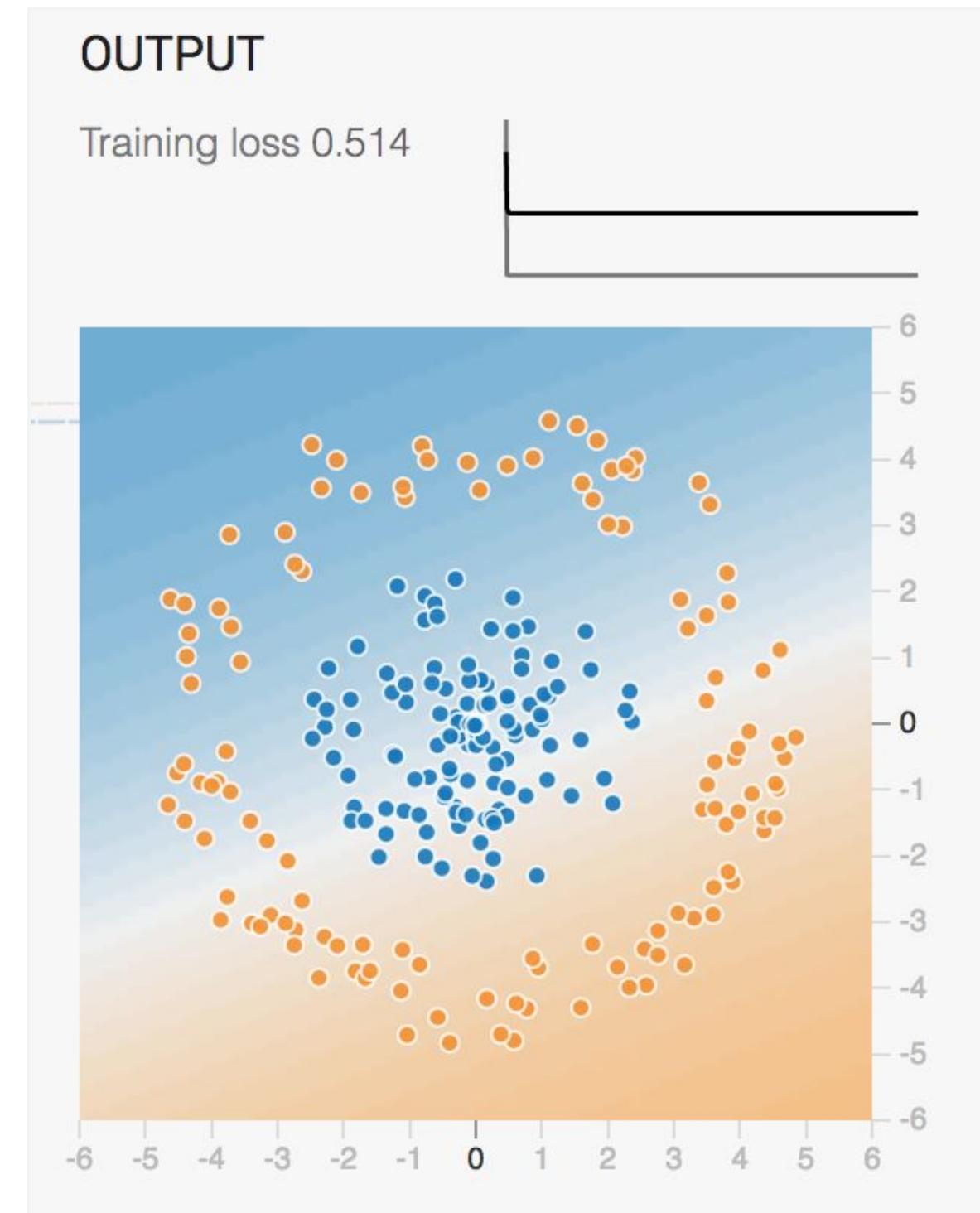
What about datasets
that aren't easy to
separate?

Open this link:
<https://goo.gl/ou9iMB>



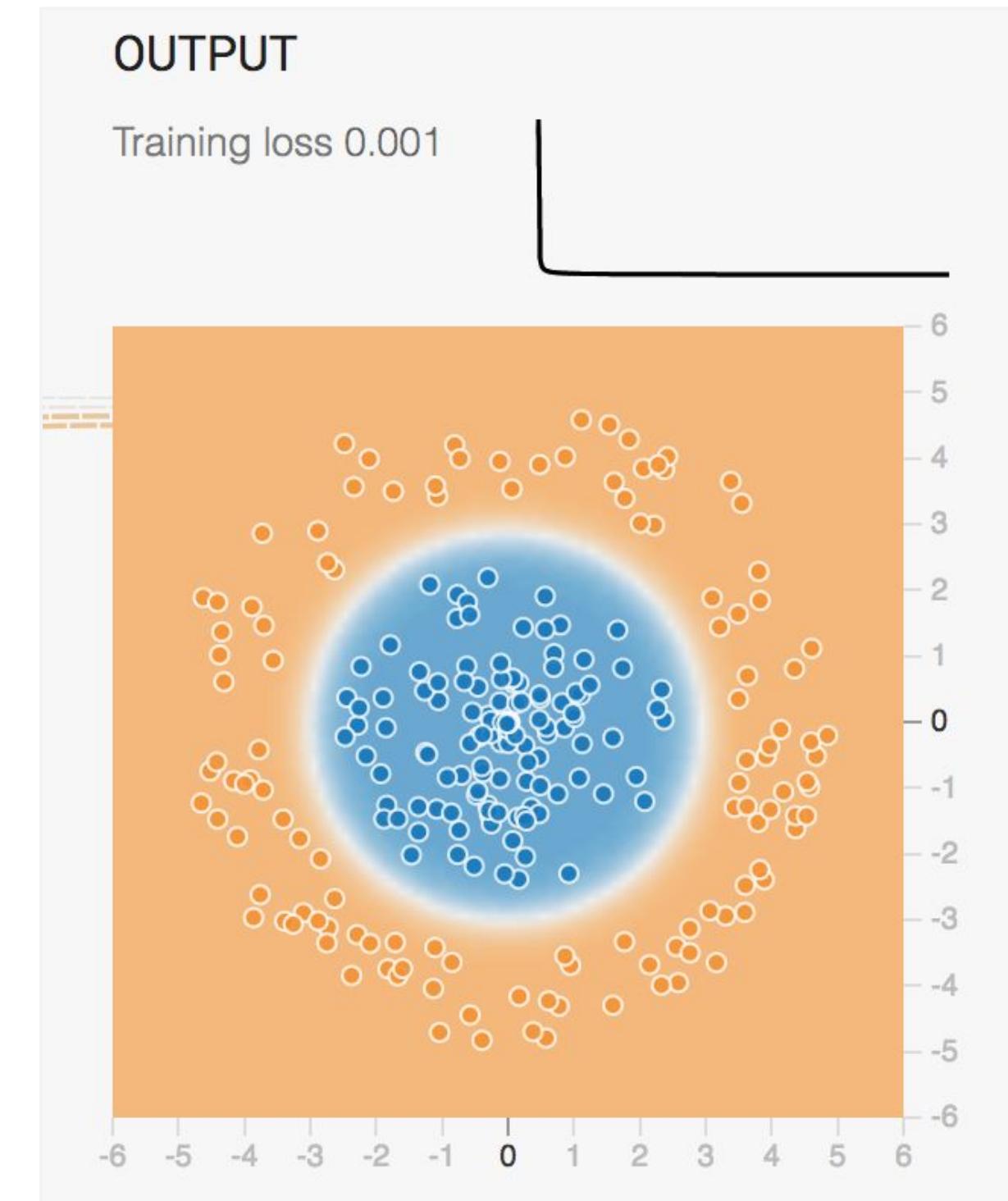
The limitations of linear models

The decision boundary does a poor job
of dividing the data by class.



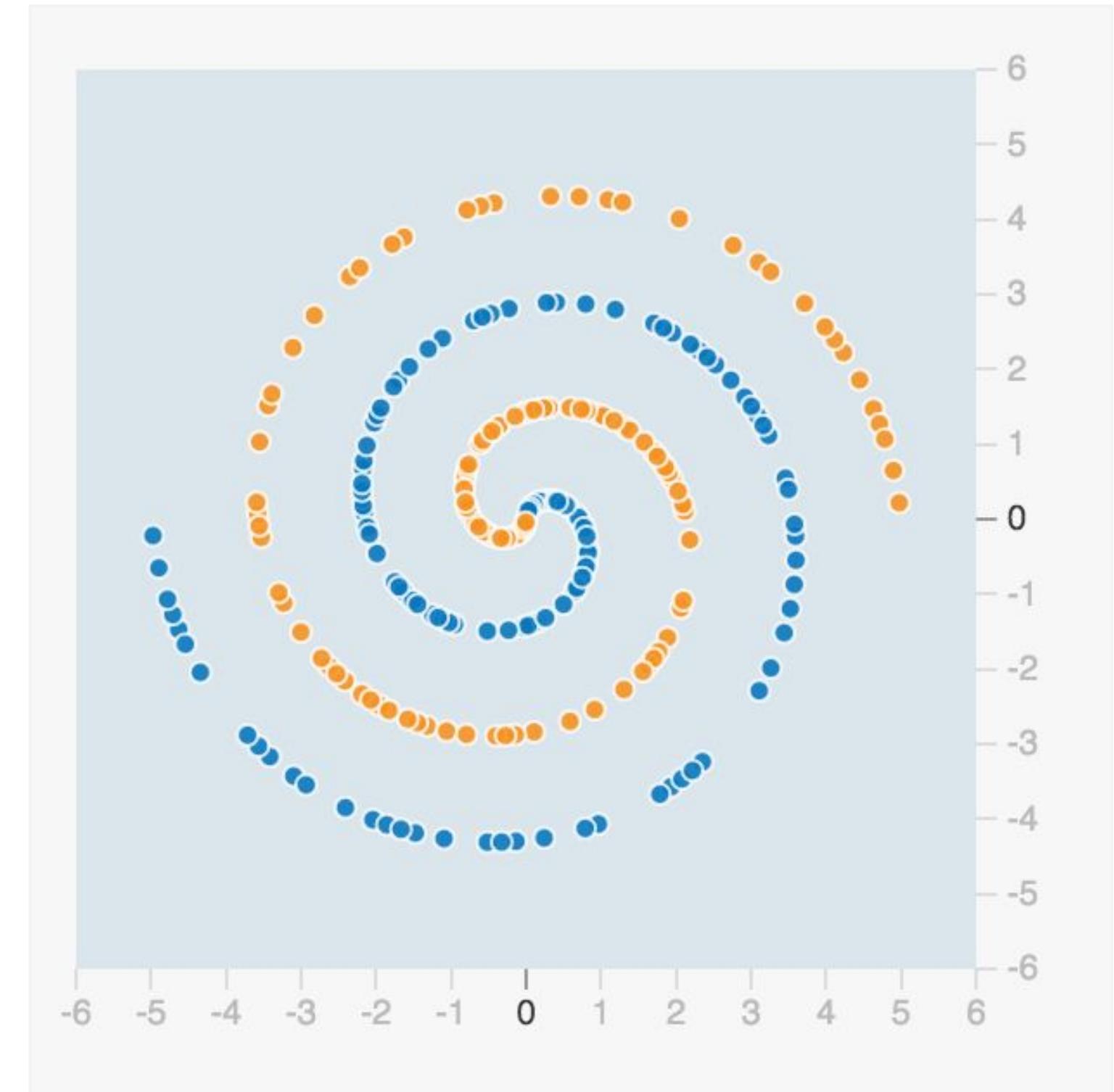
Introducing non-linear features

Output after selecting the X1 squared
and X2 squared features.



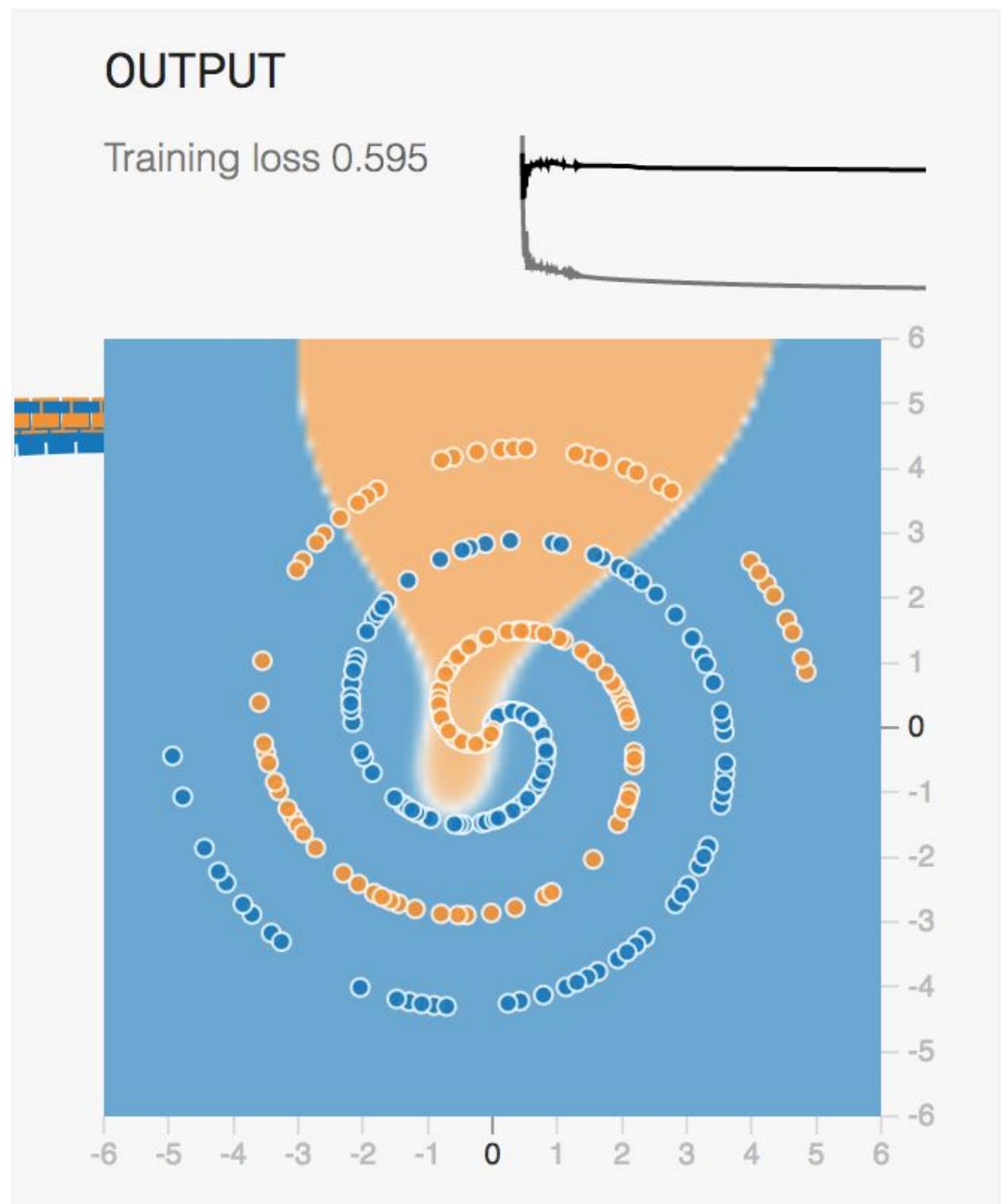
Try learning a
linear model for
this dataset

Open this link:
<https://goo.gl/1v28Pd>



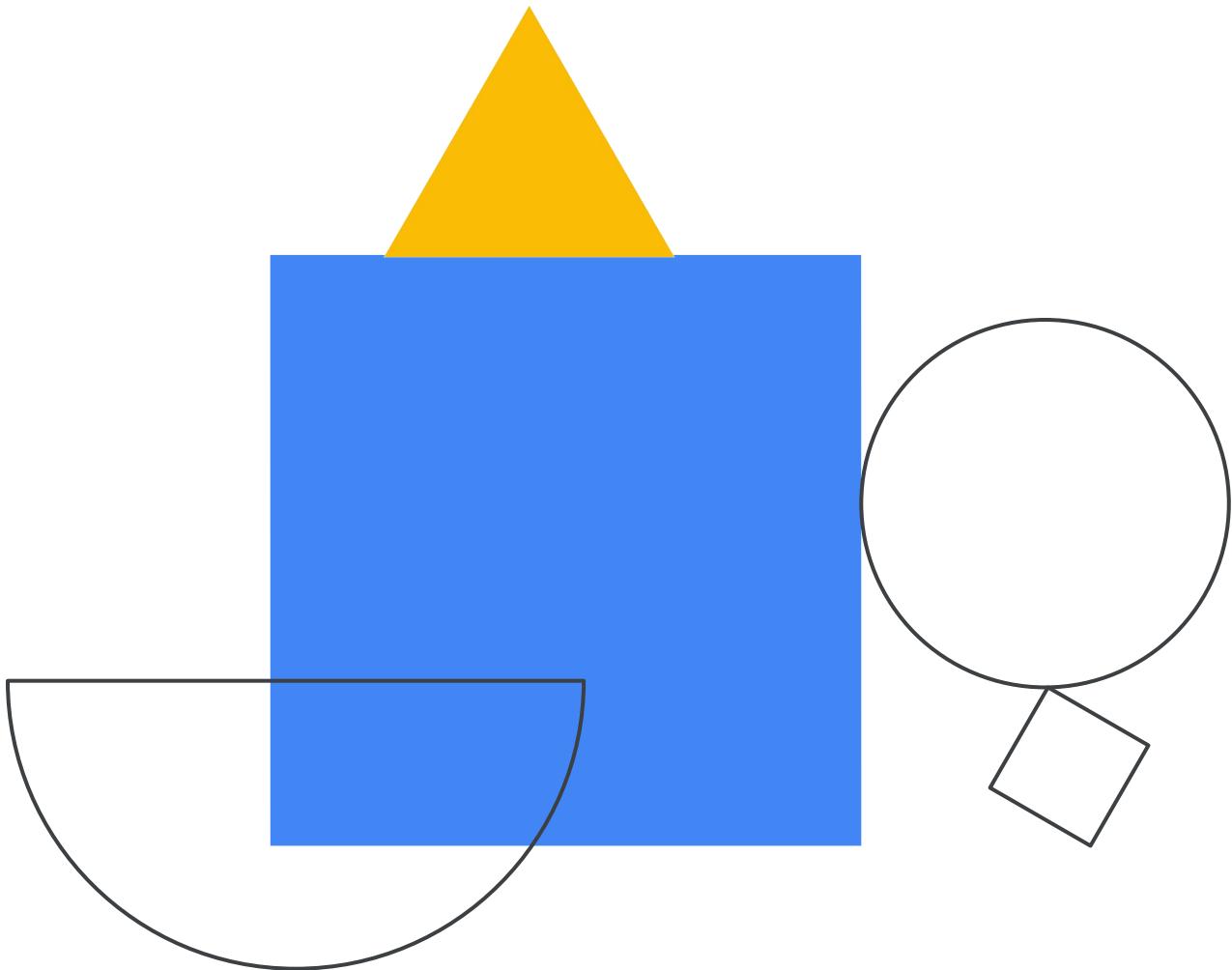
Try learning a
linear model for
this dataset

Open this link:
<https://goo.gl/1v28Pd>



Lab intro

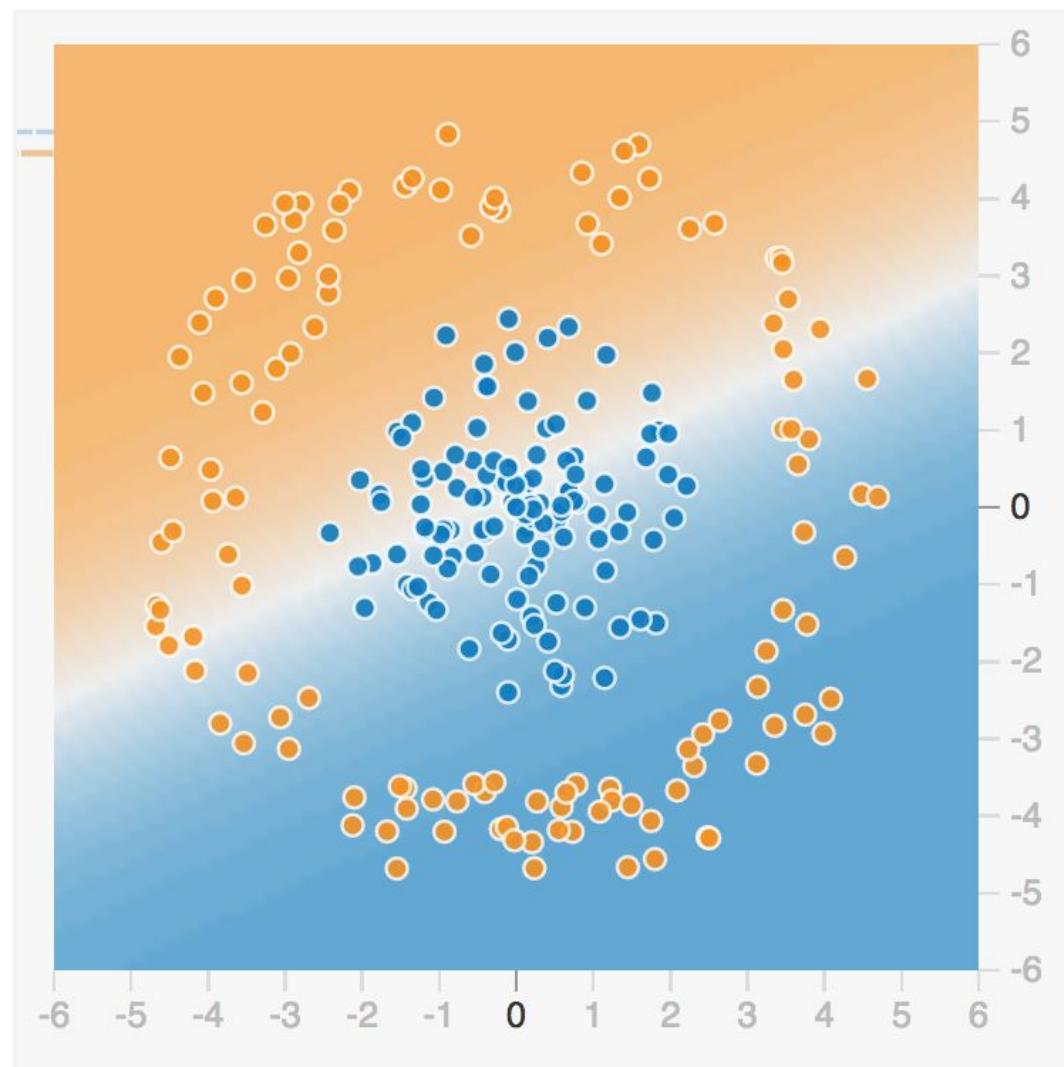
Develop an intuitive understanding of
neural networks using Tensorflow
Playground



Activation function

Try solving this
with a neural
network

<https://goo.gl/VyoRWX>



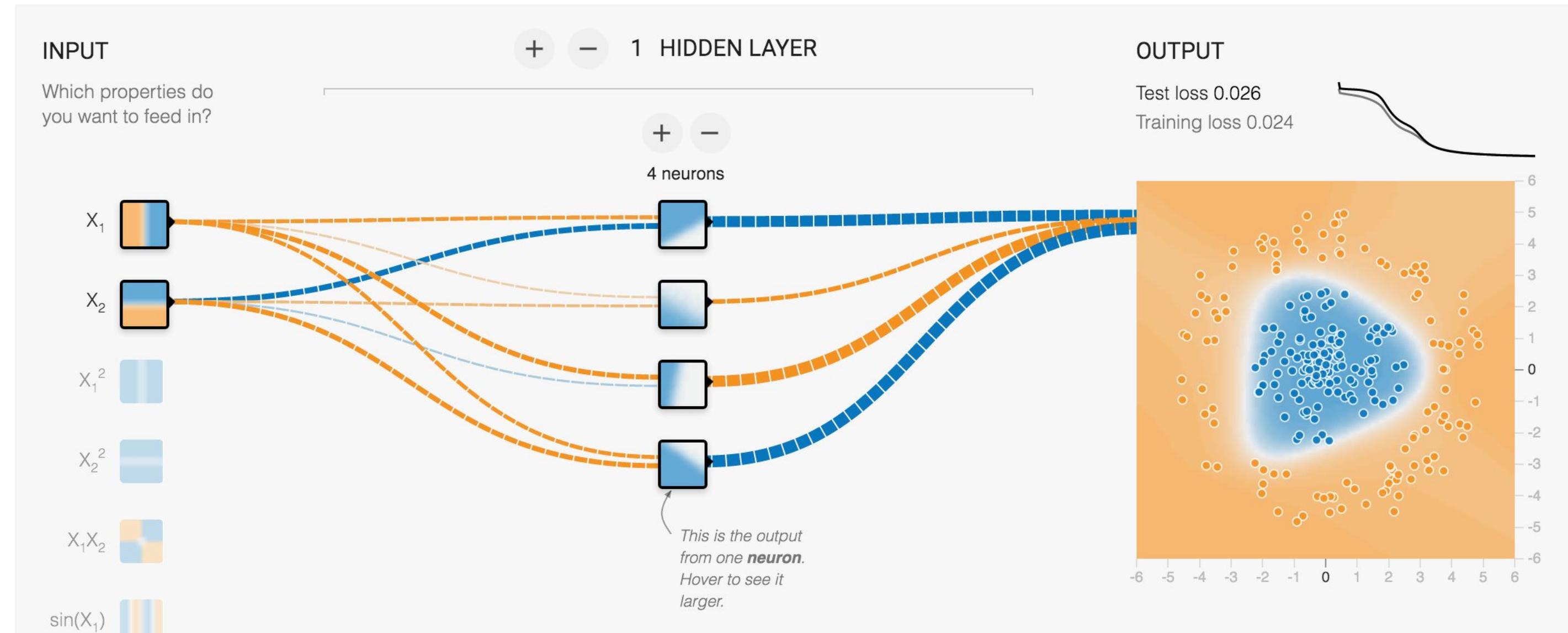
Activation

ReLU

+ - 0 HIDDEN LAYERS

Batch size: 10

More neurons means more input combinations (features)

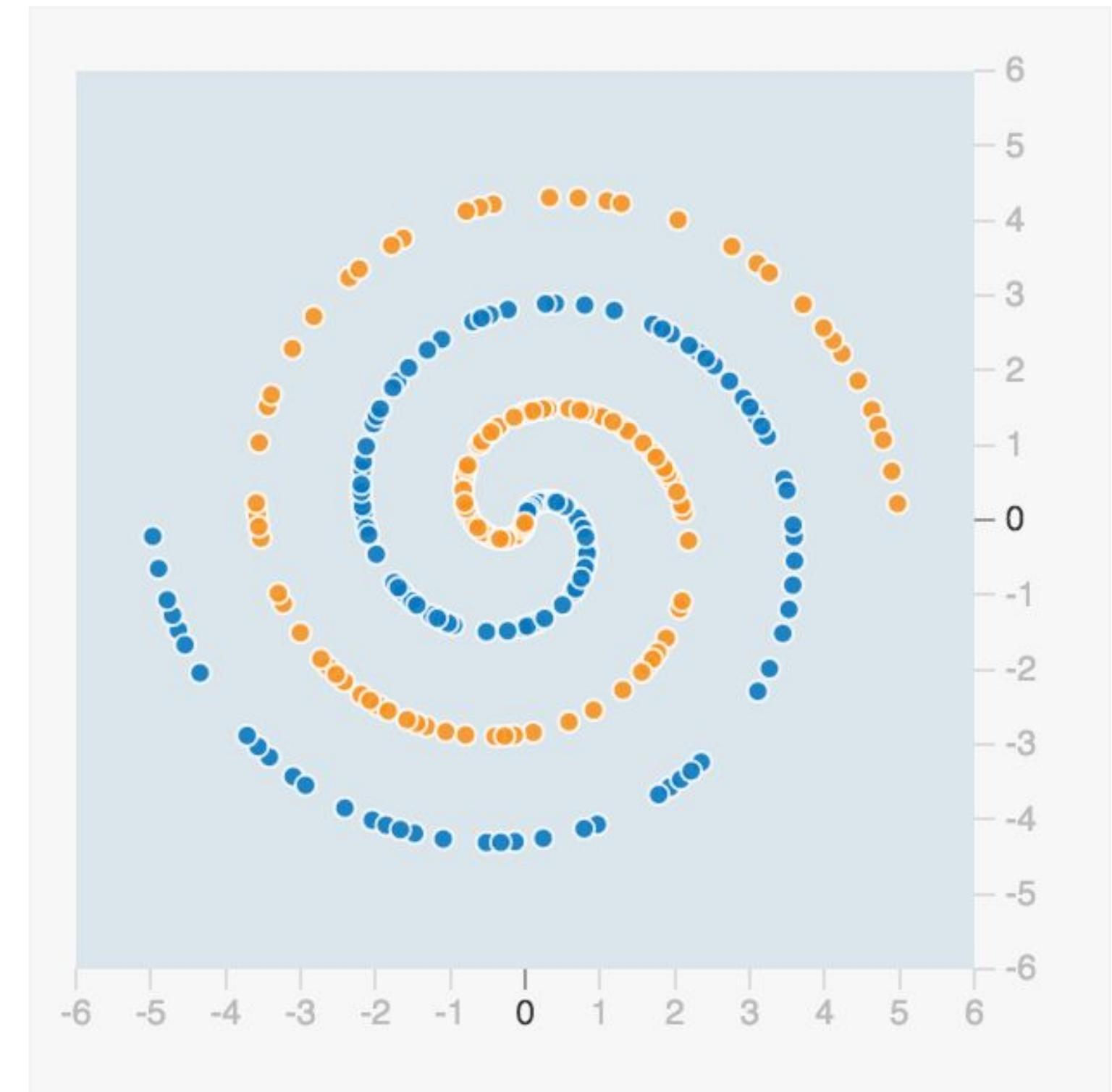


<https://goo.gl/SjQf5V>

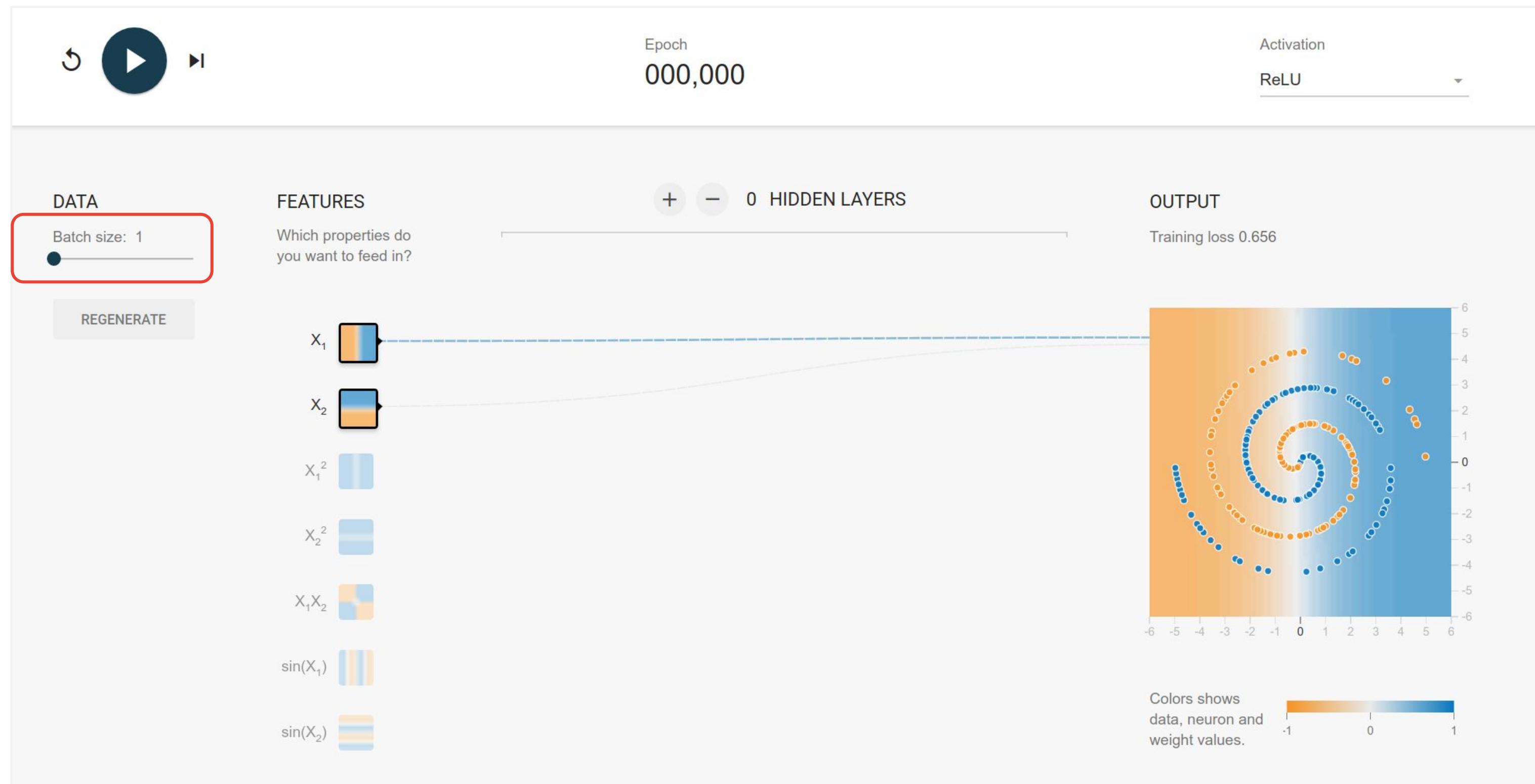
Will a set of lines work?

Open this link:

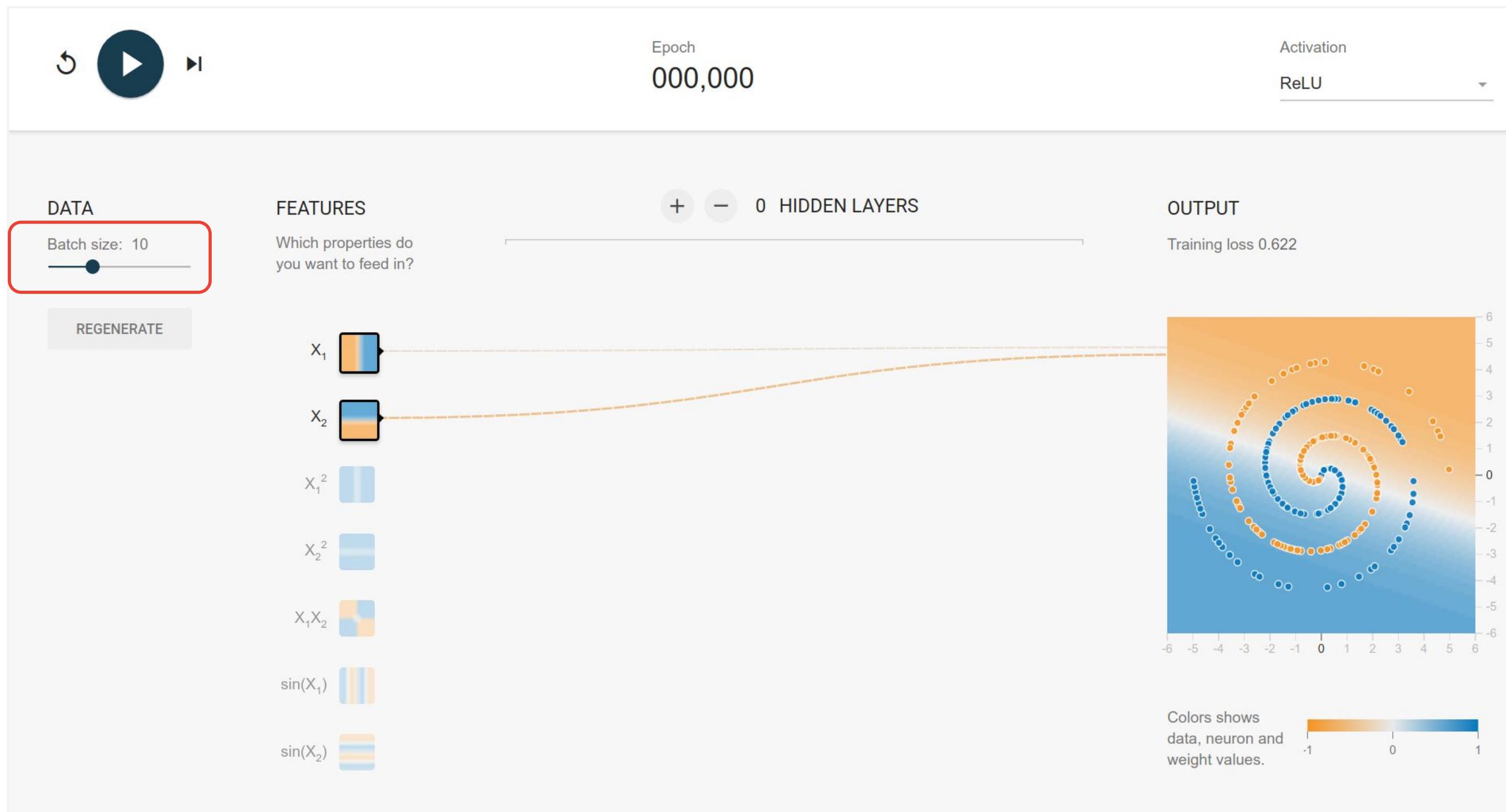
<http://goo.gl/hrXd9T>



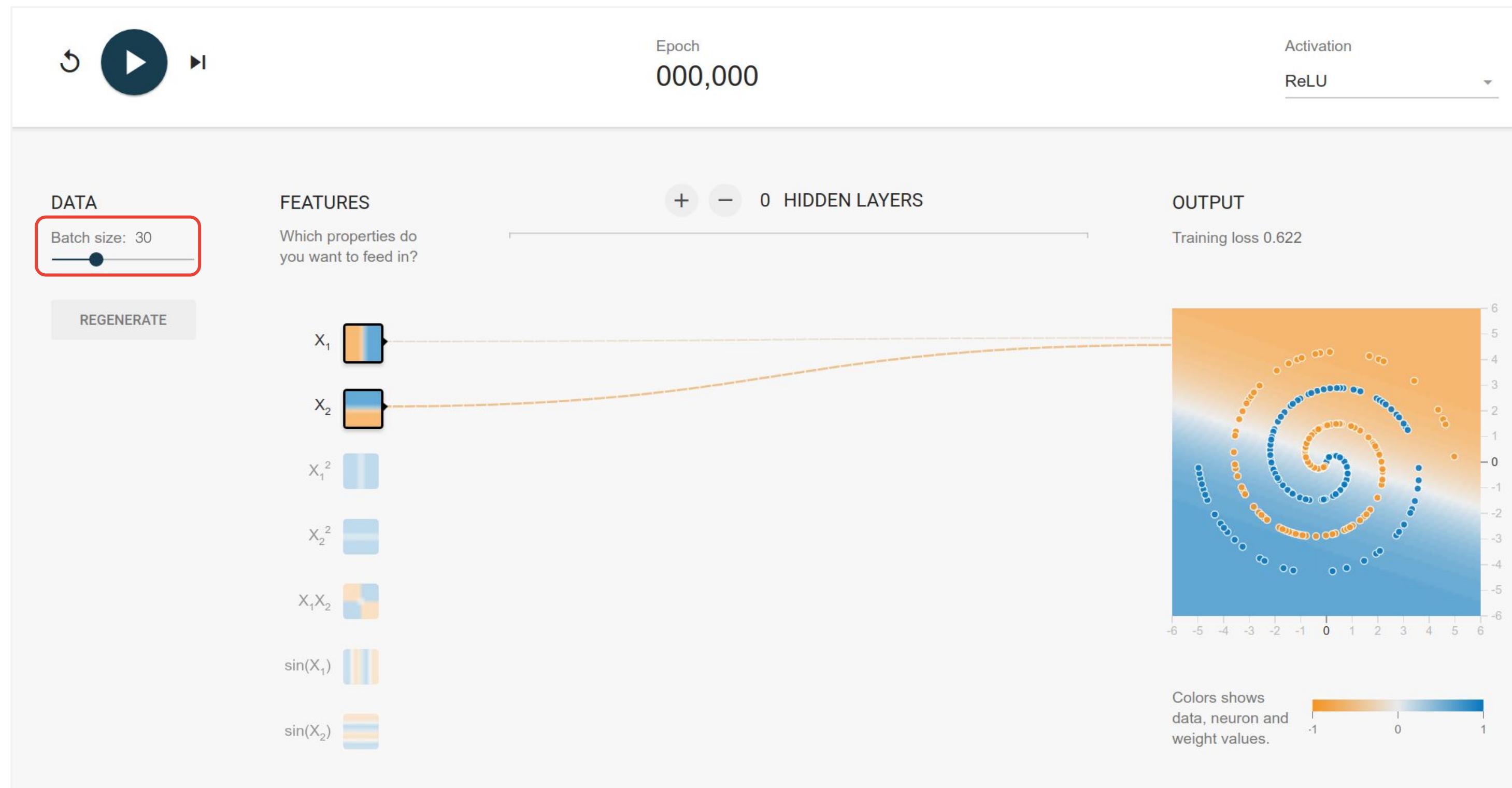
Experimenting with batch sizes 1, 10, and 30



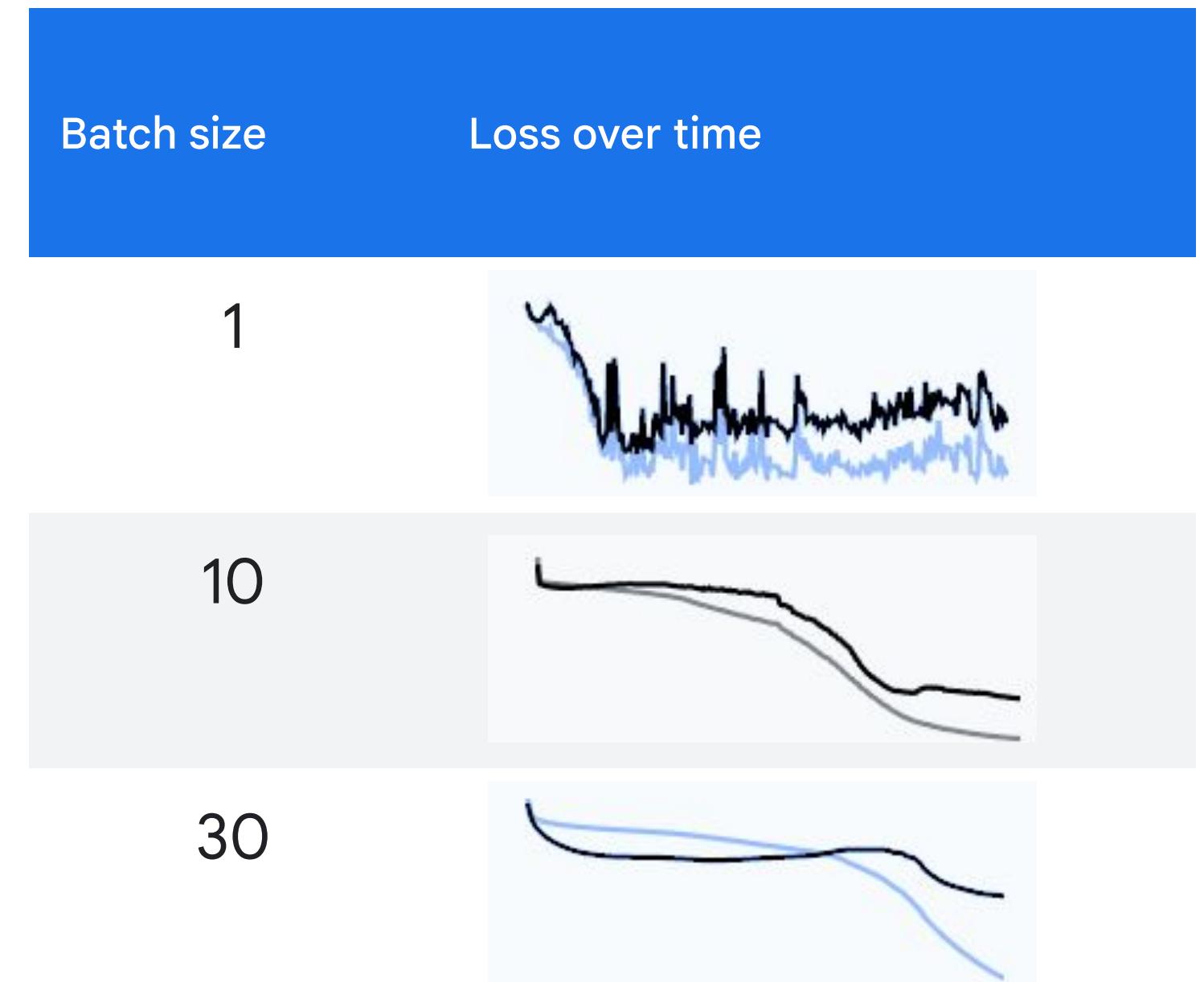
Experimenting with batch sizes 1, 10, and 30



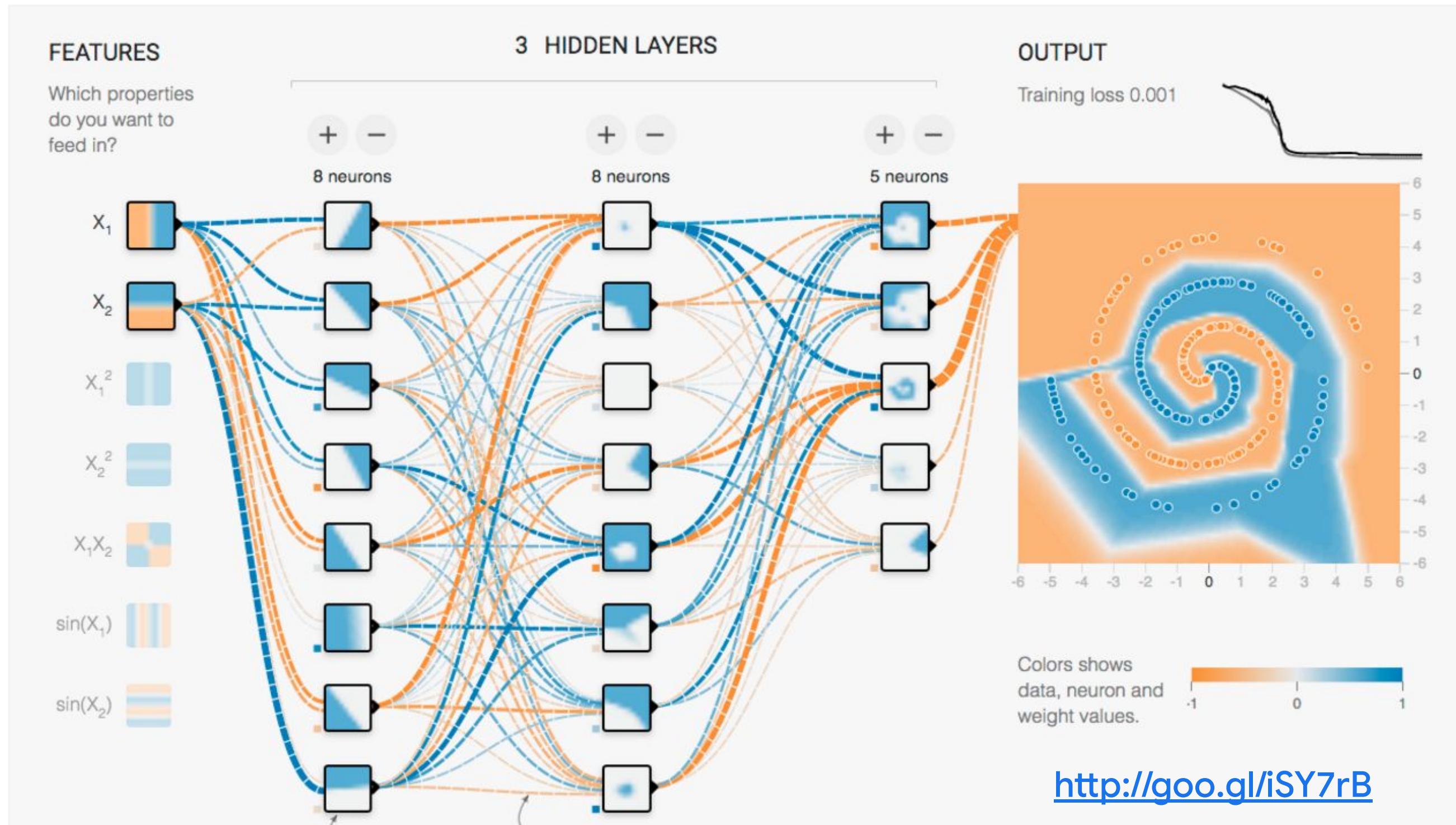
Experimenting with batch sizes 1, 10, and 30



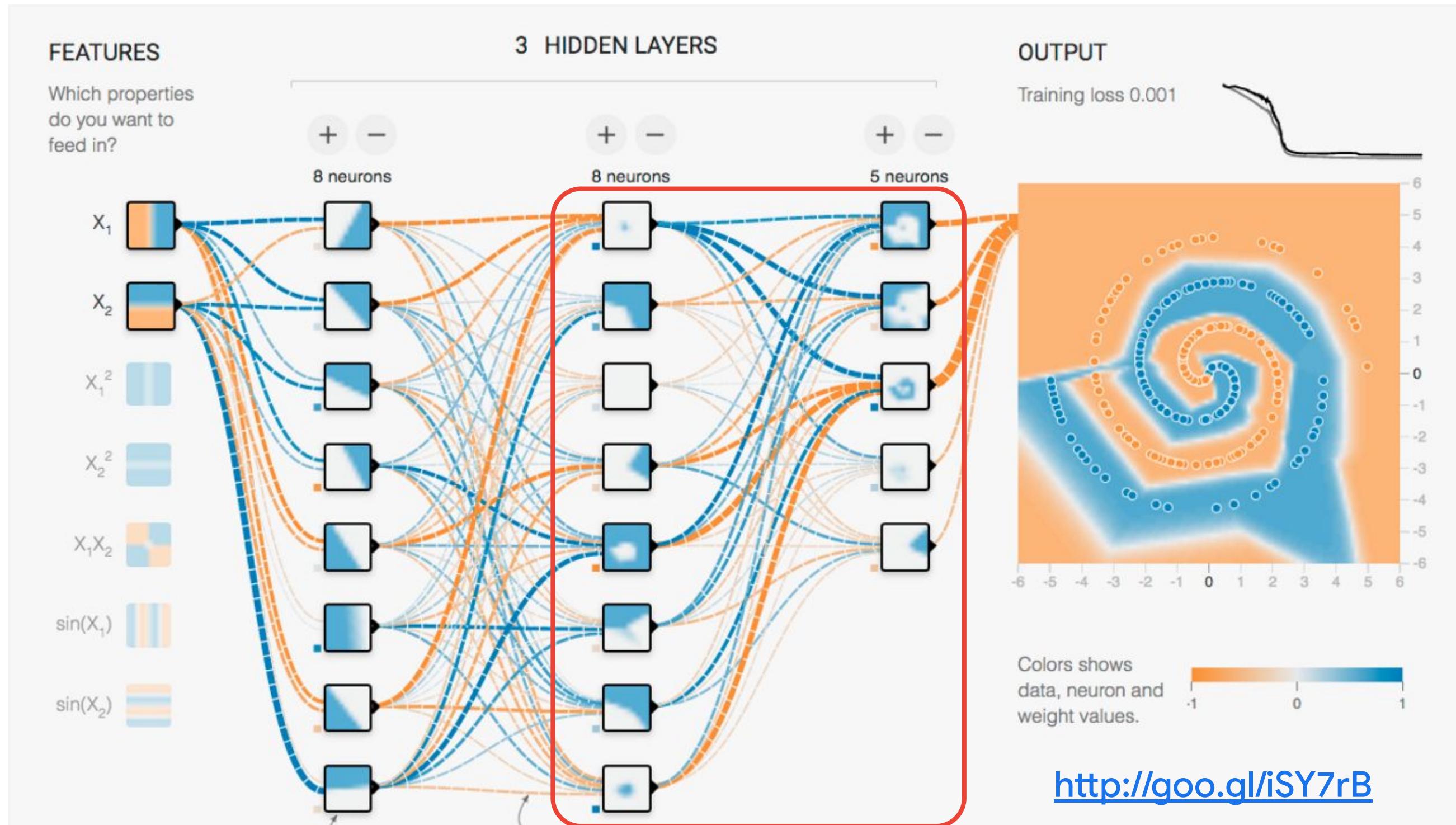
Experimenting with batch size: **Observations**



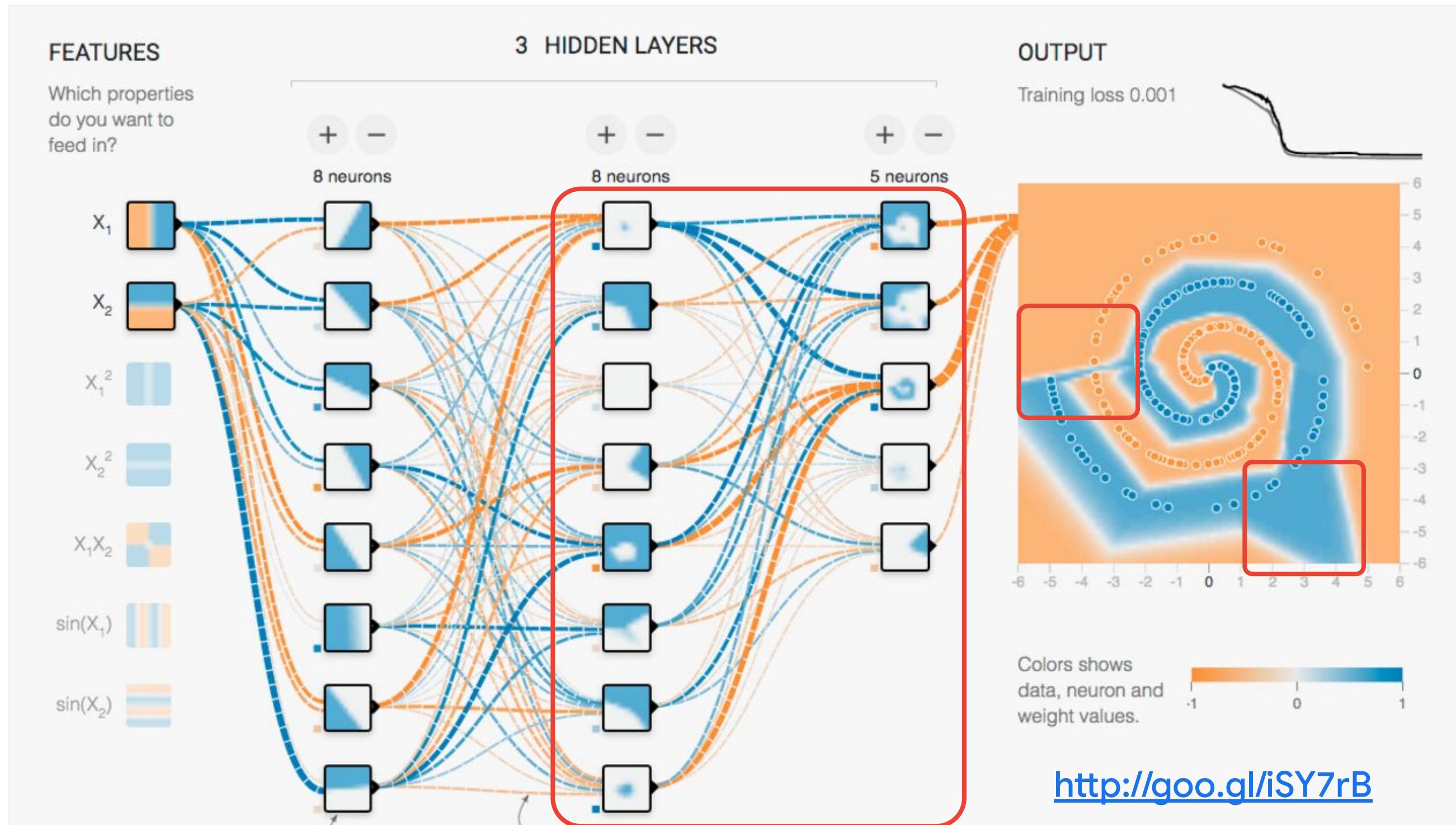
More hidden layers leads to more hierarchies of features



More hidden layers leads to more hierarchies of features



More hidden layers leads to more hierarchies of features

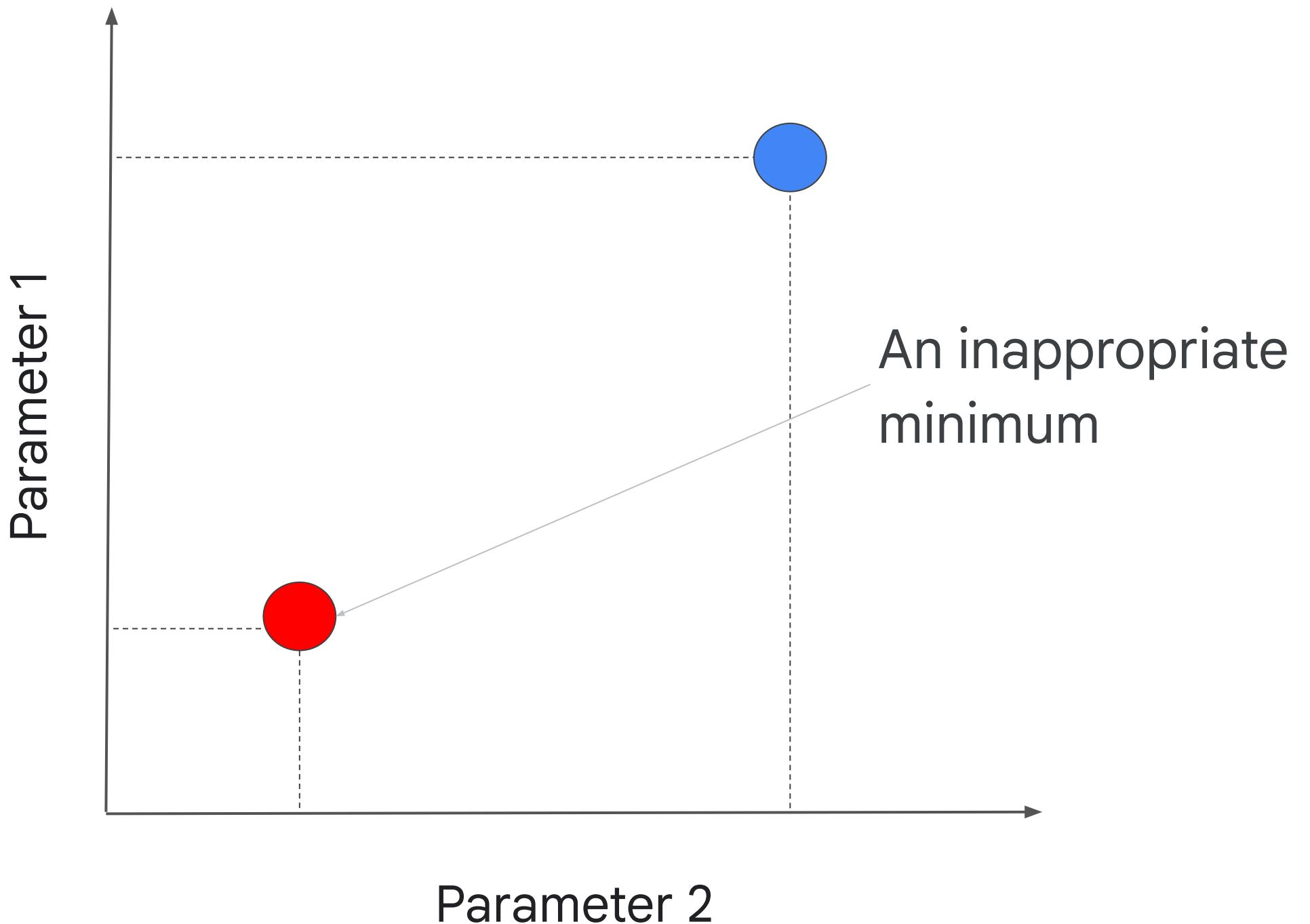


- You trained models using gradient descent.
- Current approach has consequences: long training times, suboptimal minima, and inappropriate minima.

Performance metrics

Inappropriate minima

- Doesn't reflect the relationship between features and label
- Won't generalize well



Skewed data can make inappropriate strategies seductive

1000 parking spaces.

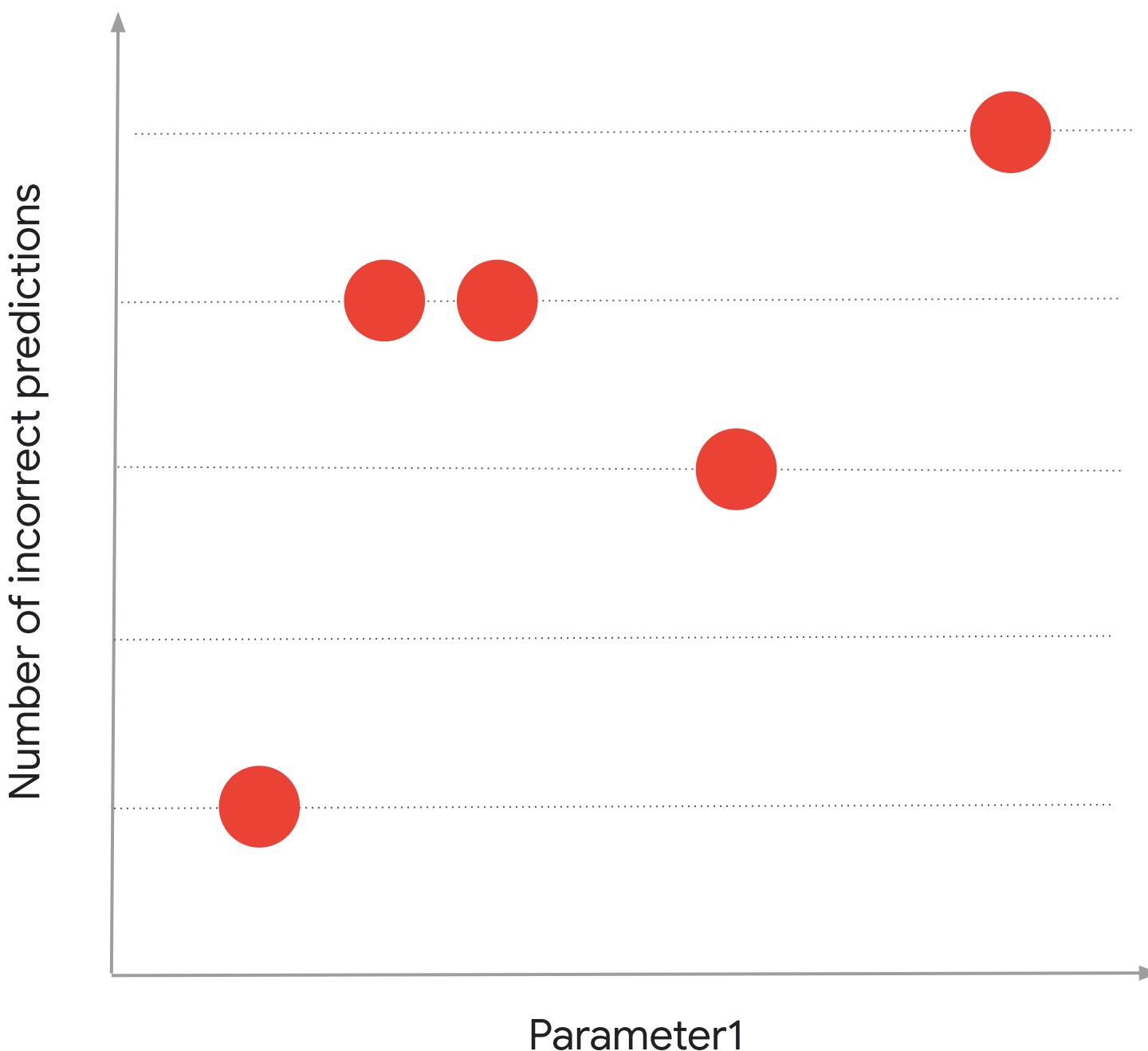
990 of them are taken.

10 are available.

An ML model that always reported that a space was occupied would be right 99/100 times.



In search of a perfect loss function



Performance metrics allow you to measure what matters

Loss functions

- During training
- Harder to understand
- Indirectly connected to business goals

Performance metrics

- After training
- Easier to understand
- Directly connected to business goals

Confusion matrices, precision, and recall

Use a confusion matrix to assess classification model performance

		Model predictions	
		Positive	Negative
Labels	Positive	True positives (TP) Type II error	False negatives (FN) Model says: no
	Negative	False positives (FP) Type I error Model says: yes	True negatives (TN)

True positives (TP)

False positives (FP)

Type I error

Model says: yes

False negatives (FN)

Type II error

Model says: no

True negatives (TN)

True and false positives when predicting parking spots

		Model predictions	
		Positive	Negative
References	Positive	True positives (TP) Available parking space exists . Model predicts it is available.	False negatives (FN) Type II error Available parking space exists. Model doesn't predict it.
	Negative	False positives (FP) Type I error Available parking space doesn't exist . Model predicts it is available.	True negatives (TN) Available parking space doesn't exist. Model correctly doesn't predict it.

True and false positives when predicting parking spots

		Model predictions	
		Positive	Negative
References	Positive	True positives (TP) Available parking space exists . Model predicts it is available.	False negatives (FN) Type II error Available parking space exists. Model doesn't predict it.
	Negative	False positives (FP) Type I error Available parking space doesn't exist . Model predicts it is available.	True negatives (TN) Available parking space doesn't exist. Model correctly doesn't predict it.

Precision: True positives / total classified as positive

		Model predictions	
		Positive	Negative
References	Positive	True positives (TP) Available parking space exists. Model predicts it is available.	False negatives (FN) Type II error Available parking space exists. Model doesn't predict it.
	Negative	False positives (FP) Type I error Available parking space doesn't exist. Model predicts it is available.	True negatives (TN) Available parking space doesn't exist. Model correctly doesn't predict it.

Precision

Recall: True positives / all actual positives in our reference

		Model predictions	
		Positive	Negative
References	Positive	True positives (TP) Available parking space exists. Model predicts it is available.	False negatives (FN) Type II error Available parking space exists. Model doesn't predict it.
	Negative	False positives (FP) Type I error Available parking space doesn't exist. Model predicts it is available.	True negatives (TN) Available parking space doesn't exist. Model correctly doesn't predict it.

Recall

Classify each
image as either
“cat” or “not cat”



Classify each
image as either
“cat” or “not cat”



You: Cat



You: Not cat



You: Not cat



You: Cat



You: Cat



You: Cat



You: Not cat



You: Not cat

Hypothetical results from your classification ML model



You: Cat
ML: Not cat



You: Not cat
ML: Cat



You: Not cat
ML: Cat



You: Cat
ML: Not cat



You: Cat
ML: Cat



You: Not cat
ML: Cat



You: Not cat
ML: Not cat



You: Cat
ML: Cat

Hypothetical results from your classification ML model

Accuracy = 3 / 8 (0.375)



You: Cat
ML: Not cat



You: Not cat
ML: Cat



You: Not cat
ML: Cat



You: Cat
ML: Not cat



You: Cat
ML: Cat



You: Not cat
ML: Cat



You: Not cat
ML: Not cat



You: Cat
ML: Cat

How precise was your “cat” ML model?



You: Not cat
ML: Cat



You: Not cat
ML: Cat



You: Cat
ML: Cat



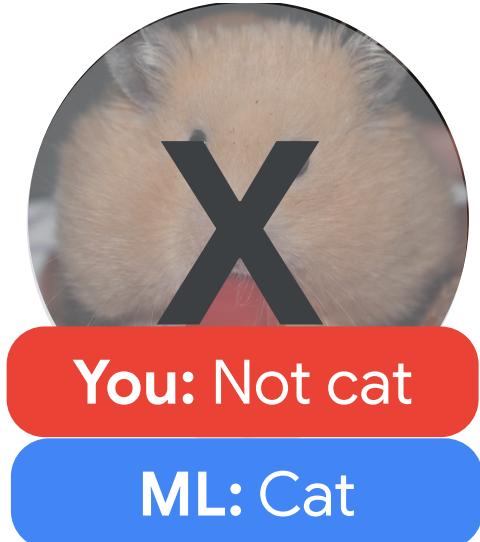
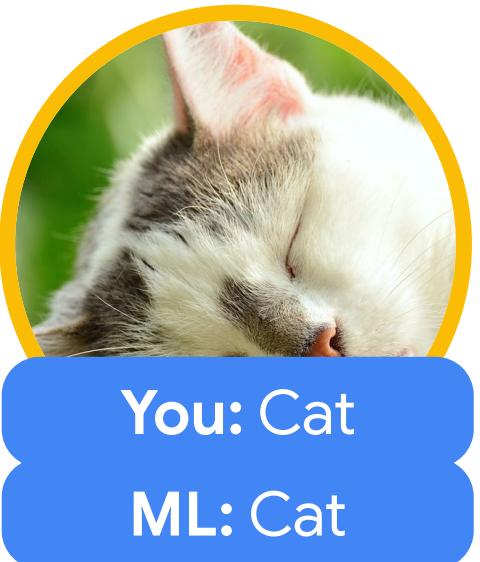
You: Not cat
ML: Cat



You: Cat
ML: Cat

How precise was your “cat” ML model?

Precision = TP / (TP + FP) = 2 / 5 (0.40)



What recall did our model have?



You: Cat
ML: Not cat



You: Cat
ML: Not cat



You: Cat
ML: Cat



You: Cat
ML: Cat

What recall did our model have?

Recall = TP / (TP + FN) = 2 / 4 (0.50)



You: Cat

ML: Not cat



You: Cat

ML: Not cat



You: Cat

ML: Cat



You: Cat

ML: Cat

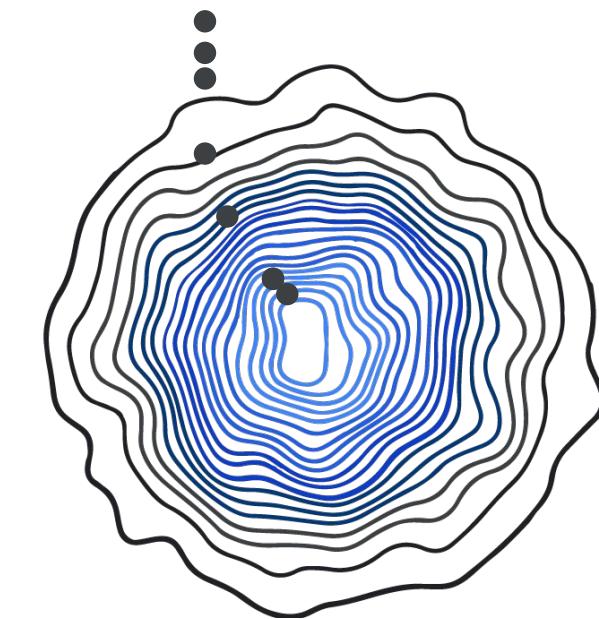
Optimization identifies the best ML model parameters

$$y = B + W^T \times X$$

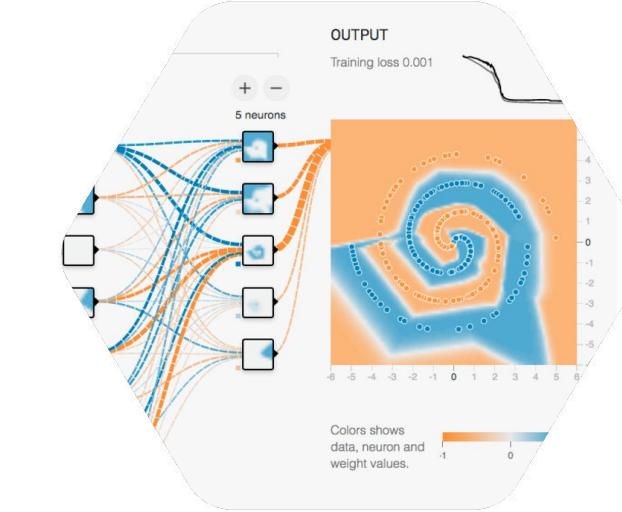
Output Bias term Weight Input

Model parameters

Models are sets of parameters and hyper-parameters.

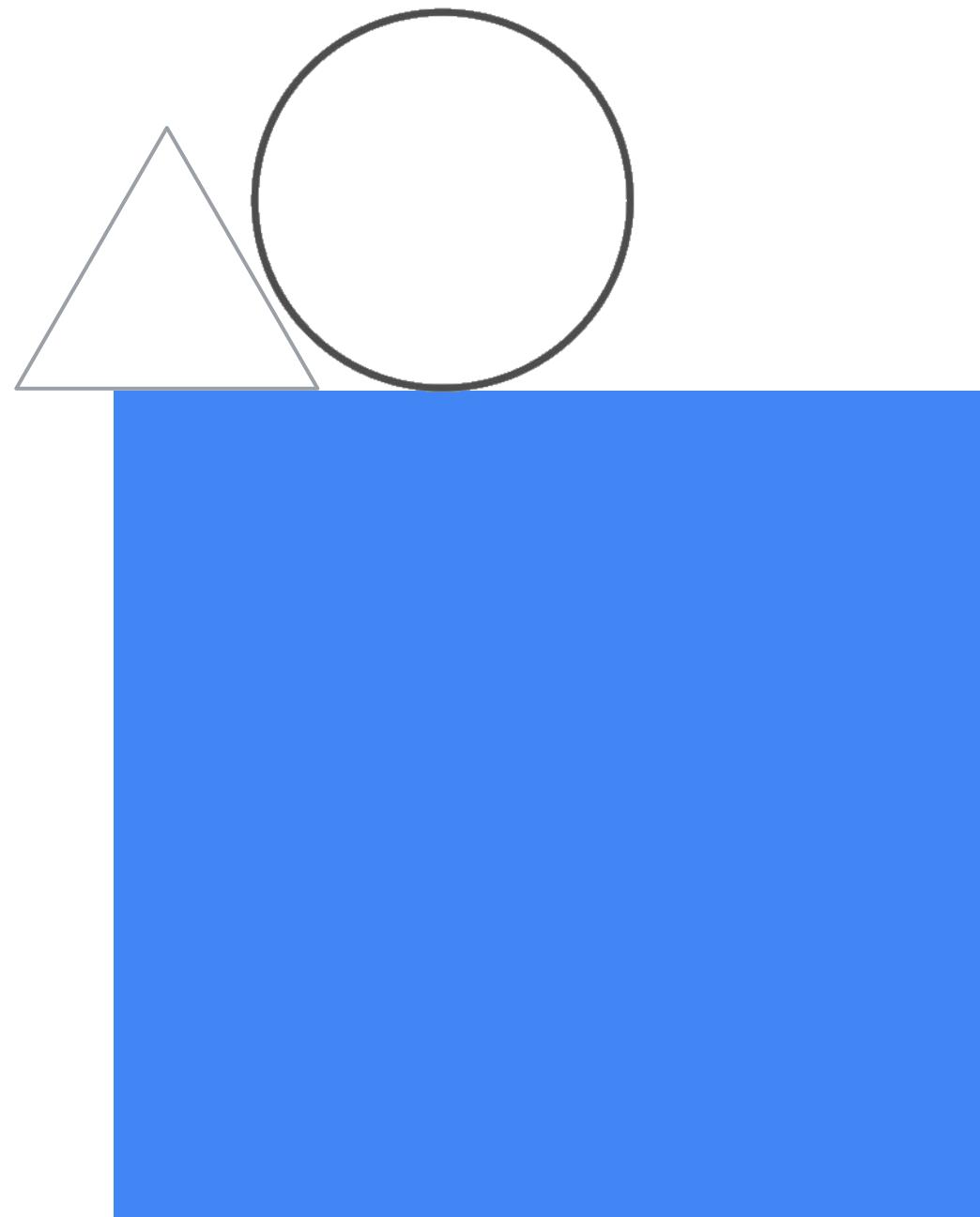


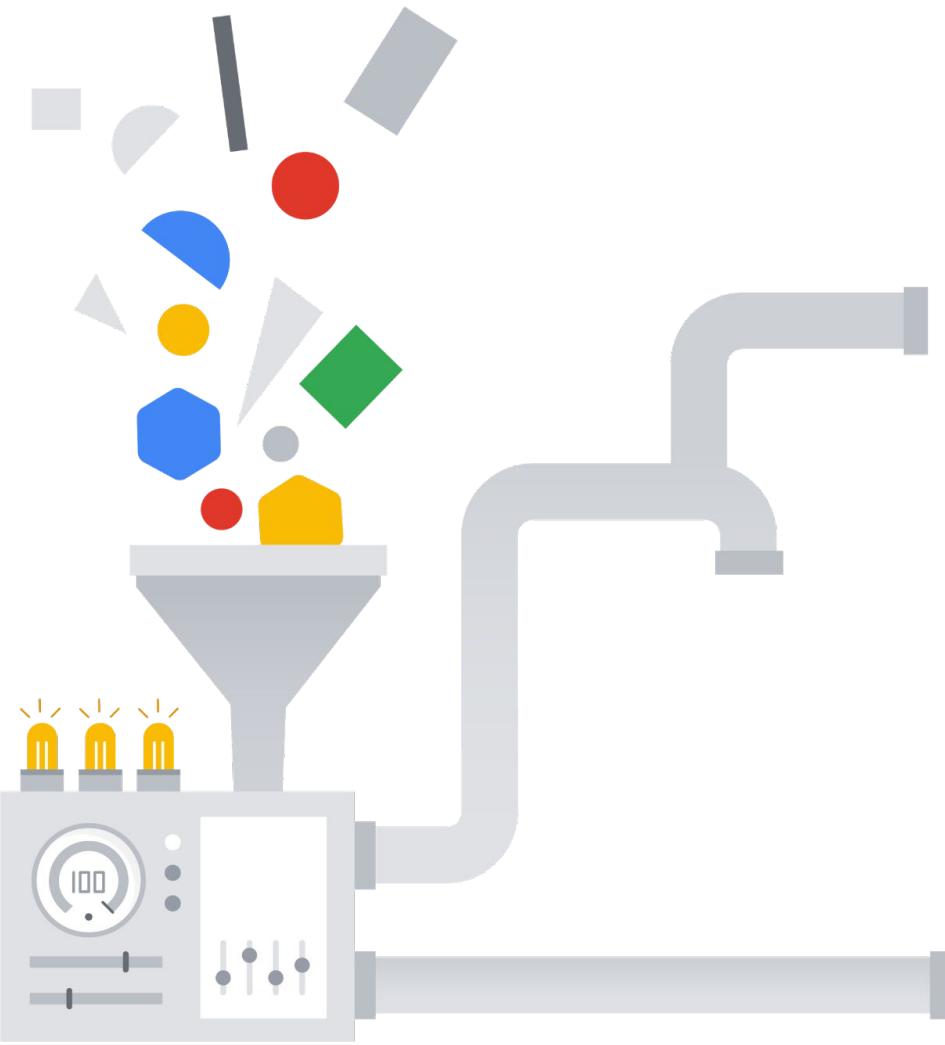
Find the best parameters by optimizing loss functions through gradient descent.



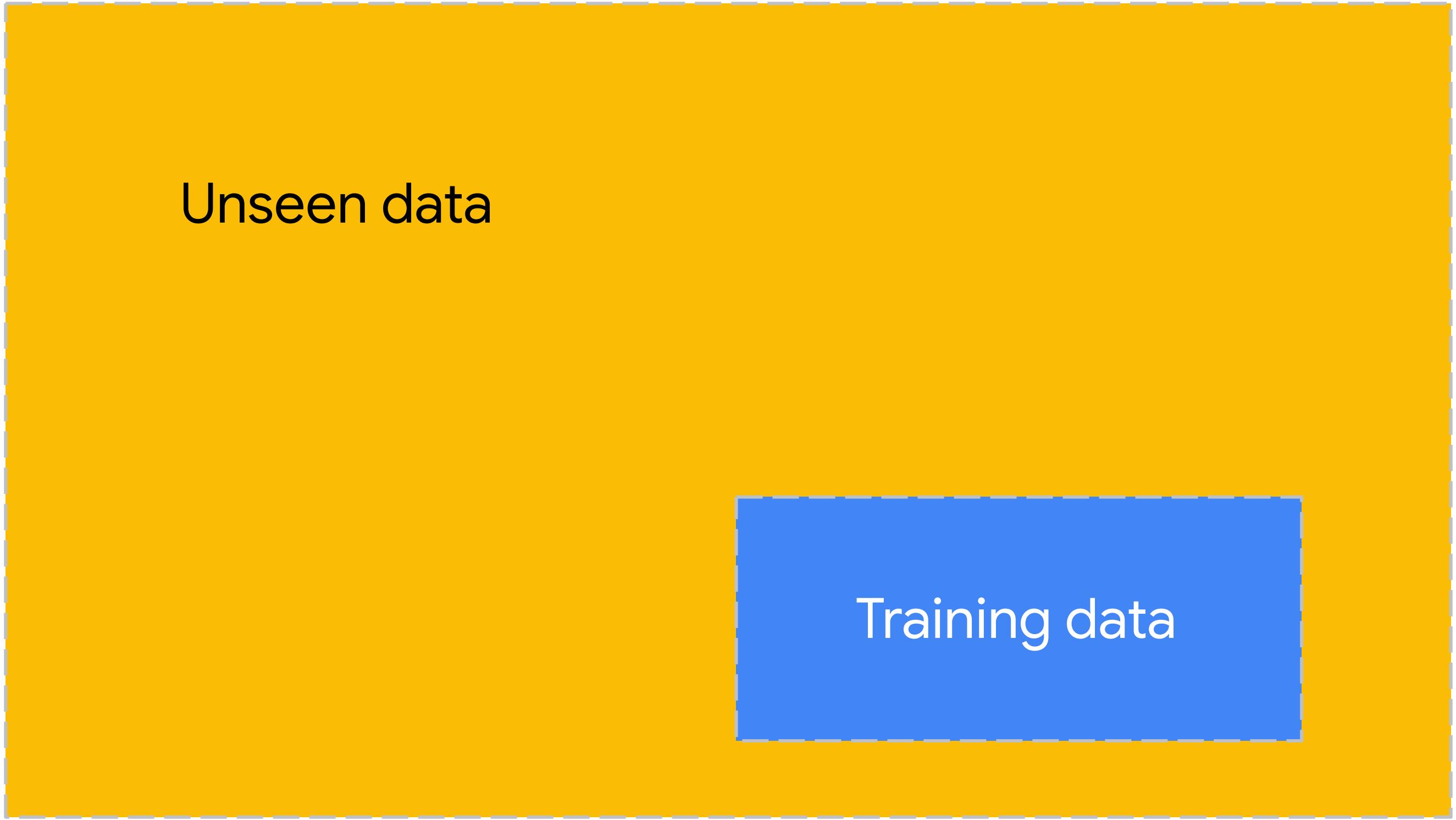
Experiment with neural networks in the TensorFlow playground.

Generalization and Sampling





When is the most accurate ML
model **not** the right one to pick?



Unseen data

Training data

In this module, you learn to ...

01

Assess if your model is overfitting

02

Gauge when to stop model training

03

Create repeatable training, evaluation,
and test datasets

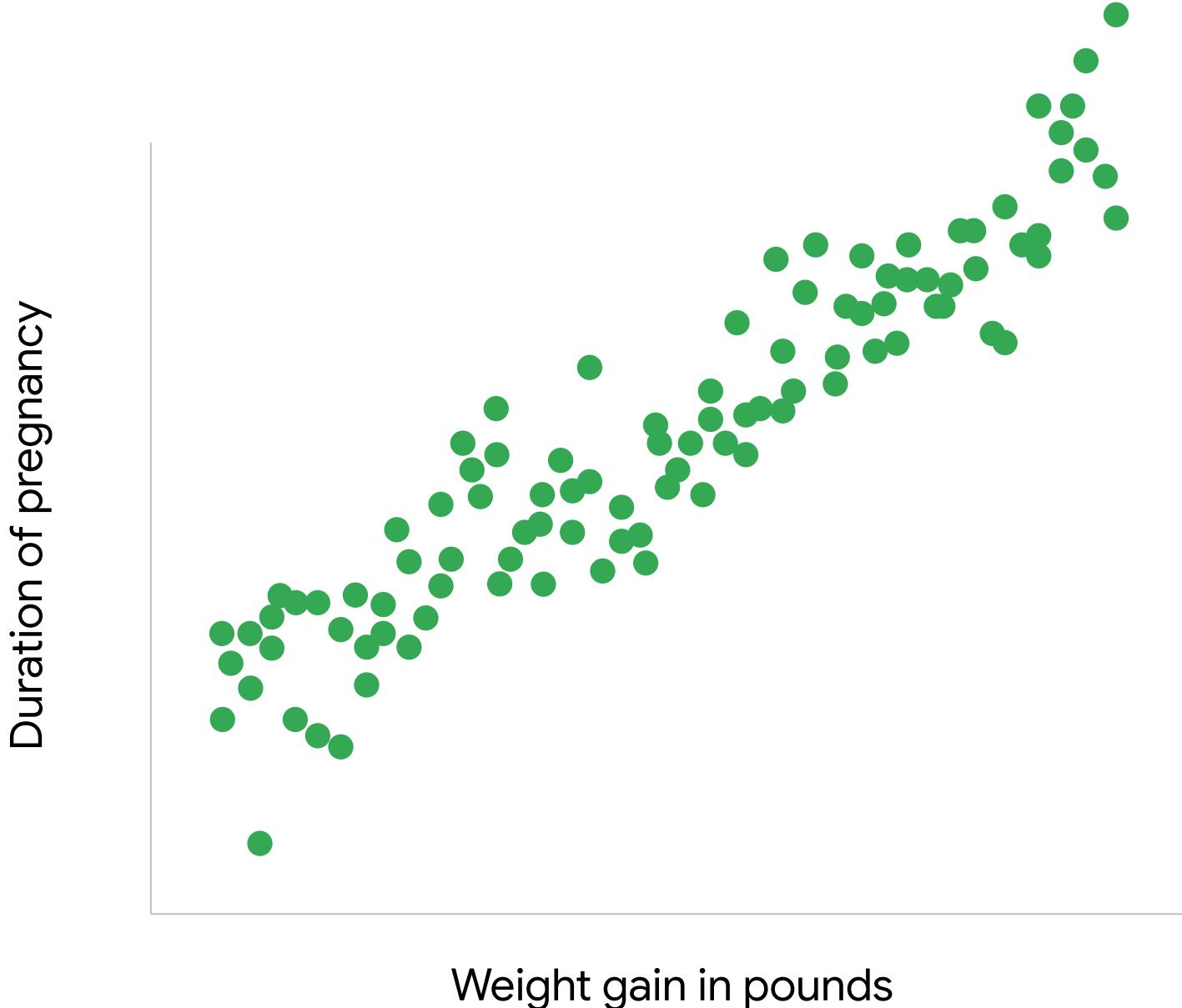
04

Establish performance benchmarks



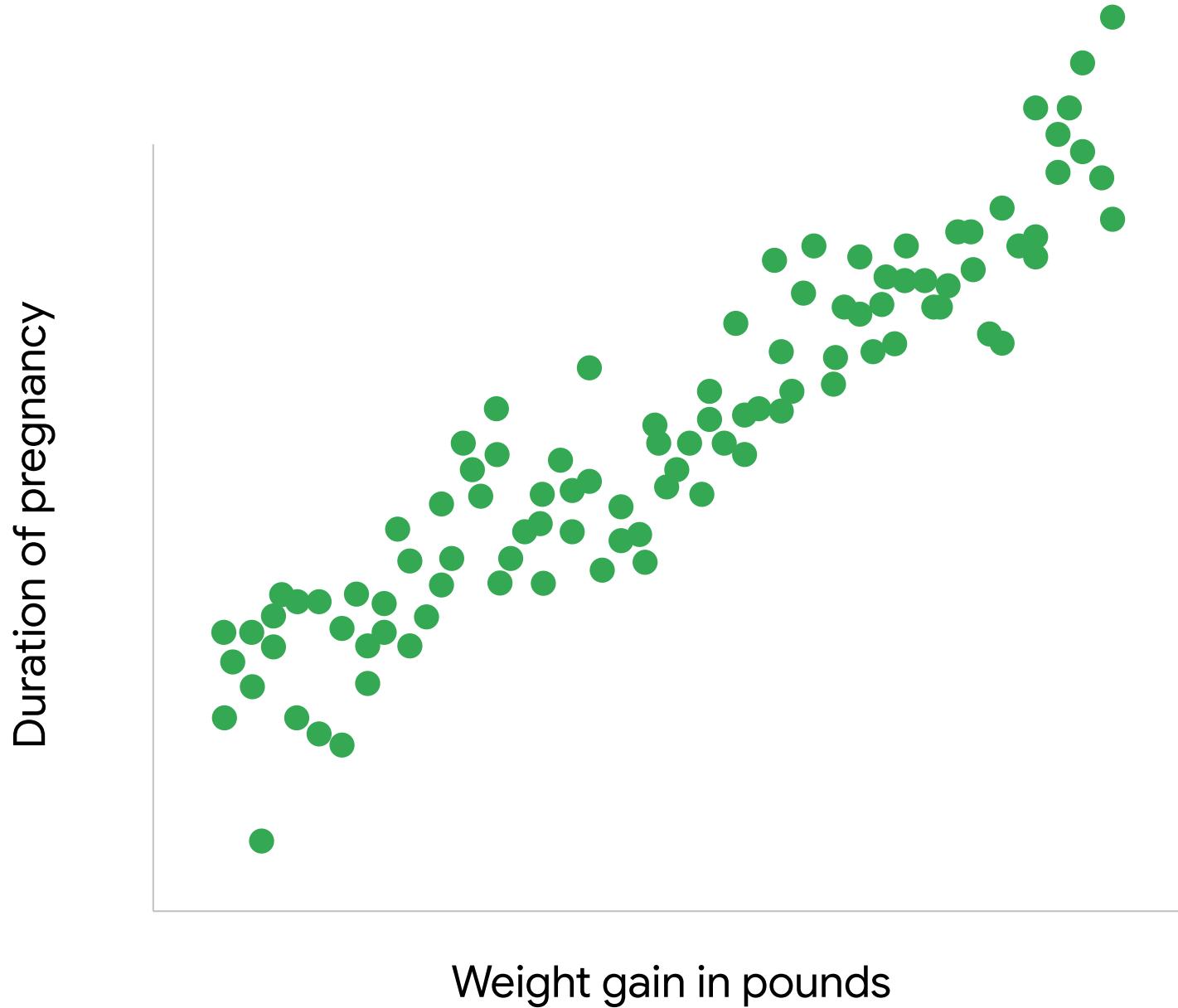
Generalization

Suppose we want to predict duration of pregnancy based on mother's weight gain in pounds



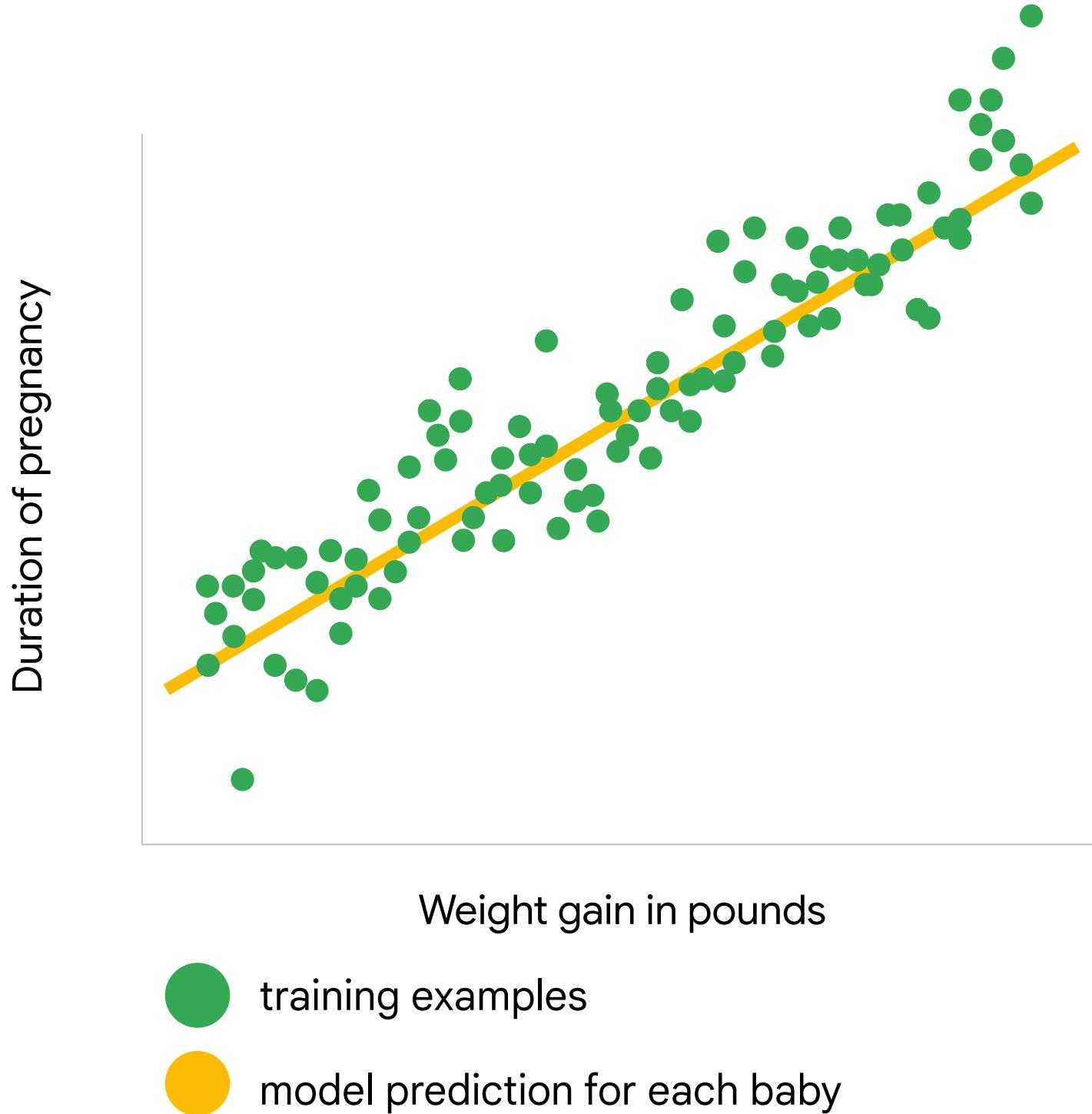
Suppose we want to predict duration of pregnancy based on mother's weight gain in pounds

What is the error measure to optimize?



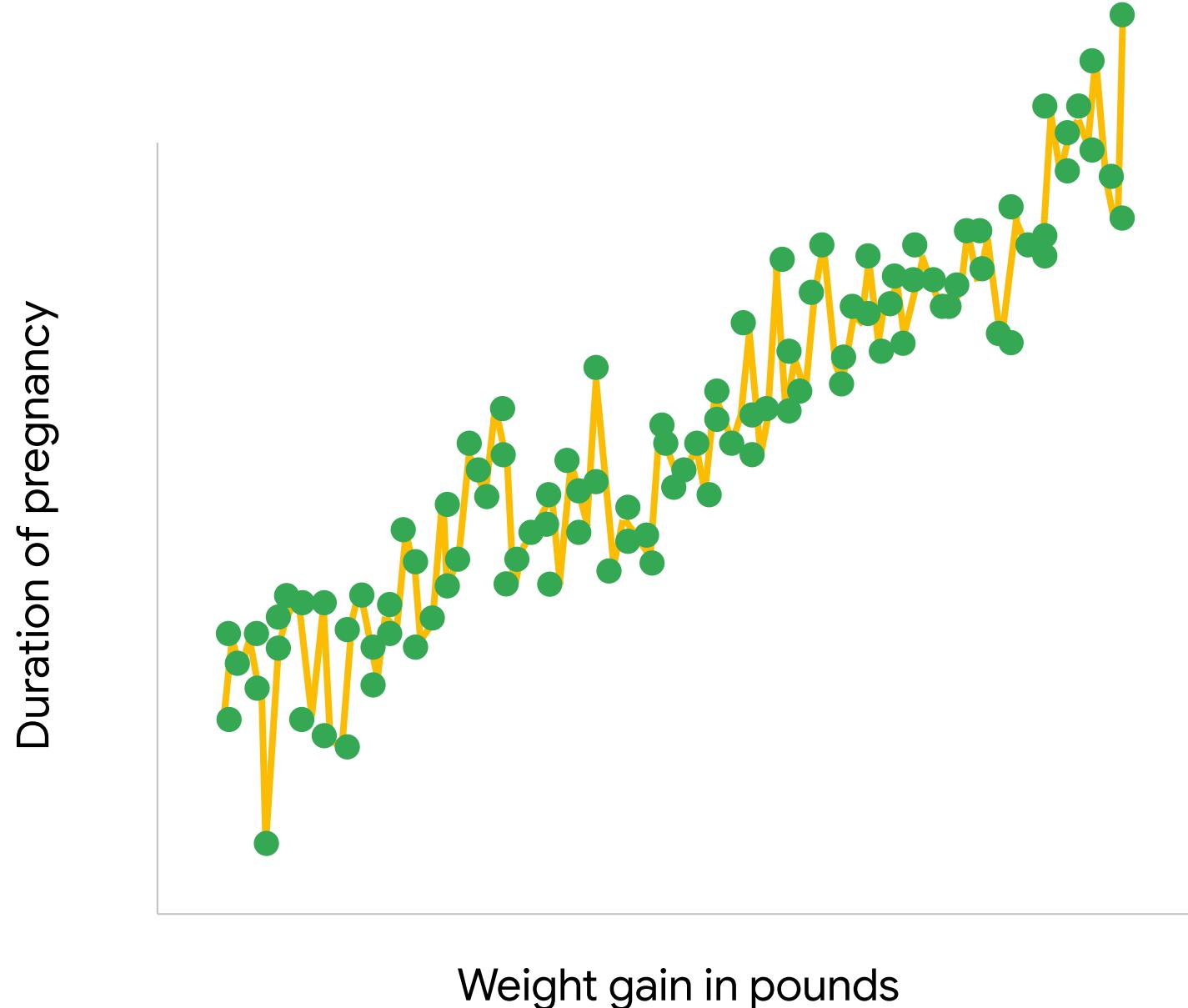
Model 1 is a linear
model using linear
regression

RMSE = 2.224



Model 2 has more
free parameters

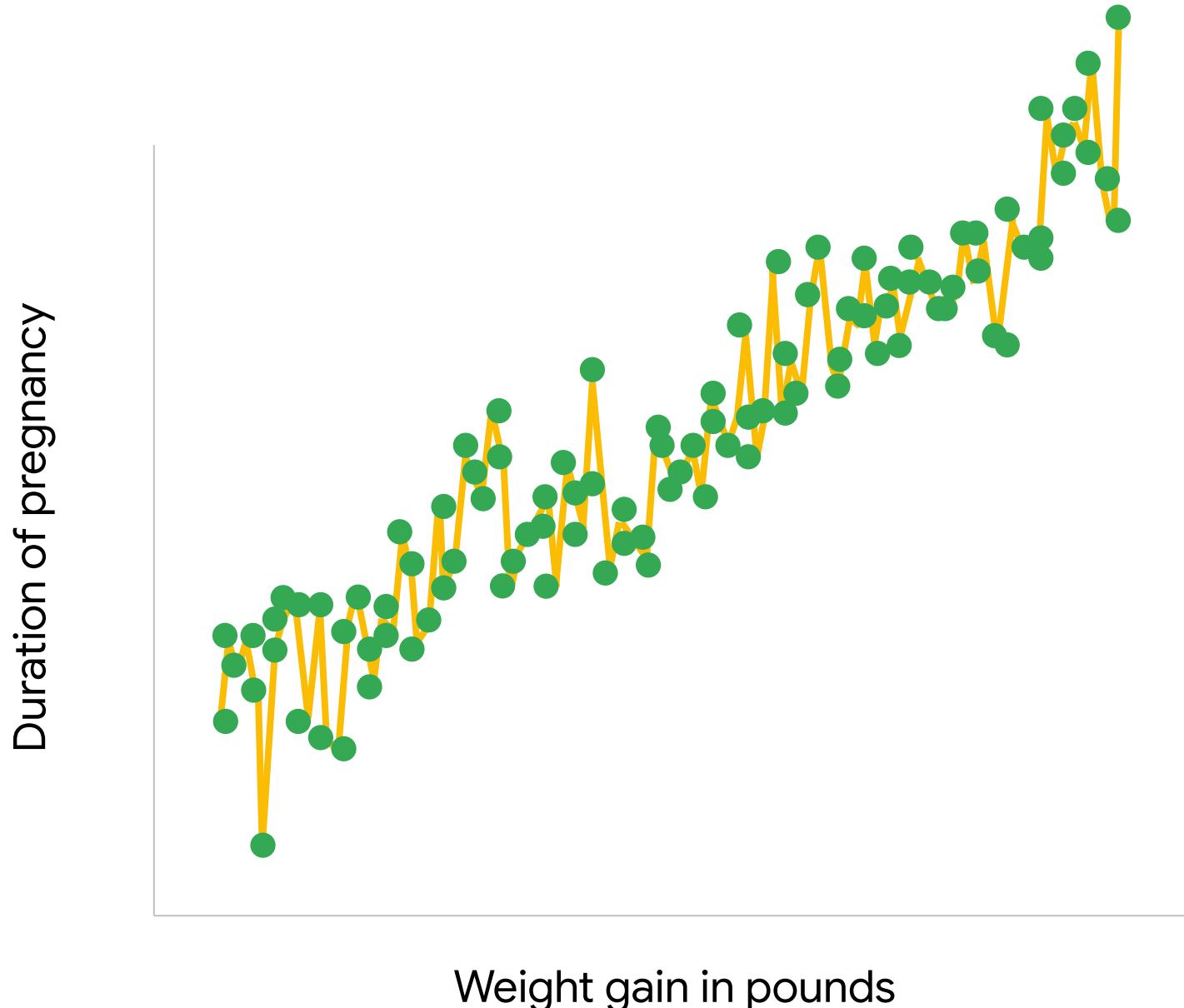
RMSE = 0



Model 2 has more
free parameters

RMSE = 0

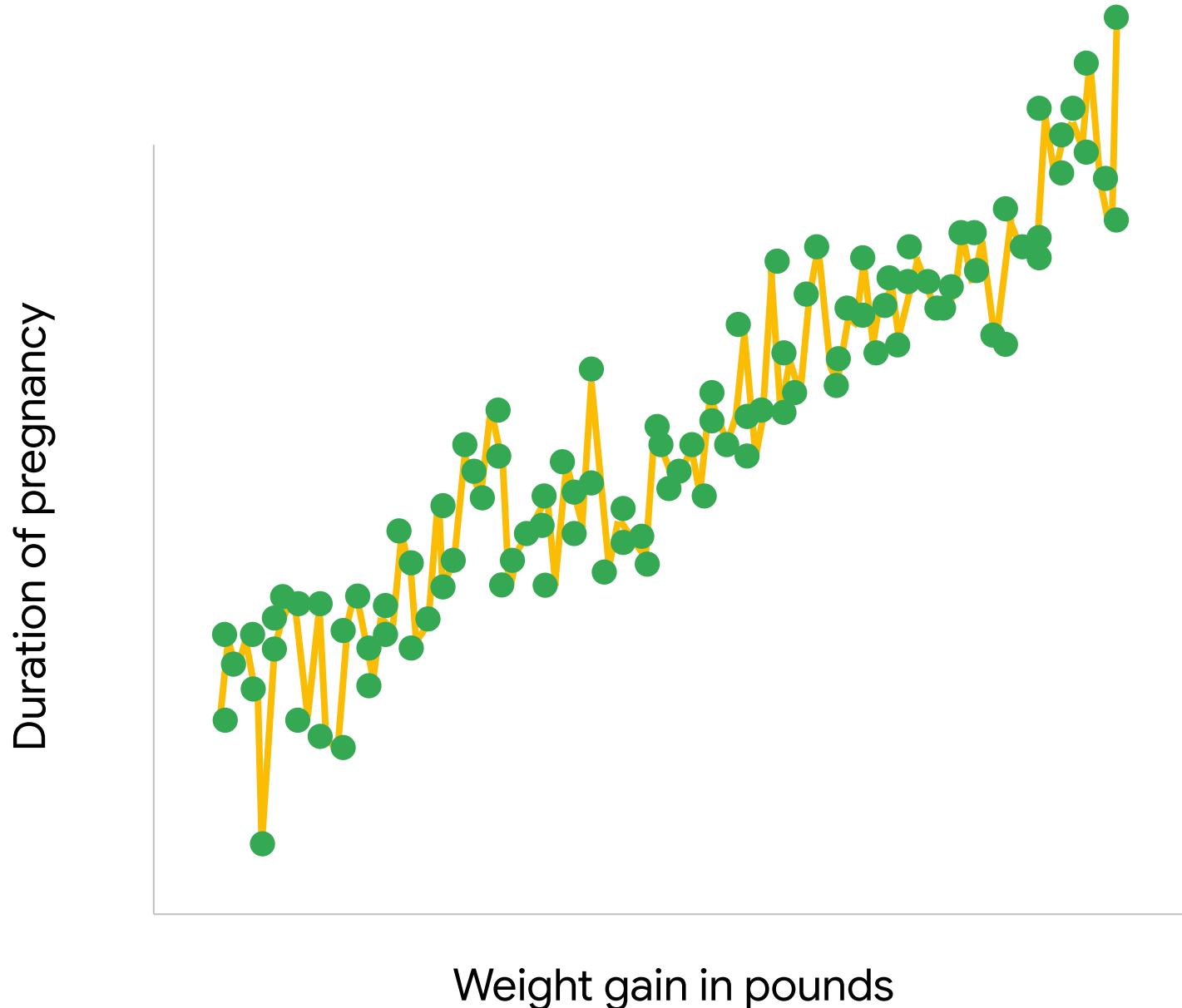
Which model is better?



Model 2 has more
free parameters

RMSE = 0

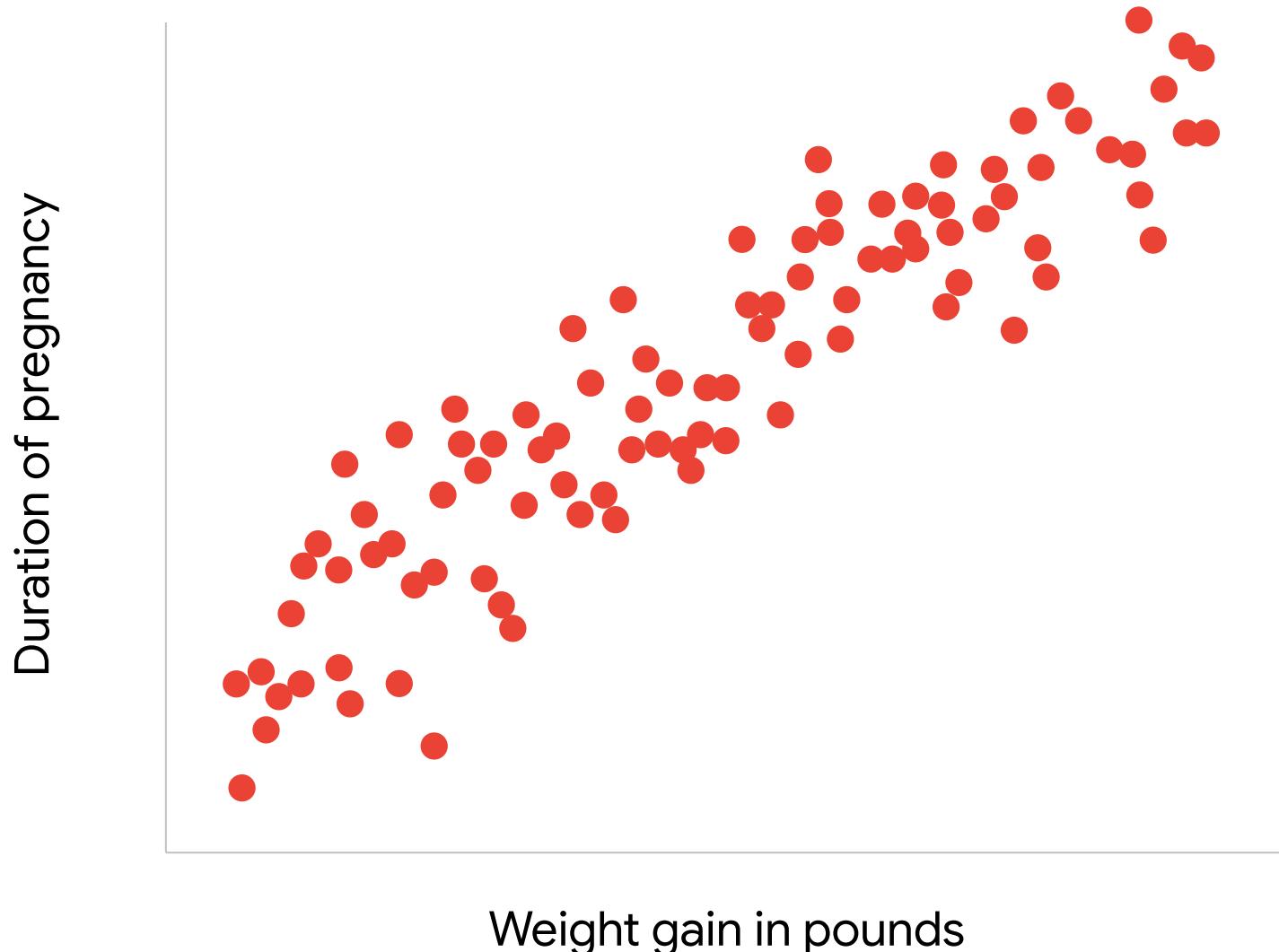
Which model is better?
How can you tell?



Does the model generalize to new data?

Need data that were not
used in training

New data the model hasn't seen before.



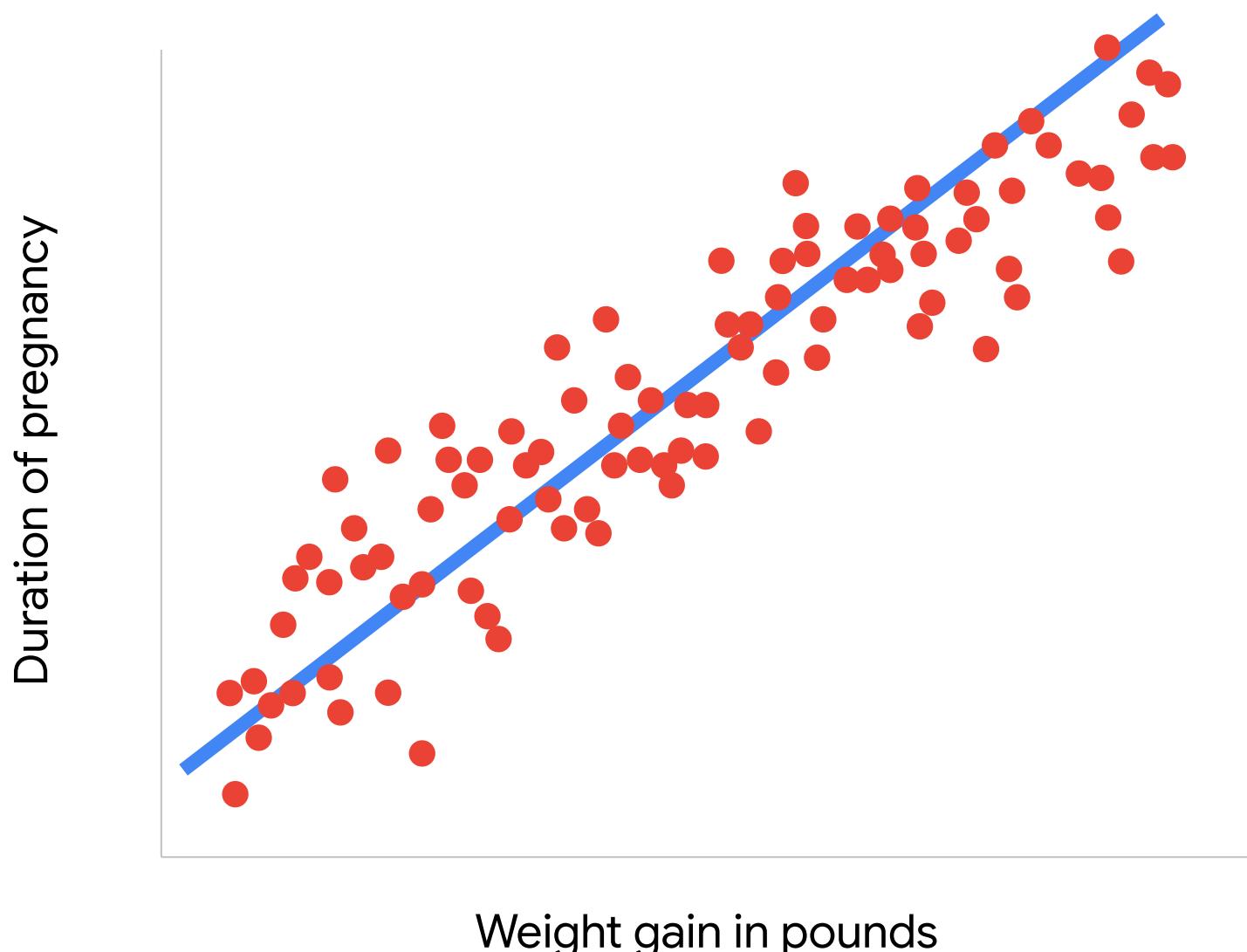
Model 1 generalizes well

Old RMSE = 2.224

New RMSE = 2.198

Pretty similar = good

New data the model hasn't seen before.



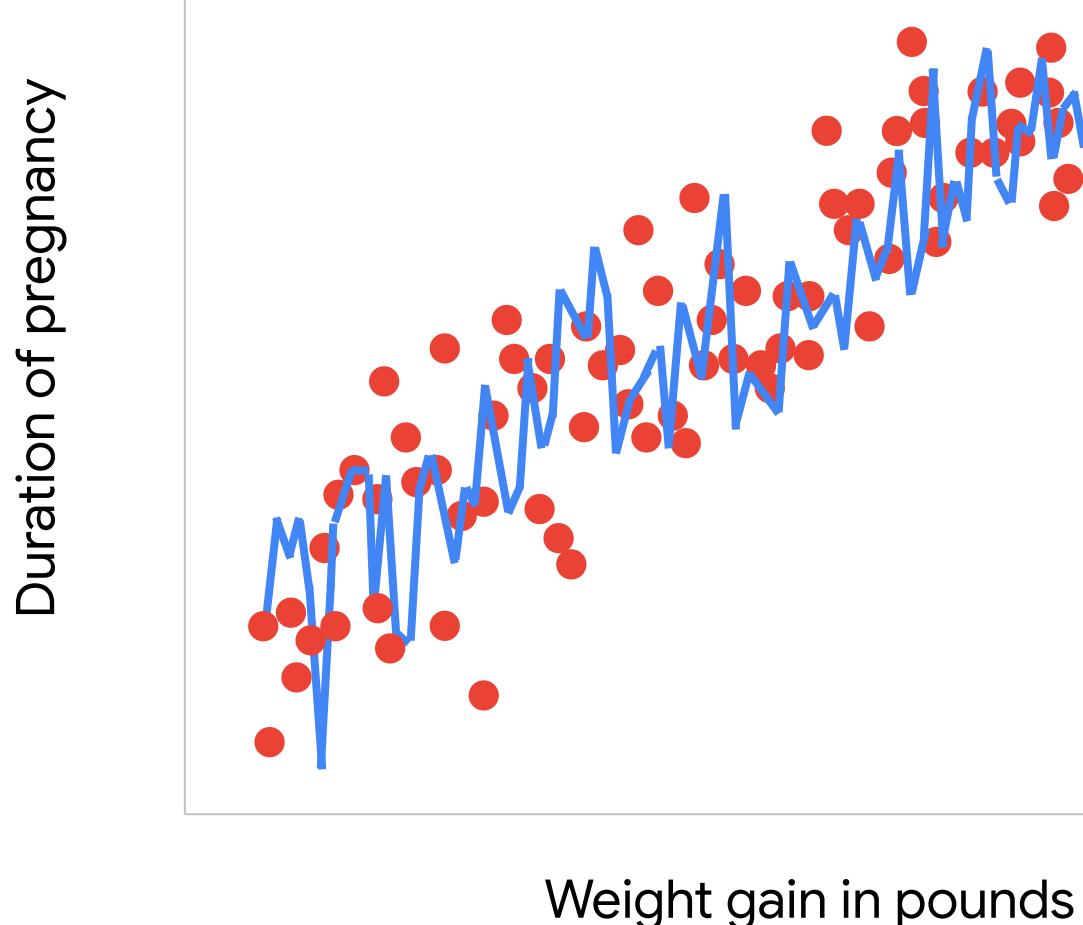
**Model 2 does not
generalize well**

Old RMSE = 0

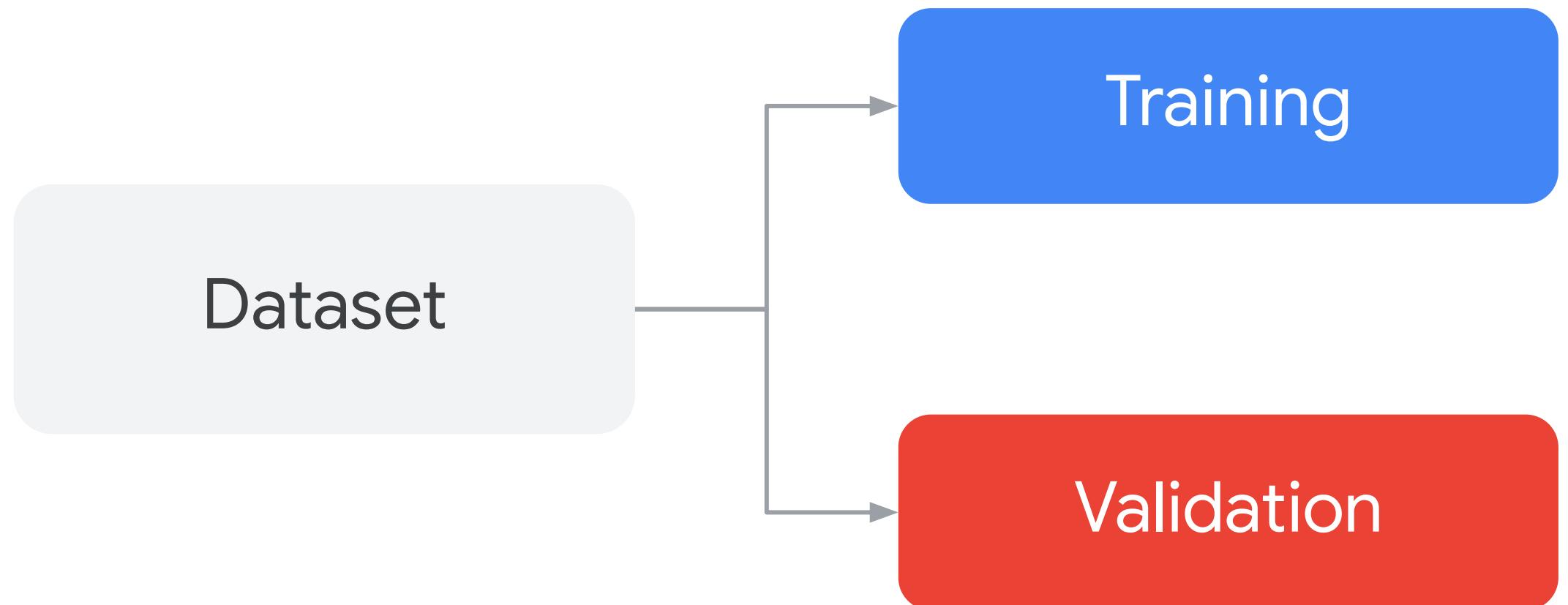
New RMSE = 3.2

This is a red flag.

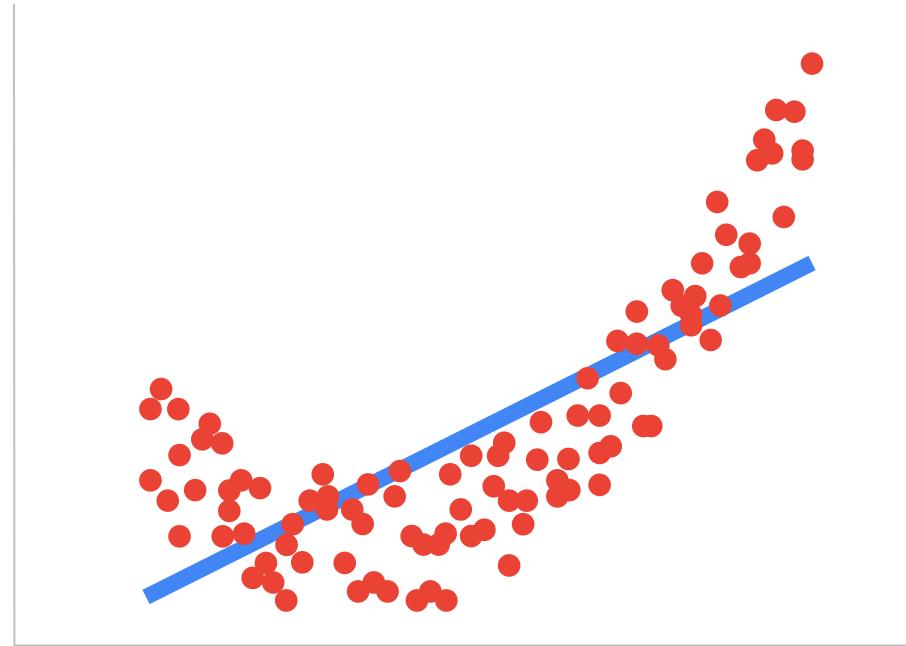
New data the model hasn't seen before.



**Split the dataset
and experiment
with **models****

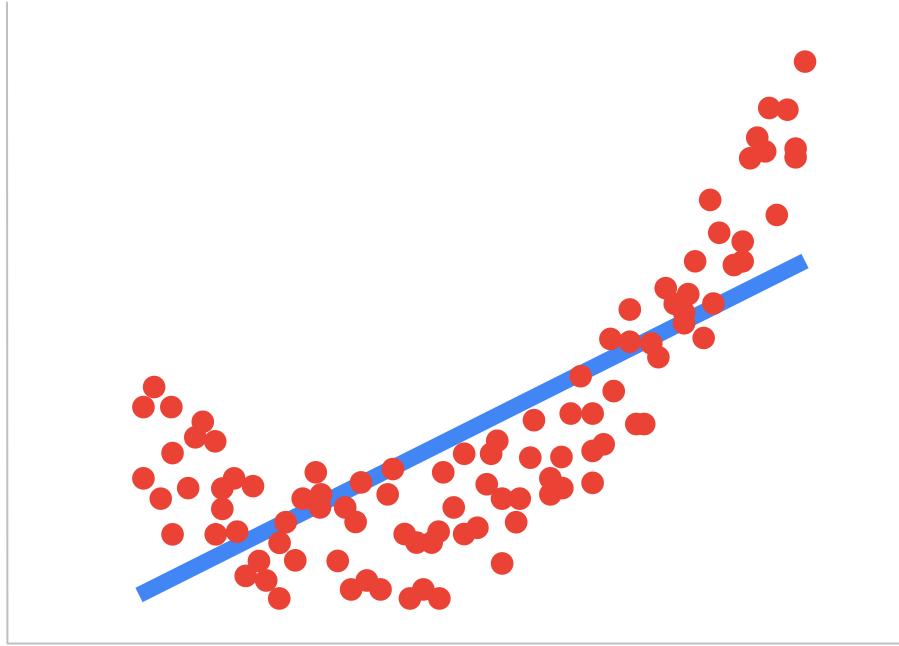


Beware of overfitting as you increase model complexity

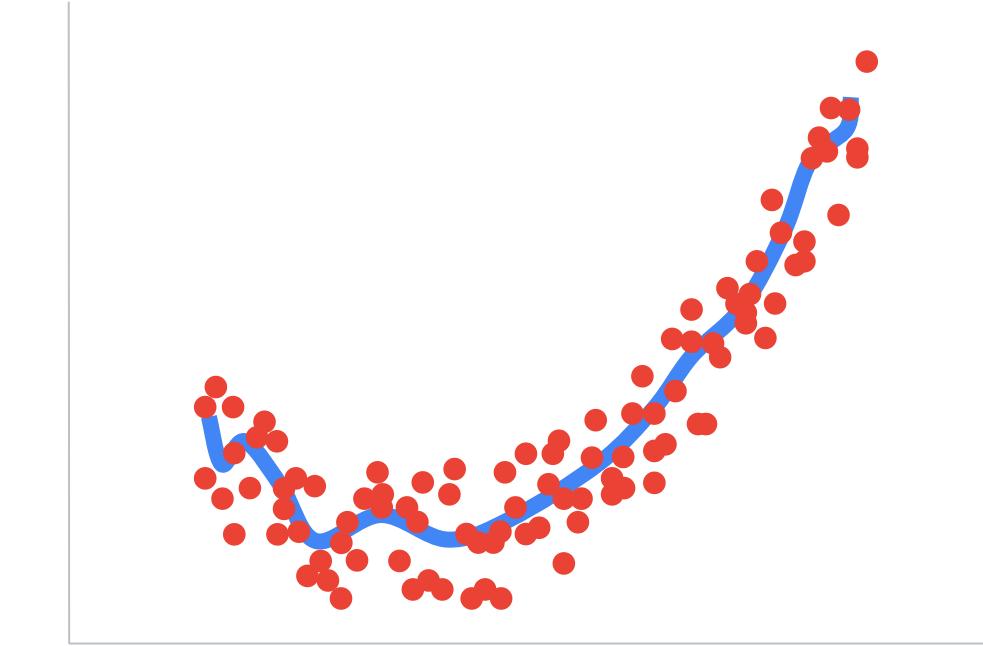


Underfit

Beware of overfitting as you increase model complexity

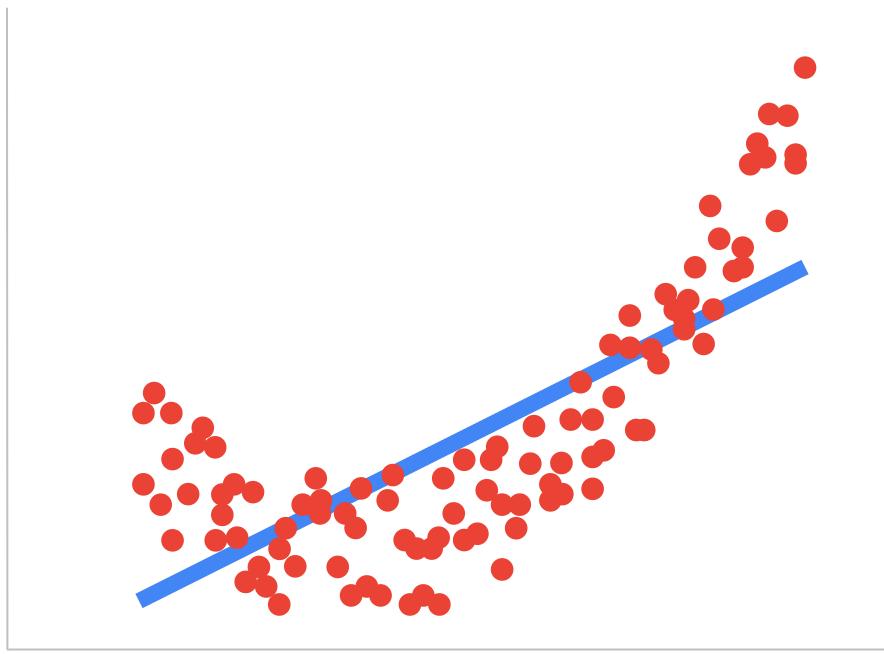


Underfit

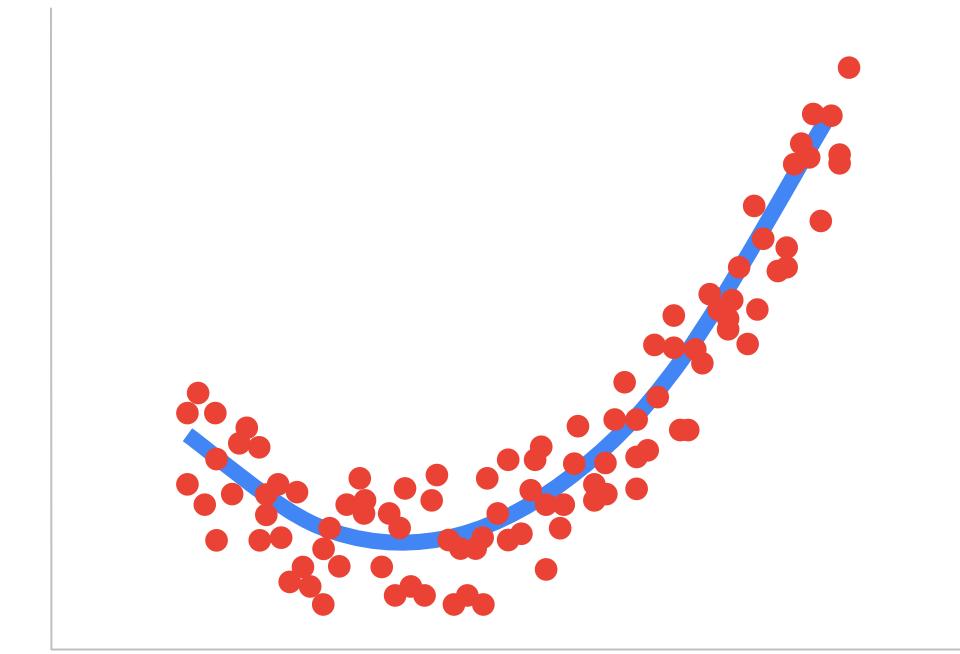


Overfit

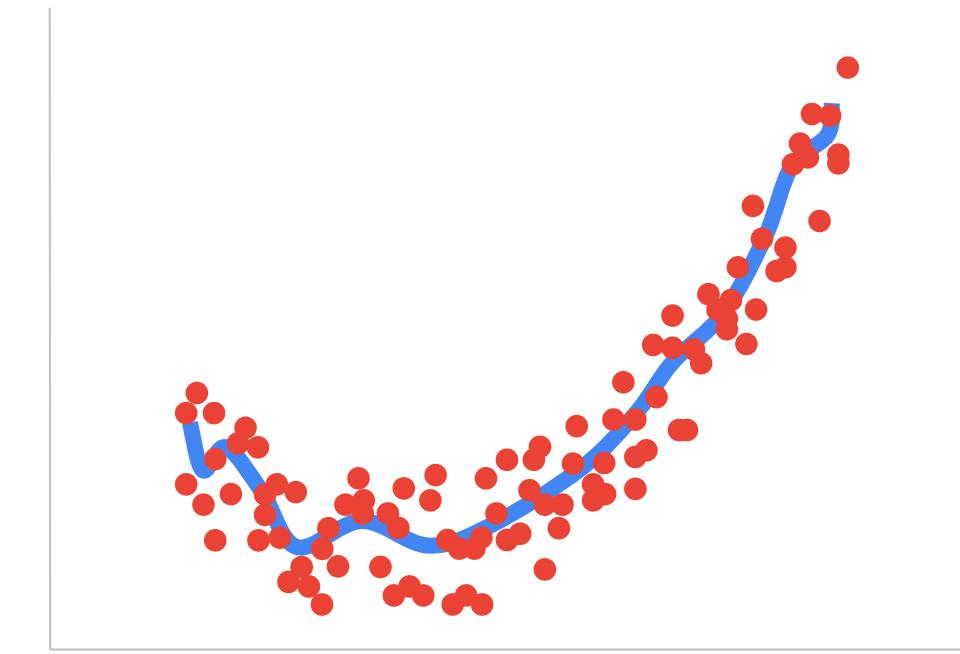
Beware of overfitting as you increase model complexity



Underfit

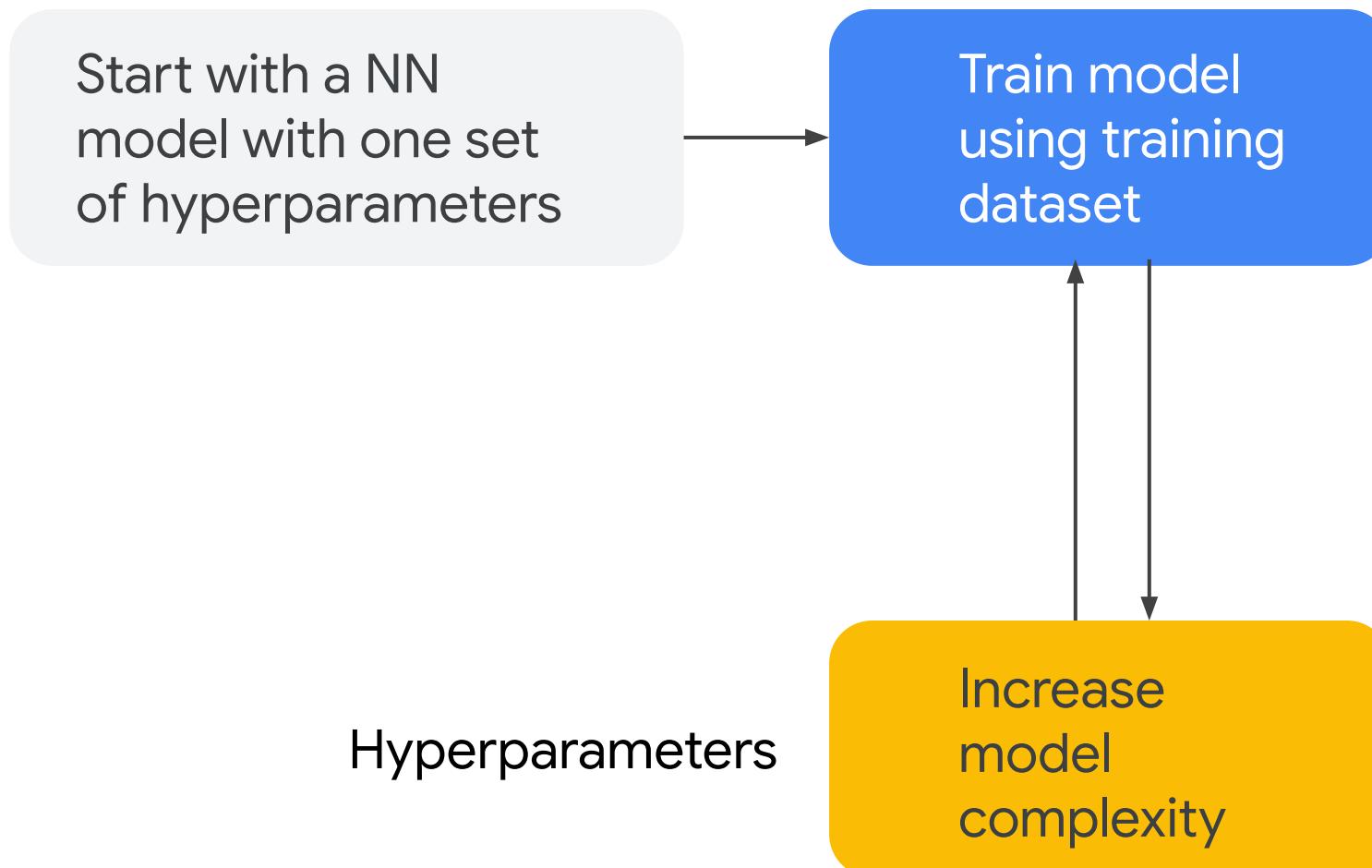


Fit

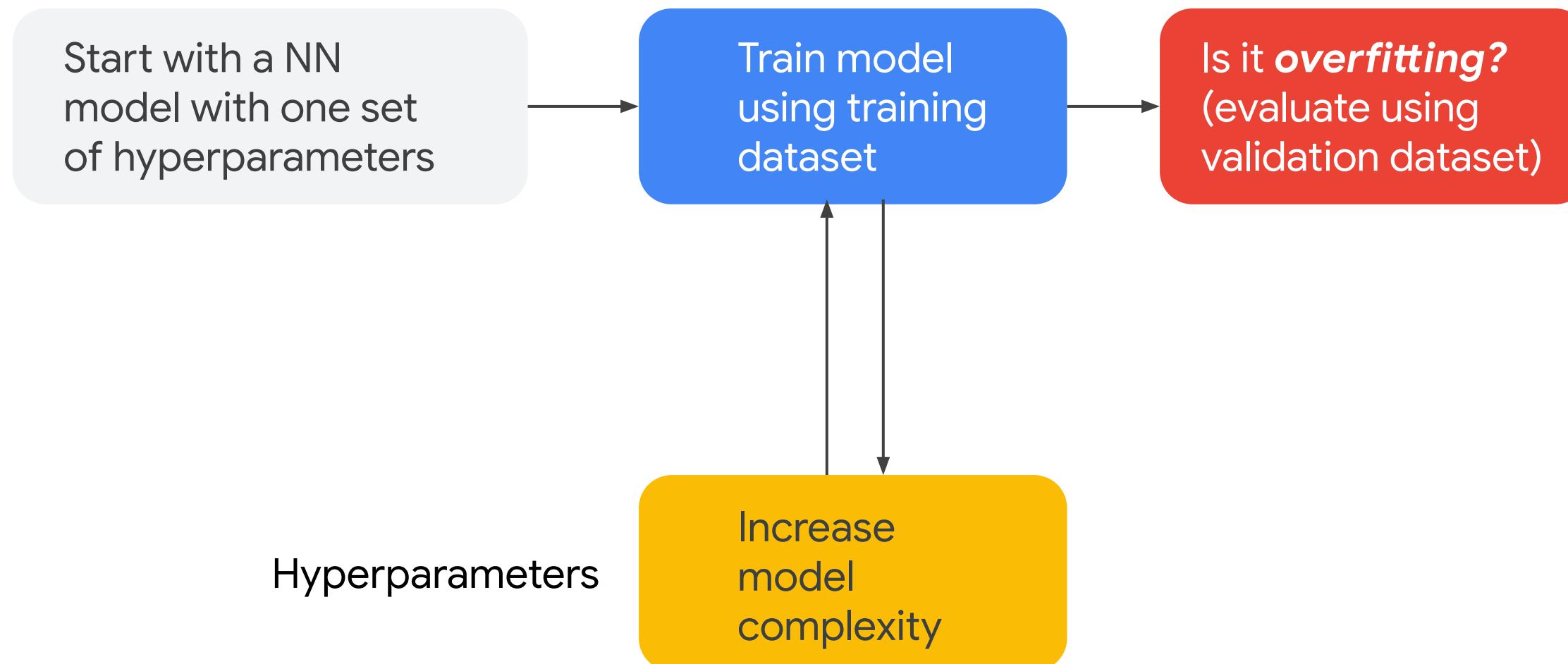


Overfit

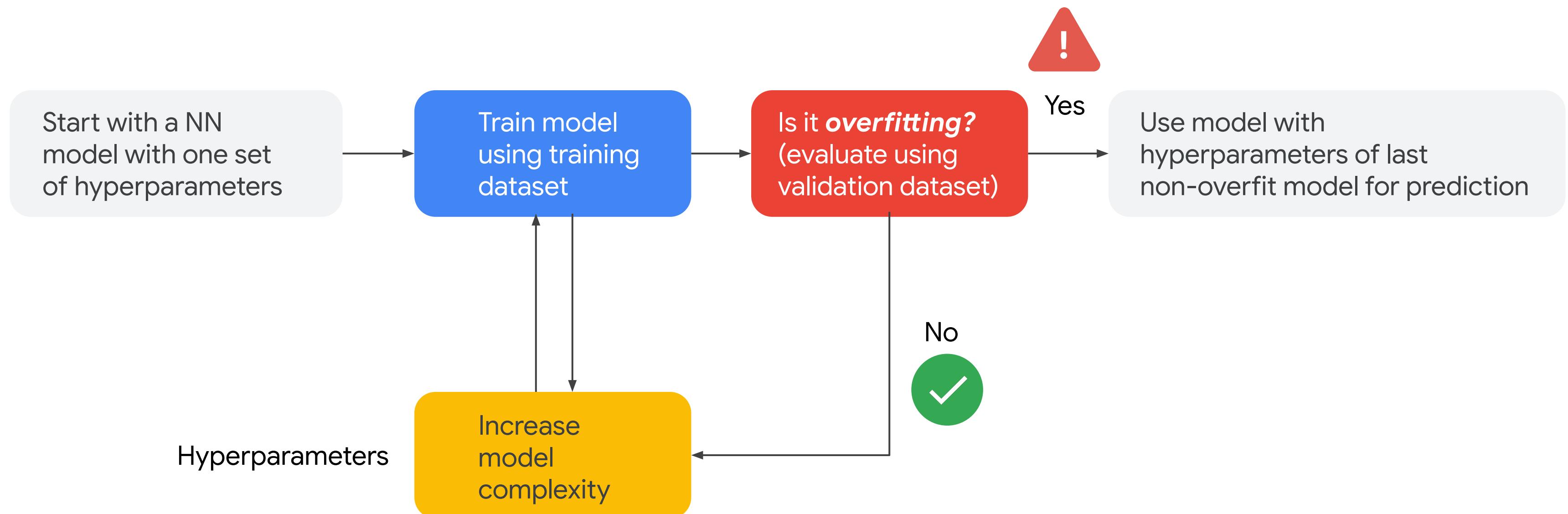
You can use the validation dataset to experiment with model complexity



You can use the validation dataset to experiment with model complexity

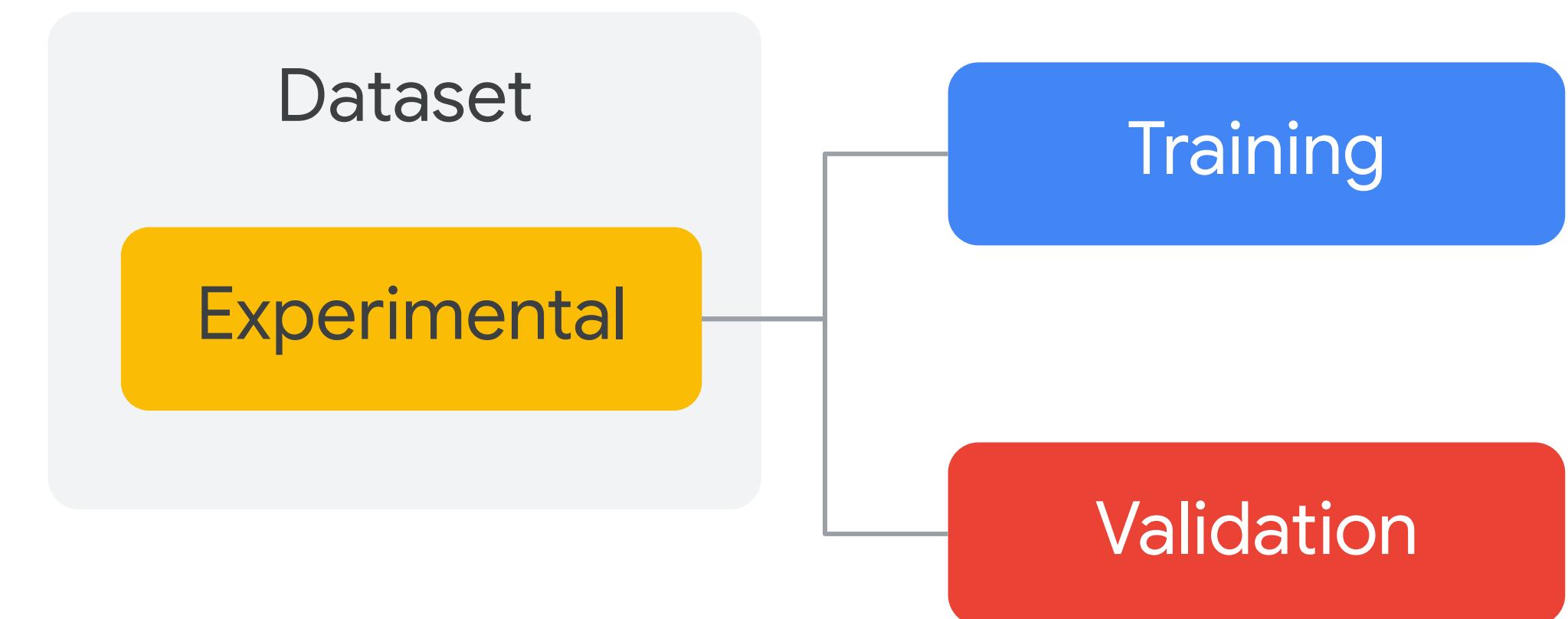


You can use the validation dataset to experiment with model complexity

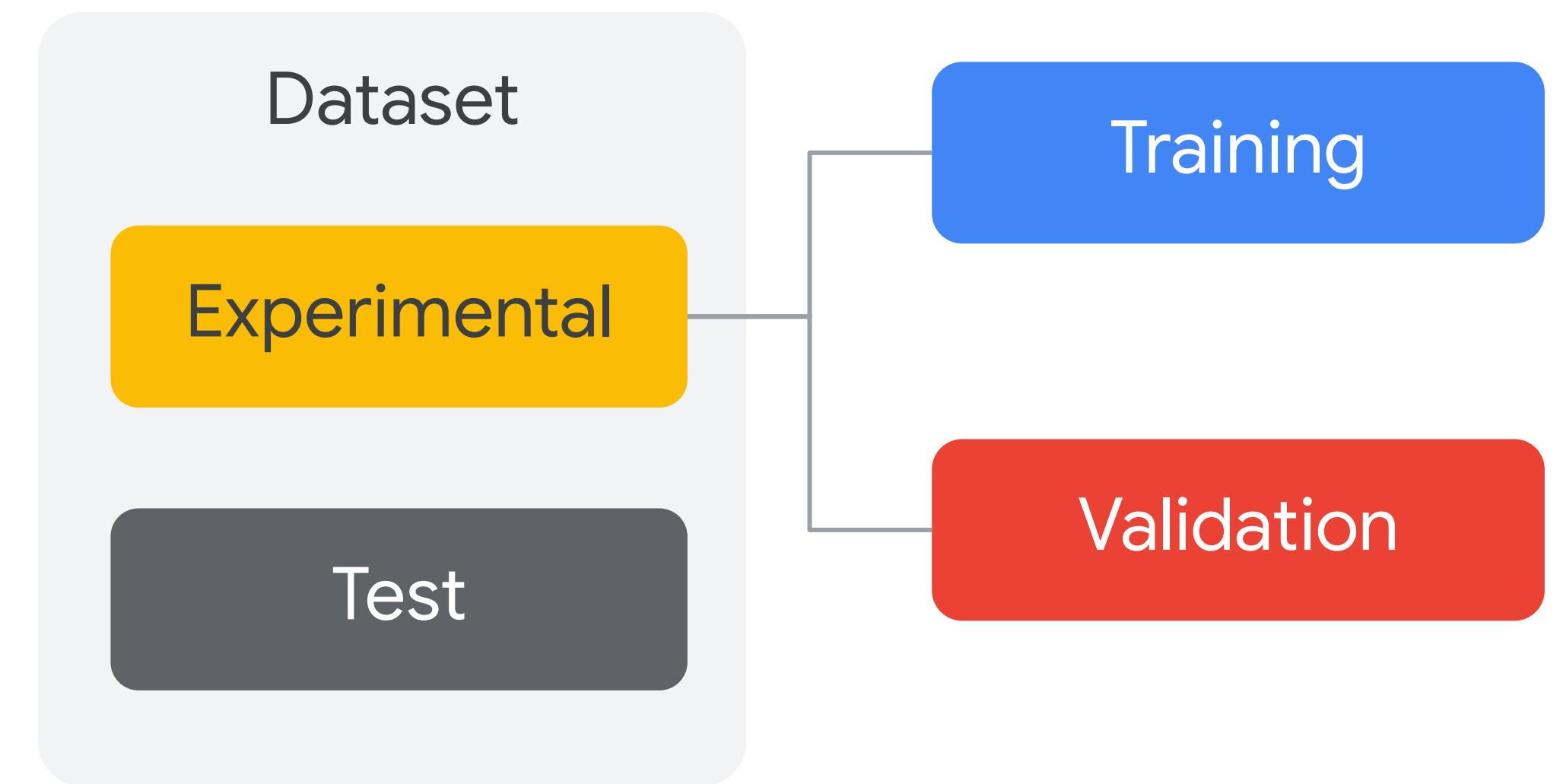


Vertex AI

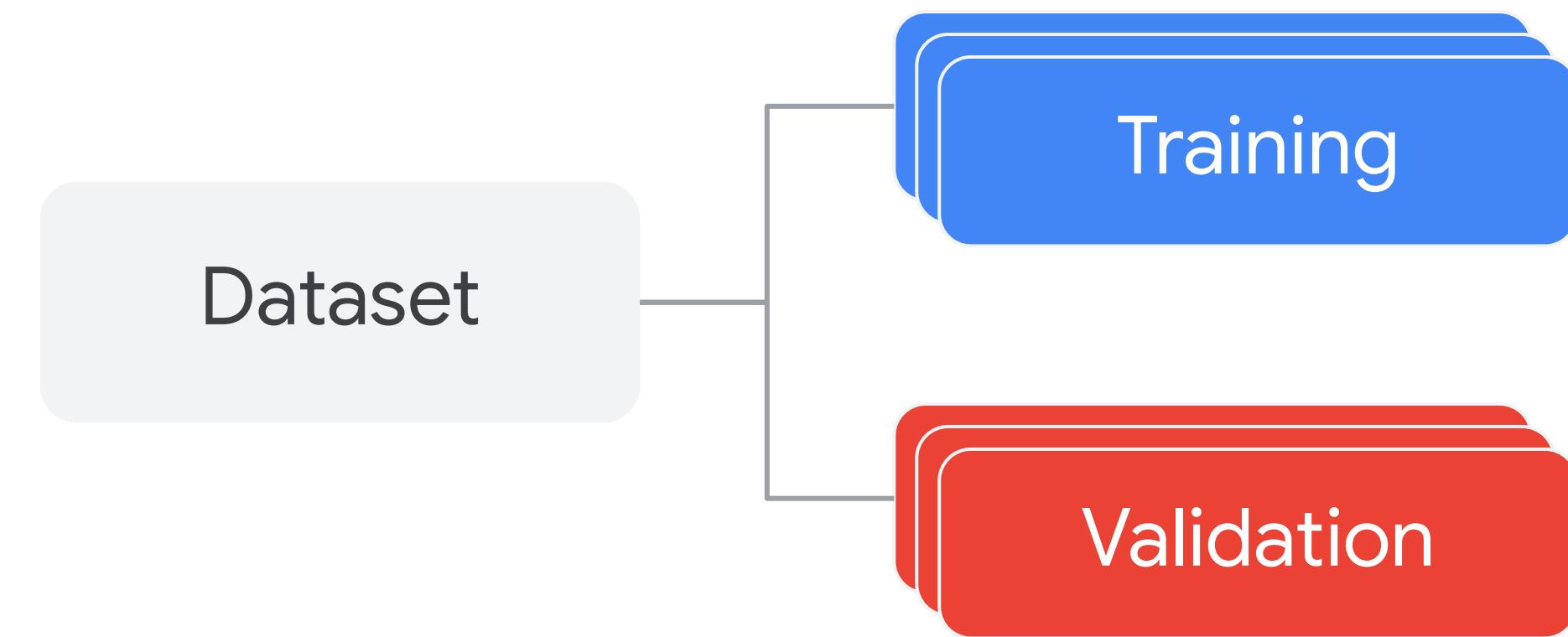




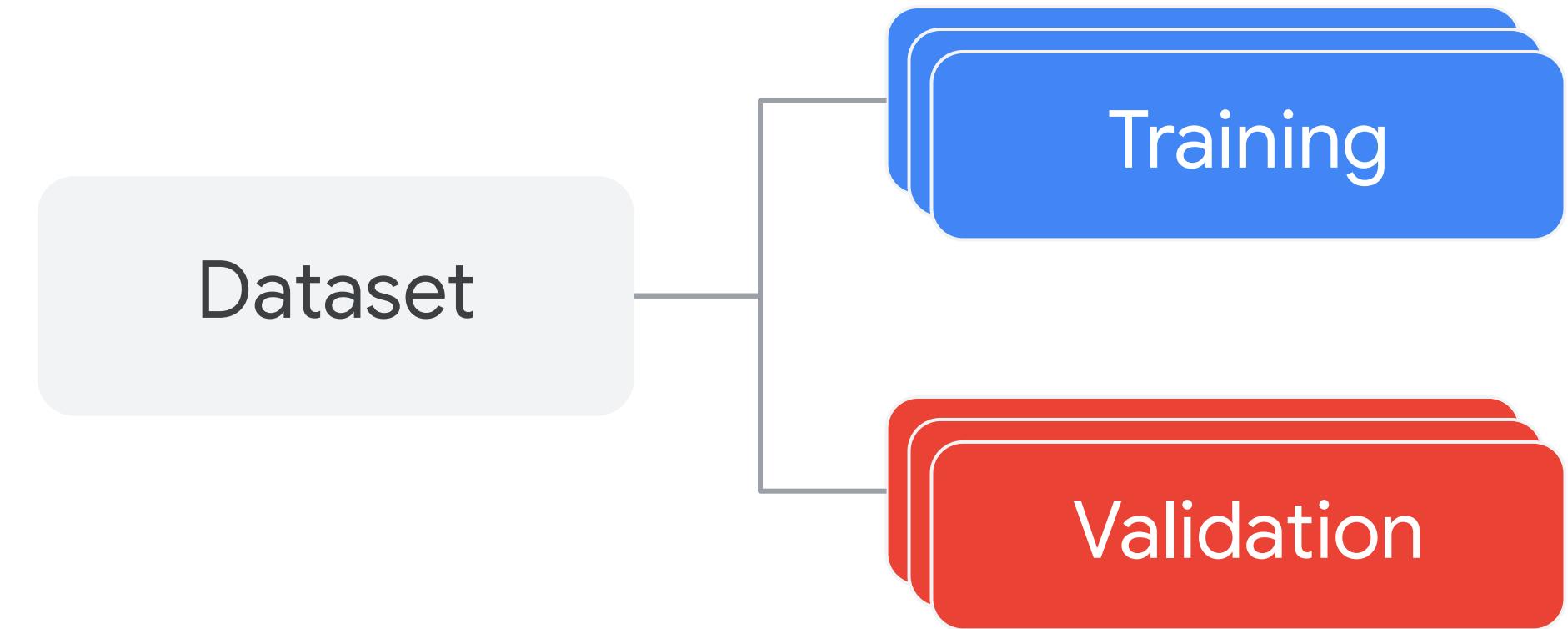
**Evaluate the final
model with
independent
test data**



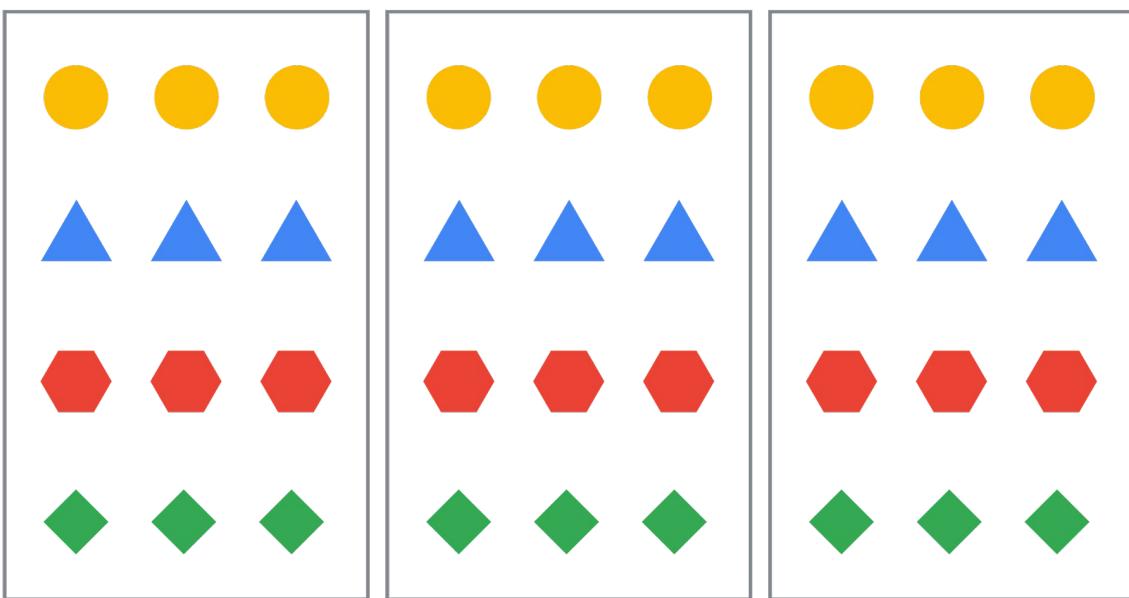
No one likes to **waste** data

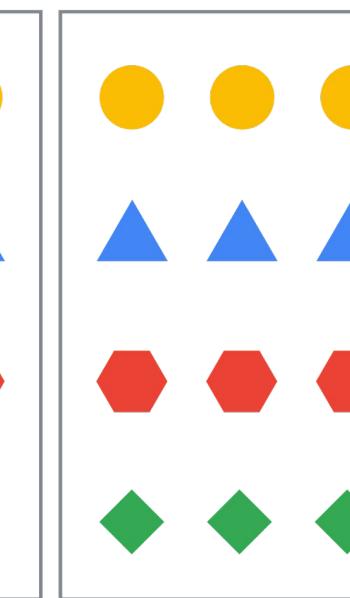
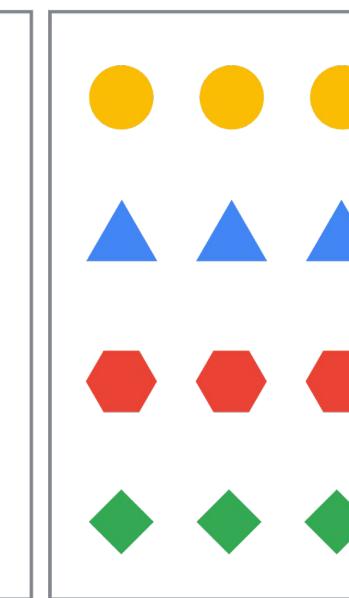
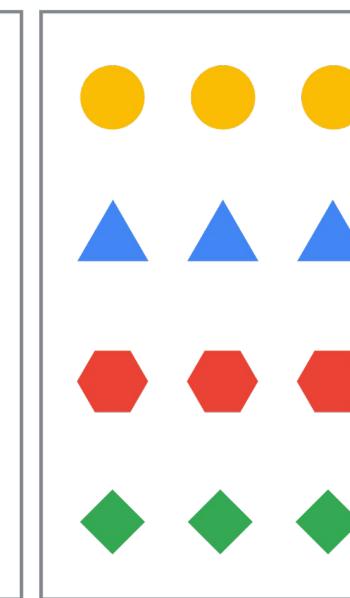
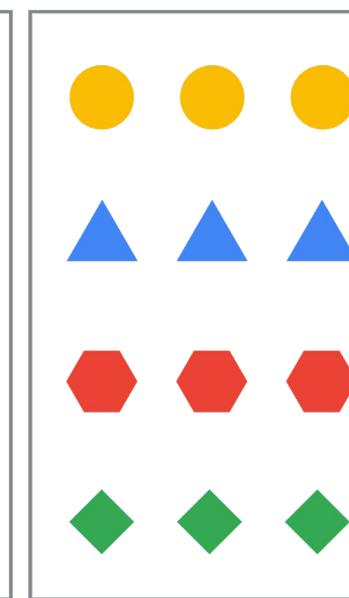
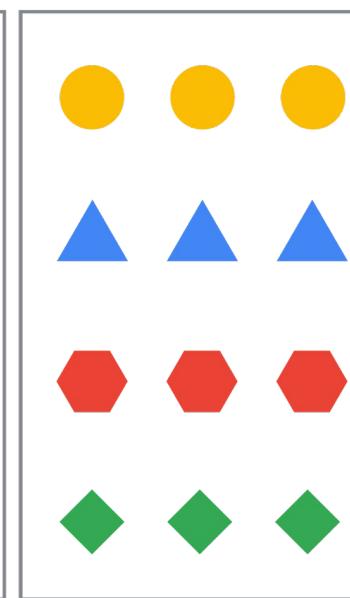
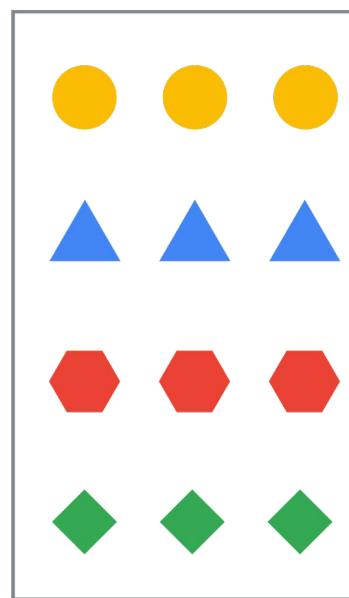
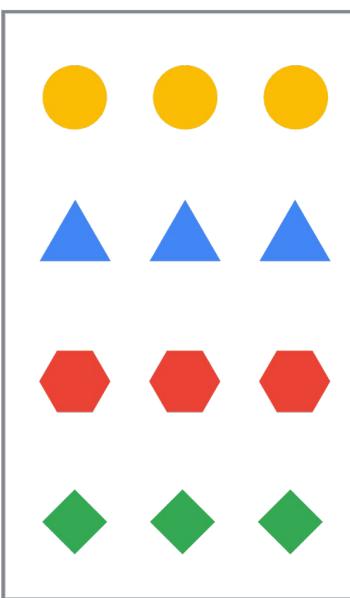


**Evaluate the final
model with
cross-validation**



Splitting your dataset allows for testing your model against simulated real-world data by holding out subsets of data from training.







We often have large datasets in BigQuery that we want to use for machine learning

Row	date	airline	departure_airport	departure_schedule	arrival_airport	arrival_delay
1	2019-08-07	TZ	SRQ	1255	IND	-14.0
2	2019-03-05	TZ	SRQ	2117	IND	-9.0
3	2020-04-12	TZ	SRQ	2000	IND	-17.0
4	2021-04-16	TZ	SRQ	1215	IND	-5.0
5	2021-03-20	TZ	SRQ	645	IND	14.0
6	2020-04-06	TZ	SRQ	1235	IND	-8.0

We often have large datasets in BigQuery that we want to use for machine learning

Dataset

Row	date	airline	departure_airport	departure_schedule	arrival_airport	arrival_delay
1	2019-08-07	TZ	SRQ	1255	IND	-14.0
2	2019-03-05	TZ	SRQ	2117	IND	-9.0
3	2020-04-12	TZ	SRQ	2000	IND	-17.0
4	2021-04-16	TZ	SRQ	1215	IND	-5.0
5	2021-03-20	TZ	SRQ	645	IND	14.0
6	2020-04-06	TZ	SRQ	1235	IND	-8.0

We often have large datasets in BigQuery that we want to use for machine learning

Dataset

Training

Row	date	airline	departure_airport	departure_schedule	arrival_airport	arrival_delay
1	2019-08-07	TZ	SRQ	1255	IND	-14.0
2	2019-03-05	TZ	SRQ	2117	IND	-9.0
3	2020-04-12	TZ	SRQ	2000	IND	-17.0
4	2021-04-16	TZ	SRQ	1215	IND	-5.0
5	2021-03-20	TZ	SRQ	645	IND	14.0
6	2020-04-06	TZ	SRQ	1235	IND	-8.0

We often have large datasets in BigQuery that we want to use for machine learning

Dataset

Training

Validation

Row	date	airline	departure_airport	departure_schedule	arrival_airport	arrival_delay
1	2019-08-07	TZ	SRQ	1255	IND	-14.0
2	2019-03-05	TZ	SRQ	2117	IND	-9.0
3	2020-04-12	TZ	SRQ	2000	IND	-17.0
4	2021-04-16	TZ	SRQ	1215	IND	-5.0
5	2021-03-20	TZ	SRQ	645	IND	14.0
6	2020-04-06	TZ	SRQ	1235	IND	-8.0

We often have large datasets in BigQuery that we want to use for machine learning

Dataset

Training

Validation

Test

Row	date	airline	departure_airport	departure_schedule	arrival_airport	arrival_delay
1	2019-08-07	TZ	SRQ	1255	IND	-14.0
2	2019-03-05	TZ	SRQ	2117	IND	-9.0
3	2020-04-12	TZ	SRQ	2000	IND	-17.0
4	2021-04-16	TZ	SRQ	1215	IND	-5.0
5	2021-03-20	TZ	SRQ	645	IND	14.0
6	2020-04-06	TZ	SRQ	1235	IND	-8.0

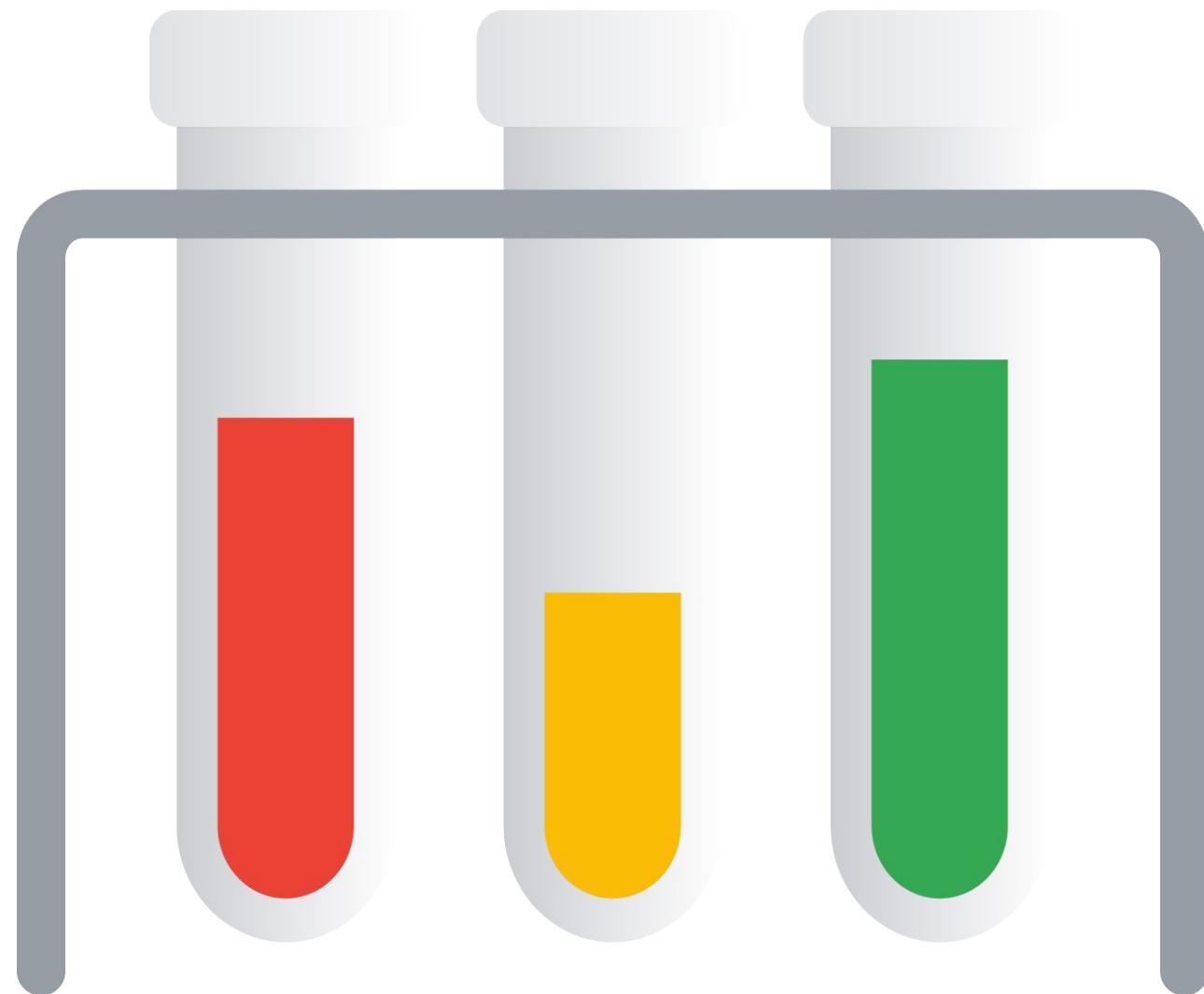
It's easy to get a random 80% of your dataset for training

```
#standardSQL
SELECT
    date,
    airline,
    departure_airport,
    departure_schedule,
    arrival_airport,
    arrival_delay
FROM
    `bigquery-samples.airline_ontime_data.flights`
WHERE
    RAND() < 0.8
```

RAND will return a number between 0 and 1.

**However,
experimentation
requires **repeatability****

You need to know which specific data was involved in training, validation, and testing.



Naive random splitting is not repeatable

Order of rows in BigQuery is not certain without ORDER BY.

Hard to identify and split the remaining 20% of data for validation and testing.

RAND() will return different results each time

The screenshot shows the BigQuery Query Editor interface. The top section is a code editor with the following SQL query:

```
1 #standardSQL
2 SELECT
3   date,
4   airline,
5   departure_airport,
6   departure_schedule,
7   arrival_airport,
8   arrival_delay
9 FROM
10   `bigquery-samples.airline_ontime_data.flights`
11 WHERE
12   RAND() < 0.8 # returns different records each time
13 LIMIT 5;
```

The bottom section shows the results of the query, which are five rows of flight data. The columns are: Row, date, airline, departure_airport, departure_schedule, arrival_airport, and arrival_delay. The results are as follows:

Row	date	airline	departure_airport	departure_schedule	arrival_airport	arrival_delay	
1	2005-07-07	NW	DTW		700	MSP	-9.0
2	2005-07-04	NW	DTW		700	MSP	-9.0
3	2005-07-06	NW	DTW		700	MSP	19.0
4	2005-07-08	NW	DTW		700	MSP	-19.0
5	2005-07-05	NW	DTW		700	MSP	36.0

Solution: Split a dataset into training/validation/test using the hashing and modulo operators

```
#standardSQL
SELECT
  date,
  airline,
  departure_airport,
  departure_schedule,
  arrival_airport,
  arrival_delay
FROM
  `bigquery-samples.airline_ontime_data.flights`
WHERE
  MOD(ABS(FARM_FINGERPRINT(date)),10) < 8
```

Note:

Even though we select date, our model wouldn't actually use it during training.

Hash value on the Date will always return the same value.

Then we can use a modulo operator to only pull 80% of that data based on the last few hash digits.

Carefully choose which field will split your data



We hypothesize that flight delay depends on the carrier, time of day, weather, and airport characteristics (# of runways, etc.) We want to predict flight delays.

Carefully choose which field will split your data



We hypothesize that flight delay depends on the carrier, time of day, weather, and airport characteristics (# of runways, etc.) We want to predict flight delays.

What field should we split our data on?

- Hash on date?
- Hash on airport?
- Hash on carrier name?

Carefully choose which field will split your data



We hypothesize that flight delay depends on the carrier, time of day, weather, and airport characteristics (# of runways, etc.) We want to predict flight delays.

What field should we split our data on?

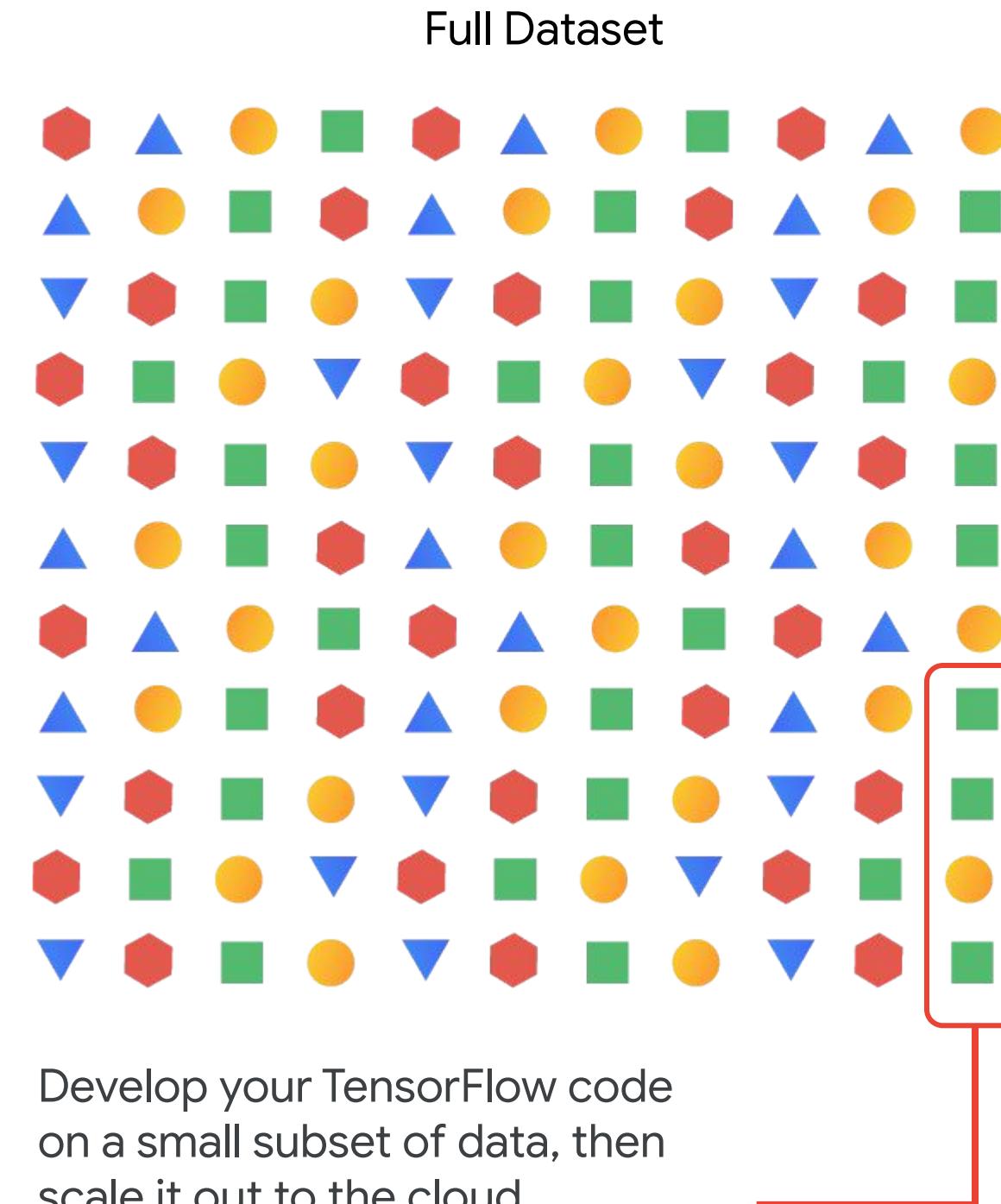
- Hash on date?
- Hash on airport?
- Hash on carrier name?

Split your data on a field you can afford to lose.

What if we Hash and split on **date**?

What if we Hash and split on **airport name**?

**Developing the ML model
software on the entire
dataset can be expensive;
you want to develop on
a smaller sample**



Pitfall: Chaining hashes to create subsets won't work

```
#standardSQL
SELECT
    date,
    airline,
    departure_airport,
    departure_schedule,
    arrival_airport,
    arrival_delay
FROM
    `bigquery-samples.airline_ontime_data.flights`
WHERE
    MOD(ABS(FARM_FINGERPRINT(date)),70) = 0
    AND
        MOD(ABS(FARM_FINGERPRINT(date)),10) < 8
```



Then take 1 in 70 flights.

Take 80% of the dataset? Incorrect!

All records here will also be divisible by 10 (there is no new filtering happening!)

```
#standardSQL
SELECT
    date,
    airline,
    departure_airport,
    departure_schedule,
    arrival_airport,
    arrival_delay
FROM
    `bigquery-samples.airline_ontime_data.flights`
WHERE
    MOD(ABS(FARM_FINGERPRINT(date)),70) = 0
AND
    MOD(ABS(FARM_FINGERPRINT(date)),10) < 8
```

How we want to split our data

