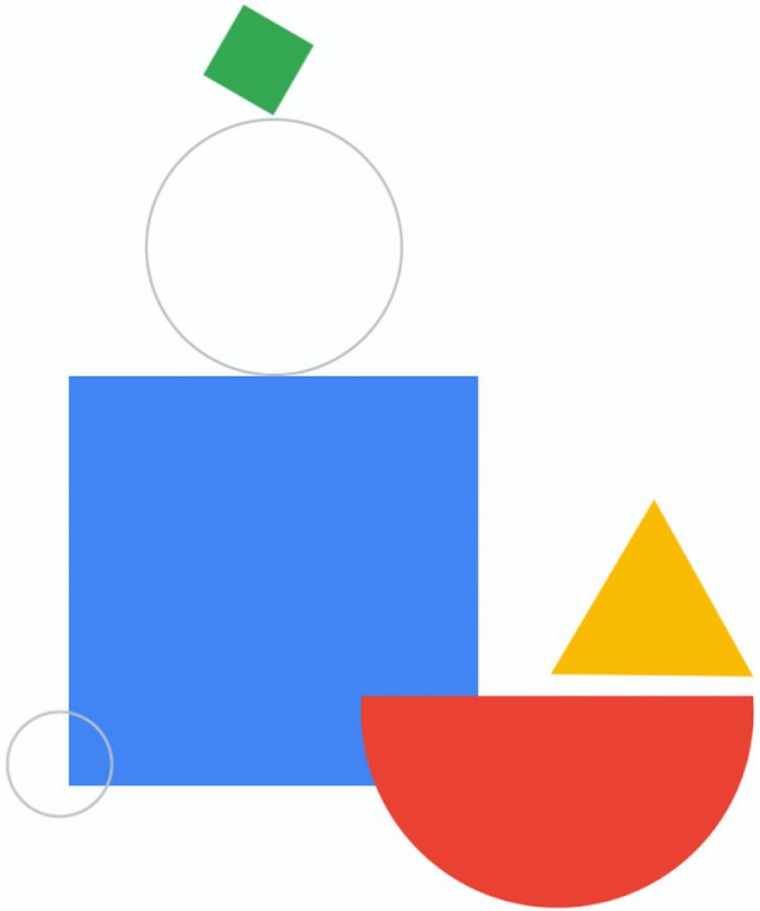
Google Cloud Academy

The Path to Partner Technical Readiness Professional Machine Learning Engineer Certification

Session 2: Welcome to the Professional Machine Learning engineer Certification Program

We will Begin in:

<<15:00->>



Instructor: Ben Ahmed

Instructor: Ben Ahmed

Google Cloud
Authorized Trainer

Working in IT industry for 12 years. A mix of roles, hands on engineering roles in ML (NLP, CNN's, DNN's, VertexAI, Python, TensorFlow, PyTorch, CNN's,) and MLOps solution designing

Years of experience instructing

a Authorized Google Cloud Trainer, I have now been running workshops with Google Customers for over 4 Years for Google. Conducted +25 Workshops, +600 Attendees

Other pertinent tech roles

Machine learning Engineer for Banking (HSBC) & Oil & Gas (BP), and for Startups,

The following course materials are copyright protected materials.

They may not be reproduced or distributed and may only be used by students attending this Google Cloud Partner Learning Services program.

Course etiquette

Please silence your phone and take calls outside.

Recording this class is prohibited.

Ask questions via Chat, If time permits

Week's agenda

Machine Learning Basics: Model Metrics

Part 2

Instructor: Ben Ahmed

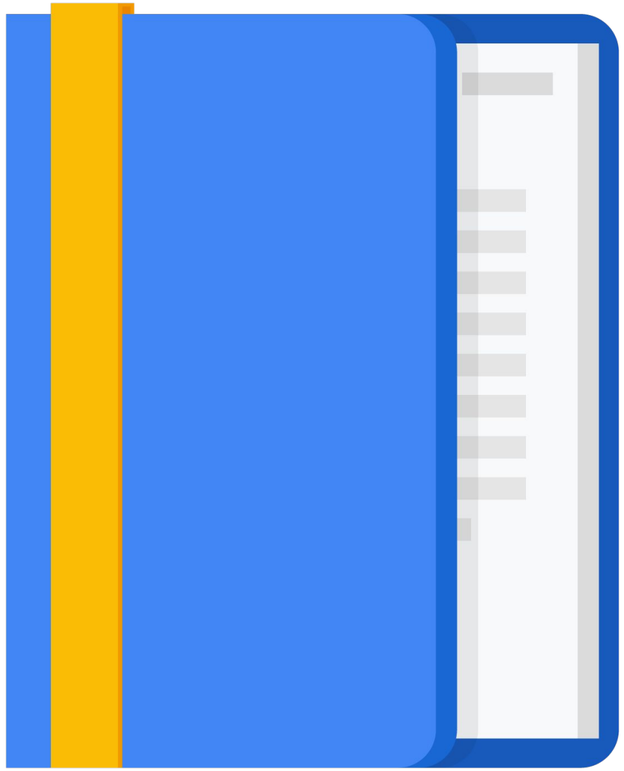
Course Agenda

Generalization in ML

Sampling

Regularizations

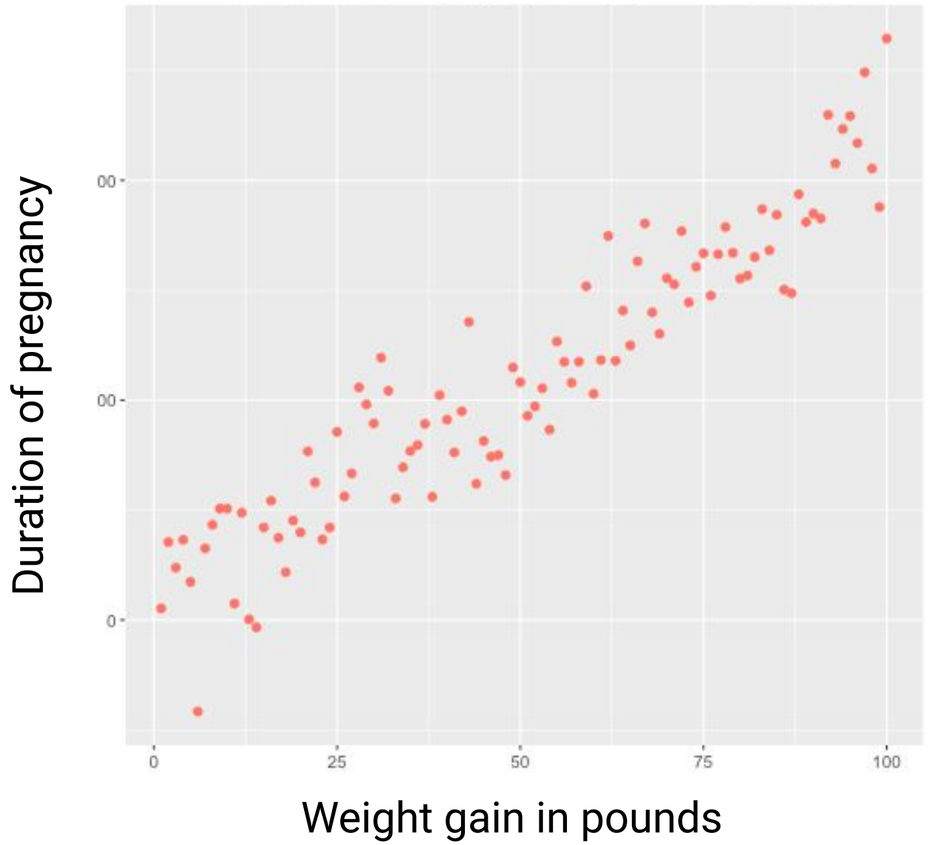
ML Model Performance Evaluation



Needed For the Exam

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

What is the error measure to optimize?



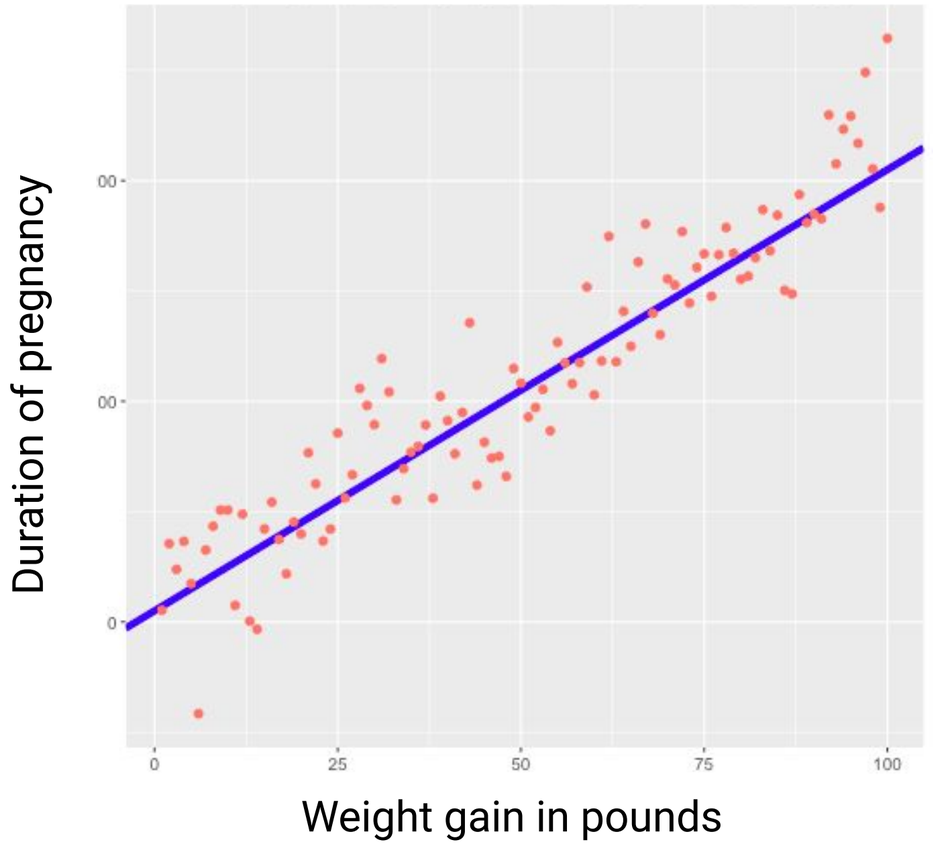
Needed For the Exam

Model 1 is a linear model using linear regression

Red = training examples

Blue = model prediction for each baby

RMSE = 2.224



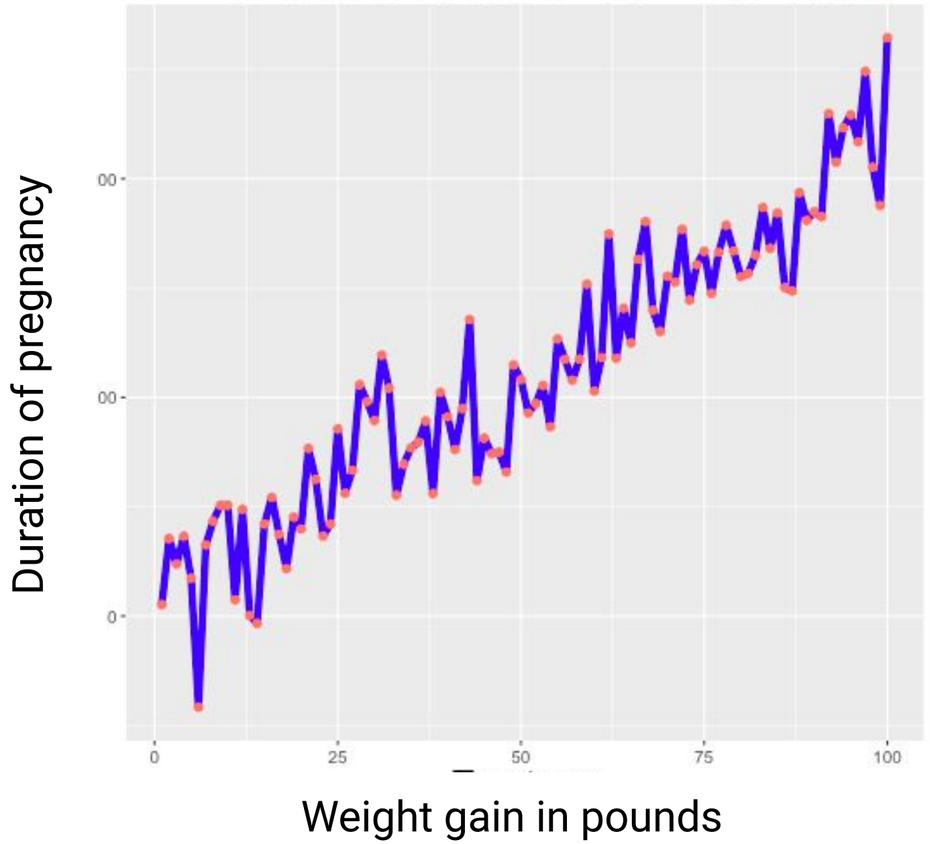
Needed For the Exam

Model 2 has more free parameters

RMSE = 0

Which model is better?

How can you tell?

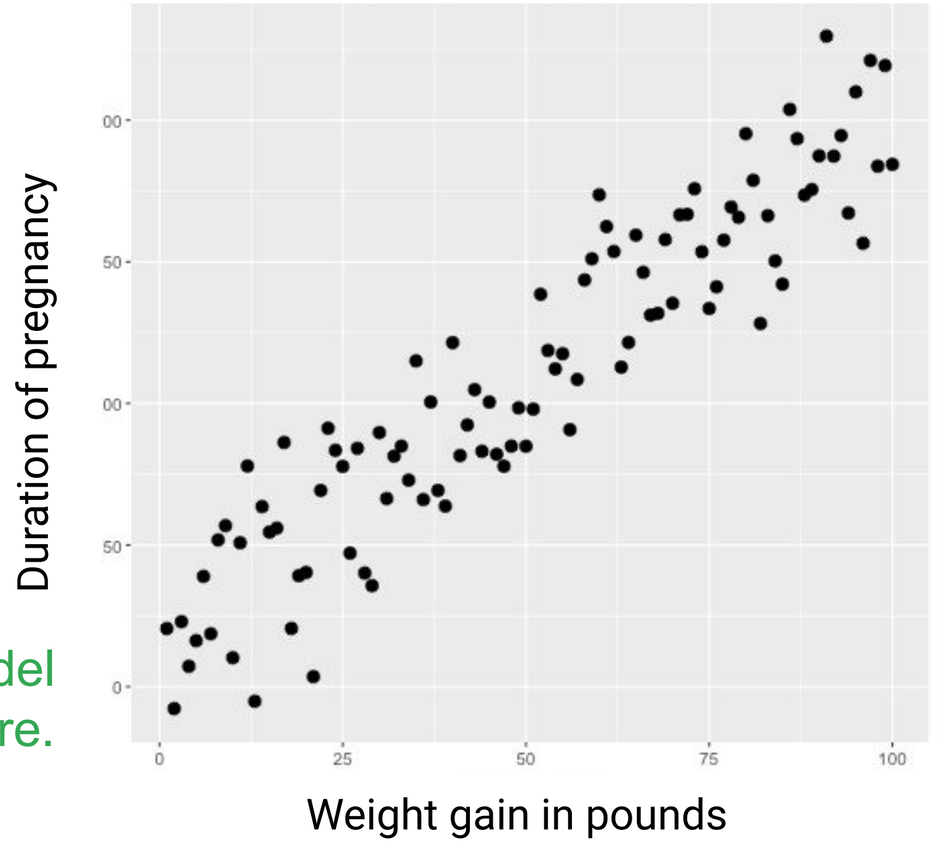


Needed For the Exam

Does the model generalize to new data?

Need data that were not used in training.

New data the model hasn’t seen before.



Needed For the Exam

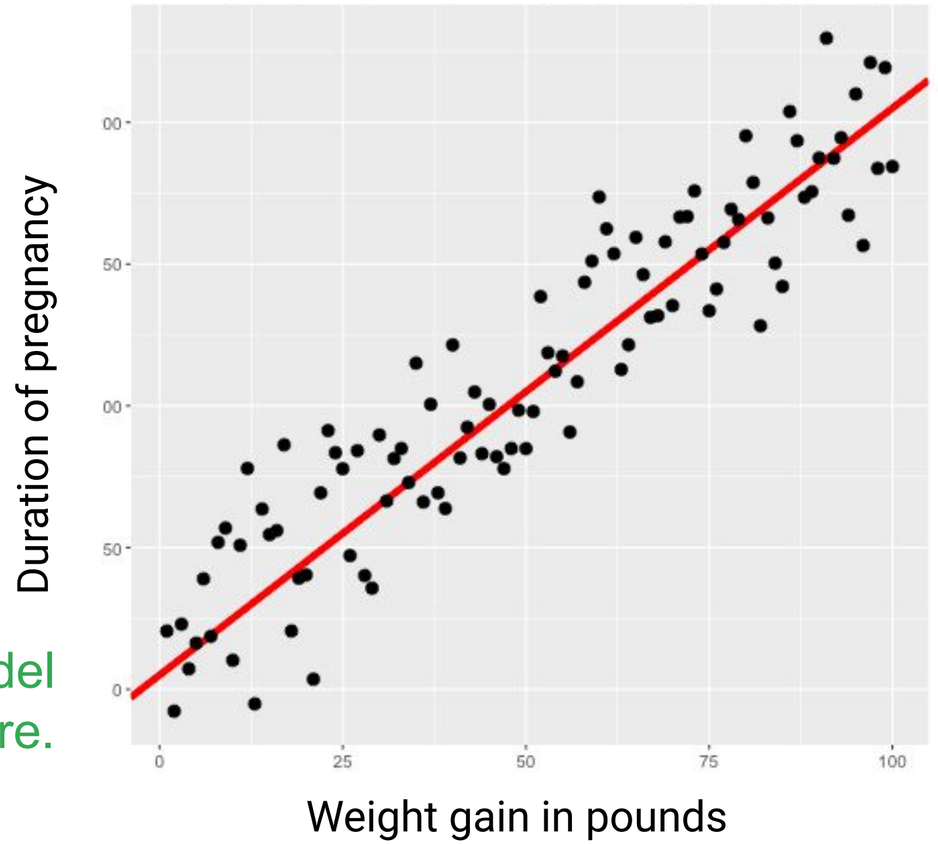
Model 1 generalizes well

Old RMSE = 2.224

New RMSE = 2.198

Pretty similar = good

New data the model hasn't seen before.



Needed For the Exam

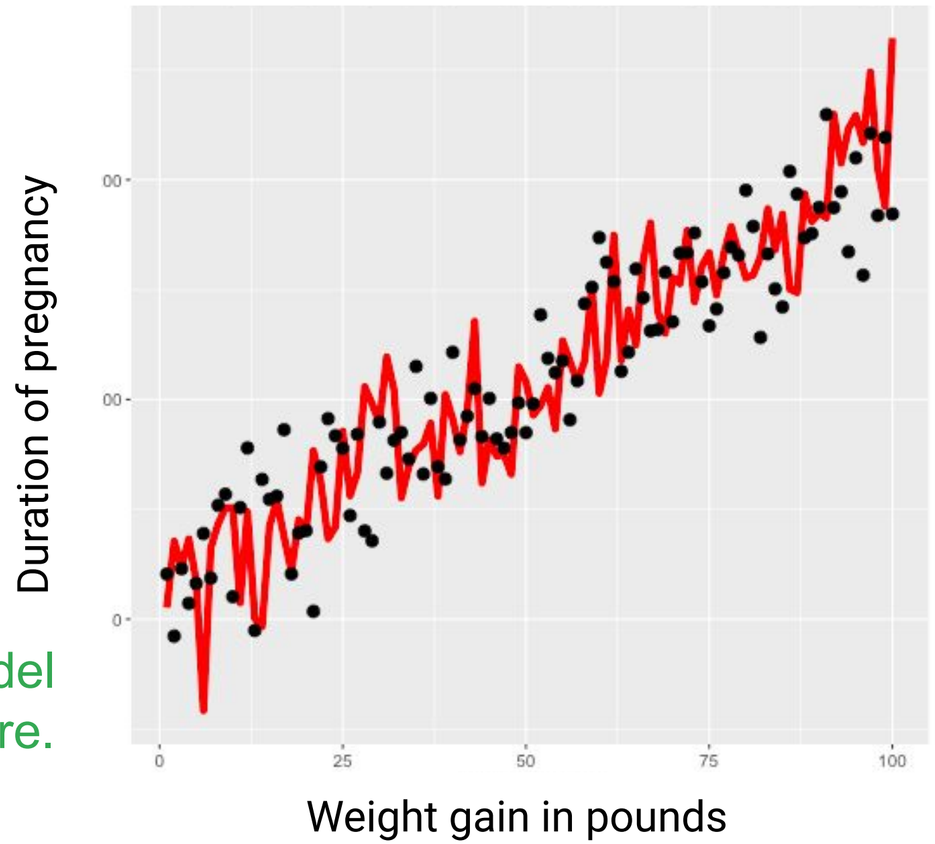
Model 2 does not generalize well

Old RMSE = 0

New RMSE = 3.2

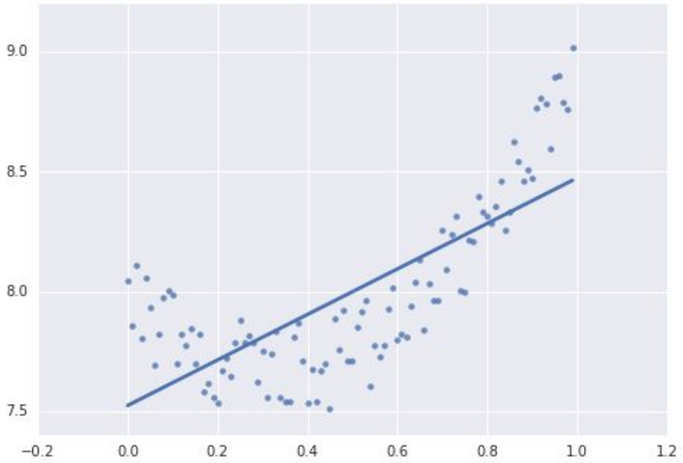
This is a red flag

New data the model hasn't seen before.



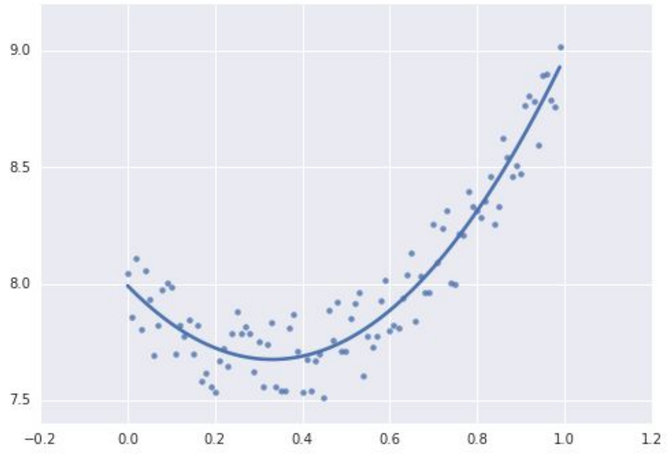
Needed For the Exam

Beware of overfitting as you increase model complexity

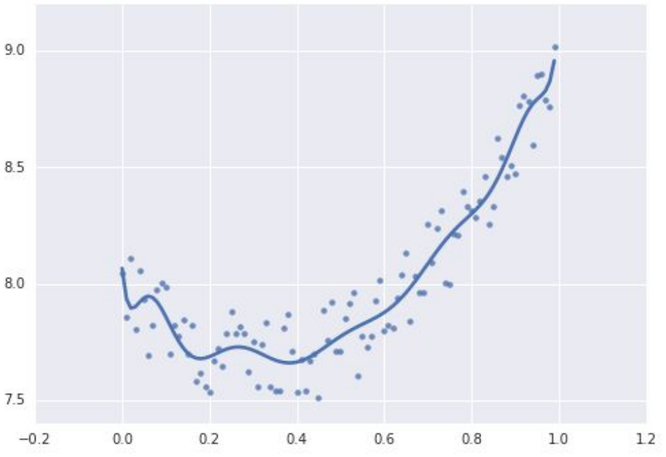


Underfit

High Bias



Fit

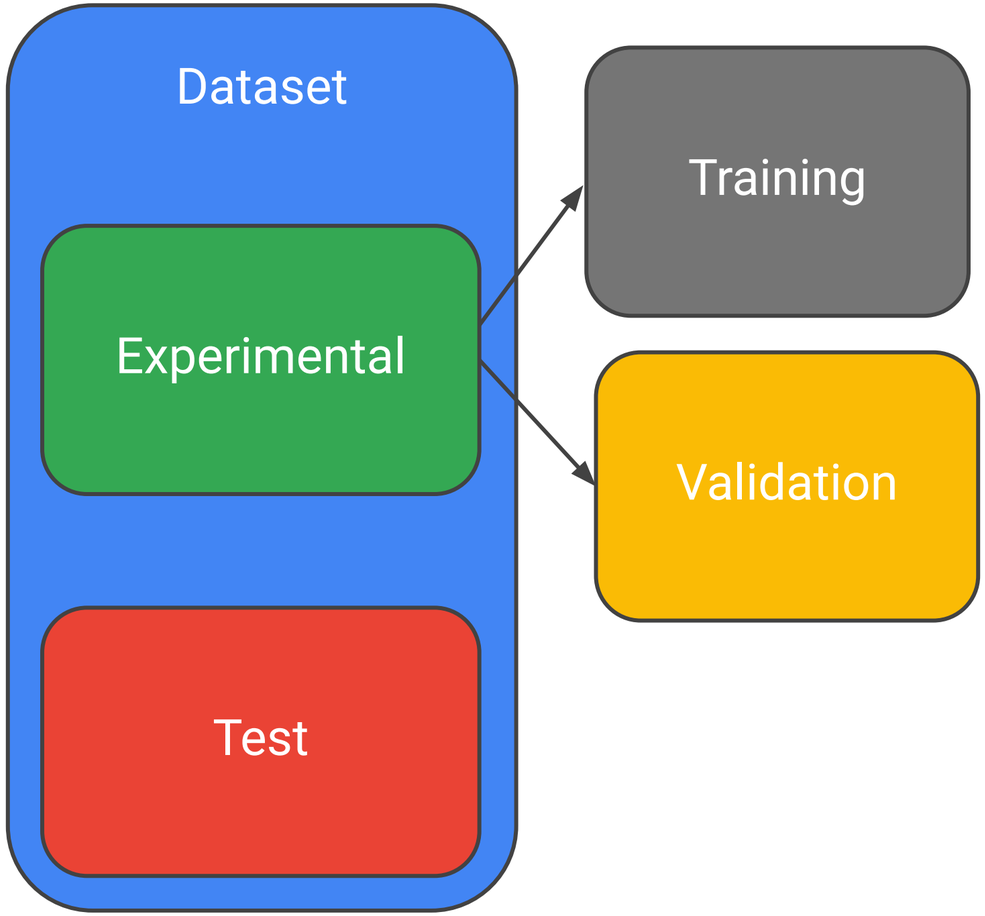


Overfit

High Variance

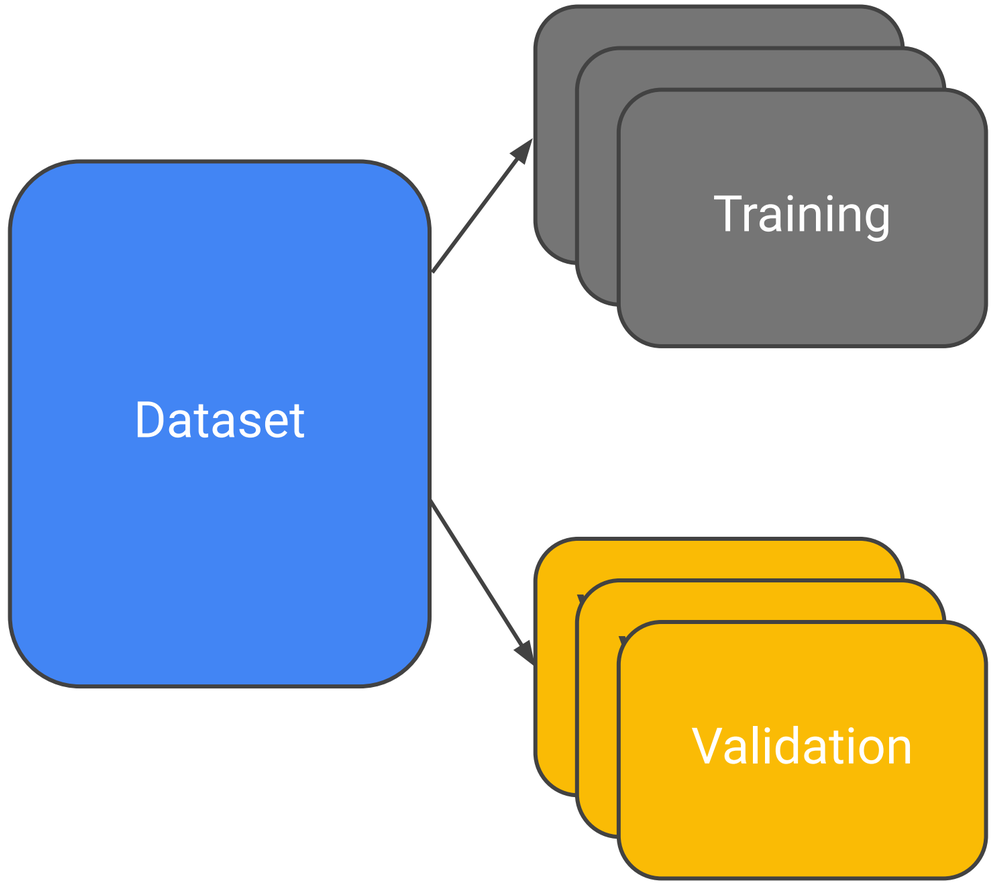
Needed For the Exam

Evaluate the final model with independent test data



Needed For the Exam

Evaluate the final model with cross-validation



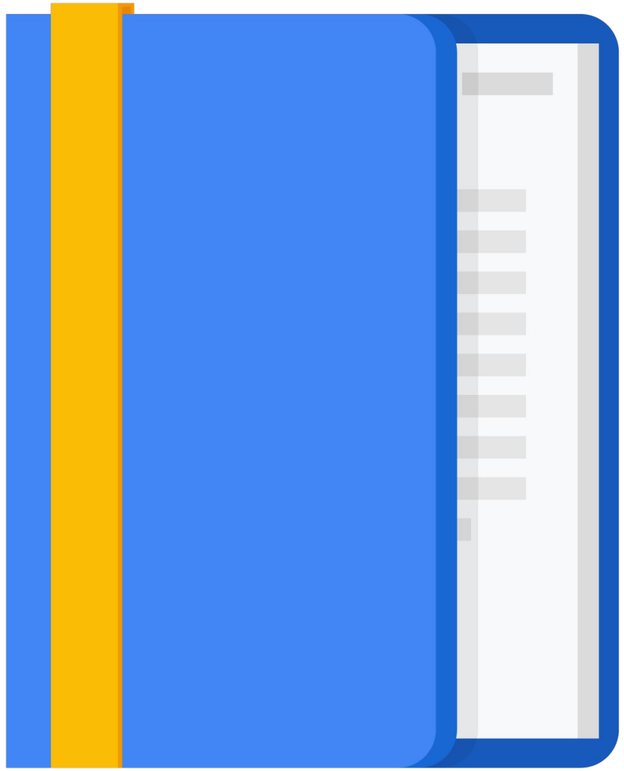
Course Agenda

Generalization in ML

Sampling

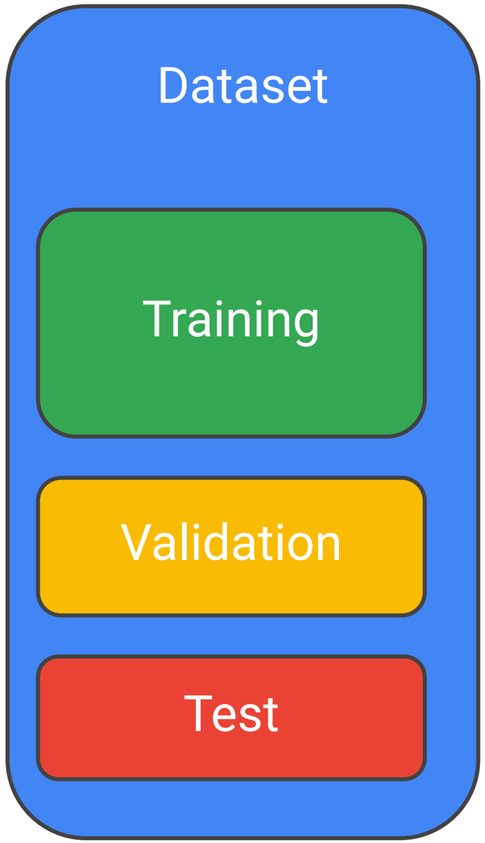
Regularizations

ML Model Performance Evaluation



FYI: For your knowledge

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	2004-08-07	TZ	SRQ	1255	IND	-14.0
2	2004-03-05	TZ	SRQ	2117	IND	-9.0
3	2004-04-12	TZ	SRQ	2000	IND	-17.0
4	2003-04-16	TZ	SRQ	1215	IND	-5.0
5	2005-03-20	TZ	SRQ	645	IND	14.0
6	2003-04-06	TZ	SRQ	1235	IND	-8.0

FYI: For your knowledge

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

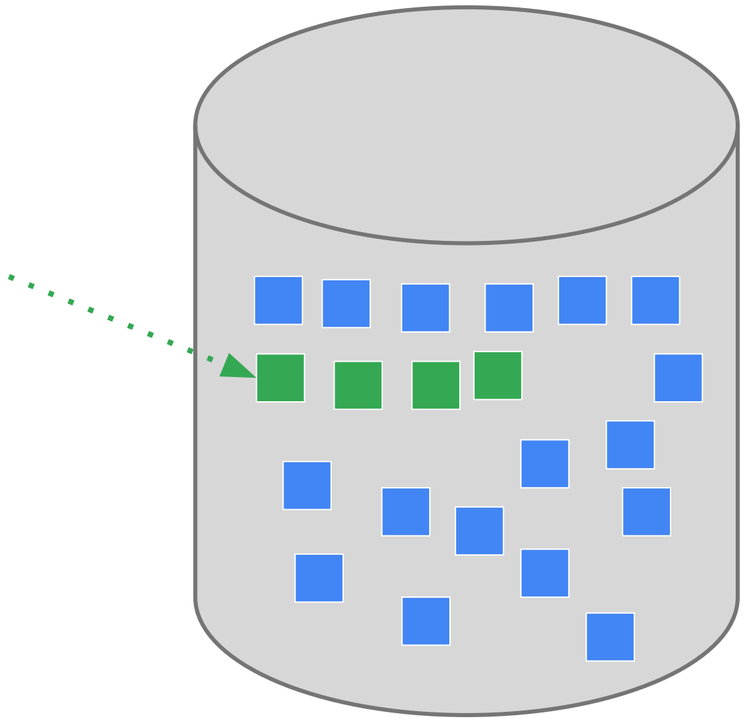
```
#standardSQL
SELECT
  date,
  airline,
  departure_airport,
  departure_schedule,
  arrival_airport,
  arrival_delay
FROM
  `bigquery-samples.airline_ontime_data.flights`
WHERE
  RAND() < 0.8
```

RAND will return a number between 0 and 1.

FYI: For your knowledge

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

Develop your TensorFlow code on a small subset of data, then scale it out to the cloud.



Full Dataset

We can extend this to creating 3 splits

```
#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)),700) >= 350
    AND
  MOD(ABS(FARM_FINGERPRINT(date)),700) < 525
```

Then take 1 in 70 flights.

Ignore the 50% of the dataset (training).

Choose data between 350 and 524 which is a new 25% sample for Validation.

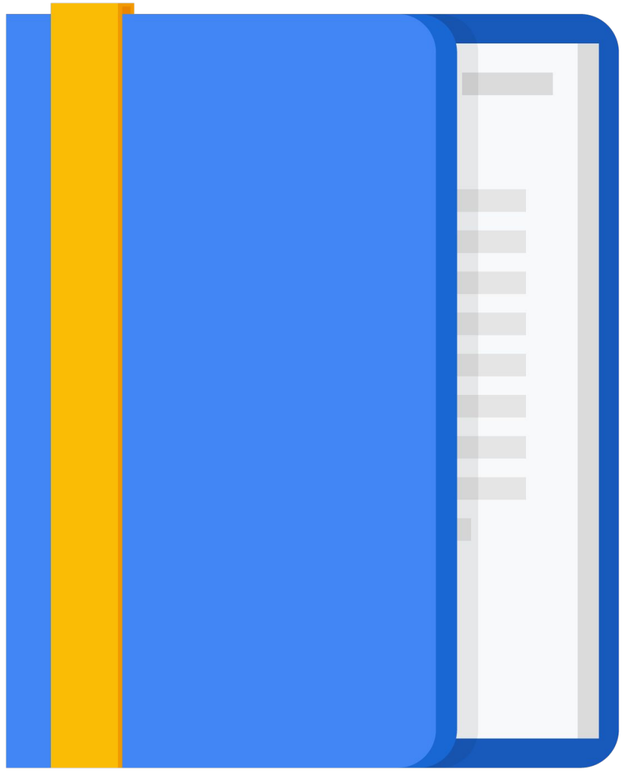
Course Agenda

Generalization in ML

Sampling

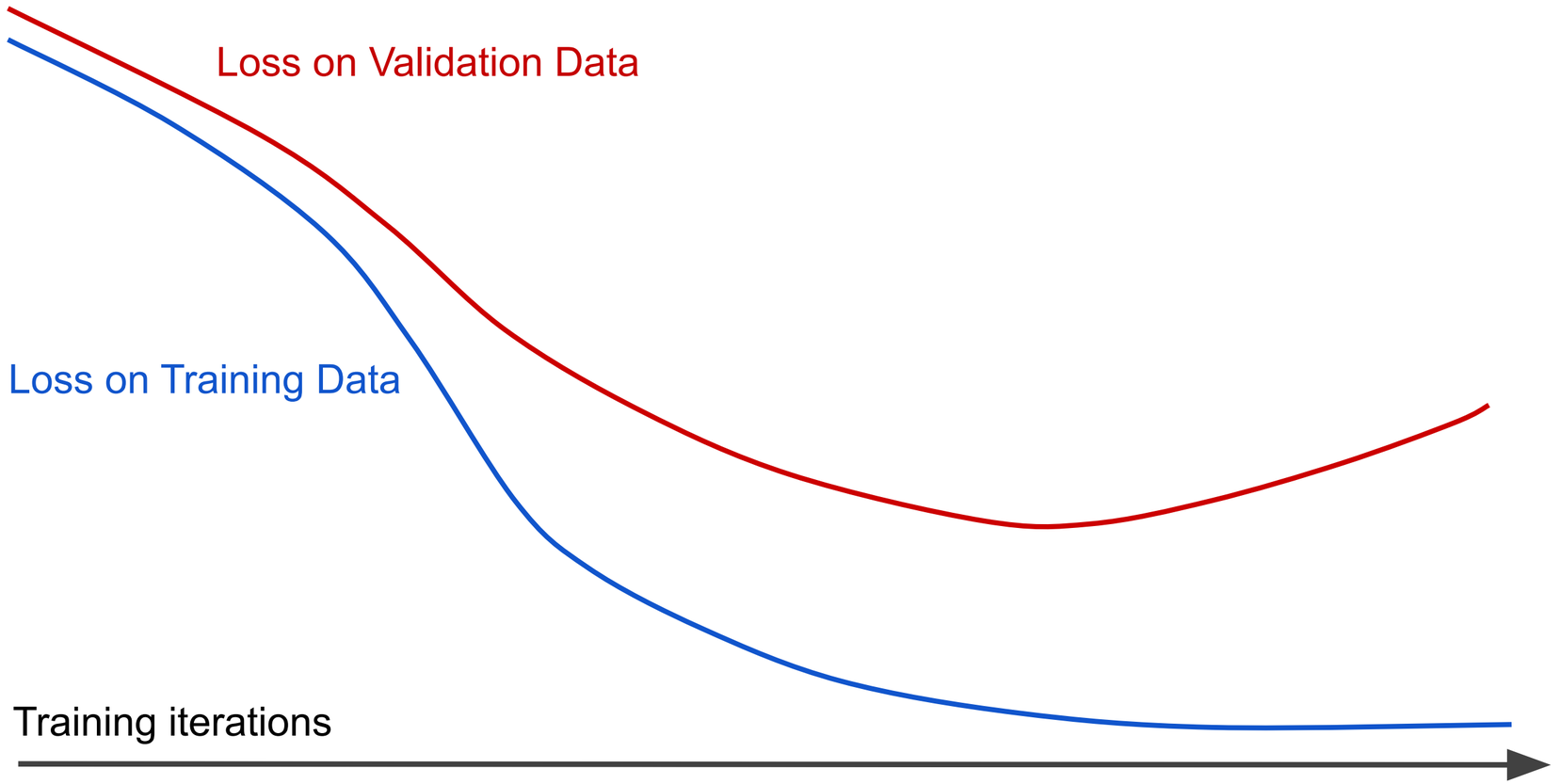
Regularizations

ML Model Performance Evaluation



Needed For the Exam

What is happening here? How can we address this?



REGULARIZATION!

Needed For the Exam

The simpler the better

Regularization is used when…

- You have too many features in your model
- You don’t know which feature should be discarded
- You want to reduce overfitting

Don’t cook with every spice in the spice rack!

Needed For the Exam

Occam's razor

When presented with competing hypothetical answers to a problem, one should select the one that makes the fewest assumptions. The idea is attributed to William of Ockham (c. 1287–1347).

source:https://en.wikipedia.org/wiki/Occam%27s_razor

Needed For the Exam

Regularization is a major field of ML research

Early Stopping

Parameter Norm Penalties

- L1 regularization
- L2 regularization
- Max-norm regularization

Dataset Augmentation

Noise Robustness

Sparse Representations

. . .

We will look into these methods.

FYI: For your knowledge

How can we measure model complexity?

Needed For the Exam

Logistic Regression Regularization Quiz

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

Needed For the Exam

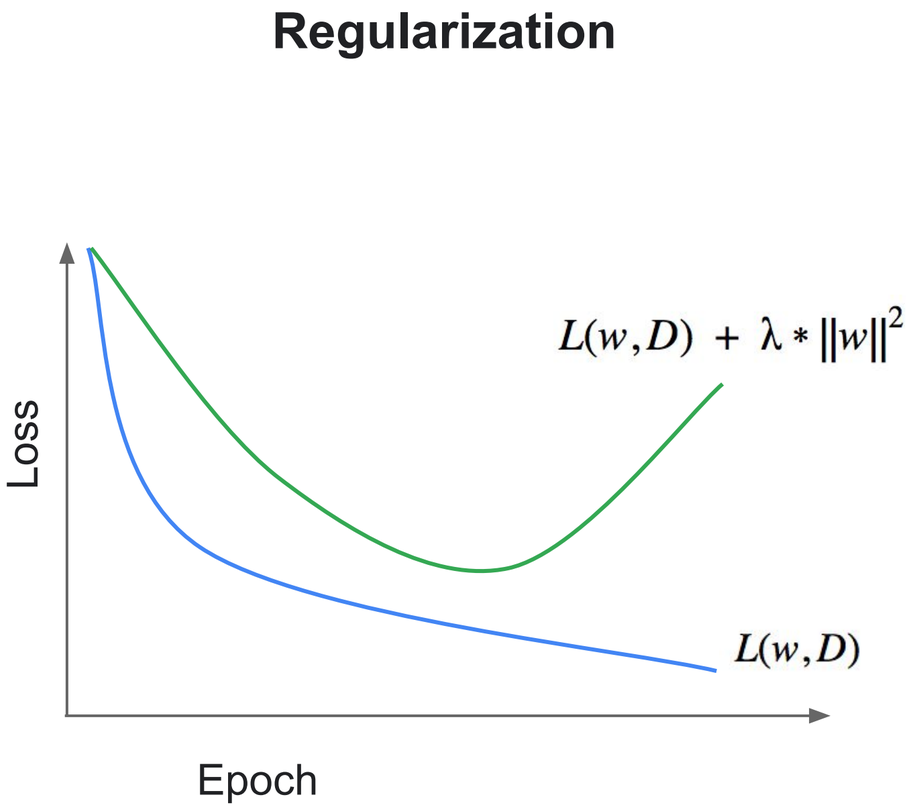
Logistic Regression Regularization Quiz

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

Needed For the Exam

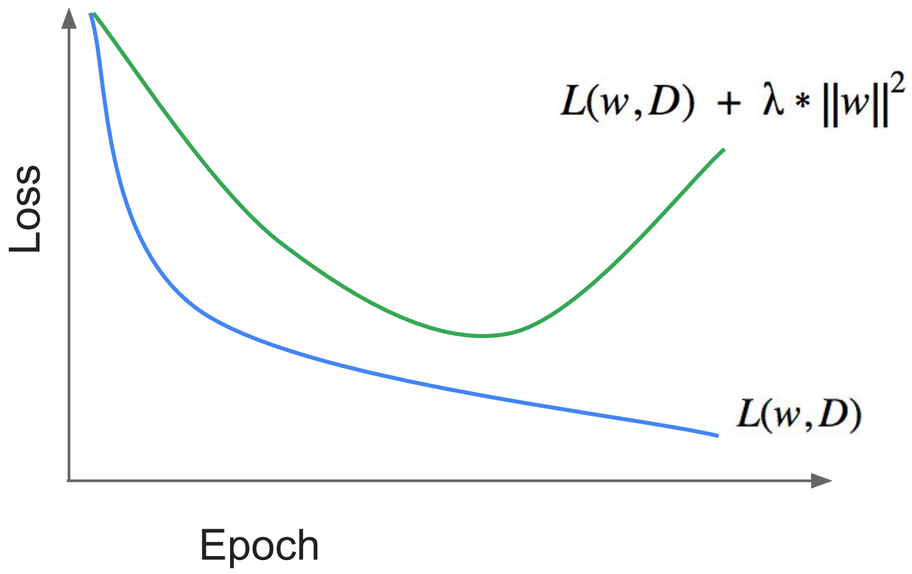
Often, we do both L1, L2 regularization and early stopping to counteract overfitting



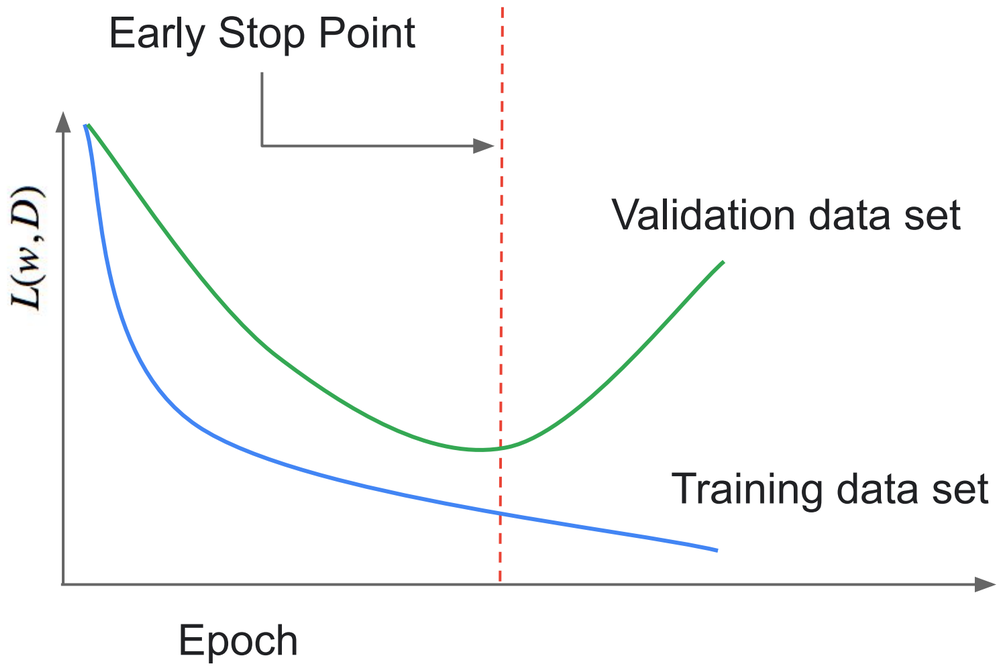
Needed For the Exam

Often, we do both regularization and early stopping to counteract overfitting

Regularization



Early stopping



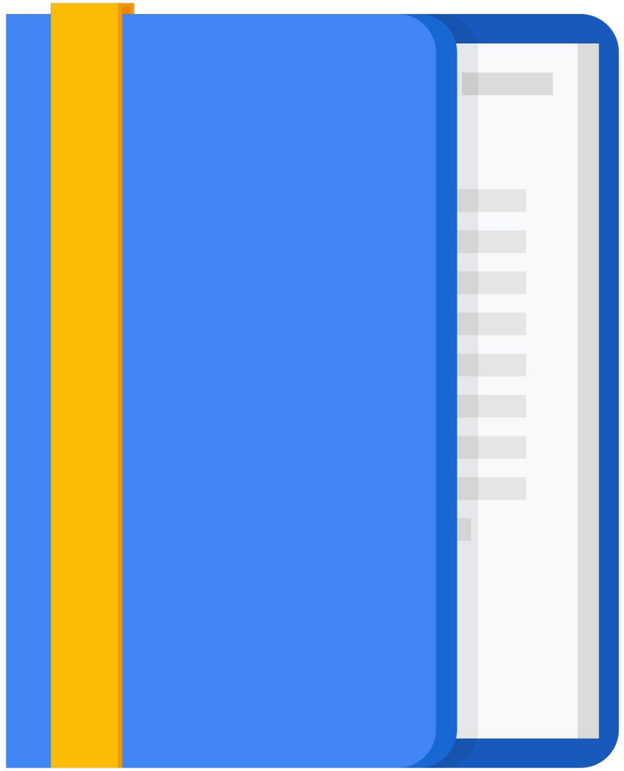
Course Agenda

Generalization in ML

Sampling

Regularizations

ML Model Performance Evaluation

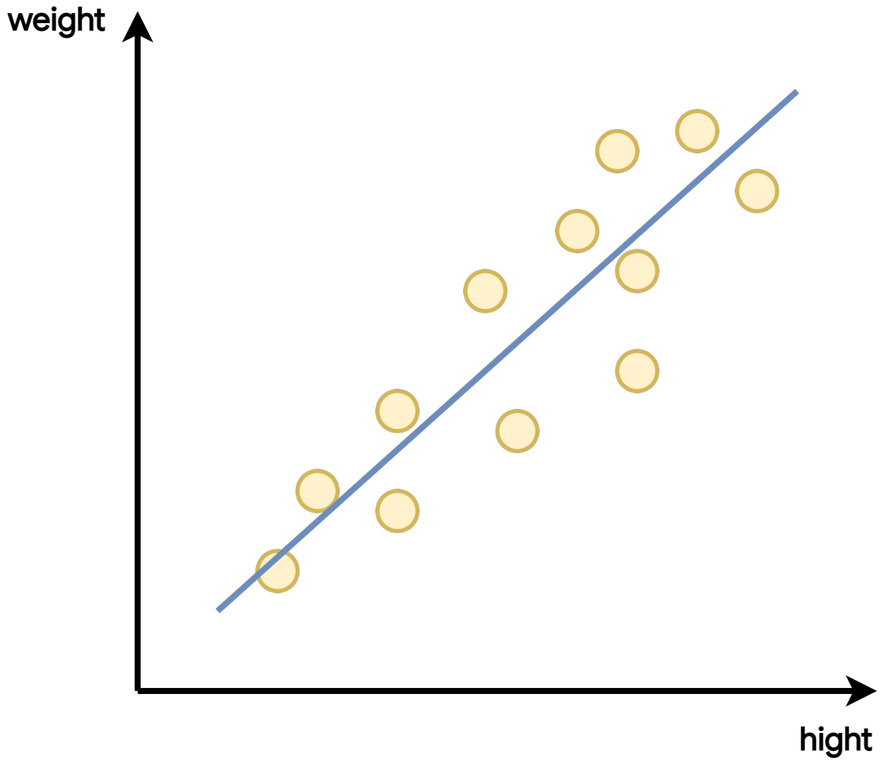


FYI: For your knowledge

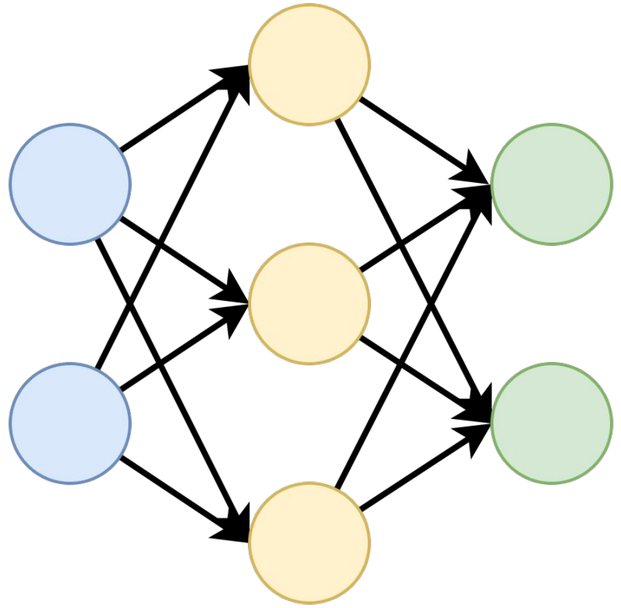
How do we compare algorithms?

FYI: For your knowledge

Is this

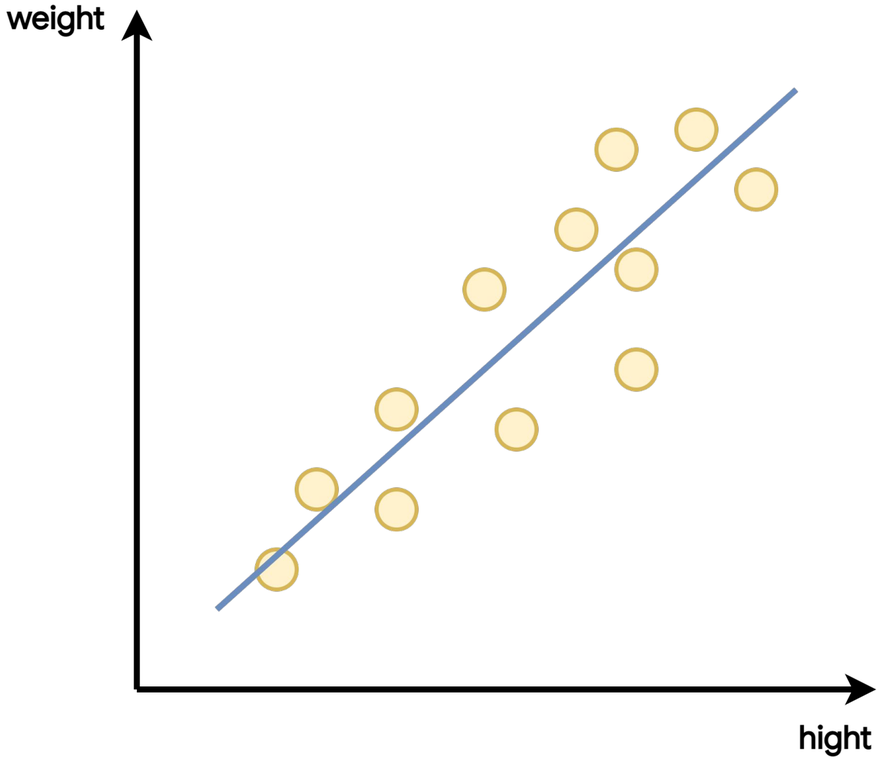


better than

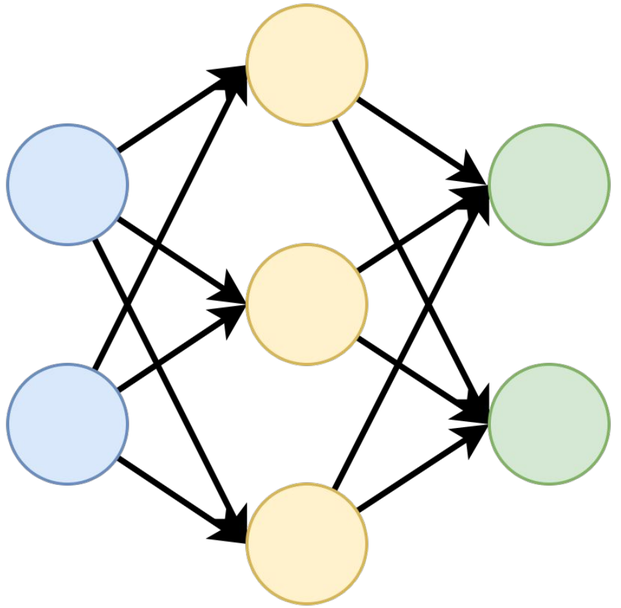


FYI: For your knowledge

Regression problems are easy



< = >



The lower RMSE wins (pss… careful not overfitting)

For classification problems?

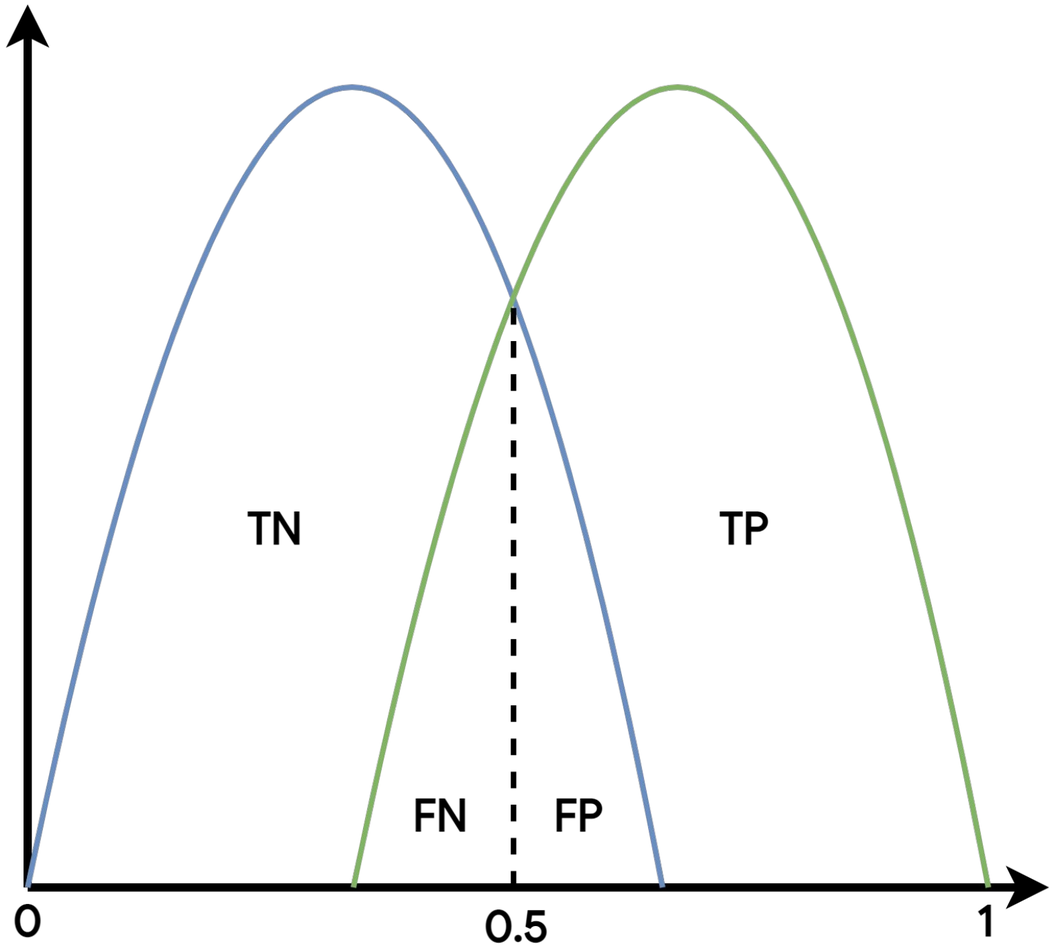
Needed For the Exam

Precision or Recall?

- If you want to reduce False Negative (Type 2 Error) then you should focus on Recall

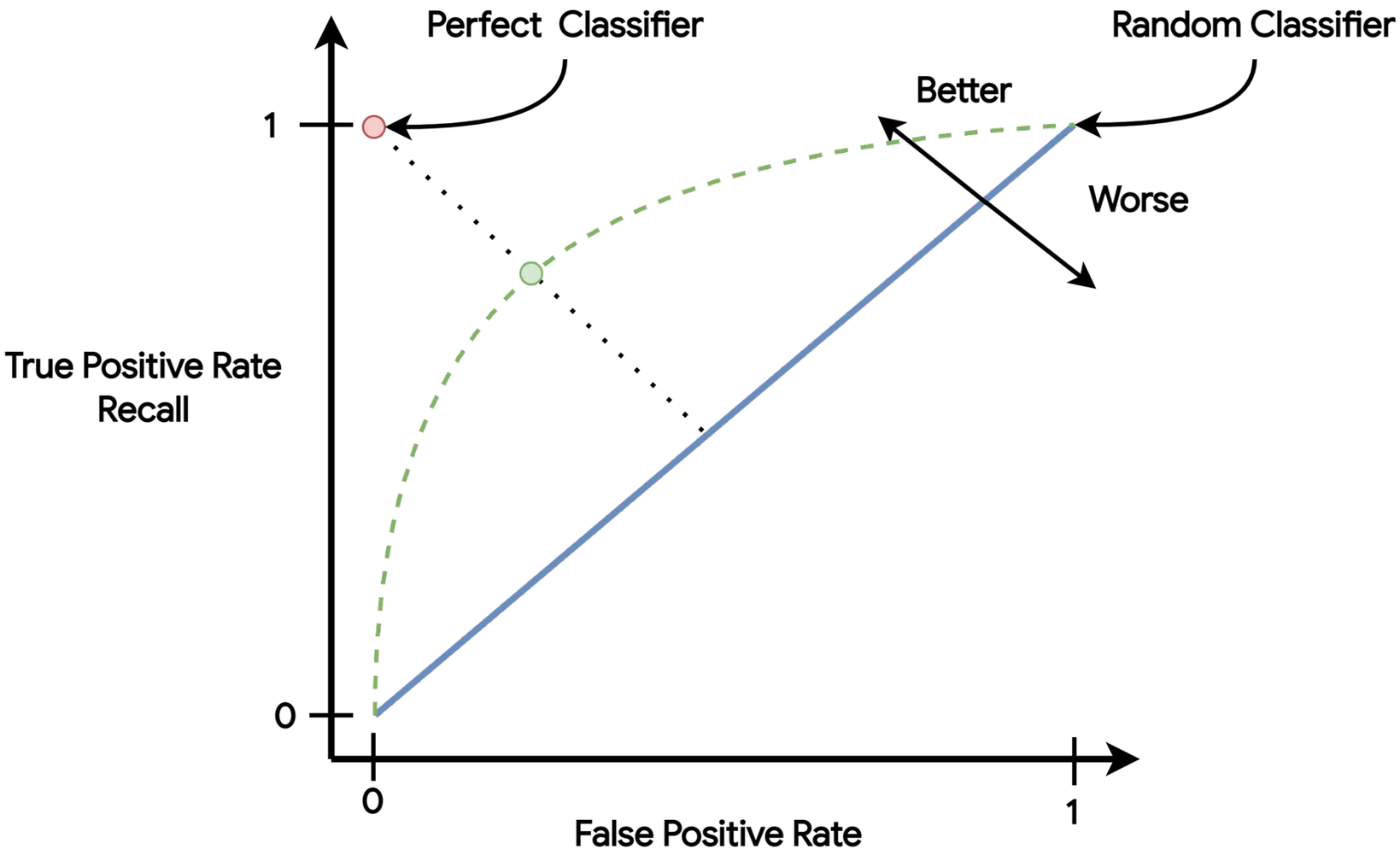
- If you want to reduce False Positive (Type 1 Error) then you should focus on Precision

- Threshold can be adjusted according with your business needs



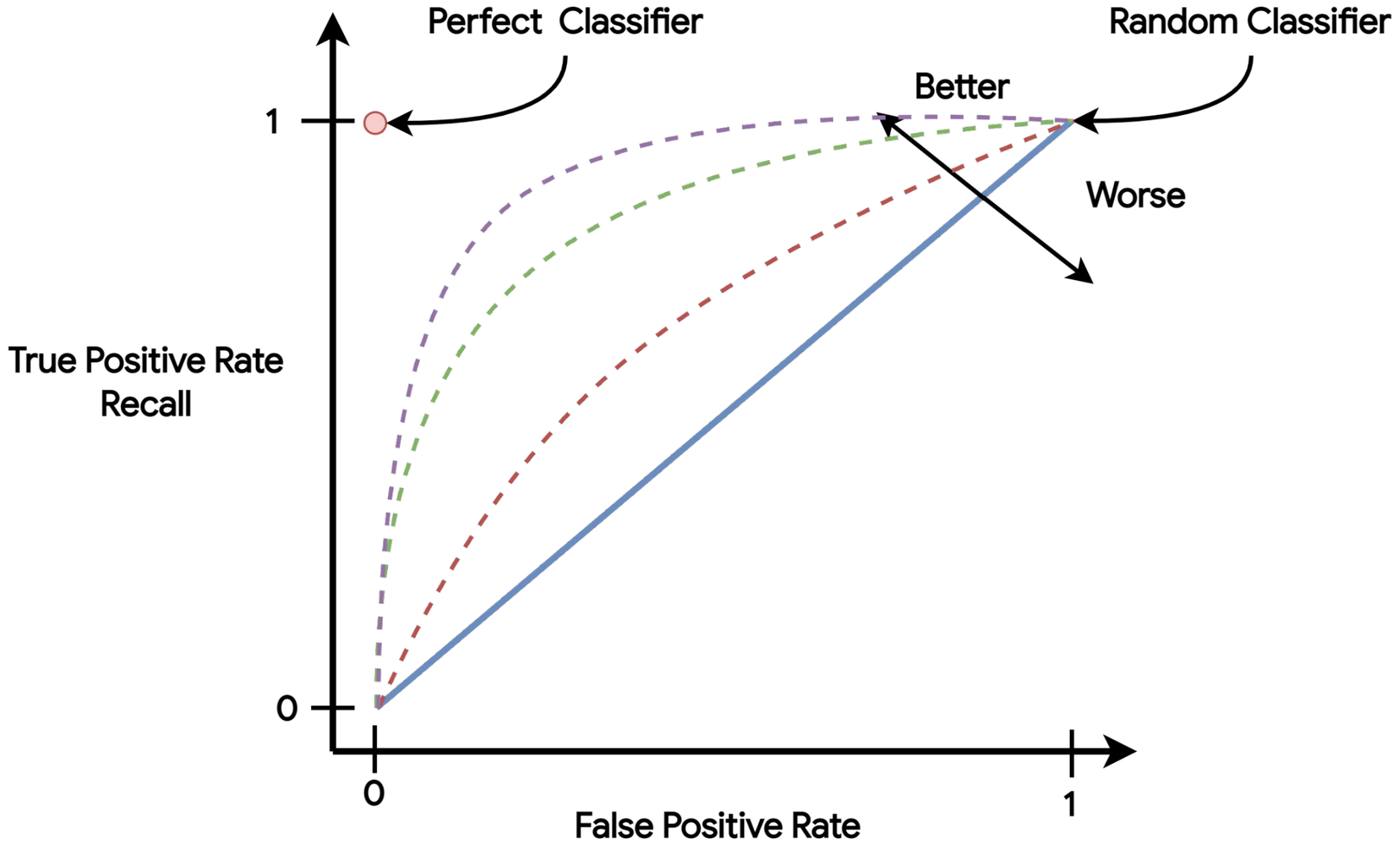
Needed For the Exam

How do I find the best threshold? Let's ROC



Needed For the Exam

How do I compare classifiers? Area Under Curve (AUC)



End of Section
Thank you

Google Cloud

Feature Engineering: Introduction to Feature Engineering

Instructor: Ben Ahmed

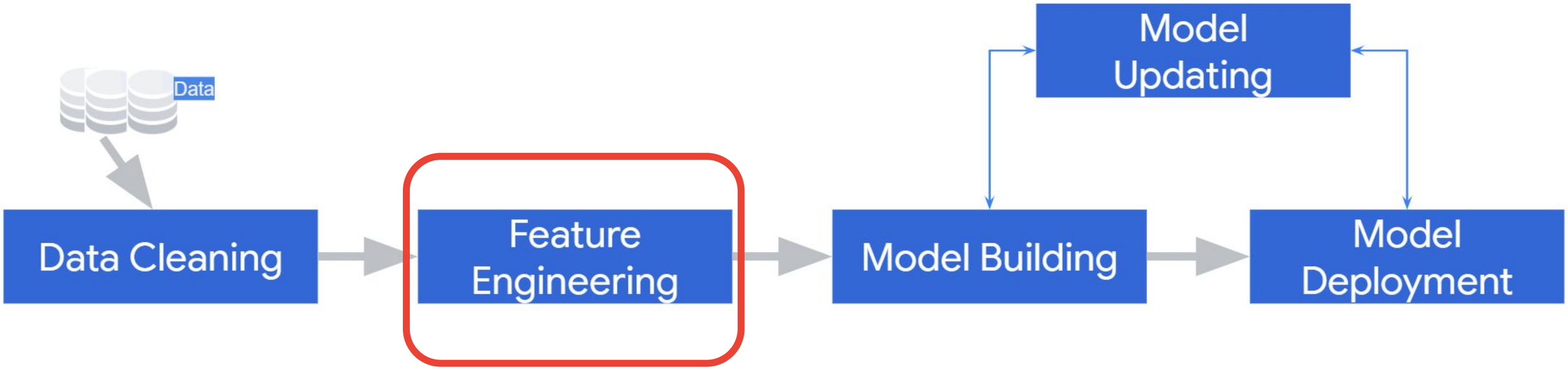
LAB Course: Feature Engineering

LAB: Using Feature Store

LAB: Performing Basic Feature Engineering In Keras

LAB: Performing Advanced Feature Engineering in Keras:

Feature engineering in predictive modeling



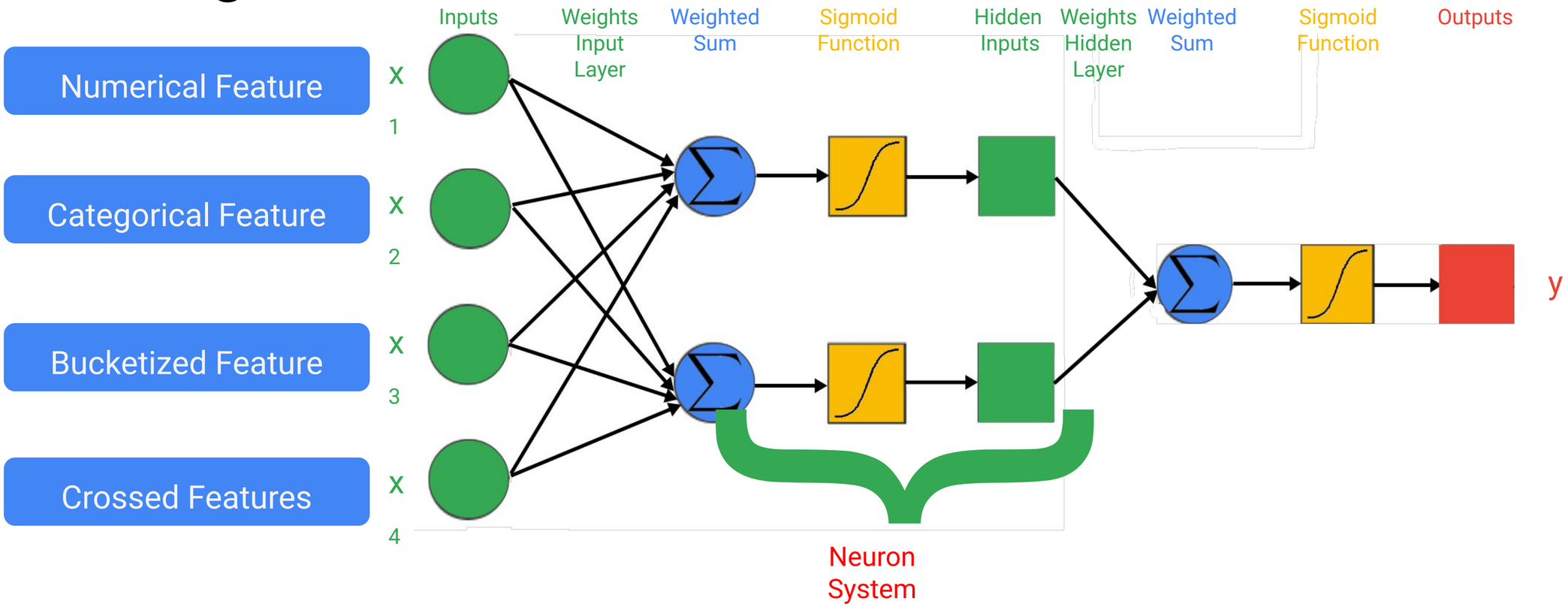
1 Process (What?) of feature engineering

Definition
This process attempts to create additional relevant features from the existing raw features in the data and to increase the predictive power of the learning algorithm.

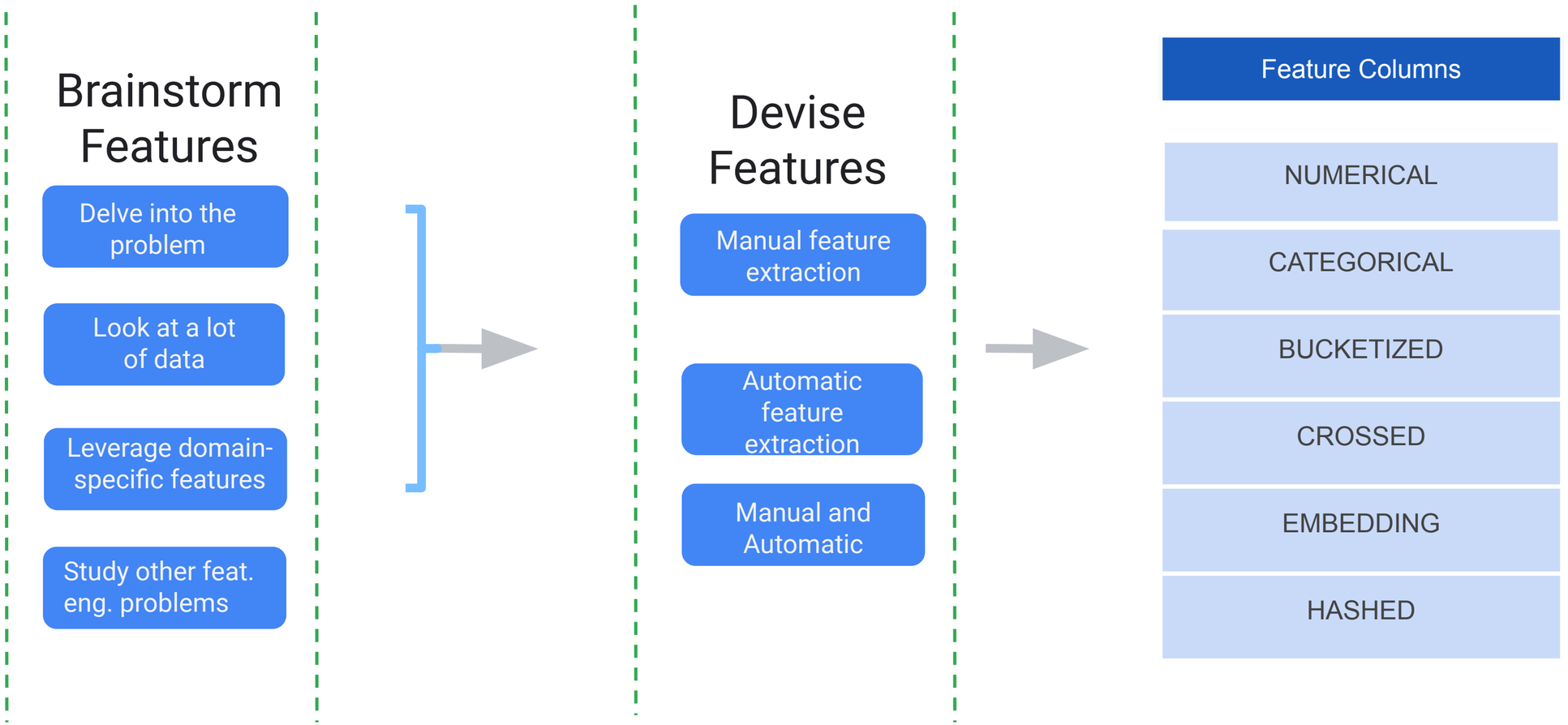
“Feature engineering is the process of transforming raw data into features that better represent the underlying problem to the predictive models, resulting in improved model accuracy on unseen data.”

Prof. Andrew Ng

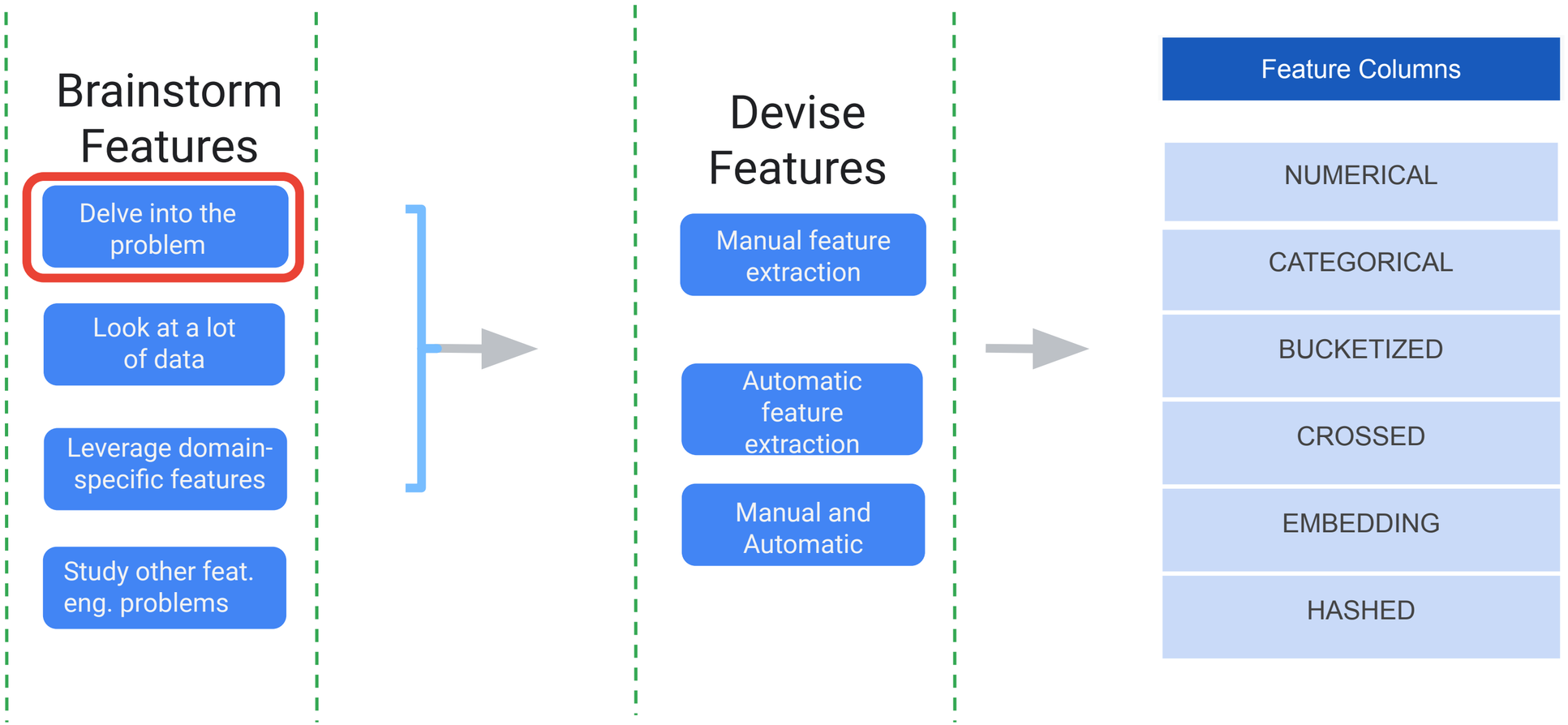
2 Purpose (Why): To improve the accuracy of models by increasing the predictive power of learning algorithms



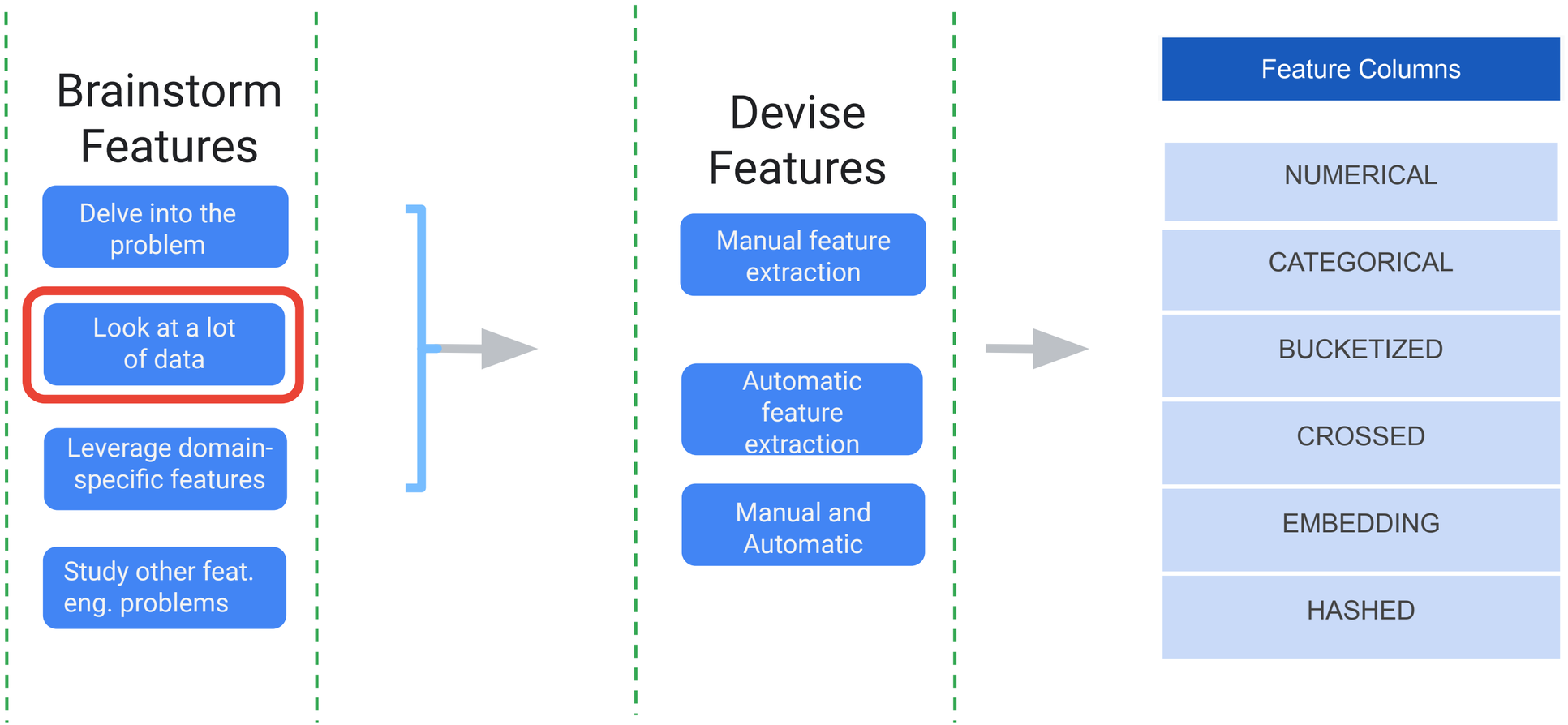
3 Process (How?) of feature engineering



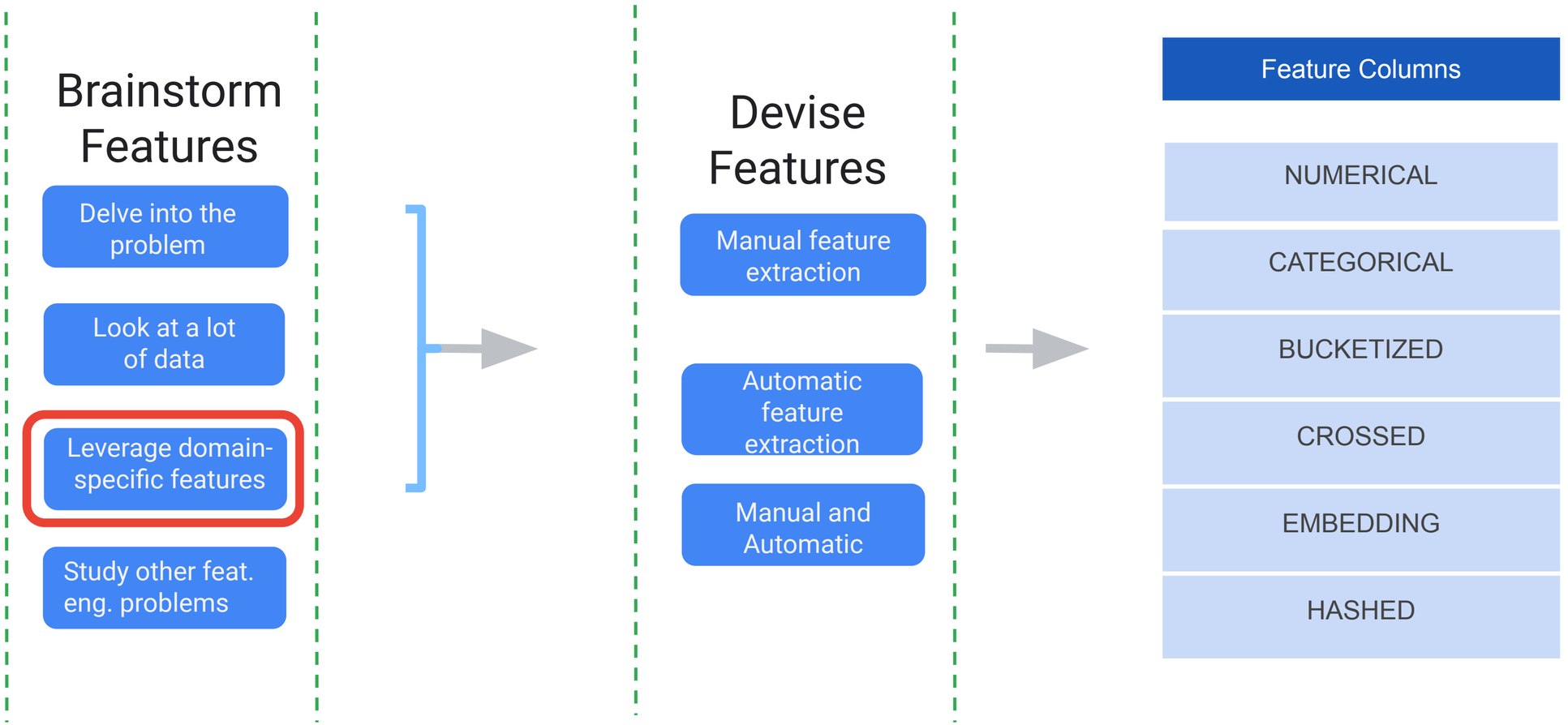
3 Process (How?) of feature engineering



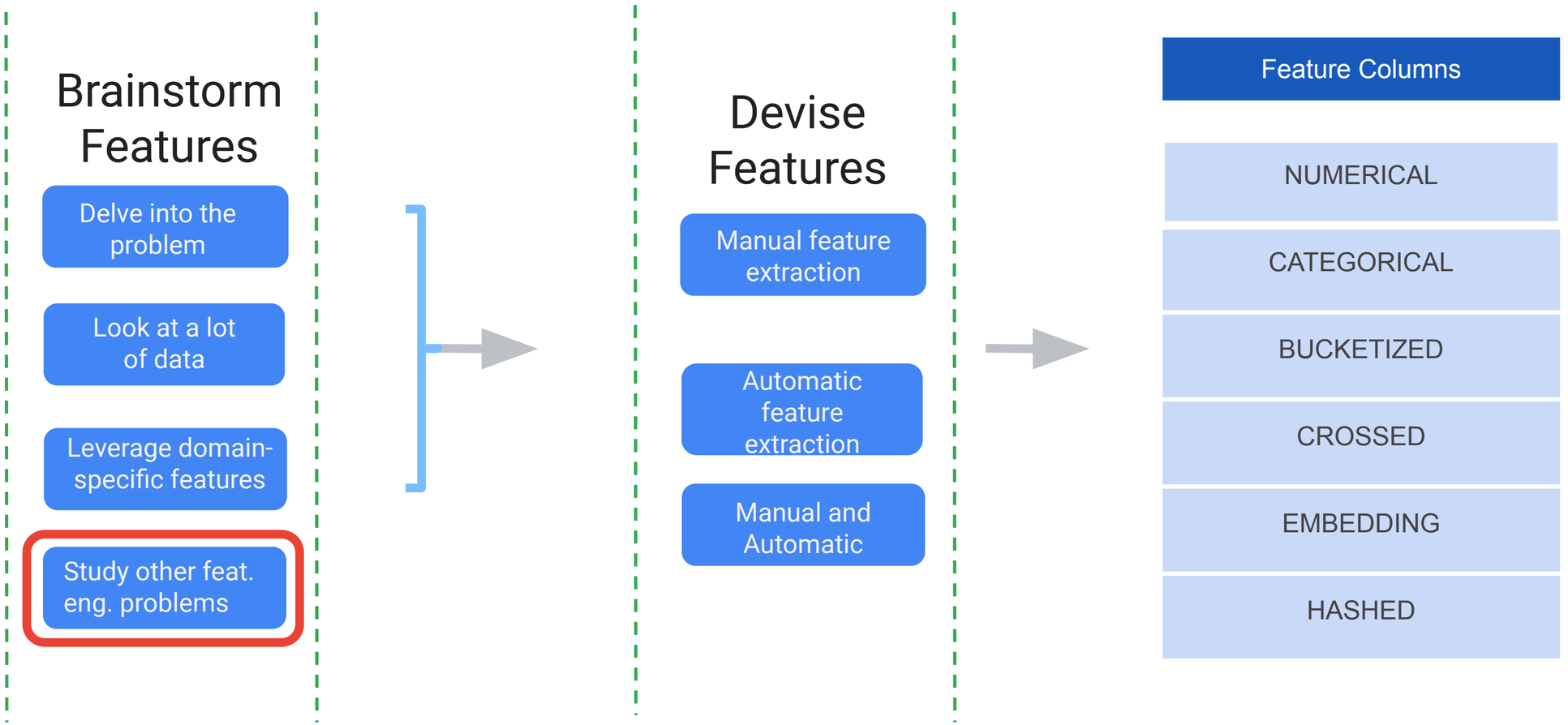
3 Process (How?) of feature engineering



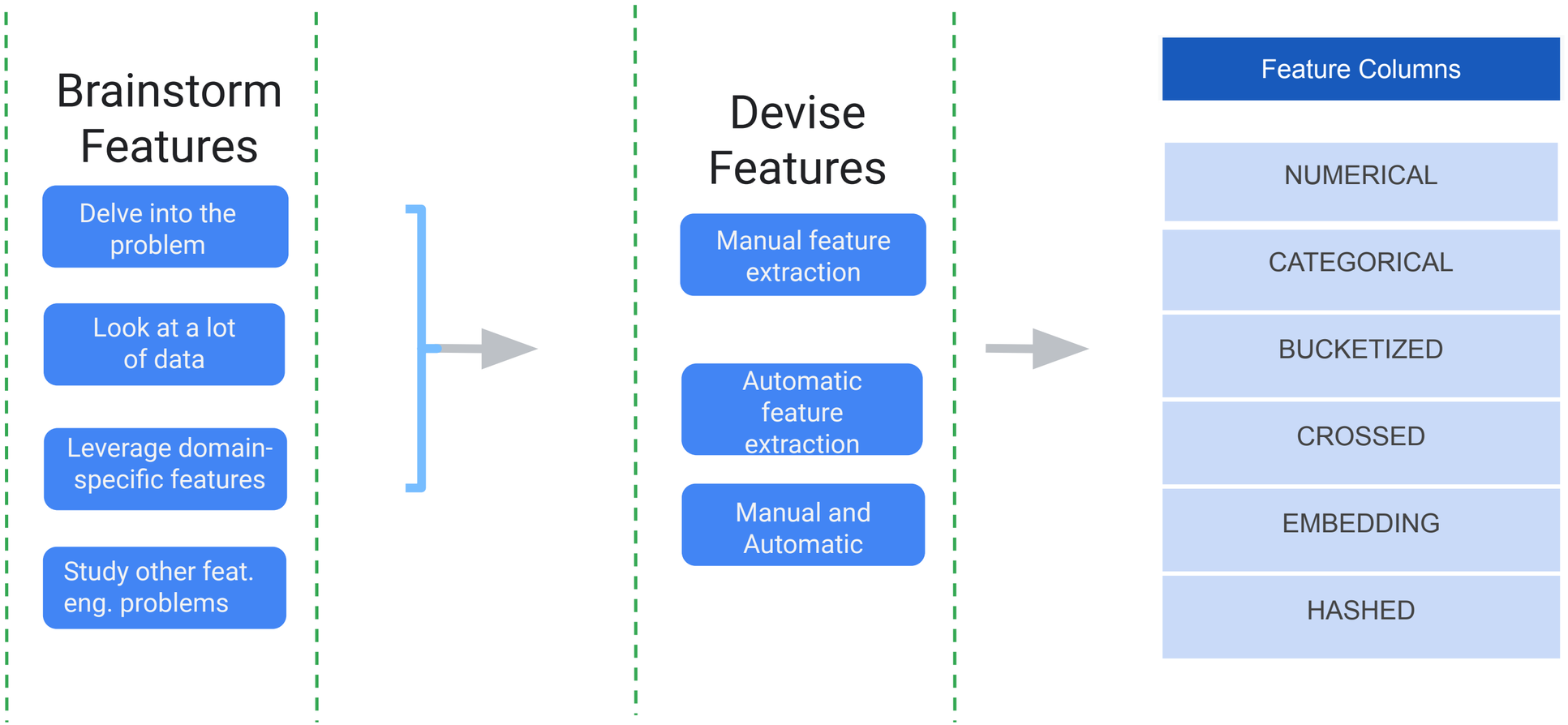
3 Process (How?) of feature engineering



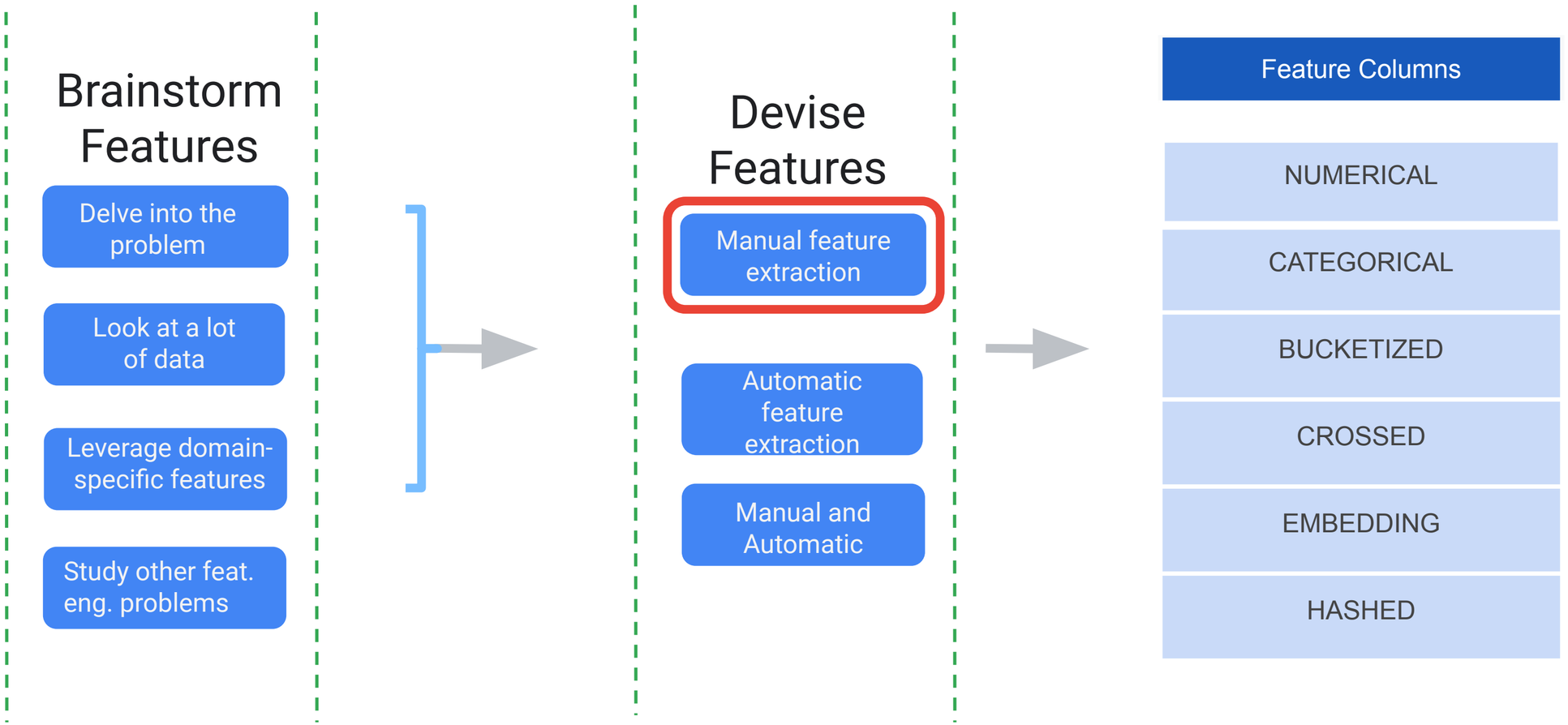
3 Process (How?) of feature engineering



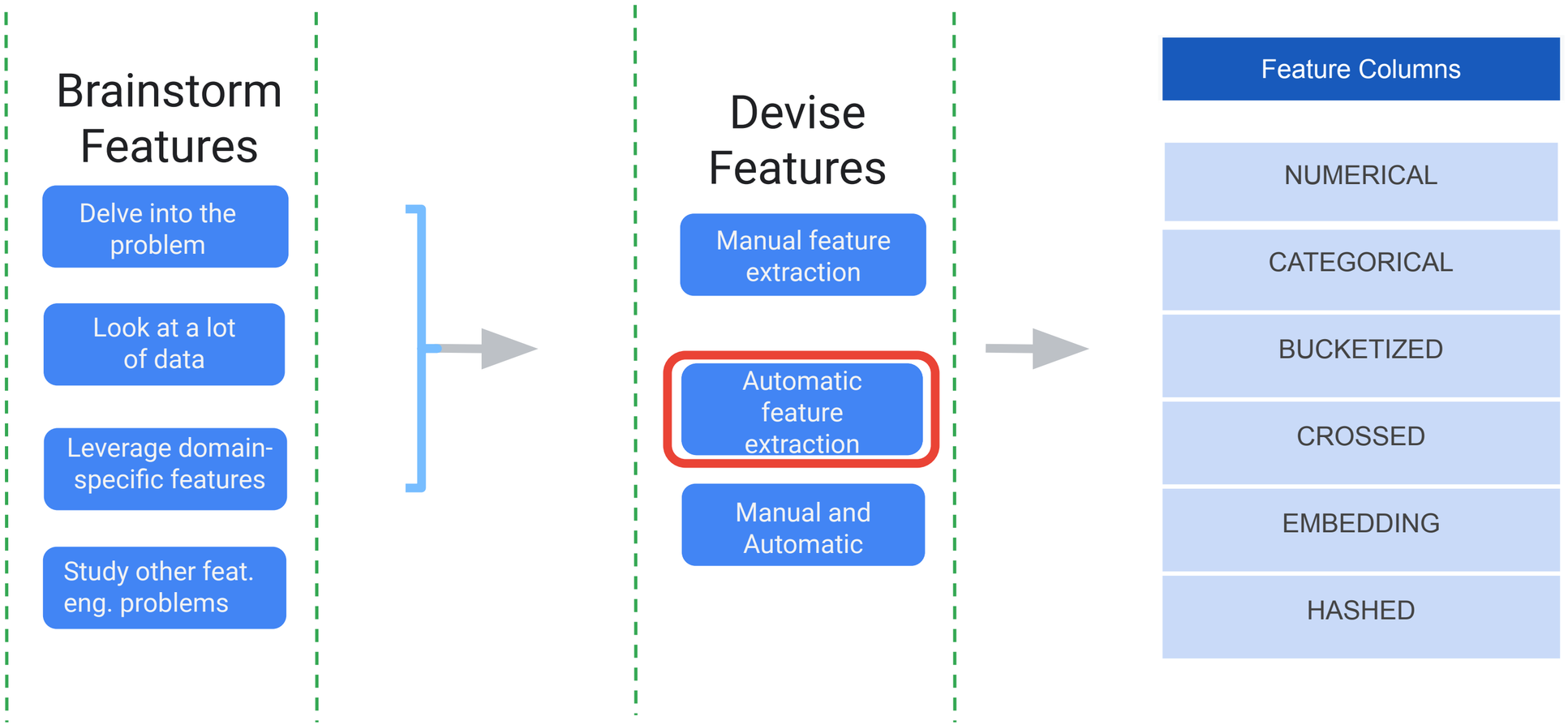
3 Process (How?) of feature engineering



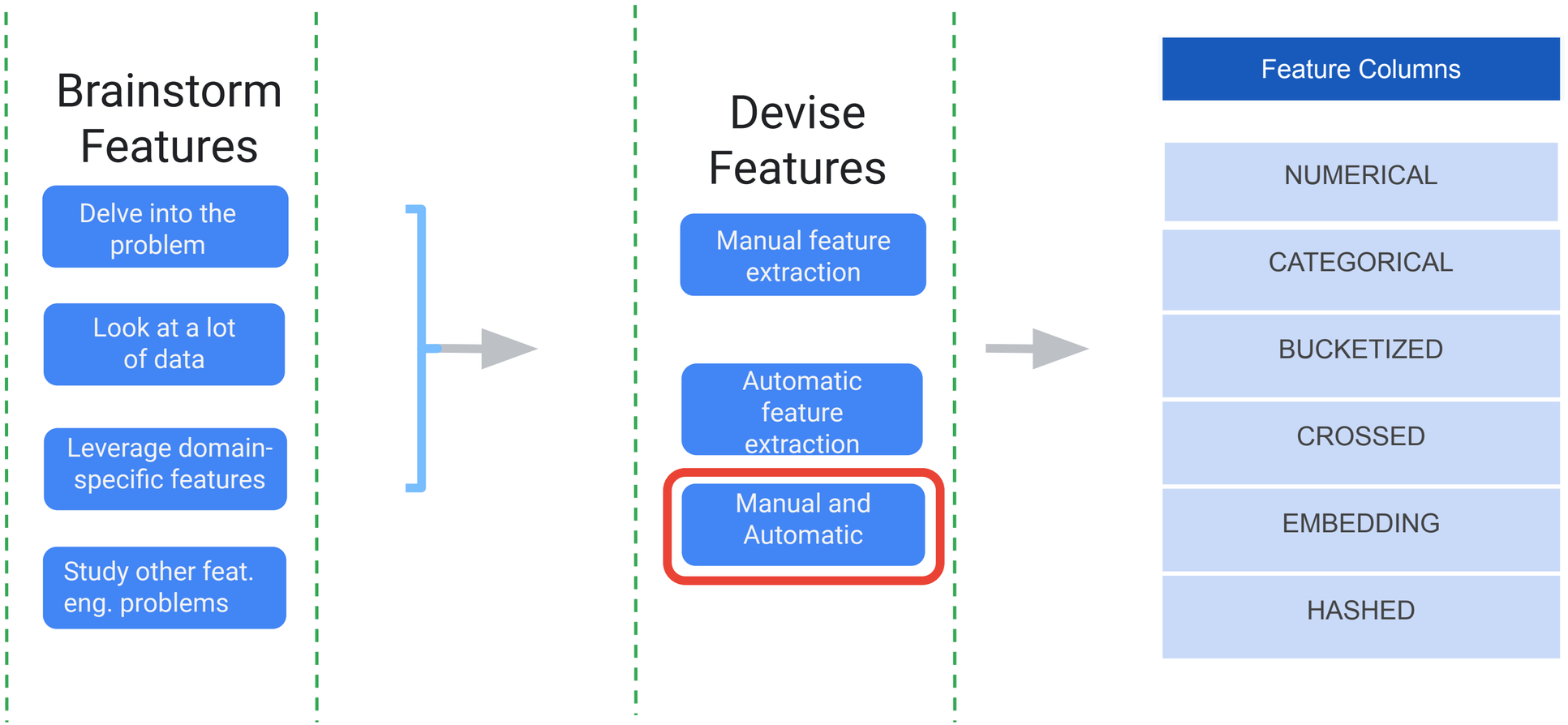
3 Process (How?) of feature engineering



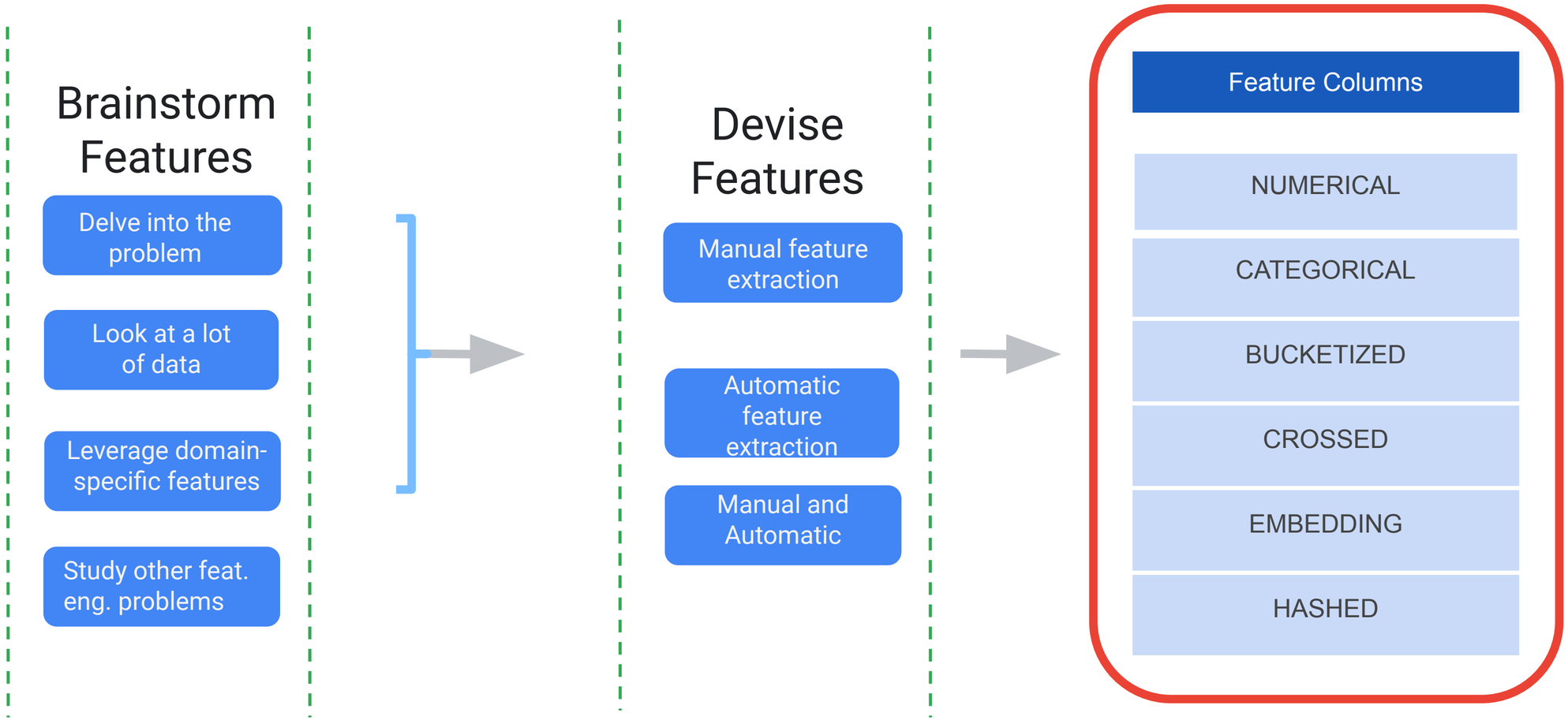
3 Process (How?) of feature engineering



3 Process (How?) of feature engineering



3 Process (How?) of feature engineering



5 Feature engineering types

Type	Example
Using indicator variables to isolate key information.	Isolates a specific area for our training dataset.
Highlighting interactions between two or more features.	Sum of two features, product of two features, etc.
Representing the same feature in a different way.	Create a new feature “grade” with "Elementary School," "Middle School," and "High School" as classes.
	Group similar classes, and then group the remaining ones into a single "Other" class.
	Transform categorical features into dummy variables.

5 Feature engineering types

Type	Example
Using indicator variables to isolate key information.	Isolates a specific area for our training dataset.
Highlighting interactions between two or more features.	Sum of two features, product of two features, etc.
Representing the same feature in a different way.	Create a new feature “grade” with "Elementary School," "Middle School," and "High School" as classes.
	Group similar classes, and then group the remaining ones into a single "Other" class.
	Transform categorical features into dummy variables.

5 Feature engineering types

Type	Example
Using indicator variables to isolate key information.	Isolates a specific area for our training dataset.
Highlighting interactions between two or more features.	Sum of two features, product of two features, etc.
Representing the same feature in a different way.	Create a new feature “grade” with "Elementary School," "Middle School," and "High School" as classes.
	Group similar classes, and then group the remaining ones into a single "Other" class.
	Transform categorical features into dummy variables.

5 Feature engineering types

Type	Example
Using indicator variables to isolate key information.	Isolates a specific area for our training dataset.
Highlighting interactions between two or more features.	Sum of two features, product of two features, etc.
Representing the same feature in a different way.	Create a new feature “grade” with "Elementary School," "Middle School," and "High School" as classes.
	Group similar classes, and then group the remaining ones into a single "Other" class.
	Transform categorical features into dummy variables.

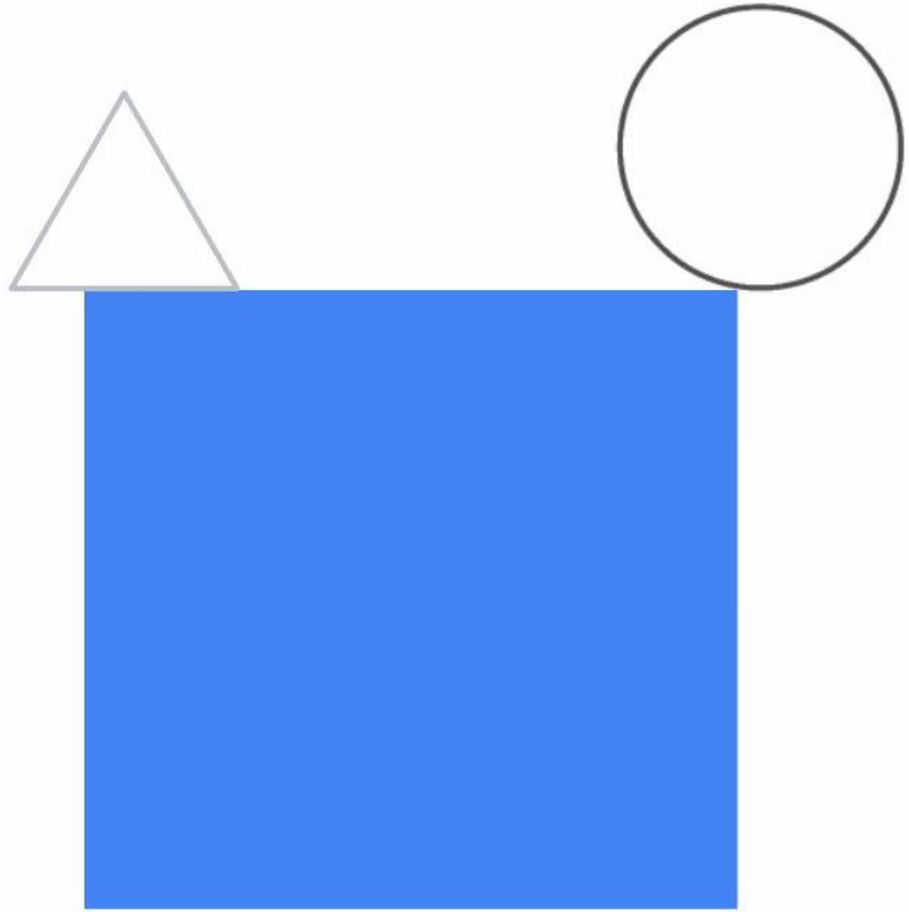
5 Feature engineering types

Type	Example
Using indicator variables to isolate key information.	Isolates a specific area for our training dataset.
Highlighting interactions between two or more features.	Sum of two features, product of two features, etc.
Representing the same feature in a different way.	Create a new feature “grade” with "Elementary School," "Middle School," and "High School" as classes.
	Group similar classes, and then group the remaining ones into a single "Other" class.
	Transform categorical features into dummy variables.

Break time

We will resume in:

<<15:00->>



Feature Selection with Correlation Analysis

What makes a good feature?

1

Be related to the objective

2

Be known at prediction-time

3

Be numeric with meaningful magnitude

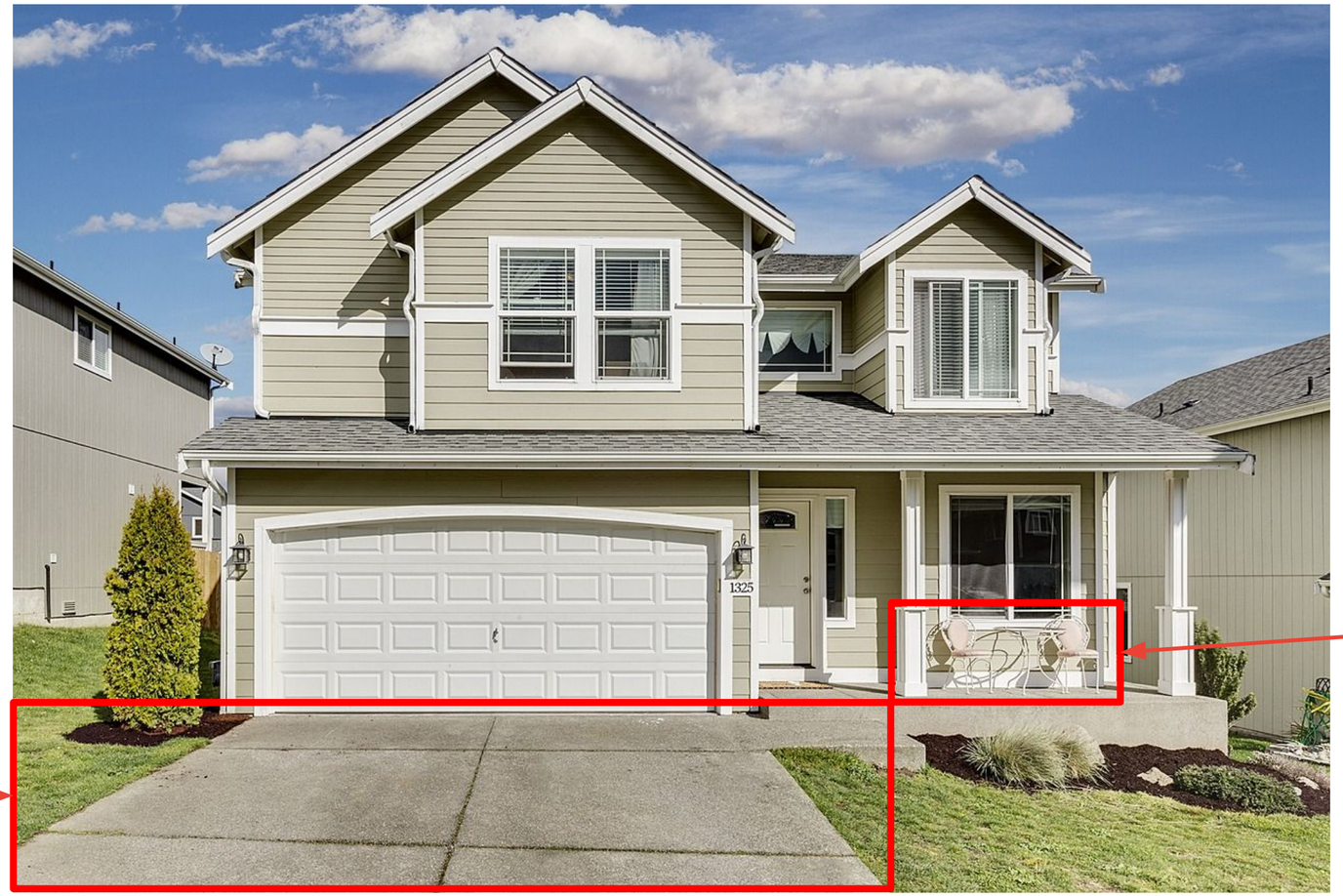
4

Have enough examples

5

Bring human insight to problem

Be related to the objective



Number of chairs on the porch

Number of concrete blocks in driveway

1 Is related to the objective

Raw Data


```
{
  "vendor_id": "CMT",
  "pickup_datetime": "2010-02-05 01:20:05 UTC",
  "dropoff_datetime": "2010-02-05 01:41:28 UTC",
  "pickup_longitude": "-73.979935",
  "pickup_latitude": "40.761105",
  "dropoff_longitude": "-73.96623",
  "dropoff_latitude": "40.689831",
  "rate_code": "1",
  "passenger_count": "1",
  "trip_distance": "84.8",
  "payment_type": "CAS",
  "fare_amount": "0.0",
  "extra": "0.0",
  "mta_tax": "0.0",
  "imp_surcharge": null,
  "tip_amount": "0.0",
  "tolls_amount": "0.0",
  "total_amount": "0.0",
  "store_and_fwd_flag": "0"
},
```


Feature Engineering

Feature Vector

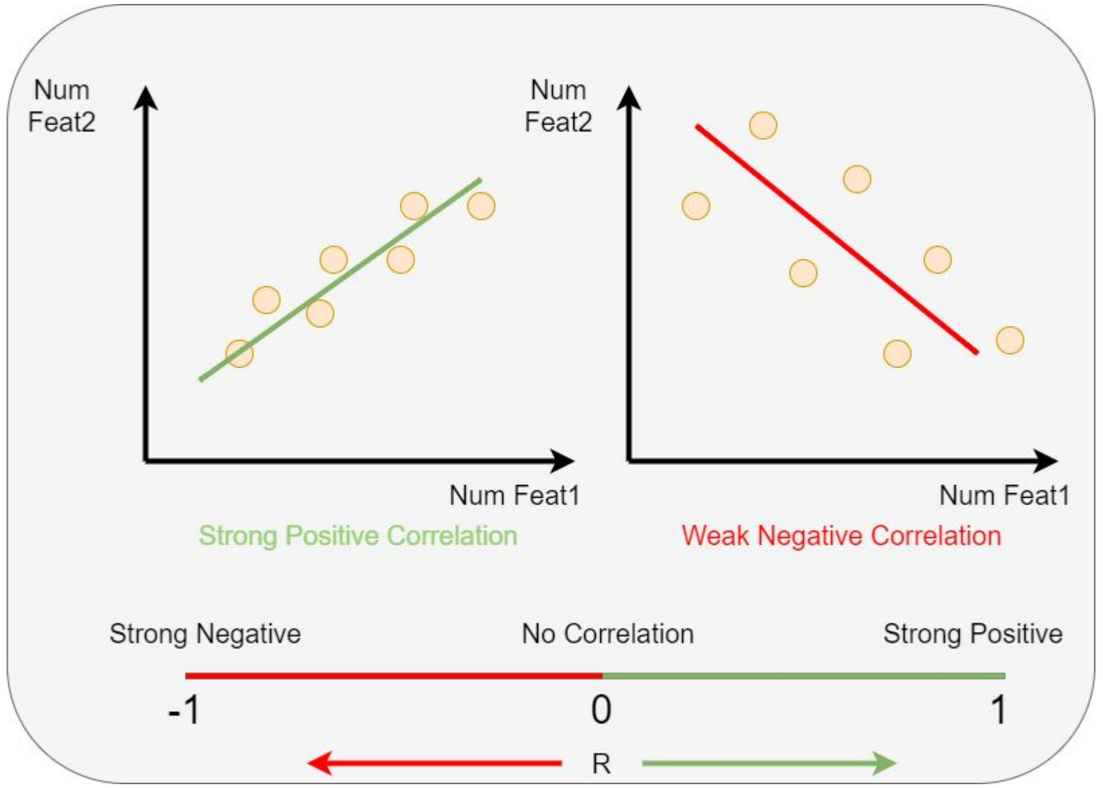

```
{
  "fare_amount": "2.5",
  "dayofweek": "1",
  "pickuplon": "-73.937463",
  "pickuplat": "40.758052",
  "dropofflon": "-73.937477",
  "dropofflat": "40.758062"
},
```


Main tools for feature correlation

1. Pearson Coefficient
2. Chi-Squared Test
3. Mutual Information – Information Gain
4. T-Test / ANOVA
5. PCA

1 Pearson Coefficient

- Returned values (R, p-value)
- Between numeric to numeric
- For p-value < 0.5% gives us strong confidence

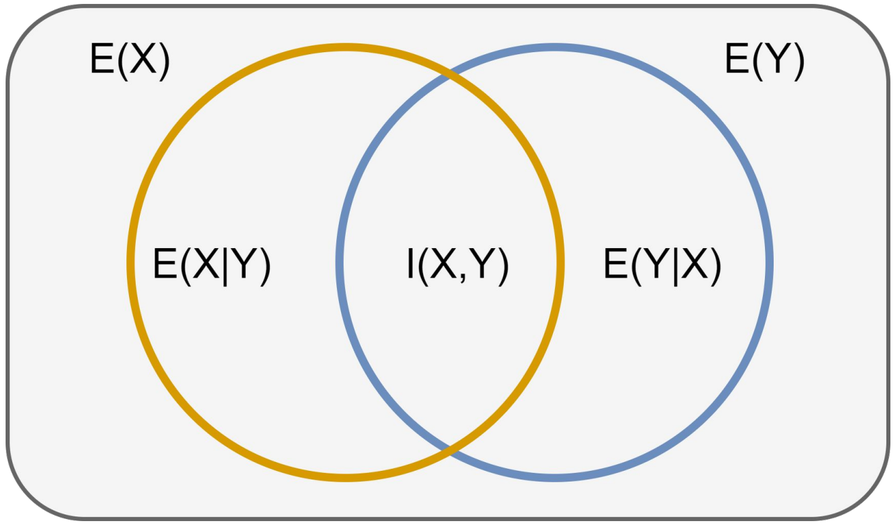


2 Chi-Squared Test

- Returned values (X2, p-value)
- Between categorical to categorical
- Test of Independence between features
- Test of dependence between a feature and the target

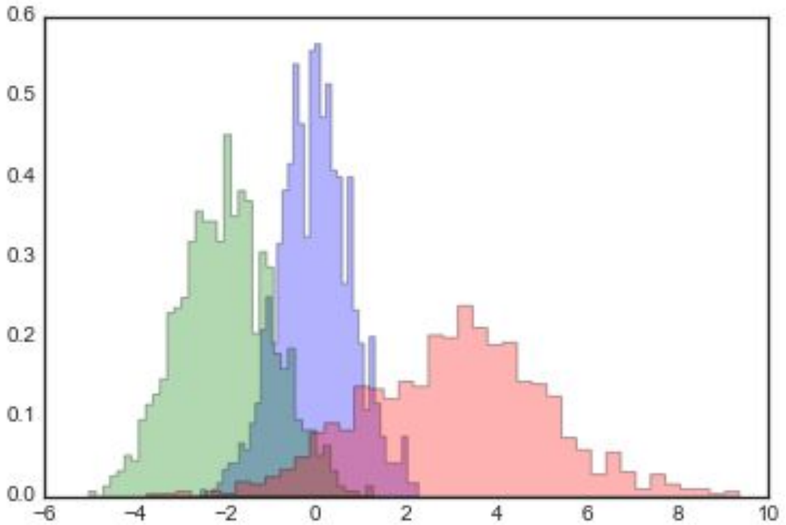
3 Mutual Information – Information Gain

- Share entropy between a feature and the target
- Between categorical to categorical
- Test of dependence between a feature and the target



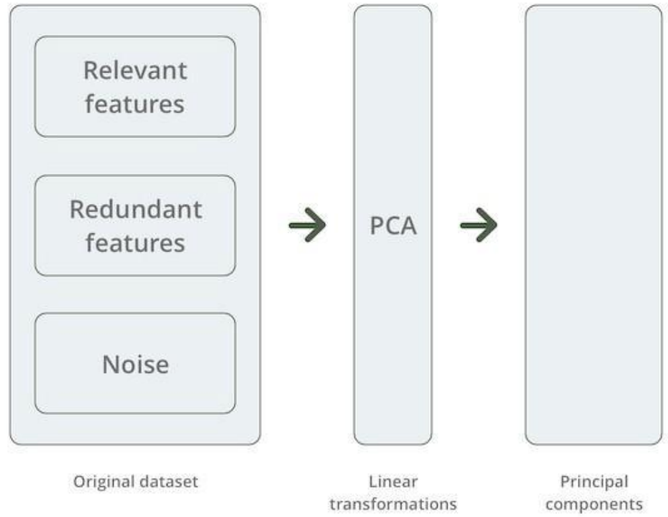
4 T-Test / ANOVA: Analysis of Variance

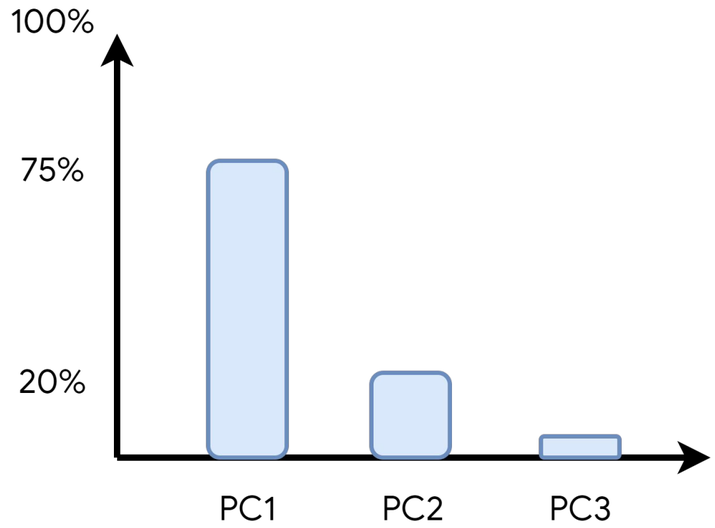
- Use T-Test between binary category and numerical feature
- Use ANOVA between multi-category and numerical feature
- Test of dependence between a feature and the target



5 PCA: Principal Component Analysis

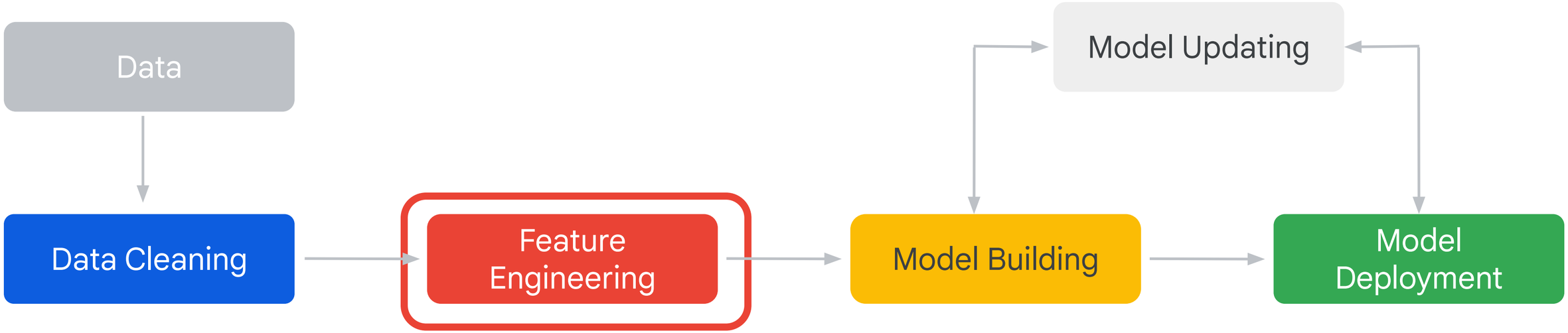
- PCA is used for dimension reduction.
- PCA finds out the most important features
- Use also for clustering features





Feature Columns

Feature engineering in predictive modeling



1 - Process (What?) of feature engineering

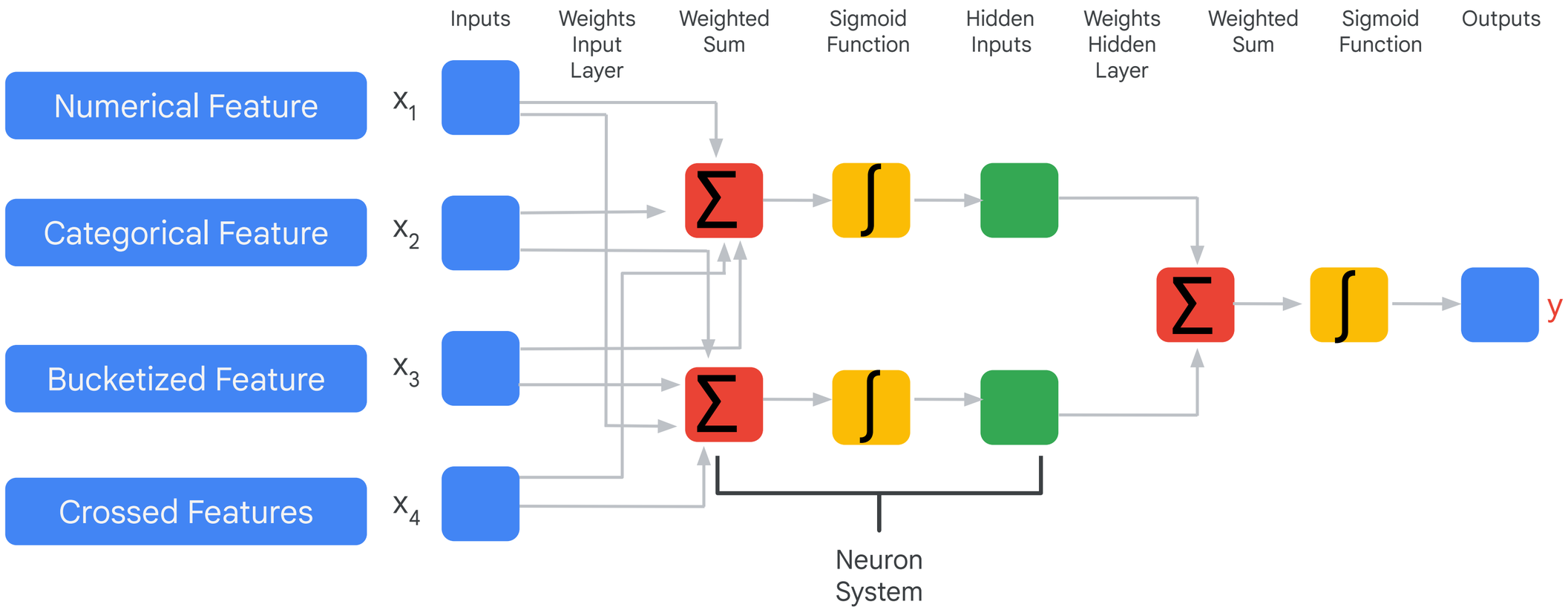
Definition

This process attempts to create additional relevant features from the existing raw features in the data and to increase the predictive power of the learning algorithm.

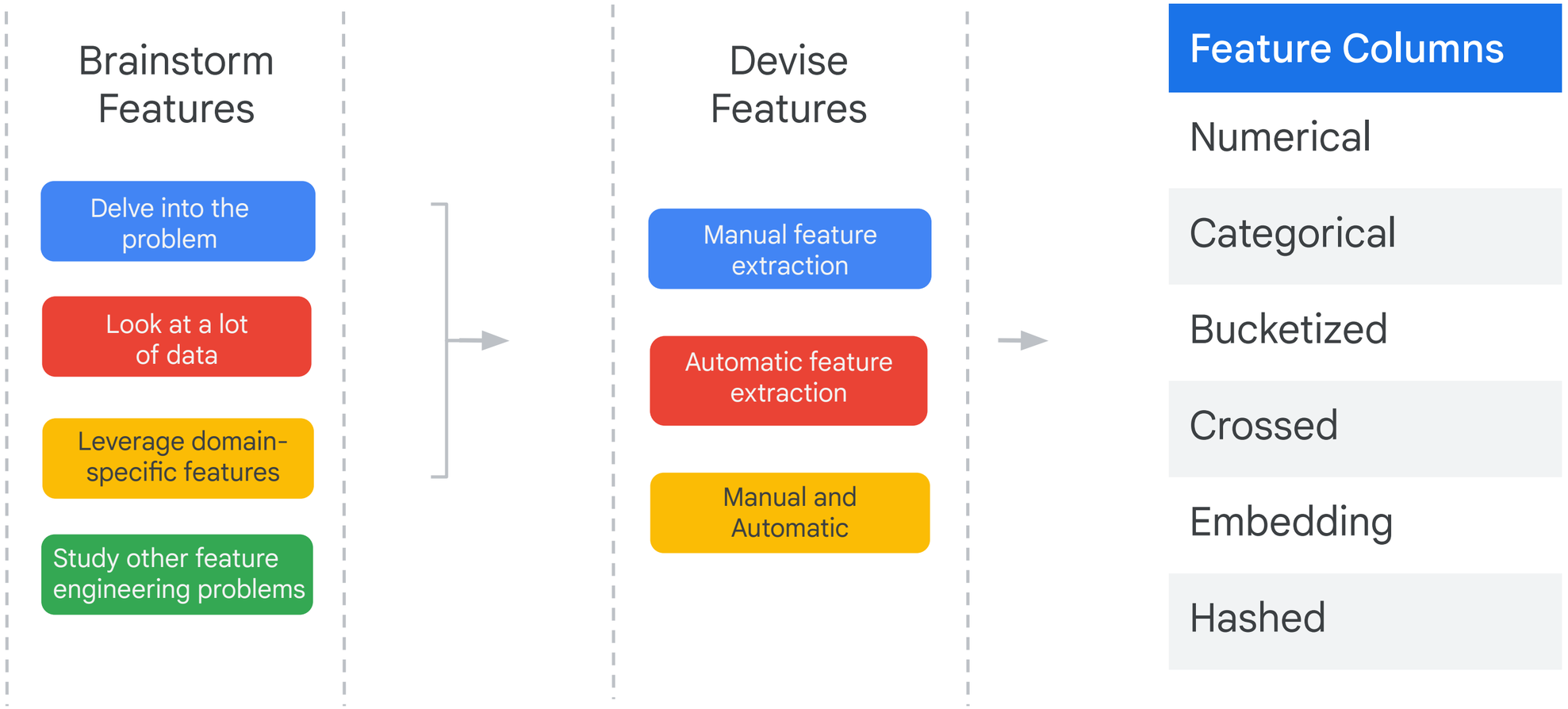
Feature engineering is the process of transforming raw data into features that better represent the underlying problem to the predictive models, resulting in improved model accuracy on unseen data.

Prof. Andrew Ng

2 - Purpose (Why): To improve the accuracy of models by increasing the predictive power of learning algorithms

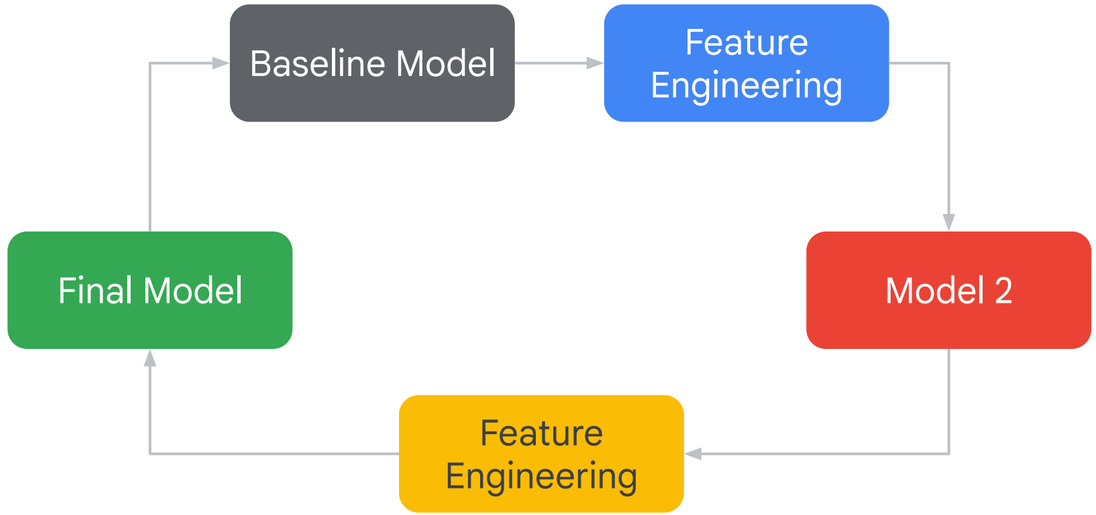


3 - Process (How?) of feature engineering



4 - Iterative Process

Example: Process can continue until RMSE is lowest



Raw data are converted to numeric features in different ways

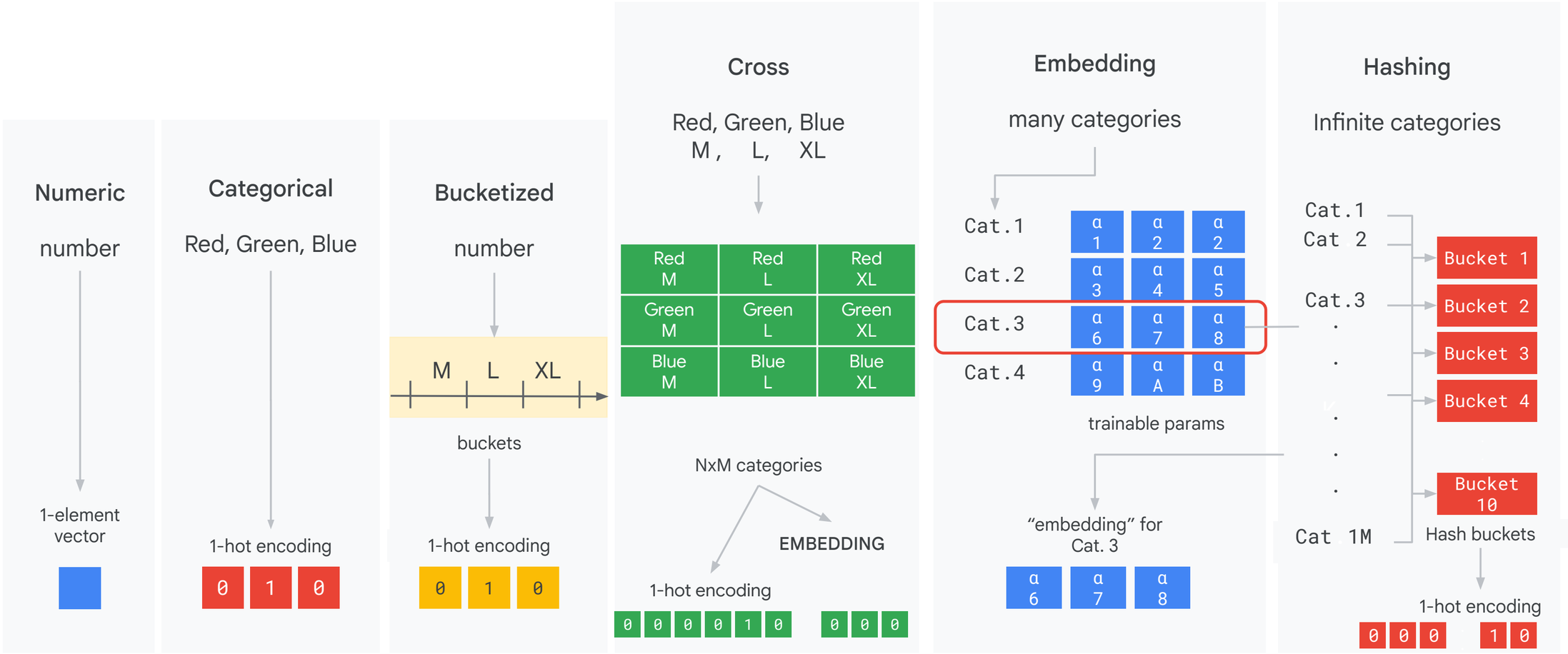
```
{
  "transactionId": 42,
  "name": "Ice Cream",
  "price": 2.50,
  "tags": ["cold", "dessert"],
  "servedBy": {
              "employeeId": 72365,
              "waitTime": 1.4,
              "customerRating": 4},
  "storeLocation": {
  "latitude":   35.3,
  "longitude": -98.7}
},
```



```
[..., 1, 2.50, …,                ]
[..., 0, 8.99, …,                ]
[..., 0, 3.45, …,                ]
…
```

In estimator API,
this is a feature column.

Feature columns



With vocab list, vocab files, or identity

Partition by range

Capture feature interactions

Learn new representations

Limit vocab. Size

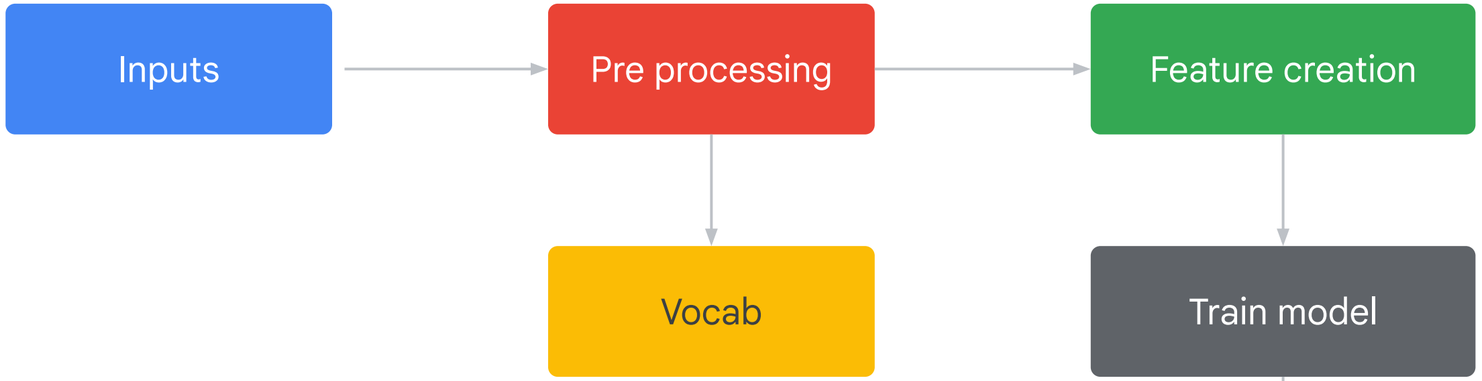
Categorical variables should be one-hot encoded

```
{
  "transactionId": 42,
  "name": "Ice Cream",
  "price": 2.50,
  "tags": ["cold", "dessert"],
  "servedBy": {
             "employeeId": 72365,
             "waitTime": 1.4,
             "customerRating": 4},
  "storeLocation": {
  "latitude":   35.3,
  "longitude": -98.7}
}
```

8345	72365	87654	98723	23451
0	1	0	0	0

```
tf.feature_column.categorical_column_with_vocabulary_list(
     'employeeId',
    Vocabulary_list = ['8345', '72365', '87654', '98723', '23451']),
```

Preprocess data to create a vocabulary of keys



Vocabulary of keys:

8345	72365	87654	98723	23451
0	1	0	0	0

The vocabulary and the mapping of the vocabulary needs to be identical at prediction time

8345	72365	87654	98723	??????
0	0	0	0	0

Options for encoding categorical data

If you know the keys beforehand:

```
tf.feature_column.categorical_column_with_vocabulary_list('employeeId',
    vocabulary_list = ['8345', '72345', '87654', '98723', '23451']),
```

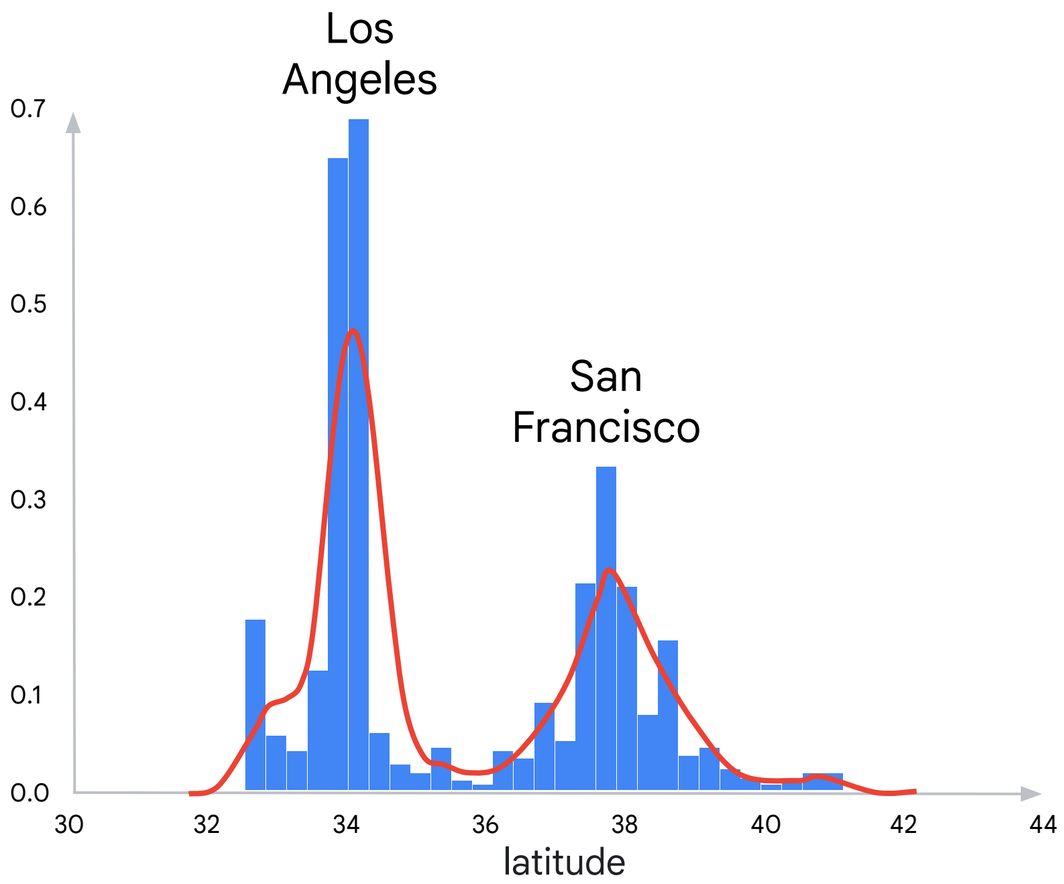
If your data is already indexed; i.e., has integers in [0-N):

```
tf.feature_column.categorical_column_with_identity('employeeId',
    num_buckets = 5)
```

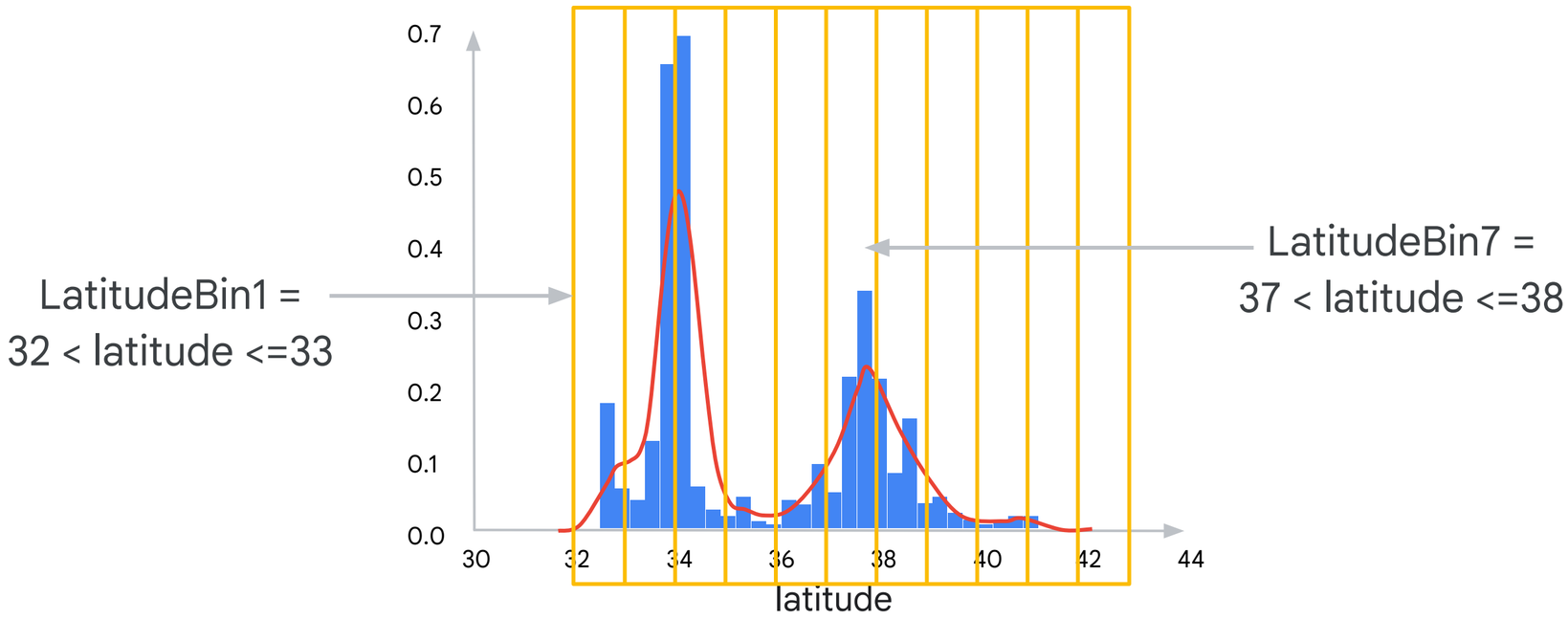
If you don’t have a vocabulary of all possible values:

```
tf.feature_column.categorical_column_with_hash_bucket('employeeId',
    hash_bucket_size = 500)
```

Exact floats are not meaningful

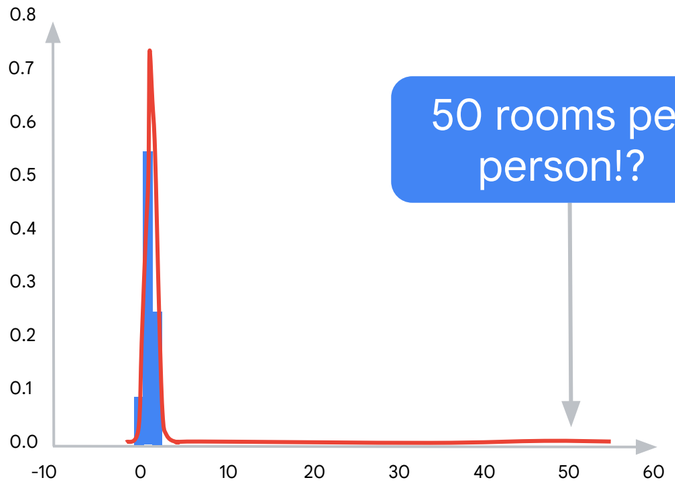


Discretize floating point values into bins

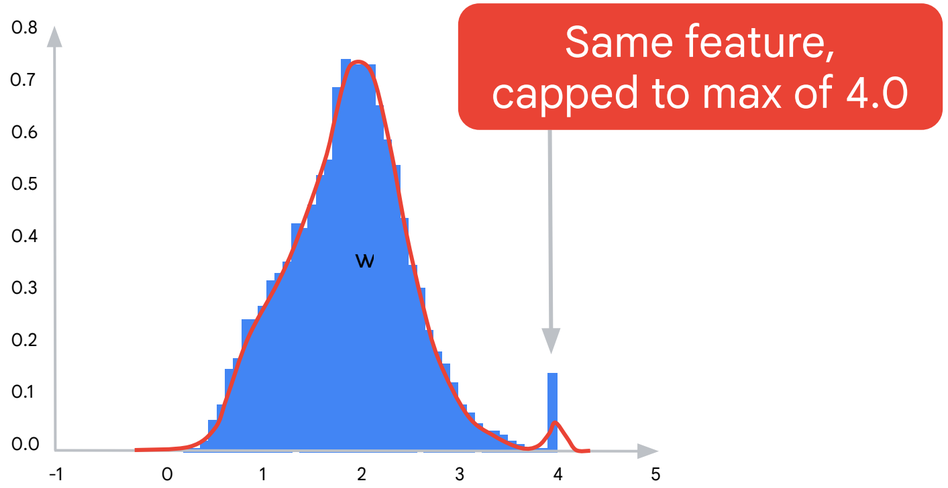



```
lat = tf.feature_column.numeric_column('latitude')
dlat = tf.feature_column.bucketized_column(
          lat, boundaries=np.arange(32,42,1).tolist()
```

Crazy outliers will hurt trainability



Rooms Per Person



Capped Rooms Per Person

```
features['capped_rooms'] = tf.clip_by_value(
    features['rooms'] ,
    clip_value_min=0,
    clip_value_max=4
)
```

BUCKETIZE: BigQuery ML

Spatial and Features

```
CONCAT(
   ML.BUCKETIZE(pickuplon, GENERATE_ARRAY(-78, -70, 0.01)),
   ML.BUCKETIZE(pickuplat, GENERATE_ARRAY(37, 45, 0.01)),
   ML.BUCKETIZE(dropofflon, GENERATE_ARRAY(-78, -70, 0.01)),
   ML.BUCKETIZE(dropofflat, GENERATE_ARRAY(37, 45, 0.01)),
) AS pickup_and_dropoff
```

```
{
 “fare_amount”: “2.5”,
 “passengers”: “1.0”,
 “euclidean”: “248.06733188206664”,
 “day_hr”: {
   “dayofweek_hourofday”: :1.0”
},
 “pickup_and_dropoff”: “bin_402bin_370bin_403bin_370”
},
```

pickup_and_dropoff
bin_406bin_379bin_406bin_379
bin_402bin_376bin_402bin_376
bin_422bin_366bin_423bin_366
bin_404bin_378bin_405bin_380
bin_403bin_374bin_403bin_374
bin_401bin_376bin_406bin_368
bin_407bin_376bin_407bin_376
bin_404bin_378bin_404bin_377
bin_402bin_376bin_403bin_376
bin_402bin_376bin_402bin_376
bin_401bin_372bin_401bin_372
bin_401bin_375bin_401bin_375
bin_404bin_380bin_404bin_380

Ideally, features should have a similar range

Typically [0,1] or [-1,1]

```
features['scaled_price'] =
      (features['price'] - min_price) /
(max_price - min_price)
```

Feature crosses

01

Feature crosses memorize!

02

The goal of ML is generalization.

03

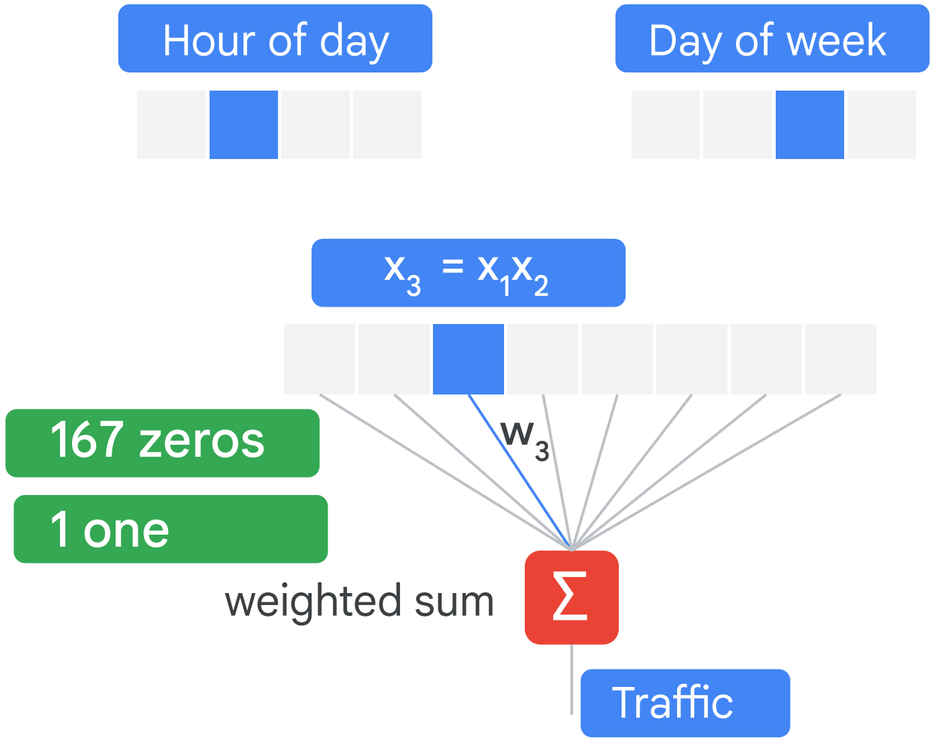
Memorization works when you have lots of data.

04

Feature crosses are powerful.

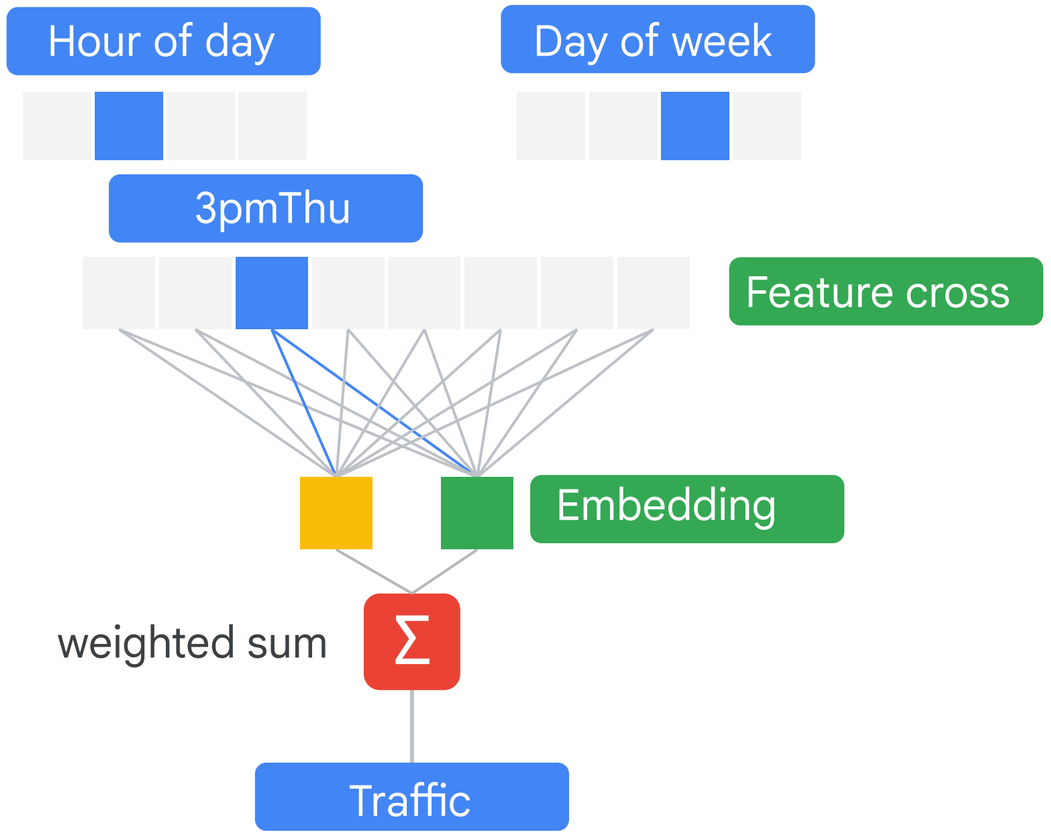
Feature crosses lead to sparsity

Allow the model to learn traffic patterns by creating a new feature that combines the time of day and day of week (this is called a feature cross).




```
traffic_fc = feature_column.crossed_column([hour_of_day, day_of_week])
```

Creating an embedding column from a feature cross




```
traffic_embedding = feature_column.embedding_column(traffic_fc, dimension=2)
```

TRANSFORM clause: BigQuery ML

```
CREATE OR REPLACE MODEL feat_eng.final_model
TRANSFORM(
    fare_amount,
        #SQRT( (pickuplon-dropofflon)*(pickuplon-dropofflon) +(pickuplat-dropofflat)*(pickuplat-dropofflat) ) AS
euclidean,
          ST_Distance(ST_GeogPoint(pickuplon, pickuplat), ST_GeogPoint(dropofflon, dropofflat)) AS euclidean,
        ML.FEATURE_CROSS(STRUCT(CAST(EXTRACT(DAYOFWEEK FROM pickup_datetime) AS STRING) AS dayofweek,
           CAST(EXTRACT(HOUR FROM pickup_datetime) AS STRING) AS hourofday)) AS day_hr,
    CONCAT(
        ML.BUCKETIZE(pickuplon, GENERATE_ARRAY(-78, -70, 0.01)),
        ML.BUCKETIZE(pickuplat, GENERATE_ARRAY(37, 45, 0.01)),
        ML.BUCKETIZE(dropofflon, GENERATE_ARRAY(-78, -70, 0.01)),
        ML.BUCKETIZE(dropofflat, GENERATE_ARRAY(37, 45, 0.01)),
    ) AS pickup_and_dropoff
)
OPTIONS(input_label_cols=[`fare_amount`], model_type=`linear_reg`, l2_reg=0.1
AS
SELECT * FROM feat_eng.feateng_training_data
```

Feature-engineered input data pipeline

- Numerical
- Categorical
- Bucketized
- Feature cross

```
num_c = ['longitude',  'latitude',  'housing_median_age', 'total_rooms', 'total_bedrooms',   'population',
'households', 'median_income']

cat_i_c = ['ocean_proximity']
for feature_name in cat_i_c:
        vocabulary = dataframe[feature_name].unique()
        cat_c = tf.feature_column.categorical_column_with_vocabulary_list(feature_name, vocabulary)
        one_hot = feature_column.indicator_column(cat_c)
        feature_columns.append(one_hot)


age_buckets = feature_column.bucketized_column(Age, boundaries=[18, 25, 30, 35, 40, 45, 50, 55, 60, 65])
feature_columns.append(age_buckets)


vocabulary = dataframe['ocean_proximity'].unique()
ocean_proximity = tf.feature_column.categorical_column_with_vocabulary_list('ocean_proximity', vocabulary)
crossed_feature = feature_column.crossed_column([age_buckets, ocean_proximity], hash_bucket_size=1000)
crossed_feature = feature_column.indicator_column(crossed_feature)
feature_columns.append(crossed_feature)
```

Build the DNN

Keras Sequential Model API

```
feature_layer = tf.keras.layers.DenseFeatures(feature_columns)
model = tf.keras.Sequential([
  feature_layer,
  layers.Dense(12,  input_dim=8, activation='relu'),
  layers.Dense(8, activation='relu'),
  layers.Dense(1, activation='linear', name='median_house_value')
])
model.compile(optimizer='adam',
              loss='mse',
              metrics=['mse'])
history = model.fit(train_ds,
          validation_data=val_ds,
          epochs=32)
```

Demo

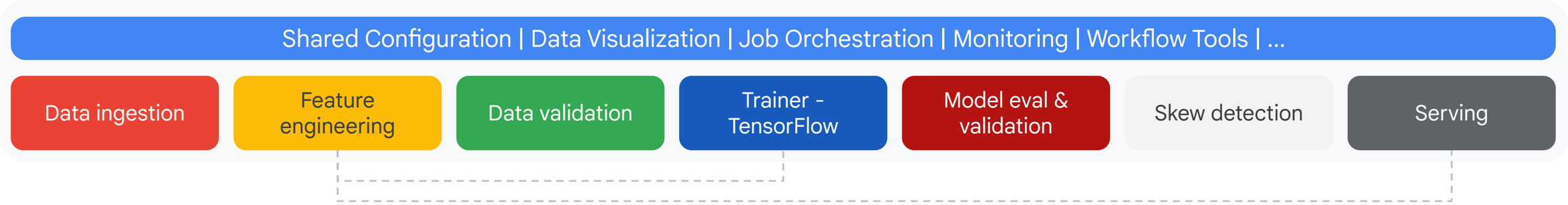
Vertex Feature Store

Feature Store in Vertex AI

TensorFlow Transform: A part of TFX

Productionizing machine learning requires more than just a learning algorithm.

TFX is an end-to-end ML platform based on TensorFlow.

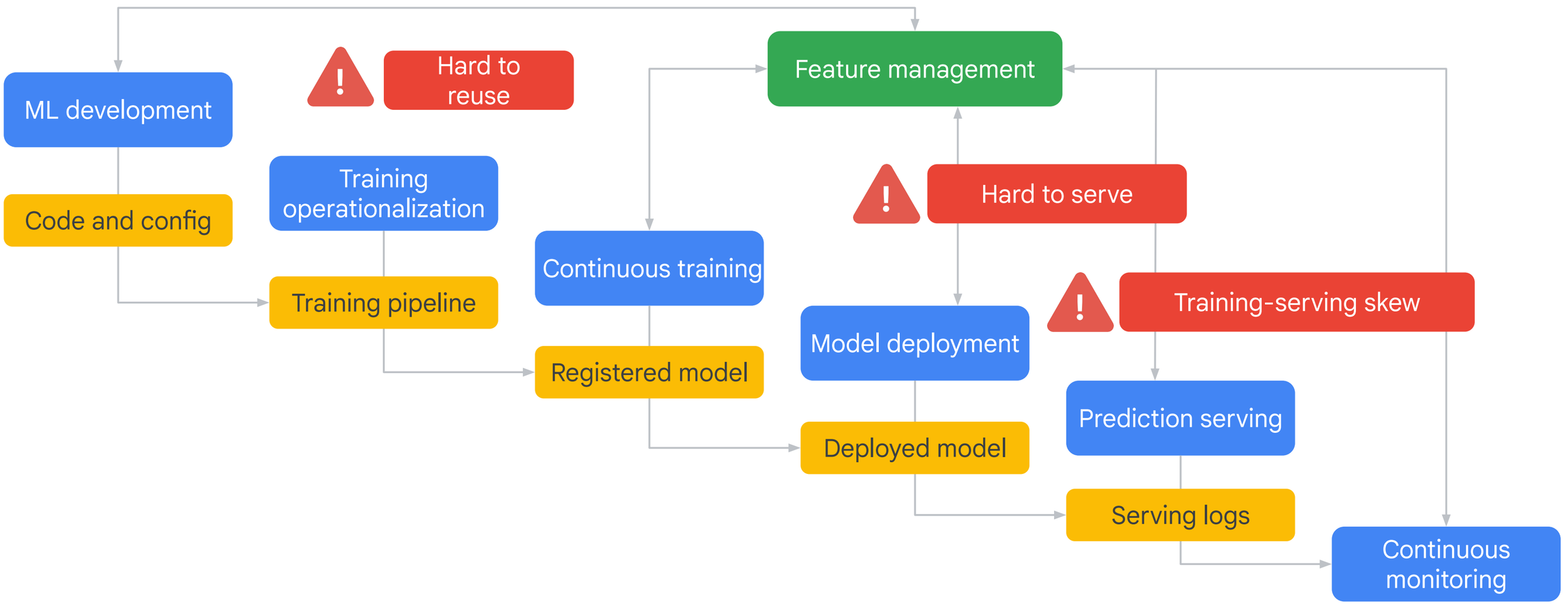


Artifacts produced by tf.Transform's are consumed at both training and serving time to avoid skew.

How do they solve these challenges?

Vertex AI Feature Store solves the feature management problems.

Feature management pain points



Fully managed Feature Store

A rich feature repository to serve, share, and re-use ML features.

Share and reuse ML features across use cases.

Centralized feature repository with easy APIs to search and discover features, fetch them for training/serving, and manage permissions.

Serve ML features at scale with low latency.

Offload the operational overhead of handling infrastructure for low-latency scalable feature serving.

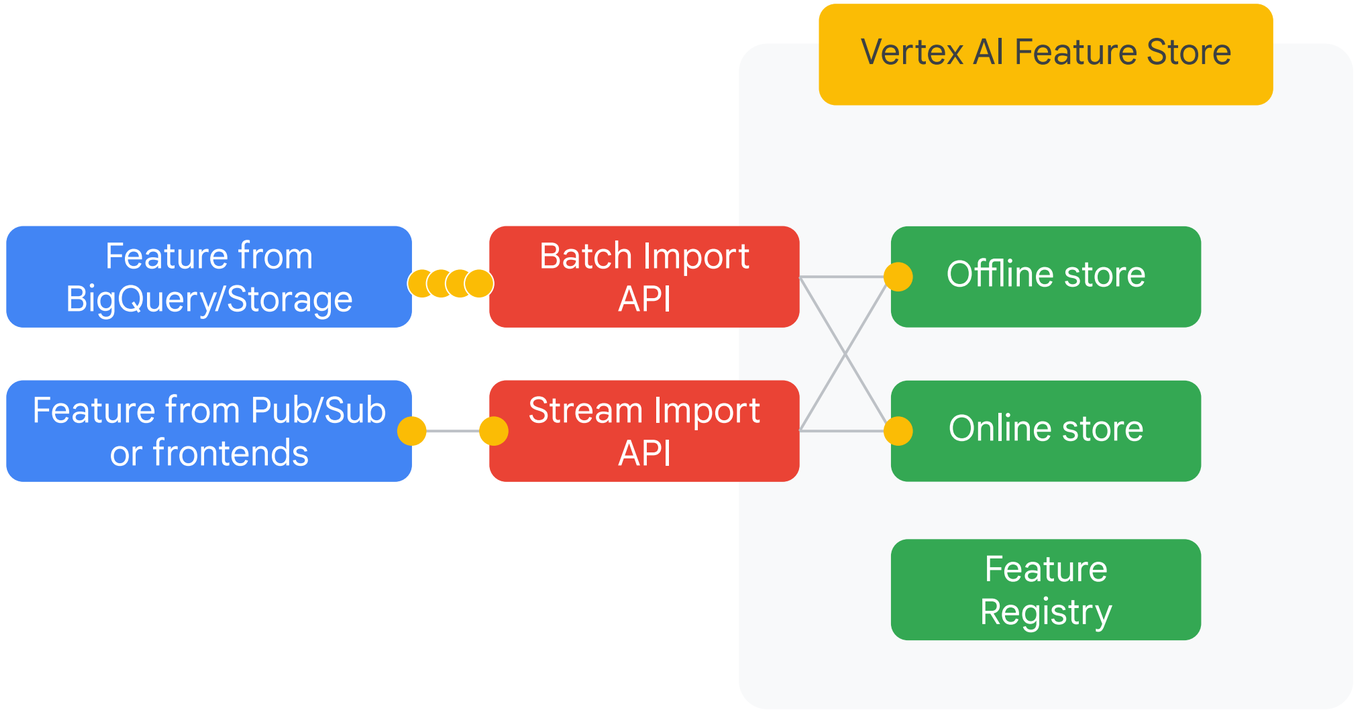
Alleviate training-serving skew.

Compute feature values once; re-use for training and serving. Track and monitor for drift and other quality issues.

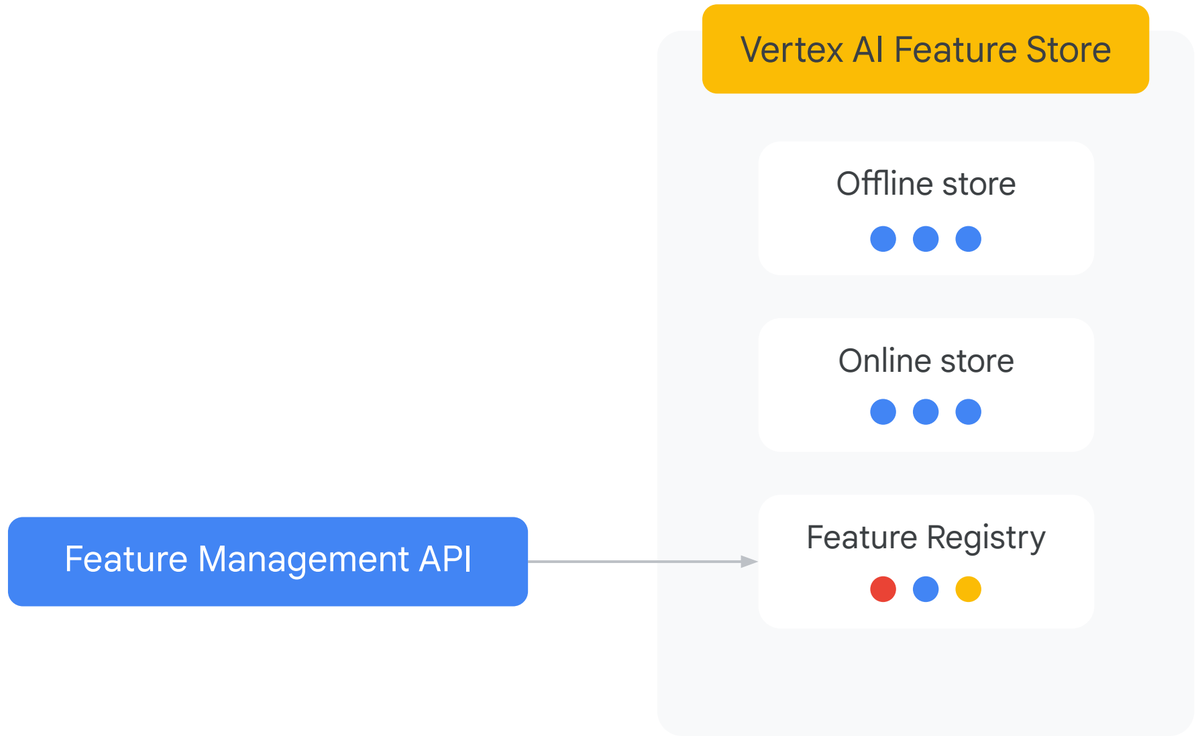
Use batch and streaming feature ingestion.

Ingest features efficiently in large batches or in real time as data streams in. ETA for streaming ingestion is Q3 2021.

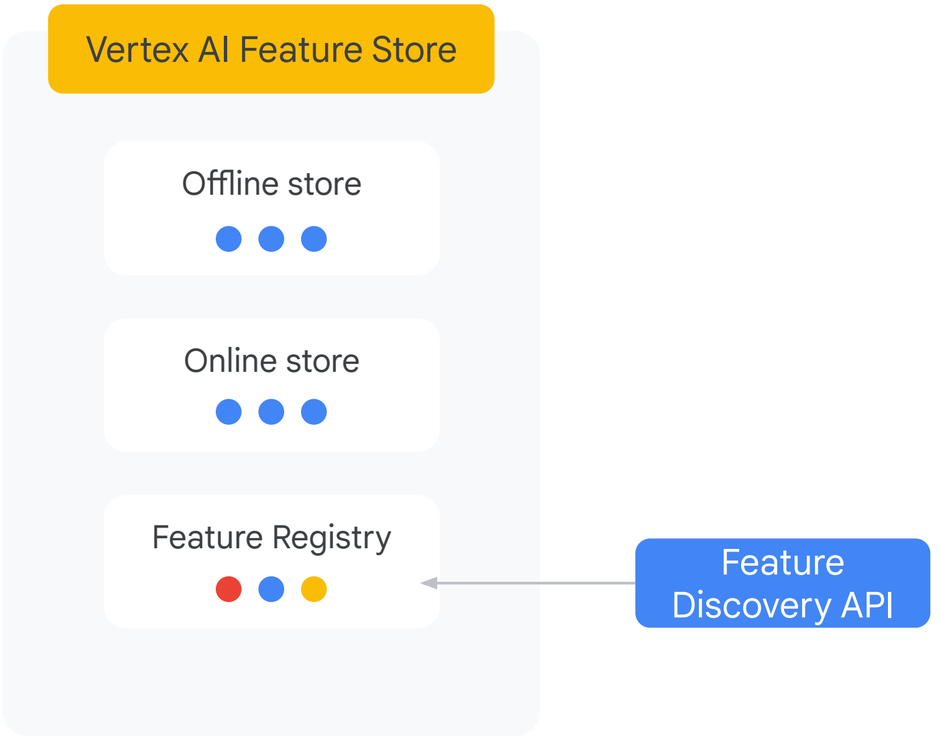
With Vertex AI Feature Store, you can store features with Batch and Stream Import APIs...



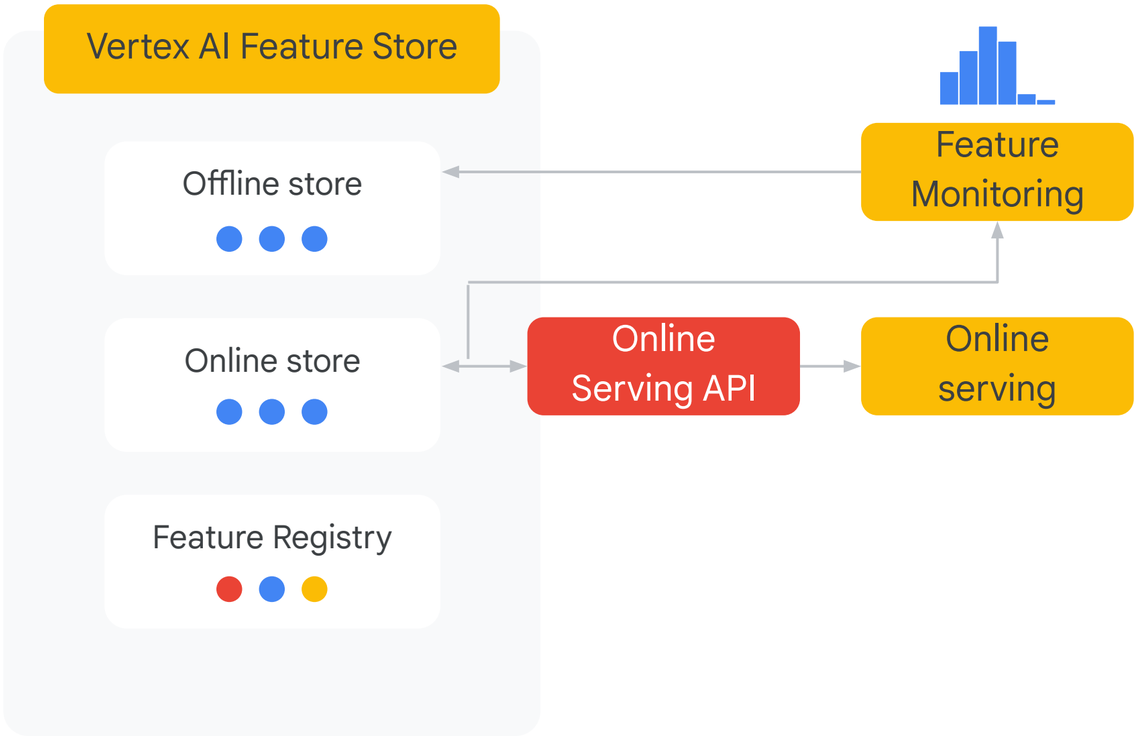
...and register the feature to its Feature Registry.



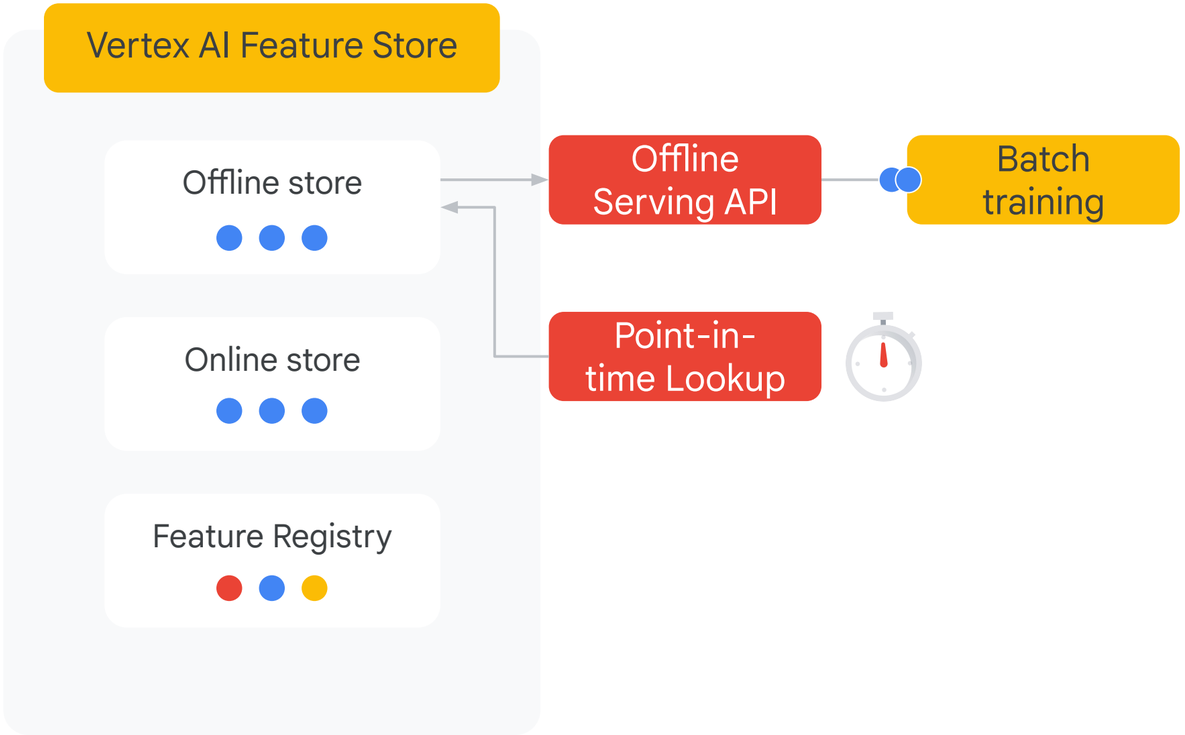
So, other researchers and ML engineers can easily find the feature with Discovery API...



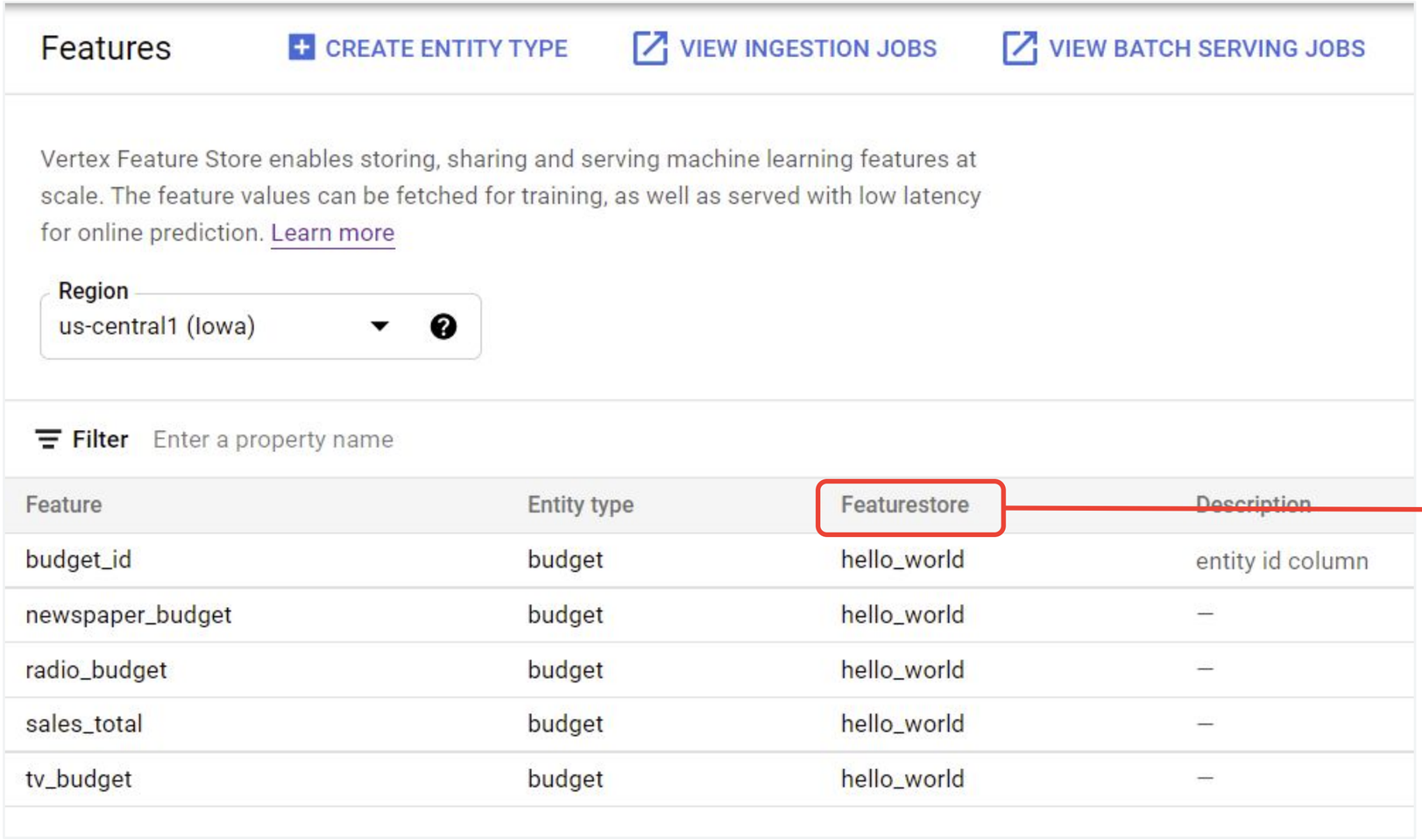
...and retrieve the feature value for fast online serving with continuous Feature Monitoring.



Also, retrieve feature batches for training jobs, with point-in-time lookup to prevent data leaks.

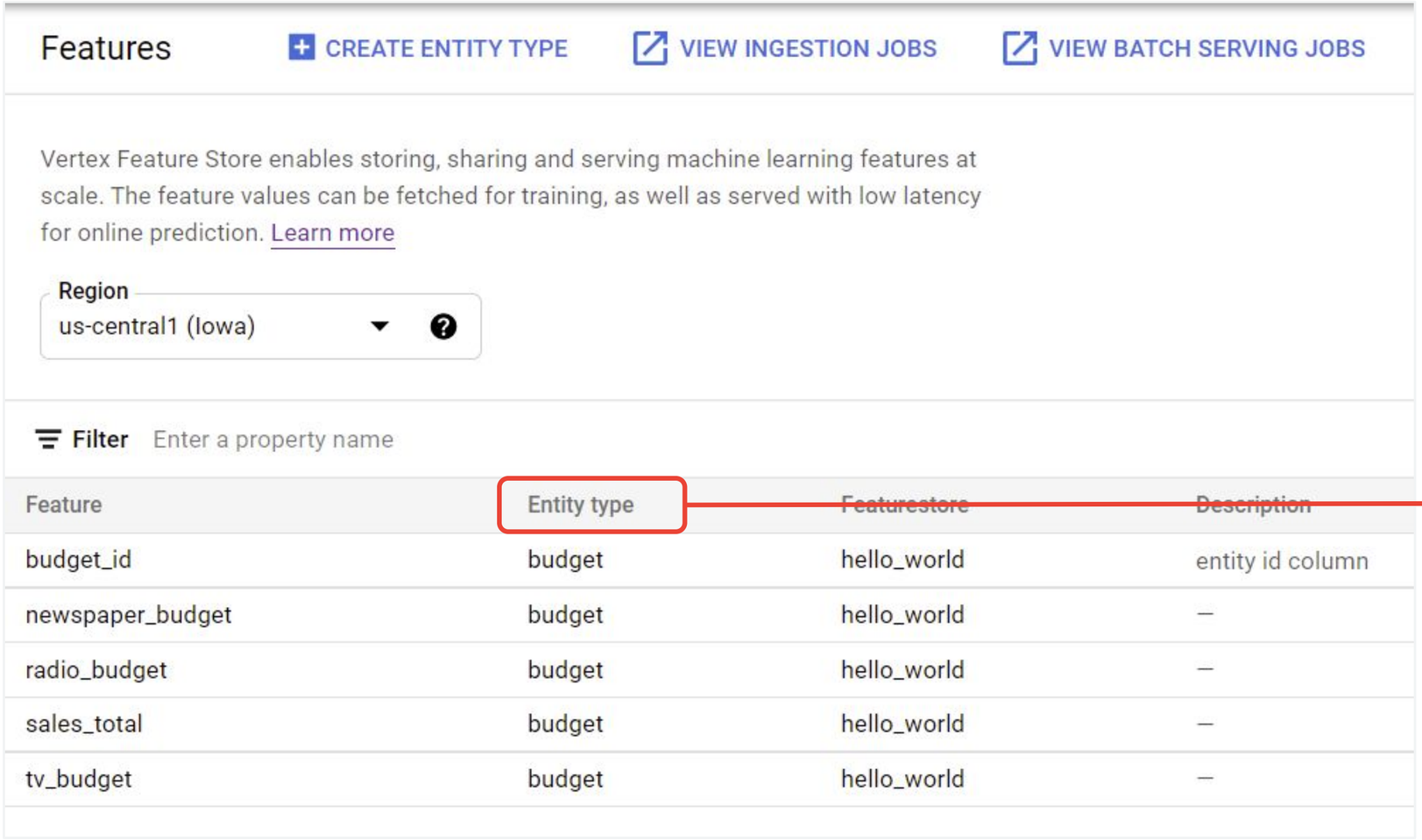


Feature Store



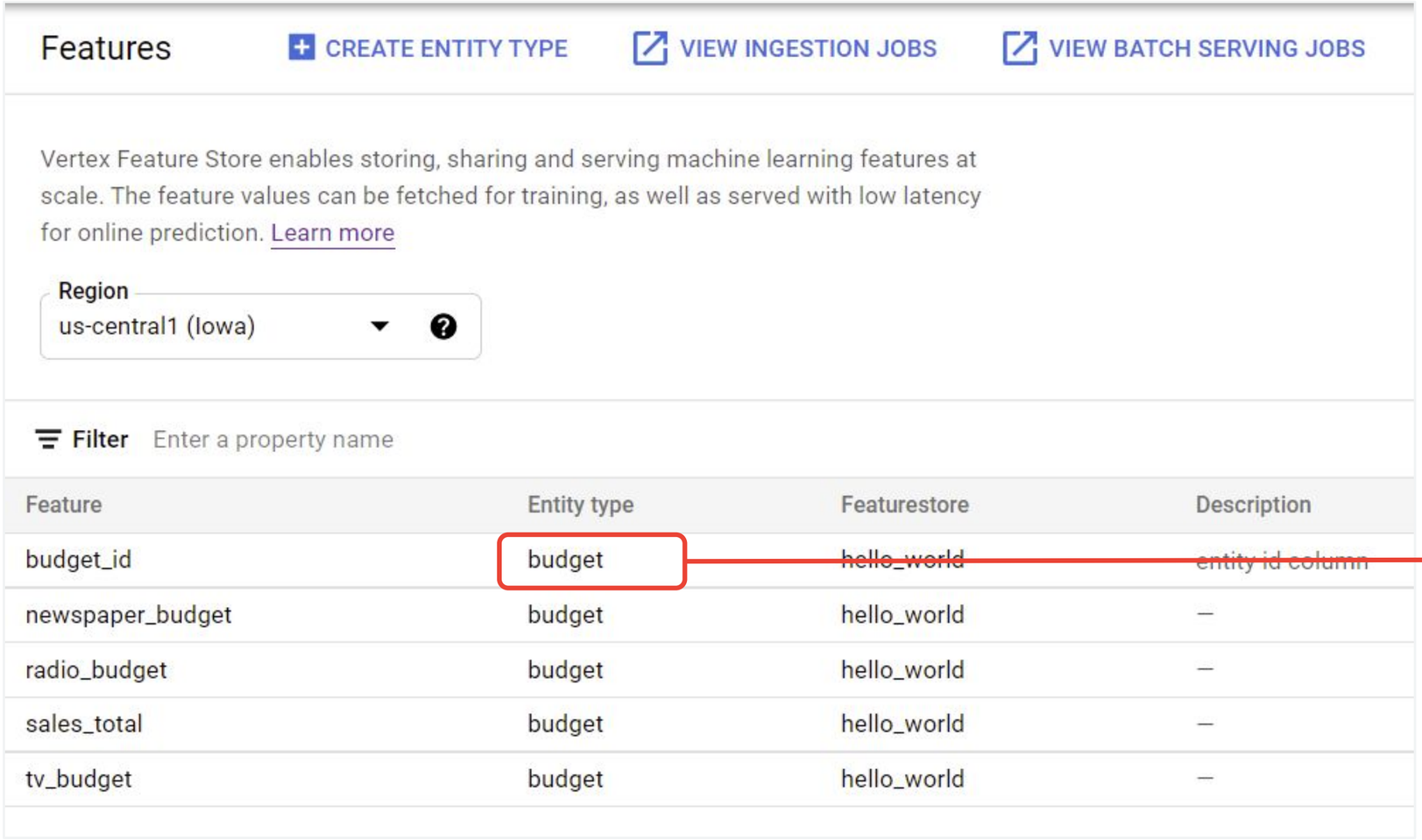
- Feature Store is a top-level container for features and their values.
- Permitted users can add and share their features.
- Users can define features and ingest values from various sources.

Entity type



- An *entity type* is a collection of semantically related features.
- You define your own entity types based on the concepts that are relevant to your use case.

Entity



- An *entity* is an instance of an entity type.
- *Movie_01* and *movie_02* are entities of the *movies* entity type.
- Each entity must have a unique ID and must be of type STRING.

Entity view

Entity type

Feature

When you retrieve values from a featurestore, the service returns an entity view that contains the feature values that you requested.

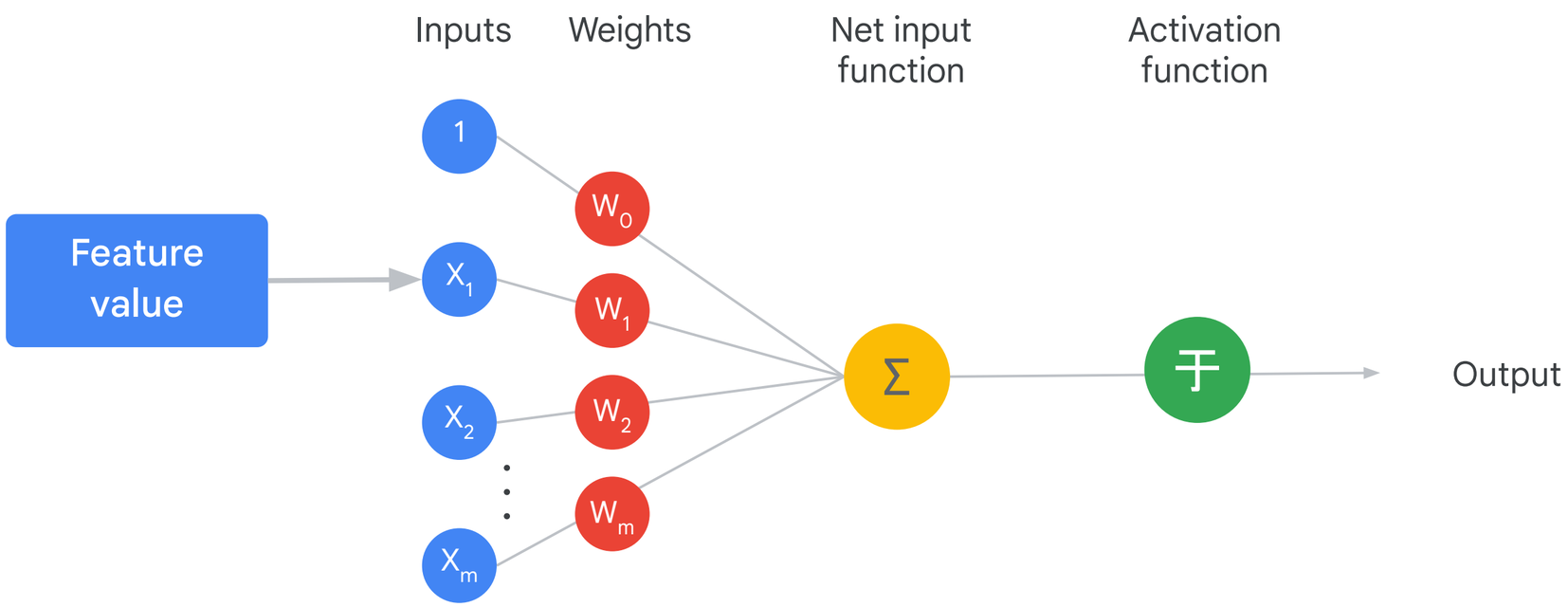
Row	movie_id	average_rating	title	genres	update_time
1	movie_01	4.9	The Shawshank Redemption	Drama	1970-01-01 00:26:19.023315 UTC
2	movie_02	4.2	The Shining	Horror	1970-01-01 00:26:19.023315 UTC
3	movie_03	4.5	Cinema Paradiso	Romance	1970-01-01 00:26:19.023315 UTC
4	movie_04	4.6	The Dark Knight	Action	1970-01-01 00:26:19.023315 UTC

Entity

Feature value

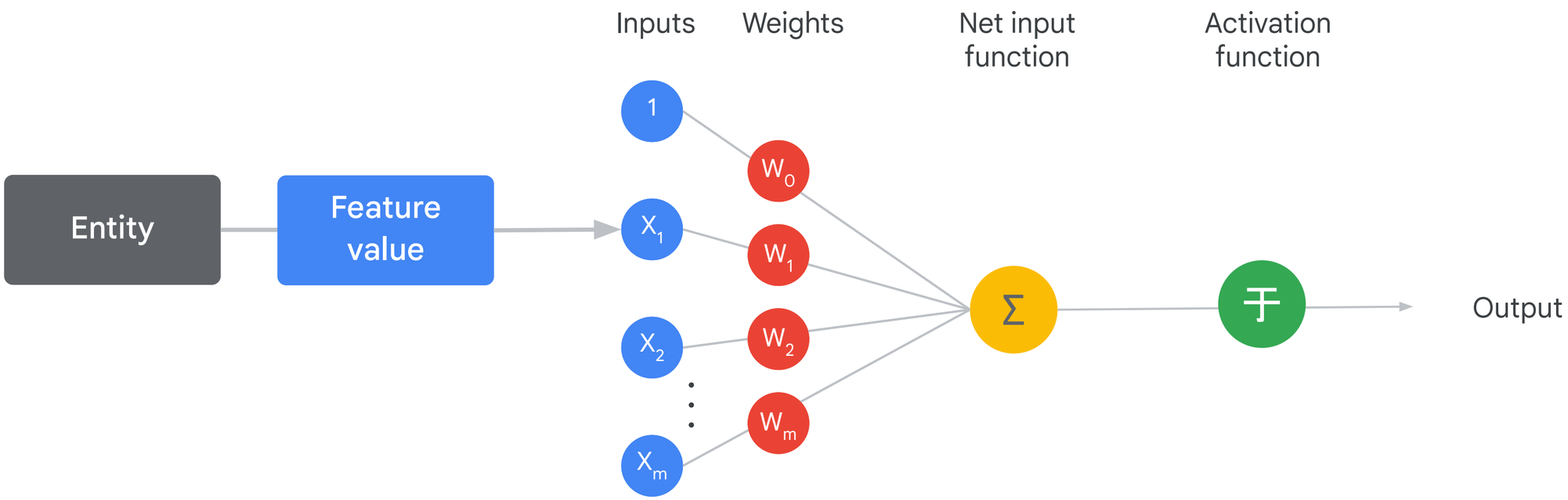
Timestamp

Summary: What is a feature?



A measurable or recordable property of an entity,
which is passed as input to a machine learning model

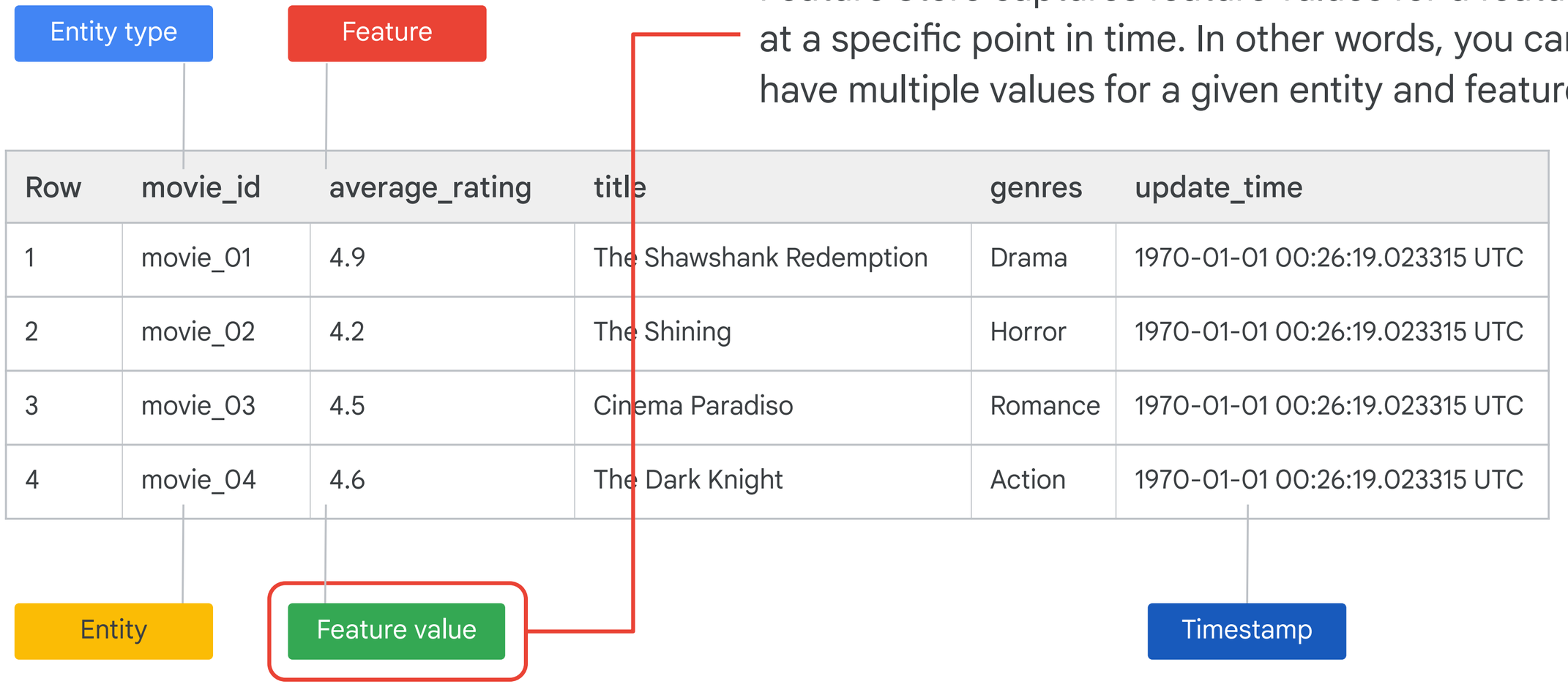
A feature describes some entity



Examples: Age of user, price of product, category of web page

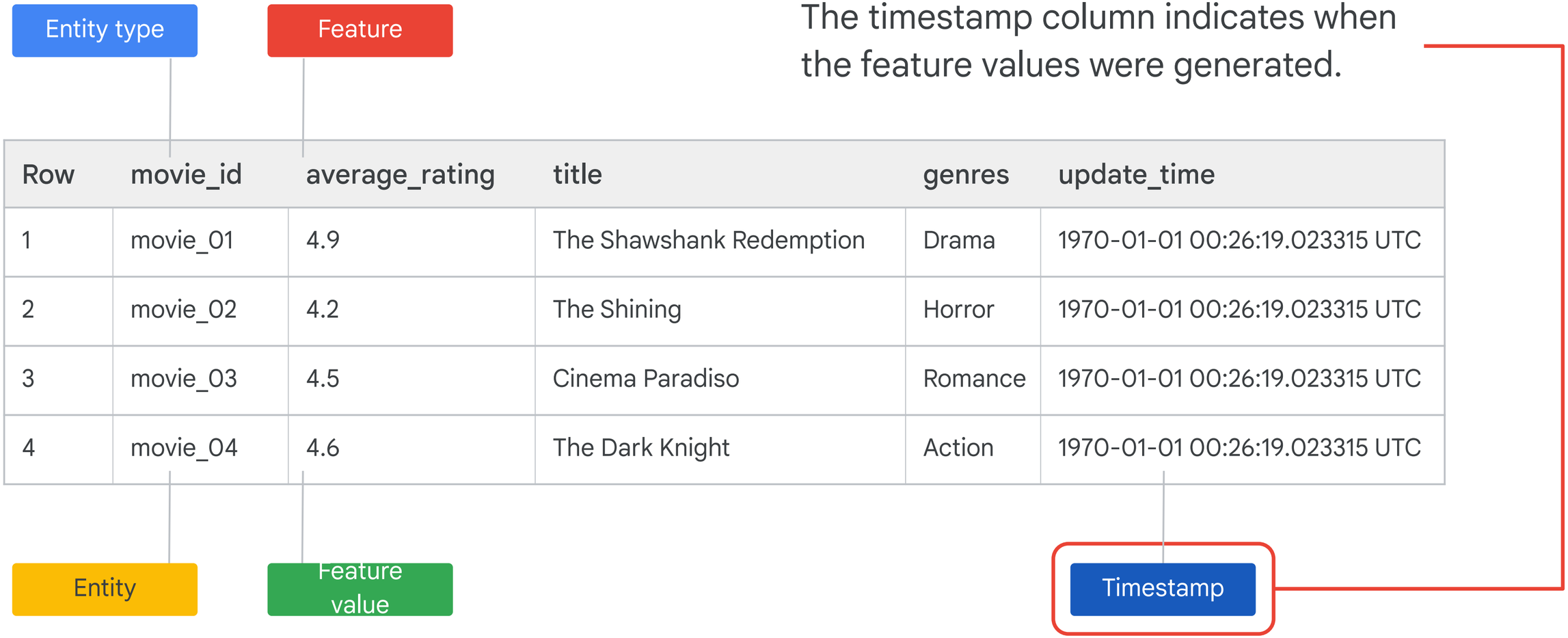
Feature value

Feature Store captures feature values for a feature at a specific point in time. In other words, you can have multiple values for a given entity and feature.



Row	movie_id	average_rating	title	genres	update_time
1	movie_01	4.9	The Shawshank Redemption	Drama	1970-01-01 00:26:19.023315 UTC
2	movie_02	4.2	The Shining	Horror	1970-01-01 00:26:19.023315 UTC
3	movie_03	4.5	Cinema Paradiso	Romance	1970-01-01 00:26:19.023315 UTC
4	movie_04	4.6	The Dark Knight	Action	1970-01-01 00:26:19.023315 UTC

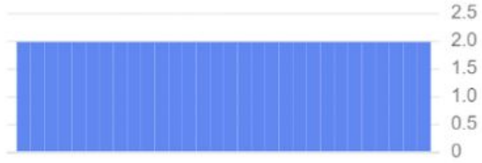
Timestamp



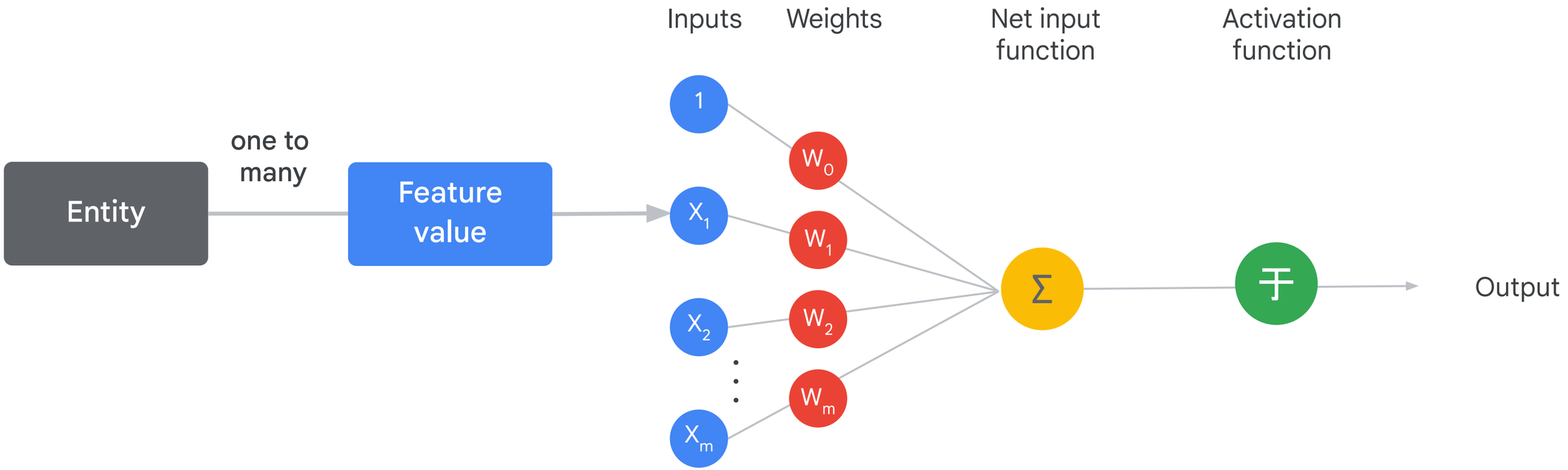
Row	movie_id	average_rating	title	genres	update_time
1	movie_01	4.9	The Shawshank Redemption	Drama	1970-01-01 00:26:19.023315 UTC
2	movie_02	4.2	The Shining	Horror	1970-01-01 00:26:19.023315 UTC
3	movie_03	4.5	Cinema Paradiso	Romance	1970-01-01 00:26:19.023315 UTC
4	movie_04	4.6	The Dark Knight	Action	1970-01-01 00:26:19.023315 UTC

Summary: Feature list

Filter Enter a property name

Feature	Type	Count	Missing	Mean	Std. dev.	Zeros	Values	Most common	Unique	Distribution
budget_id	STRING	1,200	0	—	—	—	—	"budget_999" Frequency: 2	1,200	
newspaper_budget	INT64	1,200	0	30.1	21.715	12	Min: 0 Median: 25 Max: 114	—	—	
radio_budget	INT64	1,200	0	22.79	14.789	24	Min: 0 Median: 23 Max: 49	—	—	
sales_total	INT64	1,200	0	13.54	5.24	0	Min: 1 Median: 12 Max: 27	—	—	
tv_budget	INT64	1,200	0	146.57	85.61	6	Min: 0 Median: 149 Max: 296	—	—	

Summary: A one-to-many relationship between an entity and a feature

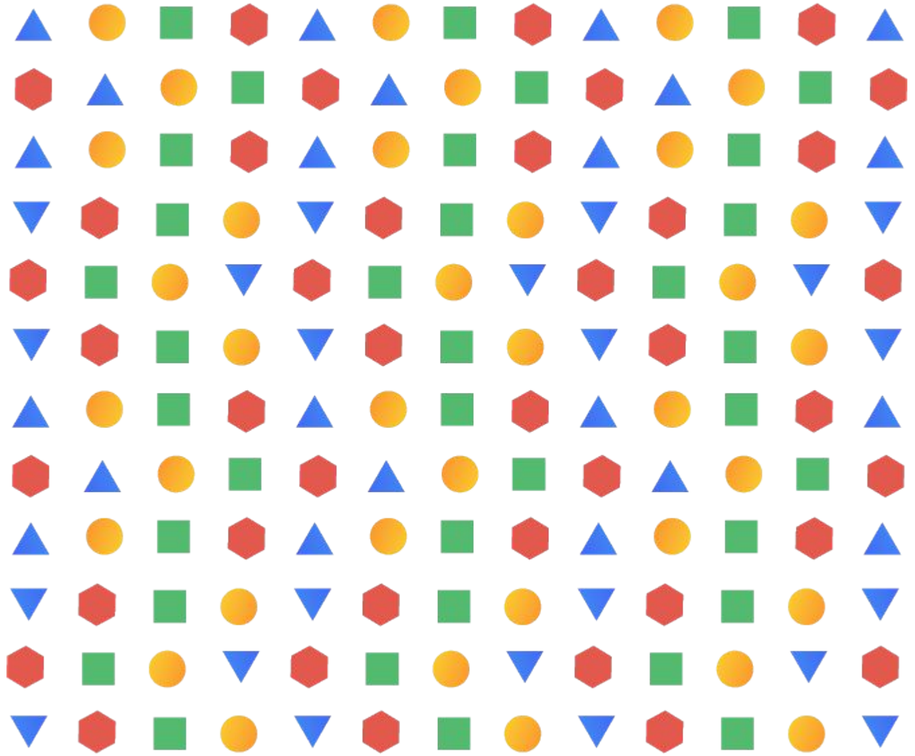


Summary: Feature Store data model

Row	movie_id	average_rating	title	genres	update_time
1	movie_01	4.9	The Shawshank Redemption	Drama	1970-01-01 00:26:19.023315 UTC
2	movie_02	4.2	The Shining	Horror	1970-01-01 00:26:19.023315 UTC
3	movie_03	4.5	Cinema Paradiso	Romance	1970-01-01 00:26:19.023315 UTC
4	movie_04	4.6	The Dark Knight	Action	1970-01-01 00:26:19.023315 UTC

Prerequisites

Data is pre-processed and feature values are clean and tidy: no missing values, correct data type, one-hot encoding of categorical values already done.

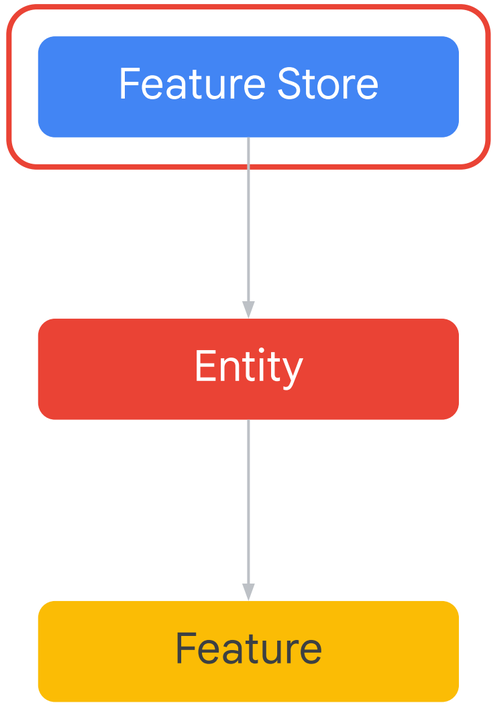


Source data requirements

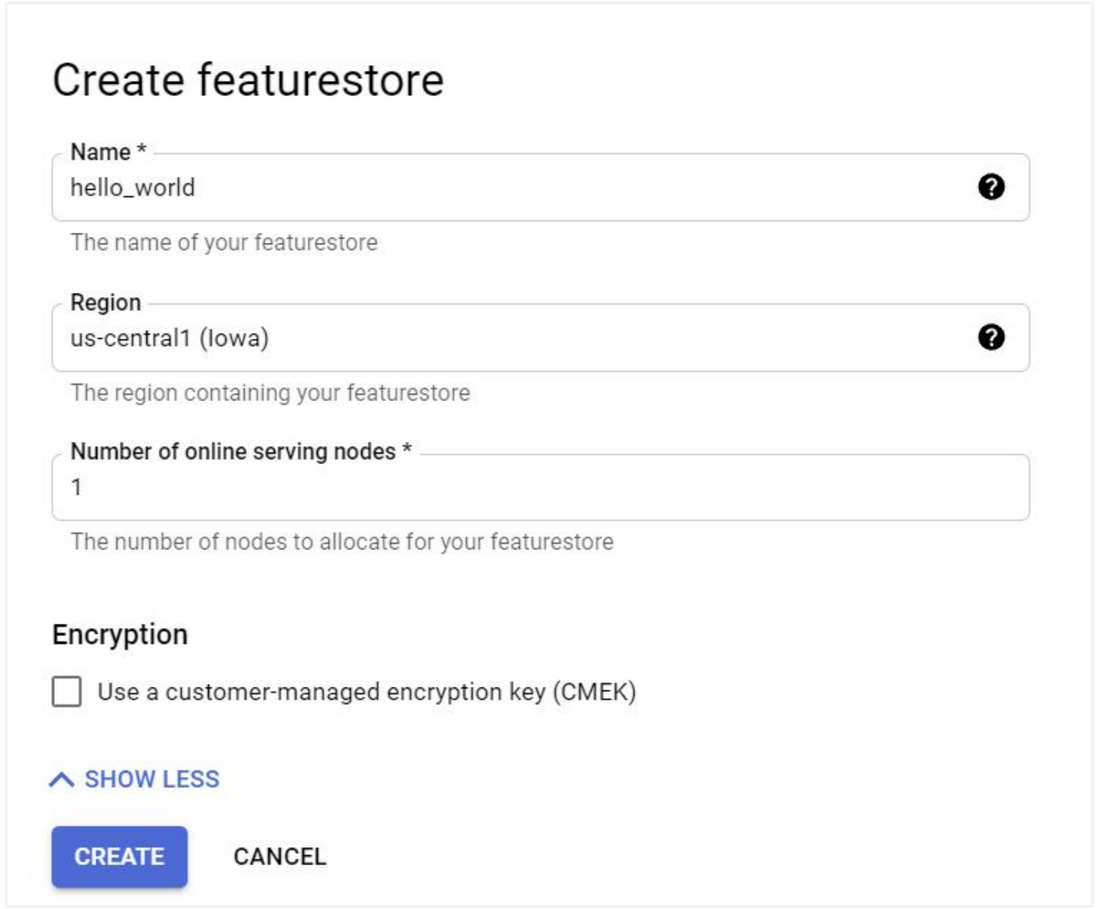
- Include a column for entity IDs, and the values must be of type STRING.
- Your source data value types must match the value types of the destination feature in the featurestore.
- All columns must have a header that is of type STRING. There are no restrictions on the names of the headers. The column header depends on your file type:
  - BigQuery: Column name
  - Avro: Defined by the Avro schema that is associated with the binary data
  - CSV files: First row
- If you provide a column for feature generation timestamps, use right format for your file type:
  - BigQuery tables: TIMESTAMP column.
  - Avro: Type long and logical type timestamp-micros.
  - CSV files: RFC 3339 format.
- CSV files cannot include array data types; use Avro or BigQuery instead.
- For array types, you cannot include a null value in the array, although you can include an empty array.

Step 1.

Create a featurestore: Walkthrough

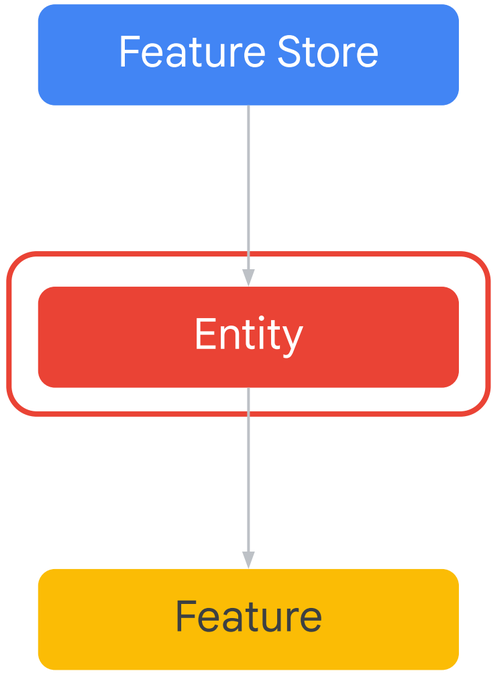


Create featurestore



Step 2.

Create an entity type



Create entity type

Entity types group and contain related features. For example, a “movies” entity type might contain features like “title” and “genre”. Learn more

Region
us-central1 (Iowa)

Featurestore *
hello_world

Entity type name *
budget_id

Must start with a letter or underscore. Can use letters, numbers, and underscores.

Description

Optional text description of the entity type

Feature monitoring PREVIEW

Feature monitoring PREVIEW

Provides descriptive statistics and distribution shapes. Enables feature monitoring for all features in the entity type. You can also edit feature monitoring at the feature level, which will override this setting.

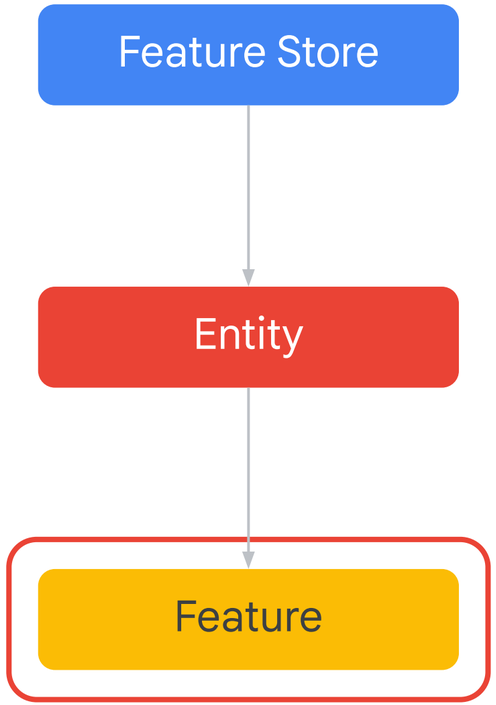
Disabled

Monitoring time interval
1 days

CREATE CANCEL

Step 3.

Add features



Add features

A feature is a measurable attribute of an entity type. After you add features in your entity type, you can then associate your features with values stored in BigQuery or Cloud Storage. Learn more

Feature names can include lowercase letters, numbers and underscores and must start with a lowercase letter or underscore

Feature name *	Value type *	Description	Override monitoring values	Feature monitoring	Interval *
Feature name 1 *	Value type 1 *	Description 1	Override	Disabled	Interval 1: 1 days

+ ADD ANOTHER FEATURE

SAVE CANCEL

Edit entity type info

Entity types group and contain related features. For example, a “movies” entity type might contain features like “title” and “genre”. Learn more

Entity type name *
budget

Must start with a letter or underscore. Can use letters, numbers, and underscores.

Description
media budgets

Optional text description of the entity type

Feature monitoring PREVIEW

Provides descriptive statistics and distribution shapes. Enables feature monitoring for all features in the entity type. You can also edit feature monitoring at the feature level, which will override this setting.

Enabled

Monitoring time interval
1 days

UPDATE CANCEL

FEATURES ENTITY TYPE PROPERTIES

Basic info

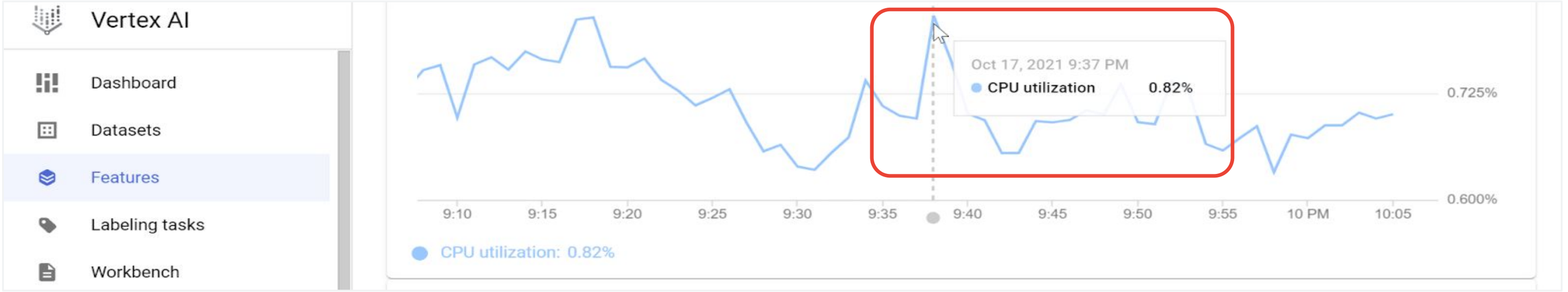
Name	budget
Region	us-central1
Featurestore	hello_world
Created	Oct 8, 2021, 3:00:55 PM
Updated	Oct 17, 2021, 10:01:07 PM
Description	media budgets

Feature monitoring PREVIEW

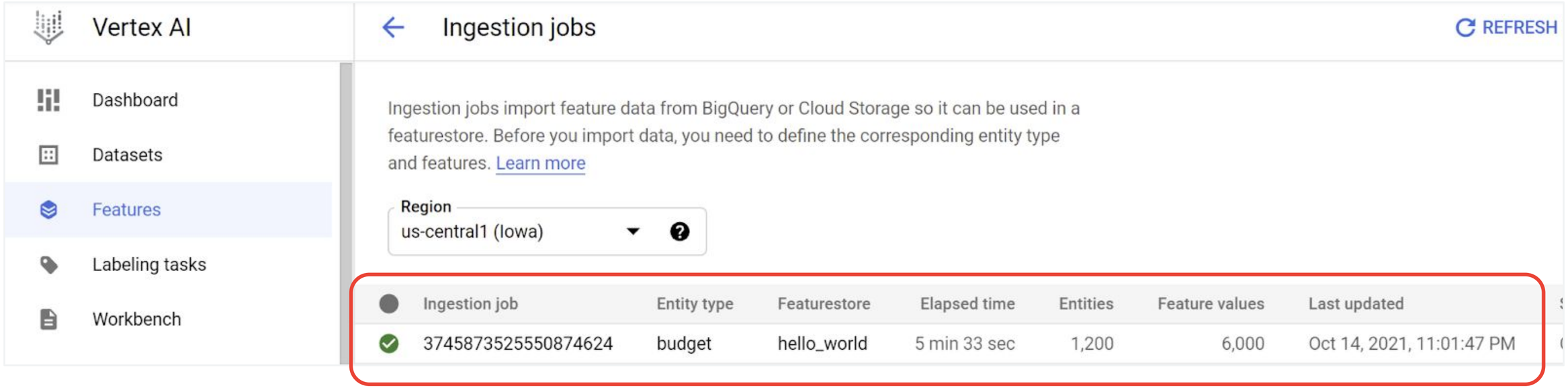
Status	Enabled
Time interval	1 day

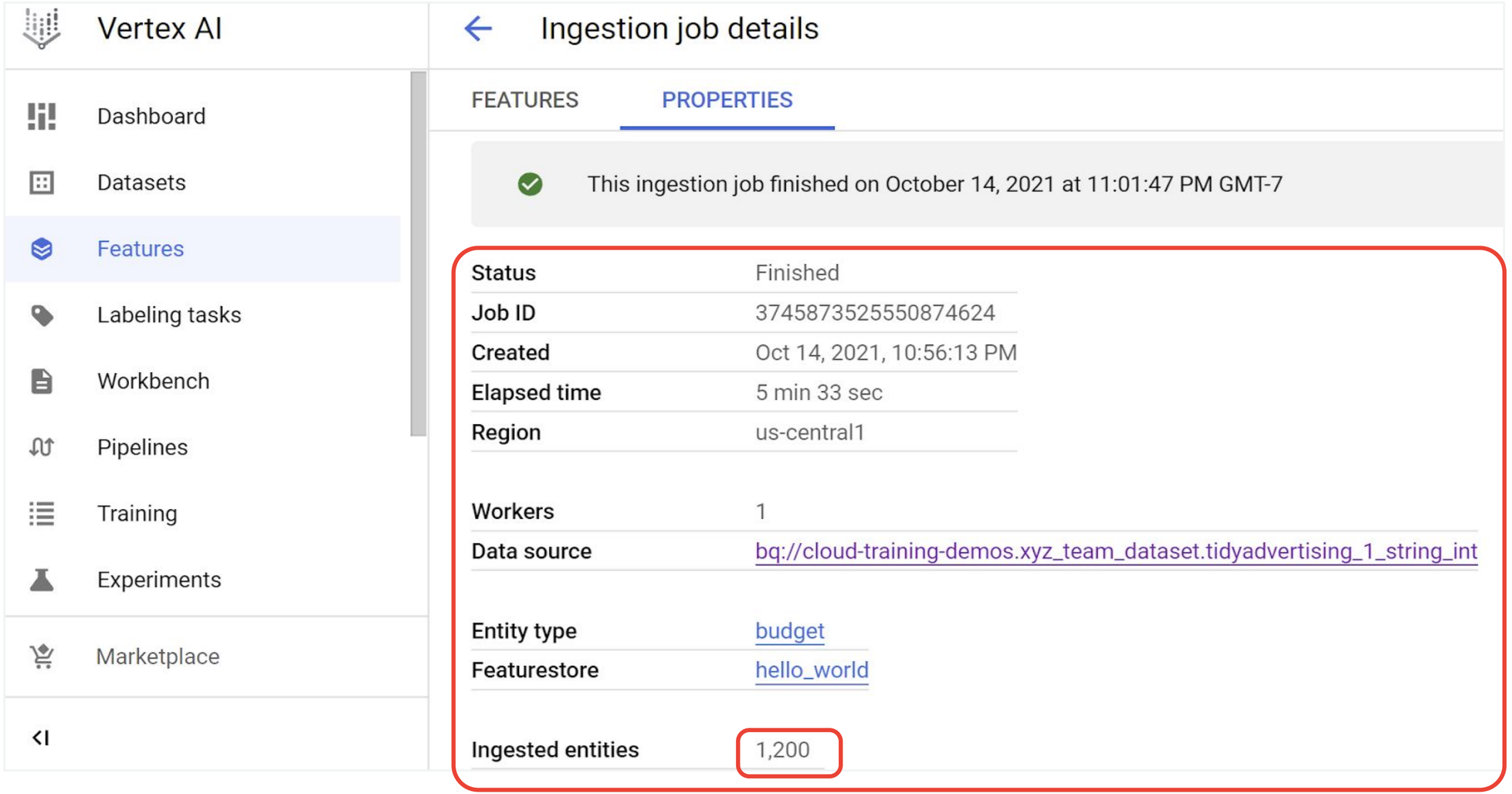
Feature value type distribution

BOOL	0 (0%)	INT64_ARRAY	0 (0%)
BOOL_ARRAY	0 (0%)	STRING	1 (20%)
DOUBLE	0 (0%)	STRING_ARRAY	0 (0%)
DOUBLE_ARRAY	0 (0%)	BYTES	0 (0%)
INT64	4 (80%)		



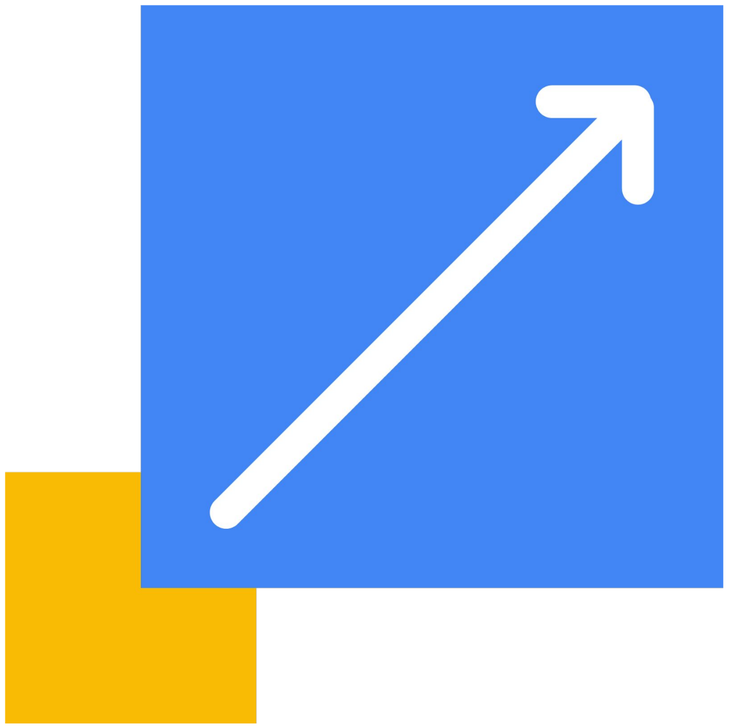
Ingestion jobs



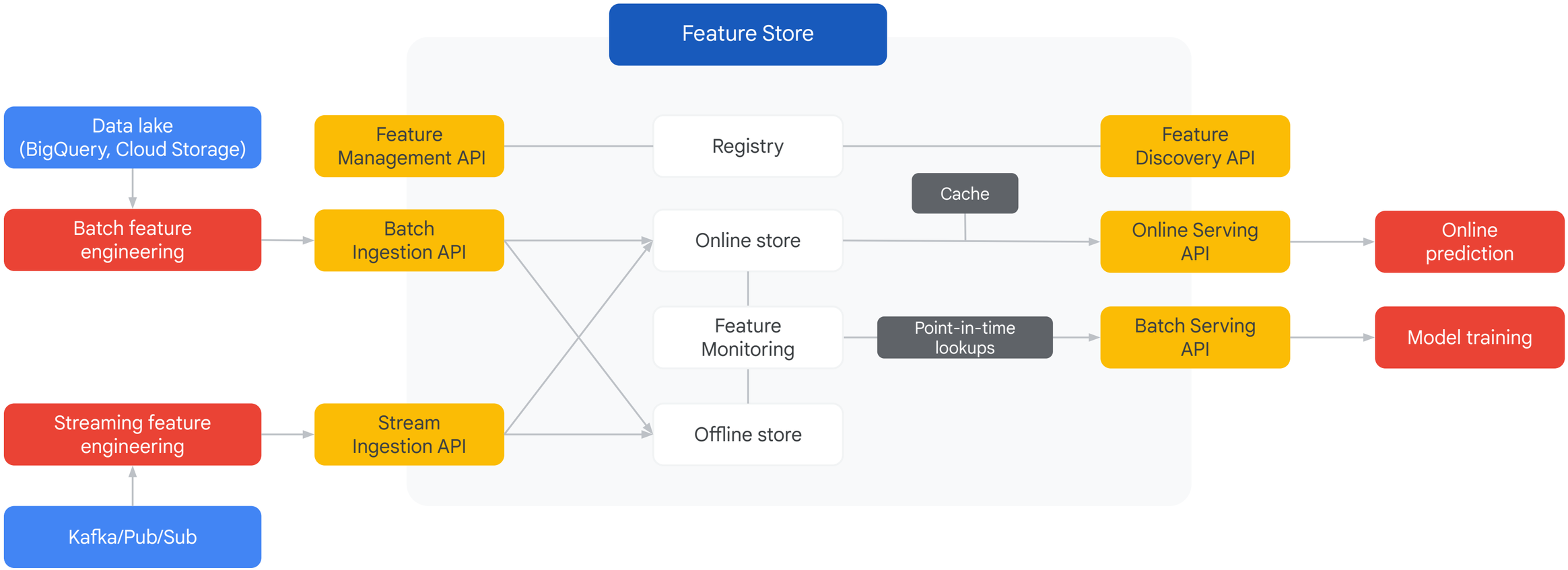


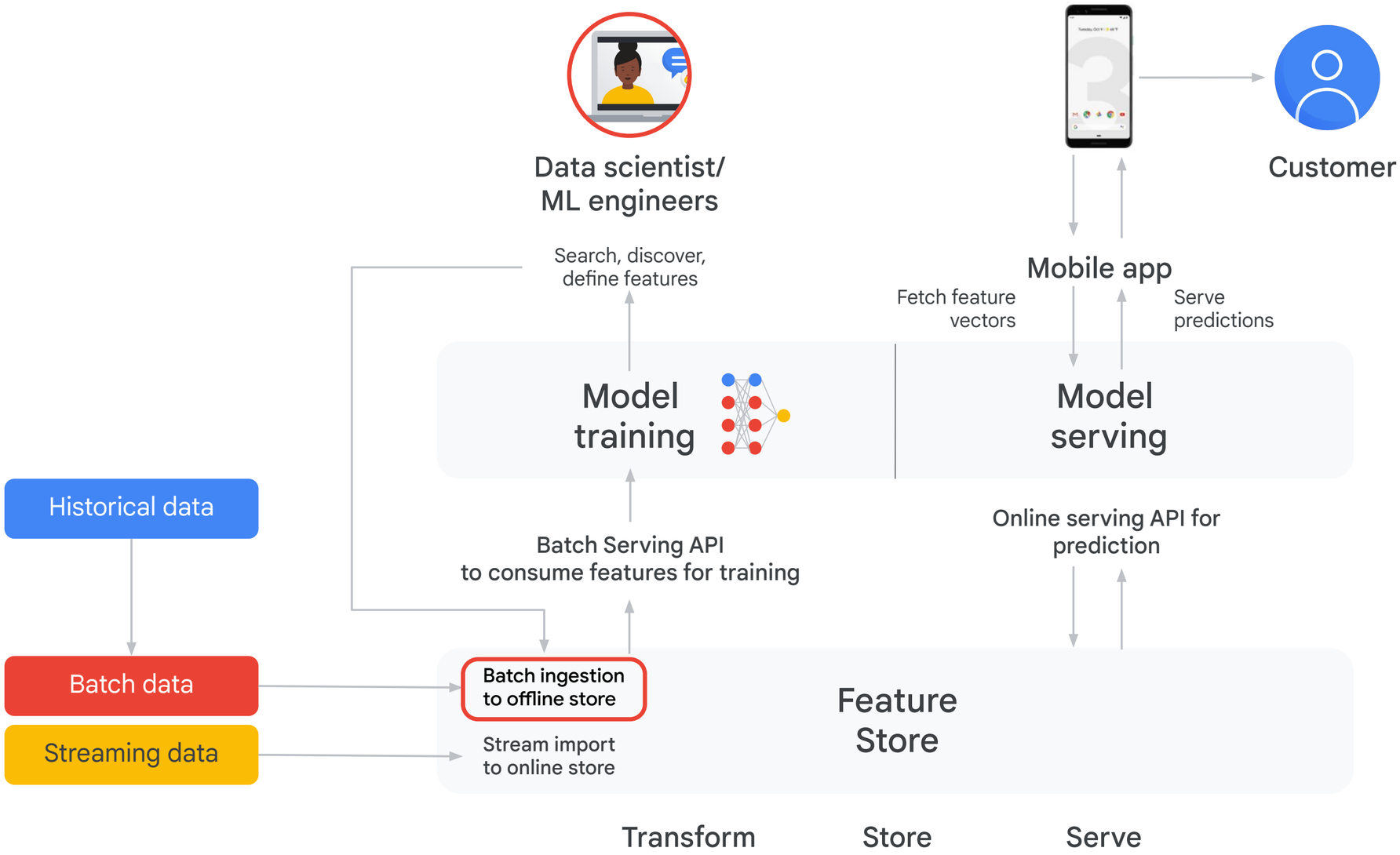
Feature serving

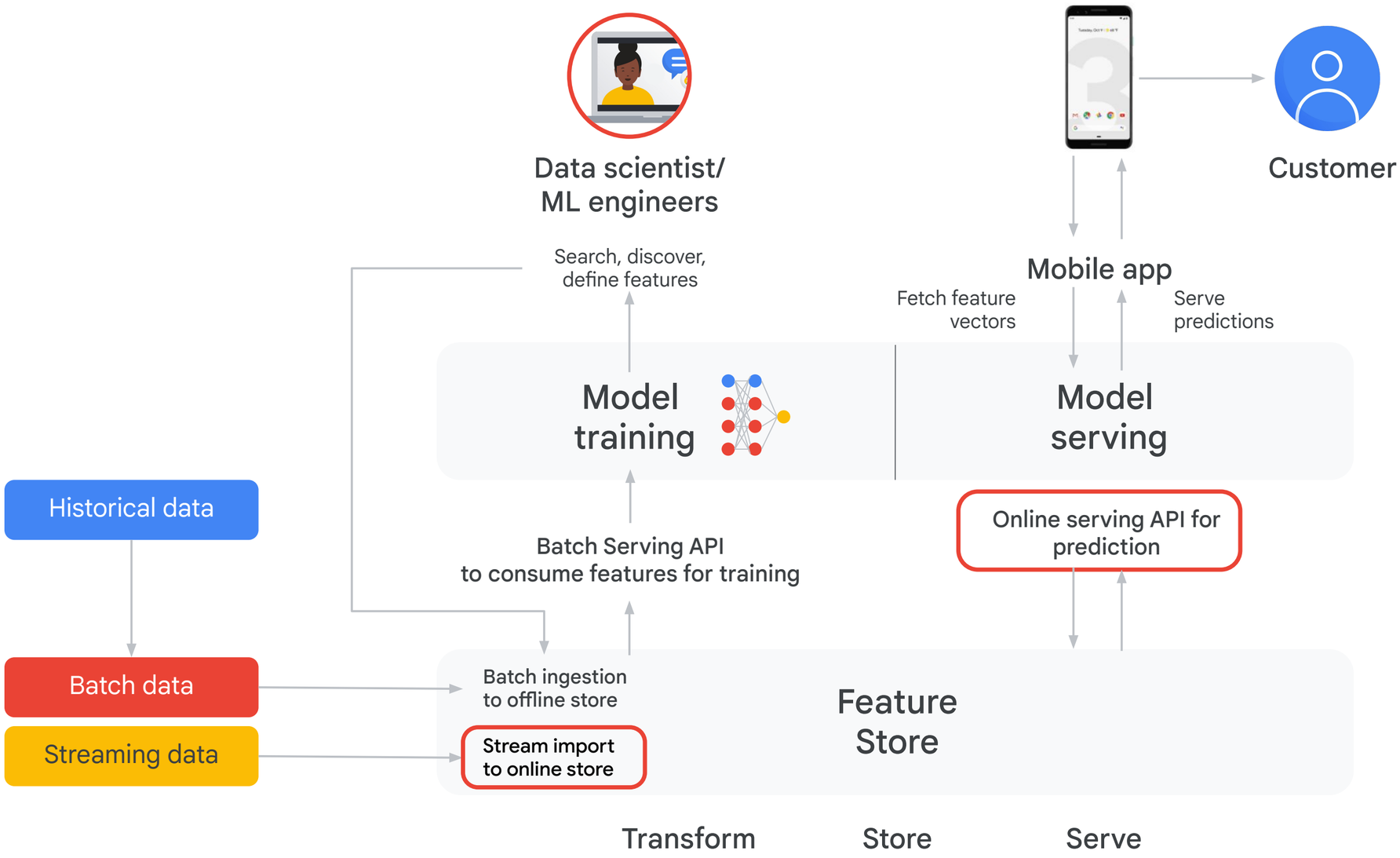
- *Feature serving* is the process of exporting stored feature values for training or inference.
- Feature Store offers two methods for serving features: batch and online.



Simple data scientist—friendly APIs and SDKs abstract away the underlying complexity







Recommended Additional Activities

Production Machine Learning Systems

Questions and answers

BigQuery ML: Introduction to BigQuery ML

Instructor: Ben Ahmed

LAB Course: BigQuery for Machine Learning:

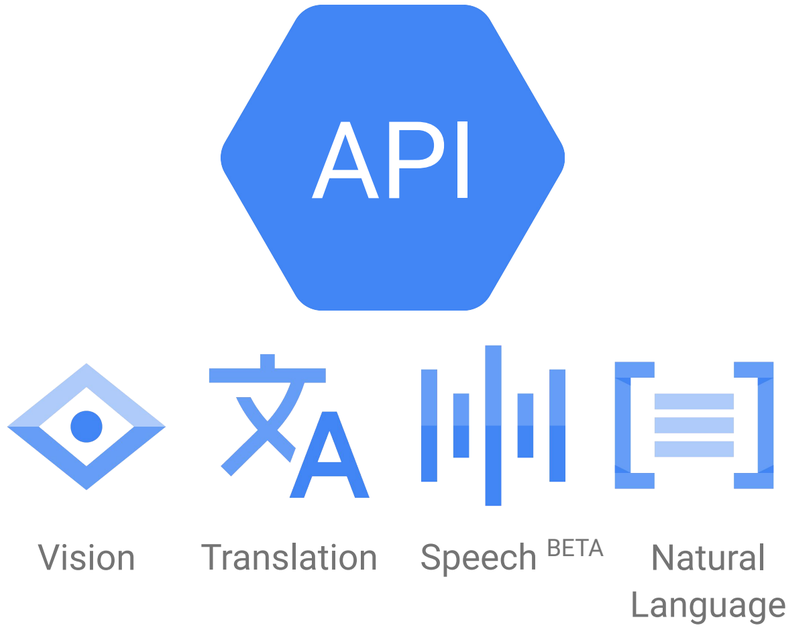
LAB: Getting Started with BigQuery Machine Learning

LAB: Machine Learning with Google BigQuery ML

The following course materials are copyright protected materials.

They may not be reproduced or distributed and may only be used by students attending this Google Cloud Partner Learning Services program.

BigQuery ML is a way to build custom models



Pre-trained models

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

Build your own model

Developers
Data Analysts
ML engineers

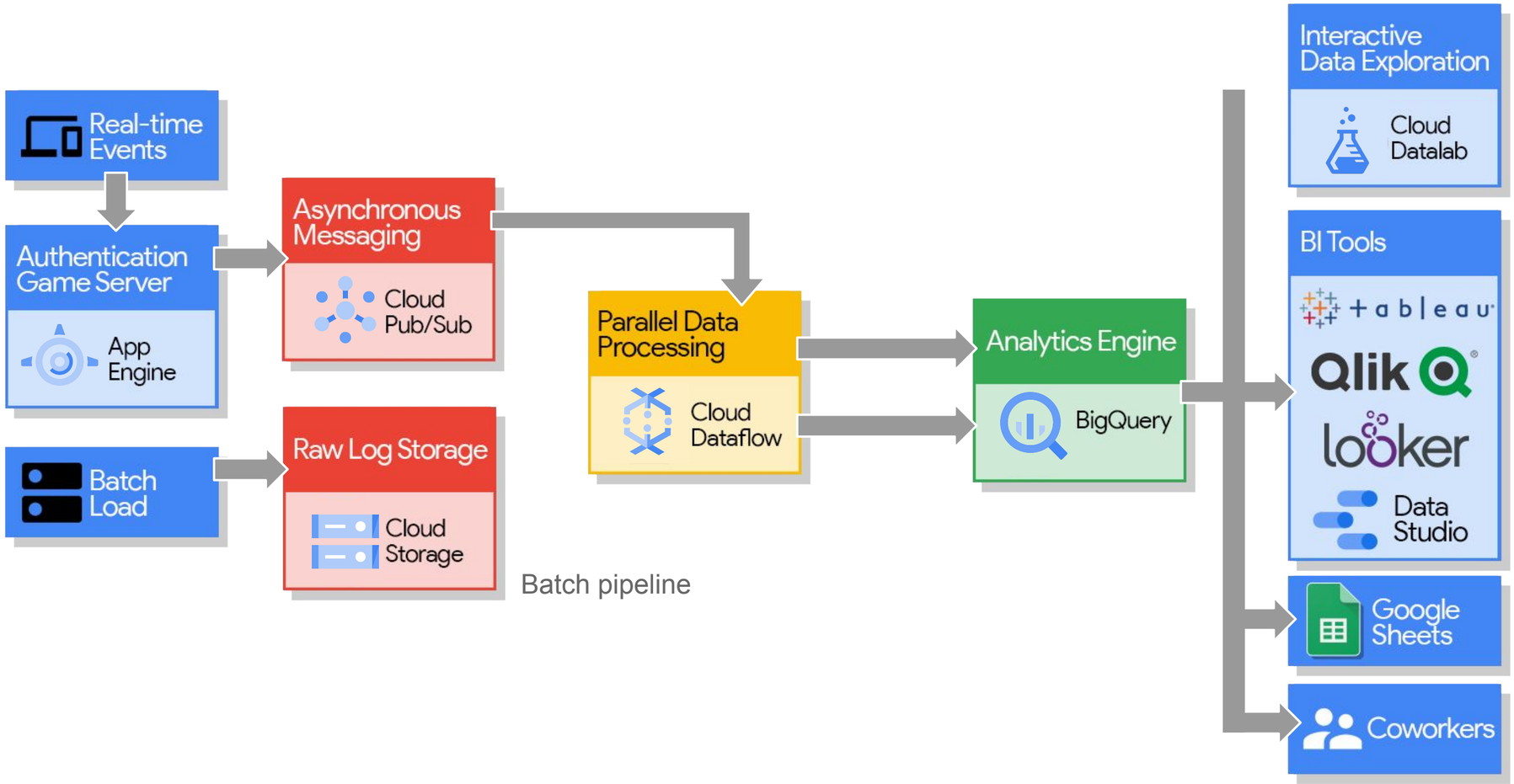
Google Cloud

BigQuery is a petabyte-scale fully-managed data warehouse

1. It’s serverless
2. Flexible pricing model
3. Data encryption and security
4. Geospatial data types and functions
5. Foundation for BI and AI

BigQuery

Typical BigQuery data warehouse architecture

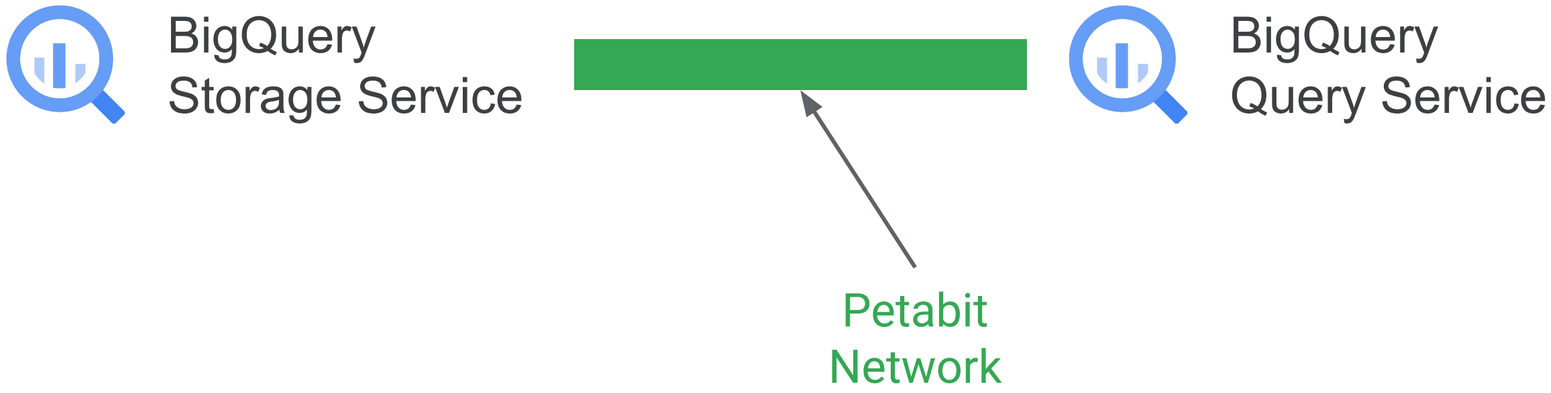


BigQuery is two services in one

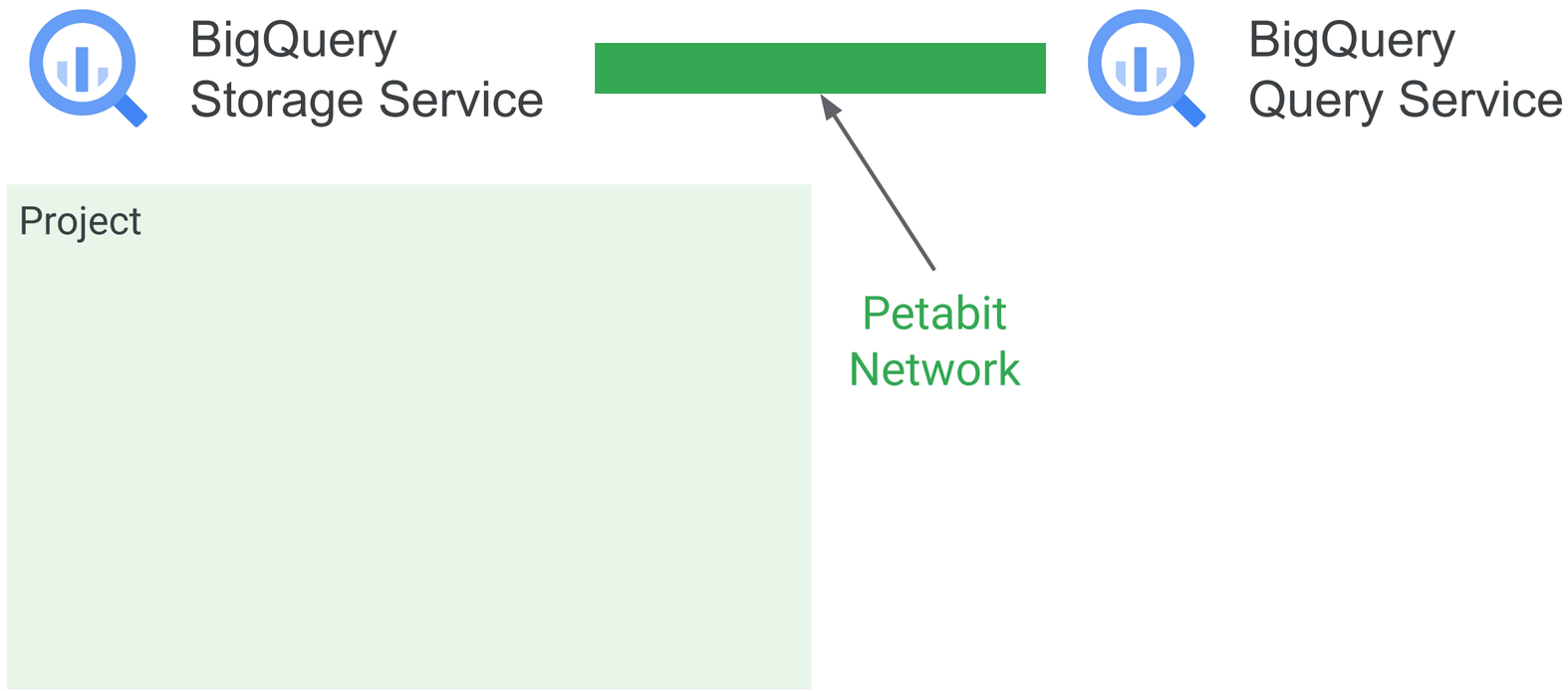
BigQuery

1. **Fast SQL Query Engine**
2. Managed storage for datasets

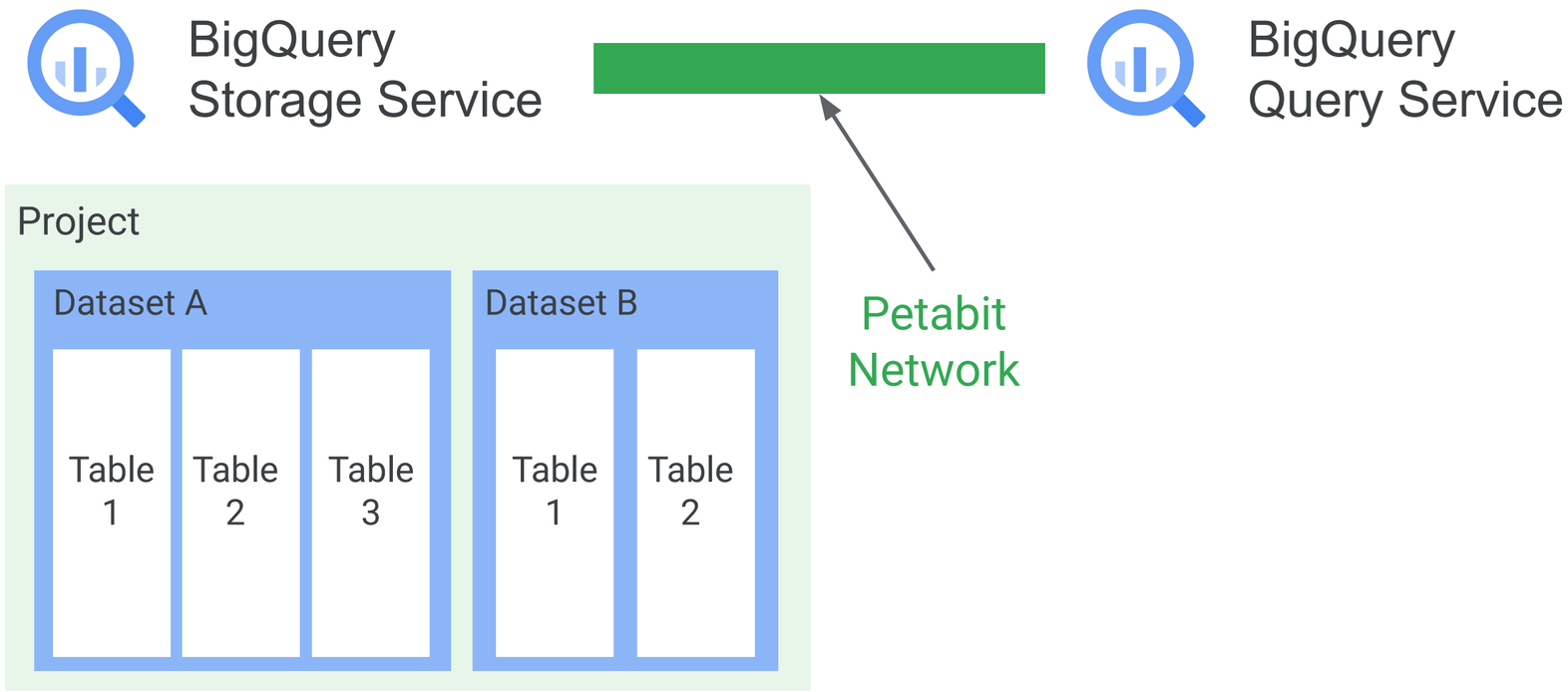
How does BigQuery work?



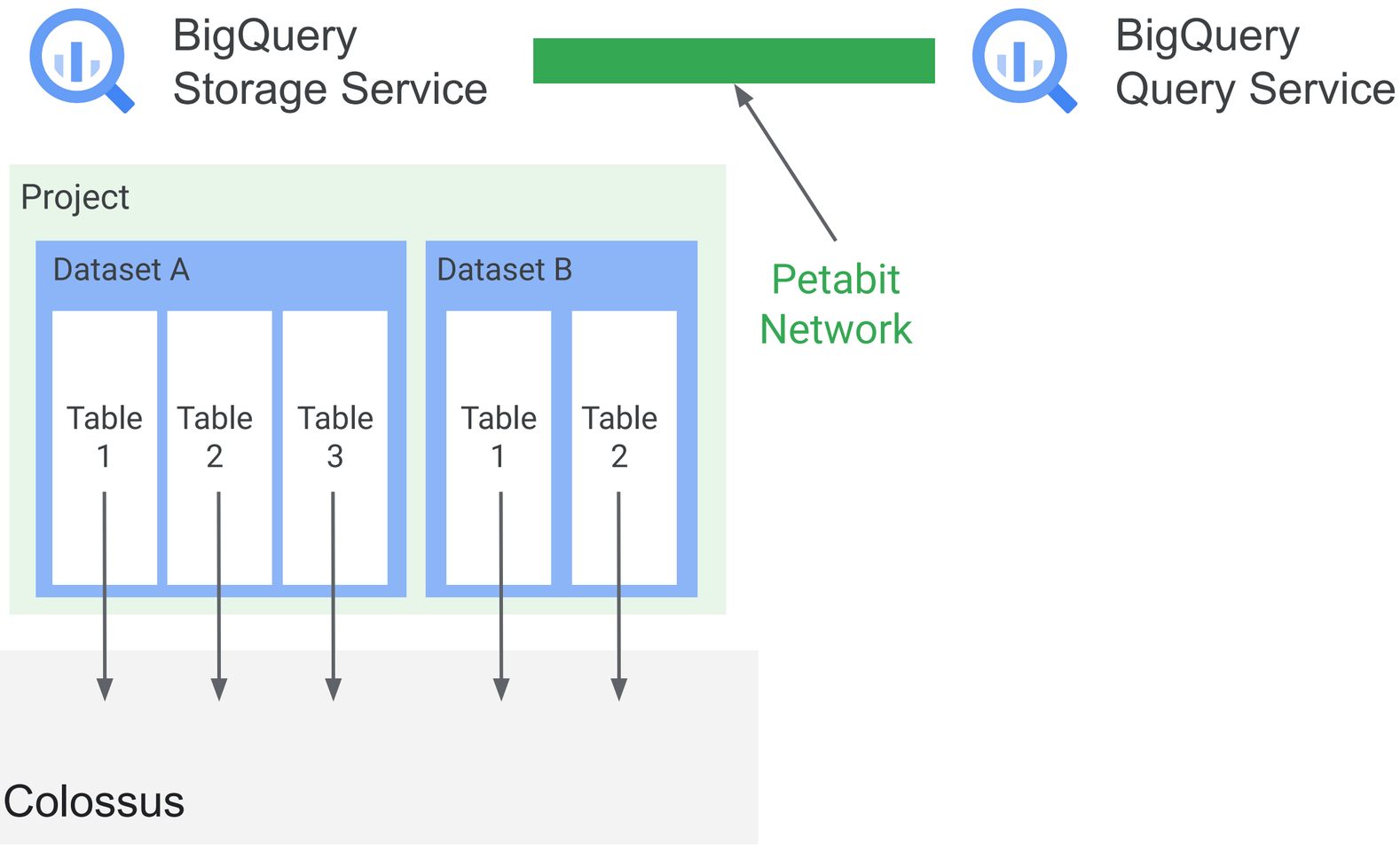
How does BigQuery work?



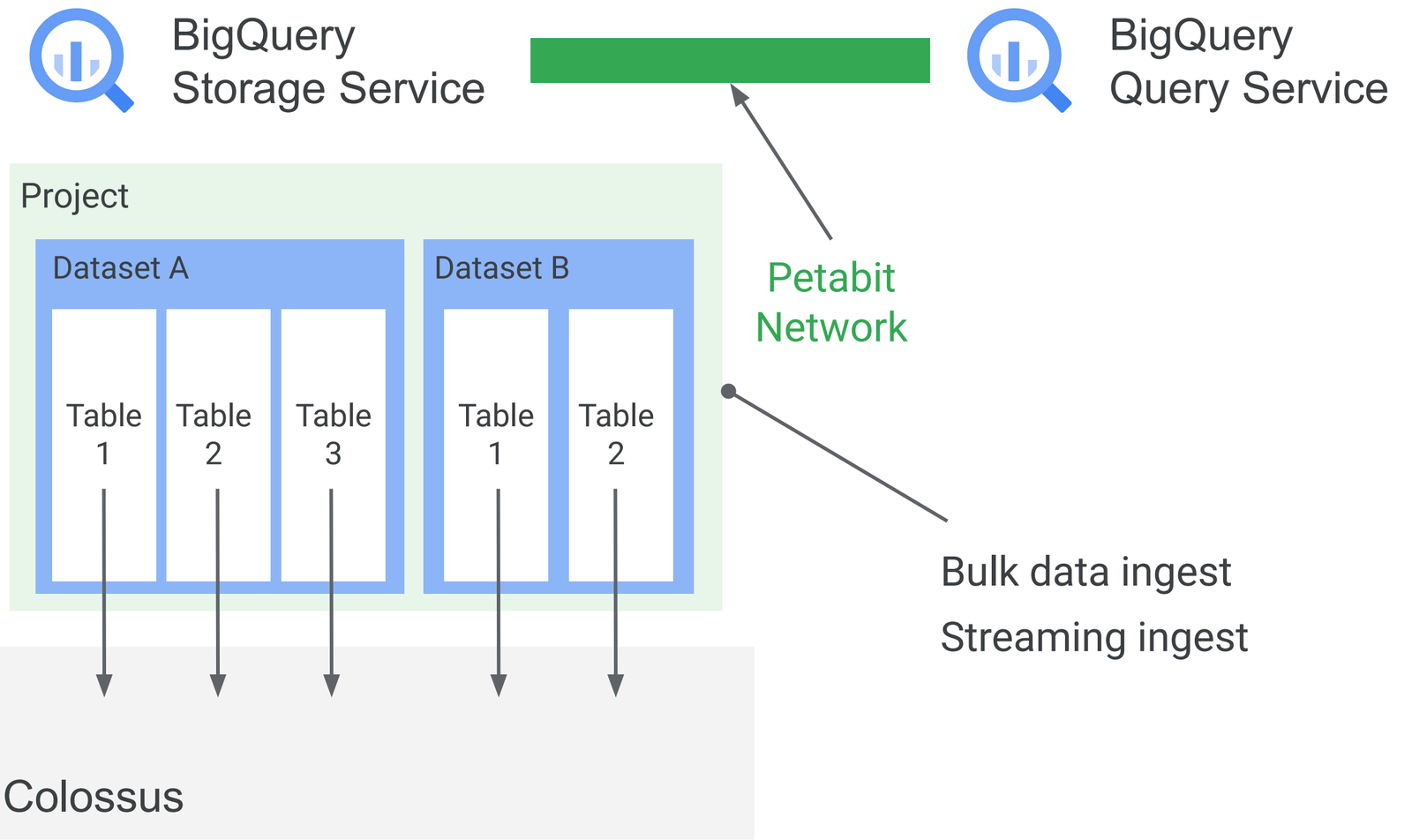
How does BigQuery work?



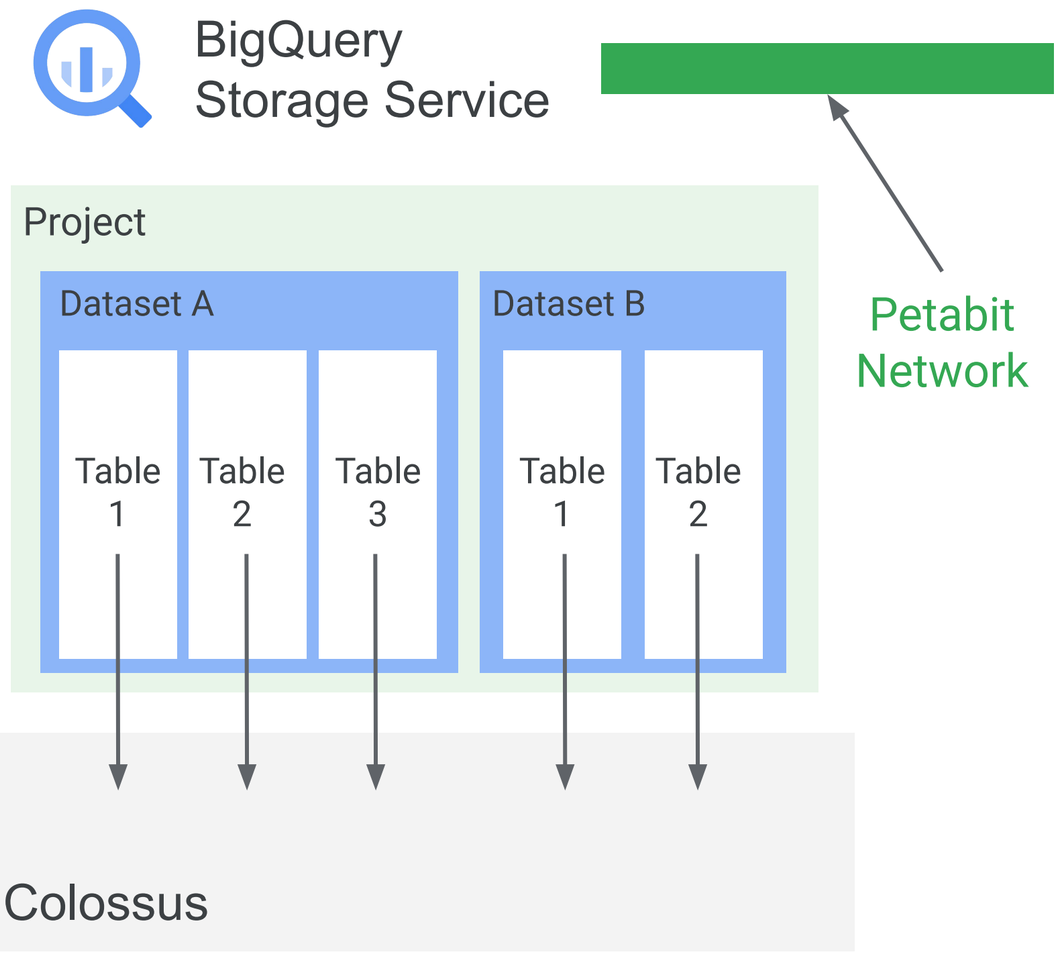
How does BigQuery work?



How does BigQuery work?



How does BigQuery work?



Petabit Network


```
#standardSQL
SELECT
  COUNT(*) AS total_trips
FROM
`bigquery-public-data.san_francisco_bikeshare.bikeshare_trips`
```

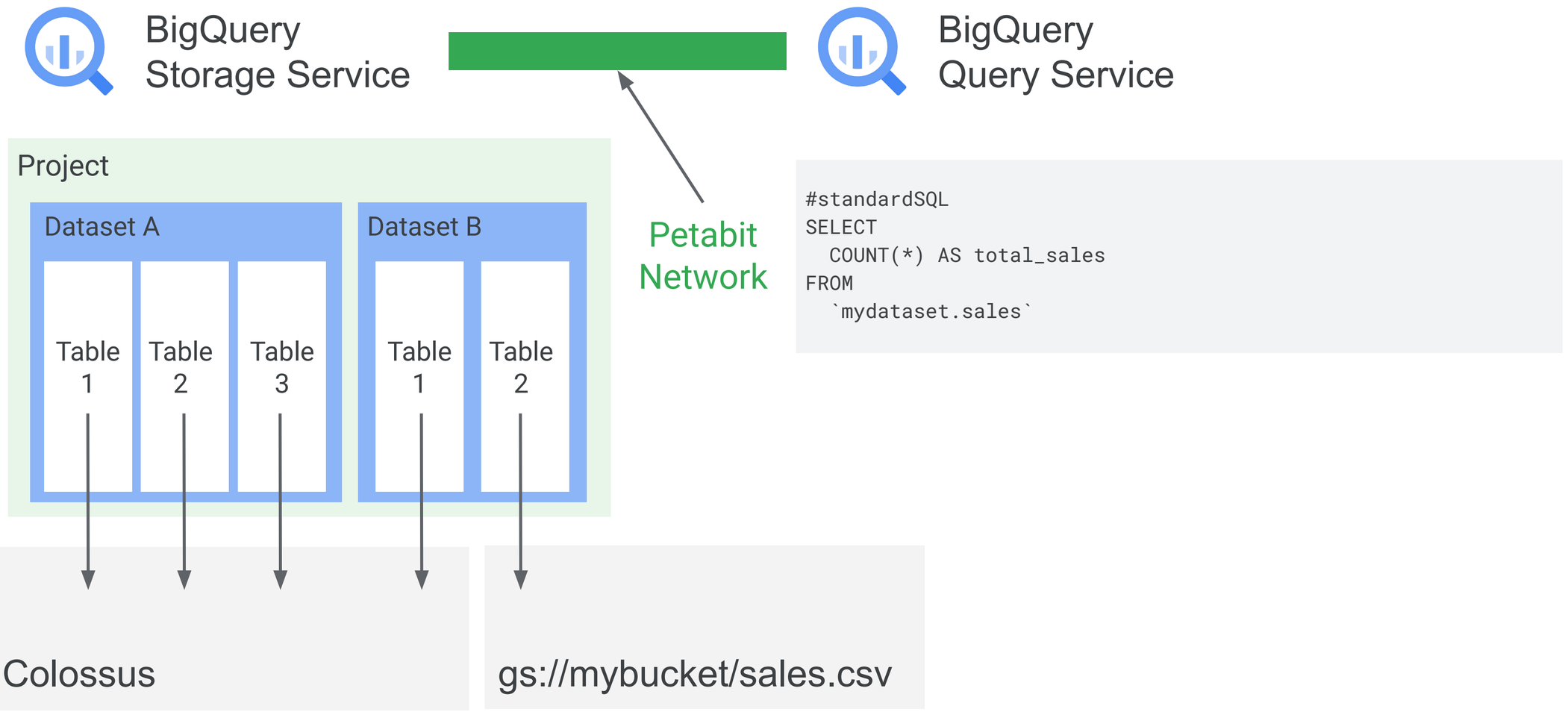
Web UI

bq

REST API (7 languages supported)

Google Cloud

How does BigQuery work?



BigQuery supports standard SQL queries for analysis

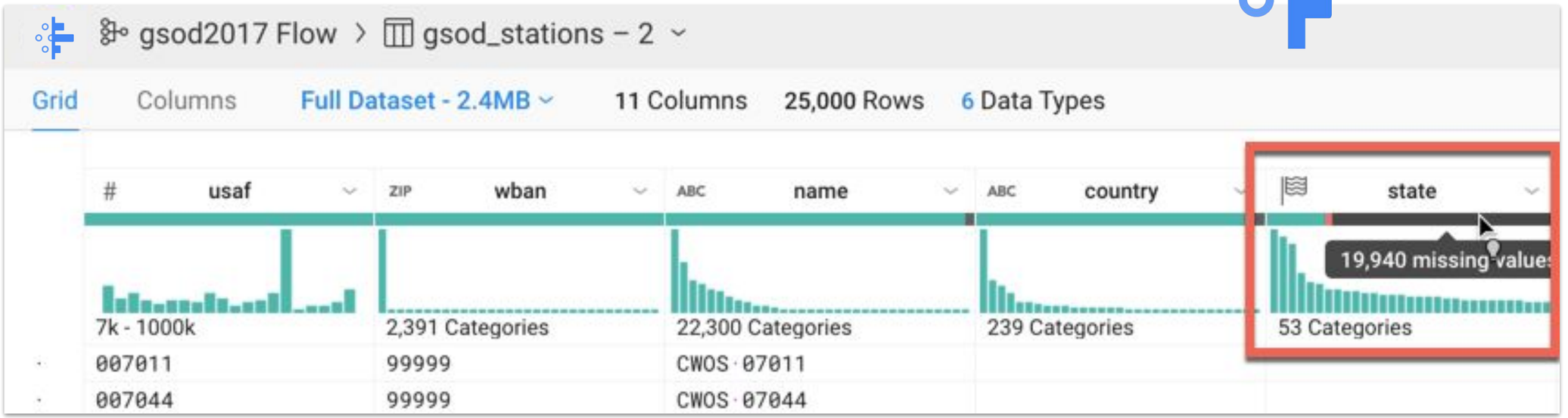
```
#standardSQL
SELECT
  COUNT(*) AS total_trips
FROM
  `bigquery-public-data.san_francisco_bikeshare.bikeshare_trips`
```

Row	total_trips
1	1947419

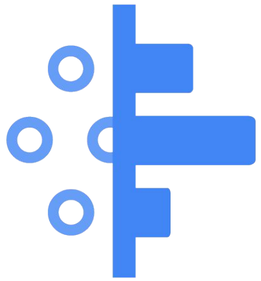
Common SQL operations include deduplication and cleansing

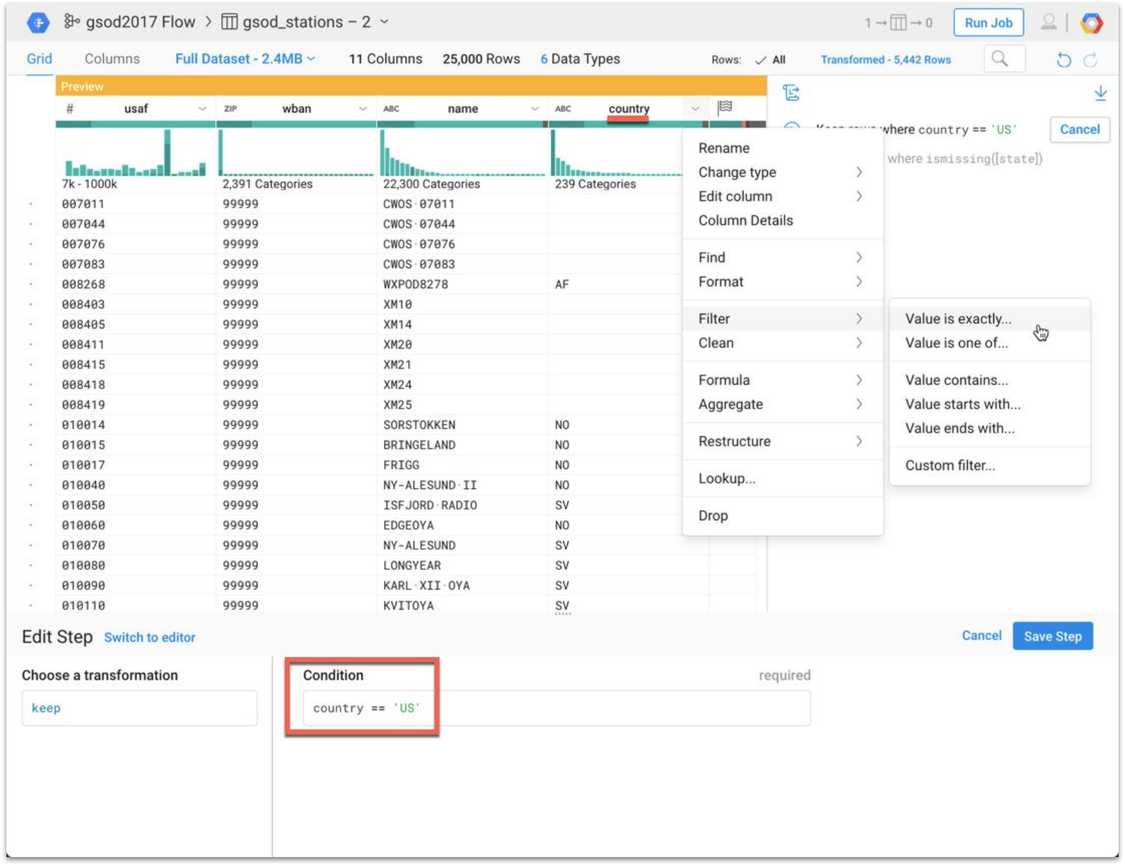
```
# Top 5 stations for casual bike share users
SELECT
  start_station_name,
  COUNT(*) AS total_trips
FROM
  `bigquery-public-data.san_francisco_bikeshare.bikeshare_trips`

WHERE c_subscription_type = 'Customer'
AND member_birth_year IS NOT NULL

GROUP BY start_station_name
ORDER BY total_trips DESC
LIMIT 5
```

Dataprep: Cleaning and transforming data with a UI



Common SQL operations include deduplication and cleansing





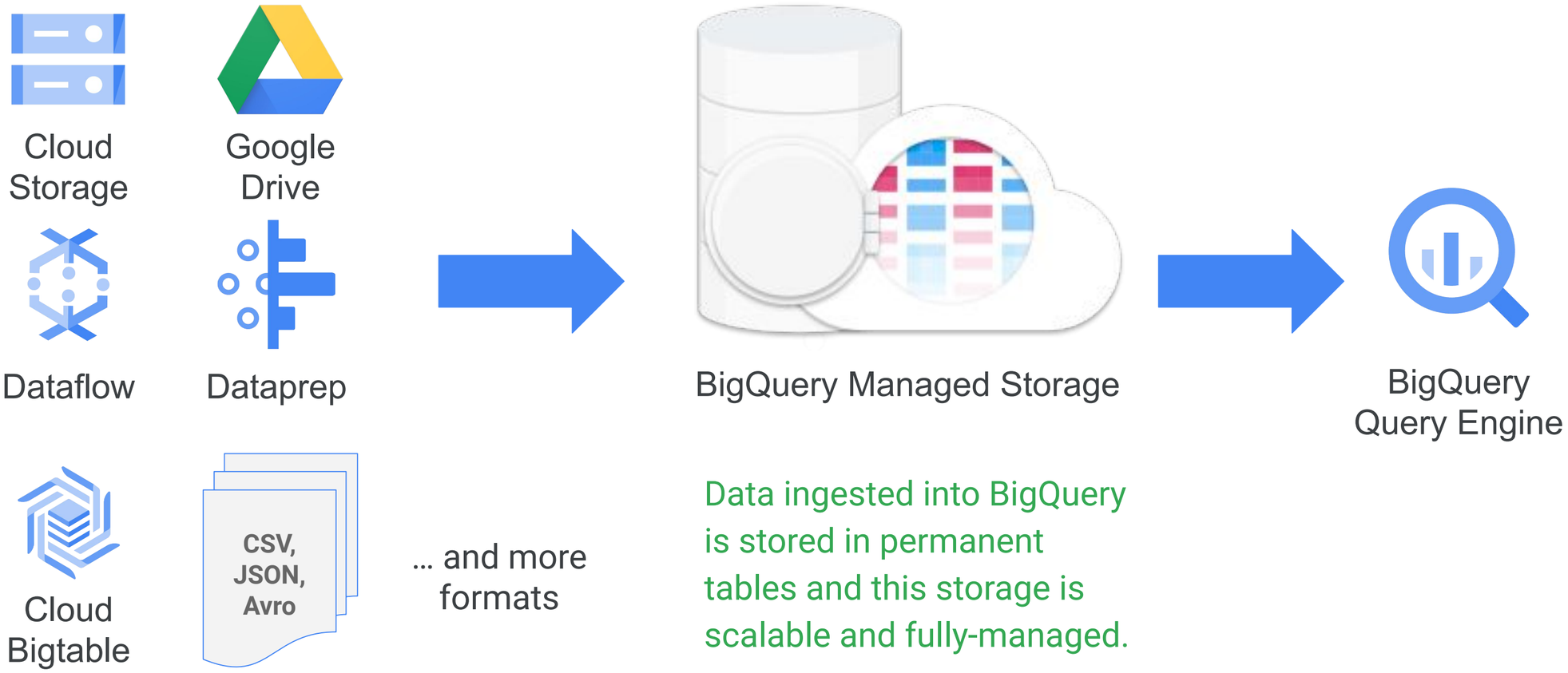
Google Cloud

BigQuery is two services in one

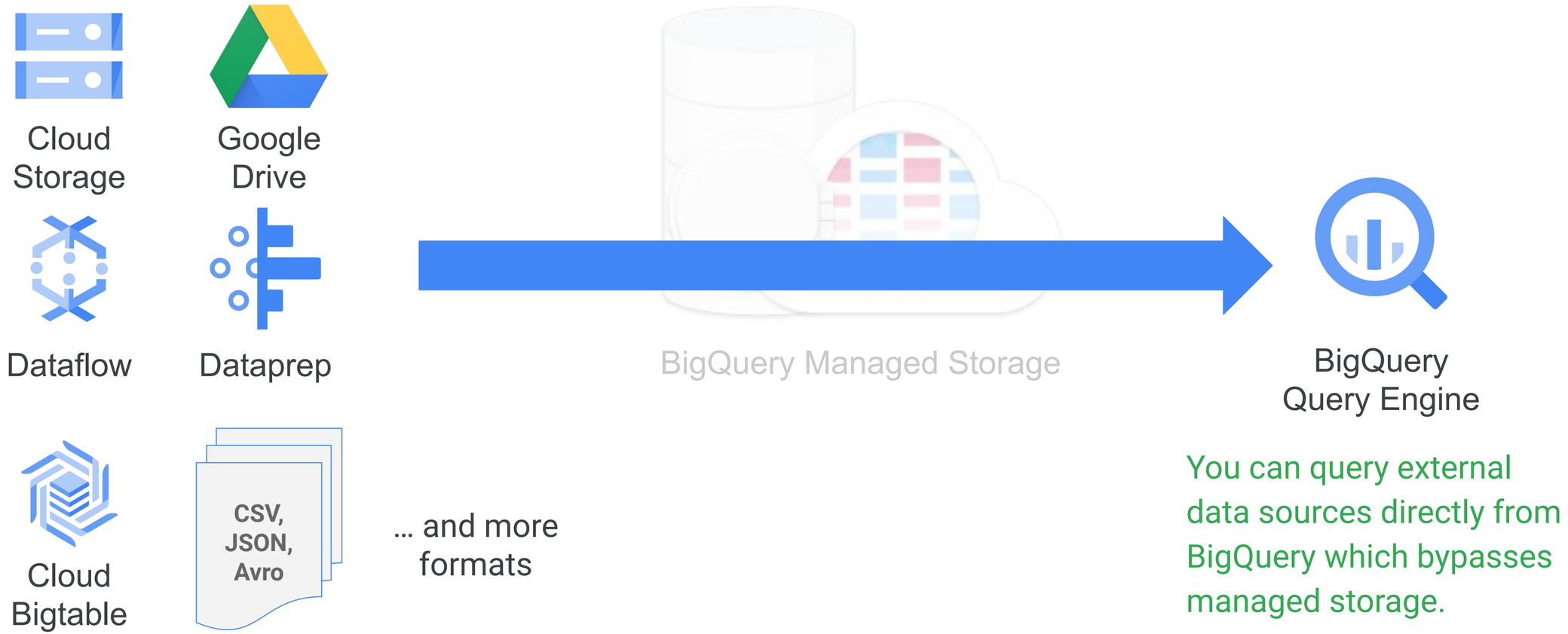
BigQuery

1. Fast SQL Query Engine
2. **Managed storage for datasets**

Use native BigQuery storage for the highest performance

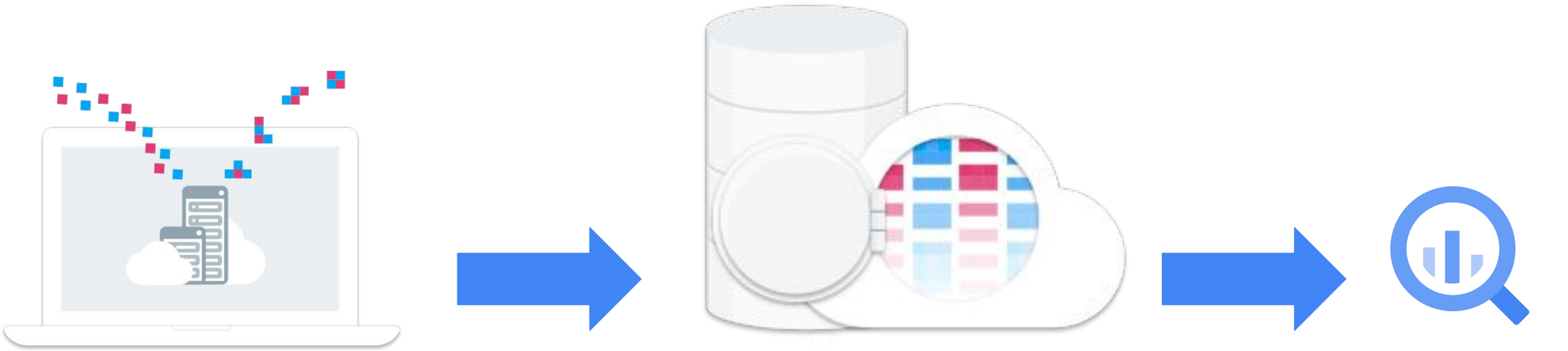


BigQuery can query external (aka federated) data sources in Cloud Storage and Drive directly



You can query external data sources directly from BigQuery which bypasses managed storage.

Streaming records into BigQuery through the API



Streaming Record Inserts

BigQuery Managed Storage

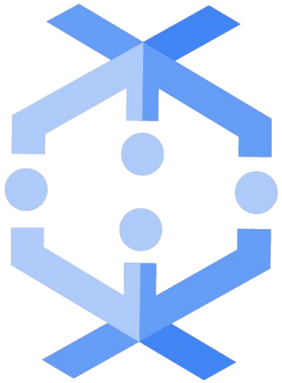
BigQuery
Query Engine

Streaming data allows you to query data without waiting for a full batch load.

BigQuery supports streaming data ingestion directly or via Dataflow pipelines

BigQuery

or



Dataflow

Challenge: Normalization vs Denormalization for reporting

Payments	Event Timestamps
Pickups	Dropoffs

Use STRUCTs and ARRAYS for consolidated reporting

Job information | Results | JSON | Execution details

Row	order_id	service_type	payment_method	event.status	event.time	pickup.latitude	pickup.longitude	destination.latitude	destination.longitude	total_distance_km
1	CR-6098	GO_CAR	CASH	DRIVER_FOUND	2018-12-31 01:45:15.667 UTC	-6.9228116	107.5930599	-6.912966	107.609604	2.9749999046325684
				COMPLETED	2018-12-31 02:06:56.894 UTC					
				PICKED_UP	2018-12-31 02:05:40.075 UTC					
				CREATED	2018-12-31 01:44:59.239 UTC					

BigQuery natively supports ARRAYs as data types.

Use STRUCTs and ARRAYS for consolidated reporting

Job information | Results | JSON | Execution details

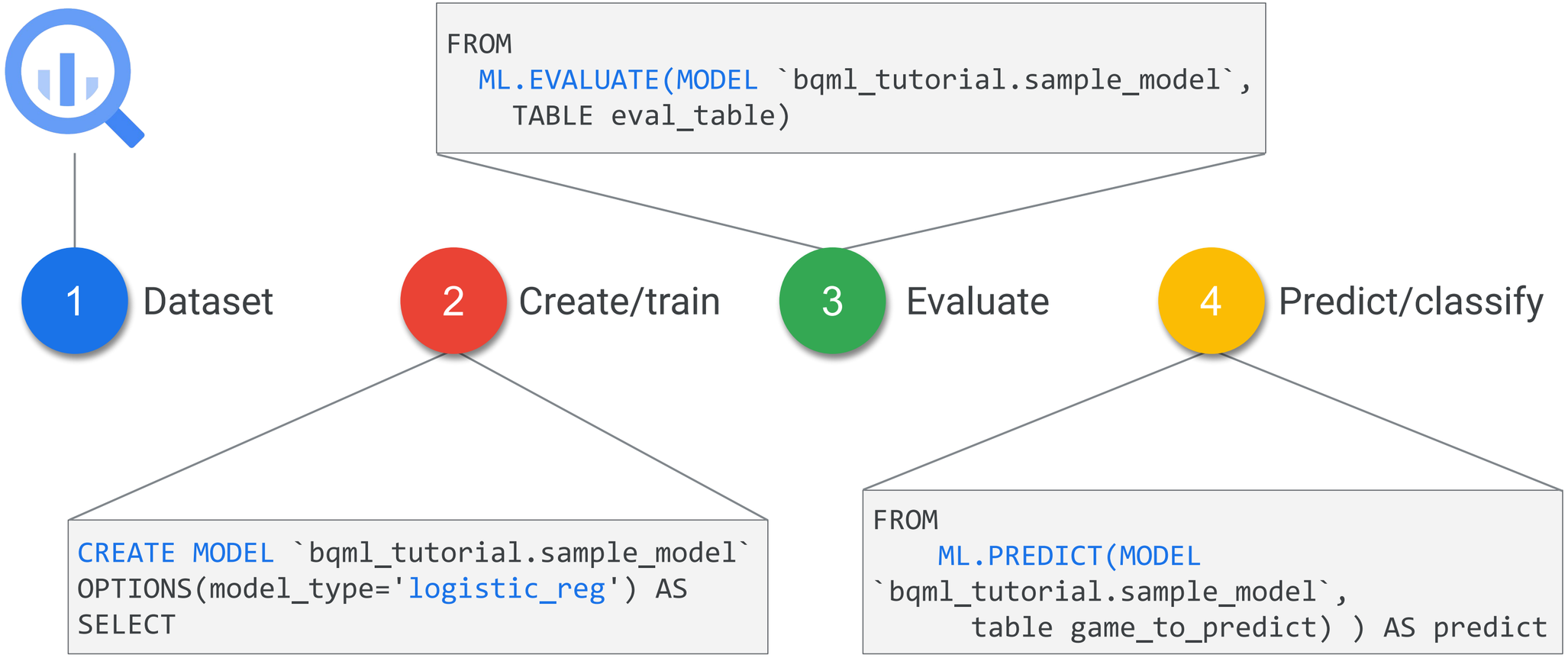
Row	order_id	service_type	payment_method	event.status	event.time	pickup.latitude	pickup.longitude	destination.latitude	destination.longitude	total_distance_km
1	CR-6098	GO_CAR	CASH	DRIVER_FOUND	2018-12-31 01:45:15.667 UTC	-6.9228116	107.5930599	-6.912966	107.609604	2.9749999046325684
				COMPLETED	2018-12-31 02:06:56.894 UTC					
				PICKED_UP	2018-12-31 02:05:40.075 UTC					
				CREATED	2018-12-31 01:44:59.239 UTC					

Schema | Details | Preview

Field name	Type	Mode
order_id	STRING	NULLABLE
service_type	STRING	NULLABLE
payment_method	STRING	NULLABLE
event	RECORD	REPEATED
event. **status**	STRING	NULLABLE
event. **time**	TIMESTAMP	NULLABLE
pickup	RECORD	NULLABLE
pickup. **latitude**	FLOAT	NULLABLE

BigQuery natively supports ARRAYs as data types (and STRUCTs too).

Working with BigQuery ML



SQL query to extract data

1

```
SELECT
  url, title
FROM
  `bigquery-public-data.hacker_news.stories`
WHERE
  LENGTH(title) > 10
  AND LENGTH(url) > 0
LIMIT 10
```

url	title
http://www.bbc.co.uk/news/business-27732743	Vodafone reveals direct government wiretaps
https://www.kickstarter.com/projects/appdocu/a...	Doc – App: The Human Story
http://www.starwebworld.com/android-jelly-bean...	Android Jelly Bean: Streaming Audio Through th...
http://www.myplanetdigital.com/digital_strateg...	Why Canadian Tech Entrepreneurs Need to Man/Wo...
http://startupislandconference.com/index.html	StartupConference June 13. - 16. 2013, HVAR Cr...
http://kopimism.org/	Kopimism Hactivism Meetup Tomorrow (Sunday) in...
http://unearthedgadget.com/xbox-live-gold-2/14...	Xbox Live Gold Membership Is It Really Worth -...
https://evertale.com	Evertale changes the way people remember
http://www.racketboy.com/retro/commodore-amiga...	Commodore Amiga: A Beginner's Guide
http://www.extremetech.com/extreme/156393-cold...	Cold fusion reactor "independently verified"

Use regex to get source + train on words of title

1

txtclass_words — LINK SHARING

```
WITH extracted AS (
SELECT source, REGEXP_REPLACE(LOWER(REGEXP_REPLACE(title, '[^a-zA-Z0-9 $.-]', ' ')), "  "
FROM
  (SELECT
    ARRAY_REVERSE(SPLIT(REGEXP_EXTRACT(url, '.*://(.[^/]+)/'), '.'))[OFFSET(1)] AS source
    title
  FROM
    `bigquery-public-data.hacker_news.stories`
  WHERE
    REGEXP_CONTAINS(REGEXP_EXTRACT(url, '.*://(.[^/]+)/'), '.com$')
    AND LENGTH(title) > 10
  )
)
, ds AS (
SELECT ARRAY_CONCAT(SPLIT(title, " "), ['NULL', 'NULL', 'NULL', 'NULL', 'NULL']) AS words
extracted
WHERE (source = 'github' OR source = 'nytimes' OR source = 'techcrunch')
)
SELECT
source,
words[OFFSET(0)] AS word1,
words[OFFSET(1)] AS word2,
words[OFFSET(2)] AS word3,
words[OFFSET(3)] AS word4,
words[OFFSET(4)] AS word5
FROM ds
```

Run | Save query | Save view | More — This query will process 204.

Query results — SAVE RESULTS — EXPLORE IN DATA STUDIO

37293	nytimes	the	socratic	shrink	NULL	NULL
37294	nytimes	still	stuck	in	a	climate
37295	nytimes	as	unlimited	data	plans	are
37296	nytimes	disney	s	neuroscience	advertising	lab

Create classification model

2

```
CREATE OR REPLACE MODEL advdata.txtclass
OPTIONS(model_type='logistic_reg', input_label_cols=['source'])
AS

WITH extracted AS (
...
)
, ds AS (
SELECT ARRAY_CONCAT(SPLIT(title, " "), ['NULL', 'NULL', 'NULL',
'NULL', 'NULL']) AS words, source FROM extracted
WHERE (source = 'github' OR source = 'nytimes' OR source =
'techcrunch')
)
SELECT
source,
words[OFFSET(0)] AS word1,
words[OFFSET(1)] AS word2,
words[OFFSET(2)] AS word3,
words[OFFSET(3)] AS word4,
words[OFFSET(4)] AS word5
FROM ds
```

Query to extract training data

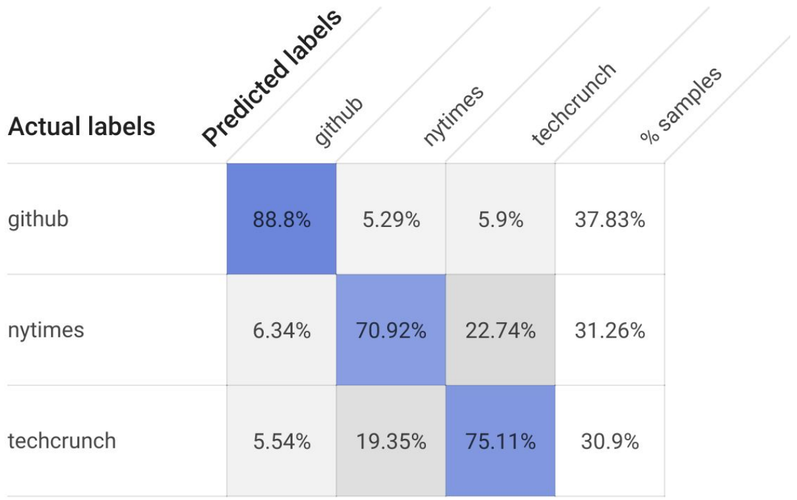
Evaluate model

```
SELECT * FROM ML.EVALUATE(MODEL advdata.txtclass)
```

3

precision	recall	accuracy	f1_score	log_loss	roc_auc
0.783	0.783	0.79	0.783	0.858	0.918

(BigQuery ML splits the training data and reports evaluation statistics on the held-out set)



Predict using trained model

4

```
SELECT * FROM ML.PREDICT(MODEL advdata.txtclass,(
     SELECT 'government' AS word1, 'shutdown' AS word2, 'leaves'
AS word3, 'workers' AS word4, 'reeling' AS word5
     UNION ALL SELECT 'unlikely', 'partnership', 'in', 'house',
'gives'
     UNION ALL SELECT 'fitbit', 's', 'fitness', 'tracker', 'is'
     UNION ALL SELECT 'downloading', 'the', 'android', 'studio',
'project'
))
```

Row	predicted_source	word1	word2	word3	word4	word5
1	nytimes	government	shutdown	leaves	workers	reeling
2	nytimes	unlikely	partnership	in	house	gives
3	techcrunch	fitbit	s	fitness	tracker	is
4	techcrunch	downloading	the	android	studio	project

"Batch prediction"

Linear Classifier (Logistic regression)

```
create or replace model models.will_buy_banana_example
options(model_type='logistic_reg', input_label_cols=['banana']) AS

with purchases_data AS (
select
  CAST(user_id AS string) customer_id,
  CAST(zip_code AS string) home_zipcode,
  ['dawn', 'morning', 'afternoon', 'night'][OFFSET(CAST
(TRUNC(order_hour_of_day/6) AS INT64))] AS time_of_day,
  (SELECT 24852 IN (SELECT product_id FROM UNNEST(order_lines))) AS
banana
FROM operations.orders_with_lines
JOIN operations.customers_loyalty
ON user_id = id
)
select * from purchases_data
```

DNN Classifier

```
create or replace model models.will_buy_banana_example
options(model_type='dnn_classifier', hidden_units=[64, 8],
input_label_cols=['banana']) AS

with purchases_data AS (
select
  CAST(user_id AS string) customer_id,
  CAST(zip_code AS string) home_zipcode,
  ['dawn', 'morning', 'afternoon', 'night'][OFFSET(CAST
(TRUNC(order_hour_of_day/6) AS INT64))] AS time_of_day,
  (SELECT 24852 IN (SELECT product_id FROM UNNEST(order_lines))) AS
banana
FROM operations.orders_with_lines
JOIN operations.customers_loyalty
ON user_id = id
)
select * from purchases_data
```

Linear regression in SQL

```
CREATE OR REPLACE MODEL ch09edu.bicycle_model
OPTIONS(input_label_cols=['duration'],
        model_type='linear_reg')
AS

SELECT
  duration
  , start_station_name
  , CAST(EXTRACT(dayofweek from start_date) AS STRING)
         as dayofweek
  , CAST(EXTRACT(hour from start_date) AS STRING)
         as hourofday
FROM
  `bigquery-public-data.london_bicycles.cycle_hire`
```

DNN regression in SQL

```
CREATE OR REPLACE MODEL ch09edu.bicycle_model
OPTIONS(input_label_cols=['duration'],
        model_type='dnn_regressor', hidden_units=[32, 4])
AS

SELECT
  duration
  , start_station_name
  , CAST(EXTRACT(dayofweek from start_date) AS STRING)
         as dayofweek
  , CAST(EXTRACT(hour from start_date) AS STRING)
         as hourofday
FROM
  `bigquery-public-data.london_bicycles.cycle_hire`
```

BigQuery preprocessing - Feature Engineering

1 Representation transformation

Converting a numeric feature to a categorical feature (through bucketization) and converting categorical features to a numeric representation (through one-hot encoding, learning with counts, sparse feature embeddings, and so on). Some models work only with numeric or categorical features, while others can handle mixed type features. Even when models handle both types, they can benefit from different representation (numeric and categorical) of the same feature.

2 Feature construction

Creating new features either by using typical techniques, such as polynomial expansion (by using univariate mathematical functions), or feature crossing (to capture feature interactions). Features can also be constructed by using business logic from the domain of the ML use case.

Basic feature engineering

Example of preprocessing in BigQuery

```
SELECT
   (tolls_amount + fare_amount)
      AS fare_amount,
   DAYOFWEEK(pickup_datetime)
      AS dayofweek,
   HOUR(pickup_datetime)
            AS hourofday,
     ...
   FROM
     `nyc-tlc.yellow.trips`
   WHERE
     trip_distance > 0
```

Built-in SQL math and data processing functions

Data processing functions

Specify SQL filtering operations

Basic feature engineering

Example of dates and time

```
EXTRACT(DAYOFWEEK FROM  pickup_datetime) AS dayofweek,

EXTRACT(HOUR FROM pickup_datetime) AS hourofday,
```

```
CONCAT(CAST(EXTRACT(DAYOFWEEK FROM pickup_datetime) AS
STRING), CAST(EXTRACT(HOUR FROM pickup_datetime) AS
STRING)) AS hourofday,
```

Basic feature engineering

Example of dates and time

DATA STUDIO OUTPUT: From EXTRACT dates/time queries

	dayofweek	hourofday
1.	Week 6	4 AM
2.	Week 5	3 AM
3.	Week 4	2 AM
4.	Week 7	1 AM
5.	Week 3	12 AM

Note

For all non-numeric columns other than TIMESTAMP, BigQuery ML performs a one-hot encoding transformation that generates a separate feature for each unique value in the column.

BigQuery ML labs

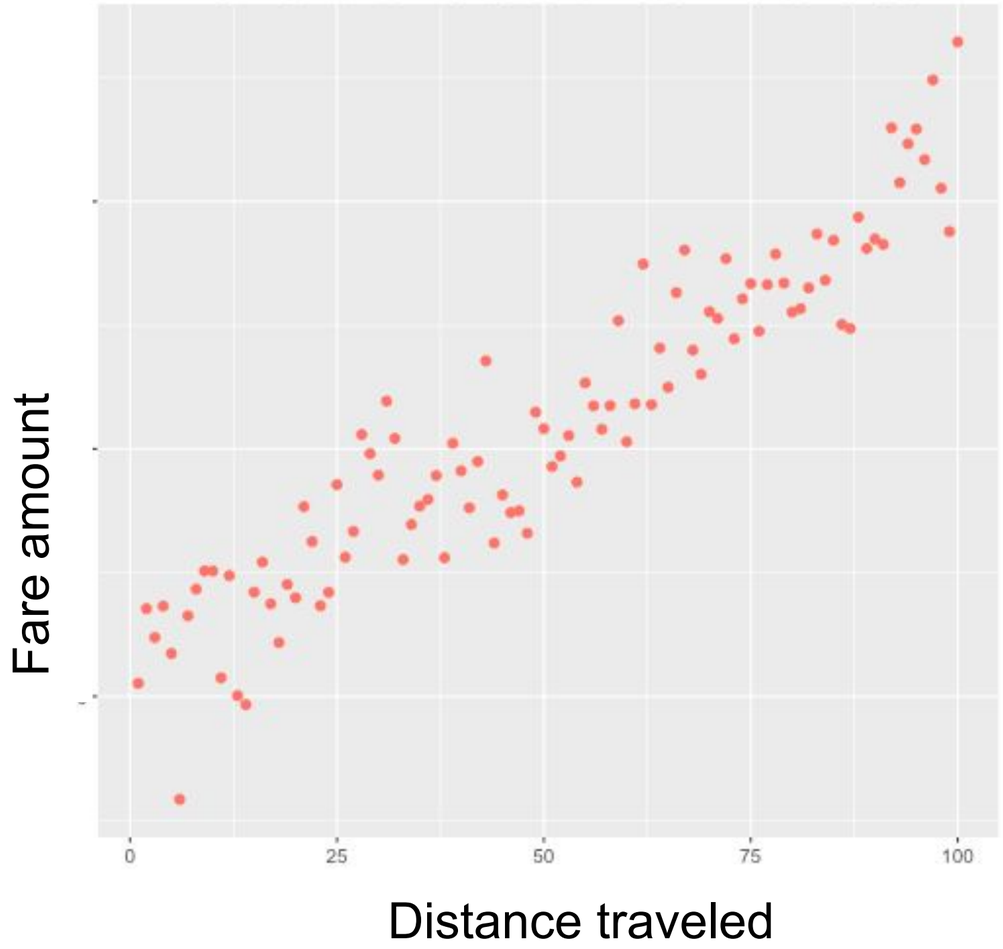
Feature Engineering in BigQuery ML

- Basic feature engineering
- Advanced feature engineering

Regression problem: Predicting taxi fare

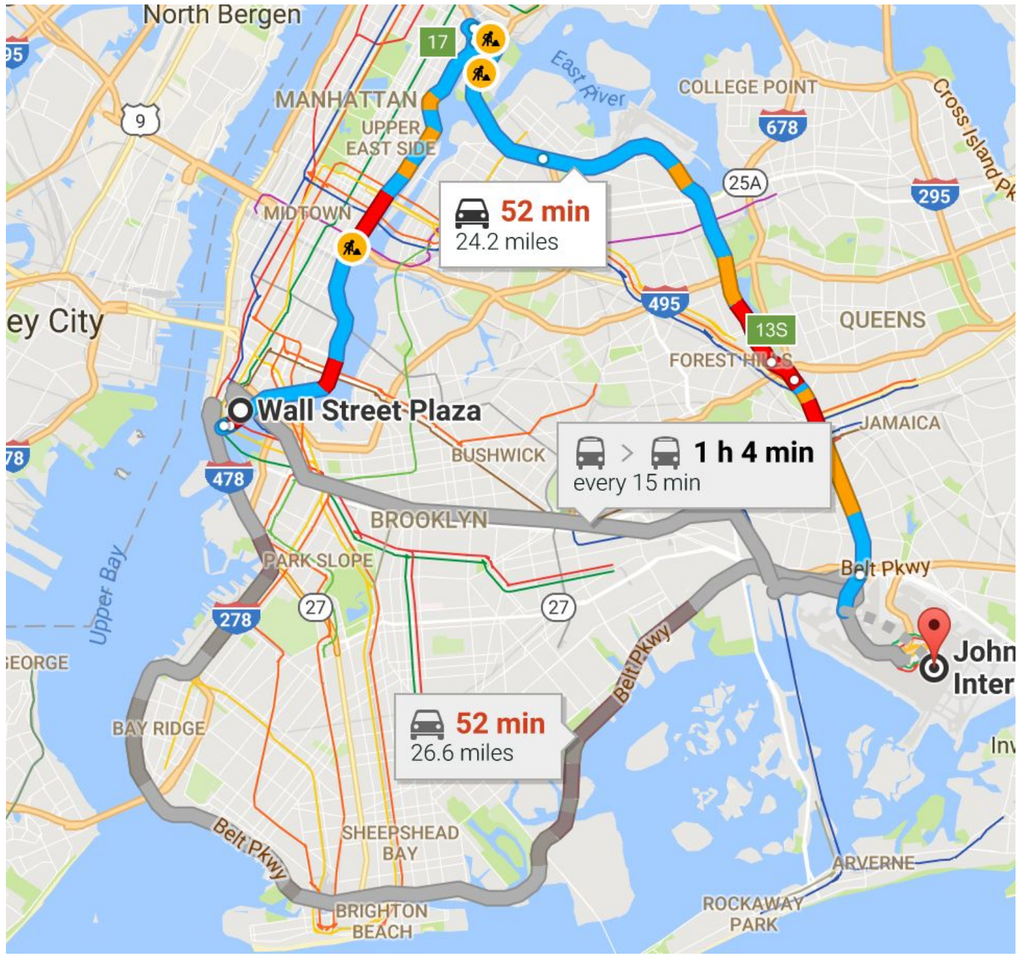
Problem: Predict taxi fare amount based on distance travelled

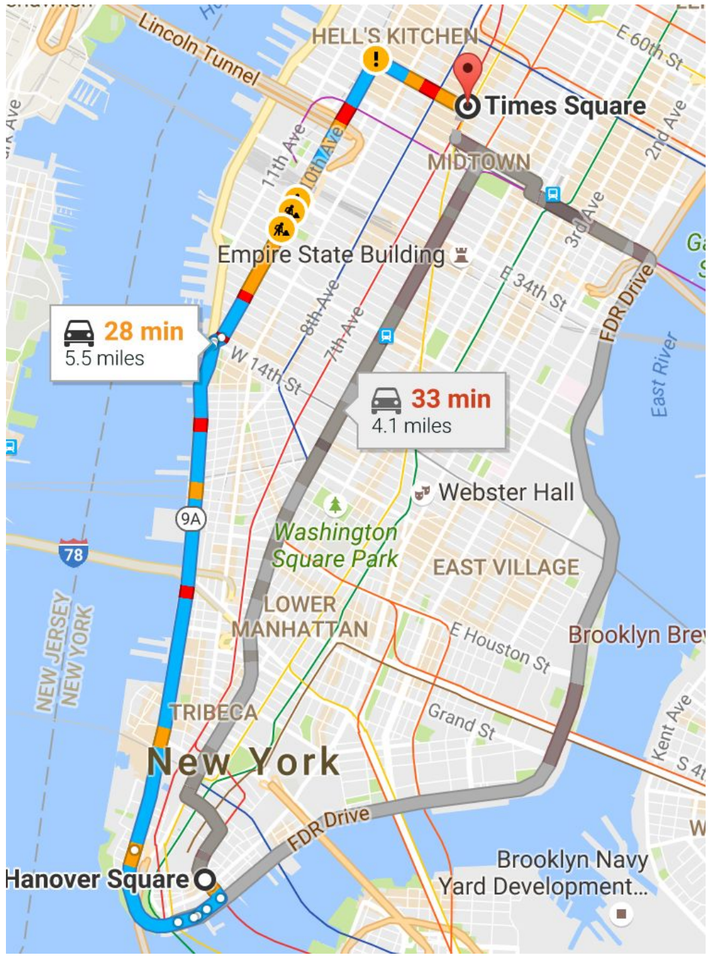
What is the error measure to optimize? RMSE

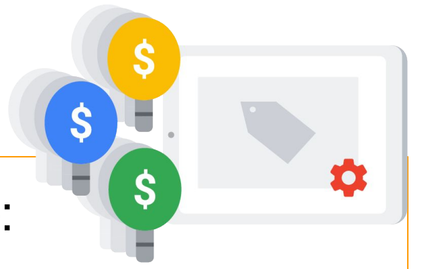


Google Cloud

ML problem: Estimating taxi fare







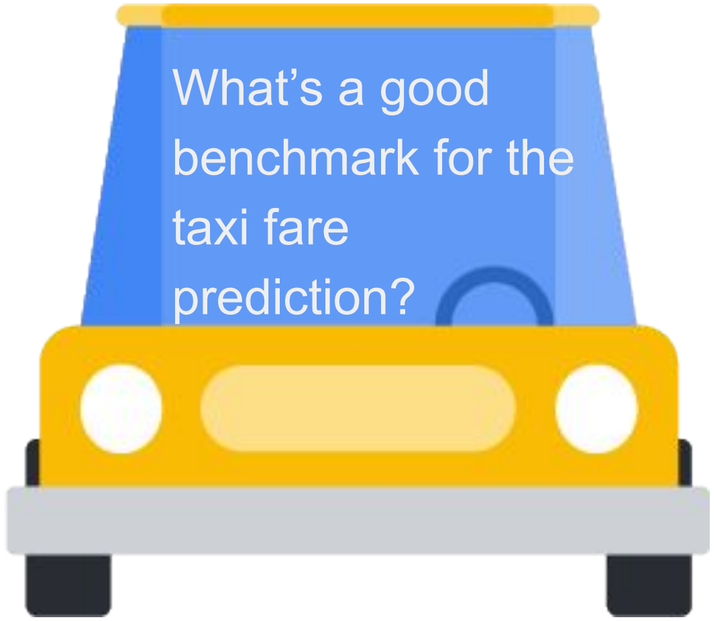
Taxi fare:

Fare amount
+
Distance travelled
+
Number of passengers
+
Pickup and dropoff location
+
Pickup date and time

Baselines are important; It helps to know what error metric is “reasonable” and/or “good” for the problem

A baseline helps you set a goal for a good value for the error metric.

Often a simple heuristic rule can function as a good benchmark.



Evaluate model

```
SELECT * FROM ML.EVALUATE(MODEL feat_eng.baseline_model)
```

mean_absolute_error	mean_squared_error	mean_absolute_log_error	median_absolute_error	R squared
5.21	68.88	0.2581	3.790	.2261

(BigQuery ML splits the training data and reports evaluation statistics on the held-out set)

RMSE

```
SELECT SQRT(mean_squared_error) AS rmse FROM ML.EVALUATE(MODEL
feat_eng.baseline_model)
```

rmse
08.299

The primary evaluation metric for this ML problem is the root mean-squared error. RMSE measures the difference between the predictions of a model and the observed values.

Advanced feature engineering

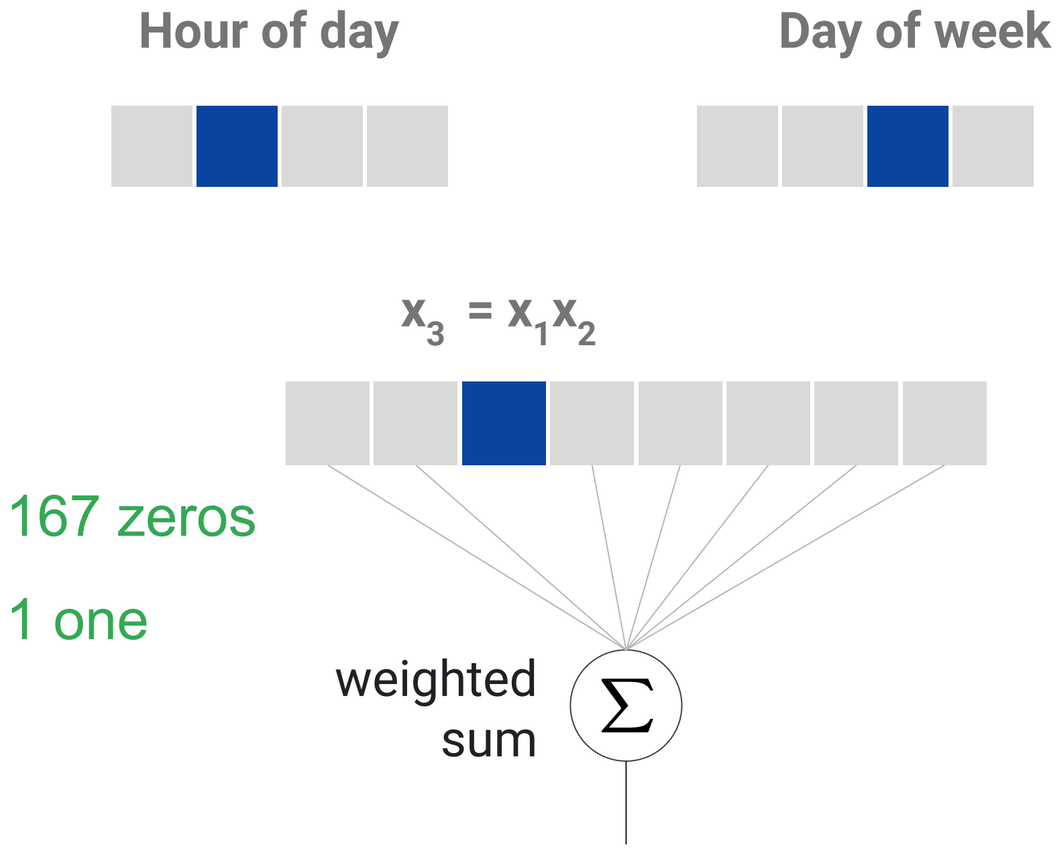
BigQuery ML preprocessing functions

- ML.FEATURE_CROSS(STRUCT(features)) does a feature cross of all the combinations.
- ML.POLYNOMIAL_EXPAND(STRUCT(features), degree) creates x, x2, x3, etc.
- ML.BUCKETIZE(f, split_points) where split_points is an array.

Feature crosses

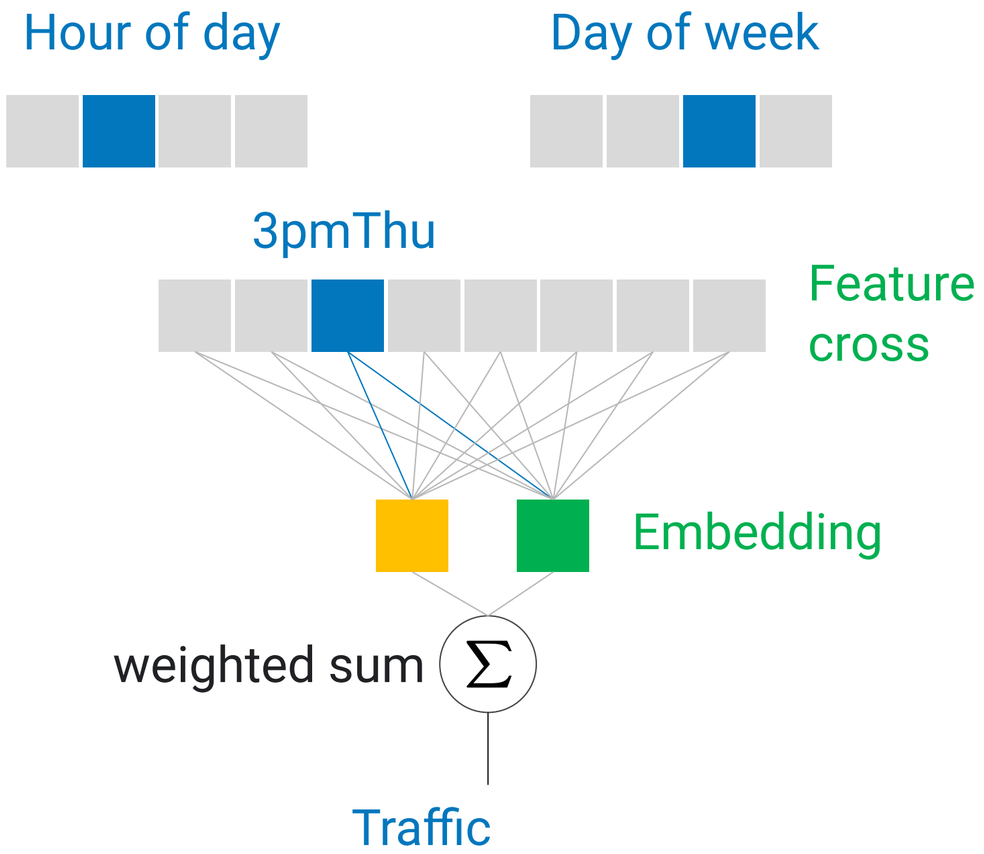
1. Feature crosses memorize!
2. The goal of ML is generalization.
3. Memorization works when you have lots of data.
4. Feature crosses are powerful.

Feature crosses lead to sparsity

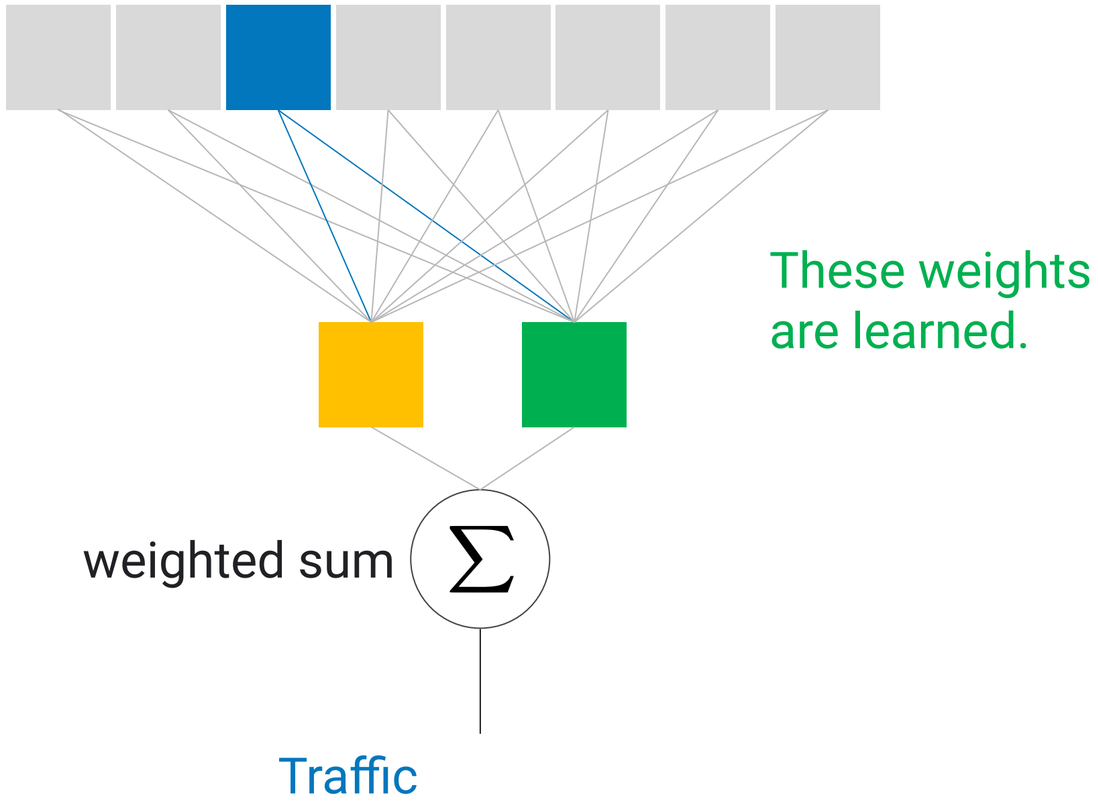


Allow the model to learn traffic patterns by creating a new feature that combines the time of day and day of week (this is called a feature cross).

Creating an embedding column from a feature cross



The weights in the embedding column are learned from data



Feature cross

```
Spatial and Features
```

Schema | Details | Preview

Field name	Type	Mode
fare_amount	FLOAT	NULLABLE
passengers	FLOAT	NULLABLE
euclidean	FLOAT	NULLABLE
day_hr	RECORD	NULLABLE
day_hr. dayofweek_hourofday	STRING	NULLABLE


```
{
  "fare_amount": "2.5",
  "passengers": "1.0",
  "euclidean": "248.06733188206664",
  "day_hr": {
    "dayofweek_hourofday": "1_0"
  },
  "pickup_and_dropoff": "bin_402bin_370bin_403bin_370"
},
```

BUCKETIZE

Spatial Features

```
CONCAT(
    ML.BUCKETIZE(pickuplon, GENERATE_ARRAY(-78, -70, 0.01)),
    ML.BUCKETIZE(pickuplat, GENERATE_ARRAY(37, 45, 0.01)),
    ML.BUCKETIZE(dropofflon, GENERATE_ARRAY(-78, -70, 0.01)),
    ML.BUCKETIZE(dropofflat, GENERATE_ARRAY(37, 45, 0.01))
) AS pickup_and_dropoff
```

```
{
  "fare_amount": "2.5",
  "passengers": "1.0",
  "euclidean": "248.06733188206664",
  "day_hr": {
    "dayofweek_hourofday": "1_0"
  },
  "pickup_and_dropoff": "bin_402bin_370bin_403bin_370"
},
```

pickup_and_dropoff
bin_406bin_379bin_406bin_379
bin_402bin_376bin_402bin_376
bin_422bin_366bin_423bin_366
bin_404bin_378bin_405bin_380
bin_403bin_374bin_403bin_374
bin_401bin_376bin_406bin_368
bin_407bin_376bin_407bin_376
bin_404bin_378bin_404bin_377
bin_402bin_376bin_403bin_376
bin_402bin_376bin_402bin_376
bin_401bin_372bin_401bin_372
bin_401bin_375bin_401bin_375
bin_404bin_380bin_404bin_380

TRANSFORM clause: BigQuery ML

```
CREATE OR REPLACE MODEL feat_eng.final_model
TRANSFORM(
  fare_amount,
    #SQRT( (pickuplon-dropofflon)*(pickuplon-dropofflon) + (pickuplat-dropofflat)*(pickuplat-dropofflat) ) AS euclidean,
      ST_Distance(ST_GeogPoint(pickuplon, pickuplat), ST_GeogPoint(dropofflon, dropofflat)) AS euclidean,
    ML.FEATURE_CROSS(STRUCT(CAST(EXTRACT(DAYOFWEEK FROM pickup_datetime) AS STRING) AS dayofweek,
        CAST(EXTRACT(HOUR FROM pickup_datetime) AS STRING) AS hourofday)) AS day_hr,
   CONCAT(
     ML.BUCKETIZE(pickuplon, GENERATE_ARRAY(-78, -70, 0.01)),
     ML.BUCKETIZE(pickuplat, GENERATE_ARRAY(37, 45, 0.01)),
     ML.BUCKETIZE(dropofflon, GENERATE_ARRAY(-78, -70, 0.01)),
     ML.BUCKETIZE(dropofflat, GENERATE_ARRAY(37, 45, 0.01))
  ) AS pickup_and_dropoff
)
OPTIONS(input_label_cols=['fare_amount'], model_type='linear_reg', l2_reg=0.1)
AS

SELECT * FROM feat_eng.feateng_training_data
```

TRANSFORM ensures that transformations are automatically applied during ML.PREDICT

```
CREATE OR REPLACE MODEL feat_eng.final_model
TRANSFORM(
  fare_amount,
    #SQRT( (pickuplon-dropofflon)*(pickuplon-dropofflon) + (pickuplat-dropofflat)*(pickuplat-dropofflat) ) AS euclidean,
      ST_Distance(ST_GeogPoint(pickuplon, pickuplat), ST_GeogPoint(dropofflon, dropofflat)) AS euclidean,
    ML.FEATURE_CROSS(STRUCT(CAST(EXTRACT(DAYOFWEEK FROM pickup_datetime) AS STRING) AS dayofweek,
        CAST(EXTRACT(HOUR FROM pickup_datetime) AS STRING) AS hourofday)) AS day_hr,
   CONCAT(
     ML.BUCKETIZE(pickuplon, GENERATE_ARRAY(-78, -70, 0.01)),
     ML.BUCKETIZE(pickuplat, GENERATE_ARRAY(37, 45, 0.01)),
     ML.BUCKETIZE(dropofflon, GENERATE_ARRAY(-78, -70, 0.01)),
     ML.BUCKETIZE(dropofflat, GENERATE_ARRAY(37, 45, 0.01))
  ) AS pickup_and_dropoff
)
OPTIONS(input_label_cols=['fare_amount'], model_type='linear_reg', l2_reg=0.1)
AS

SELECT * FROM feat_eng.feateng_training_data
```

Evaluate model

```
SELECT * FROM ML.EVALUATE(MODEL feat_eng.baseline_model)
```

mean_absolute_error	mean_squared_error	mean_absolute_log_error	median_absolute_error	R squared
5.21	68.88	0.2581	3.790	.2261

(BigQuery ML splits the training data and reports evaluation statistics on the held-out set)

RMSE

```
SELECT SQRT(mean_squared_error) AS rmse FROM ML.EVALUATE(MODEL
feat_eng.benchmark_model)
```

rmse
08.299

The primary evaluation metric for this ML problem is the root mean-squared error. RMSE measures the difference between the predictions of a model and the observed values.

Evaluate model

```
SELECT * FROM ML.EVALUATE(MODEL feat_eng.final_model)
```

mean_absolute_error	mean_squared_error	mean_absolute_log_error	median_absolute_error	R squared
2.26	21.65	0.0687	1.3582	0.7567

(BigQuery ML splits the training data and reports evaluation statistics on the held-out set)

RMSE

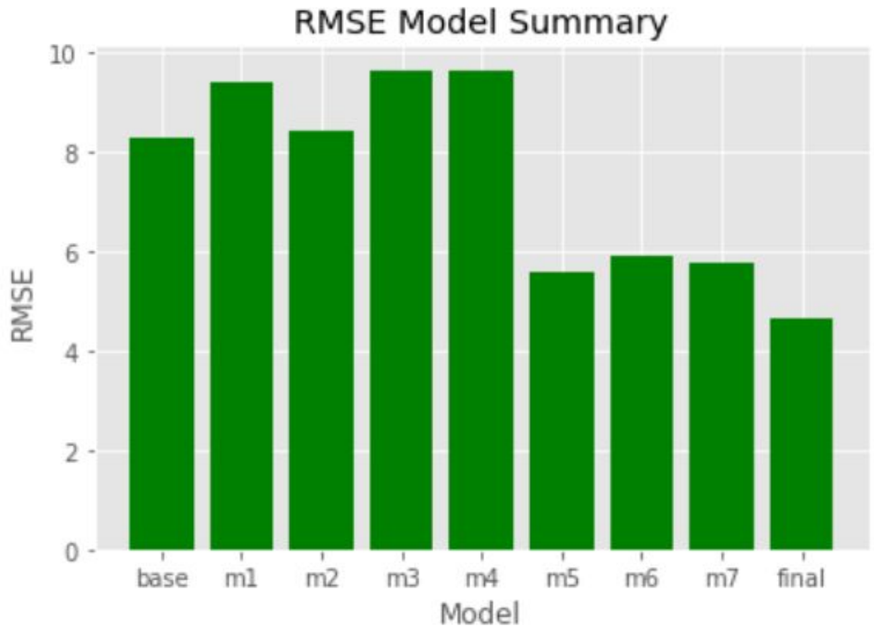
```
SELECT SQRT(mean_squared_error) AS rmse FROM ML.EVALUATE(MODEL
feat_eng.final_model)
```

rmse
4.653

The primary evaluation metric for this ML problem is the root mean-squared error. RMSE measures the difference between the predictions of a model and the observed values.

RMSE: Summary table and visualization

Model	RMSE	Description
baseline_model	8.29	--Baseline model - no feature engineering
model_1	9.431	--EXTRACT DayOfWeek from the pickup_datetime feature
model_2	8.408	--EXTRACT hourofday from the pickup_datetime feature
model_3	8.328	--Feature cross dayofweek and hourofday -Feature Cross does lead ot overfitting
model_4	9.657	--Apply the ML.FEATURE_CROSS clause to categorical features
model_5	5.588	--Feature cross coordinate features to create a Euclidean feature
model_6	5.906	--Feature cross pick-up and drop-off locations features
model_7	5.75	--Apply the BUCKETIZE function
final_model	4.653	--Apply the TRANSFORM clause and L2 Regularization



Google Cloud

BigQuery ML feature engineering summary

- Remove examples that you don't want to train on.
- Compute vocabularies for categorical columns.
- Compute aggregate statistics for numeric columns.
- Consider advanced feature engineering using ML.FEATURE_CROSS, TRANSFORM, and BUCKETIZE.

In BigQuery

Thank you
End of
Session 2

Google Cloud


Google Cloud