Google Cloud

Feature Engineering

Evan Jones

Feature Engineering

Scale to large datasets

Find good features

Preprocess with Cloud MLE

Objectives

Turn raw data to features

What raw data do we need to collect to predict the price of a house?

Lot Size
Number of Rooms

Historical sale price

Location, location, location

Can we use this raw data we've cleaned and collected?


```
0 : {
  house_info : {
    num_rooms: 6
    num_bedrooms: 3
    street_name: “Main Street”
    num_basement_rooms: -1
    ...
  }
}
```

Raw data must be mapped into numerical feature vectors

```
0 : {
  house_info : {
    num_rooms: 6
    num_bedrooms: 3
    street_name: “Main Street”
    num_basement_rooms: -1
    ...
  }
}
```

```
[
  6.0,
  1.0,
  0.0,
  0.0,
  0.0,
  9.321,
  -2.20,
  1.01,
  0.0,
  ...,
]
```

What makes a feature "good"?

```
0 : {
  house_info : {
  ✓ num_rooms: 6
  ✓ num_bedrooms: 3
  ✓ street_name: "Main Street"
  ✓ num_basement_rooms: -1
    ...
  }
}
```

Objectives

Turn raw data to features

Compare good vs bad features

What makes a good feature?

What makes a good feature?

1. Be related to the objective

What makes a good feature?

1. Be related to the objective
2. Be known at prediction-time

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


1325

Choose the good features

A) Breed

B) Age

C) Eye Color

Objective: Good racehorse

A) Breed

B) Age

C) Eye Color

Objective: Eye disease

A) Breed

B) Age

C) Eye Color

Different problems in the same domain may need different features

Quiz

Are these features related to the objective or not?

Predict total number of customers who will use a certain discount coupon

Predict total number of customers who will use a certain discount coupon

1 Font of the text with which the discount is advertised on partner websites

Predict total number of customers who will use a certain discount coupon

1. Font of the text with which the discount is advertised on partner websites
2. Price of the item the coupon applies to

Predict total number of customers who will use a certain discount coupon

1. Font of the text with which the discount is advertised on partner websites
2. Price of the item the coupon applies to
3. Number of items in stock

Predict whether a credit card transaction is fraudulent

Predict whether a credit card transaction is fraudulent

1 Whether cardholder has purchased these items at this store before

Predict whether a credit card transaction is fraudulent

1. Whether cardholder has purchased these items at this store before
2. Credit card chip reader speed

Predict whether a credit card transaction is fraudulent

1. Whether cardholder has purchased these items at this store before
2. Credit card chip reader speed
3. Category of item being purchased

Predict whether a credit card transaction is fraudulent

1 Whether cardholder has purchased these items at this store before

2 Credit card chip reader speed

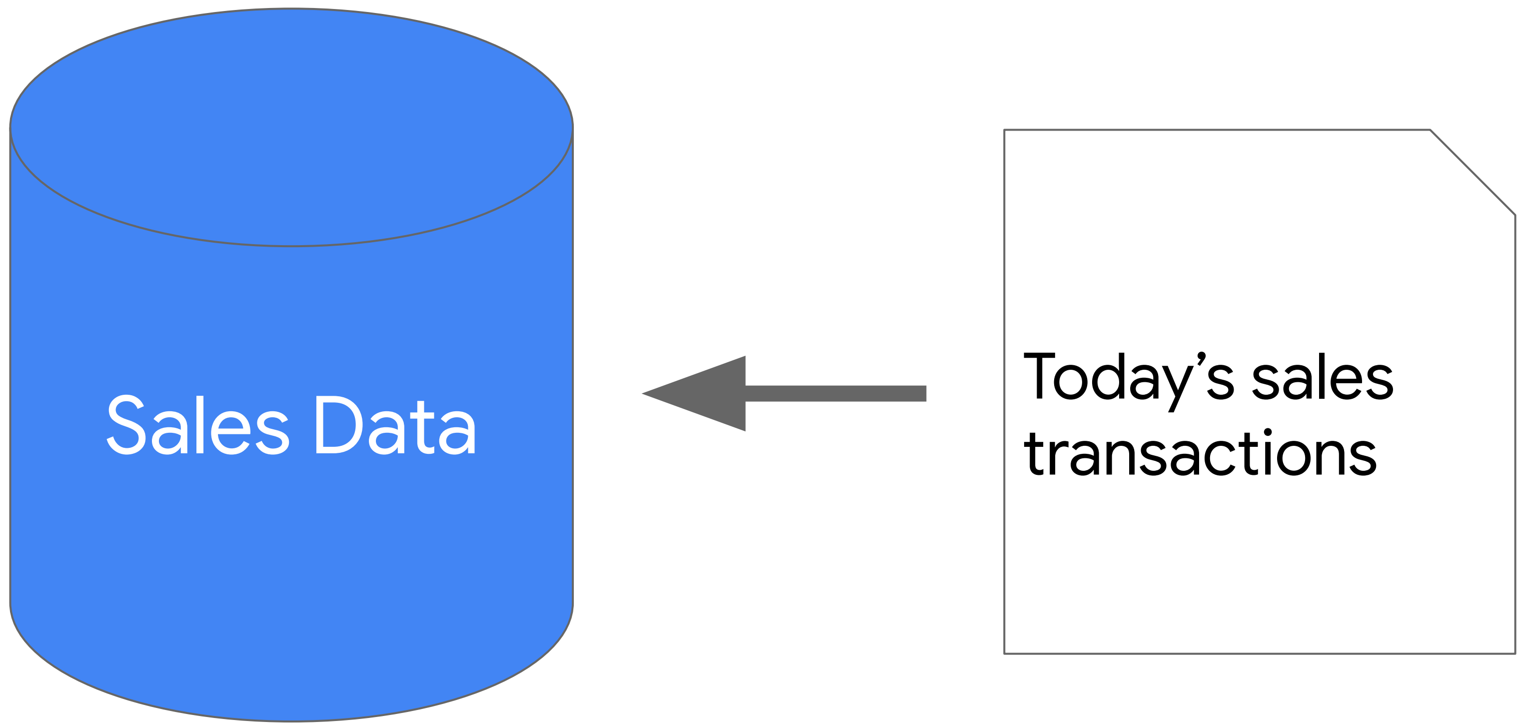
3 Category of item being purchased

4 Expiry date of credit card

Google Cloud

Be known at prediction-time


QUARTZ



Sales Data
Today's sales
transactions

Some data could be known immediately, and some other data is not known in real time.

SOLD: $300,000

What’s wrong with the second feature?

✅ **city_id**:“br/sao_paulo”

❌ **inferred_city_cluster_id**:219

Feature definitions should not change over time

city_id:"br/sao_paulo"

inferred_city_cluster_id:219

Quiz

Is the value knowable at prediction time or not?

Predict total number of customers who will use a certain discount coupon

1 Total number of discountable items sold

Predict total number of customers who will use a certain discount coupon

1. Total number of discountable items sold

2. Number of discountable items sold the previous month

Predict total number of customers who will use a certain discount coupon

1. Total number of discountable items sold

2. Number of discountable items sold the previous month

3. Number of customers who viewed ads about item

Predict whether a credit card transaction is fraudulent

1 Whether cardholder has purchased these items at this store before

You cannot train with current data and predict with stale data


```
SELECT name, COUNT(trans_id) AS count
FROM sales_warehouse
WHERE
   # filter out last three days
   trans_time <
   TIMESTAMP_SUB(
      CURRENT_TIMESTAMP(), INTERVAL 3 DAY
      )
```

Predict whether a credit card transaction is fraudulent

1. Whether cardholder has purchased these items at this store before
2. Whether item is new at store (and can not have been purchased before)

Predict whether a credit card transaction is fraudulent

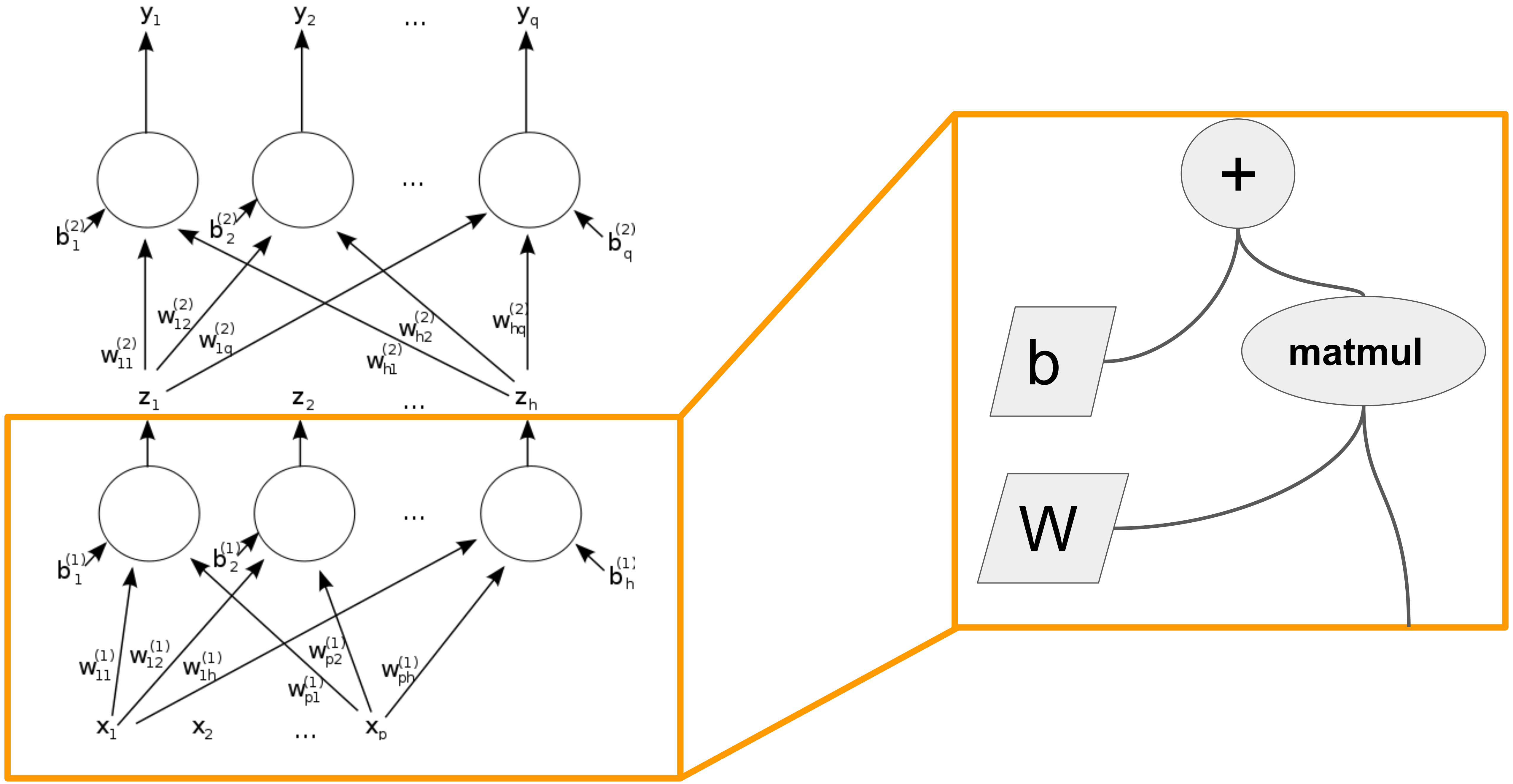
1. Whether cardholder has purchased these items at this store before
2. Whether item is new at store (and can not have been purchased before)
3. Category of item being purchased

Predict whether a credit card transaction is fraudulent

1. Whether cardholder has purchased these items at this store before
2. Whether item is new at store (and can not have been purchased before)
3. Category of item being purchased
4. Online or in-person purchase?

Be numeric with meaningful magnitude

Neural networks are weighing and adding machines



Quiz

Which of these are numeric?

Predict total number of customers who will use a certain discount coupon

1. Percent value of the discount (e.g. 10% off, 20% off, etc.)

Discount coupon usage

1. Percent value of the discount
(e.g. 10% off, 20% off, etc.)

2. Size of the coupon
(e.g. 4 cm2, 24 cm2, 48 cm2, etc.)

Discount coupon usage

1. Percent value of the discount
(e.g. 10% off, 20% off, etc.)

2. Size of the coupon
(e.g. 4 cm2, 24 cm2, 48 cm2, etc.)

3. Font an advertisement is in
(Arial, Times New Roman, etc.)

Discount coupon usage

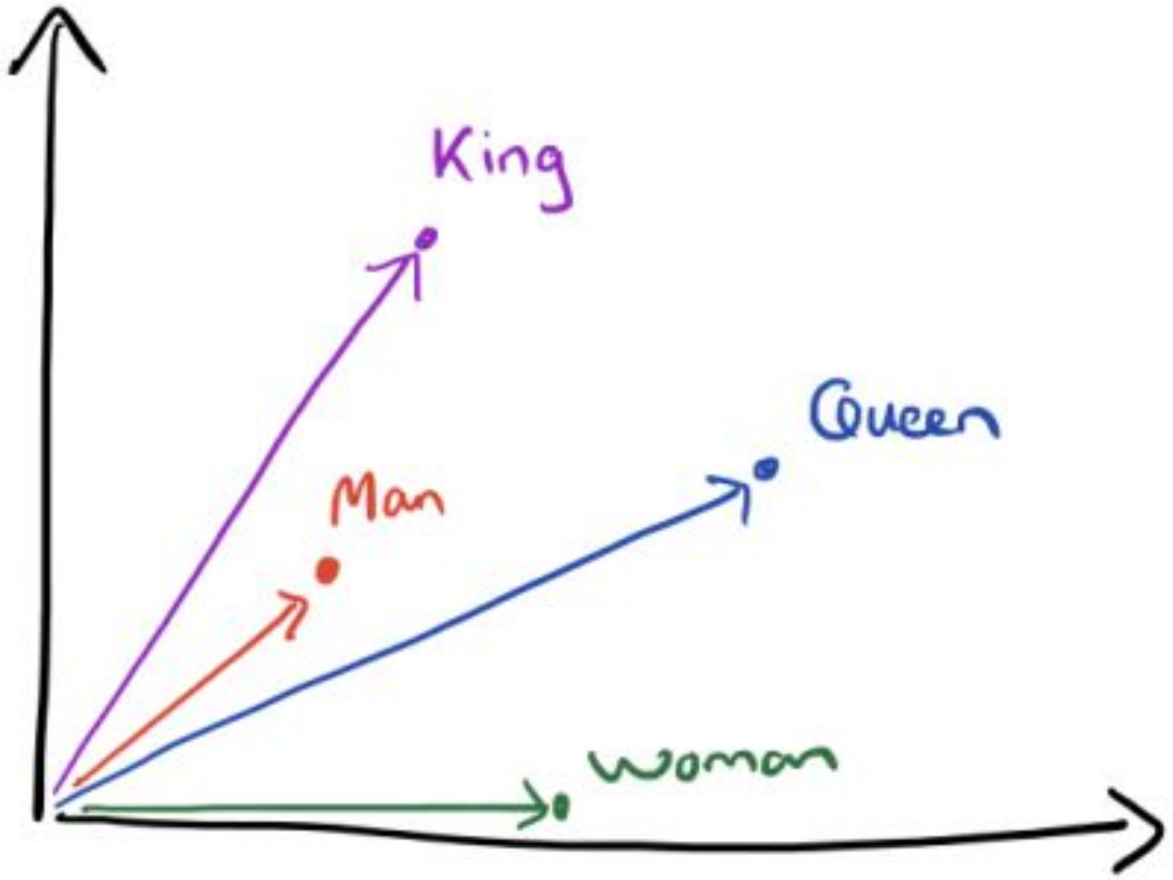
4 Color of coupon (red, black, blue, etc.)

Discount coupon usage

4 Color of coupon (red, black, blue, etc.)

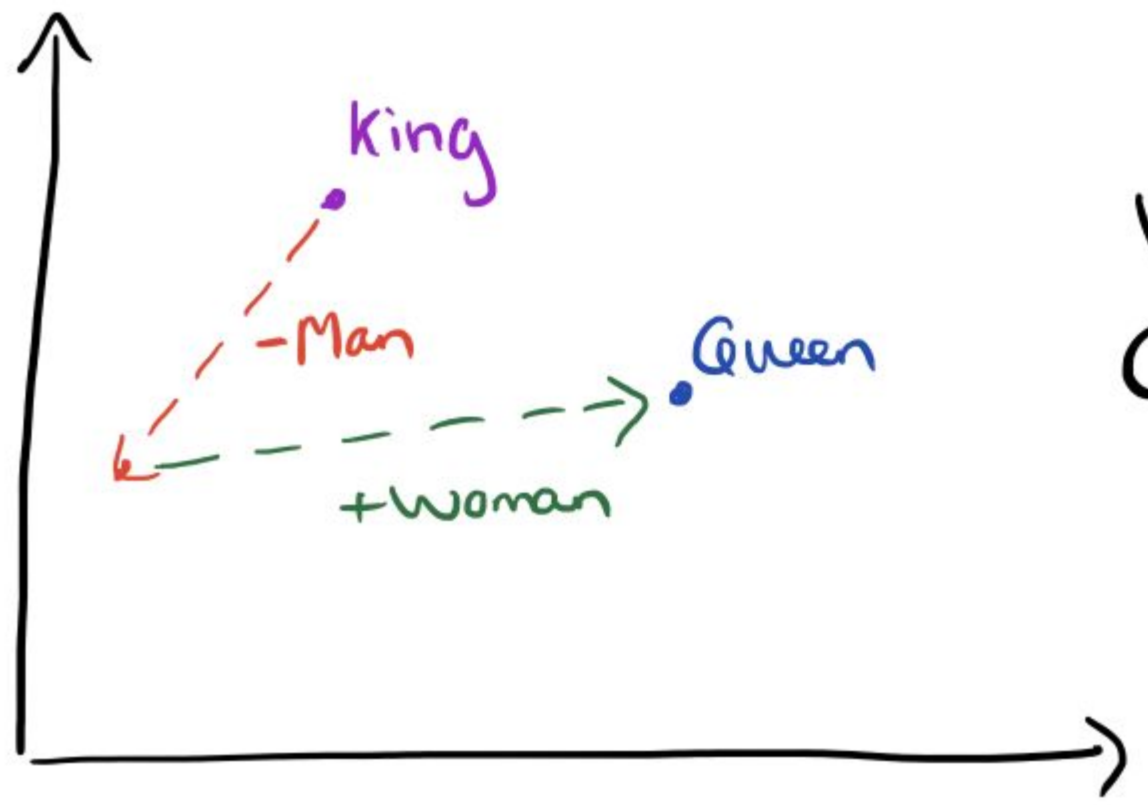
5 Item category (1 for dairy, 2 for deli, 3 for canned goods, etc.)

Word2vec



Word
Vectors

Word2vec



Vector
Composition

Have enough examples

Evan Jones

Avoid having values of which you don't have enough examples


1325

Quiz

Which of these will it be difficult to get enough examples?

Discount coupon usage

1. Percent discount of coupon (20%, 30%, etc.)

Discount coupon usage

1. Percent discount of coupon (20%, 30%, etc.)
2. Date that promotional offer starts

Discount coupon usage

1. Percent discount of coupon (20%, 30%, etc.)
2. Date that promotional offer starts
3. Number of customers who opened advertising email

Predict whether a credit card transaction is fraudulent

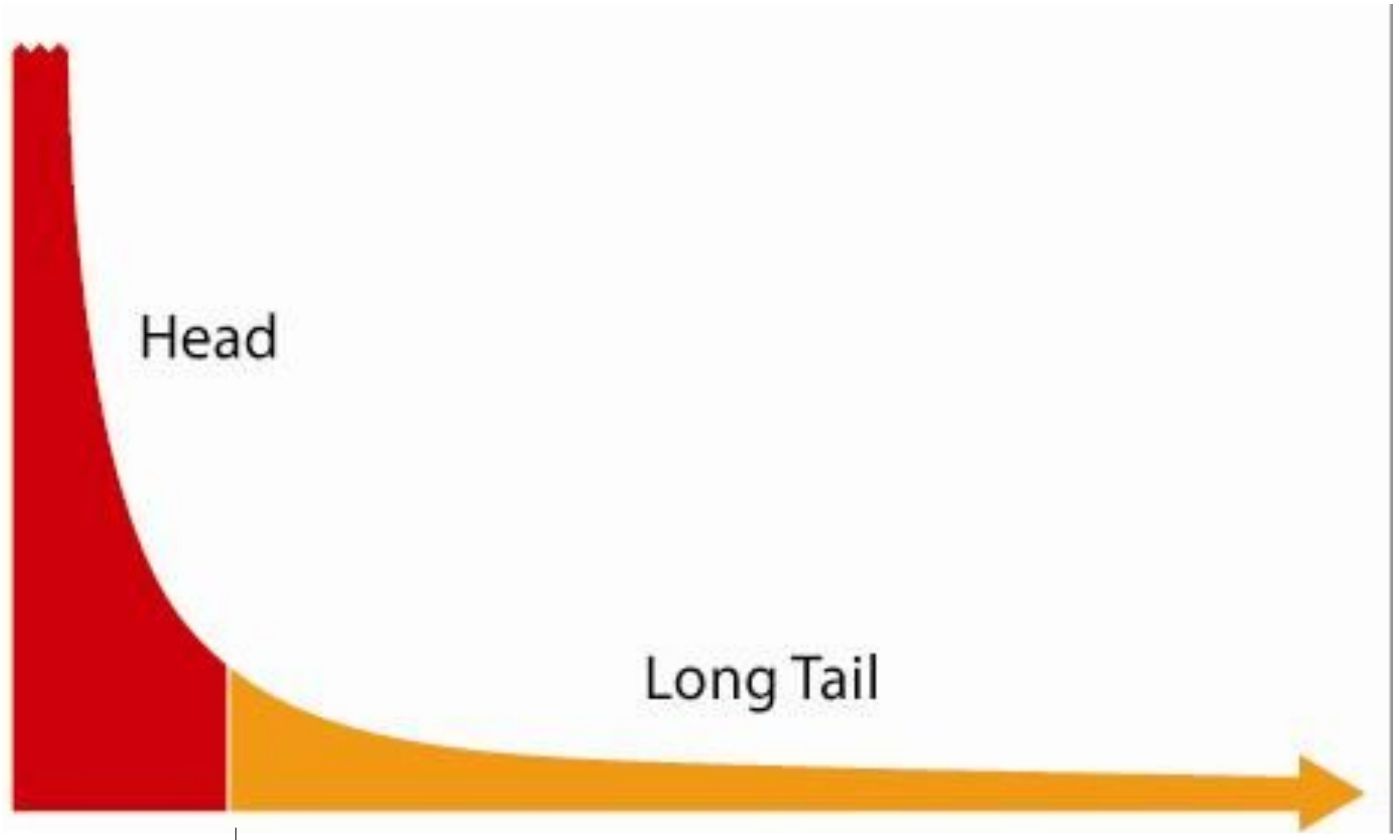
1 Whether cardholder has purchased these items at this store before

Predict whether a credit card transaction is fraudulent

1. Whether cardholder has purchased these items at this store before
2. Distance between cardholder address and store

Predict whether a credit card transaction is fraudulent

1. Whether cardholder has purchased these items at this store before
2. Distance between cardholder address and store



Group all customer over 50+ miles into a single group

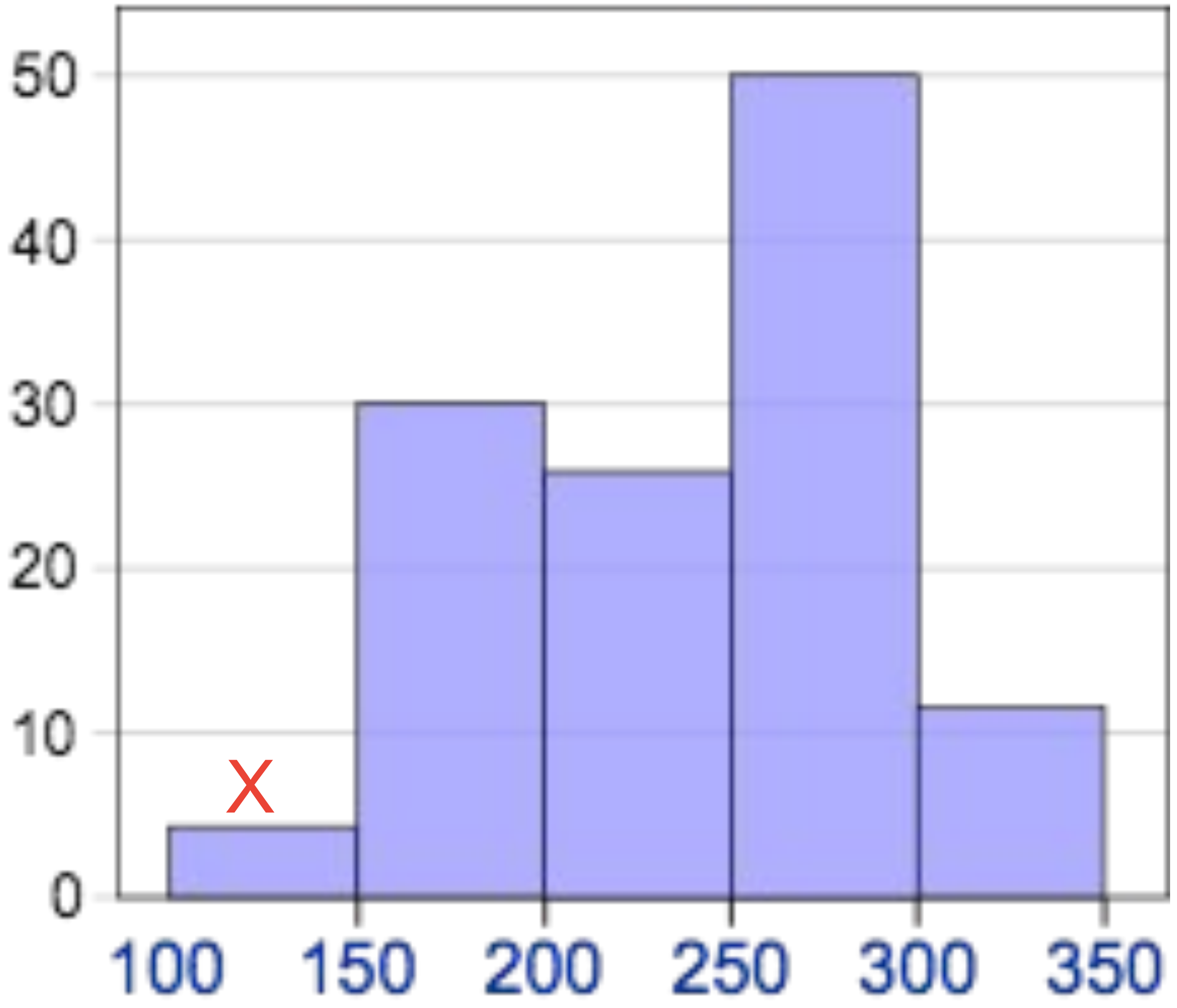
Predict whether a credit card transaction is fraudulent

1. Whether cardholder has purchased these items at this store before
2. Distance between cardholder address and store

Predict whether a credit card transaction is fraudulent

1 Whether cardholder has purchased these items at this store before

2 Distance between cardholder address and store



Predict whether a credit card transaction is fraudulent

1. Whether cardholder has purchased these items at this store before
2. Distance between cardholder address and store
3. Category of item being purchased
4. Online or in-person purchase?

Google Cloud

Bring human insight to problem

Evan Jones

Objectives

Turn raw data to features

Compare good vs bad features

Represent features

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
        }
    },
```

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

Numeric values can be used as-is

```
{
        "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
        }
    },
```

```
[ , 2.50, ..., 1.4,        ]
...
```

```
INPUT_COLUMNS = [
    ...,
tf.feature_column.numeric_column('price'),
    ...
]
```

numeric_column is a type of feature column

Overly specific attributes should be discarded

```
{
        "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
        }
    },
```

Overly specific attributes should be discarded

```
{
        "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
        }
    },
```

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

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

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

Categorical variables should be one-hot encoded

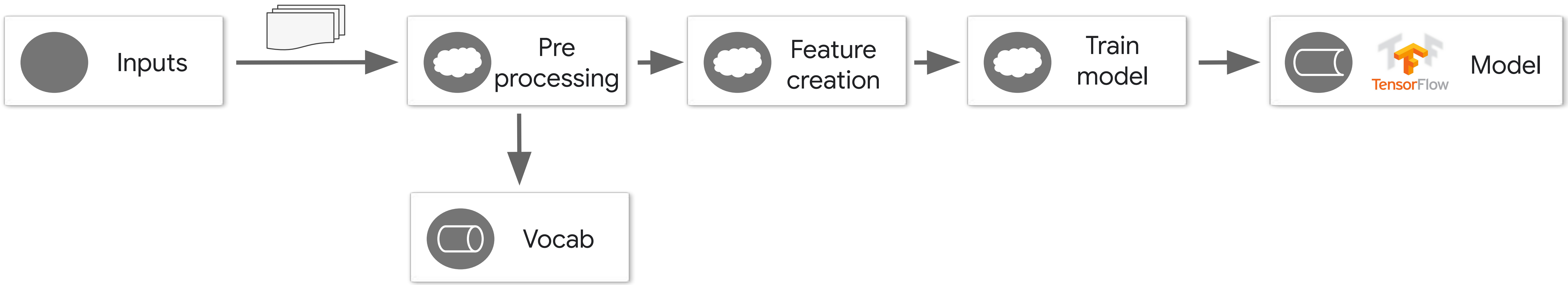
```
{
        "transactionId": 42,
        "name": "Ice Cream",
        "price": 2.50,
        "tags": ["cold", "dessert"],
        "servedBy": {
            "employeeId": 72365,
            "waitTime": 1.4,
            "customerRating": 4
        },
        "storeLocation": {
            "latitude":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

Don’t know the list of keys? Create a vocabulary

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']),
```

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)
```

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)
```

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
        }
    },
```

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
        }
    },
```

```
[..., 4, ...]
```

(OR)

```
[..., 0,0,0,1,0, ...]
```

Don’t mix magic numbers with data

```
{
        "transactionId": 42,
        "name": "Ice Cream",
        "price": 2.50,
        "tags": ["cold", "dessert"],
        "servedBy": {
            "employeeId": 72365,
            "waitTime": 1.4,
            "customerRating": -1
        },
        "storeLocation": {
            "latitude":   35.3,
            "longitude": -98.7
        }
    },
```

Don’t mix magic numbers with data

```
{
        "transactionId": 42,
        "name": "Ice Cream",
        "price": 2.50,
        "tags": ["cold", "dessert"],
        "servedBy": {
            "employeeId": 72365,
            "waitTime": 1.4,
            "customerRating": -1
        },
        "storeLocation": {
            "latitude":   35.3,
            "longitude": -98.7
        }
    },
```

```
[..., 4,1, ...]  # 4
[..., 0,0, ...]  # -1
```

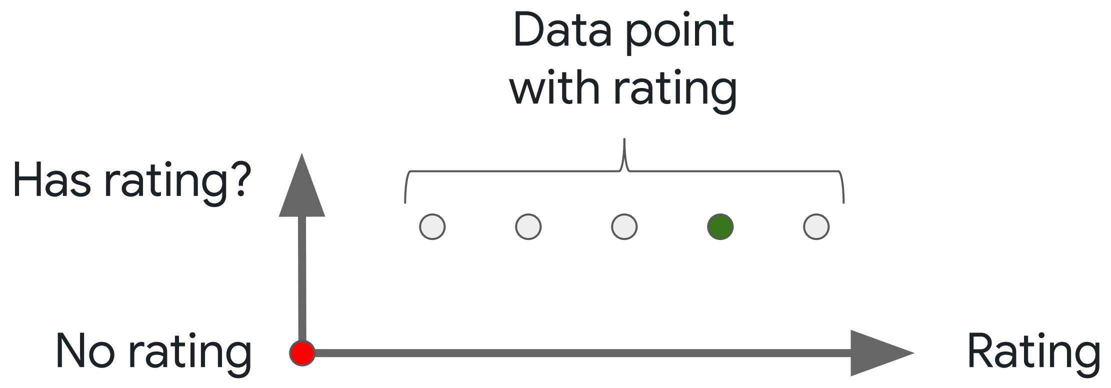
Don’t mix magic numbers with data

```
{
        "transactionId": 42,
        "name": "Ice Cream",
        "price": 2.50,
        "tags": ["cold", "dessert"],
        "servedBy": {
            "employeeId": 72365,
            "waitTime": 1.4,
            "customerRating": -1
        },
        "storeLocation": {
            "latitude":   35.3,
            "longitude": -98.7
        }
    },
```

```
[..., 4,1, ...]  # 4
[..., 0,0, ...]  # -1
```

(OR)

```
[..., 0,0,0,1,1, ...]  # 4
[..., 0,0,0,0,0, ...]  # -1
```



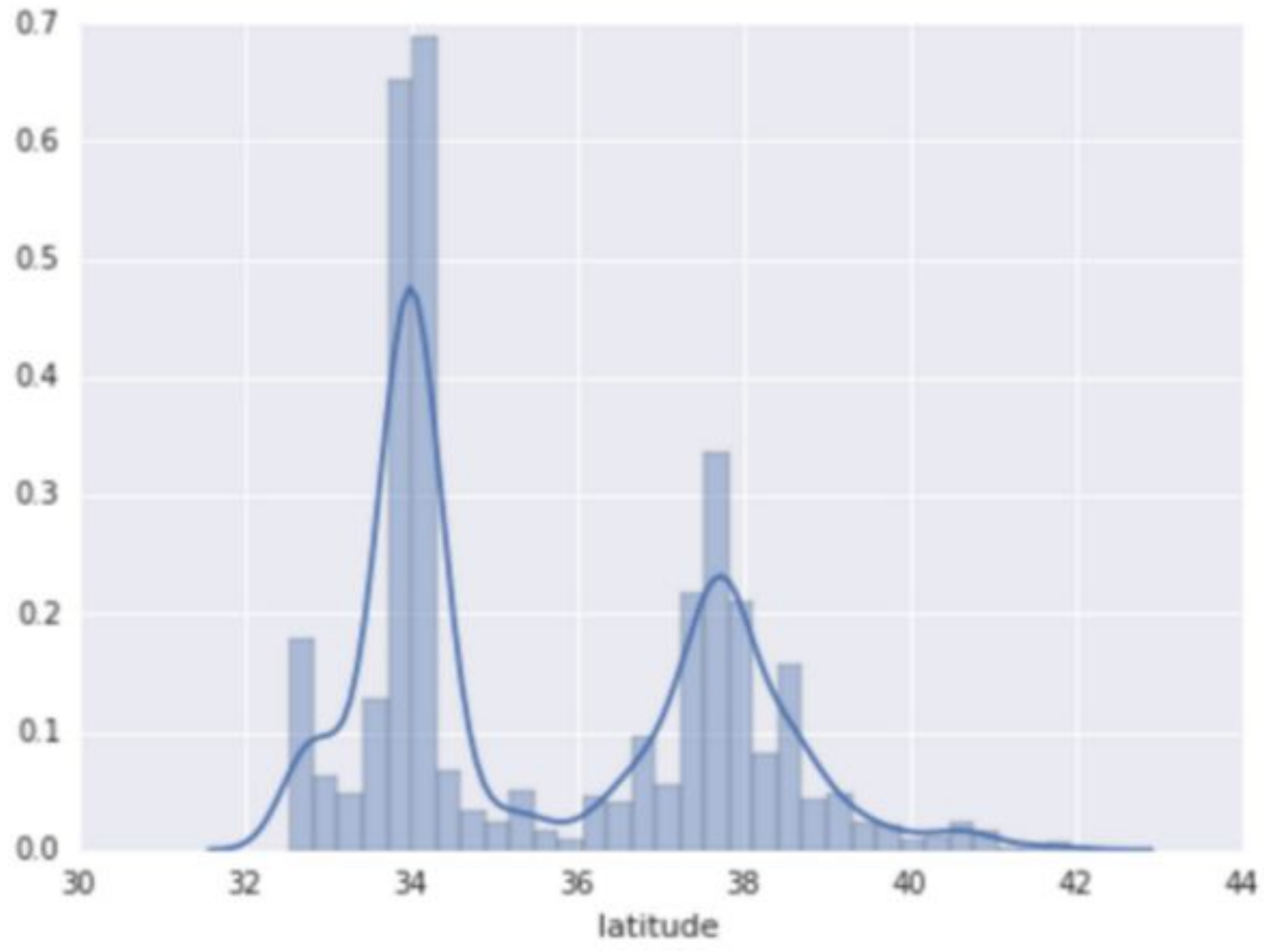
ML vs Statistics

ML = lots of data, keep outliers and build models for them

ML = lots of data, keep outliers and build models for them

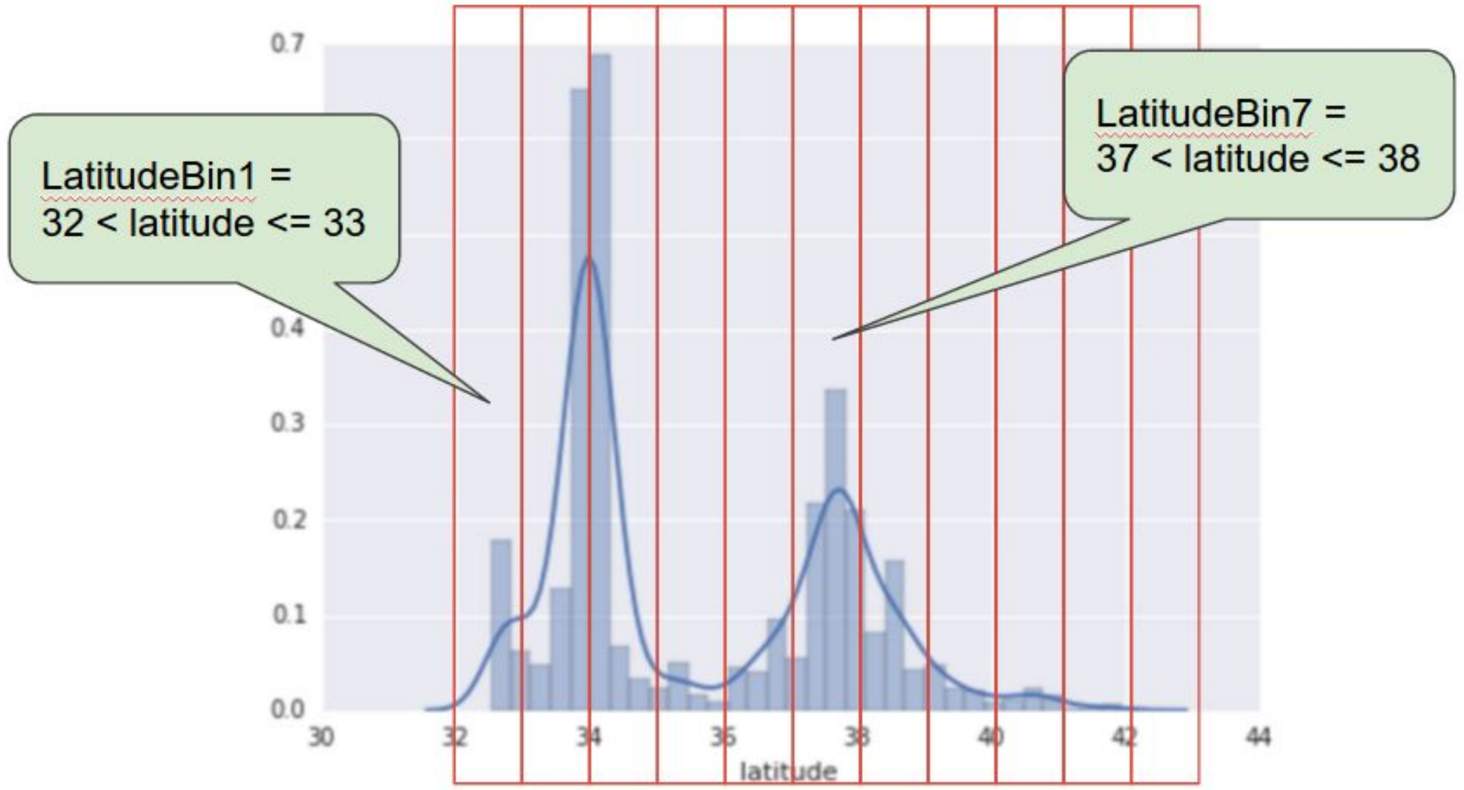
Statistics = “I’ve got all the data I’ll ever get”, throw away outliers

Exact floats are not meaningful



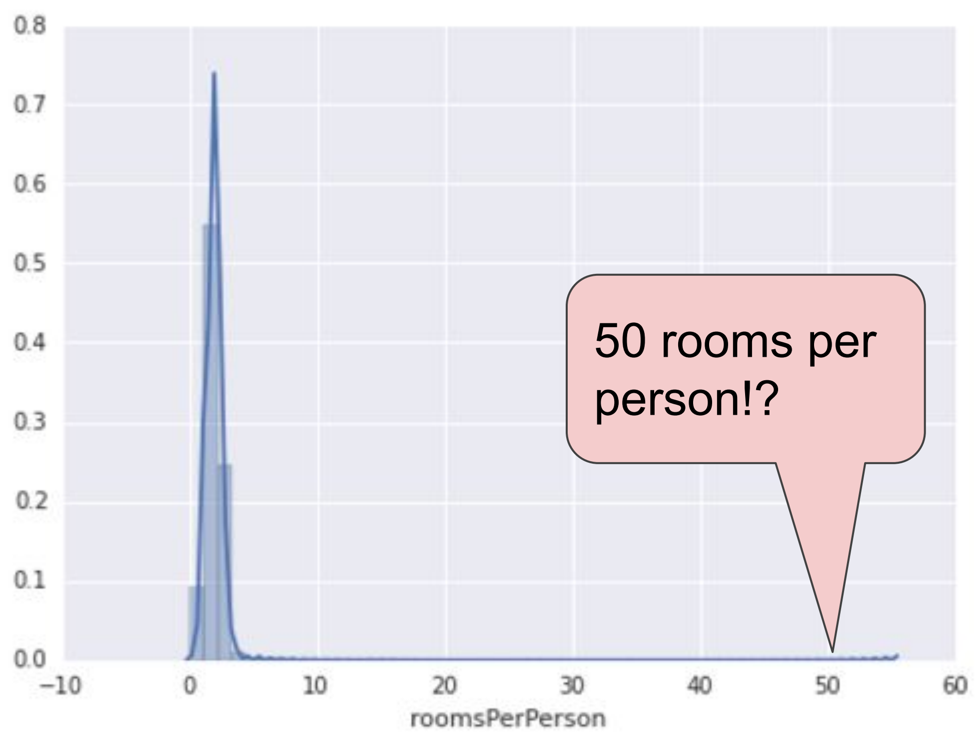
LOS ANGELES SAN FRANCISCO

Discretize floating point values into bins

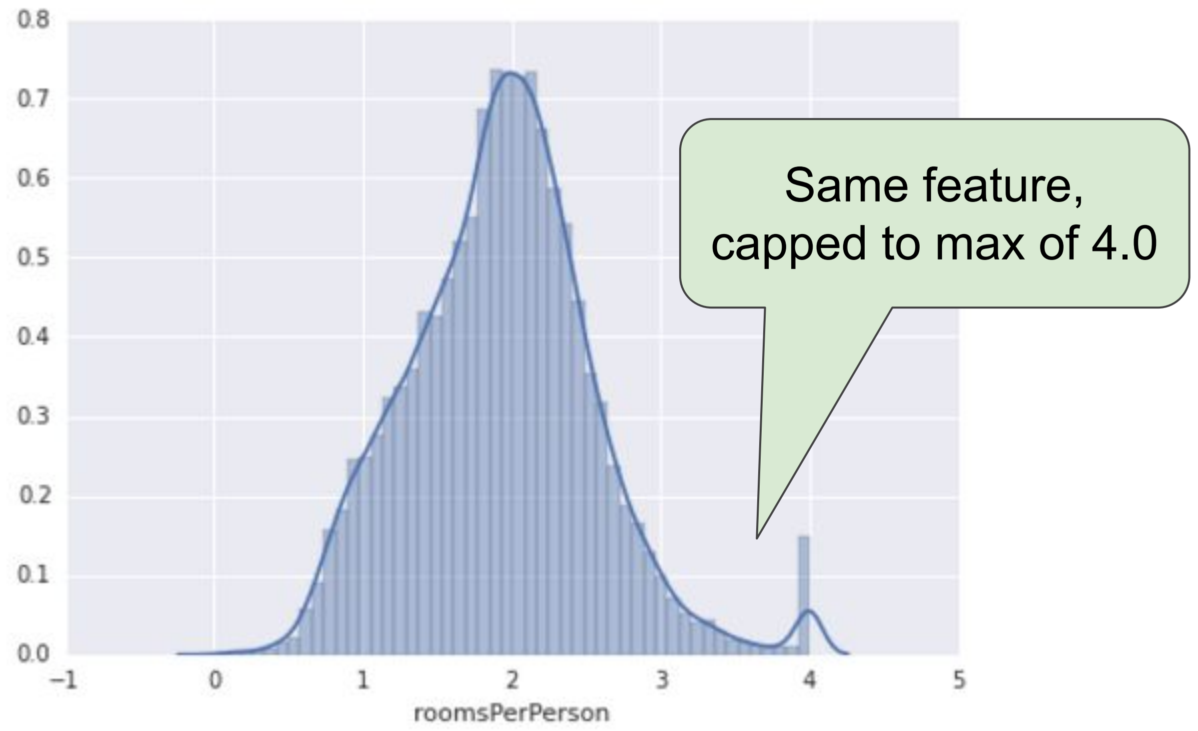



```
lat = tf.feature_column.numeric_column('latitude')
dlat = tf.feature_column.bucketized_column(
   lat, boundaries=np.arange(32,42,1).tolist()
   )
```

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
)
```

Ideally, features should have a similar range

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

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

Lab

Improve the accuracy of a model by adding new features with the appropriate representation