Deep Learning Walkthrough - 01

Cassie Kozyrkov

Chief Decision Scientist, Google Cloud

GitHub: kozyrkov; Twitter: @quaesita

Google Cloud

Step 1 | Define your objective

- State the outcomes
- Pick success metrics
- Set test criteria

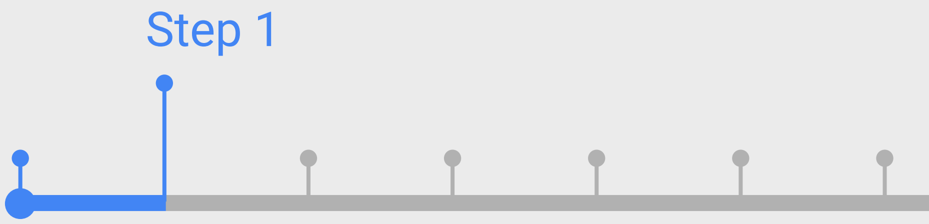
Chief actors:

Decision maker

Domain expert

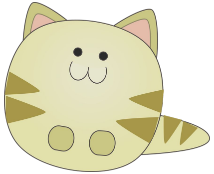
Guide to step 1

A. Write down outputs/labels
B. Consider mistakes
C. Assign project scoring
D. Create performance metric
E. Think about loss function
F. Compare the functions
G. Set performance criteria

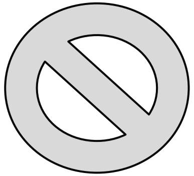


Labels/Outputs

What do you want your system to output for you?

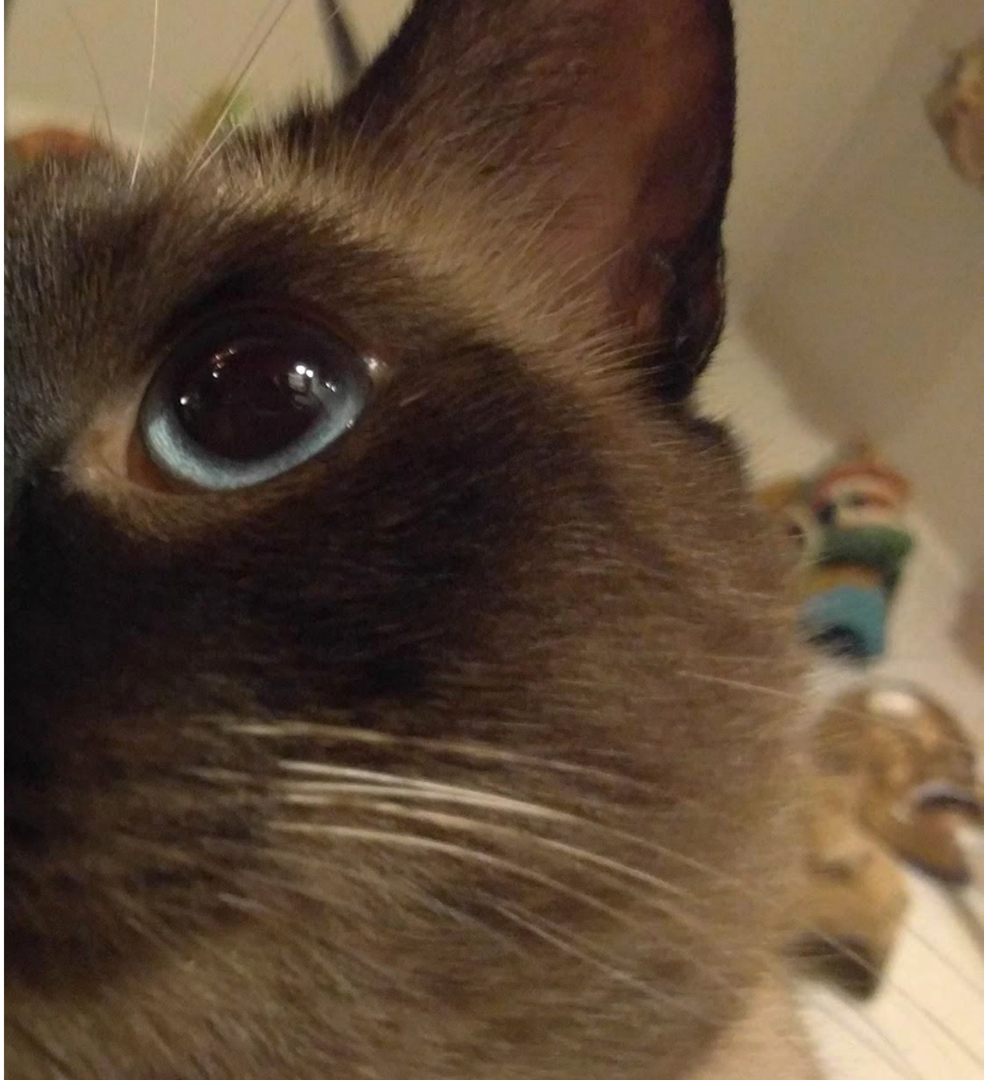


Cat



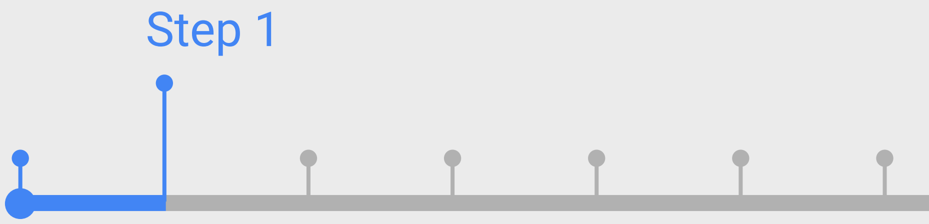
Not cat

Google Cloud



Guide to step 1

A. Write down outputs/labels
B. Consider mistakes
C. Assign project scoring
D. Create performance metric
E. Think about loss function
F. Compare the functions
G. Set performance criteria



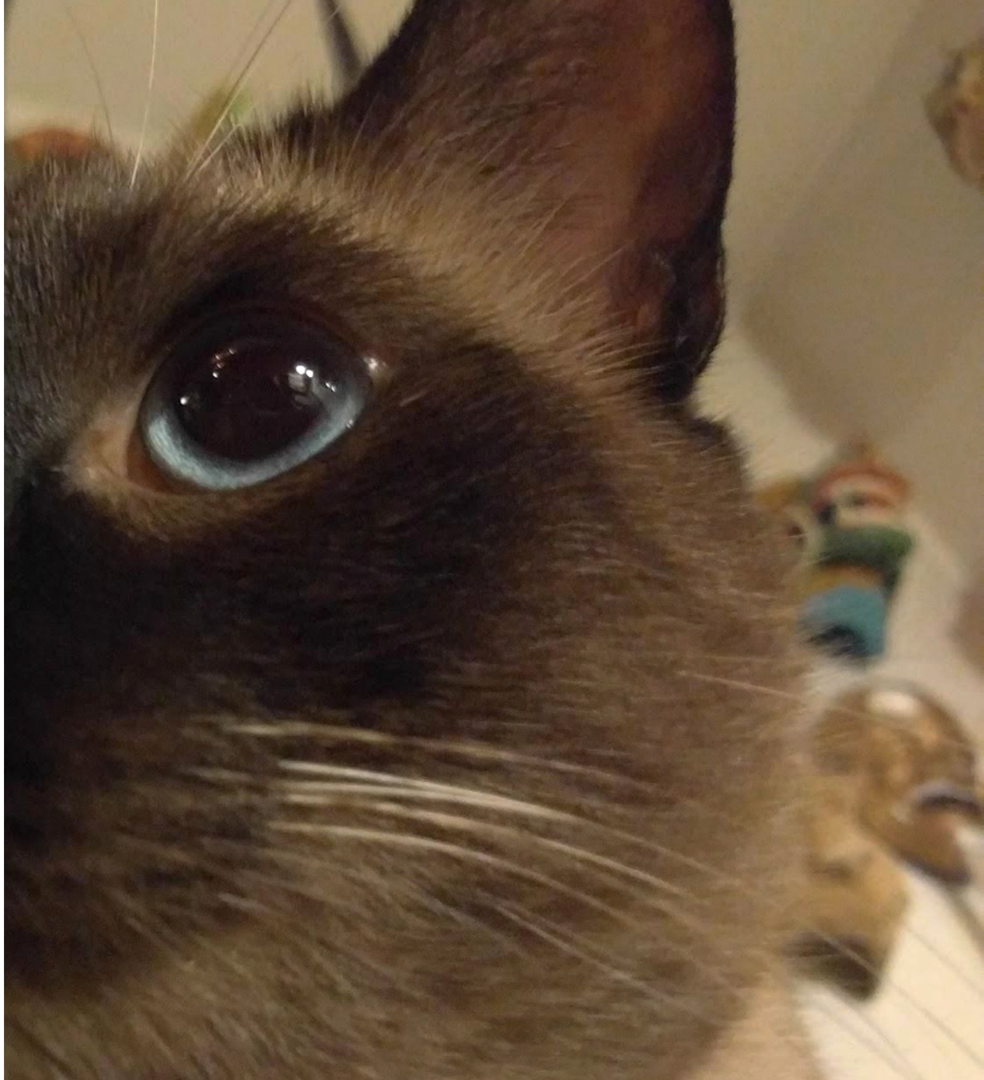
Decision-maker is boss

The decision maker is responsible for deciding which mistakes are more important for the business and for choosing how to calculate the score.

- All outputs equally important?
- All mistakes equally bad?

Labels/Outputs

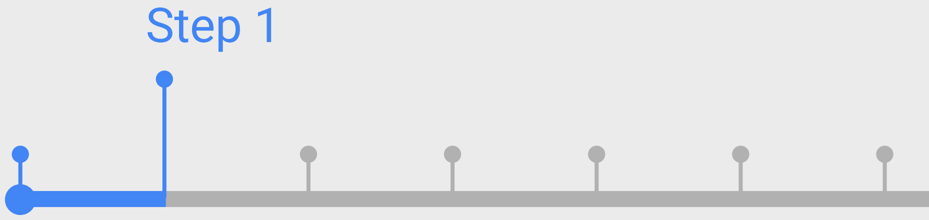
Our decision-maker says all mistakes are equally bad for this project.



Google Cloud

Guide to step 1

A. Write down outputs/labels
B. Consider mistakes
C. Assign project scoring
D. Create performance metric
E. Think about loss function
F. Compare the functions
G. Set performance criteria

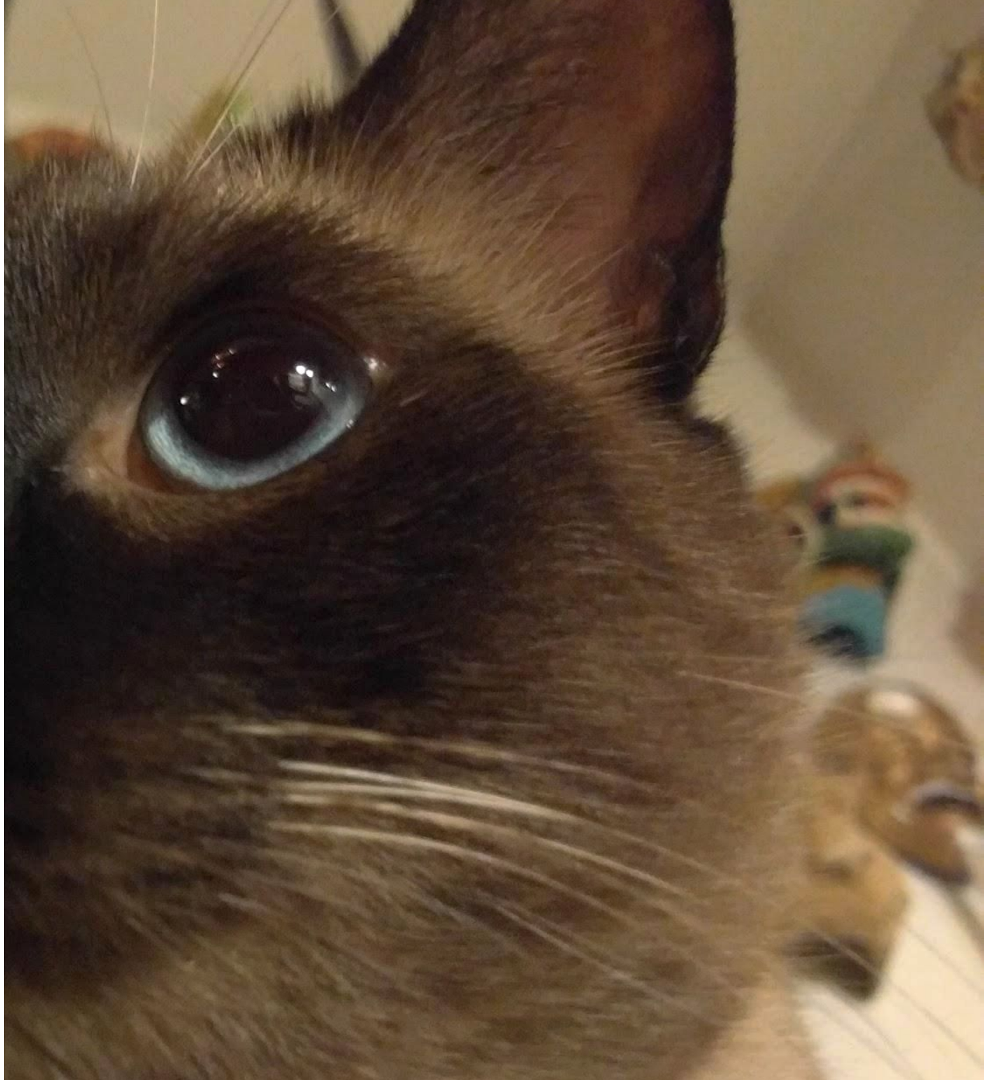


Google Cloud

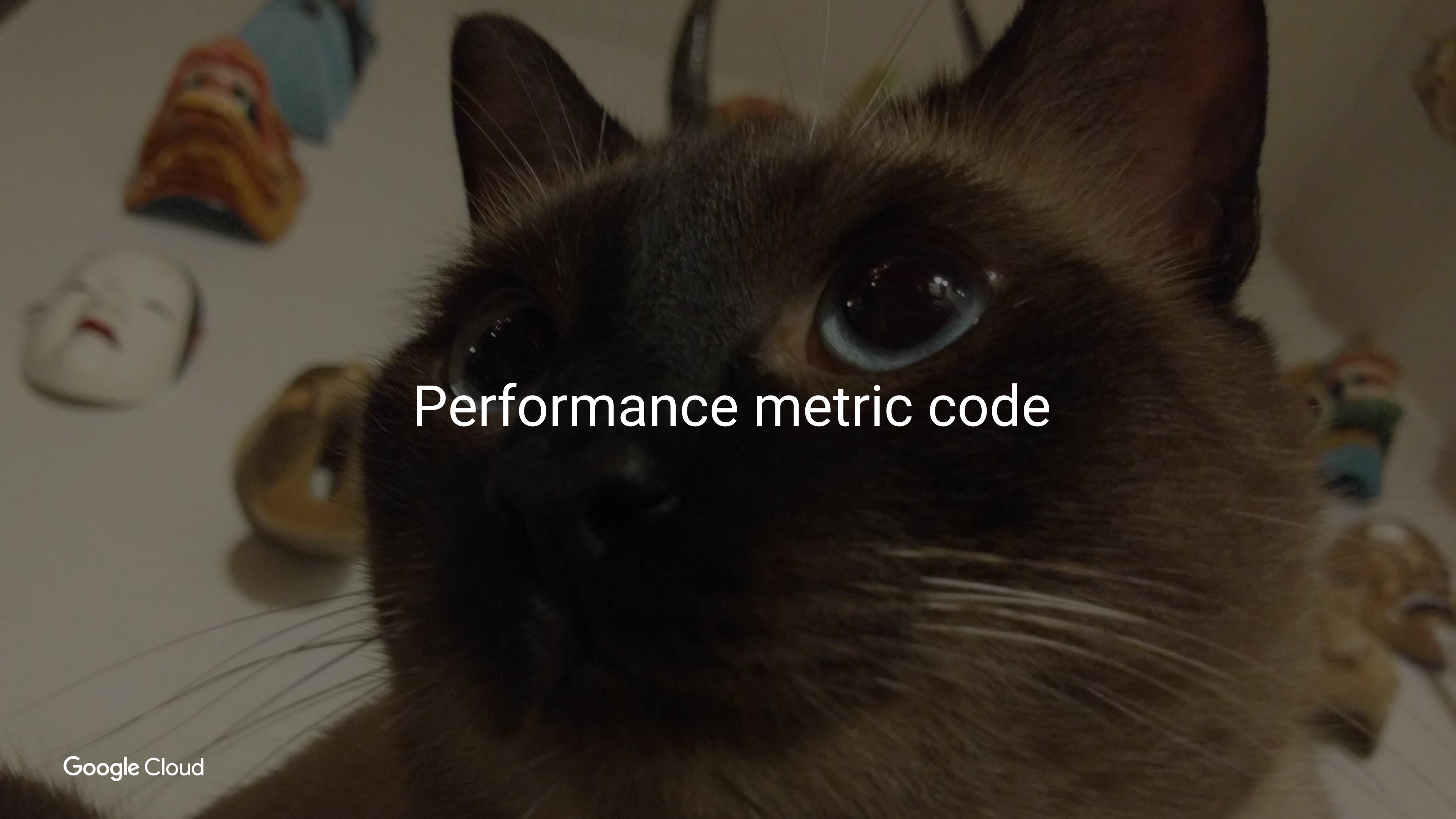
Performance metric

If we consider all mistakes to be equally bad, our metric is:

- Accuracy



Google Cloud



Performance metric code

Google Cloud

jupyter step_1_to_3 Last Checkpoint: a few seconds ago (autosaved)

Logout

File Edit View Insert Cell Kernel Help

Trusted Python 2

Run Code

Performance Metric and Requirements

Author(s): kozyr@google.com

Before we get started on data, we have to choose our project performance metric and decide the statistical testing criteria. We'll make use of the metric code we write here when we get to Step 6 (Training) and we'll use the criteria in Step 9 (Testing).

In [1]:
```
# Required libraries:
import numpy as np
import pandas as pd
import seaborn as sns
```

Performance Metric: Accuracy

We've picked accuracy as our performance metric.

$$\text{Accuracy} = \frac{\text{correct predictions}}{\text{total predictions}}$$

Performance Metric: Accuracy

We've picked accuracy as our performance metric.

$$\text{Accuracy} = \frac{\text{correct predictions}}{\text{total predictions}}$$

In [2]:

```
# Accuracy metric:
def get_accuracy(truth, predictions, threshold=0.5, roundoff=2):
  """
  Args:
    truth: can be Boolean (False, True), int (0, 1), or float (0, 1)
    predictions: number between 0 and 1, inclusive
    threshold: we convert predictions to 1s if they're above this value
    roundoff: report accuracy to how many decimal places?

  Returns:
    accuracy: number correct divided by total predictions
  """

  truth = np.array(truth) == (1|True)
  predicted = np.array(predictions) >= threshold
  matches = sum(predicted == truth)
  accuracy = float(matches) / len(truth)
  return round(accuracy, roundoff)
```

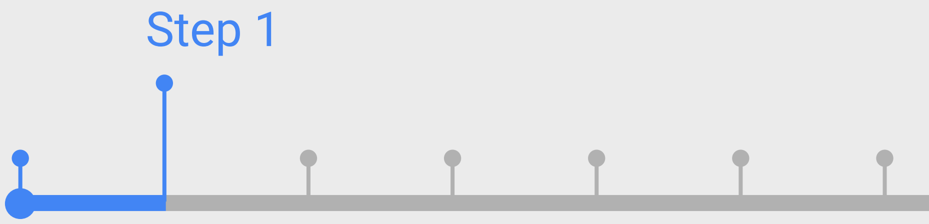
In [3]:

```
# Try it out:
acc = get_accuracy(truth=[0, False, 1], predictions=[0.2, 0.7, 0.6])
print 'Accuracy is ' + str(acc) + '.'
```

```
Accuracy is 0.67.
```

Guide to step 1

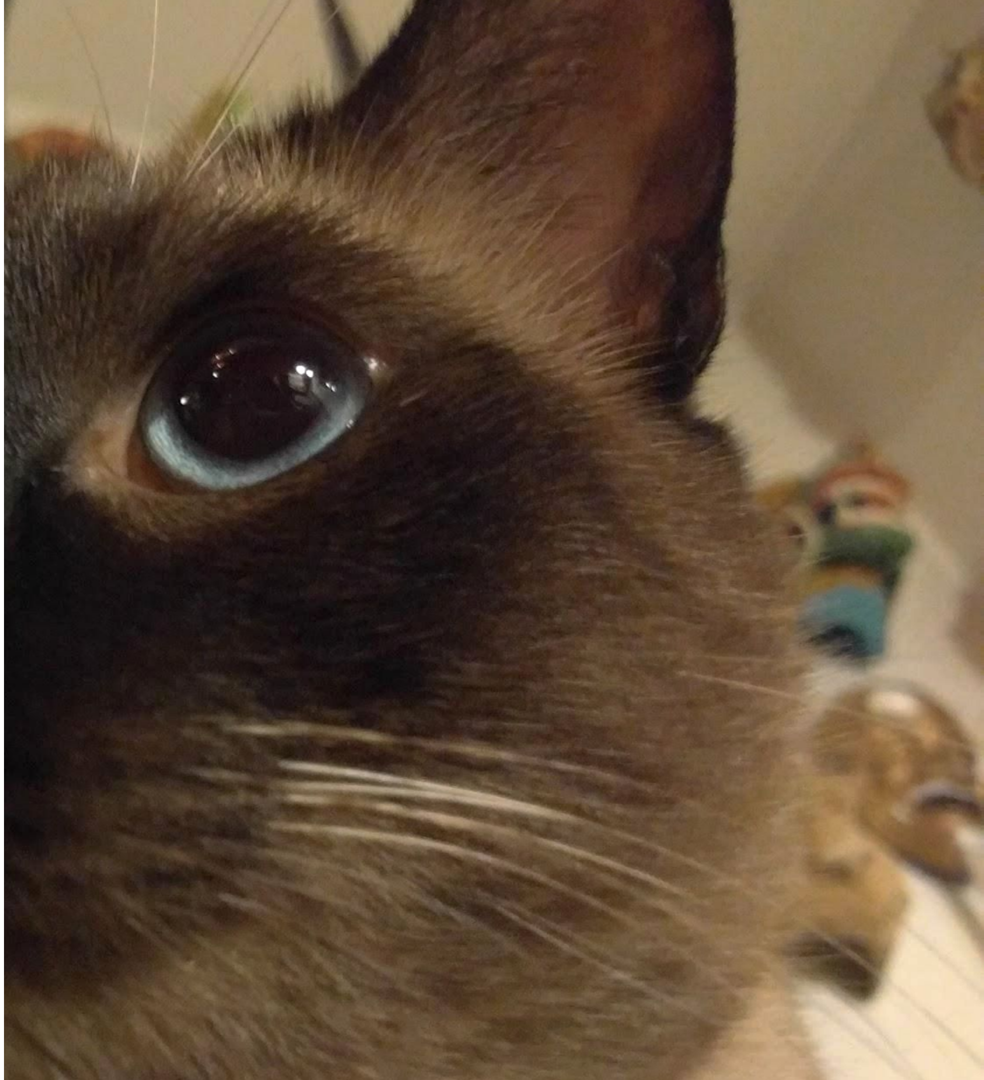
A. Write down outputs/labels
B. Consider mistakes
C. Assign project scoring
D. Create performance metric
E. **Think about loss function**
F. Compare the functions
G. Set performance criteria



Loss function

We'll be minimizing:

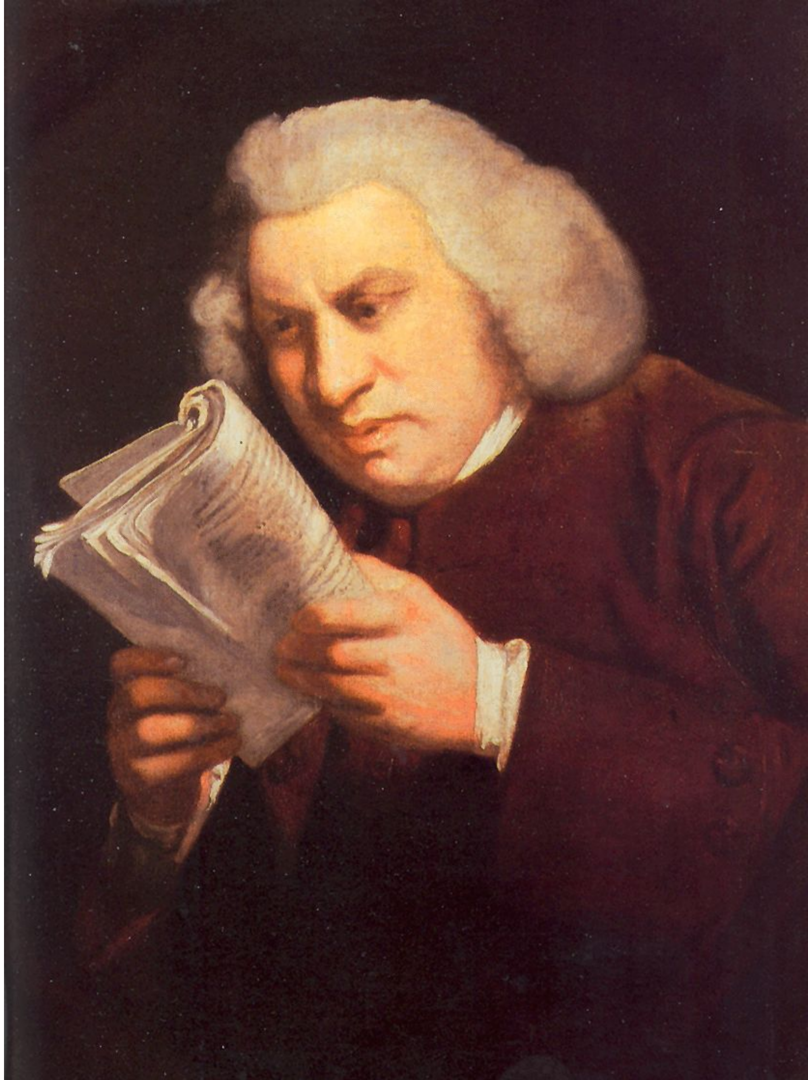
- Cross-entropy loss



Google Cloud

Cross-entropy loss?

$$H(y, \hat{y}) = \sum_i y_i \log \frac{1}{\hat{y}_i} = -\sum_i y_i \log \hat{y}_i$$



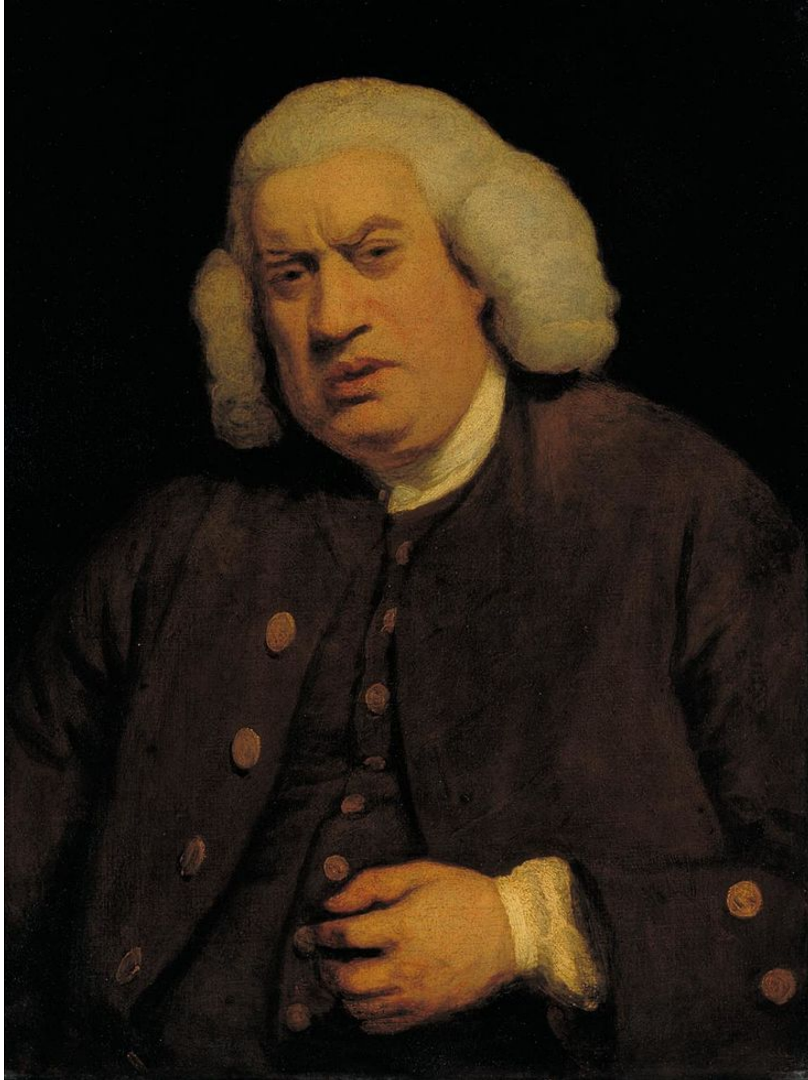
Google Cloud

Cross-entropy loss

"Are your 1s near 1?"

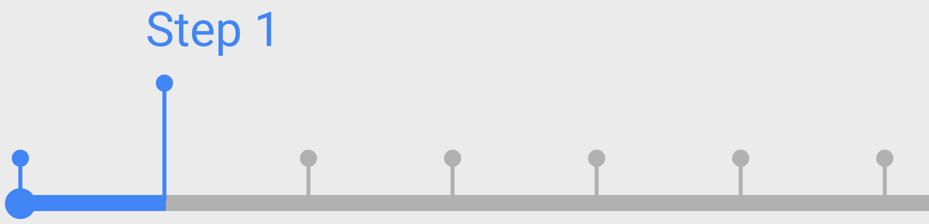
-log(1) = 0 → no penalty
-log(0) = inf → big penalty

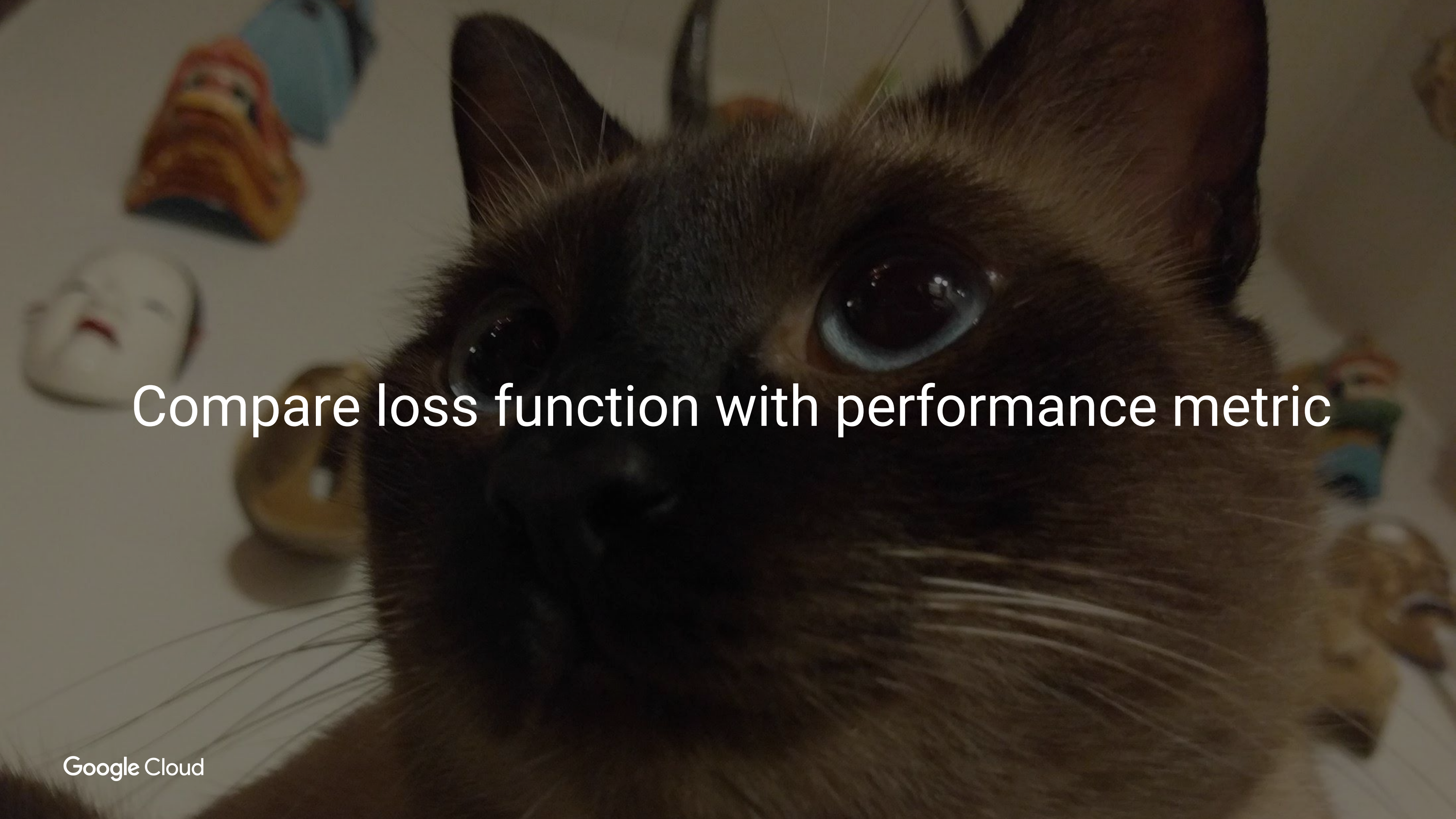
$$H(y, \hat{y}) = \sum_i y_i \log \frac{1}{\hat{y}_i} = -\sum_i y_i \log \hat{y}_i$$



Guide to step 1

A. Write down outputs/labels
B. Consider mistakes
C. Assign project scoring
D. Create performance metric
E. Think about loss function
F. Compare the functions
G. Set performance criteria





Compare loss function with performance metric

Compare Loss Function with Performance Metric

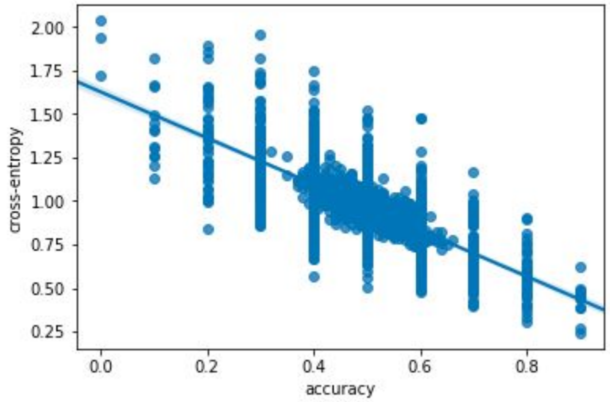
In [4]:
```
def get_loss(predictions, truth):
  # Our methods will be using cross-entropy loss.
  return -np.mean(truth * np.log(predictions) + (1 - truth) * np.log(1 - predictions))
```

In [5]:
```
# Simulate some situations:
loss = []
acc = []
for i in range(1000):
    for n in [10, 100, 1000]:
        p = np.random.uniform(0.01, 0.99, (1, 1))
        y = np.random.binomial(1, p, (n, 1))
        x = np.random.uniform(0.01, 0.99, (n, 1))
        acc = np.append(acc, get_accuracy(truth=y, predictions=x, roundoff=6))
        loss = np.append(loss, get_loss(predictions=x, truth=y))

df = pd.DataFrame({'accuracy': acc, 'cross-entropy': loss})
```

In [6]:
```
# Visualize with Seaborn
import seaborn as sns
%matplotlib inline
sns.regplot(x="accuracy", y="cross-entropy", data=df)
```

Out[6]: <matplotlib.axes._subplots.AxesSubplot at 0x7f3f94c59650>



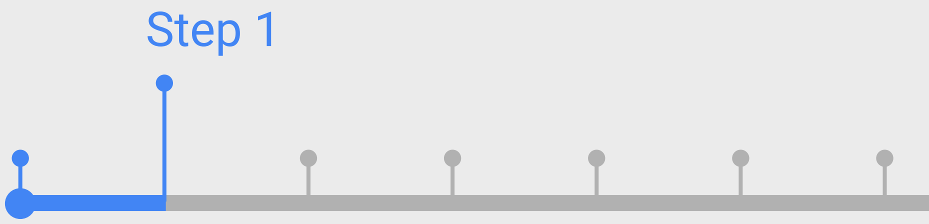
Evaluation

In ML, the proof of the pudding (model) is always in the eating (performance on new data).

Always evaluate performance based on your business metric.

Guide to step 1

A. Write down outputs/labels
B. Consider mistakes
C. Assign project scoring
D. Create performance metric
E. Think about loss function
F. Compare the functions
G. Set performance criteria

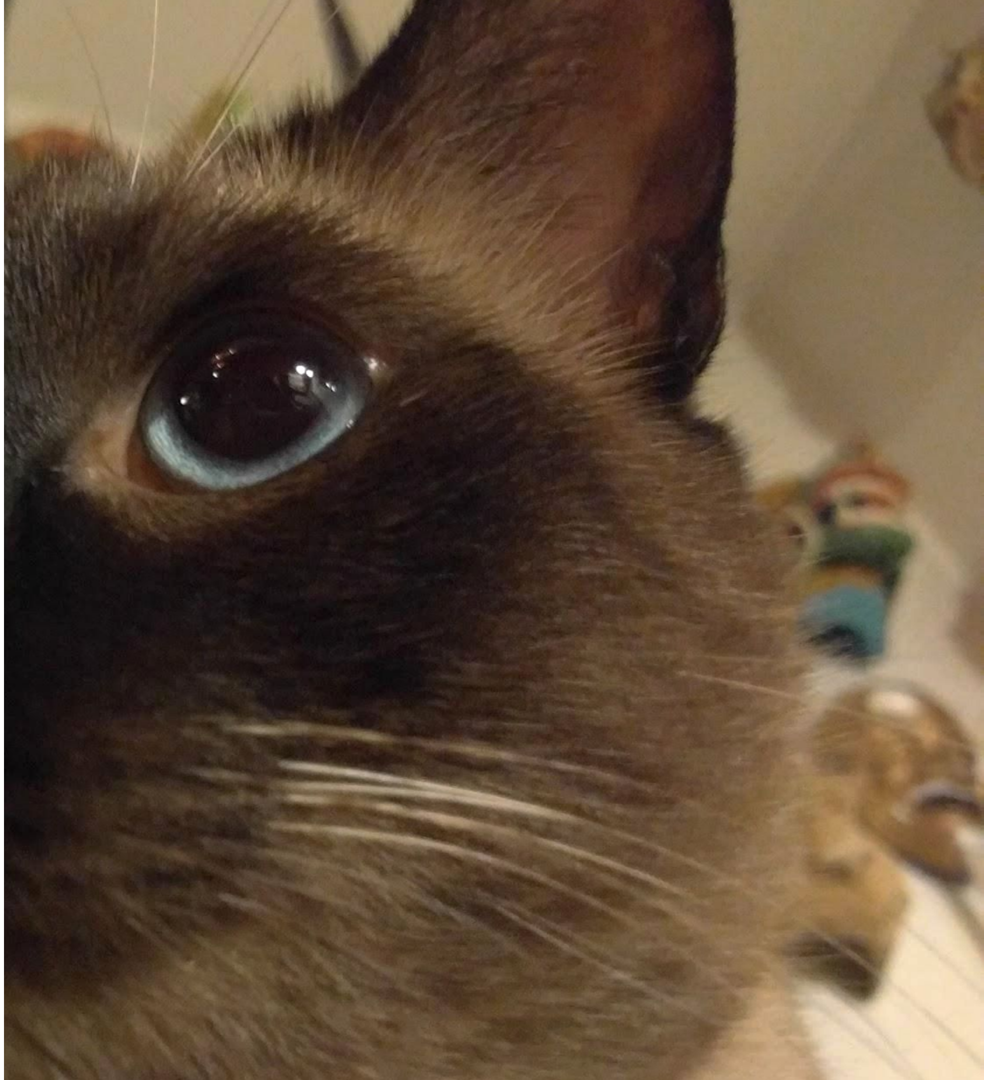


Set performance criteria

What's the minimum project performance you'll accept

- to productionize?
- to launch?

Decide now



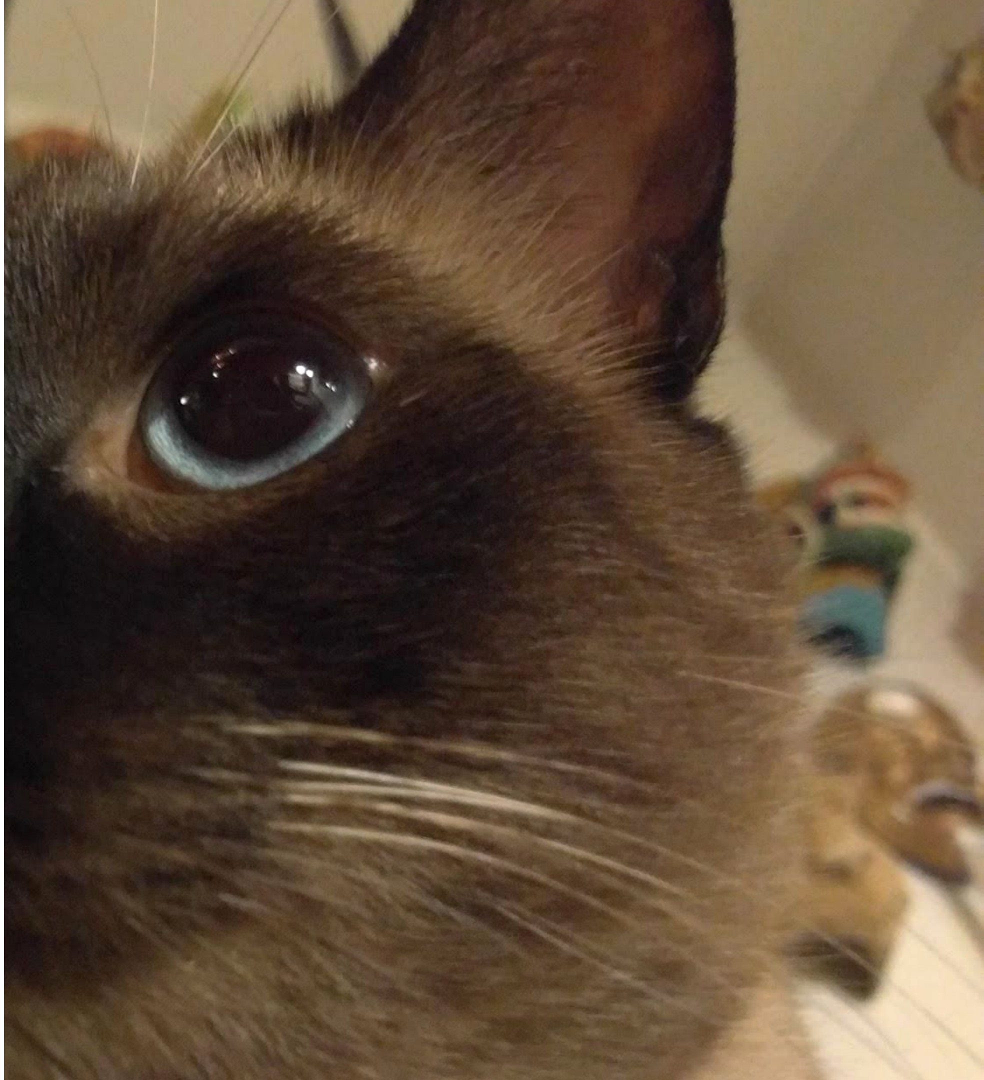
Set performance criteria

What's the minimum project performance you'll accept

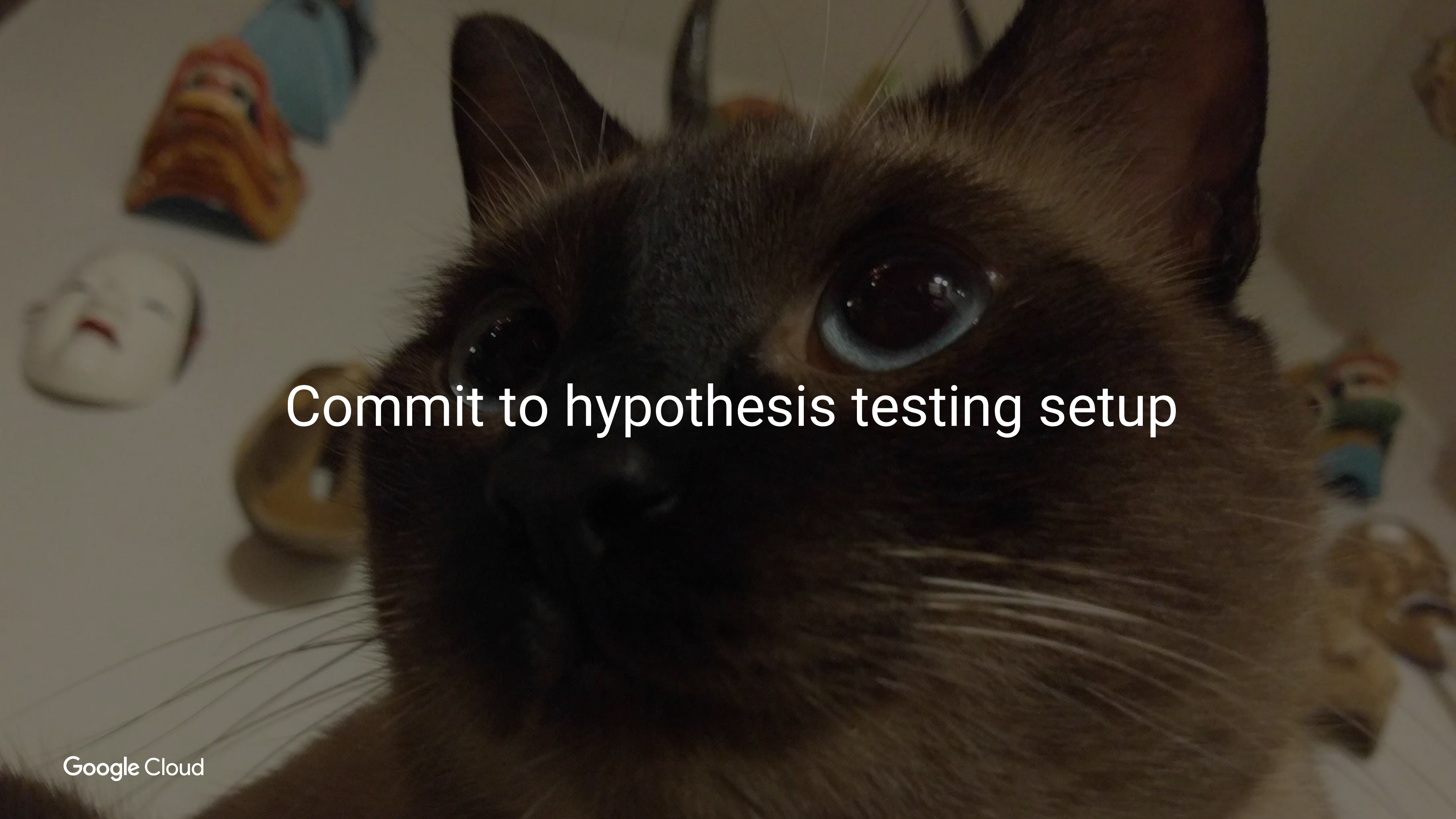
- to productionize?
- to launch?

Minimum accuracy requirement:

80%



Google Cloud


Commit to hypothesis testing setup
Google Cloud

Hypothesis Testing Setup

In [7]:
```
# Testing setup:
SIGNIFICANCE_LEVEL = 0.05
TARGET_ACCURACY = 0.80

# Hypothesis test we'll use:
from statsmodels.stats.proportion import proportions_ztest
```

In [8]:
```
# Using standard notation for a one-sided test of one population proportion:
n = 100  # Example number of predictions
x = 95  # Example number of correct predictions
p_value = proportions_ztest(count=x, nobs=n, value=TARGET_ACCURACY, alternative='larger')[1]
if p_value < SIGNIFICANCE_LEVEL:
  print 'Congratulations! Your model is good enough to build. It passes testing. Awesome!'
else:
  print 'Too bad.  Better luck next project.  To try again, you need a pristine test dataset.'
```

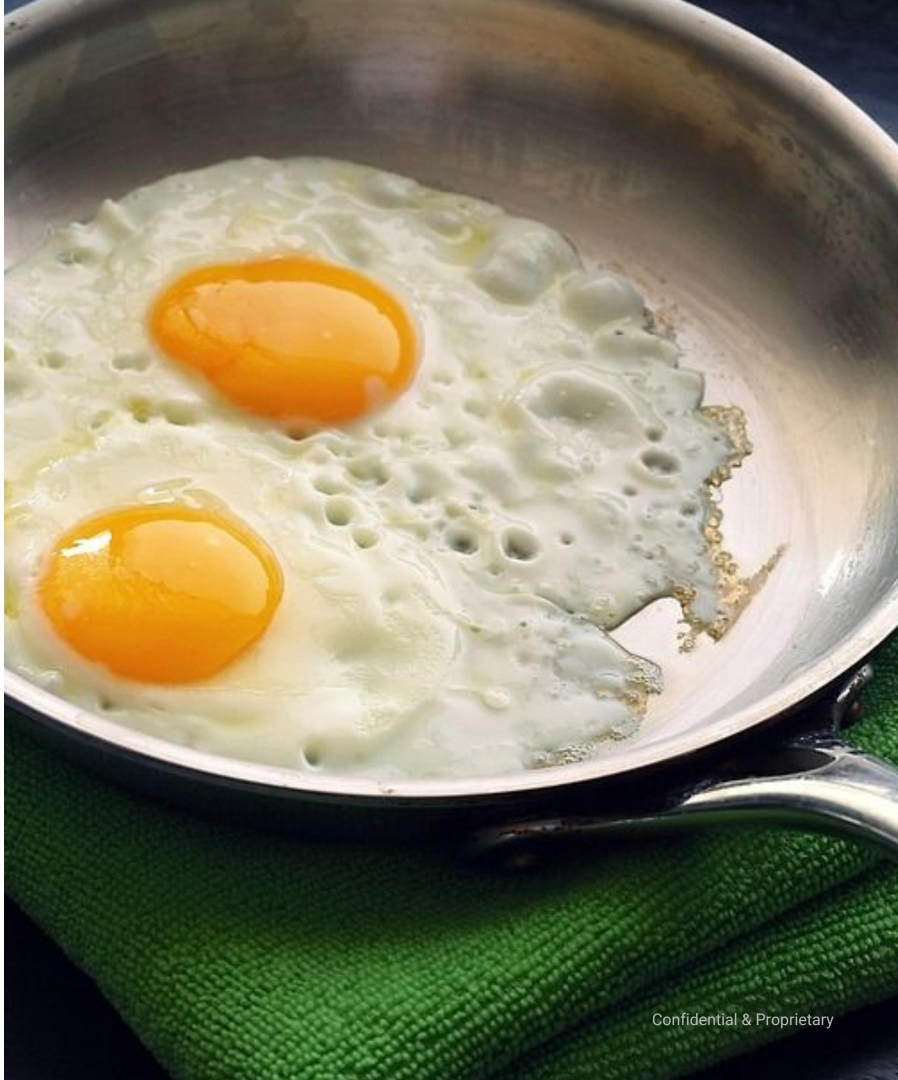
```
Congratulations! Your model is good enough to build. It passes testing. Awesome!
```

In practice

!

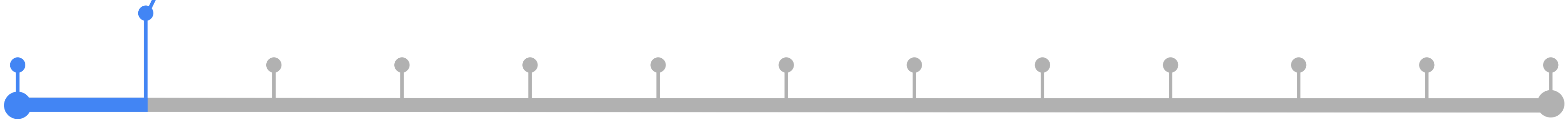
When thinking about launch criteria, you should consider a few more things:

- What are you comparing?
  - ML vs next best recipe
- Do you need live testing?
- Is the live metric different?



Step 1 is finished | You now have a document articulating:

- Labels / mistakes
- Performance metric
- Minimum required performance



Chief actors:

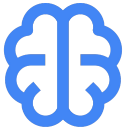
Decision maker

Domain expert

Google Cloud

Step 1 is finished | You now have a document articulating:

Chief actors:



Decision maker

Domain expert