Counterfactual evaluation of machine learning models

Michael Manapat
@mlmanapat
Stripe

About me

- Distant Past: Graduate student and Postdoctoral Fellow in Math
- Previously: Engineer at Google
- Now: Manager of the Machine Learning team at Stripe (10 engineers and data scientists as of 8/2015)

mlm@stripe.com | @mlmanapat

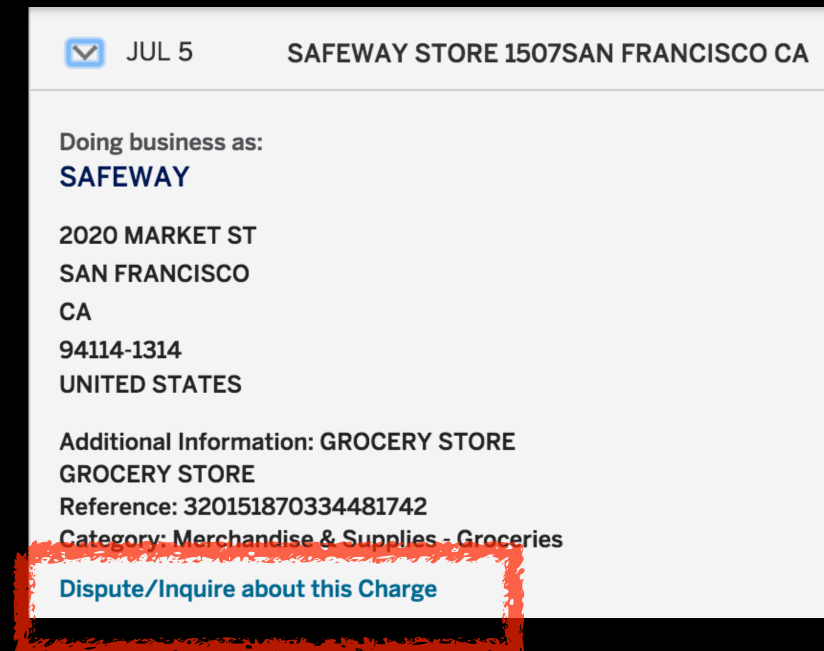
About Stripe

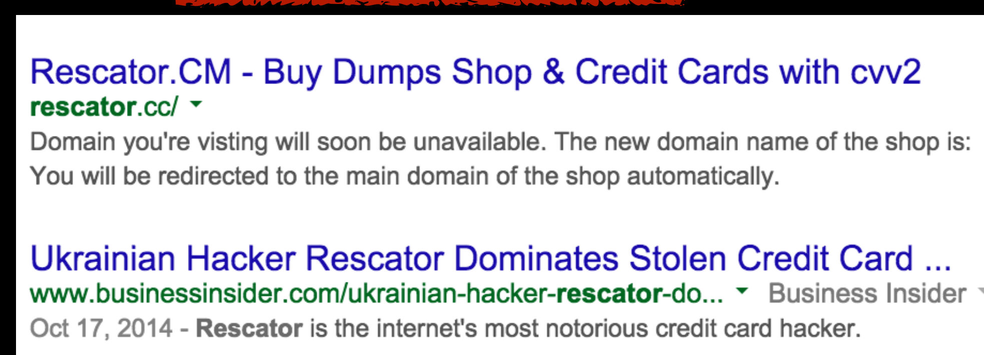
- Unified tools and APIs for accepting and managing payments online

```
stripe.Charge.create(
  amount=40000,
  currency="usd",
  source={
    "number": '4242424242424242',
    "exp_month": 12,
    "exp_year": 2016,
    "cvc": "123"
  }
)
```

Charge Outcomes

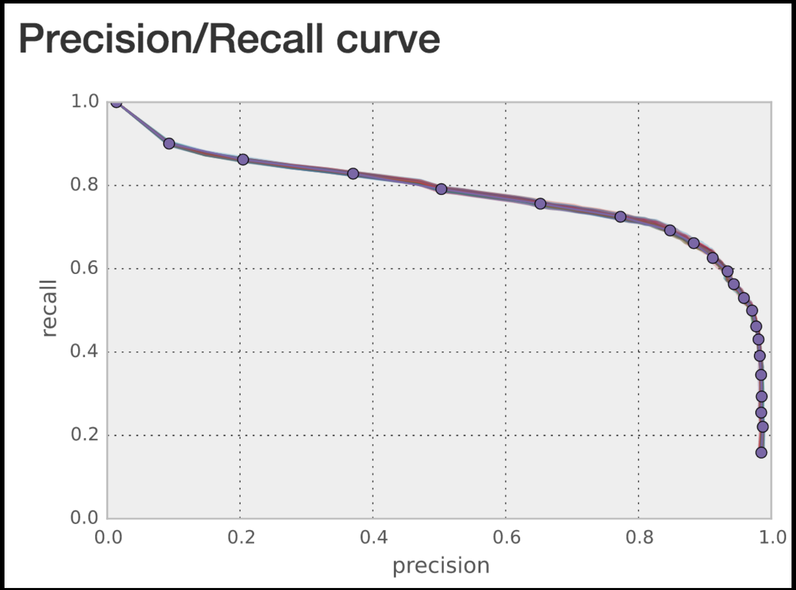
- Nothing
- Refunded
- Disputed ("charged back")
    - "Card testing"
    - "Card cashing"
    - Can take > 60 days to get raised

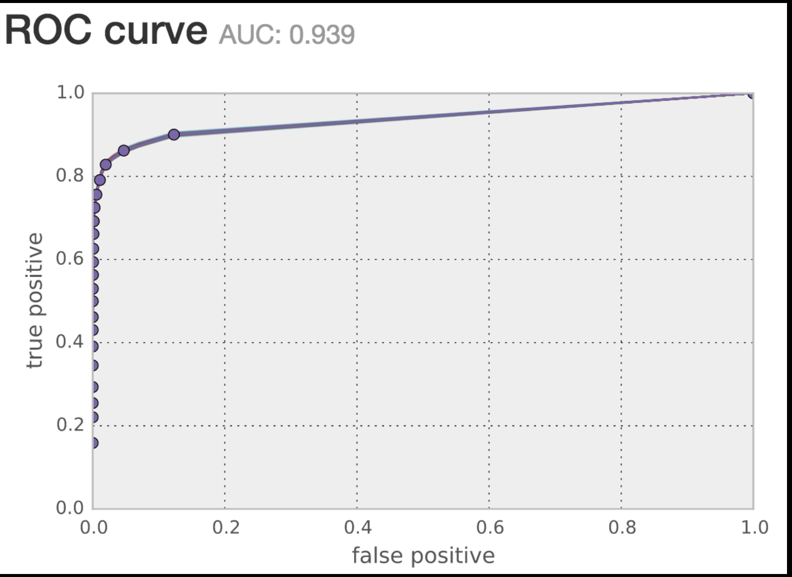




Model Building

- December 31st, 2013
  - Train a **binary classifier** for **disputes** on data from Jan 1st to Sep 30th
  - Validate on data from Oct 1st to Oct 31st (need to wait ~60 days for labels)
- Based on validation data, pick a policy for actioning scores: block if **score > 50**



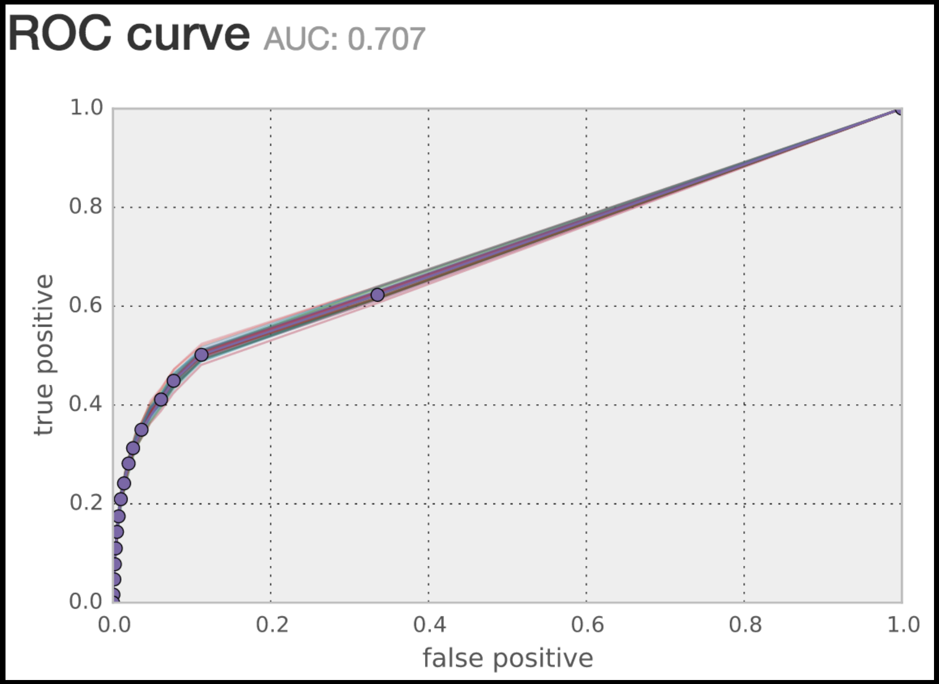


Questions

- Validation data is > 2 months old. How is the model doing?
- What are the **production** precision and recall?
- Business complains about high false positive rate: what would happen if we changed the policy to "block if **score > 70**"?

Next Iteration

- December 31st, 2014. We repeat the exercise from a year earlier
  - Train a model on data from Jan 1st to Sep 30th
  - Validate on data from Oct 1st to Oct 31st (need to wait ~60 days for labels)
  - Validation results look much worse



Next Iteration

- We put the model into production, and the results are terrible
  - From spot-checking and complaints from customers, the performance is worse than even the validation data suggested
- What happened?

Next Iteration

- Existing model already blocking a lot of fraud
- Training and validating only on data for which we had labels
- Possible solution: we could run both models in parallel

Fundamental Problem

For **evaluation**, **policy changes**, and **retraining**, we want the same thing:

> An approximation of the distribution of charges and outcomes that would exist in the absence of our intervention (blocking)

First attempt

- Let through some fraction of charges that we would ordinarily block

```
if score > 50:
  if random.random() < 0.05:
    allow()
  else:
    block()
```

- Straightforward to compute **precision**

Recall

1,000,000 charges	Score < 50	Score > 50
Total	900,000	100,000
Not Fraud	890,000	1,000
Fraud	10,000	4,000
Unknown	0	95,000

- Total "caught" fraud = (4,000 * 1/0.05)
- Total fraud = (4,000 * 1/0.05) + 10,000
- Recall = 80,000 / 90,000 = **89%**

Training

- Train only on charges that were not blocked
- Include weights of 1/0.05 = 20 for charges that would have been blocked if not for the random reversal

```
from sklearn.ensemble import \
RandomForestRegressor
...
r = RandomForestRegressor(n_estimators=10)
r.fit(X, Y, sample_weight=weights)
```

Training

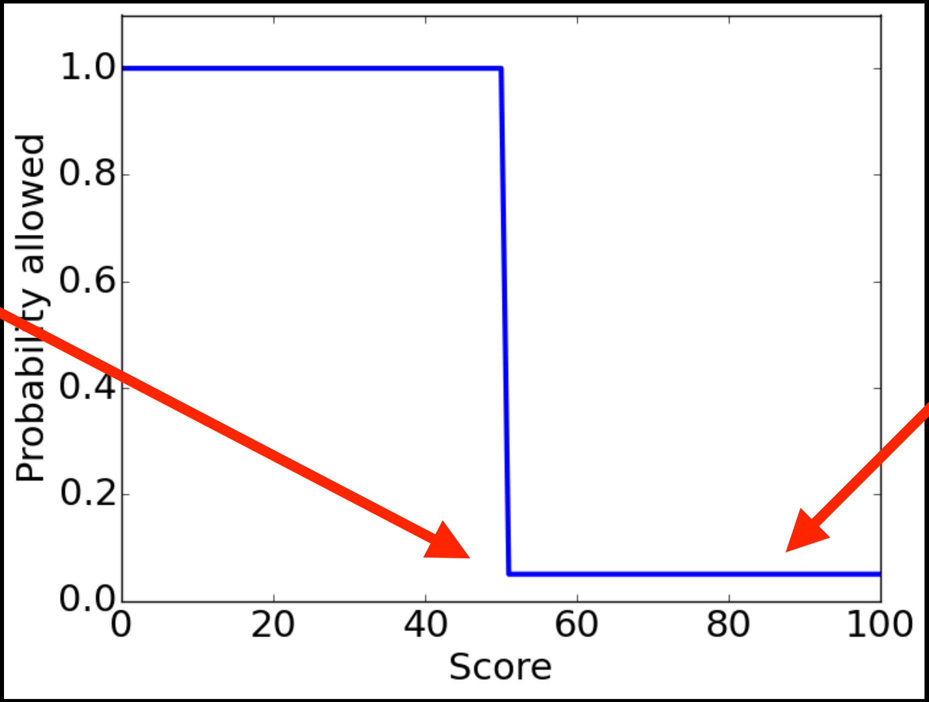
- Use weights in validation (on hold-out set) as well

```
from sklearn import cross_validation
X_train, X_test, y_train, y_test = \
  cross_validation.train_test_split(
    data, target, test_size=0.2)
r = RandomForestRegressor(...)
...
r.score(
  X_test, y_test, sample_weight=weights)
```

Better Approach

- We're letting through 5% of all charges we think are fraudulent. Policy:

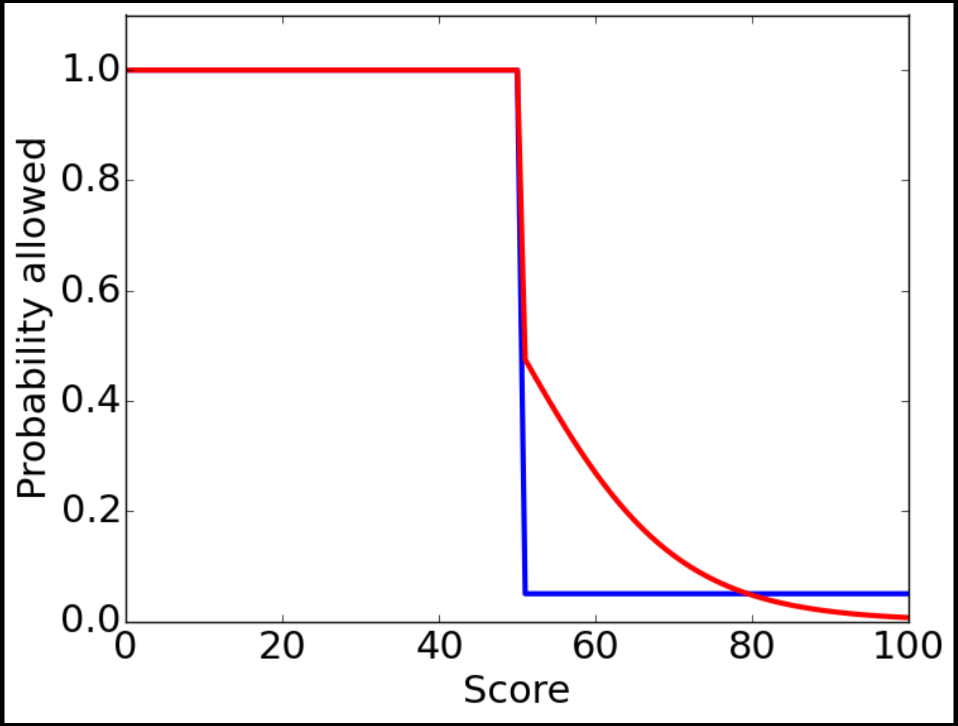
Could go either way



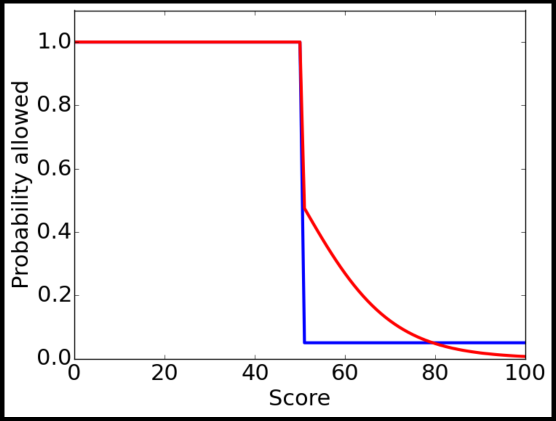
Very likely to be fraud

Better Approach

- **Propensity function**: maps classifier scores to P(Allow)
- The higher the score, the lower probability we let the charge through
- Get information on the area we want to improve on
- Letting through less "obvious" fraud ("budget" for evaluation)



Better Approach




```
def propensity(score):
  # Piecewise linear/sigmoidal
  ...
ps = propensity(score)
original_block = score > 50
selected_block = random.random() < ps
if selected_block:
  block()
else:
  allow()
log_record(
  id, score, ps, original_block,
  selected_block)
```

ID	Score	p(Allow)	Original Action	Selected Action	Outcome
1	10	1.0	Allow	Allow	OK
2	45	1.0	Allow	Allow	Fraud
3	55	0.30	Block	Block	-
4	65	0.20	Block	Allow	Fraud
5	100	0.0005	Block	Block	-
6	60	0.25	Block	Allow	OK

Analysis

- In any analysis, we only consider samples that were **allowed** (since we don't have labels otherwise)
- We weight each sample by **1 / P(Allow)**
  - **"**geometric series"
  - cf. weighting by 1/0.05 = 20 in the uniform probability case

ID	Score	P(Allow)	Weight	Original Action	Selected Action	Outcome
1	10	1.0	1	Allow	Allow	OK
2	45	1.0	1	Allow	Allow	Fraud
4	65	0.20	5	Block	Allow	Fraud
6	60	0.25	4	Block	Allow	OK

Evaluating the "block if **score > 50**" policy

Precision = 5 / 9 = **0.56**
Recall = 5 / 6 = **0.83**

ID	Score	P(Allow)	Weight	Original Action	Selected Action	Outcome
1	10	1.0	1	Allow	Allow	OK
2	45	1.0	1	Allow	Allow	Fraud
4	65	0.20	5	Block	Allow	Fraud
6	60	0.25	4	Block	Allow	OK

Evaluating the "block if **score > 40**" policy

Precision = 6 / 10 = **0.60**
Recall = 6 / 6 = **1.00**

ID	Score	P(Allow)	Weight	Original Action	Selected Action	Outcome
1	10	1.0	1	Allow	Allow	OK
2	45	1.0	1	Allow	Allow	Fraud
4	65	0.20	5	Block	Allow	Fraud
6	60	0.25	4	Block	Allow	OK

Evaluating the "block if **score > 62**" policy

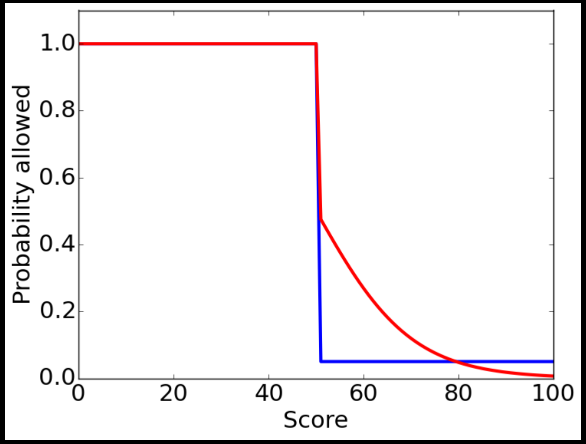
Precision = 5 / 5 = **1.00**
Recall = 5 / 6 = **0.83**

Analysis

- Precision, recall, etc. are estimates
  - Variance of the estimates <u>decreases</u> the <u>more</u> we allow through
- Bootstrap to get error bars
  - Pick rows from the table uniformly at random with replacement and repeat computation

New models

- Train on weighted data (as in the uniform case)
- Evaluate (i.e., cross-validate) using the weighted data
- Can test arbitrarily many models and policies offline (bandit: exploitation vs. exploration)



Counterfactual Estimation and Optimization of Click Metrics for Search Engines

Lihong Li[1] Shunbao Chen[1] Jim Kleban[2]* Ankur Gupta[1]
[1]Microsoft Inc.
Redmond, WA 98052
{lihongli,shchen,ankurg}@microsoft.com
[2]Facebook Inc.
Seattle, WA 98101
jim.kleban@gmail.com

ABSTRACT

Optimizing an interactive system against a predefined online metric is particularly challenging, when the metric is computed from user feedback such as clicks and payments. The key challenge is the *counterfactual* nature: in the case of Web search, any change to a component of the search engine may result in a different search result page for the same query, but we normally cannot infer reliably from search log how users would react to the new result page. Consequently, it appears impossible to accurately estimate online metrics that depend on user feedback, unless the new engine is run to serve users and compared with a baseline in an A/B test. This approach, while valid and successful, is unfortunately expensive and time-consuming. In this paper, we propose to address this problem using causal inference techniques, under the contextual-bandit framework. This approach effectively allows one to run (potentially infinitely) many A/B

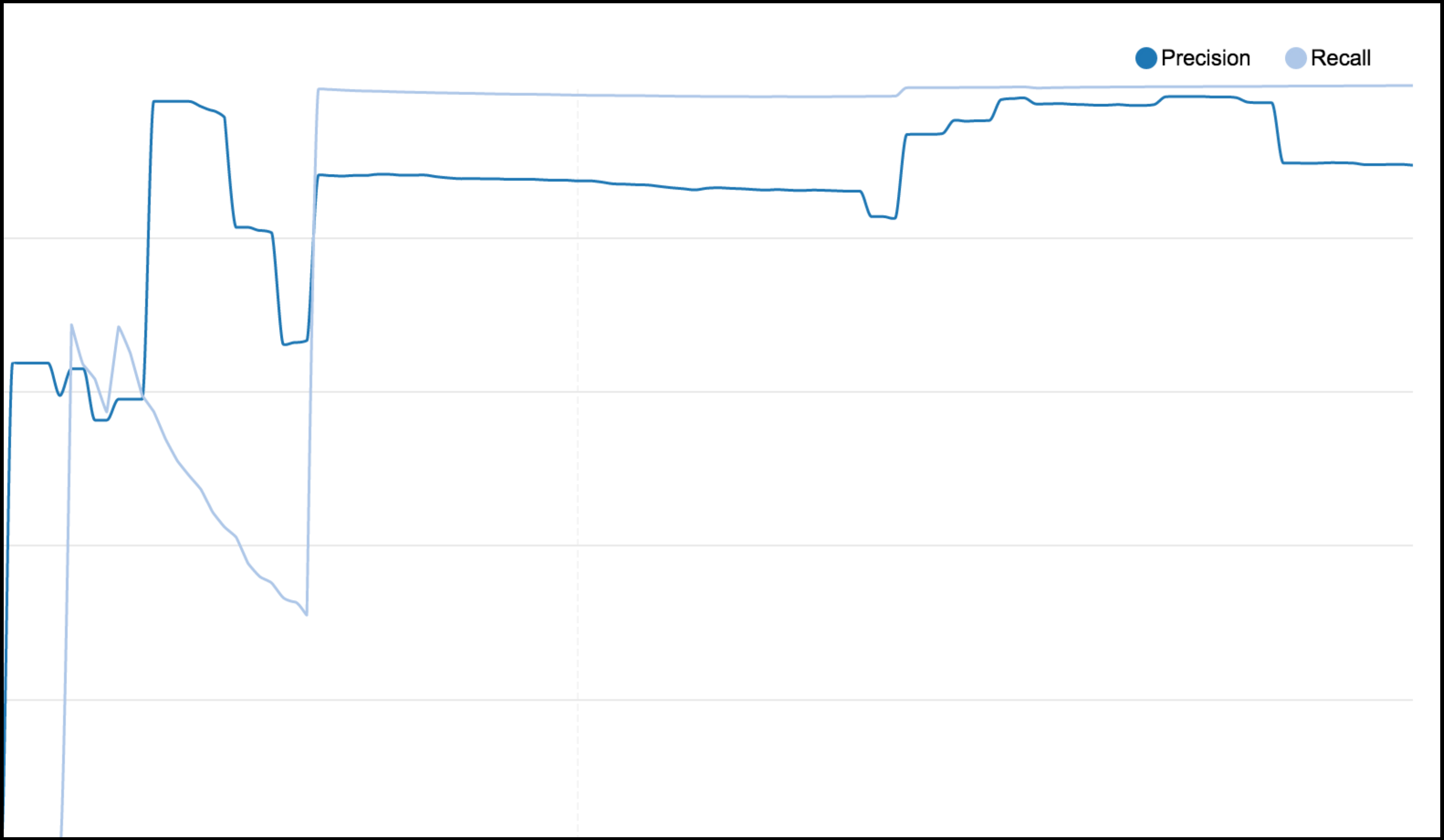
1. INTRODUCTION

The standard approach to evaluating ranking quality of a search engine is to evaluate its ranking results on a set of human-labeled examples and compute *relevance metrics* like mean average precision (MAP) [1] and normalized discounted cumulative gain (NDCG) [17]. Such an approach has been highly successful at facilitating easy comparison and improvement of ranking functions (*e.g.*, [6, 32, 34]).

However, such offline relevance metrics have a few limitations. First, there can be a mismatch between users' actual information need and the relevance judgments of human labelers. For example, for the query "tom cruise," it is natural for a judge to give a high relevance score to the actor's official website, http://tomcruise.com. However, search log from a commercial search engine suggests the opposite—users who issue that query are often more interested in news about the

Technicalities

- Independence and random seeds
- ? =>



Conclusion

- If some policy is actioning model scores, you can inject randomness in production to understand the counterfactual
- Instead of a "champion/challenger" A/B test, you can evaluate arbitrarily many models and policies in this framework

Thanks!

- Work by Ryan Wang (@ryw90), Roban Kramer (@robanhk), and Alyssa Frazee (@acfrazee)



- We're hiring engineers/data scientists!
- mlm@stripe.com | @mlmanapat