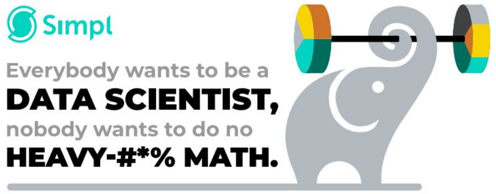
The final stage of grief (about bad data) is acceptance

Chris Stucchio
Director of Data Science @ Simpl
https://www.chrisstucchio.com

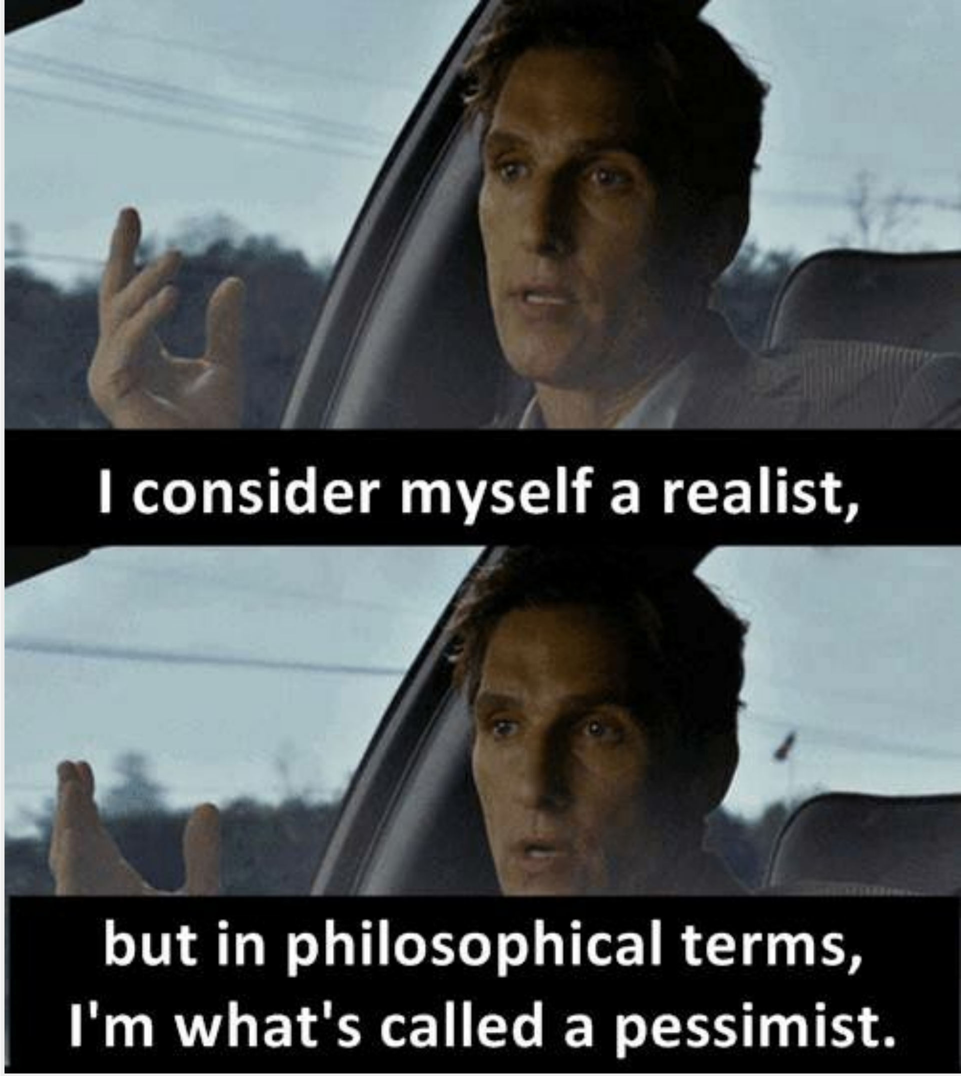


This talk is NOT about

Data cleaning, data monitoring, data pipeline management, improving your data in any way.

This talk is about

Drawing correct inferences from low quality data


I consider myself a realist,
but in philosophical terms, I'm what's called a pessimist.

A recipe for bad data

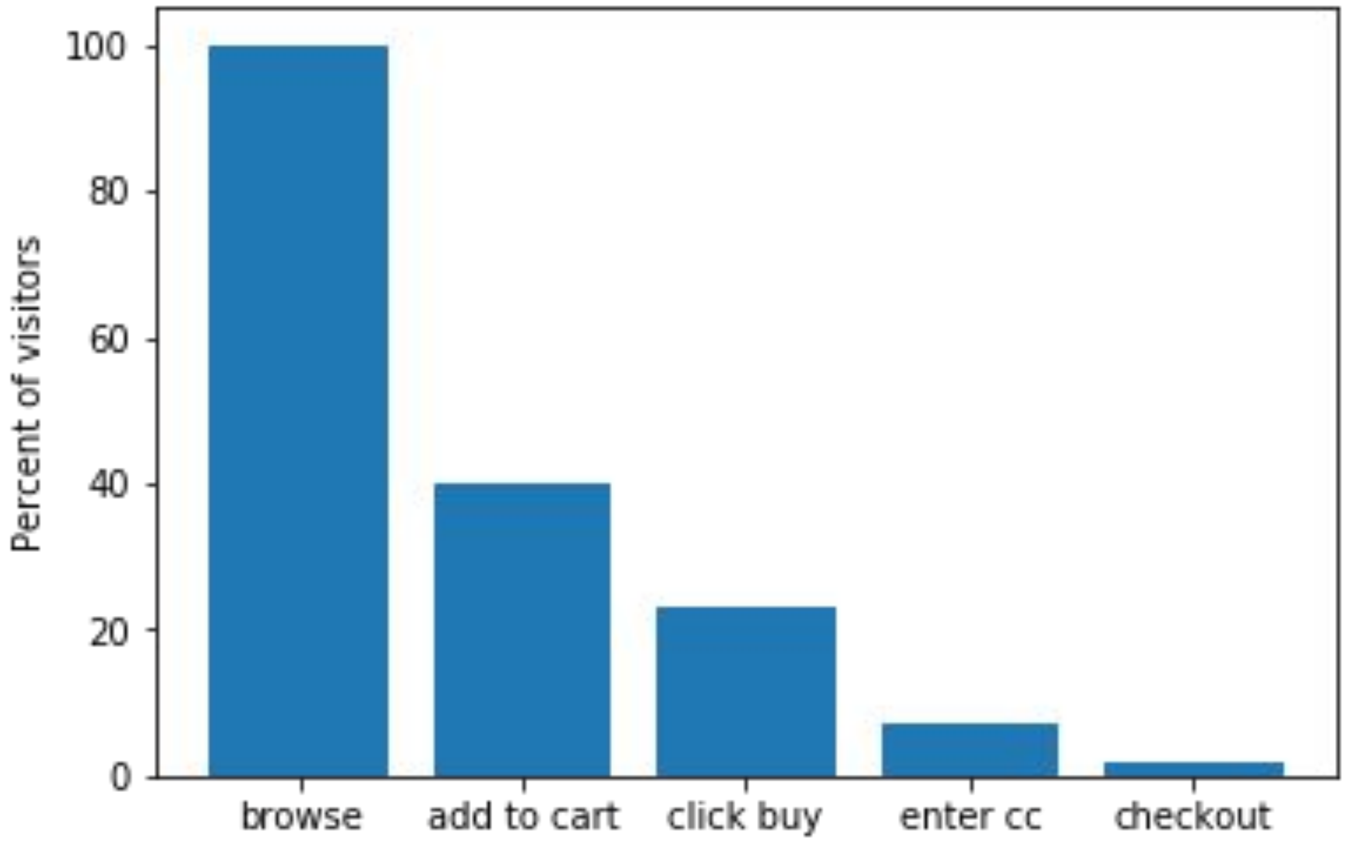
Ordinary data science

- Get reasonably clean data
- Do some cleaning, e.g. `cityname.lower()`
- Train a predictive model (e.g. a neural network, gradient boosting) on the resulting data set

Key idea of this talk

- Get unfixably dirty data.
- Identify **latent/hidden variables** that the data is predictive of
- Build model to predict latent variables
- Train your final model on the latent variables.

Missing data



Funnel Analysis

Funnel analysis

```
requestID |Enter CC    | Purchase
----------+------------+----------
abc       | 12:00      | 1:00
def       | 12:01      | null
ghi       | null       | null
jkl       | null       | 1:03
```

Funnel analysis

```
requestID |Enter CC    | Purchase
----------+------------+----------
abc       | 12:00      | 1:00
def       | 12:01      | null
ghi       | null       | null
jkl       | null       | 1:03
```

WTF is this? I don't even know?

User made purchase without filling in CC?

Where does the data come from?

Tracking pixel sends request to our server whenever CC is entered...

Single server collecting thousands of data points per second…

Putting it into hundreds of SqlLite databases…

Stored on 4 disks…

= thousands of disk seeks+fsyncs/sec.

Engineer in me asks: **“Is it possible that some data is getting lost?”**

A recipe for bad data

Ordinary data science

- Take data as given
- `df[‘purchase’]=~df[‘purchase_time’].isnull()`
- Compute final conversion rate as `df[‘purchase’].mean()`

Key idea of this talk

- Recognize that we lost records of conversions that happened.
- Identify **latent/hidden variables:** data loss rate, true number of conversions.
- Build model to identify these hidden variables
- Final model: `conversion_rate=true_conversions / visits`

Model the data generating process

P(enter CC) = A

P(purchase | enter CC) = B

P(event observed | event occurs) = D

First two probabilities are interesting to customers - funnel transition probabilities (what we want to measure).

Third probability is interesting to us - how well our data collector works.

Data reported to us

100k unique visits

In 40k cases, we saw CC entered but no purchase

In 10k cases, we saw CC entered and purchase

In 5k cases, we saw **no CC entered** but still a purchase

Questions: What is the conversion rate? How many events are we losing?

Modeling the data

# enter CC ← Binom(100,000 unique visits, P(Enter CC))

40k ← Binom(# enter cc, P(observed))

# purchase ← Binom(# enter CC, P(submit | Enter CC))

15k ← Binom(# form submit, P(observed))

The green represents observable data and the red represents latent (hidden) variables. Blue is what the customer wants to see.

Questions: What is the conversion rate? How many events are we losing?

PyMC to the rescue

```
model = pymc.Model()

with model:
    form_fill_CR = pymc.Uniform('form_fill_cr', lower=0, upper=1)
    submit_CR = pymc.Uniform('submit_cr', lower=0, upper=1)
    observe_rate = pymc.Uniform(observed, lower=0, upper=1)

    form_fill_actual = pymc.Binomial('form_fill_actual', n=100000, p=form_fill_CR)
    form_fill_obs = pymc.Binomial('form_submit_obs', n=form_fill_actual , p=observe_rate,
observed=40000)

    submit_actual = pymc.Binomial('submit_actual', n=form_fill_actual, p=submit_CR)
    submit_observed = pymc.Binomial('submit_observed', n=submit_actual, p=observe_rate,
observed=15000)
```

Final results

Purchase CR (naive): 15k purchases / 100k visits = 15%

Purchase CR (implicit stats model): 16.7 purchases / 100k visits - 11% higher!

Rate of data loss = 10%

Data collection system to be fixed!

(But we can give customers more accurate numbers until that happens…)

Model your fundamental relationships

By understanding where the data comes from, you can build a model of how the data must fit together.

- **Enter CC** before **Form Submit**. (Or “open email” before “click link in email”, “display ad” before “click ad”.)

Data which is present leaves clues to data which is missing.

Mislabeled
data,
inconsistent
formats

And no one cares

Problem: Identify phishing and fraud

My phone: Google Pixel XL 2

My location: Mostly Bangalore, sometimes Hyderabad

Problem: Identify phishing and fraud

Attempted account access: this Nokia thing

Location: Jaipur

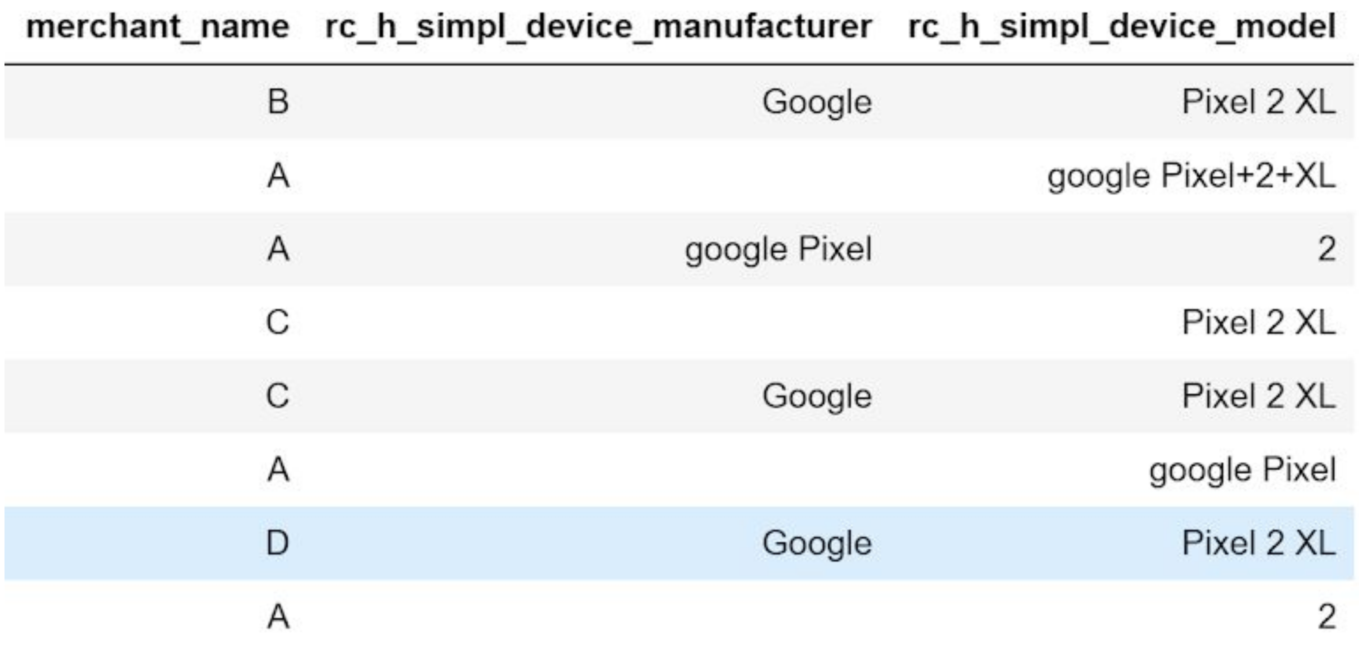
Does this seem right?

Brilliant plan

Flag phones that don’t match previous device used

(“Google”, “Pixel 2 XL”) != (“google Pixel”, “2”)

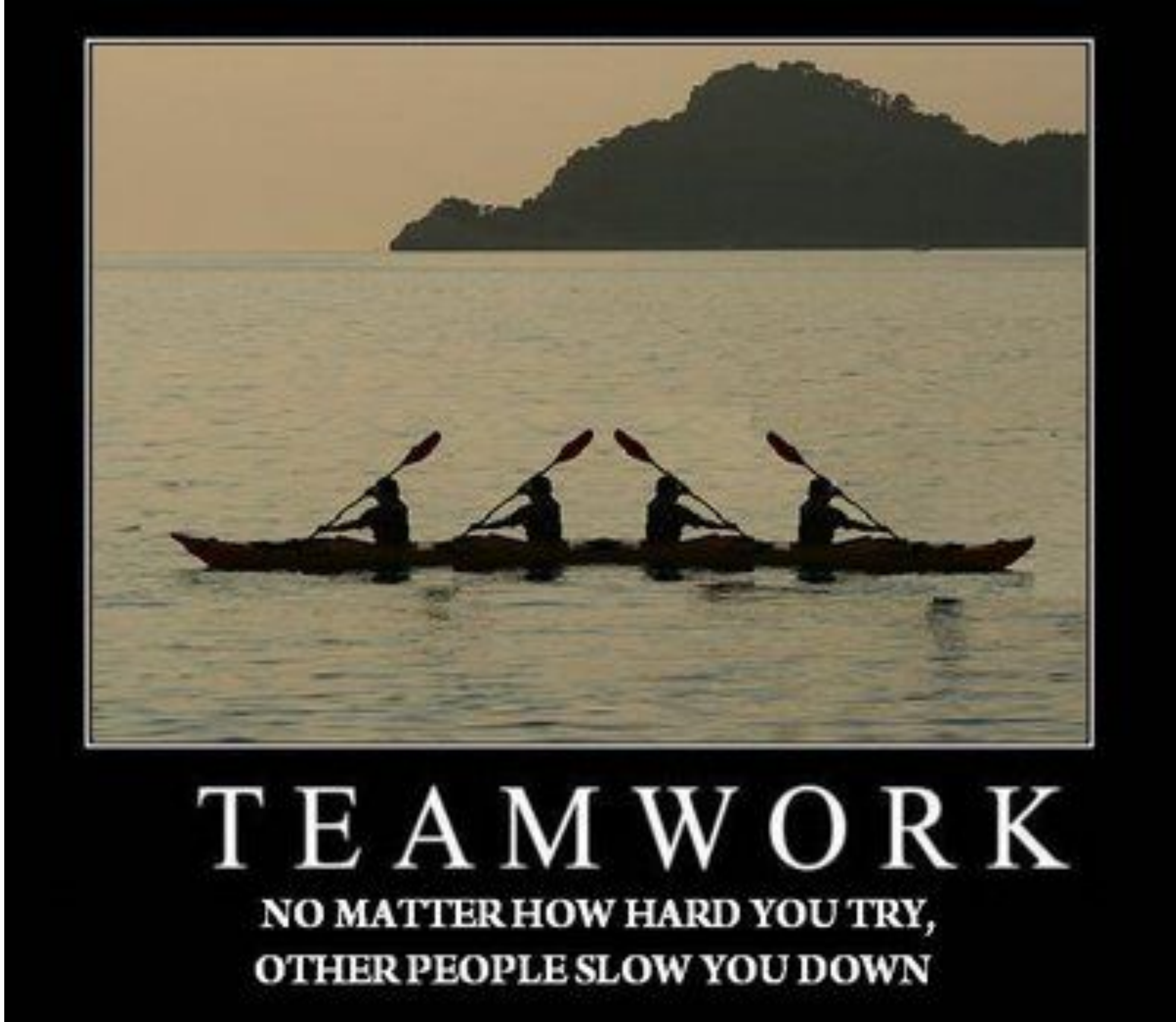
My device history:



merchant_name	rc_h_simpl_device_manufacturer	rc_h_simpl_device_model
B	Google	Pixel 2 XL
A		google Pixel+2+XL
A	google Pixel	2
C		Pixel 2 XL
C	Google	Pixel 2 XL
A		google Pixel
D	Google	Pixel 2 XL
A		2

People involved in getting the data fixed:

- Partners
- Bizdev
- Product managers
- Engineering



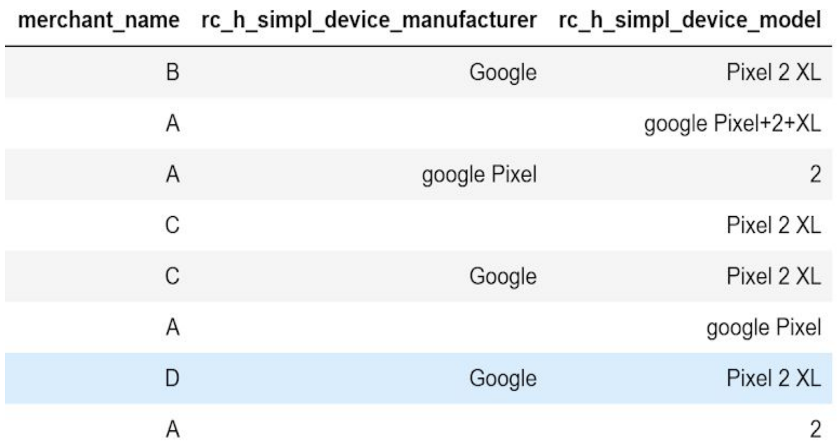
Mathematically model our bad data

Latent variable (unobservable) = actual underlying devices.

Data (observable): Label = L(Device, Observer)

Data set:

[User ID, Observer, L(Device, Observer)]



merchant_name	rc_h_simpl_device_manufacturer	rc_h_simpl_device_model
B	Google	Pixel 2 XL
A		google Pixel+2+XL
A	google Pixel	2
C		Pixel 2 XL
C	Google	Pixel 2 XL
A		google Pixel
D	Google	Pixel 2 XL
A		2

My user history at Simpl

Time for linear algebra

Columns: (merchant, manufacturer, model) combinations

Row:
user

Cell:
An observation of a device string associated to a user.

Dimension:
(# users) x (# device strings x merchants)

Incomplete matrix.

"Google" "Pixel XL 2", "", A	"iPhone X", "", A	"Google Pixel" "2", "B"	"Google Pixel" "XL2", "C"	"iPhone" "10", "B"
1	0	1	NaN	0
1	0	NaN	1	0
0	1	0	NaN	1
1	1	1	NaN	0

Rank = # devices

Low rank matrix completion = classic problem in data science.

(But mostly only seen in recommendation engines.)

Low rank approximation

Each device corresponds to a row vector in low rank approximation.

Complete matrix using low rank approximation.

Observations not matching low rank approximation = possible attack.

$$M = \sum_{j}^{\#\ \text{devices}} \vec{u}^j \otimes \vec{d}^j + E$$

$\vec{u}^j$

"Google" "Pixel XL 2", "", A	"iPhone X", "", A	"Google Pixel" "2", "B"	"Google Pixel" "XL2", "C"	"iPhone" "10", "B"
1	0	1	NaN	0
1	0	NaN	1	0
0	1	0	NaN	1
1	**1**	1	NaN	0

$\vec{d}^j$

1	0	1	1	0

Google Pixel XL 2 vector

Mathematically like collaborative filtering

User = document

Device observation = word

Real world device (hidden variable) = topic

Possible attacker: a document that fits into multiple topics.

Sketch of solution

Topic model

Random errors/ attacks

$$M = \sum_{j=1}^{D} \vec{u}^j \otimes \vec{d}^j + E$$

$$\sum_{j=1}^{D} \vec{u}^j \otimes \vec{d}^j = O(\#\ \text{devices})$$

$$\|E\|_1 = O(1)$$

$$(M^\dagger M)_{ij}$$

# of users seen with both device string *i* and device string *j*

$$(M^\dagger M)_{ij} \sim \sum_{j=1}^{D} N_j \vec{d^j} \otimes \vec{d^j} + O(1)$$

(These device vectors are similar but not identical to the ones in the prev slide.)

Collaborative filtering, simple version

1. Compute $M^\dagger M$ - by construction must be sparse self-adjoint matrix of size O(N^2) + dense error term of size O(N).
2. Apply thresholding - Truncate terms of size O(N) to zero.
3. Find eigenvectors. Eigenvectors of $M^\dagger M$ = right singular vectors of M = device profile vectors $\vec{d}^j$

In production:

1. For any given user, map their device string to a device vector.
2. Track devices associated to a user, i.e. user_id -> j.
3. If unexpected devices are seen, flag as potential fraud.

How we know it works

1. Reproduces results of some string matching fixes we did, e.g. `“Google+Pixel”.replace(‘+’,’ ‘).lower() == “google pixel”`.
2. Reproduces (“HMD Global”, _) ~ (“Nokia”, _) and (“Huawei, _) ~ (“Honor”, _).
3. Users with multiple devices are rare according to model, as expected.

Get some nonsense results for device strings that have very few users.

This is fully expected from the model: O(N^2) ~ O(N) if N is small, so no clean value for threshold.

Hard for scammers to exploit this: need to identify users with rarely seen phones before they can attack. By definition such users are rare.

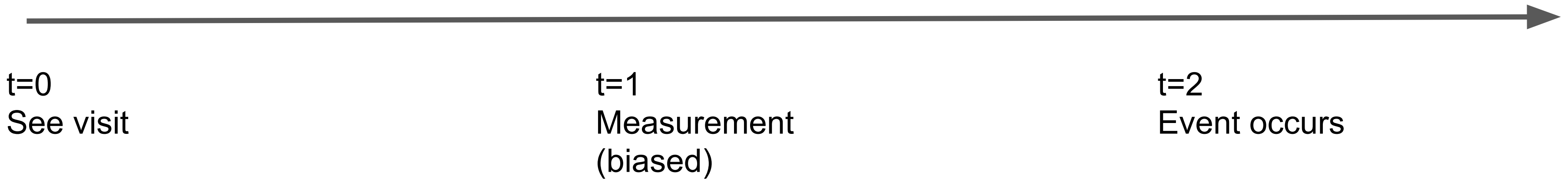
Delayed reactions

Act today, discover outcome tomorrow

WATCH OUT!
OOF!
OK... IT DIDN'T HURT. I THINK I'M GOOD...
MUNCH
MUNCH
AAUUUGHHH!!!
PORTUGUESEGEESE.COM

Pervasive problem in real world

- Send email today. User checks their email tomorrow, clicks email.
- Lend money today. Payment due date end of month. Delinquency data available at end of month + 30 days.
- Buy stock today. Sell stock in 5-10 days. Only learn profit/loss at that time.



Concrete version of the problem

A/B testing an email:

- “Valentines day sale, 2 days left!”
- “Only 2 days left to get your sweety something!”

Want to estimate click through rates of emails as quickly as possible, then send best version to everyone.

Delay bias is introduced because people don’t open an email the instant it’s sent.

Background

Simple version of the problem: measuring a conversion rate.

No delay version: want to find conversion rate γ. One visitor reaches the site...and they convert! What is our opinion of the conversion rate?

$$\begin{aligned} P(\gamma|\text{conversion}) &= \frac{P(\text{conversion}|\gamma)P(\gamma)}{P(\text{conversion})} \\ &= \frac{\gamma P(\gamma)}{P(\text{conversion})} \end{aligned}$$

Background

Simple version of the problem: measuring a conversion rate.

No delay version: want to find conversion rate γ. One visitor reaches the site...and they **do not** convert! What is our opinion of the conversion rate?

$$\begin{aligned} P(\gamma | !\text{conversion}) &= \frac{P(!\text{conversion}|\gamma) P(\gamma)}{P(!\text{conversion})} \\ &= \frac{(1-\gamma) P(\gamma)}{P(!\text{conversion})} \end{aligned}$$

Background

Simple version of the problem: measuring a conversion rate.

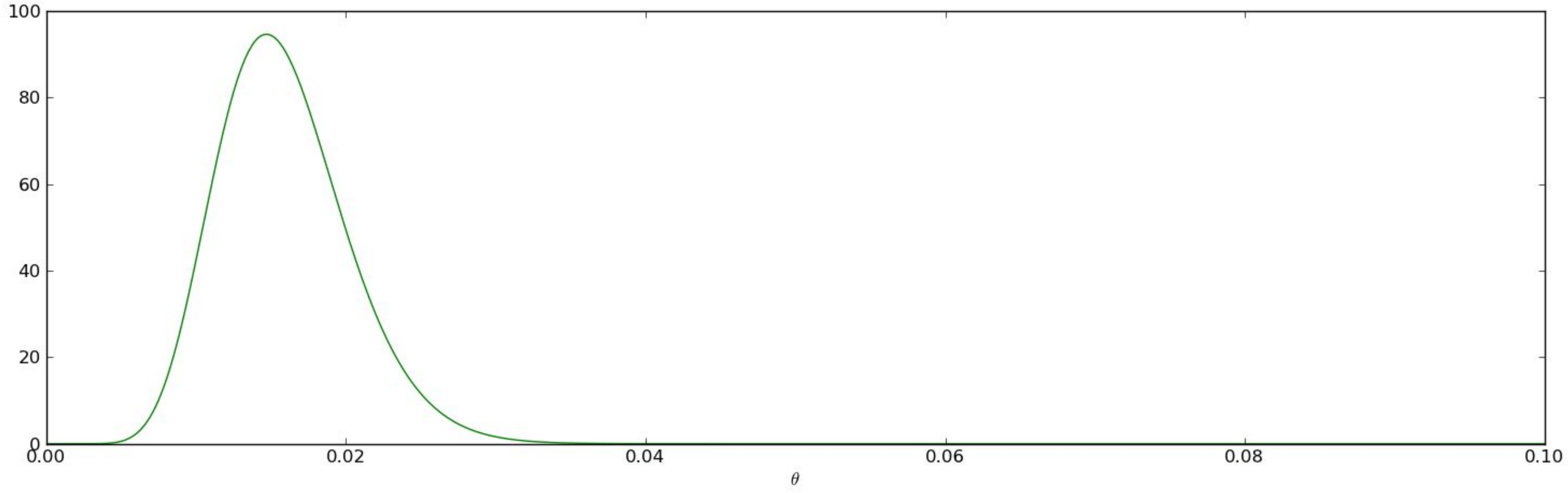
No delay version: N visitors, k conversions.

Use previous two formulas recursively:

$$P(\gamma | N \text{ visits}, k \text{ conversions}) \\ = C\gamma^k (1-\gamma)^{N-k} P(\gamma)$$

Background

Posterior after 794 impressions, 12 clicks.
Clustered around 12/794=0.015, as expected.



Adding in delays

If you send an email and it isn't clicked in the next day, it might get read/clicked later.

Important question you can answer: how long does it take before 5% / 50% / 90% of emails that **will** get clicked **actually do**?

Data set: $t_{e,1}, \ldots, t_{e,N}$ $t_{o,1}, \ldots, t_{o,M}$

Time lag between send/now for emails that received NO clicks.

Time lag between send/click for emails that did get clicked.

Survival models

Fact 1: An email is sent at 12:00:00. It’s now 12:00:30 and it has not been clicked. This tells us almost nothing about whether it will get clicked.

Fact 2: An email was sent at 12:00:00 Jan 1 2017. It’s now July 1 2019 and it has not been clicked. This tells us the email will probably never be clicked.

Open question: What about at 12 hours, 24 hours, 72 hours, etc?

Survival models

The solution to this problem is called a **survival model**. Fasih Khatib already spoke about this.

Definition:

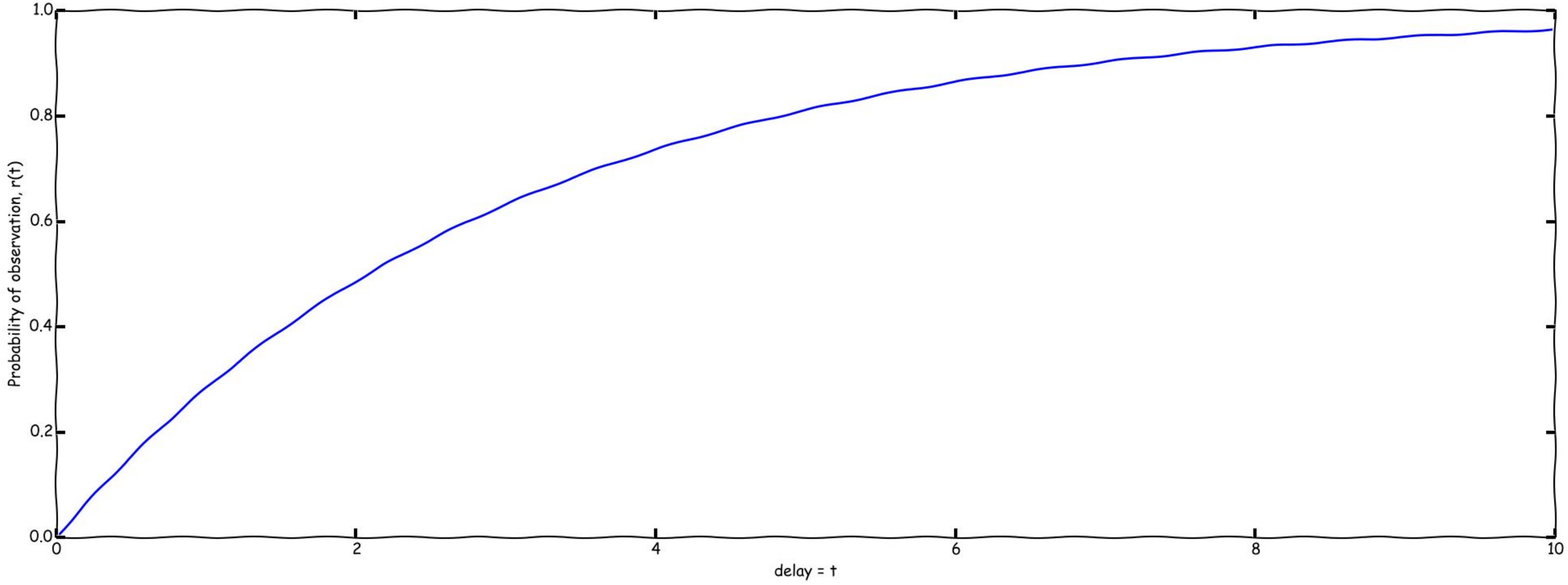
$$r(t) = P\left(\text{event is observed before } t_e + t \middle| \text{event occurs at } t_e \text{ AND event is eventually observed}\right)$$

Goal of survival models: find r(t)

Intuitive interpretation: “If we know the email is eventually clicked, what are the odds it’s clicked before time t?”

Survival models

Inverse survival curve:



Error correction

Now assume the email was sent at t=0, and it’s now time t. The email has not been clicked. The likelihood of this is:

$$\begin{aligned} P(\text{click} < t|\gamma) &= \gamma r(t) \\ P(!\text{click} < t|\gamma) &= 1 - \gamma r(t) \end{aligned}$$

We can do the same recursion as in the no delay case to get a moderately more complex formula.

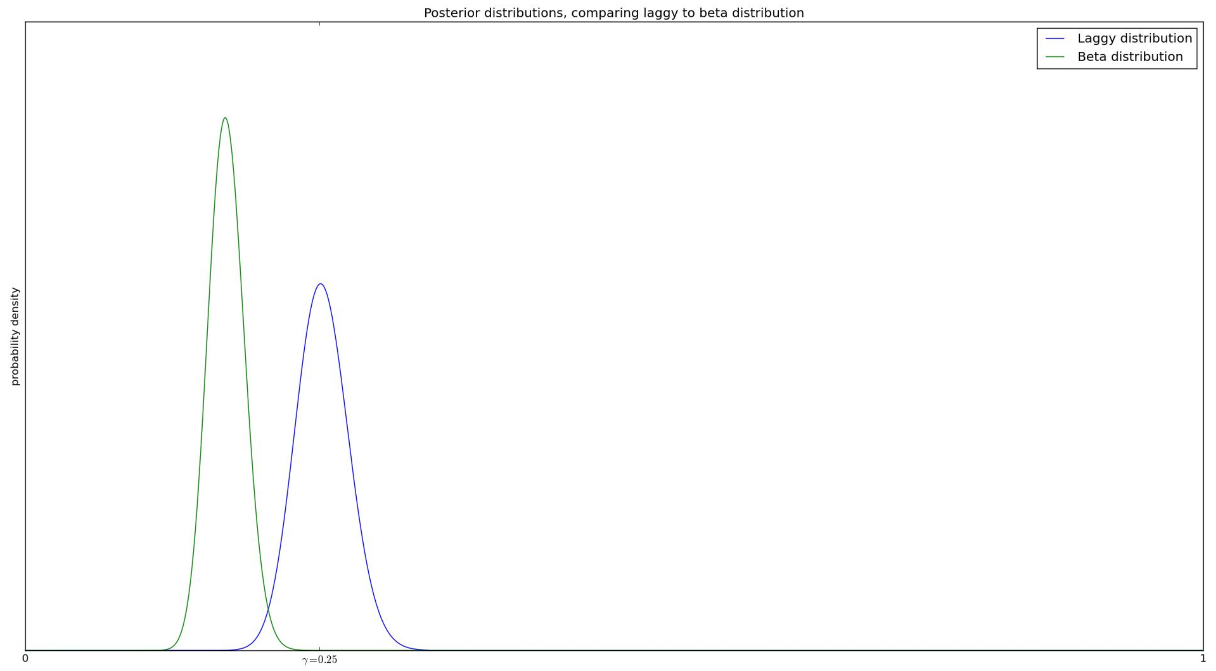
Error correction

$$\begin{aligned} P(\gamma | t_{e,i}, \ldots, t_{O,j}) &= C \left(\prod_{j=0}^{M} \gamma r(t_{O,j}) \right) \\ &\times \left(\prod_{j=0}^{N} [1 - \gamma r(t_{O,j})] \right) P(\gamma) \end{aligned}$$

Error correction

Results of error correction.

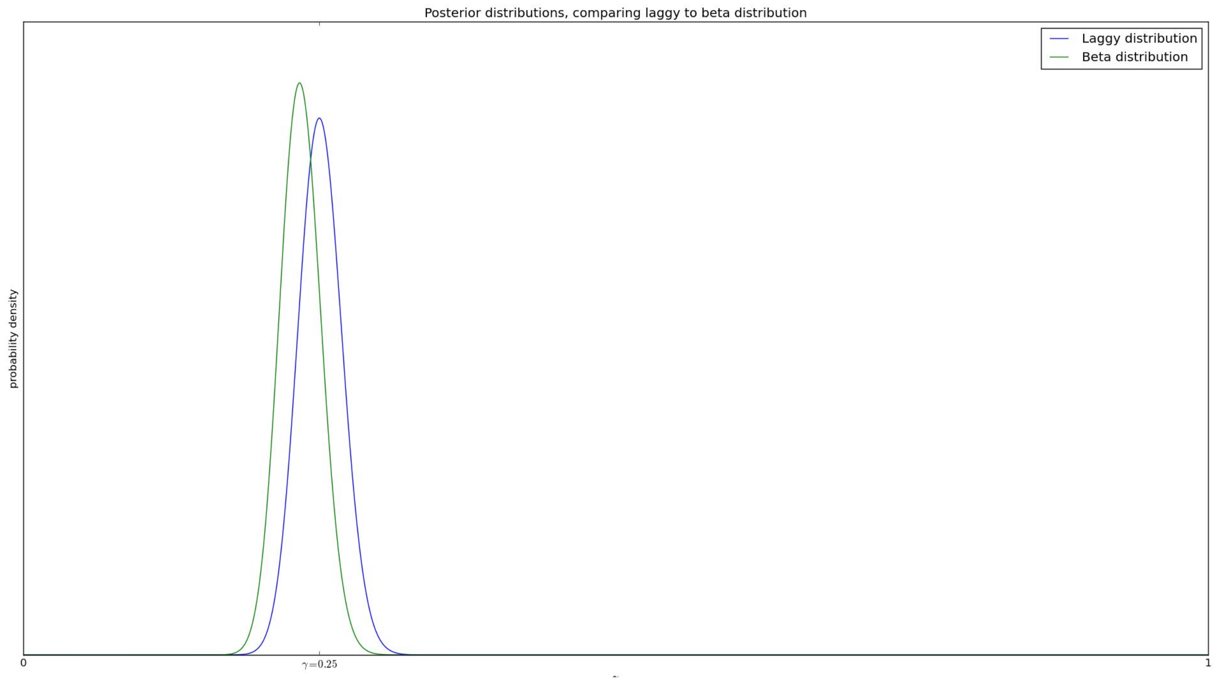
Simulated data, true conversion rate was 25%, but with lag the estimated conversion rate was lower.



Error correction

Results of error correction.

Simulated data, same situation but waiting a longer time to measure CR.



Key idea

There are two major features influencing whether a user opens/clicks an email:

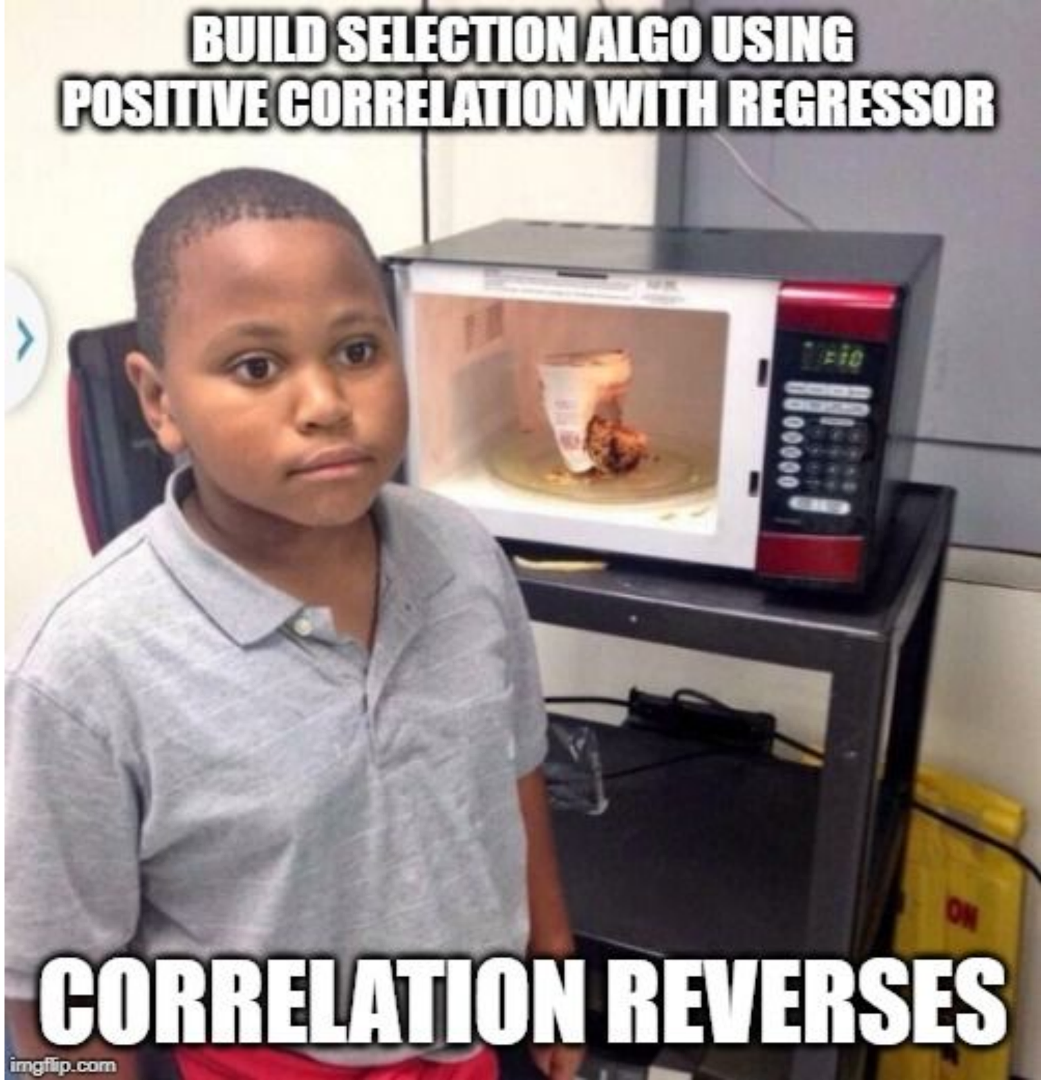
1. Conversion rate, i.e. how persuasively the subject/email is written.
2. Time - whether the user has actually checked their email.

If we ignore time, we lose accuracy and underestimate the true conversion rate.

By incorporating time into our model we can accurately measure it, albeit with greater uncertainty.

Self-Imposed Selection Bias

When the tails come apart


BUILD SELECTION ALGO USING POSITIVE CORRELATION WITH REGRESSOR
CORRELATION REVERSES
imgflip.com

Selection is primary goal of many ML algos

But training data comes only from people selected:

- Find and display the ad with highest CTR: but only get data for ads which were displayed
- Identify the best borrowers, and lend money only to them. Performance data only available for loans which were issued
- Identify best students. Performance data only available for admitted students

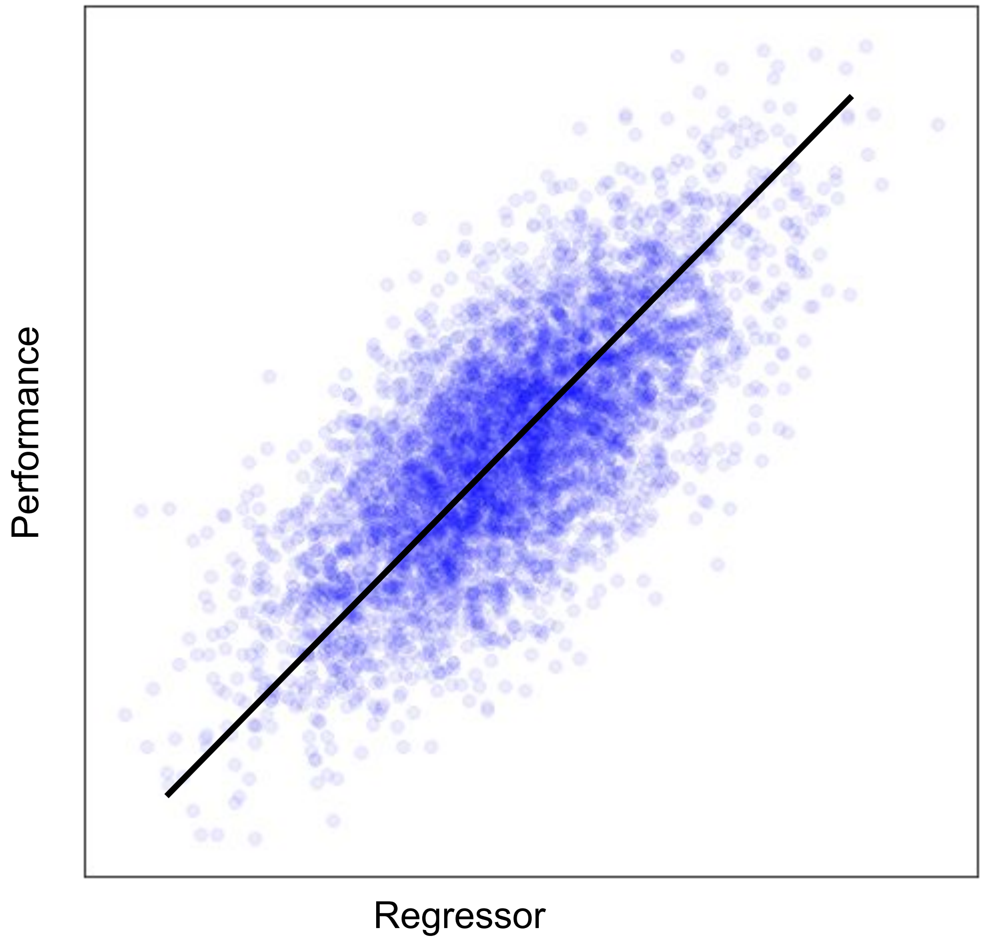
Counterfactual is often unavailable!

Pernicious effects of selection bias

Consider two variables X and Y.

These variables are clearly correlated with each other.

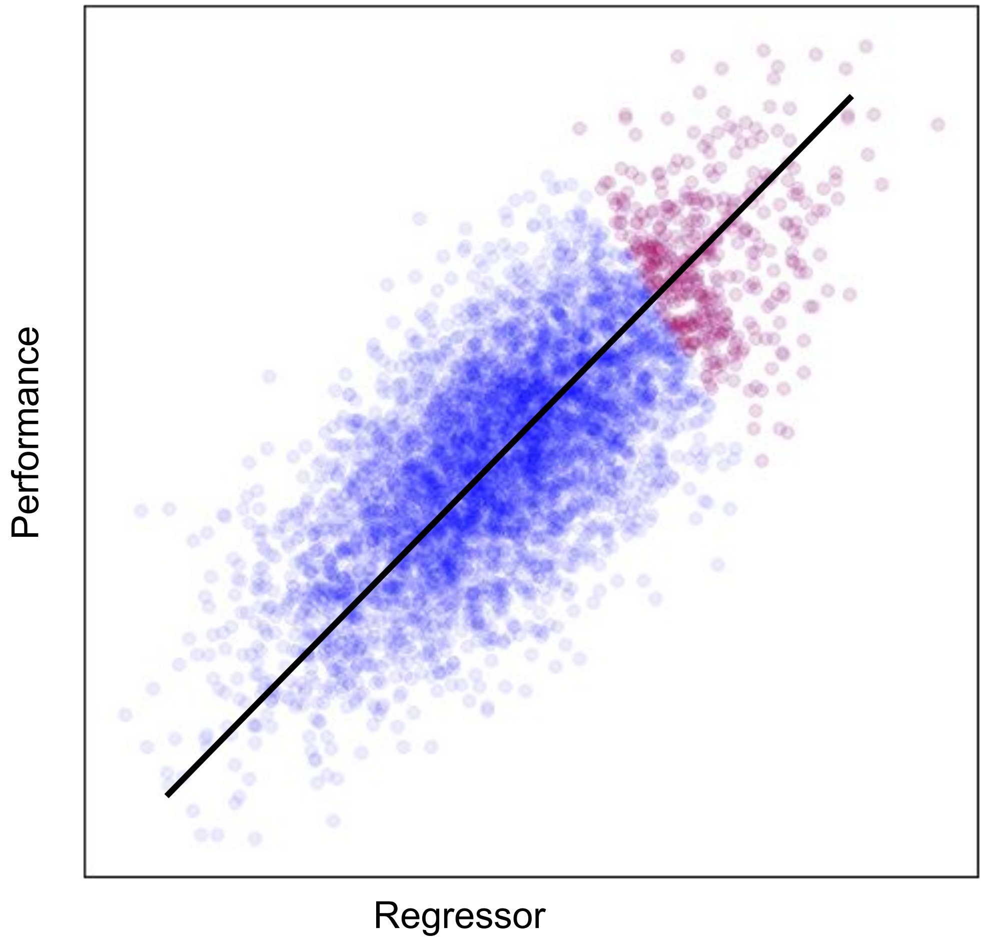
This is the early days of our model.



Pernicious effects of selection bias

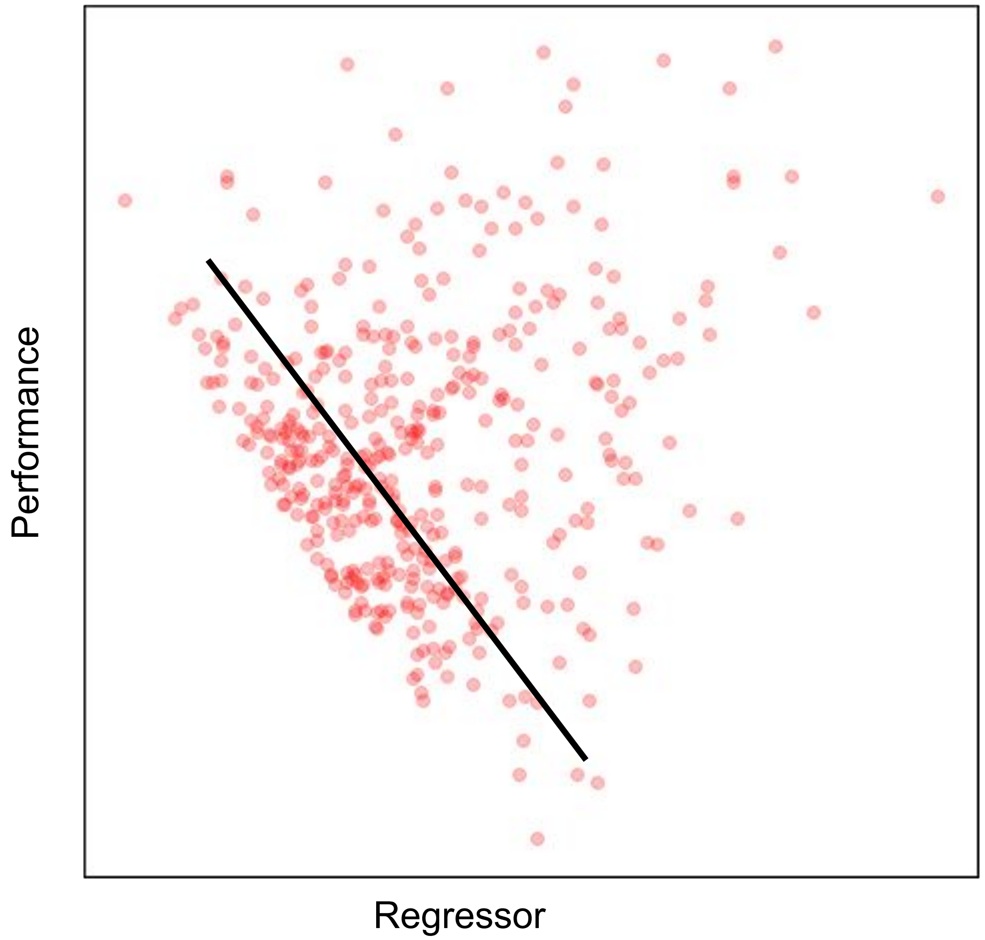
Approve only the people that the model says is good.

Our model is working!

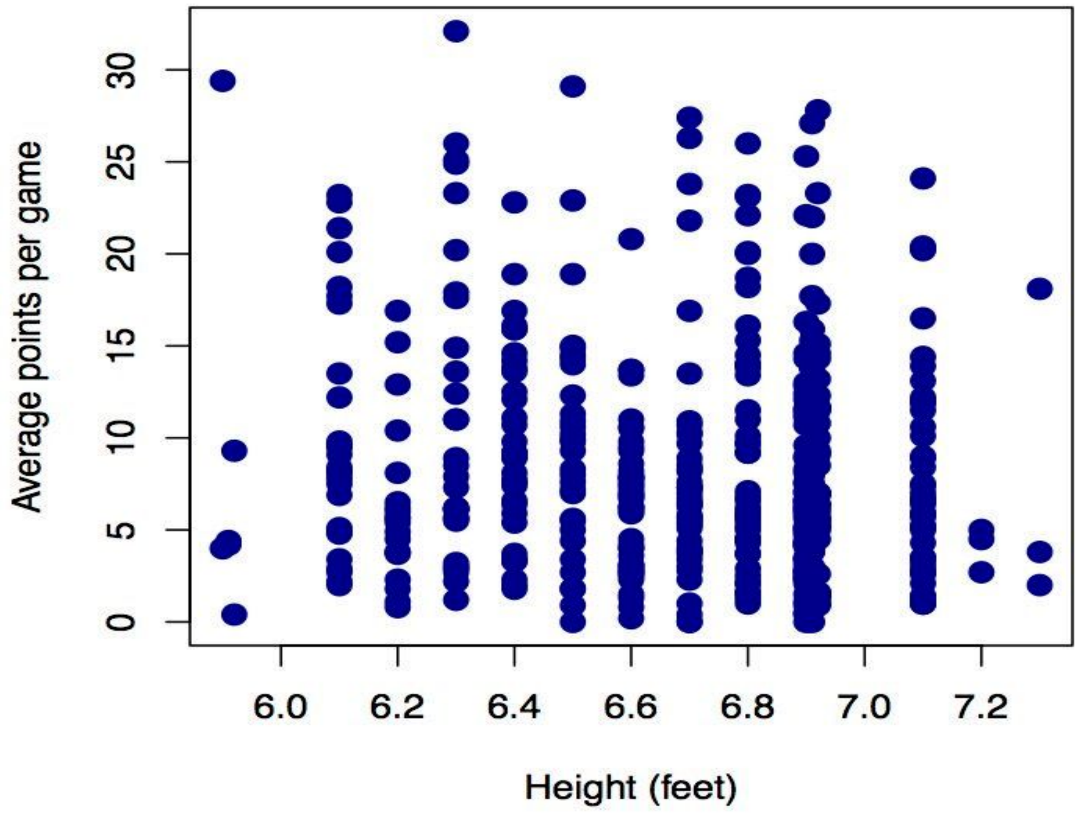


Pernicious effects of selection bias

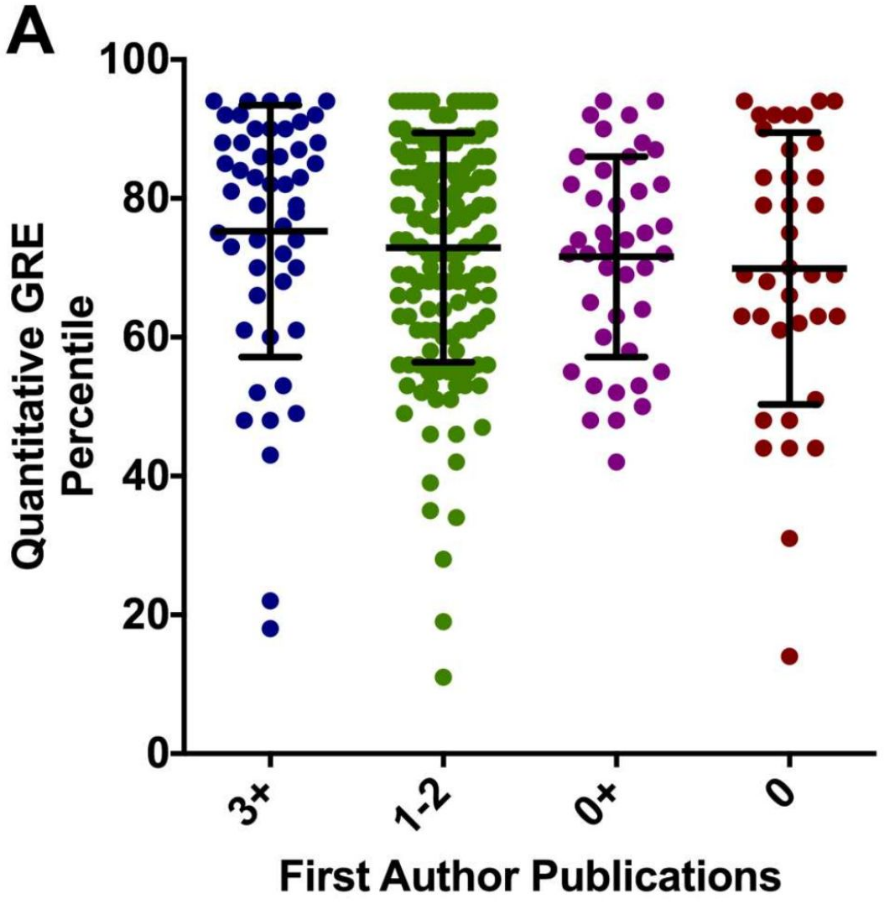
Our training data suddenly exhibits anti-correlation between X and Y!



Height vs Performance in the (American) National Basketball Association



Publications among Ph.D. students



Heckman Correction

Pervasive problem in econometrics

Heckman received a Nobel Prize in Economics for a statistical methodology to solve this, but it’s hard to combine that with modern ML algos. Entirely based on normal distributions...

Heckman Correction

Pervasive problem in econometrics

Open problem: How to use Heckman correction on NN or GBM? (Talk to me if you have ideas!)

Heckman received a Nobel Prize in Economics for a statistical methodology to solve this, but it's hard to combine that with modern ML algos. Entirely based on normal distributions...

The data is bad.
Deal with it.