Hazardous Models and Risk Mitigation in Real Estate

DataEngConf SF, April 2018

David Lundgren & Xinlu Huang

Opendoor

Who has modeled time-to-event data before?

Who has modeled time-to-event data before?

What's the half-life of a startup in Silicon Valley?

pied piper

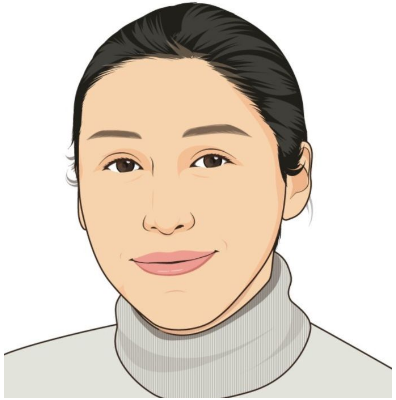
Who has modeled time-to-event data before?

What's the half-life of a startup in Silicon Valley?

When's my team going to score another goal?

Did you use survival analysis?

Introduction

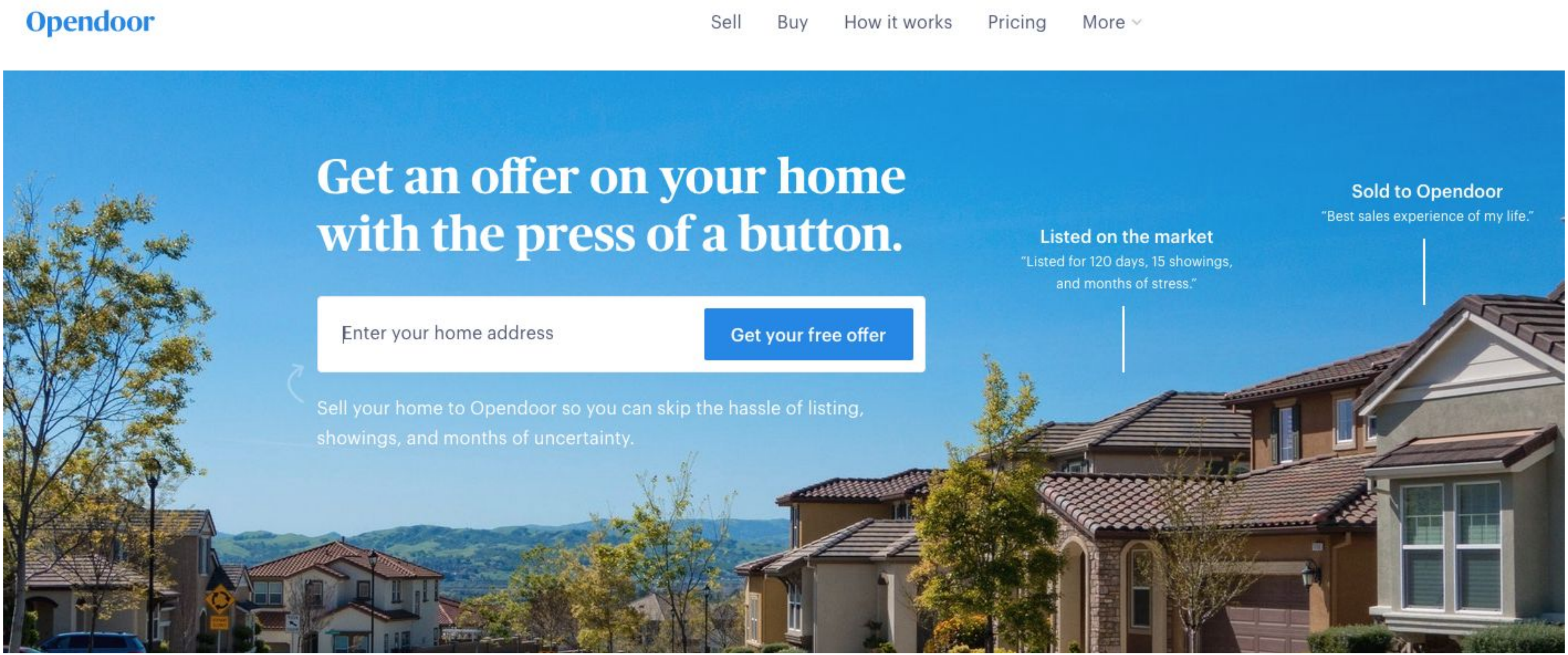


Xinlu Huang

David Lundgren

Talk Structure

- Real Estate 100 and Opendoor 101
  - Modeling Liquidity via Days-on-market
  - Home Sale Case Studies
- Pay Attention to the Negative Space (Model 1)
- Solve a Simpler Problem (Model 2)
- A General Recipe for Survival Analysis (Model 3)
- Q & A


Opendoor
Sell
Buy
How it works
Pricing
More
Get an offer on your home with the press of a button.
Enter your home address
Get your free offer
Sell your home to Opendoor so you can skip the hassle of listing, showings, and months of uncertainty.
Listed on the market
"Listed for 120 days, 15 showings, and months of stress."
Sold to Opendoor
"Best sales experience of my life."

How a home's duration on the market impacts Opendoor

- Opendoor bears the risk in reselling the home
- Time-on-market varies substantially by home
- Our unit costs are driven by how long it takes us to find a buyer for a home

The Problem

How long will it take us to find a buyer for a home?

Home 1

Listed ~$800k

Home 1

Listed ~$800k

6+ months on the market

Home 2

Listed ~$300k

Home 2

Listed ~$300k

1 month on the market

Framing the Problem

Framing the Problem

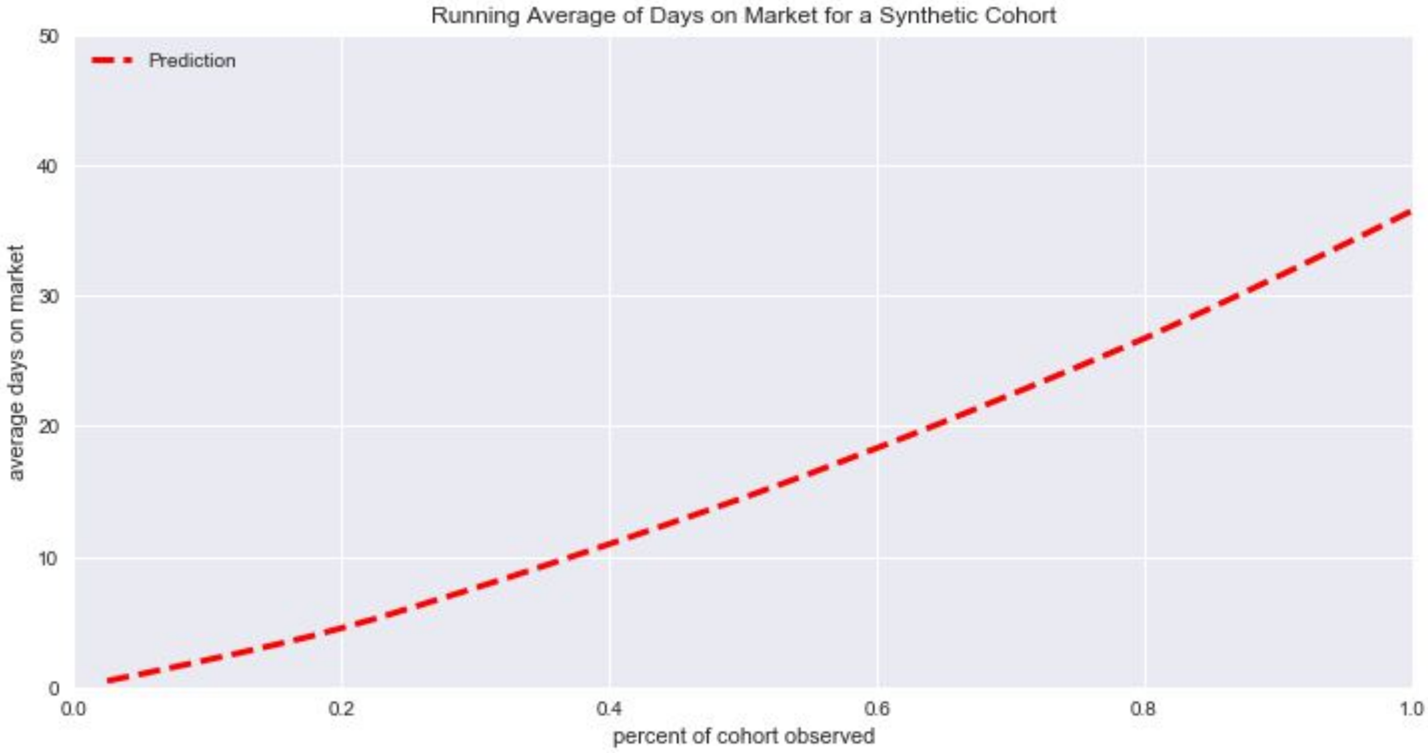
Home	List Price	Square Feet	Other Features	Days-on-market (y)
423 Main Street	$200k	2000	....	30
111 Side Road	$200k	2200	...	100
...				
52 Downtown Ave	$400k	1945		n/a
90 Outskirts Lane	$300k	2100		n/a

Model #1: Linear Regression

Home	List Price	Square Feet	Other Features	Days-on-market (y)
423 Main Street	$200k	2000	....	30
111 Side Road	$200k	2200	...	100
...				

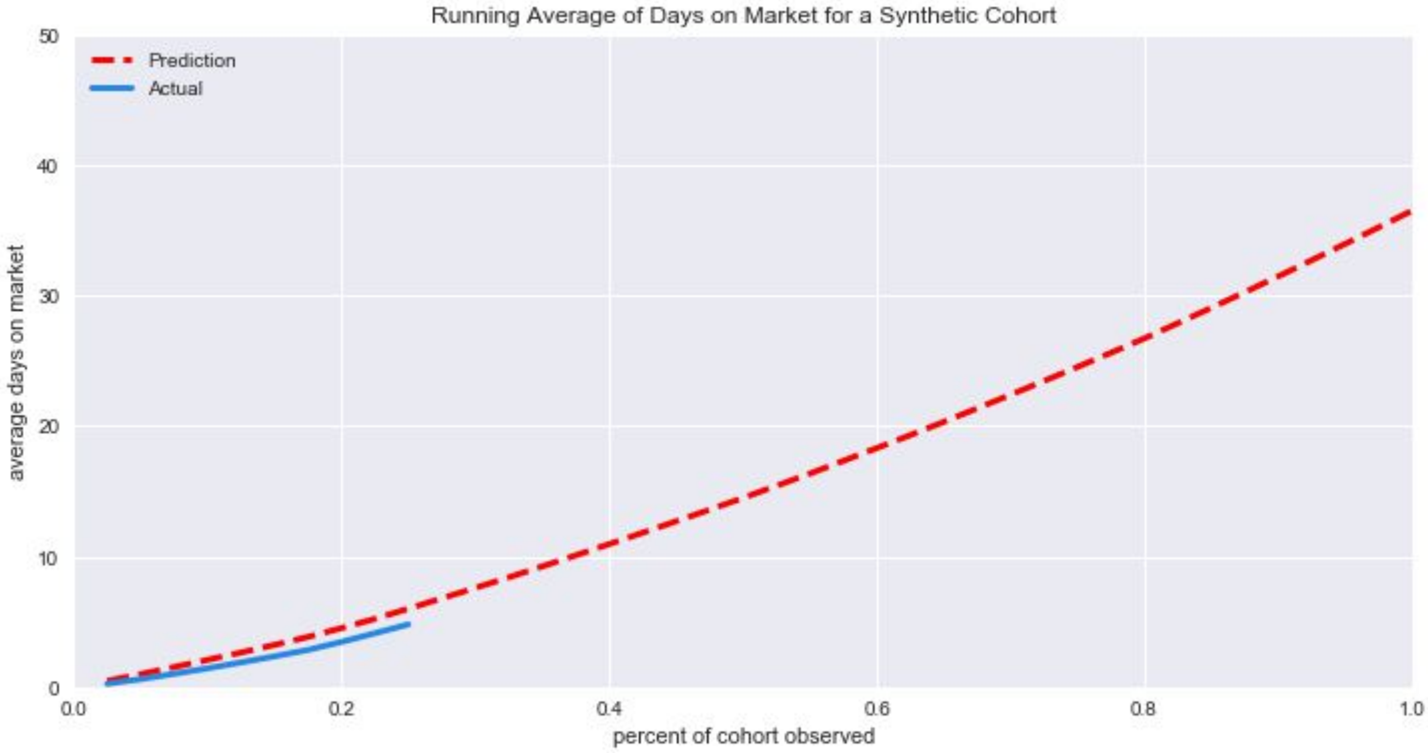
Does it work?

Results



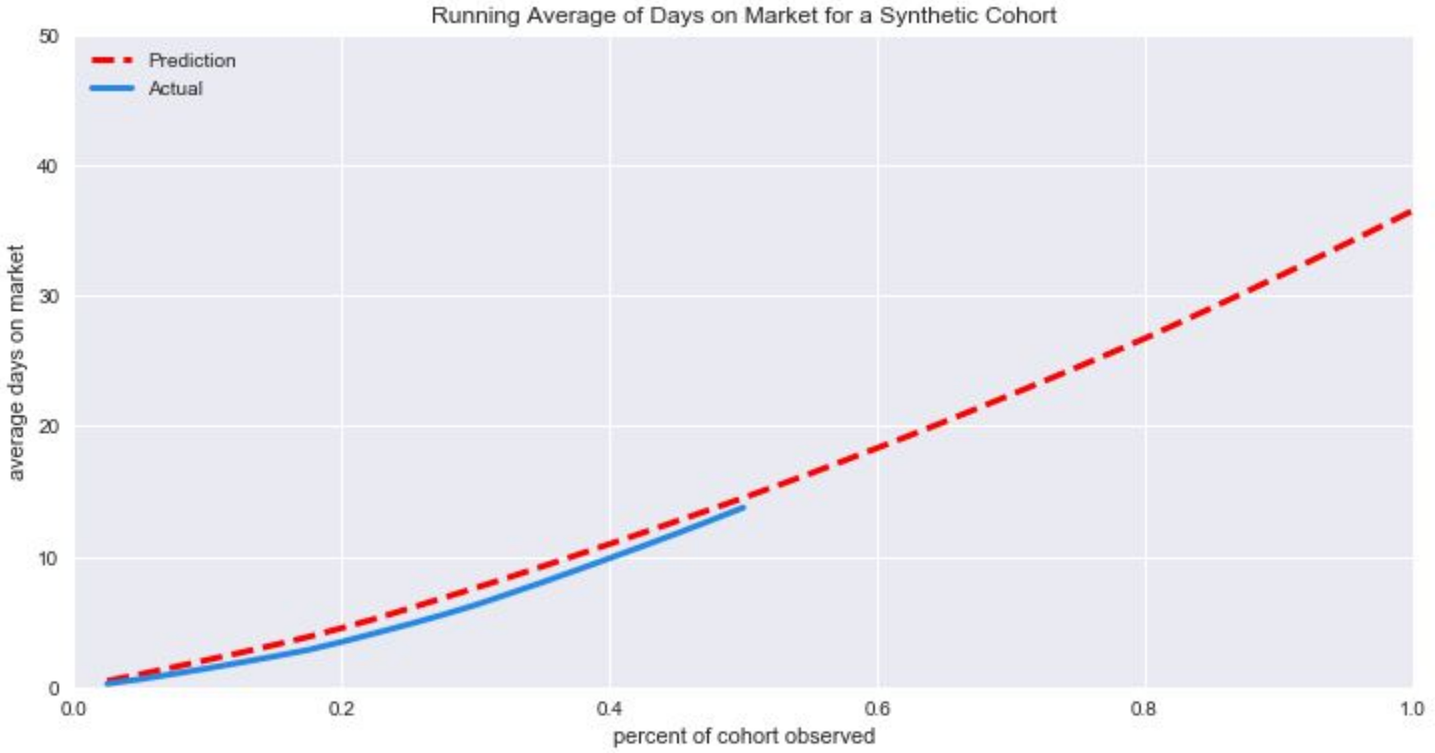
Running Average of Days on Market for a Synthetic Cohort
Prediction
average days on market
percent of cohort observed
0
10
20
30
40
50
0.0
0.2
0.4
0.6
0.8
1.0

Results

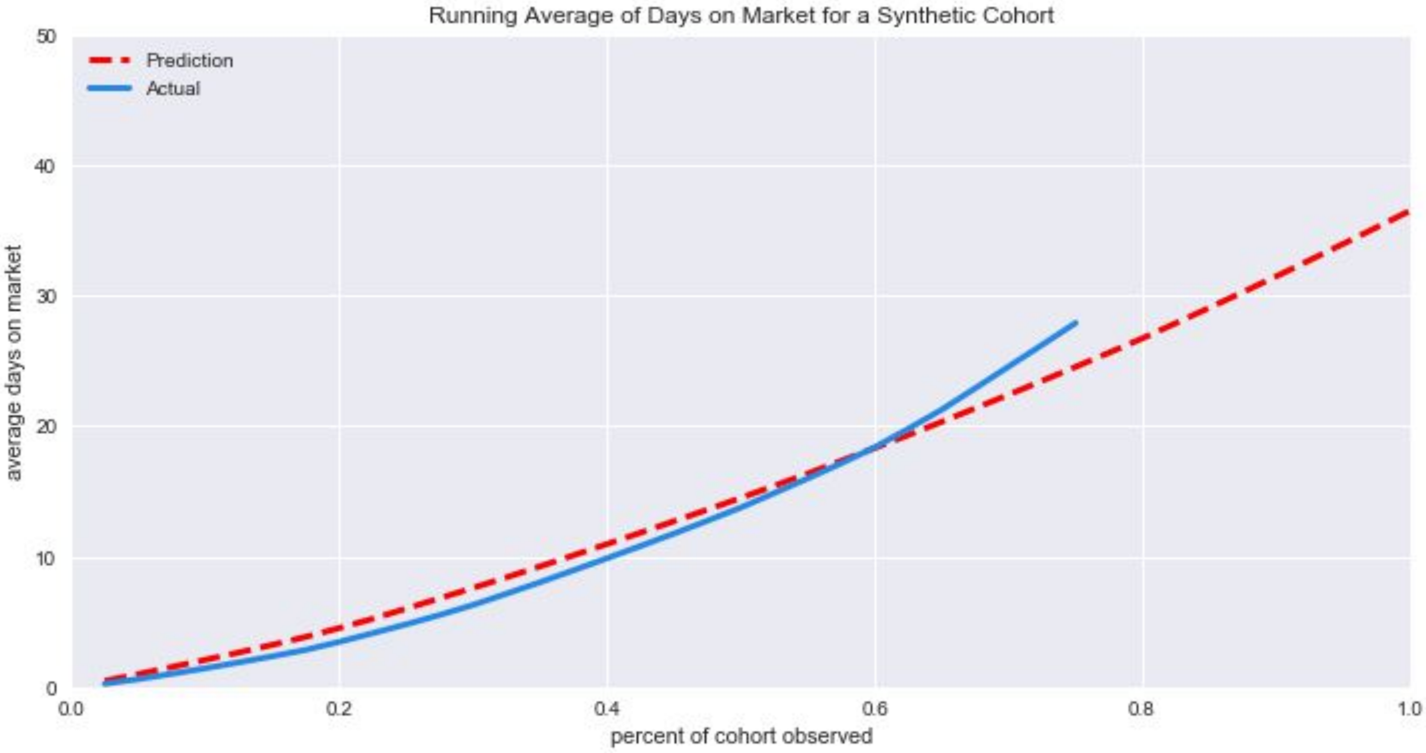


Running Average of Days on Market for a Synthetic Cohort
Prediction
Actual
average days on market
percent of cohort observed
0
10
20
30
40
50
0.0
0.2
0.4
0.6
0.8
1.0

Results

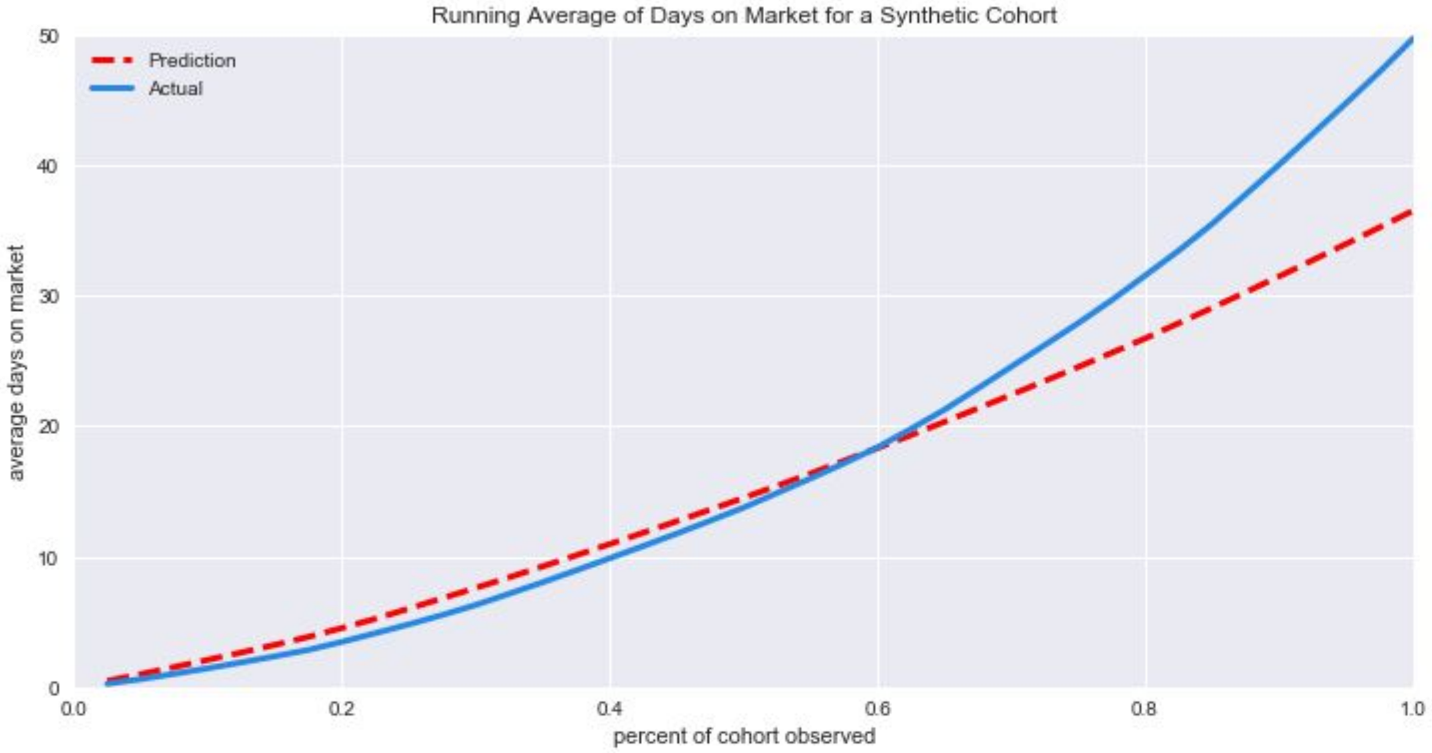

Running Average of Days on Market for a Synthetic Cohort
Prediction
Actual
average days on market
percent of cohort observed
0
10
20
30
40
50
0.0
0.2
0.4
0.6
0.8
1.0

Results



Running Average of Days on Market for a Synthetic Cohort
Prediction
Actual
average days on market
percent of cohort observed
0
10
20
30
40
50
0.0
0.2
0.4
0.6
0.8
1.0

Results



Running Average of Days on Market for a Synthetic Cohort
Prediction
Actual
average days on market
percent of cohort observed
0
10
20
30
40
50
0.0
0.2
0.4
0.6
0.8
1.0

Censoring

Model #1: Linear Regression

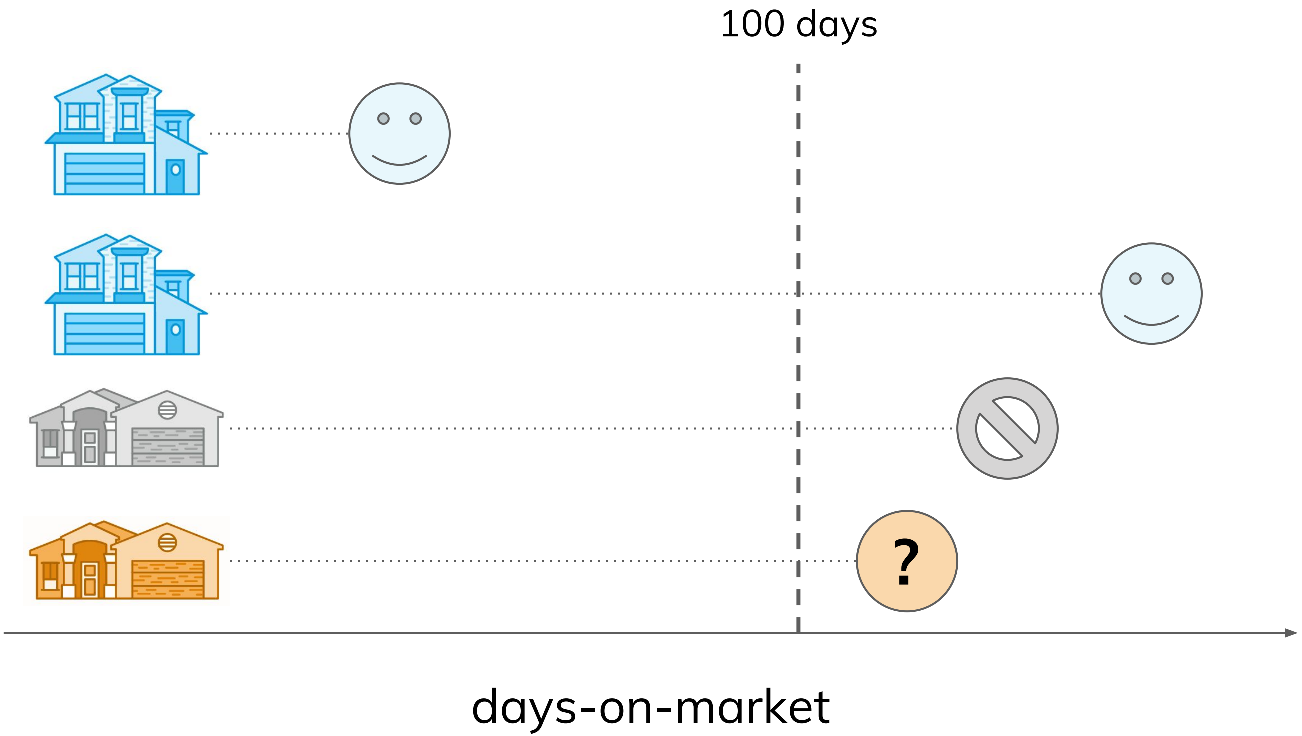
Home	List Price	Square Feet	...	Days-on-market (y)	Explanation
423 Main Street	$200k	2000	....	30	
111 Side Road	$200k	2200	...	100	
...					
52 Downtown Ave	$400k	1945		n/a	Still on market after 200 days
90 Outskirts Lane	$300k	2100		n/a	Delisted after 300 days

Model #1: Takeaway

Pay attention to the negative space

Reframing the Problem

Model #2: Classify "closed before 100 days-on-market"



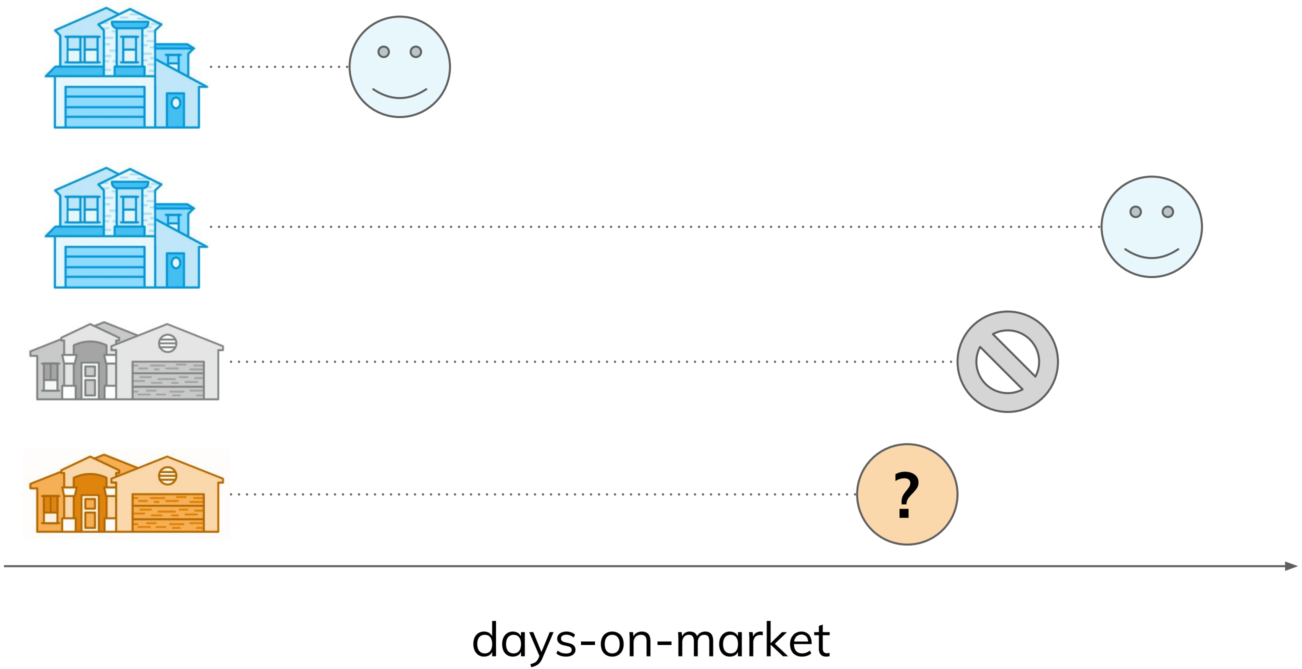
Model #2: Classify "closed before 100 days-on-market"

Home	List Price	...	Days-on-market	Closed Within 100 Days (y)
423 Main Street	$200k	...	30	1
111 Side Road	$200k	...	100	0
...				
52 Downtown Ave	$400k	...	n/a (still on market after 200 days)	0
90 Outskirts Lane	$300k	...	n/a (delisted after 300 days)	0

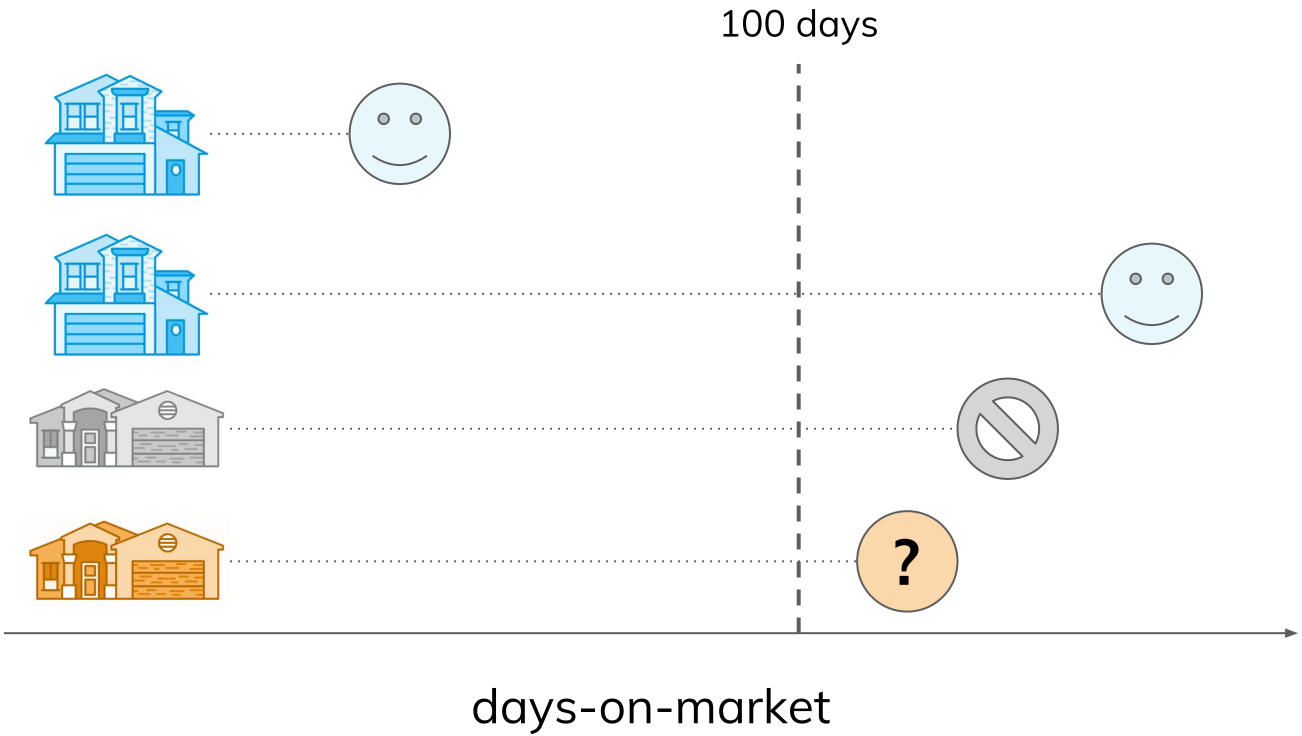
Does it Work?

Pros

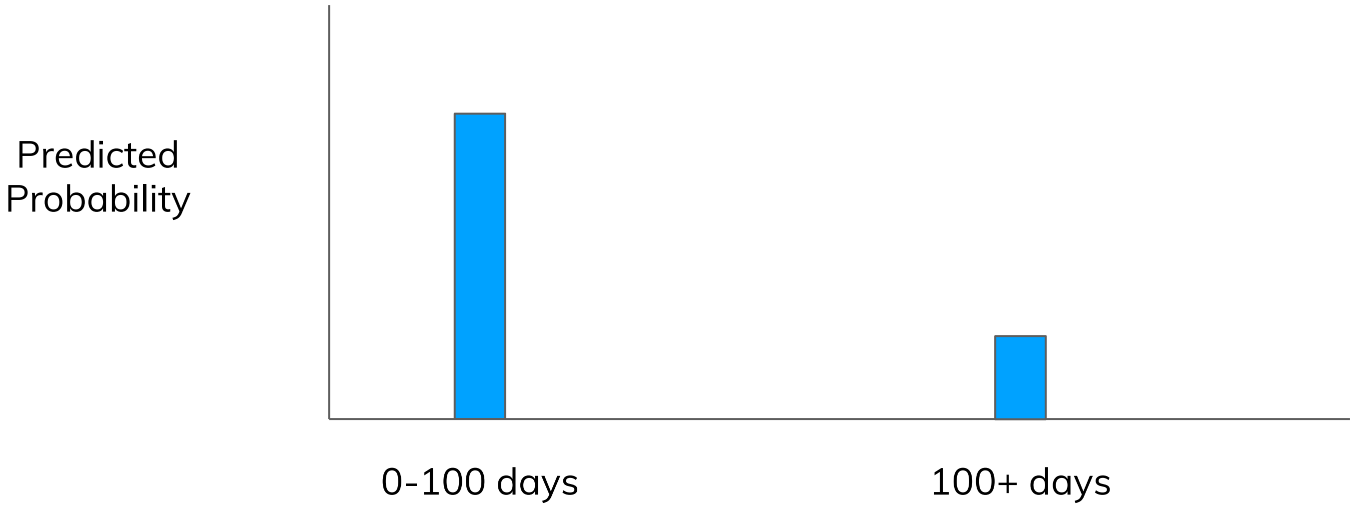
Pro: Easy to Implement



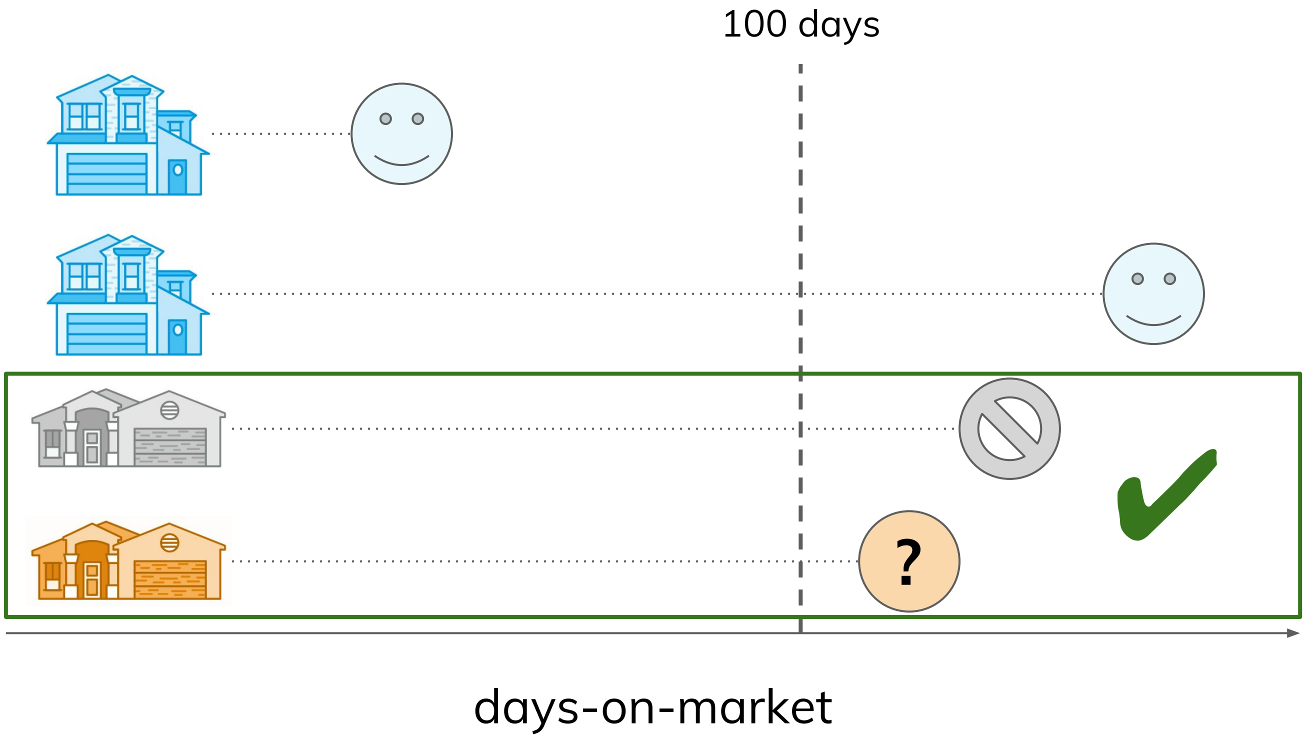
Pro: Easy to Implement - Just Set a Threshold



Pro: Easy-to-interpret Output

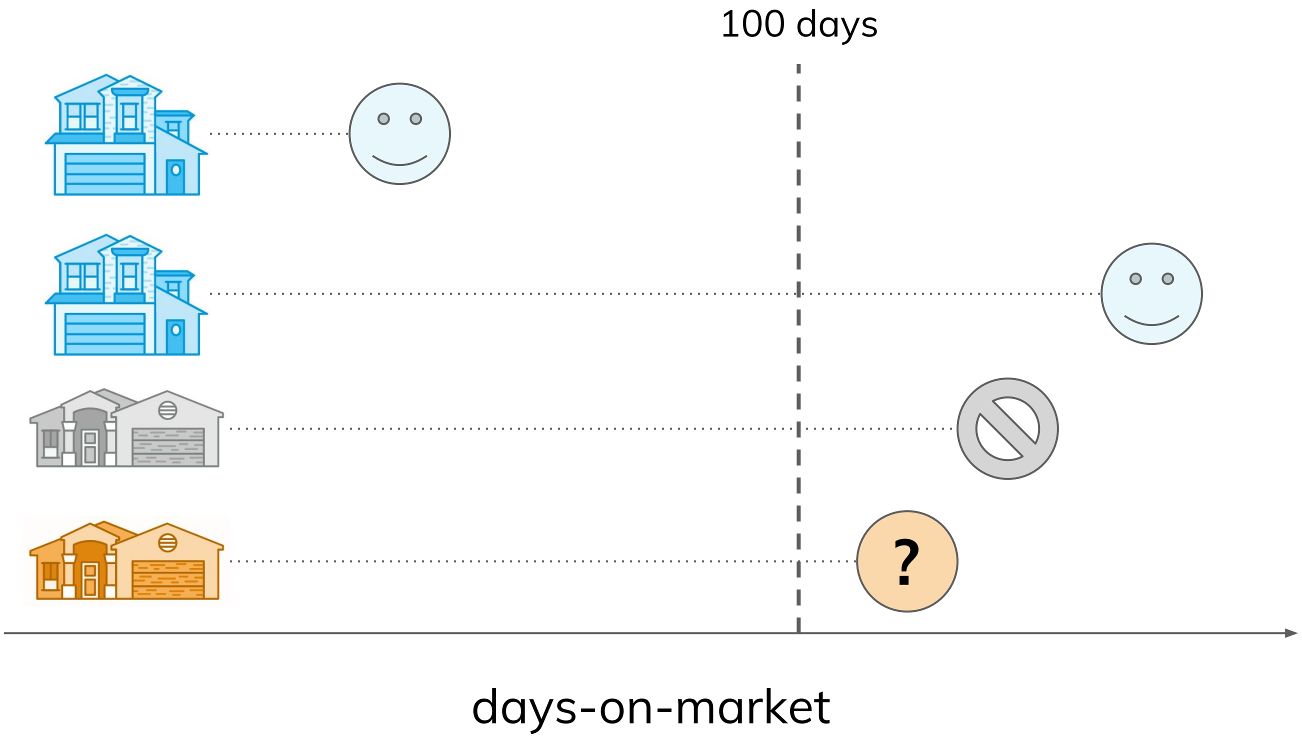


Pro: Uses Censored Data

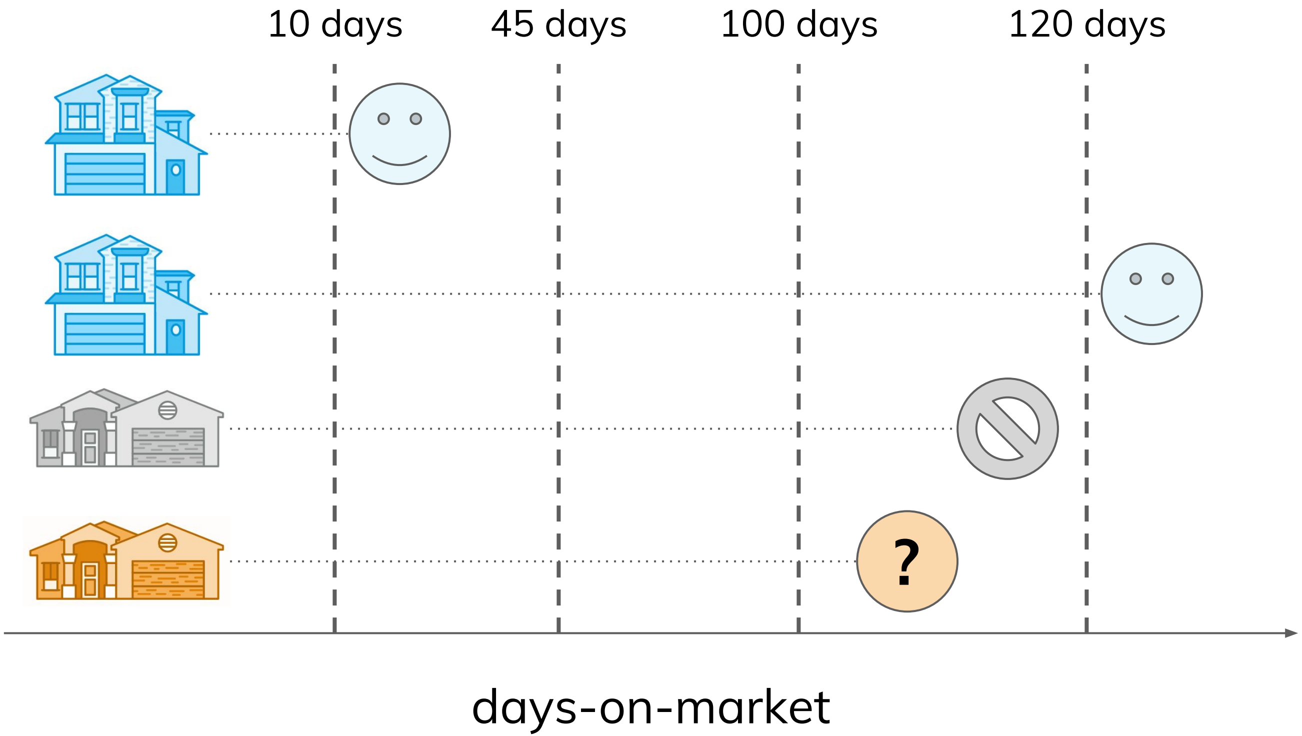


Cons

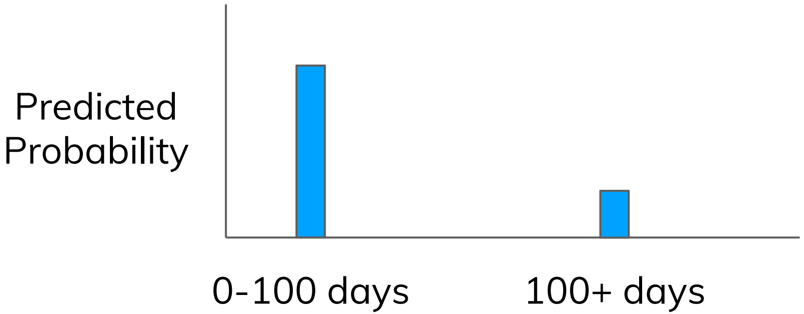
Easy to Implement - Just Set a Threshold



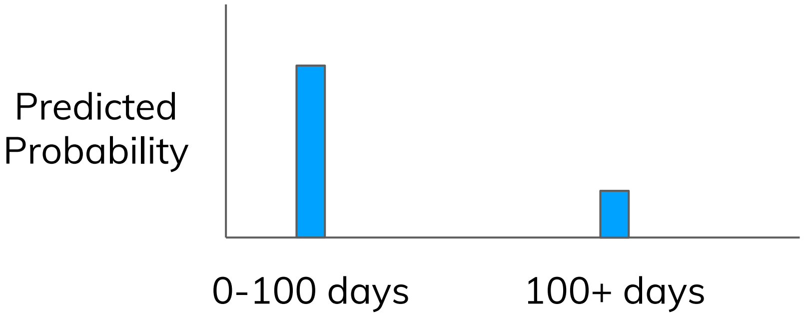
Easy to Implement - Just Set a Threshold - But Which One?



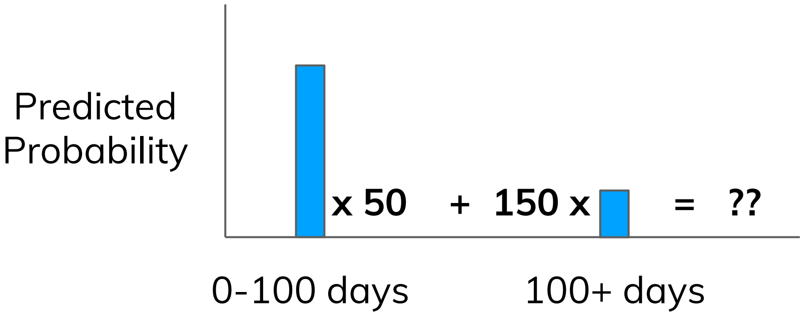
Easy-to-interpret Output



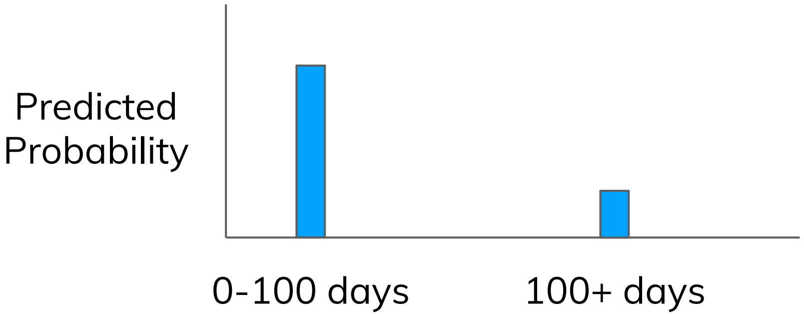
Easy-to-interpret Output



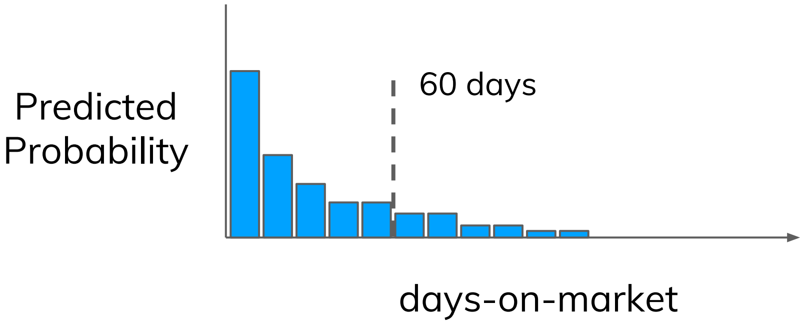
Wrong API



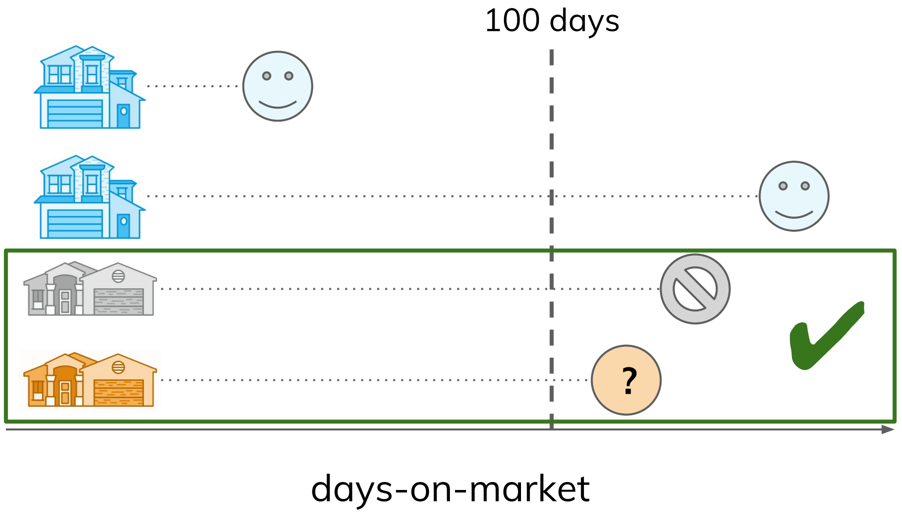
Easy-to-interpret Output



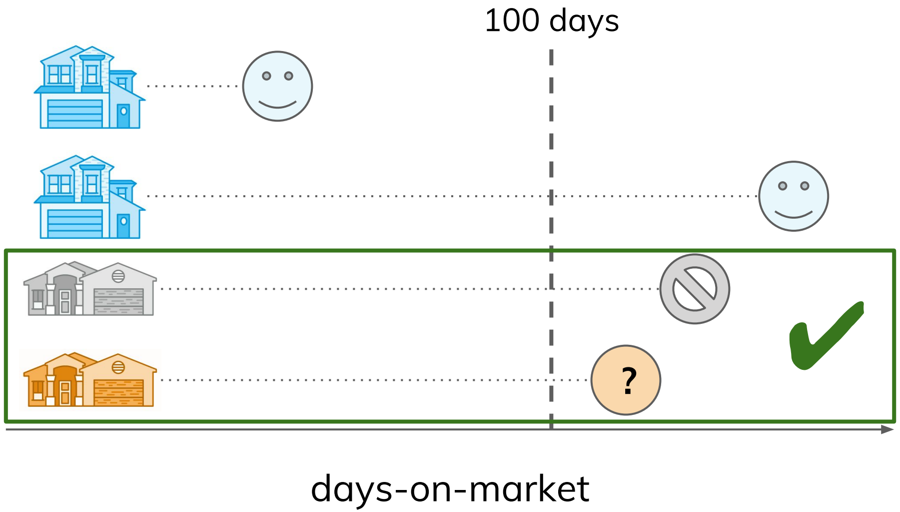
Ideal API

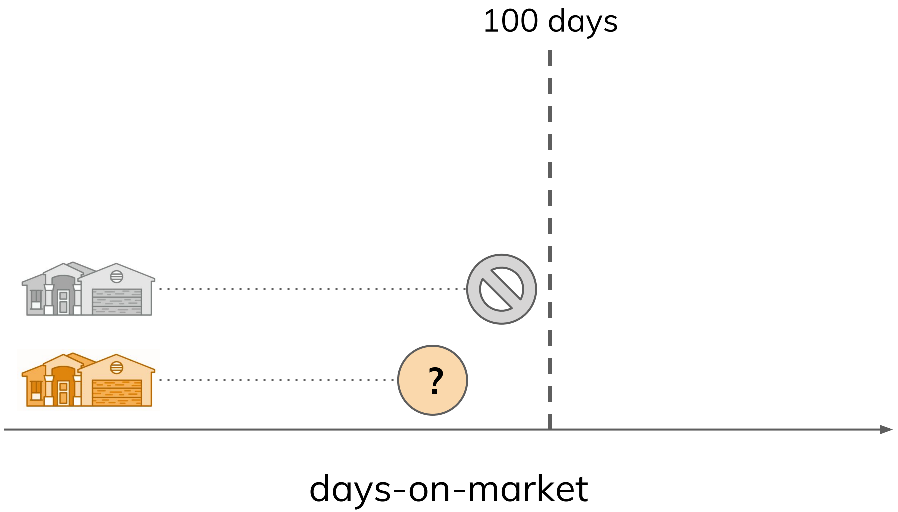


Uses Censored Data



Uses Censored Data (Partially) But Discards Recent Observations





Model #2: Takeaway

Solve a Simpler Problem

Attempt #3

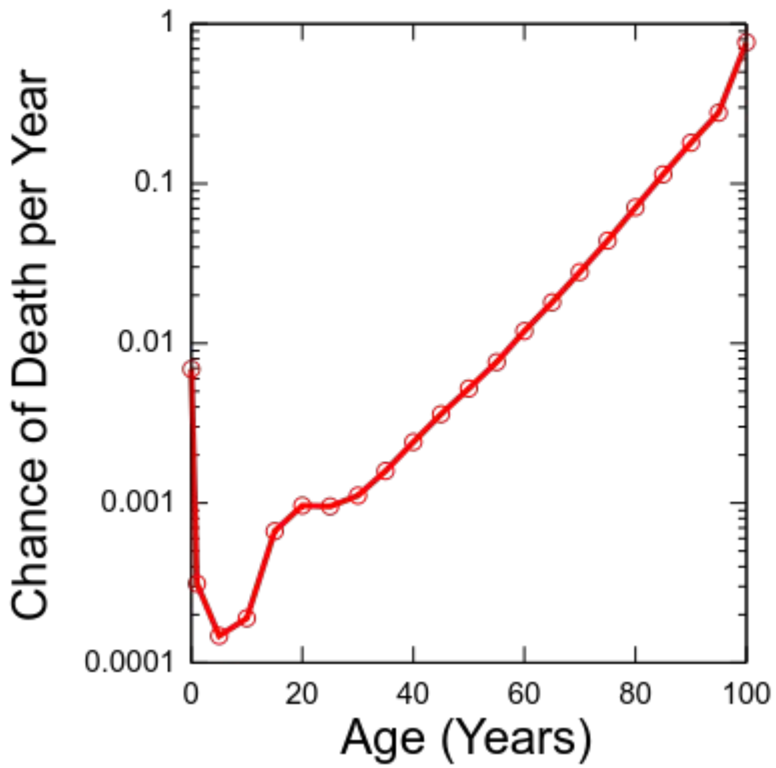
Survival Analysis

When stuck, see if someone has already solved the problem...

Actuaries & medical professionals are interested in

- What is the life expectancy of the population of city A?
- What is the probability of person B surviving the next decade?
- Given person C is 70 years old, what is his/her life expectancy?

Censored data is *always* an issue.



In this analogy, “death” is a happy event of finding a buyer:

Actuaries & medical professionals are interested in

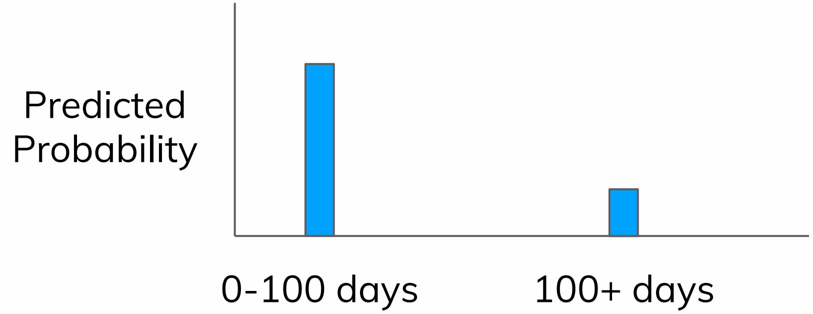
- What is the life expectancy of the population of city A?
- What is the probability of person B surviving the next decade?
- Given person C is 70 years old, what is his/her life expectancy?

Opendoor is interested in

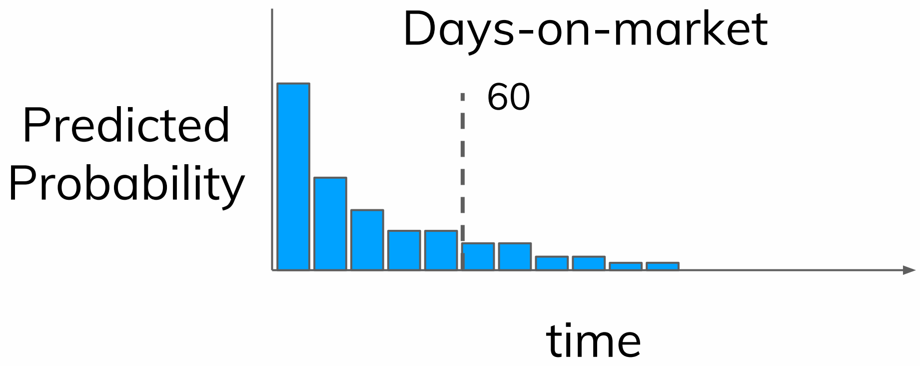
- What is the expected days on market for all listings in city A?
- What is the probability of listing B taking 10 more days to sell?
- Given listing C was on market for 70 days, how much longer until we expect to find a buyer?

Previously....

Predicted Days-on-market = **45**



With survival analysis...



Model #3: Takeaway 1

Look for Existing Solutions to *Similar* Problems

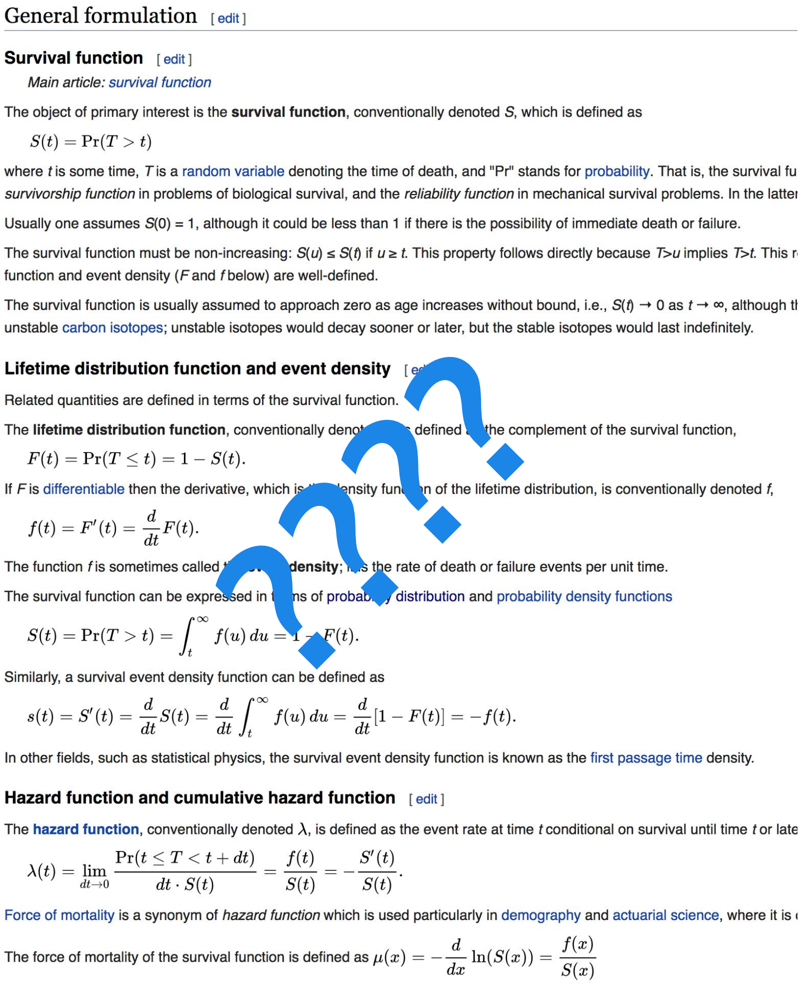
We found the right approach, but...

Hurdle #1

It's not easy to explain

The fundamental concepts requires calculus to explain well

Limited intuition and tie-ins to tangible concepts for decision makers



General formulation [edit]

Survival function [edit]

Main article: survival function

The object of primary interest is the **survival function**, conventionally denoted *S*, which is defined as

$$S(t) = \Pr(T > t)$$

where *t* is some time, *T* is a random variable denoting the time of death, and "Pr" stands for probability. That is, the survival fu
survivorship function in problems of biological survival, and the *reliability function* in mechanical survival problems. In the latter

Usually one assumes $S(0) = 1$, although it could be less than 1 if there is the possibility of immediate death or failure.

The survival function must be non-increasing: $S(u) \le S(t)$ if $u \ge t$. This property follows directly because $T>u$ implies $T>t$. This r
function and event density (*F* and *f* below) are well-defined.

The survival function is usually assumed to approach zero as age increases without bound, i.e., $S(t) \to 0$ as $t \to \infty$, although tl
unstable carbon isotopes; unstable isotopes would decay sooner or later, but the stable isotopes would last indefinitely.

Lifetime distribution function and event density [e

Related quantities are defined in terms of the survival function.

The **lifetime distribution function**, conventionally deno defined the complement of the survival function,

$$F(t) = \Pr(T \le t) = 1 - S(t).$$

If *F* is differentiable then the derivative, which is ensity fun n of the lifetime distribution, is conventionally denoted *f*,

$$f(t) = F'(t) = \frac{d}{dt}F(t).$$

The function *f* is sometimes called **density**; s the rate of death or failure events per unit time.

The survival function can be expressed in t ns of probab distribution and probability density functions

$$S(t) = \Pr(T > t) = \int_t^\infty f(u)\,du = 1 - F(t).$$

Similarly, a survival event density function can be defined as

$$s(t) = S'(t) = \frac{d}{dt}S(t) = \frac{d}{dt}\int_t^\infty f(u)\,du = \frac{d}{dt}[1 - F(t)] = -f(t).$$

In other fields, such as statistical physics, the survival event density function is known as the first passage time density.

Hazard function and cumulative hazard function [edit]

The **hazard function**, conventionally denoted λ, is defined as the event rate at time *t* conditional on survival until time *t* or late

$$\lambda(t) = \lim_{dt \to 0} \frac{\Pr(t \le T < t + dt)}{dt \cdot S(t)} = \frac{f(t)}{S(t)} = -\frac{S'(t)}{S(t)}.$$

Force of mortality is a synonym of *hazard function* which is used particularly in demography and actuarial science, where it is

The force of mortality of the survival function is defined as $\mu(x) = -\frac{d}{dx}\ln(S(x)) = \frac{f(x)}{S(x)}$

Hurdle #2

Scaling is hard with existing tools

- Lots of R packages
- Limited options for production-ready languages
- Works great for small dataset; broke down with larger ones

Hurdle #3

Modeling flexibility is hard with existing tools

- Off-the-shelf packages: model choices are limited (proportional or additive hazard models)
  - Non-flexible feature specification
  - Hard to implement time-varying features
  - ...
- Markov Chain Monte Carlo (Stan): complete freedom of model specification, but
  - Took hours to train on a tiny dataset
  - Hard to maintain

Let's try to reformulate the problem

Survival analysis made easy

Instead of telling you about…

S(t), λ(t), Cox Proportional Models, Kaplan-Meier, …

We will show you a reformulation that

- Easily scalable to large datasets
- More concretely tied to real life numbers
- Equivalent*
- Allows flexible modeling extension

* with some hand-waving. Rigorous proof left to mathematicians in the audience as an exercise.

Changing target again

Home	Ini. List Price	…	Days-on-market
423 Main Street	$200k	….	30

Changing target again

Home	Ini. List Price	...	~~Days on market~~	"Current" days on market	Sold in the next day (y)
423 Main Street	$200k	....	~~30~~	0	0
423 Main Street	$200k	....	~~30~~	1	0
423 Main Street	$200k	....	~~30~~	2	0
...					
423 Main Street	$200k	....	~~30~~	28	0
423 Main Street	$200k	....	~~30~~	29	1

30 new data rows

Changing target again

Home	Ini. List Price	...	~~Days-on-market~~	"Current" days on market	Sold in the next day (y)
423 Main Street	$200k	....	~~30~~	0	0
423 Main Street	$200k	....	~~30~~	1	0
423 Main Street	$200k	....	~~30~~	2	0
...					
423 Main Street	$200k	....	~~30~~	28	0
423 Main Street	$200k	....	~~30~~	29	1
52 Downtown Ave	$400k	...	Still on market after 200 days		

30 rows

Changing target again

Home	Ini. List Price	...	~~Days on market~~	"Current" days on market	Sold in the next day (y)
423 Main Street	$200k	....	~~30~~	0	0
423 Main Street	$200k	....	~~30~~	1	0
423 Main Street	$200k	....	~~30~~	2	0
...					
423 Main Street	$200k	....	~~30~~	28	0
423 Main Street	$200k	....	~~30~~	29	1
52 Downtown Ave	$400k	...	~~n/a~~	0	0
...					
52 Downtown Ave	$400k	...	~~n/a~~	199	0

30 rows

200 rows

Change fundamental unit of data

listings ⇒ listing-days

<u>*All*</u> listing data are used:
closed, active, delisted…

Binary classification to the rescue, again

We transformed the problem into vanilla binary classification

- Pick your favorite binary classifier, as long as
  - Log-loss minimizing
  - Calibrated probabilities
- Scalability ✔ (even though we made the dataset larger!)

How to interpret?

Prediction = probability of listing closing in the next day

(hazard rate in survival analysis parlance)

Prediction = housing clearance rate, a.k.a. inventory turnover rate

if we start with 100 homes on market today, how many will close before the end of the day/week/month/year?

✔ Model output ties directly to real world numbers, no calculus needed!

How to interpret? (cont'd)

Prediction, a.k.a. the hazard rate, is the building block

hazard rate + laws of probabilities = everything we want to know

Example: expected days on market

For each listing, we have a series of predictions (h_1, h_2, h_3, h_4, …) for each day

$$E[y] = \sum y \times P(y)$$

$$= 1 \times \boxed{h_1} + 2 \times \boxed{(1 - h_1)\ h_2} + 3 \times (1 - h_1)\ (1 - h_2)\ h_3 + 4 \times \ldots + \ldots$$

P(closing on day 1)

P(days-on-market = 2)
= P(not closing on day 1) × P(closing on day 2)

Model #3: Takeaway 2

Complex modeling technique doesn't always need complex implementation

How does it work?

Minimizing log loss in binary classification:

$$-\frac{1}{N}\sum_{i=1}^{N}[y_i \log p_i + (1-y_i)\log(1-p_i)]$$

Only one term matters depending on label ($y_i = \{0, 1\}$)

Maximizing log-likelihood estimate:

P(data | model) = P(listing$_1$ on market for D1 days|model) *

P(listing$_2$ on market for D2 days|model) * ...

$= (1-h_{11})(1-h_{12})\ldots h_{1D1} \;*\; (1-h_{21})(1-h_{22})\ldots h_{2D2}$

$\log P(\text{data} \mid \text{model}) = \log(1-h_{11}) + \log(1-h_{12}) \ldots + \log(h_{1D1}) + \ldots$

= **Spoiler alert - look at log loss function** ✔

We will show you a reformulation that is

- ✔ Easily scalable to large datasets
- ✔ More concretely tied to real life numbers
- ✔ Equivalent
- Allows flexible modeling extension

Time varying features

e.g. how does pricing change liquidity?

Not straightforward to implement in off-the-shelf survival analysis models

Cox's Time Varying Proportional Hazard model

Warning

This implementation is still experimental.

Often an individual will have a covariate change over time. An example of this is hospital patients who enter the study and, at some future time, may recieve a heart transplant. We would like to know the effect of the transplant, but we cannot condition on whether they recieved the transplant naively. Consider that if patients needed to wait at least 1 year before getting a transplant, then everyone who dies before that year is considered as a non-transplant patient, and hence this would overestimate the hazard of not recieving a transplant.

We can incorporate changes over time into our survival analysis by using a modification of the Cox model above. The general mathematical description is:

$$\lambda(t|X) = b_0(t) \exp\left(\sum_{i=1}^{d} b_i x_i(t)\right)$$

Note the time-varying $x_i(t)$ to denote that covariates can change over time. This model is implemented in lifelines as `CoxTimeVaryingFitter`. The dataset schema required is different than previous models, so we will spend some time describing this.

Dataset for time-varying regression

Lifelines requires that the dataset be in what is called the *long* format. This looks like one row per state change, including an ID, the left (exclusive) time point, and right (inclusive) time point. For example, the following dataset tracks three unique subjects.

id	start	stop	group	z	event
1	0	8	1	0	False
2	0	5	0	0	False
2	5	8	0	1	True
3	0	3	1	0	False
3	3	12	1	1	True

5 rows × 6 columns

In the above dataset, `start` and `stop` denote the boundaries, `id` is the unique identifier per subject, and `event` denotes if the subject died at the end of that period. For example, subject ID 2 had variable `z=0` up to and including the end of time period 5 (we can think that measurements happen at end of the time period.), after which it was set to 1.

So if this is the desired dataset, it can be built up first from smaller datasets. To do this we can use some helper functions provided in lifelines.

Typically, data will be in a format that looks like it comes out of a relational database. You may have a "base" table with ids, durations, and a censored flag, and possibly static covariates. Ex:

id	duration	event	var1
1	10	True	0.1
2	12	False	0.5

2 rows × 4 columns

You'll also have secondary dataset that reference taking future measurements. Example:

id	time	var2
1	0	1.4
1	4	1.2

1	10	True	0.1
2	12	False	0.5

2 rows × 4 columns

You'll also have secondary dataset that reference taking future measurements. Example:

id	time	var2
1	0	1.4
1	4	1.2
1	8	1.5
2	0	1.6

4 rows × 3 columns

where `time` is the duration from the entry event. Here we see subject 1 had a change in their `var2` covariate at the end of time 4 and at the end of time 8. We can use `to_long_format` to transform the base dataset into a long format and `add_covariate_to_timeline` to fold the covariate dataset into the original dataset.

```
from lifelines.utils import to_long_format
from lifelines.utils import add_covariate_to_timeline

base_df = to_long_format(base_df, duration_col="T")
df = add_covariate_to_timeline(base_df, cv, duration_col="time", id_col="id", event_
```

id	start	stop	var1	var2	event
1	0	4	0.1	1.4	False
1	4	8	0.1	1.2	False
1	8	10	0.1	1.5	True
2	0	12	0.5	1.6	False

4 rows × 6 columns

From the above output, we can see that subject 1 changed state twice over the observation period, finally expiring at the end of time 10. Subject 2 was a censored case, and we lost them after time 2.

You may have multiple covariates you wish to add, so the above could be streamlined like so:

```
from lifelines.utils import to_long_format
from lifelines.utils import add_covariate_to_timeline

base_df = to_long_format(base_df, duration_col="T")
df = base_df.pipe(add_covariate_to_timeline, cv1, duration_col="time", id_col="id",
            .pipe(add_covariate_to_timeline, cv2, duration_col="time", id_col="id",
            .pipe(add_covariate_to_timeline, cv3, duration_col="time", id_col="id",
```

One additional flag on `add_covariate_to_timeline` that is of interest is the `cumulative_sum` flag. By default it is False, but turning it to True will perform a cumulative sum on the covariate before joining. This is useful if the covariates describe an incremental change, instead of a state update. For example, we may have measurements of drugs administered to a patient, and we want to the covariate to reflect how much we have administered since the start. In contrast, a covariate the measure the temperature of the patient is a state update. See Example cumulative total using ``add_covariate_to_timeline`` to see an example of this.

For an example of pulling datasets like this from a SQL-store, and other helper functions, see Example SQL queries and transformations to get time varying data.

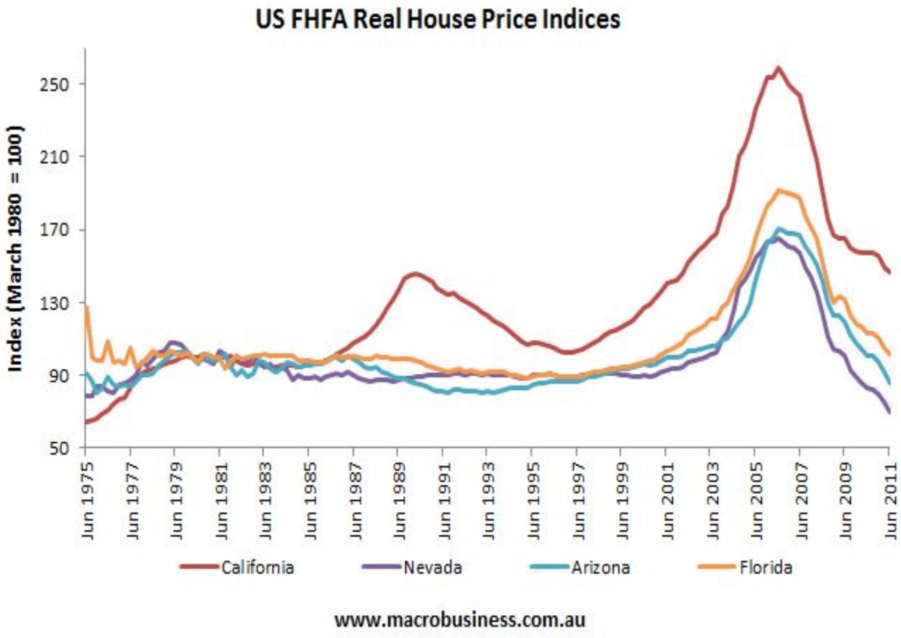
Time varying features

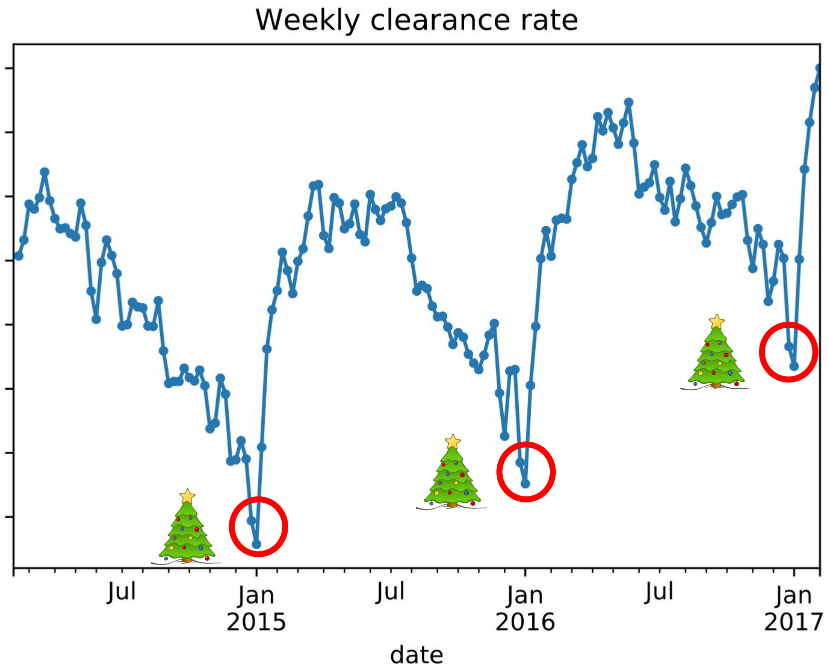
e.g. how does pricing change liquidity?

Home	Ini. List Price	"Current" list price	...	~~Days-on-market~~	"Current" days on market	Sold in the next day (y)
423 Main Street	$200k	$200k	....	~~30~~	0	0
423 Main Street	$200k	$200k	....	~~30~~	1	0
423 Main Street	$200k	$190k	....	~~30~~	2	0
...						
423 Main Street	$200k	$170k	....	~~30~~	28	0
423 Main Street	$200k	$170k	....	~~30~~	29	1

Time series analysis

~~Real life~~ housing data is not stationary

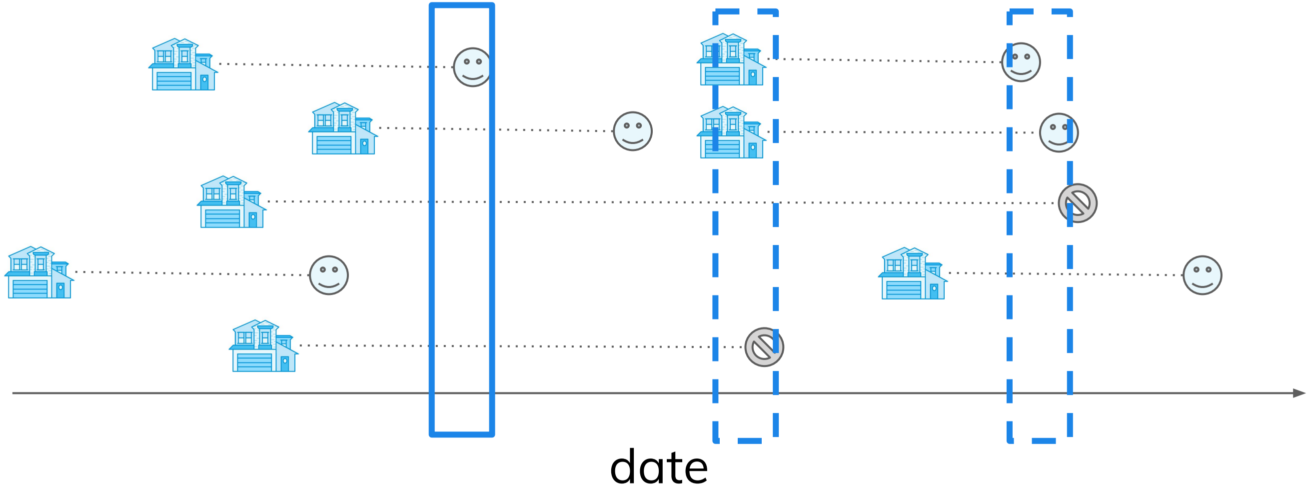




Time series analysis

- Tricky to implement in traditional survival analysis
- Listing centric view doesn't work well

Instead, train a series of models using snapshot of listings at time t



then interpolate predictions using time series techniques

Model #3: Takeaway 3

Divide and conquer

Break problem down to interpretable intermediate steps

When You Have a Hammer,
Everything Looks Like ~~a Nail~~ Survival Analysis

Survival analysis is broadly useful

Churn prediction / user lifetime analysis

- Not just if, but *when and with what probability*, a user leaves
- Full probability distribution to compute lifetime value of customers

Credit / Loan default

- Default early or default later in the loan?

System reliability

- What are the distribution of lifetime of hard drives?

Any time-to-event prediction!

Ask you doctor if survival analysis is right for you ...

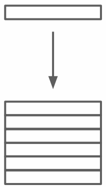
- You want to model time-to-event, or even just binary classification
- You work with censored data
- You value a full probability distribution instead of point estimate
- Time is a confounding factor (cohorts, mix shift, ...)

If survival analysis is right for you, it can be easy to use!

We've shown you a reformulation that

- ✔ Easily scalable to large datasets
- ✔ More concretely tied to real life numbers
- ✔ Allows flexible modeling extension

1. Transform



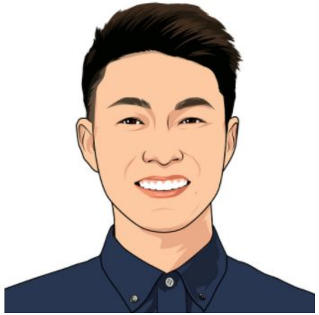
2. Binary Classify

0
0
0
1
0
0
. . .

Thanks!

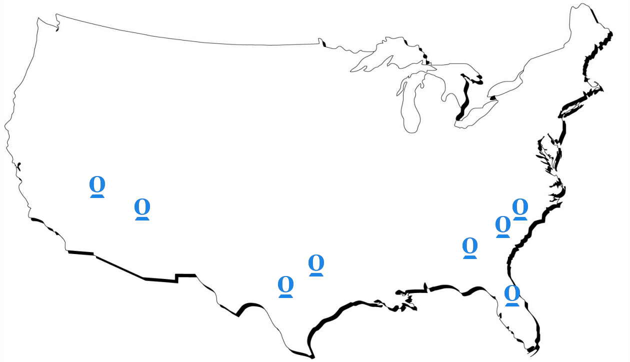





Join us at Opendoor as we change real estate!

- Founded in 2014
- $100M+ transactions per month
- In rapid expansion mode - we're currently in 8 cities (more coming!)
- We are hiring engineers and data scientists



Please contact us: dave@opendoor.com & xinlu@opendoor.com