Dockless bikes:

Flexible, adaptable, easy to deploy

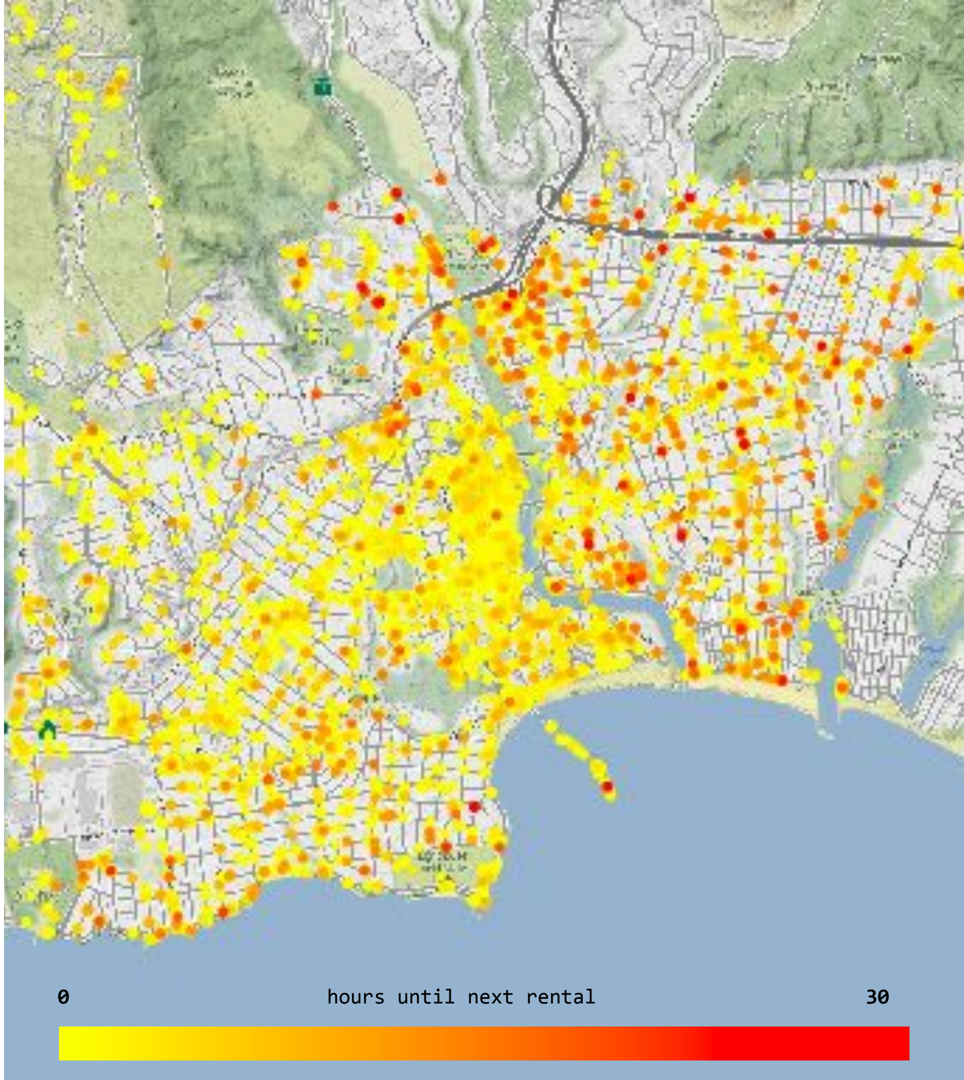
The challenge:

Idle bikes in low traffic areas
lose money and customers

Go get that bike!

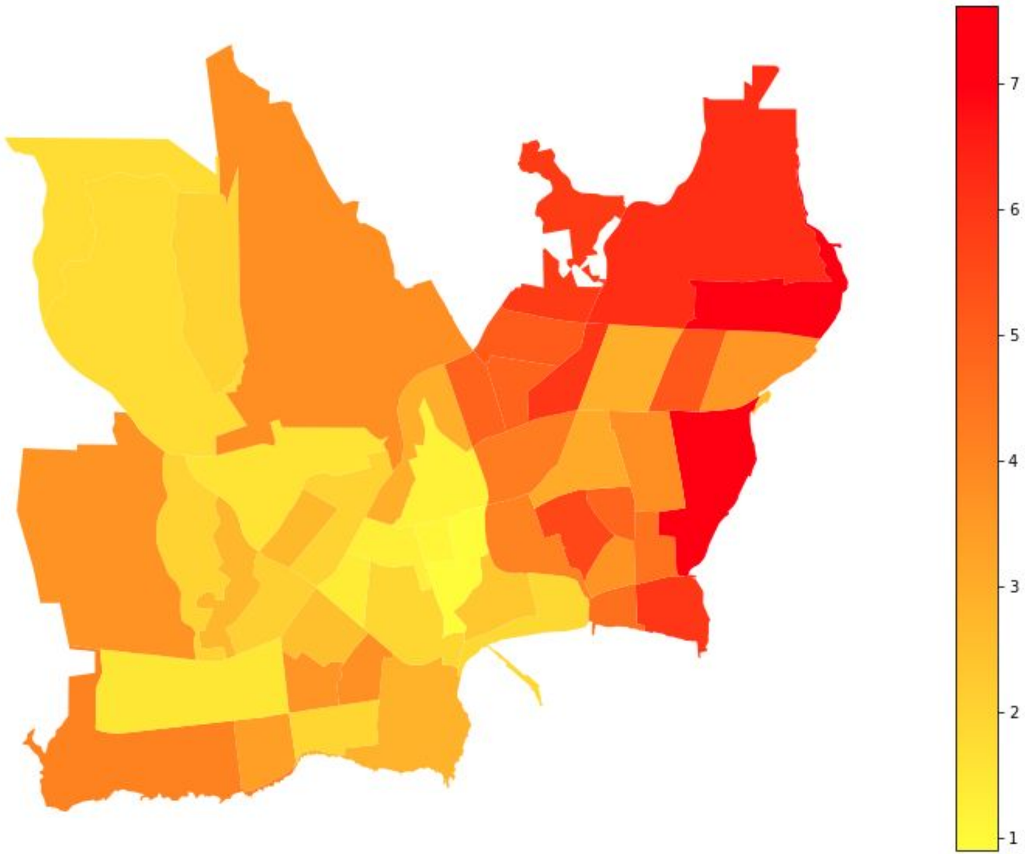
Finding underused bikes in dockless bikeshare systems

Katelyn Walker



Process

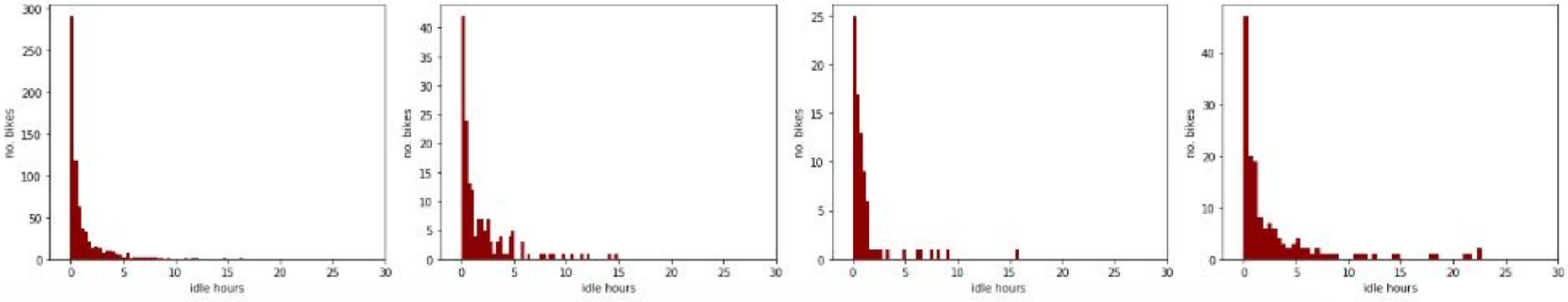
- Realtime data collection from the JUMP API, collected using an AWS EC2 server
- Aggregate data to create idle_time target, add geospatial information using shapely and geopandas
- Add additional features from spatial datasets (city zoning, US Census data)



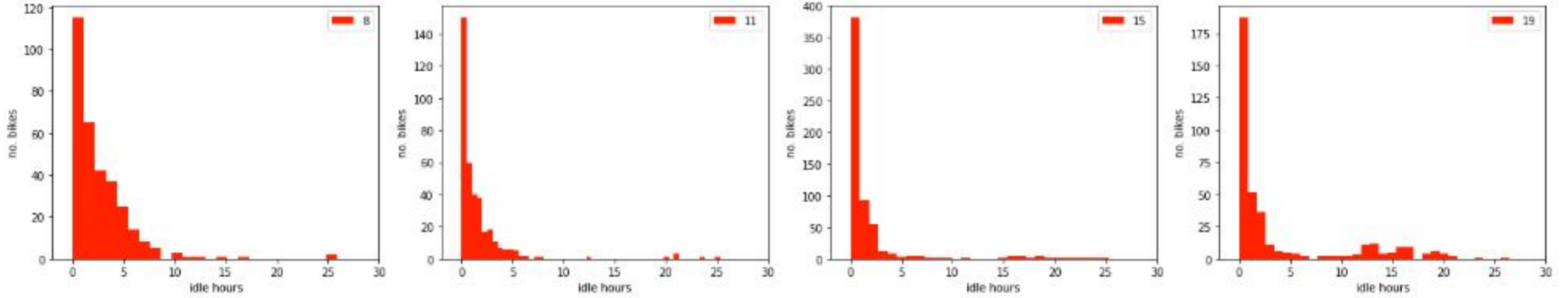
Mean hours until next rental

By census blockgroup

It's Exponential!



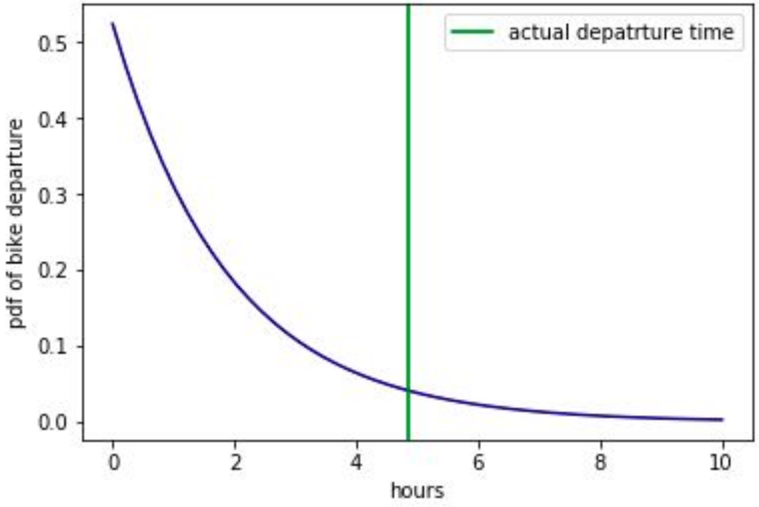
Count of hours until next rental, by census blockgroup

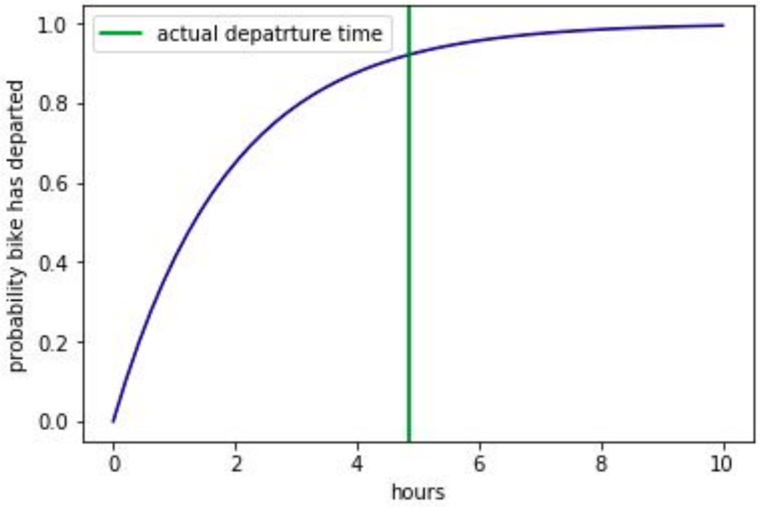


Count of hours until next rental, by time of day

Estimating the rate of departure

K-nearest neighbors to estimate the beta parameter (which is equal to the mean departure time!)





Profit modeling - when to move that bike?

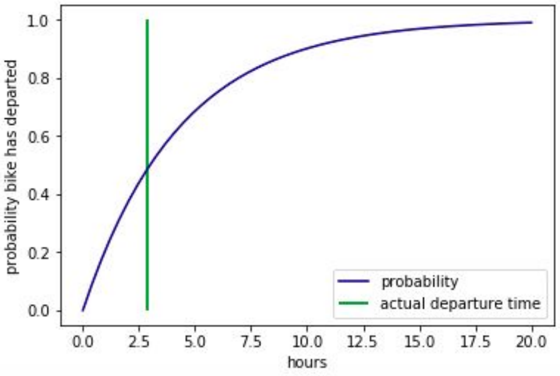
Assume a bike with normal usage makes $0.50/hr (rough estimate from average usage of bikes)

Assume a $5 cost to move a bike

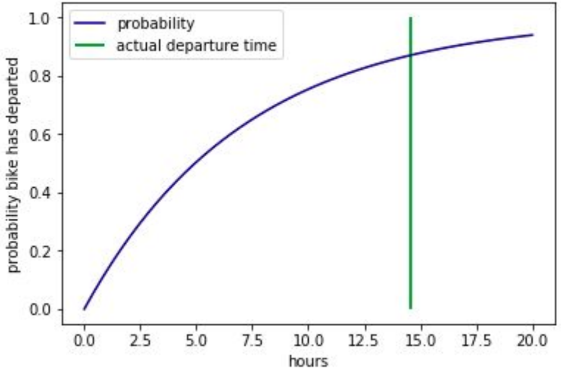
Break even is 10 hours of idle time.

At what probability are we correct enough to make the most profit?

One bike



Another bike

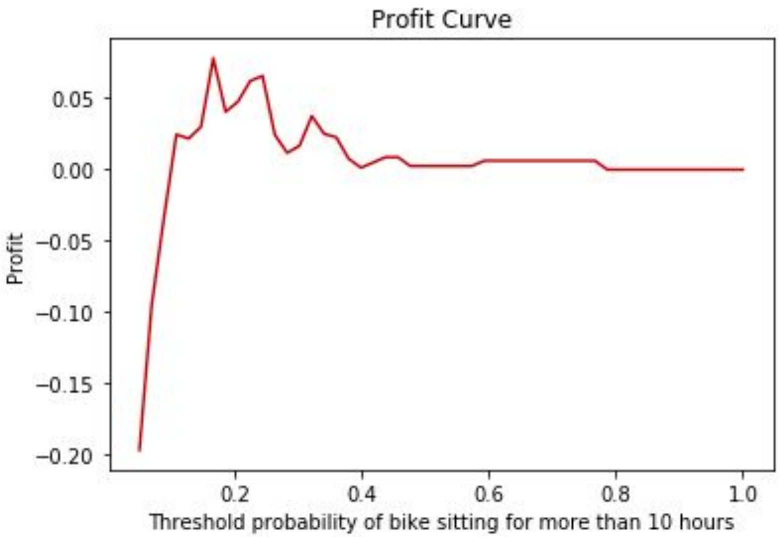


Profit modeling - when to move that bike?

To maximize profit, use a threshold 16% probability that a bike is going to sit for 10 hours.

Probability bike sits for 10 hours or more

Did bike actually sit for 10 hours?	>16% (relocate bike)	<16% (do nothing)
Yes	**50 bikes** (made money on these!)	**152 bikes**
No	**64 bikes** (lost money on these)	**1028 bikes**



Another use case - detecting broken or hidden bikes

Once a bike exceeds the time at which it was 95% likely to depart, it might be time to send someone out to check on it!

Mean idle time for all bikes: **2.8 hours**

Mean additional idle time once a bike has exceeded 95% probability of departure: **5.8 hours**

Further expansions:

Include demand