Scraping Real-Time Micromobility Data to Model Vehicle Time to Activation

Elisabeth Ericson
May 2022

Introduction

As an emergent urban transportation mode, dockless shared mobility was virtually unheard of until the last five years. Since then, shared electric scooters introduced to city streets by private companies have drawn riders, but also backlash; controversy over growth and management has led many cities to impose increasingly onerous restrictions on the vehicles' deployment and operation. Simultaneously, a growing number of urban bikeshare systems now offers dockless electric bicycles interoperable with their existing docking station infrastructure, and dockless vehicle advocates advocates have pointed to to the potential for shared micromobility to provide a first- and last-mile car alternative for urban neighborhoods poorly served by transit and traditional bikeshare.

For this project, I was interested in exploring what insights about the spatial distribution of dockless vehicle usage could be derived from publicly available General Bikeshare Feed Specification (GBFS) data. To that end, I scraped real-time dockless vehicle location data from public APIs for the city of Washington, D.C.; built a pipeline that combined cloud and local computing to extract, transform, and load the data; and created a proof-of-concept model using survival analysis to predict vehicle time to activation at the individual vehicle level.

As this work shows, studying micromobility usage patterns is made difficult by data anonymization measures and inconsistent implementation of the GBFS standard. However, collecting real-time data on inactive vehicles can nonetheless provide valuable information – either directly, with respect to the spatial and temporal distribution of access to shared mobility services, or indirectly, if considering time to activation as a proxy for hyperlocal demand, or lack thereof. In this report, I outline a data collection and processing framework that I hope to build on in future work, and find promise in survival analysis as a method to model shared vehicle time to activation.

Question definition

In effect, this project sought to answer two separate but related questions: First, given that public data on shared dockless vehicle use is limited and deidentified, how can I use high-frequency scraping of real-time inactive vehicle feeds to gain insight into the geography of shared micromobility even in the absence of high-resolution origin-destination trip data? Second, how would I use this data to build a model to predict how long an inactive vehicle will remain idle between trips?

For the first question, I was motivated by technical curiosity and a desire to learn new skills by undertaking a more ambitious data gathering project than anything I'd attempted before. (As it turned out, I had no idea at the outset just how far outside my previous experience it would take me.) The second question is one I've most often seen addressed from the perspective of the operator, whose interest is in efficiently allocating vehicles across space in response to demand, but I was inspired by my own experience as a rider of shared dockless vehicles to instead take the perspective of the user, whose interest is in not missing out on a nearby bike or scooter because someone else got to it first.

Literature review

I searched peer-reviewed literature addressing dockless shared mobility from several angles, including the spatial distribution of dockless vehicle access from an equity standpoint, the shifting landscape around open data for micromobility, and the use of survival analysis to model shared vehicle time to activation. I summarize a selection of this research below.

Spatial distribution and equity of access

The research I found comparing docked and dockless bikeshare generally suggested that dockless bikeshare improves accessibility in underserved neighborhoods compared to traditional docked bikeshare systems. Analyzing data from 32 U.S. cities with both docked and dockless micromobility, Meng and Brown (2021) found that "the distribution of docked systems is extremely unequal, and that dockless systems greatly reduce geographical inequalities relative to docked." In San Francisco, Qian et al. (2020) found that dockless bikeshare provided greater access to bikes in disadvantaged "Communities of Concern" (CoC) than docked bikeshare did, and also helped mitigate the bikeshare usage gap between CoCs and other communities. Lazarus et al. (2020) found that San Francisco dockless electric bikes "more heavily serviced lower-density areas" – which tend to have lower transit accessibility – than docked bikes did.

Despite that, inequality can still be present in dockless systems: in a study of dockless bikeshare in the Boston suburbs, Gehrke et al. (2021) found that areas with higher shares of rental housing units and minority residents generated more dockless bike trips, yet had lower access to the bikes. Previously, in a Seattle dockless bikeshare pilot program, Mooney et al. (2019) found that neighborhoods with more bikes had more college-educated residents, higher median incomes, and higher access to community resources, but "did not find disparities by racial/ethnic composition or risk of displacement" and concluded that dockless bikeshare systems "hold promise for offering equitable spatial access to bike sharing."

- address private micromobility providers, not just bikeshare
- that one study finding longer inactivity in lower-income areas

The shifting open data landscape

High-frequency scraping of GBFS feeds is an established technique in the micromobility literature, but the extent of publicly available data has shifted in response to privacy concerns. Zou et al. (2020) reconstructed dockless scooter trip trajectories by scraping, at 30-second intervals, the data of one operator whose GBFS feed included the coordinates of *active* scooters; this data is no longer available.

Moreover, as noted under "Data" below, to prevent bad actors from potentially inferring individual riders' movements, even inactive vehicle identifiers are increasingly randomized. McKenzie (2019 and 2020) extrapolated trip origins and destinations from scraped inactive vehicle data by matching persistent identifiers as they disappeared and reappeared in different locations over time. Xu et al. (2020) proposed algorithms to infer trip origins and destinations even for dynamic vehicle IDs, but not all randomization methods permit for origins and destinations to be paired. In other words, point-to-point trip-level data is becoming increasingly difficult to access even for researchers willing and able to conduct inference on scraped real-time data.

In my own analysis, I too found inconsistent deidentification strategies among operators. Four of the six vendors in my dataset implemented dynamic vehicle IDs in some form, but in that group of four, three very different methods were represented. The remaining two vendors did not appear to rotate IDs during the period I collected data. I discuss these findings further under "Data exploration" below.

Survival analysis for shared mobility

As discussed under "Analytical framework" below, idle time prediction for shared mobility is well suited to survival analysis, a set of statistical methods designed to model the time until an event occurs. Survival analysis is a general term for a collection of techniques, with Cox proportional hazards regression among the most common.

In a relevant application, Kostic et al. (2021) compared the performance of a Cox regression model to the DeepSurv survival neural network in predicting time-to-pickup for shared cars. The authors found that DeepSurv provided more accurate predictions than the Cox model, and that a "two-step" approach combining classification with survival analysis performed better than survival analysis alone. Lastly, they incorporated more advanced mathematical techniques to model correlation between vacant vehicles.

Data

The definitive source of public data on dockless micromobility is the patchwork of real-time feeds published by individual vehicle providers at the city or system level in the format set out by the General Bikeshare Feed Specification (GBFS), an open-source data standard for shared mobility in the spirit of public transit's General Transit Feed Specification (GTFS). The full GBFS standard comprises a collection of JSON file structures, several of which are specific to dock-based bikeshare

systems. Most relevant to micromobility is the `free_bike_status` feed, which contains real-time coordinates and select metadata for all currently inactive dockless vehicles.

Feed availability varies by jurisdiction, but my adopted hometown of Washington, D.C. is among a handful of cities that require shared mobility operators to maintain public API endpoints in GBFS format as a condition of their operating permits. As of May 2022, the District of Columbia is home to six dockless vehicle providers: the city's bikeshare system (Capital Bikeshare, operated by Lyft), which offers docking-optional electric bicycles alongside classic docked bikes, and five private companies (Bird, Helbiz, Lime, Lyft, and Spin).

To obtain longitudinal data on how long vehicles remained inactive between trips, I wrote a Python script to query each provider's `free_bike_status` API endpoint every 60 seconds. After an initial trial run using only Capital Bikeshare data, I conducted full-scale data collection for all six D.C. dockless vehicle providers between April 18, 2022 and May 1, 2022.

The common data standard ensured that several essential fields were populated for all providers: a vehicle identifier, the vehicle type (bicycle or scooter), latitude, and longitude. I also recorded the vehicle operator and timestamp. However, different vendor implementations and schema versions meant that fields to track reserved or disabled vehicles were populated in some feeds but not others, and only one provider, Bird, reported vehicles' battery level.

Most significantly, the treatment of unique identifiers varied substantially by vendor. Rotating the vehicle ID each time a trip ends became part of the GBFS standard in January 2020, after contributors to the project raised concerns on GitHub that the ability to reconstruct trip origins and destinations from scraped real-time data risked exposing the movements of domestic abuse survivors, minors accessing reproductive health services, and other vulnerable groups. However, as of May 2022, only one of the six operators whose feeds I scraped appeared to adhere to this standard, with implications I describe under “Data exploration” below.

Pipeline design and infrastructure

Designing the extract-transform-load (ETL) pipeline infrastructure required balancing competing considerations: reliable uptime to run the scraper continuously, without interruptions, every minute for a period of up to several weeks, but also the disk space and computing power to store, process, and model the data without incurring cloud charges beyond the scope of a student project. To meet both needs, I created a split cloud-to-local pipeline to ingest data in the cloud but transfer it to my local environment for processing and analysis.

For the cloud component, I created an Amazon Web Services (AWS) EC2 Micro instance running Ubuntu 20.04. Within it, I built a PostgreSQL 14 database to hold the incoming stream of real-time vehicle data along with error logs, if any. To maintain a single compute and storage environment, and to maximize storage space within AWS's free tier, I chose to build my own database within my EC2 instance rather than use AWS Relational Database Services (RDS) to host a

managed PostgreSQL database. I managed the cloud environment via a Secure Shell Protocol (SSH) connection from my desktop.

Still inside the EC2 instance, I ran the scraper as a stand-alone Python script scheduled to query each of the six vendors' `free_bike_status` API endpoint every minute at :00, parse the retrieved JSON data, and write the relevant fields to a database table. Error handling within the scraper script ensured that any failed scrapes would log the timestamp, API provider, and traceback details to a separate database table.

Because a single scrape contained nearly 10,000 unique records, amounting to around 1.4 million new table rows per day, I partitioned the primary `vehicles` table by timestamp range, with a new partition for each day scraped. Partitioning the table also facilitated transferring data between the cloud and local environments without interrupting the scraping process.

In my local desktop environment, also running Ubuntu 20.04, I built a second PostgreSQL 14 database with a schema mirroring that of the cloud database. In the local version, I also installed the PostGIS extension to allow spatial manipulation and querying directly in the database, which my research suggested would be more performant than Python's GeoPandas for spatial operations on tens of millions of records.

To transfer data between the two environments, I detached the oldest complete partition from the cloud database `vehicles` table, then used secure copy protocol (`scp`) to transfer the data from the EC2 instance to my desktop, where I attached it as a new partition to the local database's `vehicles` table.

Lastly, to conduct further analysis and modeling in Python, I used the Psycopg 3.0 Python package to connect to and query the database from within a Jupyter Notebook. For performance reasons, I kept data manipulation and aggregation within the SQL query to the extent possible, using both PostgreSQL and PostGIS functions, before moving to Python for exploratory analysis, visualization, and model development.

- add updated diagram

Data exploration

The dataset I assembled contains a wealth of potential insights into access to and demand for shared dockless micromobility in the District of Columbia. However, given this project's focus on building a cohesive longitudinal vehicle dataset from snapshot time series data, the treatment of unique vehicle identifiers became the centerpiece of my exploratory analysis. In future work, I want to build on these fundamental technical developments for dataset assembly to investigate the spatial distribution of shared dockless vehicles, including different operators' presence in the city's Dockless Equity Emphasis Areas and other measures of equity of access.

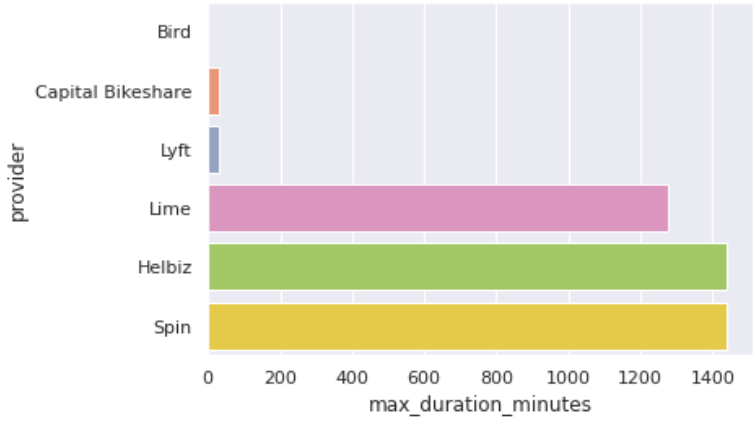
Aggregating vehicle data by unique ID

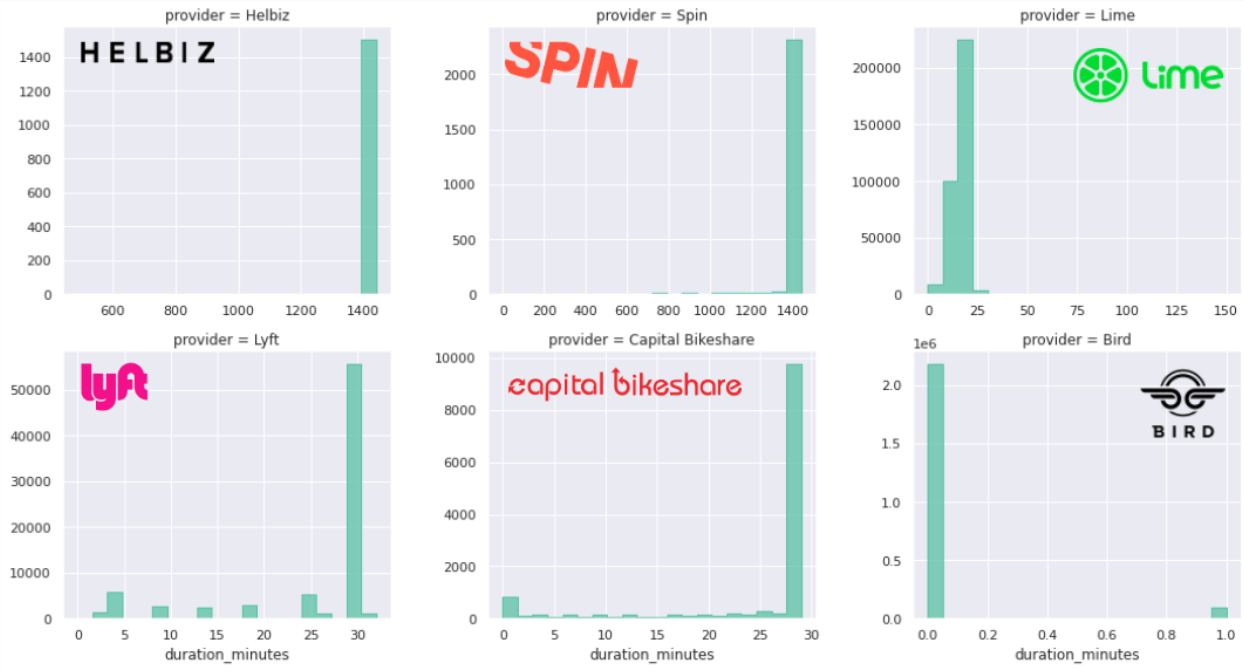
To examine the vehicle-level structure of each micromobility operator's public data, I relied on SQL aggregation queries to extract provider- and vehicle-level summary statistics from the raw time series data, which at that point numbered in hundreds of millions of rows. I computed the duration that each vehicle ID was present in the data from the difference between its earliest and latest recorded timestamp, and then used the operator-level maximum ID persistence to infer how frequently, if at all, each vendor reset its IDs.

In the data I collected, only one operator appeared to reset identifiers each time a vehicle was parked, consistent with the GBFS schema. Of the remaining five, two reset them every 30 minutes whether the vehicle was moved or not, but did *not* reset them at the conclusion of a trip; one assigned new IDs so frequently that the same identifier rarely appeared more than once; and two did not appear to rotate vehicle IDs at all.

As the table and charts below illustrate, identifier persistence varied dramatically among vendors. Over a sample 24-hour period, Bird rotated its IDs almost continuously, with none lasting longer than a minute; meanwhile, Helbiz and Spin did not reset their IDs once. Lyft, which operates both its own scooter fleet and the public-private Capital Bikeshare system, resets IDs roughly every 30 minutes. Lime's longest-lived ID appeared to be an outlier, and did not suggest a fixed randomization schedule. When I plotted the distribution of identifier lifespans by provider, as shown on the next page, Lime's distribution was the only one that seemed consistent with resetting vehicle IDs at the conclusion of each trip.

	provider	max_duration	max_duration_minutes
0	Bird	0 days 00:01:00	1
1	Capital Bikeshare	0 days 00:29:00	29
2	Lyft	0 days 00:30:00	30
3	Lime	0 days 21:16:00	1276
4	Helbiz	0 days 23:59:00	1439
5	Spin	0 days 23:59:00	1439





Vehicle identifier longevity by operator. Company logos for illustrative purposes only. This is an academic project undertaken solely for educational purposes, and I have no affiliation with the above-mentioned entities.

- mention both 'hard problems' and their conceptual solutions, but leave actual implementation for 'later work'
    - also note eugene's paper??
- some summary of lime specifically?

- Visualizations/exploratory
  - Map distribution of available vehicles by vendor
    - How to handle time?
    - Include map of priority/underserved areas?
      - Dockless Equity Emphasis Areas: https://opendata.dc.gov/datasets/DCGIS::dockless-equity-emphasis-areas/
  - Spatial autocorrelation?
  - Histogram of time to activation
  - Vehicles per capita by neighborhood?
    - By priority vs. non-priority area?
  - Correlate availability with income?
  - Correlate time to activation with availability/saturation?

Error handling and scraper interruptions

My scraper's error handler logged five scraping failures to the database between April 21, 2022 and May 1, 2022: three for Capital Bikeshare indicating an invalid JSON response, which might

happen if the API server returned an empty response or HTTP error code, and two for Spin indicating a broken connection to the API server. Each error was associated with a single missed scrape, representing one lost minute of data, and did not substantially affect the validity of the data collected. Longer outages could be handled at the modeling stage by treating the last vehicle records scraped before a connection loss as "censored," as described in the section on survival analysis below.

Survival modeling to predict time to activation

Much more than I had intended, this became a data engineering project with a brief note on predictive modeling, rather than a modeling project per se. As such, this section should be read as an early exploration of survival analysis as a modeling framework to predict shared dockless vehicle time to activation, and a first example of the type of insight my GBFS aggregation pipeline can yield.

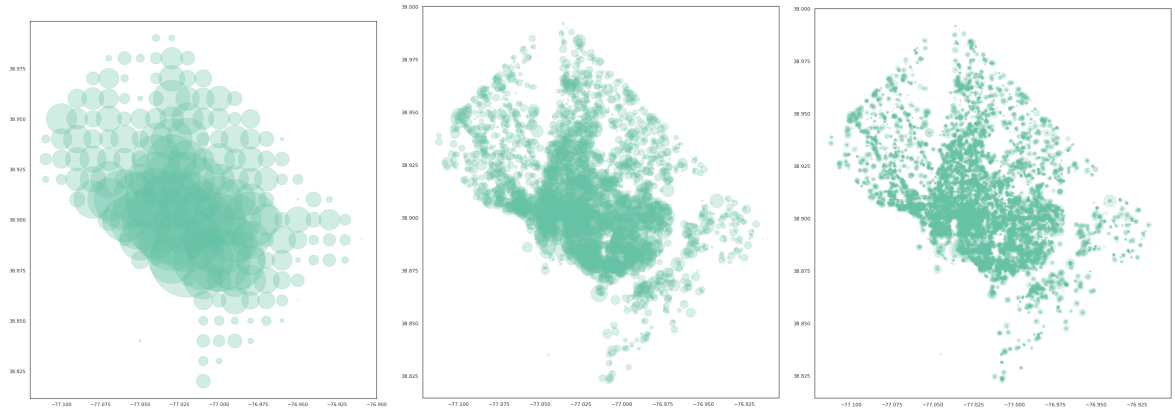
Analytical framework

Survival analysis originates from medical research modeling how long a patient or organism is expected to survive a disease process, but it can also be applied more generally to model the expected time until an event occurs. The methods of survival analysis allow for the inclusion of so-called censored data, which refers to observations whose status is known until a certain point, but unknown thereafter. In the shared mobility context, any vehicles still inactive at the end of data collection are right-censored, as are any vehicles whose IDs reset after a certain period and cannot be matched to another ID.

In my review, I found three broad approaches to survival modeling: the **statistical** approach, where the semi-parametric Cox proportional hazards regression is standard; the **classical machine learning** approach, where survival random forests seem most prominent at first glance, but which also include gradient boosted models and survival support vector machines; and the **deep learning** approach, which includes the Cox proportional hazards neural network DeepSurv (Katzman et al. 2016), among others.

Data cleaning and baseline feature engineering

I used the scikit-survival Python package to implement a Cox proportional hazards regression model using only the simplest spatial, temporal, and vehicle features: point **location** rounded to two decimal places, a spatial resolution of approximately 1.1 kilometers; **day** of the week; **hour** of the day; and **vehicle type**. I chose these features for expediency far more than efficacy, and improving spatial precision would be my first priority for the next iteration of this model.



Spatial distribution of Lime dockless vehicles, with coordinates rounded to two, three, and four decimal places, respectively.

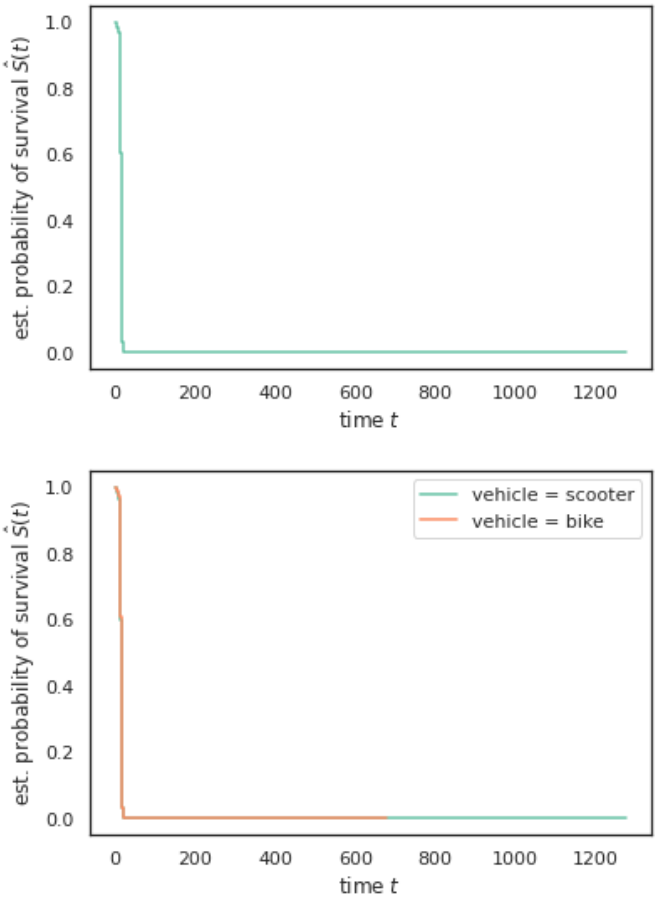
I limited the model to data from Lime, the only operator that appeared to meet the GBFS recommendation to assign a new ID each time a vehicle was parked, but maintain a consistent ID for the duration of inactivity. I coded vehicles as right-censored — that is, with an unknown activation time — if they were present at the last timestamp in the dataset, beyond which their status was unknown.

Lastly, I limited the model to vehicles that remained in a single location, reported to four decimal places (a precision of approximately 11 meters). I was unable to immediately determine why some Lime vehicle IDs appeared in more than one location, but speculated that rebalancing activity — where the company relocates inactive vehicles in response to demand — could cause a vehicle to change locations without triggering an ID reset in the way a trip would. If true, and if the likelihood of rebalancing is tied to the length of inactivity, this has the potential to introduce selection bias into the model by omitting vehicles with long idle periods parked in low-demand locations. In future work, this might be addressed by treating each vehicle position as a separate vehicle.

Results

Kaplan-Meier estimator

Central to survival analysis is the so-called survival function, which provides the probability that a subject – in this case, a bicycle or scooter – will "survive" past a certain time. A simple method to estimate a dataset's survival function is the Kaplan-Meier estimator, a non-parametric statistic that allows for the presence of right-censored data. I used scikit-survival to create Kaplan-Meier plots for the Lime dataset. As the plots show, survival drops precipitously after just a few minutes, and excluding outliers on the high end, the survival function does not appear to vary by vehicle type.



Kaplan-Meier estimator for Lime scooters and bicycles, combined (top) and separately (bottom).

However, because it is unable to account for any covariates that might influence survival other than time, the Kaplan-Meier estimator can only provide an extremely simplified estimate of the survival function.

Cox proportional hazards regression

To assess the performance of the Cox regression model, I used scikit-survival to compute Harrell's concordance index, or c-index, a measure of predictive accuracy that is the survival analysis equivalent of binary classification's area under the receiver operating characteristic (ROC) curve (Harrell et al. 1996). Mirroring the interpretation of the area under the curve (AUC), a c-index value of 1 indicates a perfectly accurate model, a value of 0.5 indicates a perfectly random model, and a value of 0 indicates a perfectly inaccurate model.

The concordance index for my baseline model was **0.63**: solidly better than a coin flip, but with clear room for improvement. The middling performance was unsurprising: the model was limited to four categorical features, of which one – vehicle type – appeared to make no difference. The single spatial feature was too coarse to adequately capture differences in built environment or transportation infrastructure across a varied urban landscape, and the model also did not consider

weather, which we might expect to influence how likely someone is to choose a mode of transport fully exposed to the elements.

To improve predictive performance, we might incorporate not just weather data – like precipitation, temperature, and wind speed – but a range of other spatial and accessibility-related features. These might include distance to an intersection or major road; distance to rail and bus stations, if possible incorporating some measure of service frequency; distance to traditional docked bikeshare; points of interest (e.g. restaurants, bars, tourist attractions) drawn from OpenStreetMap; Census population density data; some measure of employment density; and the availability of other dockless vehicles nearby.

- Map of actual vs predicted?
  - Scatterplot of actual vs predicted
- Map of model error
- Where did the model perform well/poorly?

Discussion

- limitations
- future work
- conclusions

References

- export from Zotero