Technical report for project "Zero-length rides in Helsinki bikeshare''

Miika Piiparinen, Hamid Aebadi, Maarja Mustimets

2024-10-30

This report details the chronological, technical and practical elements of our group project for the course "Introduction to Data Science" and what ended up becoming the "Biketainer" application.

Project Chronology

The whole project processing happened in 3 main phases.

Brainstorming Phase (2 weeks): Thanks to the intiative taken by Maarja, we reached a relatively swift consensus of working with open data for Helsinki city bikes. We did juggle the idea of doing something with medical data, but deemed the data there to require too much subject matter expertise to work with. We were originally going to attempt working with equivalent data from Estonia, but chose against it, as the Helsinki bicycle data was more abundant. The `data/data_info.md` defines the data we ended up using, but it also defined traffic data which we thought could provide for interesting insights. However, this data was scrapped mostly due to time constraints.

Miika started building the backbone of the application at this point, implementing a trivial flask api and some frontend components, including tapping into the Google Maps API to render the map of Helsinki. Some time was spent between Miika and Maarja for skills exchange when it comes to software development conventions in Python, as the latter was not yet experienced in the topic. Later, Maarja would migrate to mostly do analytics in the data directory with R.

Technical Fine-tuning Phase (3 weeks): In this phase of the project we knew we were locked-in to the bicycle data. Now we needed to defined our technical solution in a more concrete way. We did not have a very specific problem to solve with the data, so we started out with a plan to do some overview of the rides via graph optimization, but later we arrived at a more technically sound and interesting proposition. The main limitation in the graph optimization solution was the lack of data and clarity, which was not the case in the actual product we decided to implement.

More specifically, the general prospect initially was optimizing the network of bikes and defining the edges of the network, both in terms of physical distance and flow. We wanted to use a Ford-Fulkerson-esque algorithmic solution initially, seeking to chart out and optimize flows in the graph, but the precise form of this was unclear for a lengthy amount of time. In the end, such a solution was scrapped in favour of a predictive time-series approach.

The team drifted into a time-series based approach in a discussion forum around the middle of the project. Maarja had done thorough EDA and found that events the team would later come to call "zero-length rides" occur, where a person seemingly takes a bicycle out and returns it back in a very short timeframe. She had considered that these events could be related to – at least partially – broken bicycles, especially if they occur in quick succession. Miika had been considering time-series based solutions for his personal- and work-related projects, and wanted to implement a predictive solution as well as a Timescale database instance that'd be tied to the project.

Thus, the final formulation of the project revolved around predicting zero-length ride events, and hence giving the customer insights on where potential breakages may occur in the future. During this time, Miika was finishing up the node visualization portion in the frontend, and had started setting up the timescale database. Hamid had been given the responsibility to implement the machine learning solution known as Prophet

by Miika, and he formulated the idea of using multiple models for prediction. However, his prophet-based solution was scrapped due to testing done by Miika, where the numbers produced by Prophet were nonsensical compared to the alternative ARIMA-solution.

Polishing Phase (2 weeks): The final weeks of the project were mostly spent on the machine learning solution, as well as tying the moving parts together in a frenzy. Additionally, the presentation and this report were prepared.

Technical Solutions

Data

The project uses city bikes' usage data and city bikes' station data collected by HSL and shared on the Helsinki Region Infoshare website. The data for rides is currently available for years 2016-2023. Below is a table of the number of rides recorded each year, to give an idea of the size of the data.

Year	2016	2017	2018	2019	2020	2021	2022	2023
Number of rides	408032	1607124	3228039	3787930	3129330	2862090	2474757	2544008

The data had to be cleaned and transformed into a format usable by the machine learning solution. All such processing was done in the code `data/data_transformation.R`. The data on rides was mostly sound, only needing some imputation. However, as the data on city bike stations was, strictly speaking, only available for the years 2018-2021, adding coordinates to a ride's stations required some creativity. Namely, first all possible rides got start and end coordinates according to station IDs. Next, missing coordinates were supplemented by station name. Finally, some stations still missing coordinates now had namesakes with coordinates. These connections only happened at the end, because they do not share an ID nor a name with anything in the HSL station data, but one of their namesakes has an entry in the data by ID (and not name!). Convoluted, I know.

After processing, the data was handed over for machine learning solutions. Literally handed over in a USB stick, because no online solution could handle the volumes of data being transmitted.

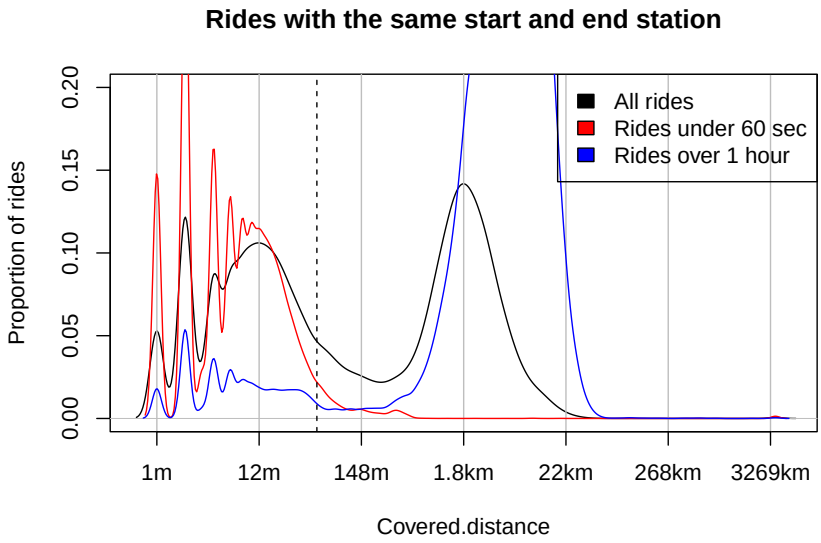
Exploratory data analysis was done mostly for internal purposes, to get an overview of the data. This exploratory data analysis is available in the code `data/eda.R`. However, some of the more interesting results relating to the final solution are available with the application, as well as in the file `data/info_sheet.html`.

The "Problem" task

Faulty bikes It is known that many rides may be initiated only to be ended almost instantly due to the rider initiating the ride and only then discovering some technical issue that makes riding the bike to their destination impossible. Because of this, the ride remains short in time and duration and the ride's Departure and Return station end up being the same.

Such rides should be observable in ride data. If we interpret very short rides that start and end at the same location as possibly indicating a broken bike, then this data can be useful in deploying bike maintenance personnel. Even though a single zero-length ride does not necessarily mean a broken ride, repeated zero-length rides or a great proportion of them should indicate an faulty bike and even an insufficient number of bikes (because if only faulty bikes remain at a popular station, they will be repeatedly undocked and repeatedly trigger a zero-length ride).

Zero-length rides The first task was to identify what such a ride - a "zero-length ride" - looks like in data. 6.5% of rides have the same Departure and Return station. Next, suitable cutoff points for a zero-length ride's distance and duration needed to be established.



The above plot uses data from rides with the same start and end station and visualises the distribution of their lengths on a logarithmic scale (variable Covered.distance, in meters). As can be seen, the rides can be grouped in two by distance: rides under ~150 meter and longer rides. Additionally, rides under a minute are very short in distance (as can be expected) and rides over and hour are mostly long in distance. The remaining long-lasting rides could be unsuccesful parkings, where the (faulty) bike was essentially immediately placed back at the starting station. The dashed line is set at 50 meters. We chose this as a cutoff because we wanted to target broken bikes, and a bike that has gone at least 50 meters seems to work to some extent - this is a somewhat conservative assumption, and maybe the cutoff should have been at 150 meters, which splits the rides into two groups especially well.

We decided to define a "zero-length ride" as a ride that has the same start and end station and

a) is under 60 seconds long in Duration, or
b) is under 50 meters long in Covered.distance.

Next, the info on zero-length rides would be employed in an application that predicts the frequent appearance of zero-length rides in stations, indicating a faulty bike.

Application software

- The backend of the project follows a very basic object-oriented paradigm. An abstract wrapper class for machine learning solutions was implemented, which means to enhance development velocity of the software in the future by allowing multiple machine learning models to interface with the product in a homogenous fashion. A restful flask api is used for the backend, from which the frontend fetches JSON packets using GET-requests. The packets were not defined very rigorously, f.e. using an open-api spec or equivalent. They mostly contain coordinate data for existing or predicted nodes. The application interfaces with an Analytics class, which talks to both the machine learning models and the timescale database.
- The timescale database is interfaced via a TimescaleClient class. It's a postgres-native solution where timestamps are converted to an index of a hypertable that allows for swift queries and aggregation of mass-amounts of timeseries data. The database runs on a simple docker container. To note, the initial insert of data into the database is parallelized for speed. The analytics class also utilizes some parallelism where applicable. The queries ended up being very fast, with millions of rows of time-series data being fetched and aggregated in seconds.
- The frontend of the project is a simple React-based application that uses the Google Maps API to

render geographical data. Due to a lack of a strong frontend engineer in the team, the GUI portion is very much an MVP, adhering to no strong design principles worth talking about.

- The machine learning solution Prophet was initially going to be used, but it gave bizarre predicitions as it's not very useful in predicting discrete values on such a small range (really the realistic prediction-space is [1..10]). ARIMA worked much better for this, and via rounding out the numbers, the solution could land some degree of accuracy in it's predictions. Sadly, the fine-tuning of the solution must be left for future development, if applicable.

Improvements

We faced many challenges that are typical in group projects, as detailed in the chronology. There were some workload imbalances and differences in experience between team members, as well differences in time put in to the project. However, at the end we arrived at a very interesting use of the data and a working application.

In terms of going further with the task and theme, there could be further insight into the patterns of zero-length rides occurring, especially in terms of time: weekdays vs weekends, evenings vs other times of day etc.

This "zero-length ride" problem could definitely solved better given access to all the city bike data that HSL presumably has access to. For example, data about stations in terms of number of bikes currntly parked or even just bike IDs in the ride data. However, limited to working with this public data, this is probably a working approximation.