Chapter 4: Data & Experimental Design

Draft version

This chapter describes the case study that was done to test the performance of DBAFS. It is structured as follows. First, the characteristics of the bike sharing system that served as a data source for the experiment, are presented. The data retrieval process is described in section two. Finally, the third section explains into detail the methodology of the experiment itself.

4.1 Data source

DBAFS' forecasting power was evaluated with data from the dockless PBSS of San Francisco, California. The system there is exploited by JUMP Bikes (https://jump.com/), an American company founded in 2010, that went through a rapid development after being acquired by ride hailing giant Uber in April 2018 (Khosrowshahi 2018). In February 2019, JUMP was active in 16 cities in the United States, as well as in Berlin, Germany. All provided bicycles have electric pedal assistance, up to a maximum speed of 32 km per hour.

In January 2018, the San Francisco Municipal Transportation Agency (SFMTA) offered JUMP Bikes the city's first offical, exclusive permit to operate a dockless PBSS, for an 18 month trial period, in order to "evaluate, collect data, and assess whether further increases would serve the public interest" (Jose 2018a). SFMTA allowed up to 250 bikes in a system area of approximately 47 km^2, which is shown in Figure 4.1. After nine months, a mid-point evaluation lead to the allowance of expanding the fleet to 500 bikes (Jose 2018b).

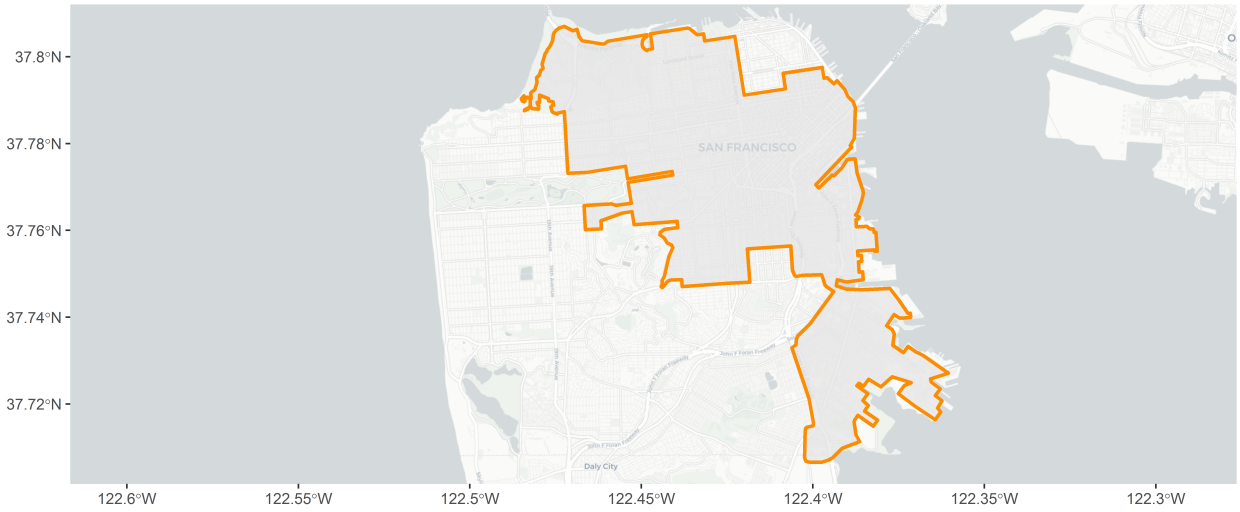

Figure 1: System area of JUMP Bikes in San Francisco

The JUMP Bikes system in San Francisco, works as follows. A user needs to download the JUMP mobile application and create an account. The app shows the real-time locations of all available bikes. An available

bike can be unlocked with a unique code that is send to the user account. A single trip costs $2 for a total time of 30 minutes. For every minute that exceeds the 30 minute timeframe, $0.07 will be charged. Options for subscription pricing, with a fixed amount per month or year, are not available. It is possible to reserve a bike up to 30 minutes before unlocking it. In that case, the bike will not be labeled as available anymore, and can not be taken by another user. However, the trip clock starts ticking from the moment of reservation. When a user wants to stop somewhere during a ride, the bike can be put 'on-hold' for maximum an hour, meaning that the bike stays inavailable during the stop. Also in this case, the regular fee will be charged. When ending the ride, the bike needs to be locked legally to a fixed object. Not doing so, will result in a fee of $25. Additionally, leaving the bike outside of the system area, results in a $25 fee as well (Harris 2018).

Since all JUMP bikes are electric, battery life forms an important issue. A fully charged bike can travel 48 to 64 kilometers with the pedal assist. Bikes with a batterly level of less than 25% are collected, and charged. Furthermore, other bikes are regularly charged during nightime. Charging takes in most cases about four to six hours, and is done in a JUMP Bikes depot. Some of the charged bikes are placed outside of the depot, where they can be picked up, while others are redistributed over the system area. In the near future, this process will change considerably, since the newly produced bikes, which will be introduced in the first months of this year, have swappable batteries, such that there is no need anymore to bring low-battery bikes to the depot (Foley 2018).

In the first year of the trial period, 63000 users took over 625000 trips. On average, trips were 4.2 kilometers long, while around 1% of the trips, exceeded 24 kilometers. Each individual bike, was on average used seven times per day. In september 2018, this was over eight times per day. Even when the size of the fleet doubled in October 2018, the utilization remained consistent until the first half of November, when it started to decrease slightly, due to cold weather (Rzepecki 2019). While the demand is high, the supply is still restricted by SFTMA, putting extra emphasis on the need for efficient rebalancing, and accurate bike availability forecasts.

4.2 Data retrieval

JUMP Bikes provided access to a database containing the geographical locations of their available bikes in San Francisco. The database fulfilled all DBAFS requirements described in section 3.4. Data collection started at September 9th, 2018, 15:41:08, Pacific Daylight Saving Time (PDT). The data had a temporal resolution of one minute, meaning that every minute, the location of each available bike in the system was recorded. Timestamps were stored with six-digit precision. Because of that, the time of recording was not exactly the same for each available bike, but could vary up to a few seconds. Therefore, before using the data in DBAFS, all timestamps were truncated to minute precision.

4.2.1 Distance data

When calculating distances, only the historical data at every quarter of an hour were used in the experiment. Hence, i_s was set equal to fifteen minutes. There were several reasons for this choice. Firstly, it is not expected that the data change drastically from minute to minute. That is, using data with a temporal resolution of one minute will probably not contain a lot more information than using data with a temporal resolution of fifteen minutes. Consequently, if a forecast for a specific timestamp is in practice a forecast for a few minutes earlier, this will not be problematic. On the other hand, using only data every fifteen minutes will decrease the size of the data with a factor fifteen, and speed up computations considerably. Furthermore, a lower order ARIMA(p, d, q) model can be used to capture the same patterns in the data. This is important, since lower order models will result in lower errors arising from parameter estimation (Brockwell and Davis 2002). After defining i_s, distance data pre-processing steps on the JUMP Bikes database server were taken as described in section 3.4.1.

4.2.2 Usage data

Based on the specific knowledge of the JUMP Bikes system presented in section 4.1, an extra restriction was added to the in section 3.4.2 described process of retrieving pick-ups from the original data. If a feature that was initially considered to be a pick-up, was not followed by a drop-off within a two hour timeframe, the feature was removed from the usage data. Taking into account the average trip length of only 4.6 kilometers, the fact that trips become more expensive after half an hour, the maximum allowed reservation time of 30 minutes, and the maximum allowed 'on-hold'-time of one hour, the threshold of two hours was considered to be a safe border. In this way, pick-ups that occur because the system operator is collecting bikes to be charged (i.e. pick-ups that do not reflect the usage intensity of the system), were taken out. Of course, there may have been some real trips that were longer than two hours. However, given that during the first year of the trial period only 1% of the trips exceeded a distance of 24 kilometers, and the electric pedal assitance allows a speed of more than 30 kilometers per hours, this was assumed to be a negligible share of the total number of trips.

4.3 Experimental design

4.3.1 Training and test periods

As mentioned in section 4.1, usage of the JUMP Bikes system in San Francisco remained rather constant in September, October, and the start of November. An exploratory analysis on the distance data at several locations in the system area enabled to draw similar conclusions, with a sudden change in temporal patterns after mid-November. Recall that in DBAFS, parameter n_w, the number of weeks between two passes through the model loop, should be chosen such that an 'old' model is not used anymore when the patterns in the historical data have changed considerably. Therefore, to test the performance of DBAFS in an adequate way, both the period used for model building, called the *training period*, and the period used for forecasting, called the *test period*, should preferably fall within a timeframe where no large changes in the data patterns occur, but without overlapping each other.

The training period, used to query data for both the cluster and model loop, spanned the first four full weeks (i.e. 2688 observations) in the data collection period, from Monday September 17th, 2018, 00:00:00 PDT, up to and including Sunday 13th, 2018, 23:45:00 PDT, as shown in Figure 4.2. Hence, a situation was simulated in which the weeks of data used for clustering m_c, and the weeks of data used for model building, m_m were both set to be four weeks. Taking into account the size and shape of the system area, along with some exploratory research on the demographic characteristics of San Francisco, K, the set of integers containing all values considered as the number of desired clusters k, was chosen to be $\{3, 4, 5, ..., 10\}$.

The obtained model structures and parameters resulting from the model loop, were used to make several forecasts during a test period of one week. For each forecast, two weeks of distance data were retrieved from the database, such that in the case of a weekly seasonality, a sufficient amount of data would be present for decomposition. Hence, $m_f = 2$. To make sure that in no case the data used for forecasting would overlap with the data used for model building, a two week period seperated the training and test period. That is, the test week ranged from Monday October 29th, 00:00:00 PDT, up to and including Sunday November 4th, 2018, 23:45:00 Pacific Standard Time (PST), as shown in Figure 4.2. Defining where and when to forecasts, was done by simulating real-world forecast requests, and sending them to the forecast loop. To do this in a realistic way, the following requirements had to be fulfilled.

- More forecast requests should occur at locations where the usage intensity of the system is higher.
- More forecast requests should occur at times when the usage intensity of the system is higher.
- It should be accounted for, that the times when the usage intensity is higher, can vary per location, and vice versa.

Bearing these requirements in mind, the following approach was developed. All pick-ups during the test week were retrieved from the database. Ten pick-ups per cluster where randomly sampled, guaranteeing

that each cluster was represented in the sample. Subsequently, $500 - 10 \times k$ pick-ups were randomly sampled from the remaining ones, regardless to which cluster they belonged. This lead to a dataset of 500 pick-ups in total, from which the location-timestamp combinations were retrieved. Pick-ups reflect the usage of the bike sharing system. That is, a random sample of them will contain more locations in areas where the usage intensity is high, and more timestamps at times when the usage intensity is high. Furthermore, the location and timestamp come as a combination, rather than as separate entities. In this way, the approach fulfills all three requirements mentioned above. The location-timestamp combinations in the sample will from now on be referred to as the *test points*. Starting from the timestamp of the test point, all time lags up to one day ahead, i.e. 96 time lags in total, were forecasted. Having 500 test points, in total, 48000 forecasts were made. To evaluate their performance, historical distance data for all forecasted time lags were retrieved from the database, and the forecast RMSE (see section 2.4.2.6) was calculated for each test point seperately.

By using the approach described above, the reported overall forecast errors will be dominated by those made during peak hours, and in crowded areas, when and where obtaining good forecasts is generally harder. However, this is intended, because reporting a large amount of forecast errors made during off-peak hours, and in non-crowded areas, or, alternatively, using adjusted error metrics such as the RMSLE, may give results that look nicer, but do not reflect the real usefulness of the forecasting system.

To compare the performance of DBAFS in a relative manner, all test points were also forecasted 96 time lags ahead with a simple baseline method. The naïve method, as described in section 2.4.3, was chosen for this task. Therefore, the baseline forecasting system will from now on be referred to as the Naïve Forecasting System (NFS). For each test point, the RMSE's of the forecasts of NFS were calculated, and compared to those of DBAFS.

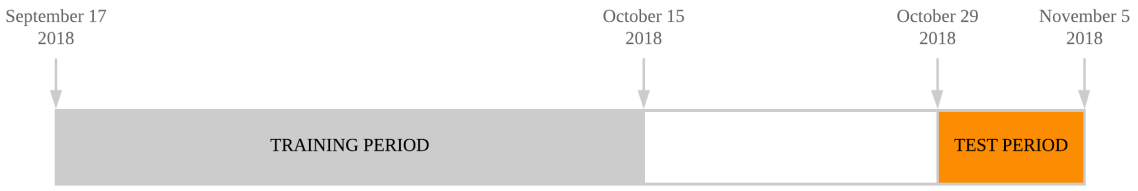

Figure 2: Training and test period

4.3.3 Benchmark

In addition to the forecast errors, the average computation times of DBAFS and NFS forecasts were compared. For a DBAFS forecast of a single test point, two weeks of distance data need to be retrieved from the database, while for a NFS forecast of a single test point, only the last observation of the historical distance data is needed. The computation time of the data retrieval, averaged over all test points, and the computation time of the forecast itself, again averaged over all test points, were added together to get an average computation time for the complete process of forecasting a test point. The benchmark was done on an Asus F541U laptop with Windows 10 OS, an Intel Core i7 processor, 8 GB RAM and 256 GB SSD. When DBAFS is implemented in a real-world situation, computations will probably be done of a large server, wich can speed them up considerably. Therefore, the benchmark should mainly be interpreted in a relative way, rather than in an absolute one.

4.3.4 Additional software

On top of those mentioned in section 3.2, some additional R packages were used for reporting the results of the experiment, as listed below.

- The `tsibblestats` package (Hyndman, O'Hara-Wild, and Wang 2018) was used to estimate the ACF (see section 2.2.1) of time series.
- The `tsfeatures` package (Hyndman et al. 2019) was used to calculate the spectral entropy (see section 2.2.3) of time series.
- Data and results were visualized graphically with the `ggplot2` package (Wickham 2016). Additionally, the packages `tidyr` (Wickham and Henry 2018) and `tibble` (Müller and Wickham 2019) were used to transform some data into formats that are compatible with `ggplot2`.
- Maps were created with the `ggspatial` package (Dunnington 2018). Besides, with the `rosm` package (Dunnington 2017), all basemap tiles were retrieved from CARTO (CARTO 2018).
- For maps with a continuous color scheme, the `orange_material` color scheme from the `ggsci` package (Xiao 2018) was used.

References

Brockwell, Peter J., and Richard A. Davis. 2002. *An Introduction to Time Series and Forecasting*. Vol. 39. doi:10.1007/978-1-4757-2526-1.

CARTO. 2018. "Attribution: Powered by CARTO." https://carto.com/attribution/.

Dunnington, Dewey. 2017. *rosm: Plot Raster Map Tiles from Open Street Map and Other Sources*. https://cran.r-project.org/package=rosm.

———. 2018. *ggspatial: Spatial Data Framework for ggplot2*. https://cran.r-project.org/package=ggspatial.

Foley, Nick. 2018. "Getting more people on JUMP bikes with our new design." https://www.uber.com/newsroom/newbike/.

Harris, Dylan. 2018. "The Complete San Francisco Bikeshare Review Guide." https://biketoeverything.com/2018/06/12/complete-san-francisco-bikeshare-review/.

Hyndman, Rob J., Yanfei Kang, Thiyanga Talagala, Earo Wang, and Yangzhuoran Yang. 2019. *tsfeatures: Time Series Feature Extraction*. https://pkg.robjhyndman.com/tsfeatures/.

Hyndman, Rob J., Mitchell O'Hara-Wild, and Earo Wang. 2018. *tsibblestats: Analysis of tidy time series*. https://github.com/tidyverts/tsibblestats.

Jose, Ben. 2018a. "SFMTA Creates Pilot to Study Electric, Stationless Bike Sharing." https://www.sfmta.com/blog/sfmta-creates-pilot-study-electric-stationless-bike-sharing.

———. 2018b. "San Francisco's Stationless Bikeshare Pilot Reaches Mid-Point Milestone." https://www.sfmta.com/blog/san-franciscos-stationless-bikeshare-pilot-reaches-mid-point-milestone.

Khosrowshahi, Dara. 2018. "Welcome, JUMP!" https://www.uber.com/newsroom/welcomejump/.

Müller, Kirill, and Hadley Wickham. 2019. *tibble: Simple Data Frames*. https://cran.r-project.org/package=tibble.

Rzepecki, Ryan. 2019. "Celebrating One Year in San Francisco." https://medium.com/@jumpbikes/celebrating-one-year-in-san-francisco-28469d5dccaa.

Wickham, Hadley. 2016. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. http://ggplot2.org.

Wickham, Hadley, and Lionel Henry. 2018. *tidyr: Easily Tidy Data with 'spread()' and 'gather()' Functions*. https://cran.r-project.org/package=tidyr.

Xiao, Nan. 2018. *ggsci: Scientific Journal and Sci-Fi Themed Color Palettes for 'ggplot2'*. https://cran.r-project.org/package=ggsci.