Technical Report

Introduction to Data Science

Quy Anh Nguyen, Aleksi Pikka, Otto Uusipaikka

October 28, 2024

1 Introduction and Purpose

During the first lecture, it was emphasized that the mini project should address some interesting problem and provide a solution that would be of interest to some party [1]. We chose the demand for city bikes in Helsinki and Espoo as the topic of our mini project. The party that could make use of our solution is the Helsinki Regional Transport Authority.

We chose this topic because of the added value that addressing the problem would bring. The added value is as follows: allocating city bikes is a difficult operations research problem, and having better demand estimation would help us optimize the allocation and improve the quality of the city bike system.

In Helsinki and Espoo, residents can buy a city bike pass for EUR 35. Each bike pass lasts for one season, i.e., from April to October of one year. With a pass, people can make an unlimited number of trips by picking up and returning a city bike at any station in the two cities. The following variables are recorded for every trip made: time of departure, time of return, station of departure, station of return, covered distance (in meters), and duration (in seconds).

The data from 2016 to 2023 is made publicly available on HSL's website [2]. The whole dataset included 55 files from HSL, amounting to 1.6 gigabytes in total. Summary statistics, as visualized in Fig. 1, show that 20,080,360 trips in total were made between 2016 and 2023. Our dataset contains 509 distinct stations, although some stations were only opened at some point after 2016.

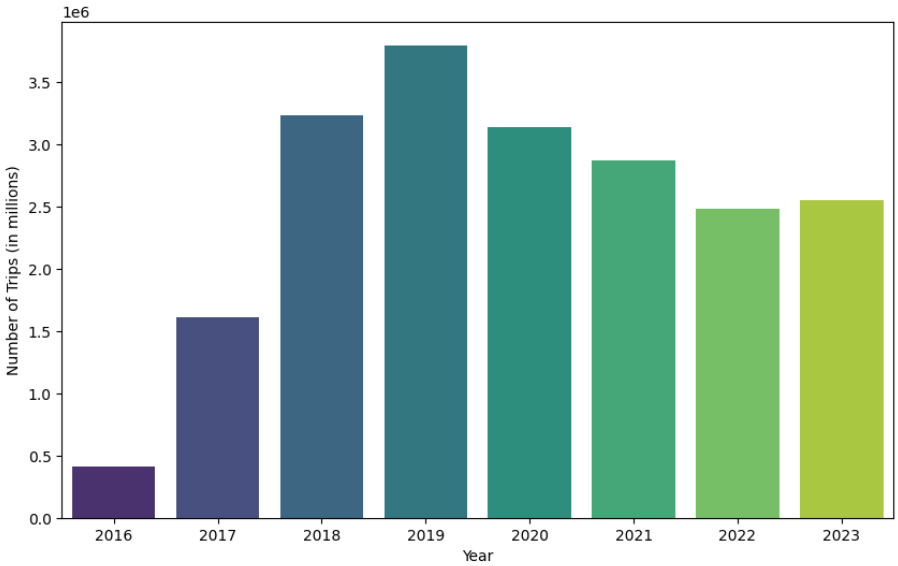

Figure 1: Distribution of Bike Trips from 2016 to 2023

We also acquired weather data from Open-Meteo [3]. We had originally planned to use the hourly temperature, rain amount, and wind speed as exogenous variables for our time-series model. However, given our limited computing resources and the already large volume of the bike data, including

exogenous variables in the model made the pipeline too slow. Thus, we omitted the weather data from our final model.

Our planned target group is the decision makers at HSL. For this target group, we believe that a Progressive Web Application (PWA) is the most suitable deliverable, as we can display all the data and predictions there in a concise and easily understandable format. The app can be viewed on any web browser; it can also be installed as a desktop or mobile app at the click of a button.

LLM Disclosure: We did use ChatGPT to generate code occasionally throughout the project. Our code was also sometimes checked and optimized by ChatGPT. However, the generated code did not turn out to be useful and was eventually omitted.

2 Data Preprocessing

We started data preprocessing by loading all files in the dataset into one Pandas DataFrame. Then, we aggregated the DataFrame by the departure station to get the number of ongoing bike trips originating from each station at each hour. We also aggregated the DataFrame by the return station to get the number of ongoing bike trips arriving at each station at each hour. These aggregations prepared the data for us to train two models, one for predicting the number of ongoing bike trips that have originated from a specific station at a specific hour and another for predicting the number of ongoing bike trips that are arriving at a specific station at a specific hour.

We also created indicator variables for each day of the week and each hour of the day. We attached the corresponding weather data to these indicator variables and used them as exogenous variables in one model. We found that mean absolute scaled error for this model was smallest when the non-seasonal parameters were (1, 1, 0) and the seasonal parameters were (0, 0, 0, 0). However, this model was not integrated into the final deployed application, as it was too computationally expensive.

3 Modeling

From the very beginning, we chose to use time-series methods, even though other tools could also have been used. For instance, regression modeling using panel data tools could have been done. We could also have built a structural model (i.e., using economic theory to derive the estimated equations), as was done by Kabra, Belavina and Girotra (2019) [4].

Ultimately, we believe that structural models are beyond the scope of this course, due to their complexity. Moreover, as each station is observed for considerably long periods of time, autoregressive time-series methods appeared more suitable than other potential solutions, such as fixed effects models. [1]

The two most basic time-series methods are probably the moving-average process and the autoregressive process. The MA(N) process explains a time-series using a weighted sum of N previous stochastic shocks and the current shock. The AR(N) process explains the time-series using a weighted sum of N previous realizations and a random shock. The ARMA model is a combination of both the AR process and the MA process. The ARMA model can be generalized to a non-stationary series, in which case it is called ARIMA [5].

Originally, we had planned to use the classic ARIMA model. To tune the parameters, we plotted the autocorrelation and tried out partial autocorrelation functions, which can oftentimes be used to determine the suitable parameter values or, at the very least, limit the parameter space. However, in this particular case, (partial) autocorrelation functions did not turn out to be helpful, so the tuned parameters didn't include their results.

We also learned that the bike data showed considerable seasonal patterns. An extreme example of this seasonal pattern is the winter, when there are no bike trips. As such, we ultimately chose the SARIMA model, which allows for incorporating seasonal trends using the seasonal autoregressive, integration, and moving average parameters.

[1] This terminology comes from econometrics.

To select appropriate parameters, we ran two types of cross validation. For simplicity's sake, we used 2023's data as test data and all preceding years' data as training data. First, non-seasonal parameters for the model with no exogenous variables were chosen by minimizing the mean squared error. Then, we used logical reasoning to deduce the seasonal parameters. Since our data is aggregated by hour, the periodicity parameter should be set to 24. Afterwards, the integration order was set to 1, while both the autoregressive and moving-average parameters were set to 0. Thus, the non-seasonal parameters were set to (1, 0, 2) and the seasonal parameters were set to to (0, 1, 0, 24), although in production we used (1, 1, 1) and (0, 1, 0, 24) instead due to memory constraint.

For the model with exogenous variables (hourly temperature, rain amount, wind speed, day of the week, and hour of the day), non-seasonal parameters were chosen by minimizing the mean absolute scaled error. Exploring different configurations with the seasonal parameters showed that including them reduced accuracy, so they were set to zeroes. Another reason for setting them to zero was that including them made the model much more computationally expensive. As for the non-seasonal parameters, they were set to (1, 1, 0).

Ultimately, what matters is the usefulness of our model. The mean absolute scaled errors were not unacceptably large (they can be found in the backend/error_metric_calculations_results.txt file), but the accuracy was still imperfect. Specifically, the SARIMA model was not very good at predicting peak hours. This is nevertheless expected, since the peak hours could have been determined by external events like music concerts, which our model couldn't account for as we lacked the relevant data.

There are a few improvements that could have been made. For instance, we could have explored more advanced machine learning methods to improve the predictions' accuracy. We could also have collected data about major events during the 2016-2023 period to improve our model's accuracy. Including exogenous regressors into the model would also have improved accuracy. However, doing so is computationally expensive, so exogenous regressors were omitted from the final web application.

4 Deployment

After training the appropriate statistical models, our next task was to deploy the system into production. The architecture of our system is shown in Fig. 2.

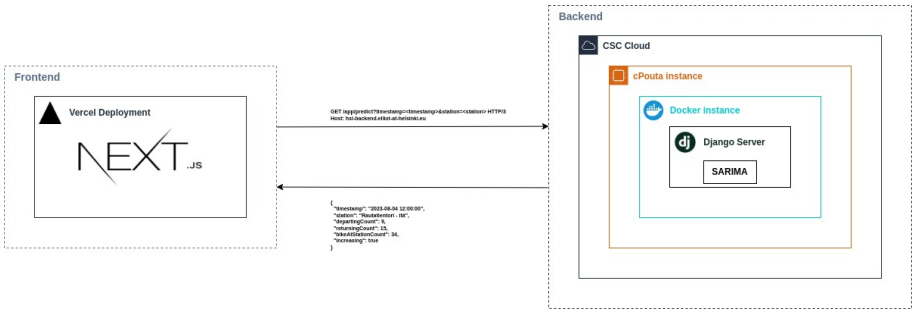

Figure 2: System Architecture

Our system follows the classic client-server architecture. Our frontend is a Next.js web application, while our backend is a Django REST server. The backend trains the prediction models upon initialization and loads the models into memory. Afterwards, it can use those trained models to make predictions and respond to REST API calls from the frontend. The frontend sends GET requests to the backend with the following parameters: timestamp and station name. The backend responds with a JSON object containing the predicted variables for the corresponding station at the specified timestamp.

In the backend, we use the models to predict the number of bikes arriving/leaving the station at the given timestamp. We assume the default number of bikes to be 28, as this is the size of a bike rack at a real station. Then, the predicted number of bikes is calculated as 28 – number of leaving bikes + number of arriving bikes. Naturally, if more bikes are arriving than leaving, the number

of bikes is increasing, and vice versa. If the predicted number of bikes is less than 5, the station is in urgent need of more bikes. If the predicted number of bikes is less than 10, the station is in need of more bikes, but not urgently.

To save costs on infrastructure, we deployed our frontend to Vercel, which offers free hosting for small web apps. For our backend, we deployed our Django server as a running Docker container. The IT Center for Science (CSC) offers a free Virtual Private Server service called cPouta for students in Finnish universities, so we rented a cPouta server and ran our Docker container there. We used Dokku to configure various settings related to the container like domain name, port forwarding, Let's Encrypt TLS certificate, and so on.

To test out our web app, visit `https://hsl-frontend.elliot-at-helsinki.eu/`. The web app is a PWA, so it can optionally be installed as a mobile or desktop app. Once in the app, select a station from the dropdown menu and enter a timestamp in YYYY-MM-DD HH:MM:SS format to get the prediction. Note that MM and SS should both be 00, since our prediction model is hourly specific. Additionally, since our free cPouta server has limited memory, we are only able to store enough prediction models in memory for two stations. The backend can be tested using API tools like Postman, Insomnia, or Hoppscotch at `https://hsl-backend.elliot-at-helsinki.eu/app/predict`.

5 Repository

The code for our project can be found at the following link:

`https://github.com/ElliotAtHelsinki/data-science-project`

The project structure is as follows:

- backend/: The Django backend
- frontend/: The Next.js frontend
- backend/main.ipynb: Code for preprocessing and aggregating the data, in addition to building the prediction models
- backend/cross_validation_parameters.ipynb: Cross validation code for the non-seasonal parameters
- backend/forecast_error_metric_calculations.ipynb: Code for calculating the error metrics of the omitted SARIMAX model, i.e., the model that used the weather data as exogenous variables
- backend/initial_model.ipynb: Code for the original ARIMA model

6 Conclusion

We used bike data from HSL and the SARIMA model to predict the number of ongoing bike trips that have originated from or are arriving at a specific station at any given hour. We performed cross-validation to select the appropriate parameters and used the tuned parameters to deploy a Progressive Web Application. We believe this app creates value for the decision makers at HSL.

References

[1] Teemu Roos. *IDS 2024 Lecture 1*. 2024. URL: `https://moodle.helsinki.fi/pluginfile.php/5917477/mod_resource/content/6/IDS2024%20Lecture%201.pdf`.

[2] Helsinki Region Transport (HSL). *Open Data*. 2024. URL: `https://www.hsl.fi/en/hsl/open-data`.

[3] Open-Meteo. *Historical Weather API*. 2024. URL: `https://open-meteo.com/en/docs/historical-weather-api`.

[4] Ashish Kabra, Elena Belavina, and Karan Girotra. "Bike-Share Systems: Accessibility and Availability". In: *Management Science* 66.9 (2019), pp. 3803–3824.

[5] Pentti Saikkonen. *Stationaariset aikasarjat*. 2015.