Estimated delays of public transports in the Metropolitan Area

1. Introduction

Frequent schedule inaccuracies of public transport lines affect both passengers and operators. Delays of transport lines can cause missed transfers and lower passenger satisfaction. The purpose of this project was to analyze and visualize estimated delays of public transport vehicles in the Helsinki metropolitan area using data provided by Helsinki Region Transport (HSL).

Our primary goal was to identify when and where the delays for the transit lines occur most frequently, and to understand what patterns or factors might contribute to them, such as line number, location and even the bus stop. By detecting these patterns, we aimed to create a system that helps HSL's board and development team to improve schedule accuracy and operational reliability by changing it from a reactive approach to a more predictive approach.

The target users of this project are the HSL development team, who can use the results to enhance scheduling and planning and passengers, who ultimately benefit from more reliable public transport. The final product combines live data retrieval, automated delay computation, and an interactive visualization dashboard. It is designed to demonstrate how open data can be used to provide valuable insights into the performance of public transport systems.

---

2. Data Collection

We used HSL's open data services, specifically the GTFS (General Transit Feed Specification) and GTFS-RT (real-time) feeds. These are publicly available datasets that include both scheduled timetables and real-time updates for public transport in the Helsinki region.

Python scripts were written to fetch, clean, and merge these datasets. The GTFS data provided scheduled stop times and route information, while the GTFS-RT feed contained live vehicle positions and delay updates. Together, they allowed us to compare scheduled versus actual arrival times and calculate estimated delays.
Data was collected using Python from HSL's open data services. Two main data sources were used:

1. Static GTFS data (General Transit Feed Specification), which contains route, trip, and stop information.
   This dataset was downloaded as a ZIP file from the TransitFeeds portal:
   https://transitfeed.com/p/helsinki-regional-transport/735/latest/download
2. Real-time GTFS-RT feed, which provides up-to-date vehicle positions and timestamps:
   https://realtime.hsl.fi/realtime/vehicle-positions/v2/hsl

The project's Python script automatically downloads, extracts, and merges these datasets. The static GTFS files (stops.txt, stop_times.txt, trips.txt, and routes.txt) provide the scheduled timetable, while the real-time feed gives the live position of each vehicle.

Each real-time vehicle record is matched with its corresponding scheduled stop based on the route ID, trip ID, and geographic proximity using the Haversine formula. This allows the program to determine how far each vehicle is from its expected position and whether it is running late or early. The system also handles missing or inconsistent data, filters our extreme values and saves all valid entries to a CSV file (hsl_vehicle_delays.csv) on the user's desktop.

Challenges in data collection

The main challenges we faced included incomplete or inconsistent delay reports. Some vehicles lacked delay data or reported it irregularly, leading to missing records. Also managing large files presented some difficulties. Another main challenge was that HSL provides data for a maximum of one year only. Despite these challenges, using automated Python scripts helped streamline the data retrieval process and ensured that updates from the APIs were handled efficiently.

---

3. Data Preprocessing

The preprocessing stage focused on cleaning, organizing, and transforming the collected data into usable format for analysis. Our goals were to remove duplicate entries, handle missing or inconsistent records, and prepare features relevant for delay prediction.

We began by filtering out incomplete rows and correcting obvious data entry errors. Duplicate entries were removed to ensure that each trip-stop combination was represented

only once. All timestamps were converted to the correct Helsinki time zone to ensure consistency, as time zone mismatches initially led to misleading results.

Each vehicle's location was compared with scheduled stop coordinates to identify the most probable current stop. The Haversine distance was used to measure the distance between real-time GPS coordinates and scheduled stop coordinates. Only stops within a 5-minute time window were considered relevant.

Any records with missing coordinates, invalid timestamps, or extreme outliers were filtered out before saving.

---

4. Exploratory Data Analysis (EDA)

Before the visualisation, by the EDA we focused on understanding the characteristics and the patterns of the dataset which we used to predict the delay times.

We examined that the overall distribution of delays showed both positive and negative values, which means that vehicles arrived early (negative values) while the others were late (positive values). We saw that there were some noticeable cases with longer delays sometimes (but less than 5 min).

By comparing the delays across routes, we observe that which transport lines tended to run late usually. We also examined which certain areas or stops have the consistent delays and finally, we visualised the findings using maps, line graphs and scatter plots, etc.

---

5. Machine Learning Model

For the estimation part of the delays, we used XGBoost and Random Forest Regressor. We used the gathered data and trained both of the models to give us estimations on the delays. By increasing the data set 10x and then to 20x we gained better insight on how the delays are scattered.

Out of the two models the Random Forest Regressor performed a lot better, resulting in us discarding the result gained from XGBoost. All of the data visualised is gained by using RFR. We tried to tweak the parameters such as increasing the number of trees, but the specific values we landed on worked the best. The performance was evaluated using Mean absolute error (MAE), Root mean squared error (RMSE) and the coefficient of determination (R^2). The RFR performed better with these metrics consistently throughout the experimentation, netting us the results of MAE = 88.78, RMSE = 115.75 and R^2 = 0.163. We used a split of 80/20 on the training data to combat overfitting, giving us the best results.

The models started to perform better once we increased the data set to around 10000 lines, which removed the uncertainty and scatterness of the data we had at the beginning. The data was difficult to work with for the models, as the performance wasn't ideal regardless of our tweaks and changes. We plotted the data and made it viewable, which in turn helped us predict the delays given the current data.

---

6. Application and Visualization Interface

We used Tableau and Streamlit to visualize the charts. Streamlit was used to create the web application to interact better with the data. The interface allows users to search by line number and view charts and histograms of delay trends. We used Tableau to design a visually polished dashboard for presentation. The dashboard displayed Line-by-line comparisons of delays.

A Streamlit web application was developed to visualize the collected delay data in real time. The dashboard allows the user to trigger live data collection and displays the results directly after retrieval. The interface includes:

- A map view showing current vehicle positions
- A bar chart of average delay per line number
- A data table with all raw delay records

The app runs interactively so users can click "Collect live data" to fetch a new batch of information from HSL's APIs. The visualization updates immediately after data retrieval.

---

7. Reflections and Learnings

What worked

The automated data retrieval system and integration of multiple GTFS data sources worked reliably. Using the GTFS-realtime allowed accurate extraction of real-time data. Our workflow was smooth and each iteration of the project was better than the last.

What didn't work

Mapping the transport lines to the correct stops was difficult for us at the beginning. All the delay times were appearing wrong because the delays were measured against the first transport line of the day instead of the current one. We figured out that we could filter the data to take into account only the delays of max. 5 minutes which worked for us a lot better and gave us finally relevant data. We also had to change to the correct time zone because

our code did not take into account the timezone of Helsinki. HSLs data is very incomplete, which resulted in difficulties when trying to gain access to data. The GTFS-format generally includes fields such as “trip_id”, which is the identifier specific to each vehicle. This made us unable to properly determine which vehicles were actually delayed, since our attempts at mapping the delays to vehicles resulted in the program overlaying the delays to the whole route.

What we learned

This project gave us a lot of experience in handling real-time APIs, geospatial calculations, and time-based data synchronization. It highlighted the importance of data validation and clear handling of edge cases (e.g. missing values). From a development view, it also demonstrated how lightweight tools like Streamlit can make complex data systems easily accessible through simple web interfaces.

---

8. Future Work

Future improvements could include:

- Continuous data collection over longer periods to build a larger historical dataset.
- Integration of weather data and traffic data to improve predictive accuracy.
- Taking into account the weekday or holidays

---

9. Conclusion

This project implemented a way to estimate and visualize public transport delays in the Helsinki metropolitan area. The system collected both static and real-time data directly from HSL’s APIs, cleaned and processed the information, computed delays, and presented the results in an interactive dashboard.