Final Project Report: Traffic Prediction

Adhwaith Natarajan Ibrahmin Mohsin James Dankwah Krish Vazirani
University of North Carolina at Chapel Hill
adhnata@ad.unc.edu imohsin@unc.edu danknja@unc.edu khv@unc.edu

Abstract

The system takes in raw CSVs of vehicle positions and GTFS tables, combines observations with scheduled stop times, and uses a feature engineering suite to get temporal (hour, day, rush-hour flags), route (route/trip/vehicle identifiers and frequencies), traffic (speed, bearing, movement state), and delay-derived features (elapsed-time based arrival/departure delays). This project uses an RF pipeline for predicting bus arrival delays using GTFS schedule data and real-time vehicle position feeds. The system ingests in raw CSVs of vehicle positions and GTFS tables, combines observations with scheduled stop times, and uses a feature engineering method to get temporal (hour, day, rush-hour flags), route (route/trip/vehicle identifiers and frequencies), traffic (speed, bearing, movement state), and delay-derived features (elapsed-time based arrival/departure delays). The Random Forest regressor is trained on the processed feature matrix to predict arrival delay in minutes. The pipeline has support for both model-based predictions and fallbacks to observed delays while being designed to handle common GTFS / real-time mismatches (robust trip/stop matching and time-proximity rematching). This approach balances between interpretability (importance of features) and practical performance to predict transit delays in real time.

1 Introduction

In recent times, improvements in predictive technology have allowed public transit agencies to rely more heavily on real-time predictions of vehicle arrival times. Such integrations have helped improve the passenger experience while optimizing internal operations, ultimately helping inform downstream applications such as dynamic dispatch and passenger information systems. Acquiring accurate short-term arrival predictions is challenging because they must reconcile scheduled (GTFS) data with noisy, asynchronous real-time feeds (GPS/AVL), heterogeneous vehicle reporting rates, and operational disruptions such as traffic congestion and dwell times at stops.

Using Random Forest, this development stream predicts bus arrival delays using GTFS schedule information and recorded vehicle-position streams. The pipeline (1) ingests GTFS tables and vehicle position logs, (2) aligns realtime observations to scheduled stops using robust matching and time-proximity heuristics, (3) extracts engineered features that capture temporal context (hour, day-of-week, rush-hour flags), route identity and frequency, vehicle motion (speed, bearing, moving/stopped state) and elapsed-time delay calculations relative to trip starts, and (4) trains the model to predict arrival delay in minutes. This approach also supports a model-free fallback that takes advantage of observed delays when a trained model is not available.

2 Related Work

Research on transit arrival and travel-time prediction falls into state-estimation work and data-driven supervised learning. State-estimation techniques such as Kalman filtering formulate vehicle dynamics and noisy observations with state and observation equations.

$$x_k = Ax_{k-1} + Bu_k + w_k, \qquad z_k = Hx_k + v_k,$$

Such are commonly used for recursive tracking and uncertainty propagation (1). In contrast, many applied systems adopt supervised regression: tree-based ensembles (e.g., Random Forests and gradient-boosted trees) are very popular because they handle heterogeneous feature types and produce interpretable importance scores (2; 3). When abundant, continuous historical streams exist, sequence models (LSTMs, temporal convolutional models) can capture longer temporal dependencies but typically require more data and engineering (4). Feature design: time-of-day and day-of-week indicators, rush-hour flags, vehicle speed and bearing, stop-sequence context, and elapsed-time delay measures computed relative to trip starts consistently improve performance in the literature (5). Evaluation conventions report mean absolute error and root mean squared error,

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n} y_i - \hat{y}_i, \qquad \mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2},$$

with operational targets often aiming for sub-2-minute MAE. Practical deployments also emphasize robust matching of real-time observations to schedules (combining identifier joins with sequence/time-proximity heuristics), modular pipelines, and graceful fallbacks for missing or inconsistent data (6).

3 Methods

We work with two complementary datasets: GTFS static schedule data and historical real-time vehicle-position logs from a transit agency (Lametro). The real-time dataset consists of 574,123 vehicle-position records collected over a three-hour window (2025-11-13 from 12:33:01 to 15:44:36). Schedule entries are parsed to obtain per-trip stop sequences and scheduled arrival/departure times; real-time records are converted to date times, converted to numeric types where appropriate, and cleaned for inconsistent column names and missing values. Observations are aligned to scheduled stops using a staged matching strategy: first attempt exact identifier joins, then match by trip and stop sequence where available, and finally rematch unmatched observations by selecting the scheduled stop with the nearest arrival time within. These records day of the week, continuous time of day, weekend, and rush hours, and 11,591 stop IDs.

Feature construction produces a compact, interpretable set of inputs that capture: temporal, route/trip, vehicle motion, and schedule-relative signals. Temporal features include hour, day of the week, continuous time of day, weekend, and rush-hour flags. Route and trip context are encoded via parsed identifier components and occurrence frequencies. Motion features incorporate numeric speed and bearing (with categorical speed bins and cardinal bearing buckets), location validity flags, and one-hot encodings of vehicle status (e.g., stopped, in-transit). Delay features are computed as elapsed actual versus scheduled seconds since trip start, producing arrival/departure delay values (converted to minutes) and boolean indicators for early/late/on-time cases; stop-sequence agreement is recorded as an additional feature.

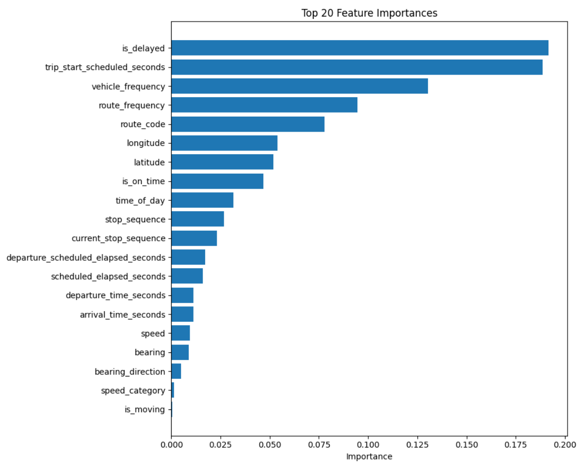

Figure 1: Top 20 most influential features.

For modeling, we cast delay prediction as a supervised regression task. Given the interpretability and robustness requirements of the highly nonlinear structure of transit delay, we adopt a Random Forest Regressor. The model is trained using an 80/20 train–test split (40,188 training examples, 10,047 test examples). Random Forests work for this setting because they handle diverse feature types, can automatically learn nonlinear interactions (e.g., time-of-day × route), and are also resilient to outliers. A fixed random state is used to guarantee predictable splits and model behavior, while hyperparameters are maintained close to defaults for stability and reproducibility. Standard regression metrics, such as mean absolute error (MAE), mean squared error (MSE), and root mean squared error (RMSE), are used to measure model performance.

The pipeline sends out the trained model and the ordered list of feature columns needed for inference. It also sends out diagnostic outputs for route and stop-level summaries (average delay, on-time percentage, worst stops) and visualizations of predictions compared to ground truth. The design focuses on modularity, meaning that the various supervised learners and alternative matching can be added to the workflow with little to no changes as well.

4 Results

Discussion. The model's MAE of approximately 1.86 minutes indicates the approach yields near-operational accuracy for short-term arrival predictions on the evaluated dataset; RMSE $\approx$ 3.42 minutes shows that occasional larger errors remain (long tail). Given that arrival delays in the dataset span from –85 to +123 minutes with great variance, this level of accuracy is notable, as the model is extracting meaningful structure from a highly noisy and heavy-tailed real-world distribution. Inspection of residuals and the predictions scatter plot suggests the model performs well for the bulk of observations (±3–4 minute band) but underestimates extreme delays; this is common in practice when atypical congestion or large dwell events occur. It is important to note that the model uses no external inputs (e.g., traffic conditions, weather, etc.), which likely contributes to its difficulty in estimating rare extreme delays. The feature importance analysis supports the design choice of combining schedule-relative elapsed features with instantaneous motion signals; temporal indicators and speed categories provide complementary information.

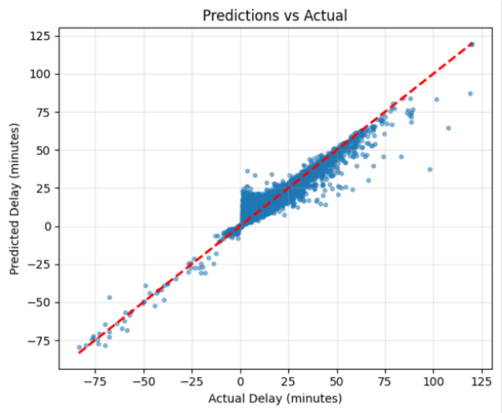

Figure 2: Predicted vs. Actual delays with $45°$ calibration line.

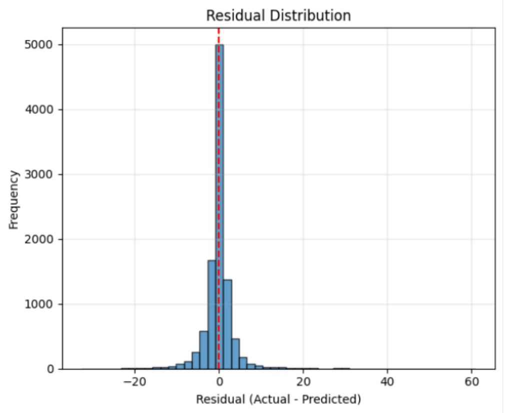

Figure 3: Residual distribution (Actual - Predicted)

The model's error characteristics suggest that it is suitable for rider-facing estimated-arrival displays, where a sub-2-minute MAE is a common goal. But, large outliers should be dealt with using extra protections (e.g., anomaly detection, fallback to recent observed delays, or flagging low-confidence predictions).

Summary. In summary, the evaluation demonstrates that a compact feature set encoding temporal context, schedule-relative elapsed times, and simple vehicle motion indicators, together with a Random Forest regressor, provides strong baseline performance (MAE 1.86 min, RMSE 3.42 min) on the available dataset. Future improvements can target reducing tail errors (e.g., richer spatial features, external traffic inputs, or sequence models) and integrating uncertainty estimates for safer operational deployment.

5 Conclusion

This undertaking set out to determine whether a simple learning pipeline could deliver reliable short-term delay predictions using only GTFS schedules and real-time vehicle-position feeds. The results show that this is feasible and that a lightweight model like Random Forest can extract enough structure and information from potentially noisy operational data to meet accuracy marks that are acceptable and expected in rider-facing use cases. Most of the system's performance comes from the data pre-processing stage, especially the staged matching between the real-time observations and scheduled stop times. The feature choices that reflect how transit delays actually unfold over the course of a trip help as well.

Even though the model performs well in general, it does struggle with rare disruptions and abnormally large delays. This isn't surprising due to the absence of other types of information (i.e. road incidents, closures, vehicle load, etc.) that affect travel times past route timings and speed. Incorporating external signals and experimenting with temporal models that track conditions that evolve over longer horizons would make the system more robust in scenarios where reliability matters most.

Overall, the work demonstrates that a modular data ingest system combined with well-engineered features can produce a practical delay-prediction tool with modest computing demands. The approach is easy to extend beyond the current build.

References

[1] G. Welch and G. Bishop. An Introduction to the Kalman Filter. University of North Carolina at Chapel Hill, Department of Computer Science, 1995.

[2] L. Breiman. Random Forests. Machine Learning, 45(1):5–32, 2001.

[3] T. Chen and C. Guestrin. XGBoost: A Scalable Tree Boosting System. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), 2016.

[4] S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. Neural Computation, 9(8):1735–1780, 1997.

[5] E. Vlahogianni, M. Karlaftis, and J. Golias. Short-term traffic forecasting: Where we are and where we're going. Transportation Research Part C: Emerging Technologies, 43:3–19, 2014.

[6] B. Ferris, K. Watkins, and A. Borning. OneBusAway: Results from Providing Real-Time Arrival Information for Public Transit. In Proceedings of the 28th International Conference on Human Factors in Computing Systems (CHI), 2010.