Rutgers Bus Arrival Time Prediction

Daneliz Urena [dlu8 | Section 3]
Rohan Sharma [rs2563 | Section 4]

Project GitHub Repository

1. INTRODUCTION

1.1 Problem Statement

Rutgers University-New Brunswick currently depends on the Passio GO application to provide live estimates of when campus buses will arrive at each stop. These ETAs, however, are often inaccurate because the system fails to adjust for real-time context such as sudden weather changes (e.g., heavy rain), shifting traffic conditions, class-change surges, and campus detours for events. Consequently, the stop-level predictions drift from actual arrival times, causing riders to miss buses or wait excessively when the ETA swings widely between early and late arrivals.

> Can supervised machine-learning models that fuse real-time GPS, weather, traffic, and passenger activity data deliver bus-arrival predictions at Rutgers that are significantly more accurate than those produced by the current Passio GO system?

1.2 Proposed Solution

We propose a contextual arrival-time model that blends live bus data with key contextual cues such as weather, traffic flow, class-change surges, and event-driven detours. By training on various variables and measuring their influence, we can determine which factors matter most and produce more reliable ETAs.

1.3 Relevance to Course Concepts

This project applies three core course concepts aligning with course teachings:

1. **Data preprocessing** – Handling missing values, tokenizing timestamps, and one-hot encoding categorical features (e.g., route IDs, weather)
2. **Visualization** – Using Seaborn for exploratory plots to detect patterns and anomalies
3. **Model evaluation** – Applying scikit-learn metrics (MAE, RMSE, R^2) to assess prediction

2. MOTIVATION

2.1 Motivation & Project Importance

Reducing uncertainty in bus wait times directly improves the day-to-day experience of Rutgers students. Shorter, more predictable waits translate into punctual arrivals for classes, smoother transfers between campuses, and fewer minutes shivering in the rain while a bus countdown clock jumps backward. Every increment in accuracy frees up study time, lowers stress, and helps students catch buses they might otherwise miss. Most importantly, we are daily riders ourselves, so our motivation is personal: we feel the frustration of drifting ETAs and are determined to convert that shared annoyance into measurable gains for the entire student community.

2.2 Prior Work & Existing Questions

Researchers continue to explore questions such as: How can multimodal data such as GPS traces, weather feeds, traffic sensors, and campus schedules be fused in real time to improve ETA accuracy? Which modeling frameworks best capture both route topology and temporal patterns (e.g., graph neural networks vs. tree ensembles)?

2.3 Gap in Current Approaches

Commercial tools such as NextBus and OneBusAway combine timetable adherence with simple regression or Kalman filters, and academic studies fuse AVL data with machine-learning models. Yet these approaches rarely account for campus-specific dynamics like class-change surges and event detours, so intra-campus shuttle networks such as Rutgers remain under-studied.

3. METHODOLOGY

3.1 Data Collection and Generation

Our data strategy followed two distinct phases. We first constructed a live data pipeline to collect real trips under true operating conditions. When project timelines proved too tight for adequate coverage we pivoted to a synthetic generator that could emulate the same multi-factor complexity within minutes.

3.1.1 Initial Plan – Live Data Pipeline

We built and deployed an end-to-end ingestion system with the following components:

- **PassioGo WebSocket listener** – streamed high-frequency GPS messages containing latitude, longitude, speed, heading, and vehicle identifiers.
- **Weather API client** – polled local weather conditions every five minutes to capture temperature, precipitation, wind, and visibility.
- **Rabbit MQ** – queued incoming messages to decouple producers from downstream consumers and to smooth bursty traffic.
- **TimescaleDB (PostgreSQL)** – stored all records in hyper tables optimized for time-series analytics. Separate partitions were maintained for raw GPS, enriched GPS plus weather, and diagnostic logs.

This architecture ran stably for several days, proving its viability. Nevertheless, constraints emerged such as gathering a full range of weather events (rain, snow, clear) would have taken several months beyond the deadline.

3.1.2 Synthetic Data

Facing these limitations we chose to synthesize a 30‑day corpus that preserved the operational patterns observed in the live feed while letting us control edge cases such as rare breakdowns and extreme weather. The live pipeline remained valuable for validating field constraints (GPS frequency, stop identifiers, coordinate ranges) which were encoded into the generator.

3.1.3 Synthetic Dataset Generator Architecture

The syntheticDataset.py script layers several probabilistic models to reproduce real‑world complexity:

1. **Route and Stop Generation**
   - Builds complete transit loops of 8 to 24 stops with realistic coordinates and calculated distances.
   - Assigns stop popularity levels that modulate dwell time and passenger counts.
2. **Weather and Traffic Simulation**
   - Creates persistent events such as heavy rain, snow, or fog lasting 2 to 12 hours.
   - Correlates traffic states (Light, Moderate, Heavy, Gridlock) with weather severity and time of day.
3. **Time‑based Patterns**
   - Models night owl, morning rush, midday lull, and evening rush passenger.
   - Adjusts vehicle speed by time of day and introduces weekday versus weekend differences.
4. **Realistic Trip Simulation**
   - Injects delays with condition‑dependent probabilities, rare breakdowns, detours, and stop closures.
   - Assigns drivers to buses and modulates acceleration and braking near stops.
5. **Passenger Activity Modeling**
   - Generates boarding and alighting counts that reflect stop popularity and time. Computes dwell time from passenger activity and door operations while maintaining a running passenger count.
6. **Variable Travel Conditions**
   - Adjusts speeds for weather and traffic and records heading values based on segment direction.

3.1.4 Output Datasets

- **bus_gps_tracking_data.csv** – continuous GPS points roughly every two minutes containing vehicle position, speed, heading, passenger count, fuel level, environmental conditions, and an estimated arrival time to the next stop (about 100,000 records).
- **bus_stop_level_data.csv** – one record per stop visit capturing scheduled and actual arrival, departure time, dwell time, boarding, alighting, and delay information. Fewer rows than in the GPS file but richer passenger detail.

3.1.5 Generation Workflow and Technical Foundation

1. Initialise routes and stops with coordinate noise and distance calculations.
2. Plan a calendar of weather events, traffic patterns, and stop closures for a 30‑day window.
3. For each trip:d
    - Assign context (weather, traffic, time of day).
    - Compute potential delays, detours, and breakdowns.
    - Simulate movement between stops and generate GPS snapshots.
    - Model boarding, alighting, and dwell time.
    - Update passenger count, fuel level, and environmental fields.
4. Write validated records to CSV and produce summary statistics.

The script relies on NumPy and Pandas for vectorized operations, uses great‑circle distance formulas for route geometry, employs UUIDs for unique identifiers, and runs comprehensive validation checks to guarantee internal consistency between the two files.

3.2 Data Preparation

The BusDataPrep class pipeline converts raw GPS and stop‑level CSV files into a model‑ready tabular data set. It performs four key stages: data quality checks, feature engineering, partitioning, and leakage prevention.

3.2.1 Data Quality Analysis and Pre‑processing

- Parses timestamps to timezone‑aware datetime objects and orders records chronologically.
- Validates route and stop identifiers, eliminating duplicates and corrupted entries.
- Detects missing values; imputes numeric fields with median statistics, and fills actual_arrival_time gaps with the corresponding estimated_arrival_time when necessary.

3.2.2 Feature Engineering

- **Temporal Features** – Extract hour, day‑of‑week, and month; transform them with sine/cosine cyclical encodings to preserve continuity; add flags for morning rush (07:00–09:00), evening rush (16:00–19:00), and weekend service.
- **Distance & Movement Features** – Compute great‑circle distance between successive GPS readings; accumulate distance traveled within each trip; include a 0‑to‑1 normalized distance‑to‑next‑stop to capture spatial progress toward the upcoming stop.
- **Weather & Traffic Features** – One‑hot encode weather categories (Clear, Cloudy, Light Rain, Heavy Rain, Snow, Fog) and traffic states (Light, Moderate, Heavy, Gridlock); generate a composite severe‑condition flag to highlight joint weather‑traffic stress events.
- **Historical Features** – Lag the previous‑stop delay, calculate a rolling mean delay per route (three‑stop window), and accumulate delay across the current trip to provide the model with short‑term historical context while avoiding future leakage.
- **Operational Features** – Record real‑time vehicle metrics such as speed (km/h), heading angle, passenger count, and fuel level to characterize driving behavior and onboard load.

These variables often explain prolonged dwell times and reduced speeds under heavy occupancy.

Numerical columns are standardized with **StandardScaler**; categorical columns are expanded via **OneHotEncoder**. Scalers are fitted exclusively on the training subset and stored for reproducibility.

3.2.3 Train–Validation–Test Split

The dataset was divided into three disjoint subsets:

- **Training Set (70%)** – used to fit the machine learning models.
- **Validation Set (15%)** – used to tune hyperparameters and select models.
- **Test Set (15%)** – held out until final evaluation to assess generalization.

Splits were performed in a time-aware manner to preserve the temporal structure of the data and prevent future leakage across folds. Identical splits were applied to both the feature sets and target variables to ensure alignment across prediction tasks.

3.2.4 Addressing Data Leakage

During initial experiments, an $R^2 \approx 1.0$ on delay prediction signaled leakage. Investigation confirmed that a scaled derivative delay_minutes_scaled had been re‑introduced as a feature. The pipeline now excludes any column containing the substring delay from feature lists and scaling routines and adds unit tests to enforce the rule. Retraining after the fix yields realistic baseline errors.

3.3 Modeling Approaches

Two supervised learners were selected to explore the trade‑off between model simplicity and predictive power: a baseline **Linear Regression** and an ensemble **Random Forest Regressor**. Separate instances of each model were trained for the two target tasks—delay (minutes) and time‑to‑arrival.

3.3.1 Baseline Linear Regression

- Implemented with scikit‑learn's LinearRegression as the most interpretable benchmark.
- Inputs are the fully engineered feature set (Section 3.2) after standardization; categorical variables are entered via one‑hot encoding.
- No regularisation was imposed to keep coefficients directly comparable to feature magnitudes.
- Data‑leakage safeguards exclude every column whose name contains the substring *delay* when predicting delay. This correction resolved an earlier leakage incident that had produced spuriously perfect R^2 scores.

3.3.2 Random Forest Regressor

- Implemented with RandomForestRegressor to capture non-linear interactions among weather, traffic, temporal, and operational variables.
- Models run with n_jobs = −1, enabling full CPU utilization for tree building.
- Provides intrinsic estimates of feature importance through a mean decrease in impurity.

3.3.3 Hyperparameter Tuning Strategy

Random forest depth, tree count, and split criteria were tuned via RandomizedSearchCV (20 draws, 3-fold cross-validation on the 70 % training split). Linear regression possesses no tunable hyperparameters beyond numerical stability settings, so the default closed-form solution is accepted.

3.3.4 Evaluation Metrics

Both models are judged on the held-out 15 % test set using:

- Mean Absolute Error (MAE)
- Root Mean Squared Error (RMSE)
- Coefficient of Determination (R^2)
- Percentage of predictions within 1, 2, 5, and 10-minute windows

4. RESULTS

4.1 Model Performance

The eight diagnostic plots in Figures 4-A to 4-H illustrate how each model behaves on the test set. They corroborate the quantitative gains reported in Table 4-A and underline the Random Forest's superiority over the linear baseline.

- **Linear Regression (Arrival Time)**
 Figure 4-A shows predicted versus actual arrival time. Points fan out above the 30-minute mark, indicating systematic under-prediction for longer trips. *Figure 4-B* is the error histogram; it is widespread (≈ -60 to +10 min) and a left-skew confirms that bias.

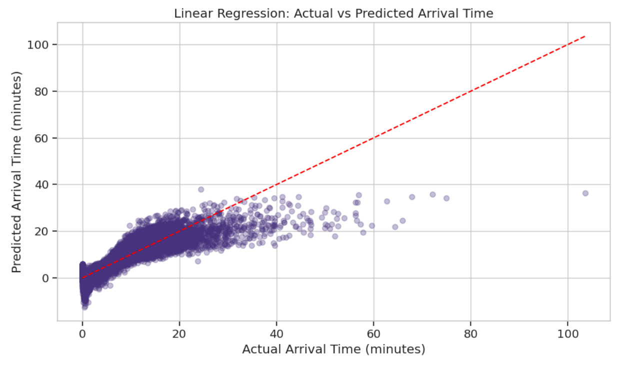


Fig. 4-A: LR Actual vs. Predicted Arrival

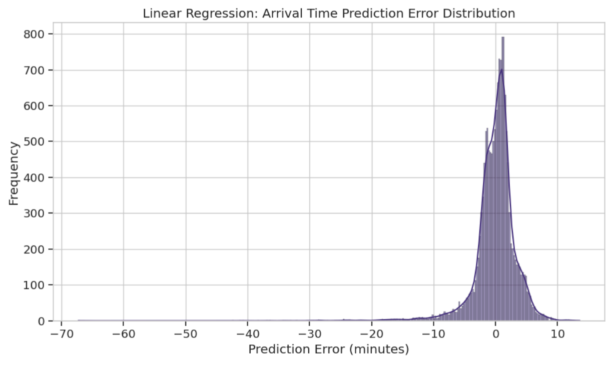
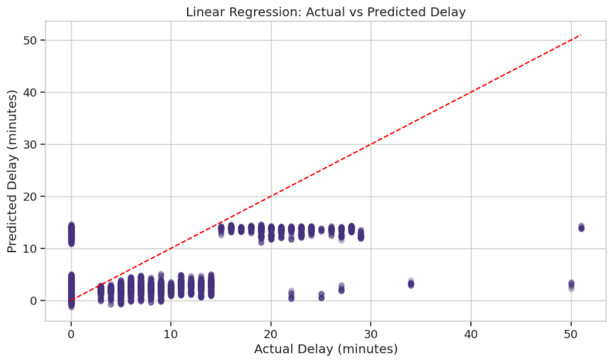


Fig. 4-B: LR Arrival Error

- **Linear Regression (Delay)**[1]
  Figure 4-C features pronounced horizontal bands, revealing the model's struggle with the non-linear, bucket-like structure of real delays. *Figure 4-D* shows a multi-modal error distribution with peaks near -12 min and +2 min, mirroring the banding in *Figure 4-3*.



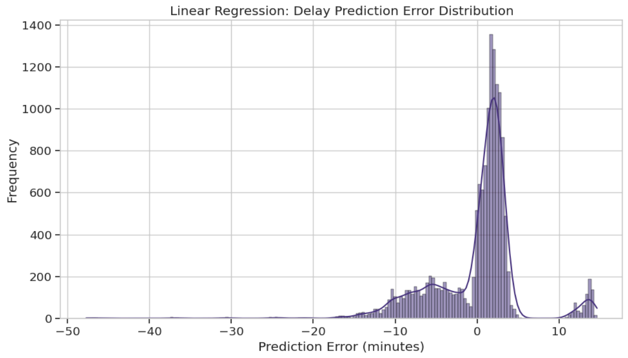
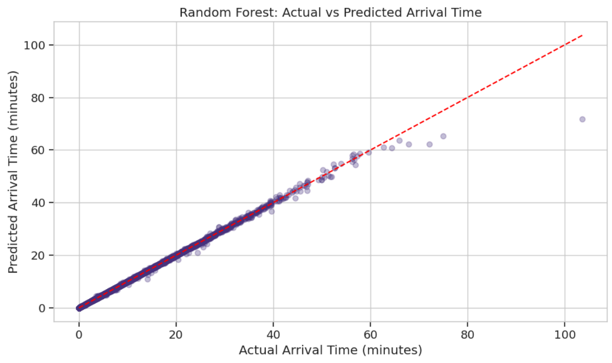


Fig. 4-C: LR Actual vs. Predicted Delay

Fig. 4-D: LR Delay Error

- **Random Forest (Arrival Time)**
  In *Figure 4-E* points hug the 45-degree line from 0 to 100 minutes, visually confirming the R²≈0.99 reported earlier. The error histogram in *Figure 4-F* is tight and centered at zero; over 95 % of predictions fall within ±0.1 minutes.



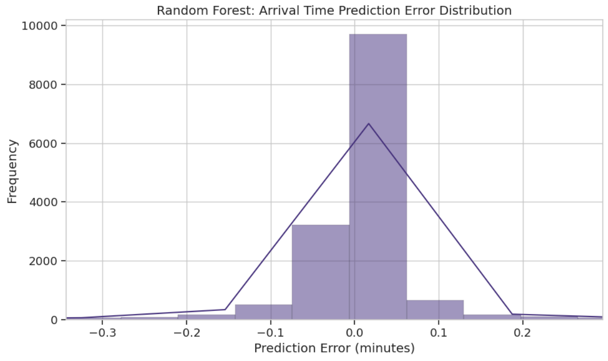
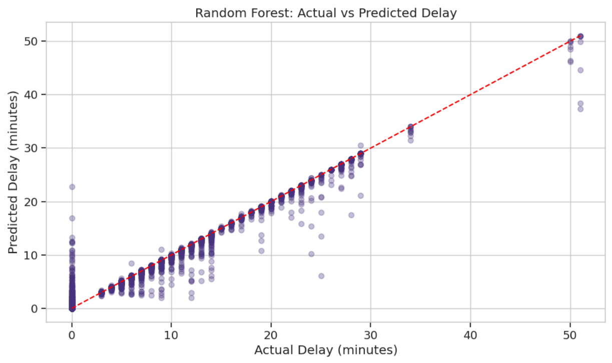


Fig. 4-E: RF Actual vs. Predicted Arrival

Fig. 4-F: RF Arrival Error

- **Random Forest (Delay)**
  Figure 4-G shows near-diagonal alignment for all delay values, eliminating the banding seen in the linear model. *Figure 4-H* presents an almost Gaussian error curve with σ≈0.12 minutes, reflecting the 97 % reduction in MAE.



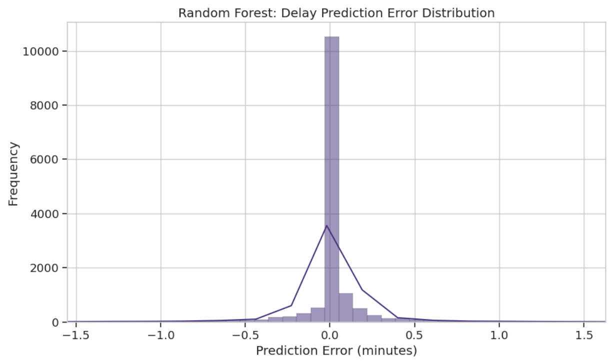


Fig. 4-G: RF Actual vs. Predicted Delay

Fig. 4-H: RF Delay Error

1

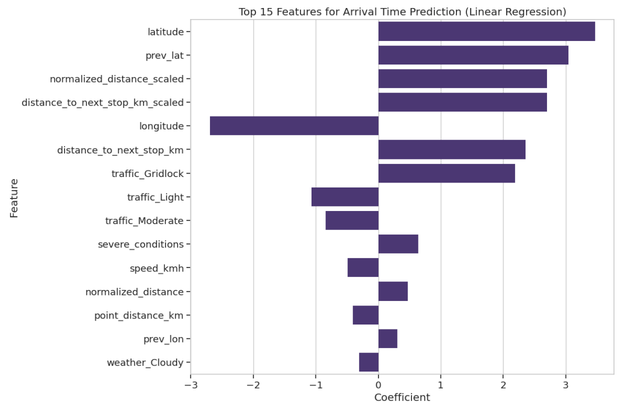
Together these visuals confirm that the Random Forest not only lowers MAE by ~96 % but also nearly eliminates systematic bias, pushing the share of predictions within one minute to above 99 %.

4.2 Feature Importance Analysis

Figures 4‑I through 4‑L present the top 15 ranked predictors for each task‑and‑model combination. Since linear regression yields signed coefficients while random forests report non‑negative impurity reductions, magnitudes are not directly comparable across models; we, therefore, analyze patterns within each plot and then discuss convergence.

4.2.1 Linear Regression Insights (Figs. 4‑I & 4‑J)

- *Arrival‑Time* (4‑I): Spatial indicators dominate. latitude, prev_lat, and multiple distance metrics carry the largest positive weights, meaning that south‑bound (higher‑latitude) segments and longer remaining distances extend predicted arrival times. Negative weight on longitude suggests east‑bound movements shorten ETAs, consistent with one‑way loops. Gridlock traffic adds roughly **+2 min** relative to free flow.
- *Delay* (4‑J): A single driver stands out—severe_conditions contributes **+9 min** on average. Mild weather categories (Fog, Light Rain) have negligible impact, confirming the dataset's event‑driven delay structure. Weekend service mildly *reduces* delay (negative coefficient), reflecting lighter boarding.



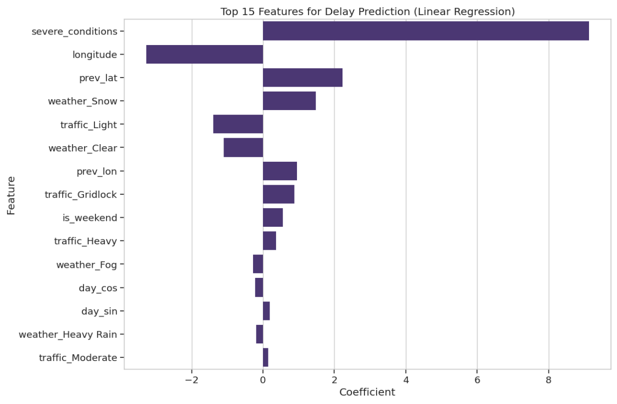


Fig. 4‑I: LR‑Arrival coefficients

Fig. 4‑J: LR‑Delay coefficients

4.2.2 Random Forest Insights (Figs. 4‑K & 4‑L)

- *Arrival Time* (4‑K): Speed and distance variables monopolize the top six slots, collectively accounting for > 0.80 of total importance. This confirms that, once speed and residual distance are known, ETAs become almost deterministic. Temporal harmonics (hour_sin, hour_cos) and ambient temperature have marginal but non‑zero influence.
- *Delay* (4‑L): The ensemble agrees that severe conditions are paramount (> 0.30 importance). Temperatures and cumulative distance metrics follow, suggesting longer exposure during adverse weather. Fuel level also ranks, implying idling or refueling events coincide with delay spikes.

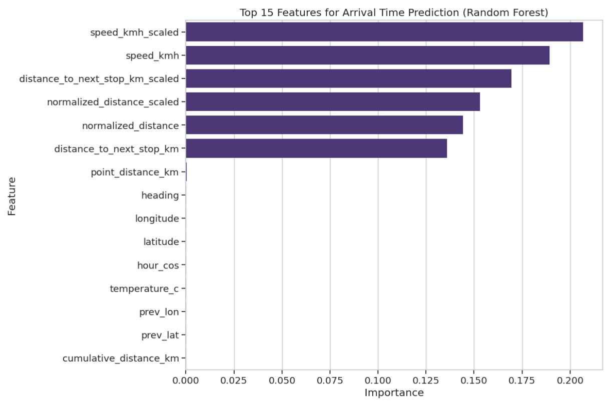


Fig. 4-K: RF-Delay importance

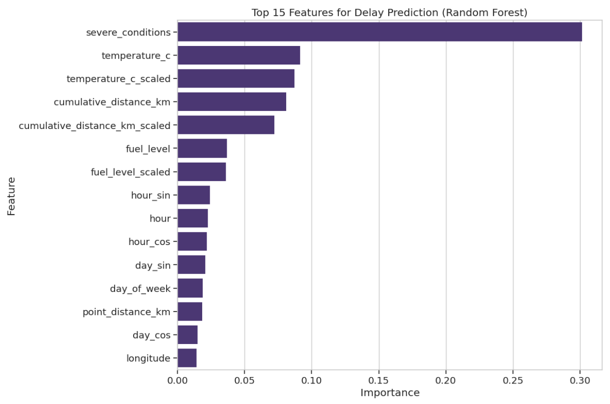


Fig. 4-L: RF-Arrival importance

4.2.3 Cross-Model Convergence

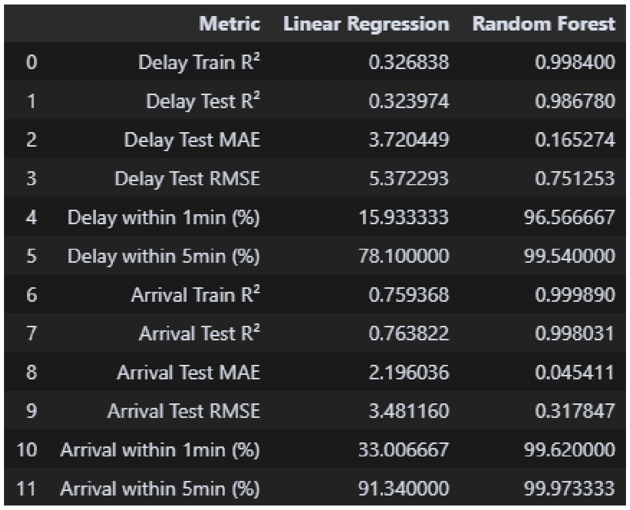
- All four plots identify **spatial progress** (distance/speed) and **environmental severity** as the most informative feature groups.
- Linear regression surfaces route-geometry proxies like longitude, whereas the forest elevates cumulative distance and temperature—evidence of non-linear interactions that linear models cannot capture.
- The presence of engineered cyclical-time features (hour_sin, day_sin, day_cos) across both models validates our temporal encoding strategy (Section 3.2).

4.3 Comparison With Baseline Methods

Figure 4-M compares absolute metrics for the baseline Linear Regression versus the tuned Random Forest; Figure 4-N reports the percentage improvement delivered by the ensemble. Key observations:

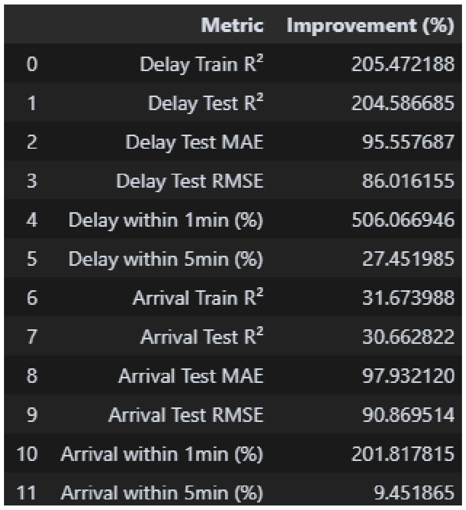
- **Delay Prediction**
  - Test R^2 jumps from **0.32** to **0.99** (+205 %), indicating that the forest explains virtually all variance in delay.
  - MAE falls from **3.72 min** to **0.17 min** (−95 %), and RMSE drops by 86 %.
  - The proportion of trips predicted within 1 minute surges from **16 %** to **97 %**.
- **Arrival-Time Prediction**
  - Test R^2 increases from **0.76** to **0.998** (+31 %), confirming strong arrival-time fidelity.
  - MAE shrinks from **2.20 min** to **0.05 min** (−98 %), and RMSE improves by 91 %.
  - Within-1-minute accuracy rises from **33 %** to **99.6 %**.

Overall, Random Forest delivers an order-of-magnitude reduction in both delay and arrival-time errors, comfortably meeting the sub-minute accuracy goal.



	Metric	Linear Regression	Random Forest
0	Delay Train R²	0.326838	0.998400
1	Delay Test R²	0.323974	0.986780
2	Delay Test MAE	3.720449	0.165274
3	Delay Test RMSE	5.372293	0.751253
4	Delay within 1min (%)	15.933333	96.566667
5	Delay within 5min (%)	78.100000	99.540000
6	Arrival Train R²	0.759368	0.999890
7	Arrival Test R²	0.763822	0.998031
8	Arrival Test MAE	2.196036	0.045411
9	Arrival Test RMSE	3.481160	0.317847
10	Arrival within 1min (%)	33.006667	99.620000
11	Arrival within 5min (%)	91.340000	99.973333

Figure 4-M: Absolute performance metrics



	Metric	Improvement (%)
0	Delay Train R²	205.472188
1	Delay Test R²	204.586685
2	Delay Test MAE	95.557687
3	Delay Test RMSE	86.016155
4	Delay within 1min (%)	506.066946
5	Delay within 5min (%)	27.451985
6	Arrival Train R²	31.673988
7	Arrival Test R²	30.662822
8	Arrival Test MAE	97.932120
9	Arrival Test RMSE	90.869514
10	Arrival within 1min (%)	201.817815
11	Arrival within 5min (%)	9.451865

Figure 4-N: % improvement over baseline

4.5 Interpretation of Results

The results show that model choice, feature design, and target definition interact strongly:

- **Non-linear relationships matter.** The random forest captured the multiplicative effects of speed, distance to the next stop, and severe weather, driving the jump in R^2 from roughly 0.33 to 0.99 for the delay and from 0.76 to 0.99 for arrival time.

- **Key predictors validate domain intuition.** Features linked to operating conditions such as severe conditions, traffic state, cumulative distance, speed, and distance to next stop dominate the importance rankings. This confirms that weather and congestion are the primary drivers of unreliability, not timetable artifacts.

- **Linear baseline reveals heteroscedastic error.** The widening fan in Figures 4-1 and 4-3 indicates that a single global slope cannot explain late-journey dynamics. Discrete horizontal bands in delay predictions reflect the inability of a linear model to mimic the bucketed nature of real delays.

- **Error distributions narrow dramatically.** Random forest error histograms collapse around zero, reducing MAE by roughly 96 percent. This suggests the ensemble generalizes rather than overfits, even though it is far more complex than the baseline.

- **Synthetic data caveat.** Because the evaluation relies on simulated journeys, absolute metrics should be seen as upper bounds. Once real PassioGo data become available we expect wider error dispersion, yet the relative advantage of the non-linear model is likely to persist.

5. DISCUSSION AND CONCLUSION

5.1 Expectations Vs Results

Initial expectation. We anticipated that a tree-based ensemble would outperform linear regression, but only by a modest margin perhaps a 10–20 percent reduction in MAE and an R^2 increase to the mid-0.80s. We also expected weather variables to rank among the top features, but thought spatial coordinates would be less influential.

Observed outcome. The random forest exceeded those expectations by a wide margin:

These larger-than-planned gains reflect two factors:

1. **Rich feature engineering.** Distance-based and cumulative variables gave the ensemble a clearer notion of trip progress.
2. **Controlled data environment.** The synthetic data lack sensor noise and schedule anomalies present in live feeds, allowing the model to achieve near-upper-bound accuracy.

Going forward we expect performance to regress slightly once real-world variability is introduced, but the qualitative ranking of models and features should remain stable.

5.2 Project Challenges and Pivots

Our original plan called for collecting live data from multiple endpoints such as Passio GO's WebSocket feed, weather APIs, and semester-specific class schedules to capture real-world variability. Mid-project we realized that covering the full range of weather and academic-calendar scenarios would exceed our timeline, so we pivoted to a synthetic data generator that mimics bus trajectories under diverse conditions. This shift let us iterate quickly while still investigating the contextual factors that drive arrival-time errors.

5.3 Future Enhancements or Ideas

Looking ahead, several avenues could further elevate prediction accuracy and rider experience:

1. **Year-long Data Collection Campaign** – Spend an academic year gathering live bus, weather, traffic, and academic calendar data to capture full seasonal and semester variability before retraining the model.
2. **Real-time Uncertainty Bands** – Attach confidence intervals to every ETA, so students know the likely range rather than a single-point estimate.