Stochastic Modeling of Urban Mobility in New York City

ECE 225A Final Project
Atharv Nair (A69042035), Nitin Shreyes Venkatesan (A69041917)
https://nycpublictransit.streamlit.app/

December 14, 2025

1 Introduction

New Yorkers often pick between a yellow cab and Citi Bike for medium-length trips. The choice is not always straightforward: taxis usually arrive faster, yet bikes are cheaper, greener, and sometimes more predictable in gridlocked streets. To reason about this trade-off we quantify *how much slower biking really is* across neighborhoods and times of day. Rather than mimic a full routing engine we zoom in on the probability distributions that control two door-to-door components (code: https://github.com/AtharvRN/NYC_Public_Transit):

- **Wait time.** How long until a taxi arrives or a bike becomes available in the nearby dock cluster?
- **Travel time.** Once a rider is moving, how long will the trip take as a function of distance and traffic patterns?

To answer these questions we fit probability distributions directly to the NYC Taxi and Citi Bike datasets (Jan-Jun 2024). We analyze Poisson and Negative-Binomial hourly arrival counts along with Weibull inter-arrival gaps for diagnostic purposes, but the deployed wait estimator ultimately uses the implied exponential mean because it matches the observed inter-arrival histogram. Likewise, we study Gamma cohorts to understand distance-dependent variance, but the app relies on a lognormal regression for travel minutes because it provides smooth predictions without bin boundaries. Both estimates feed a public Streamlit app that lets users click origin/destination pairs and instantly compare modes (Fig. 1).

Assumptions and design choices:

- Wait times rely on inter-arrival gaps as a proxy for rider experience.
- Travel-time samples are restricted to 1-120 minutes and within 12 km to keep Citi Bike and taxi cohorts comparable. Forecasts outside this range fall back to coarser heuristics.
- We only use features available historically (distance plus rush/weekend indicators). Weather, incidents, dynamic pricing, and subway usage are left for future work; adding subway headways is an obvious extension.

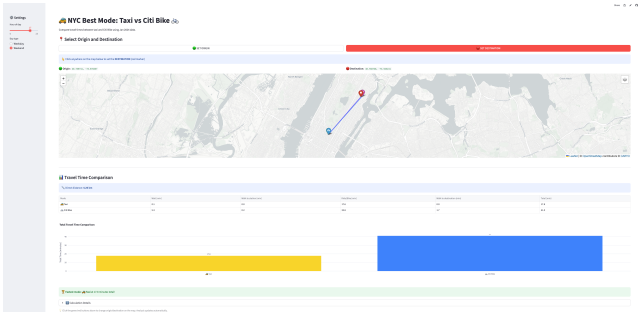


Figure 1: Public dashboard showing taxi vs Citi Bike recommendations for arbitrary O/D pairs.

2 Datasets

We use the official NYC TLC yellow taxi trip records [1] and the Citi Bike historical CSVs [2] for the first half of 2024. Both sources provide raw trip-level logs rather than aggregates, which allows us to recompute arrivals, wait gaps, and travel durations with consistent filters. Table 1 summarizes the dataset scales used in the deployed models. Subway feeds and weather data are not yet integrated; adding them is the next major item after stabilizing the taxi/bike comparisons.

Yellow taxi. Six Parquet files ('yellow_tripdata_2024-01' through '-06') contribute 17.6 million filtered trips after removing zero-distance or >120 minute rides. Each record contains pickup/dropoff timestamps, trip distance (miles), and TLC pickup/dropoff zone IDs (263 unique zones citywide). When we need coordinates for instance to compute straight-line distance between origin and destination, we use the published TLC zone centroids and convert the haversine separation into kilometers. These columns let us (i) count hourly arrivals per zone for the wait-time analysis, and (ii) compute true travel minutes while regressing on a common distance measure for both taxis and bikes. Taxi fares, passenger counts, and payment types are present in the raw files but remain unused in this iteration.

Citi Bike. The Jan-Jun 2024 CSVs add 18.3 million rides spanning 2,223 unique start stations and 2,240 end stations. Each row includes start/end timestamps, station coordinates, rideable type (classic vs electric), and membership flag. Citi Bike coverage is highly concentrated in

Table 1: Key statistics for the Jan-Jun 2024 training data.

Metric	Taxi	Citi Bike
Trips used in travel-time model	17,550,334	18,253,964
Unique pickup locations	263 TLC zones	2,336 stations
Months covered	Jan-Jun 2024	Jan-Jun 2024
Metadata	Distance, zone IDs	Station coords, rideable type
Active wait cohorts (location/hour/rush/weekend)	5,975	47,876

Manhattan and northwest Brooklyn; more than half of the rides originate in Manhattan and $\approx$ 450,000 trips begin and end at the same dock (short rebalancing hops). Station-level timestamps feed both the arrival counts and the travel-time regression.

3 Wait-Time Estimation

Our starting point was the textbook assumption that hourly arrivals can be modeled as Poisson counts [3], meaning the bucketed count N_h for a zone or station has mean and variance both equal to λ_h. This worked in the busiest taxi zones but failed for bikes: weather swings or rebalancing trucks often sent the variance far above the mean. We therefore relaxed the model to a Negative-Binomial distribution [4], which keeps the same mean λ_h but adds a dispersion term so that quieter locations are no longer forced to look Poisson. These arrival-count fits are useful for diagnostics, but the deployed wait estimator does not directly use the NB parameters.

We cannot observe actual rider waits, so we use the *inter-arrival gap* the time difference between two consecutive pickups as a proxy. This is imperfect: it assumes riders show up uniformly at random and that bikes/taxis are immediately available once a previous customer leaves. In reality, someone could arrive during a lull, or a bike dock might be empty even though the last checkout just happened. Still, the proxy is intuitive and provides a measurable quantity when actual wait data are absent.

For every zone or station we compute hourly arrival rates $\lambda_{z,h}$ split by rush vs. off-peak and weekday vs. weekend. The associated inter-arrival gaps align closely with an exponential distribution, so each bucket is summarized by its single λ parameter. Because the exponential mean equals $1/\lambda$ hours ($= 60/\lambda$ minutes) we turn every hourly arrival rate into a wait-time estimate with a single formula. At inference time the app simply looks up the relevant $\lambda_{z,h}$ bucket, converts it to minutes via $60/\lambda$, and predicts the wait time.

Figure 2 validates the shape visually. The *top-left* panel corresponds to the Financial District Citi Bike station during a weekend rush hour: the arrival histogram (blue bars) decays quickly, the Negative-Binomial curve slightly outperforms the Poisson baseline, and the exponential wait curve (green) sits on top of the empirical gaps. The *top-right* panel repeats the same story for taxi zone 3905.15 on a weekday off-peak hour, highlighting that high-throughput TLC zones exhibit the same exponential signature.

4 Travel-Time Estimation

We start by looking at simple **Gamma cohorts**. Trips are grouped by mode, 2 km distance buckets, and rush/weekend flags; the mean and variance within each group define a Gamma curve, so there is one set of parameters per cohort. These cohorts are valuable for diagnostics because they mirror Erlang-style service models widely used in queuing analysis [6], but they are not served directly in the app. When the bin spans a wide distance range the empirical histogram can keep a sharp mode near short rides while the Gamma curve spreads mass across the entire bucket, effectively mixing several sub-distributions into one and softening the peak. That limitation motivates a model that operates at the trip level instead of pre-binning.

Because we want smooth predictions for any distance, we train one **lognormal regression** over all trips (no distance or rush splits there is just one taxi model and one bike model). Lognormal regressions are widely used to describe positively skewed durations in reliability and survival analysis [7], making them a natural fit for travel minutes. A Gaussian model in raw minutes would perform poorly: travel times are strictly positive and exhibit long right tails (traffic jams, detours), so normality would assign nonzero probability to negative durations and treat outliers as symmetric deviations. The lognormal assumption keeps the lower bound at zero and stretches the upper tail more realistically. Let T denote travel minutes for a trip with distance d. We model

$$\log T = \beta_0 + \beta_1 d + \beta_2 d^2 + \beta_3 I_{\text{rush}} + \beta_4 I_{\text{weekend}} + \varepsilon,$$

where $\varepsilon \sim \mathcal{N}(0, \sigma^2)$. The design matrix is passed to `statsmodels.api.GLM`, whose iteratively reweighted least-squares solver (backed by SciPy) maximizes the Gaussian log-likelihood to obtain $\hat{\beta}$ and $\hat{\sigma}^2$. Coefficients therefore correspond to percentage changes in expected time: multiplying d by one kilometer adjusts the log mean by β_1, so the minutes scale by e^{β_1}. Table 3 shows that bikes reduce travel time by roughly 2.5% ($e^{-0.025}$), while rush-hour increases taxi time by about 6% ($e^{0.059}$).

Inference in the regression world is even simpler: we plug the requested trip's distance and rush/weekend flags

Table 2: Lognormal GLM fit metrics (log space).

Mode	Samples	$\sigma_{\log}$	$R^2_{\log}$	$\text{MAE}_{\log}$
Taxi	17,550,334	0.391	0.618	0.300
Bike	18,253,964	0.485	0.622	0.331

Table 3: Lognormal GLM parameters (log-minutes scale).

Mode	σ	β_0	β_d	β_{d^2}	β_{rush}	β_{weekend}
Taxi	0.391	1.276	0.499	−0.031	0.059	−0.106
Bike	0.485	1.218	0.702	−0.049	−0.025	0.079

into the fitted linear equation, obtain $\hat{\mu} = X\hat{\beta}$, and convert back to minutes with the lognormal mean formula $\hat{T} = \exp(\hat{\mu} + 0.5\hat{\sigma}^2)$. This correction term accounts for the asymmetry introduced by exponentiation. With only two parameter sets (taxi vs. bike), the app avoids sparse-cohort lookups yet still mirrors the Gamma diagnostics discussed earlier.

Quality metrics (log-space $R^2 \approx 0.62$ and MAE≈ 0.33 for bikes) indicate the simple feature set explains most of the variance; rush/weekend dummies only add small adjustments once distance is accounted for. Combined with Figure 2, these metrics show the lognormal regression is a better fit (especially for bikes) than the cohort baseline.

The *bottom* panels of Fig. 2 zoom in on 0-2 km taxi rides (left) and the analogous Citi Bike cohort (right). In both cases the Gamma histogram (orange) flattens the sharp mode, whereas the lognormal PDF (red) traces the empirical bars and preserves the peak—the improvement is most visible for Citi Bike, reinforcing the numerical lift reported above.

5 Website

We have deployed the website using Streamlit (https://nycpublictransit.streamlit.app/) . A typical session proceeds as follows:

1. **Select points.** Users drop origin/destination markers on the Folium base map or enter addresses via the text boxes. The app snaps each marker to the nearest taxi zone centroid and Citi Bike station.
2. **Inspect wait times.** The sidebar lists the estimated taxi wait (based on the hourly exponential mean) and Citi Bike wait for the closest station. If a cohort lacks data, the app automatically falls back to the nearest location with cached parameters.
3. **Compare travel times.** Below the wait panel, cards show the lognormal travel-time prediction for each mode and the implied total journey (wait + travel + walking).
4. **Drill into diagnostics.** A table and optional chart summarize the distributions so users can see whether rush hours or weekends materially change the recommendation.

Table 4: Out-of-sample errors (minutes) for lognormal vs. Gamma cohorts.

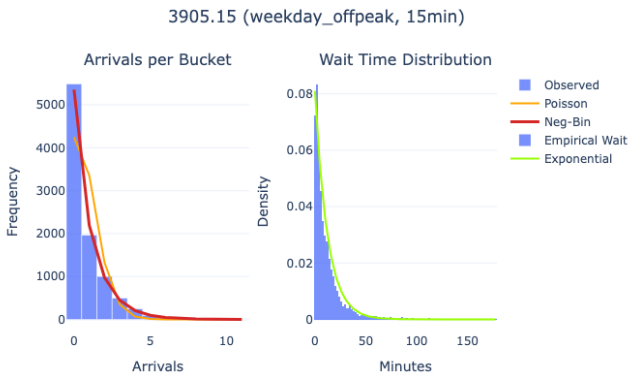
Mode	Model	MAE	RMSE
Taxi	Lognormal GLM	3.83	5.64
Taxi	Gamma cohorts	3.97	5.64
Bike	Lognormal GLM	4.40	8.51
Bike	Gamma cohorts	4.81	8.42

Figure 1 highlights the hero section (map plus summary table). Figure 3 zooms into the lower pane, where the stacked bars, winner callout, and calculation accordion reveal exactly how the underlying wait and travel components add up.

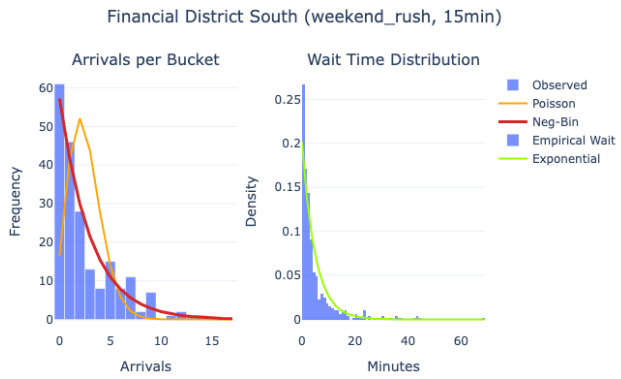
6 Limitations & Future Work

This project is purposely lightweight: it is *not* a routing engine and it does not compete with Google Maps or similar trip planners. The goal was to test whether textbook probability distributions, exponential waits and lognormal travel times, can yield useful intuition when fed with public datasets. Scaling the pipeline to citywide routing would require map matching, real-time feeds, multimodal transfers, and heavy infrastructure that are well beyond our scope. The dashboard should therefore be viewed as an explainer built on simple statistics rather than a production navigation tool.

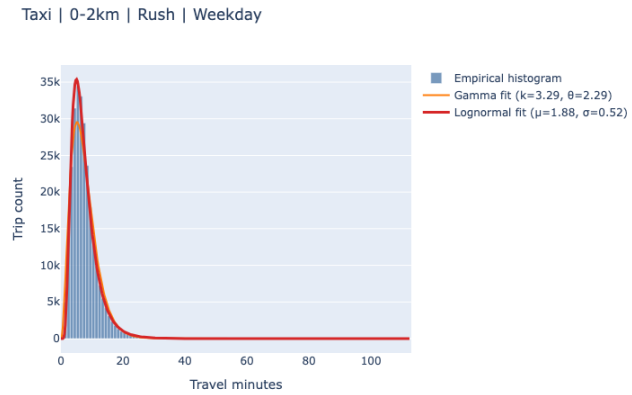
Several technical limitations remain. Wait times are derived from inter-arrival proxies, so they ignore weather, traffic, staging policies, and rider behavior. This shortcut likely inflates Citi Bike waits: the time between two checkouts does not guarantee a bike was unavailable for that entire interval, so sparse stations can exhibit artificially large means. Travel times only use haversine distance, rush-hour flags, weekend flags, and the e-bike indicator; road geometry, one-way streets, and turning delays are not modeled, which limits accuracy for longer trips. Data coverage is also narrow, only Jan-Jun 2024, and Citi Bike inventory levels or GPS traces are absent, so we cannot validate how quickly docks refill. Costs are another gap: taxi fares exist in the TLC files but were not surfaced, and Citi Bike pricing depends on membership tiers we do not track. Subway feeds (GTFS schedules, real-time headways) remain on the backlog; integrating them would make the taxi vs bike comparison more balanced but requires additional parsing and modeling. Finally, richer future work includes adding uncertainty bands around predictions, letting users upload bulk origin/destination pairs, and reporting travel-time percentiles rather than just means. These expansions would move the tool closer to operational relevance while preserving the explainable distribution-first approach.



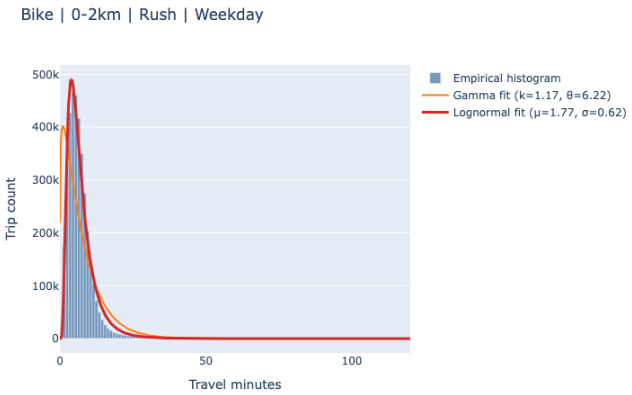
Citi Bike wait time: zone 3905.15, weekend rush



Taxi wait time: zone Financial District, weekday off-peak



Taxi travel time: 0-2 km, rush-hour weekday



Citi Bike travel time: 0-2 km, rush-hour weekday

Figure 2: Model diagnostics for a specific taxi zone and Citi Bike station. The notebook tooling can regenerate these overlays for any location; top row shows arrival/wait fits (Section 3) and bottom row compares travel time Gamma vs lognormal fits (Section 4).

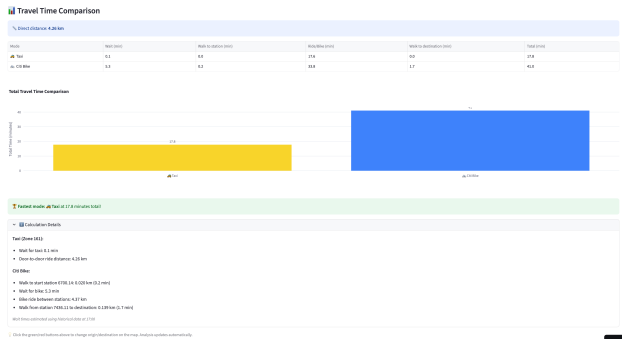


Figure 3: Detail view of the website showing total travel comparison, fastest-mode callout, and calculation breakdown for both modes.

7 Discussion

Several modeling ideas were explored but ultimately set aside in favor of the exponential/lognormal pair documented earlier. We summarize the most relevant experiments here for completeness.

Speed-based travel model. We also explored framing travel time as distance divided by a stochastic speed distribution. Trips were grouped by rush/weekend and fit with Weibull, Gamma, and lognormal densities. The Weibull model achieved the best AIC [**?**] because its hazard rate can capture the rapid increase in delays after $\approx 10\,\text{km}$, but the approach required fine-grained routing geometry to convert O/D pairs into realistic path lengths. Without that context, the speed fit collapsed to an average pace (e.g., $\approx 15\,\text{km/h}$ for Citi Bike) that was indistinguishable from our simpler distance-based regression, so we didn't pursue this further.

Loop trips and data quality. Analysis of the Citi Bike logs revealed 0.45 M rides that began and ended at the same dock, with neighborhoods such as Central Park 6 Ave contributing more than 12% of their volume to sightseeing loops. These records inflate the overall row count yet offer little signal for commuter-focused travel models because the observed duration mostly reflects leisure time rather than mobility demand. Consequently, we keep them

in the raw data (for completeness) but down-weight their influence by filtering to 1-120 minute trips and focusing on distance-based features.

Arrival Count alternatives. Before adopting the exponential inter-arrival proxy, we explored staying fully within the Poisson framework. The appeal was its closed-form waits and neat thinning/superposition properties, but the empirical histograms—especially for Citi Bike docks hit by rebalancing or weather swings—showed clear overdispersion, so the Poisson curves consistently underestimated the tails. We therefore tried a Negative-Binomial layer purely for diagnostics, then ultimately relied on the inter-arrival gaps described in Section 3.

These exploratory threads hinted at additional directions—alternative speed models, loop-trip filters, richer arrival-count fits—but pursuing them would have required more data and engineering time than we budgeted for this prototype.

8 Conclusion

We analyzed wait times and travel times for NYC taxis and Citi Bike, exported the fitted parameters, and deployed the resulting predictions in a Streamlit website. Full preprocessing scripts, notebooks, and deployment steps live in the public repository at https://github.com/AtharvRN/NYC_Public_Transit, so the entire workflow can be reproduced or extended.

9 AI Use

We used an AI coding assistant (OpenAI Codex) throughout development to draft boilerplate code, outline problem formulations, and edit this report. Each suggestion especially anything touching statistical modeling was reviewed and validated manually before inclusion, and all final code was checked into version control after local testing.

References

[1] NYC Taxi & Limousine Commission, "TLC Trip Record Data," https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page

[2] Citi Bike NYC, "Historical Trip Data," https://s3.amazonaws.com/tripdata/index.html

[3] Wikipedia, "Poisson distribution," accessed Dec. 14, 2025, https://en.wikipedia.org/wiki/Poisson_distribution

[4] Wikipedia, "Negative binomial distribution," accessed Dec. 14, 2025, https://en.wikipedia.org/wiki/Negative_binomial_distribution

[5] Wikipedia, "Gamma distribution," accessed Dec. 14, 2025, https://en.wikipedia.org/wiki/Gamma_distribution

[6] D. Gross and C. M. Harris, *Fundamentals of Queueing Theory*, Wiley, 1998.

[7] J. F. Lawless, *Statistical Models and Methods for Lifetime Data*, Wiley, 2003.