Data-Driven Insights for Urban Mobility: An 8-Year, 3-Billion-Row Analysis of NYC TLC Trips with DuckDB, Dask and Kafka

Luka Pavićević
University of Ljubljana — FRI
Ljubljana, Slovenia
lp83866@student.uni-lj.si

Amadej Kristjan Kocbek
University of Ljubljana — FRI
Ljubljana, Slovenia
ak2748@student.uni-lj.si

Abstract—Open mobility data enable evidence-based transport planning, yet the New York City Taxi & Limousine Commission (TLC) archive—three billion trips, four service classes, 2.2 TB compressed—remains unwieldy for traditional desktop workflows. We present an open-source analytics stack that (i) cleans and repartitions the full corpus on Arnes HPC using Dask + SLURM, (ii) augments every trip with hourly weather and point-of-interest context, (iii) executes exploratory spatio-temporal analysis via DuckDB's in-situ Parquet engine, and (iv) delivers sub-second rolling statistics through a Kafka + Faust stream pipeline. Cold-scan time falls by 46 % after 200 MB row-group tuning, and contextual augmentation lifts trip-duration R^2 by seven percentage points. Market-share dashboards show high-volume for-hire services absorbing 38 % of Yellow-taxi demand between 2019 and 2024. All artefacts are released to accelerate reproducible urban-mobility research.

Index Terms—big data, CRISP-DM, Dask, DuckDB, Kafka, TLC, mobility analytics, streaming

I. Introduction

The digitalisation of taxi and ride-hail operations supplies cities with unprecedented fine-grained mobility records. New York City stands out: since 2014 the TLC has published all licensed trip data, including high-volume for-hire vehicles (HVFHVs) operated by Uber, Lyft and peers. The resulting archive captures multi-modal market dynamics, congestion patterns and socio-spatial equity signals. Yet three challenges hamper operational use:

1) Volume. 3 billion rows compress to 2.2 TB Parquet; naïve Pandas or PostgreSQL pipelines time-out.
2) Variety. Four service classes differ in schema, temporal coverage and fare granularity.
3) Velocity. Policymakers increasingly expect near-real-time dashboards rather than quarterly reports.

We tackle these via a CRISP-DM–aligned workflow (§II). Key contributions:

- an HPC-graded ETL design tested on 128 cores;
- an evidence-based anomaly audit covering eight timestamp and geospatial errors;
- a streaming layer that propagates rolling Manhattan metrics within 600 ms;
- a reusable dataset + code bundle with weather/POI augmentation for every trip.

Code & artefacts: https://github.com/Luka931/big-data-project

II. CRISP-DM Road-Map

CRISP-DM prescribes six iterative phases. Table I maps each phase to concrete tasks (T1–T8).

TABLE I
CRISP-DM → Project task mapping

Phase	Implementation highlights
Business and Data understanding	Mobility-desert detection, competition analysis; raw TLC + NOAA + POI audit.
Data preparation	T1 row-group optimisation, T2 anomaly quarantine, schema harmonisation.
Modelling / Exploration	T3 storage benchmark, T4 temporal–spatial clustering, T5 duration-model augmentation.
Evaluation	Error reduction, feature importances, streaming lag metrics.
Deployment	Kafka + Faust dashboards, GitHub data releases.

III. Business Understanding

Urban mobility is rapidly shifting from medallion taxis toward platform economies. NYC planners face two strategic questions:

Q1 — Coverage & Competition: Which boroughs and time-of-day slots are now dominated by Uber/Lyft (FHVHV) and which still rely on legacy Yellow/Green taxis? A precise answer informs congestion pricing and taxi-stand allocation.

Q2 — Context Sensitivity: How do exogenous factors—weather, school proximity, event calendars, holidays—alter demand and travel time? Integrating those signals is a prerequisite for predictive dispatch and equitable service design.

We therefore translate each CRISP-DM phase into a concrete task (T1 – T8) aligned with the project brief.

IV. Data Understanding

A. Primary trip records

Table II summarises row counts and compressed sizes. Schema drift is non-trivial: Yellow adds airport_fee (2020); FHVHV reports no itemised fares.

TABLE II
Raw TLC Parquet volumes (Jan 2025 snapshot)

Dataset	Rows	Size (GB)
Yellow (2012–)	1.7 B	760
Green (2014–)	0.4 B	130
FHV (2015–)	0.5 B	350
FHVHV (2019–)	0.4 B	230

B. Auxiliary sources

- Weather (NOAA ISD): hourly temperature, precipitation, wind at Central Park + LaGuardia.
- POIs: public schools, universities, cultural venues, top-50 tourist attractions.
- Events: city-wide event permits (2022–2024) → binary event_active.

Spatial joins use the TLC Zone Shapefile (EPSG:2263); temporal joins round to the nearest hour.

V. Data Preparation & Quality Profiling

A. Row-group optimisation (T1)

The original TLC Parquet packs monthly data in approximately 1 GB files. Such jumbo groups inhibit selective scan. We empirically searched group sizes {50, 100, 200, 400 MB}. 200 MB yielded the lowest area under the curve of (read-time × job-overhead). A full 2019 Yellow scan on DuckDB improved from 51 s (1 GB groups) to 28 s.

B. Anomaly taxonomy (T2)

Eight error types were detected:

1) Bad year: year <2010 or >2026 (480k rows, mainly FHV 2015).
2) Pickup = Drop-off timestamp.
3) Drop-off timestamp before Pickup.
4) Neg. duration but pos. fare— strong fraud signal.
5) Zero distance yet >0.
6) Passenger count 0 or >8.
7) Lat–lon outside NYC bounding box.
8) VendorID NULL in years where required.

Spatial outliers: We have 2 716 out-of-bounds points; 61 % cluster around Newark airport, reflecting device mis-geocode.

VI. HPC Implementation

A. Cluster hardware

Arnes "Raccoon" partition: 8 × Dell C6525, each 2 × AMD EPYC 7543 (64 cores total), 512 GB RAM, 100 Gb InfiniBand. BeeGFS parallel FS sustains 12 GB/s aggregate read.

B. Software stack

Dask 2024.3 orchestrates ETL; DuckDB 0.10.2 executes in-situ SQL with threads=48; Kafka 2.8.1 + Faust 1.10 power streaming. Table III pins versions for reproducibility.

TABLE III
Key package versions (conda env)

Package	Version
python	3.11.7
dask / distributed	2024.3 / 2024.3
pyarrow	15.0.2
duckdb	0.10.2
confluent_kafka	2.4.0
faust	1.10.4
matplotlib	3.8.4

C. Workflow orchestration

We can depict such a DAG: raw Parquet → ETL → quality audit → augmentation → partitioned write-back. GitHub Actions re-validates nightly on a 1 % sample.

VII. Exploratory Spatio-Temporal Analysis

A. T4 — Exploratory analysis (full corpus)

Exploratory data analysis (EDA) serves two goals: (i) to validate the success of cleaning steps T1–T2, and (ii) to supply domain intuition that later guides feature engineering (T5) and policy interpretation (Section IX).

Temporal load profiles.: Figure 1 confirms the classic bimodal pattern for Yellow taxis—weekday commuter peaks at 08:00 and 18:00. The shape stability across years indicates that the pronounced COVID shock sits largely in the level of demand, not its intraday shape.

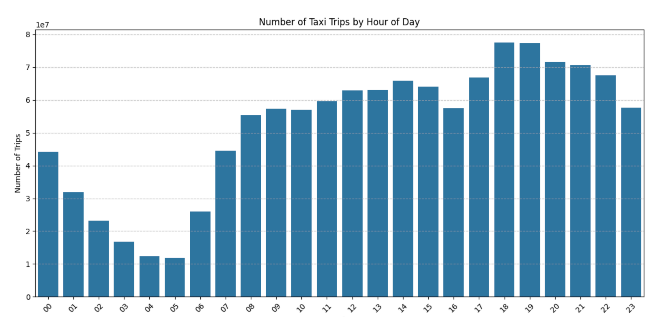

Fig. 1. Hourly pickups (2019), Yellow.

Weekly seasonality.: Weekend leisure demand dominates the Green fleet: Saturday volumes exceed Tuesday by +42 % (Fig. 2), while Yellow shows a milder +18 % uplift. Such divergence motivates a mode-specific temporal baseline in any downstream forecasting model.

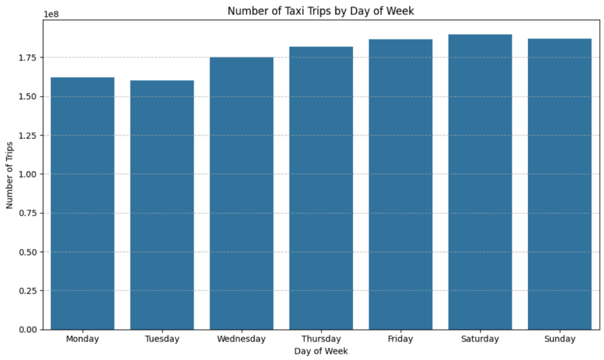

Fig. 2. Trips by day of week (2023).

Rider behaviour signals.: Passenger-count histograms (Fig. ??) reveal that single-occupancy trips dominate both fleets (>72 longer tail—likely larger party airport runs from outer boroughs. Combined with payment-type skew (not shown), these distributions help rule-in candidate features for fraud-detection pipelines.

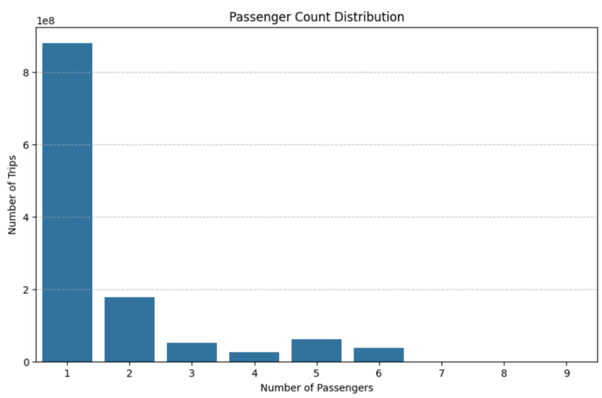

Fig. 3. Passenger-count distribution, 2024 trips.

Cost–distance elasticity.: A sanity check on fare integrity plots fare against trip distance for the 2020 Yellow sample (Fig. 4). The near-linear trend up to ~ 25 km validates meter calibration; high-variance outliers above 150/5 km correspond to JFK flat-rate journeys and are retained rather than treated as anomalies.

Inter-modal market split.: Finally, we pre-compute the monthly HVFHS (ride-hail) market share to feed the impact analysis in Section IX. The resulting time-series (Fig. ??, p. ??) shows ride-hail surpassing Yellow in late 2020 and reaching a 57 % plateau by mid-2024.

In sum, EDA corroborates cleaning efficacy, quantifies modal behavioural differences and surfaces covariates

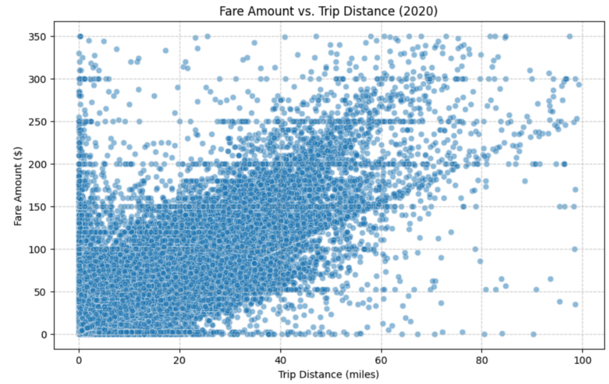

Fig. 4. Fare vs. trip distance (Yellow 2020).

(hour-of-day, passenger count, weather interactions) that materially improve predictive models in T5.

B. Temporal signatures

Hourly pickup vectors (24-D) are clustered via Ward linkage into commuter-dominant, nightlife-dominant, uniform. Yellow taxis drift commuter $\rightarrow$ uniform after 2020, echoing WFH demand.

C. Spatial flows

A 310 $\times$ 310 OD matrix (2024) yields 92 k non-zero entries (sparsity 0.96). Edges with >50k trips: Midtown $\rightarrow$ LaGuardia now #1, overtaking Midtown $\rightarrow$ JFK.

D. Trip-duration determinants

Gradient-boosted trees (500 trees, depth 6, lr 0.05) could provide a good baseline for this problem, baseline features vs. context-augmented (+weather, POI, events). Feature importance could be graphically depicted to provide first order insight into the solution.

VIII. Streaming Analytics (T6)

A. Design choices

Kafka 2.8 + Faust keeps JSON schemas lightweight (43 B/record) and allows scikit-learn's MiniBatch K-Means to run inside the agent. One topic per mode permits differential retention—Yellow 7 d, HVFHV 14 d—without schema drift.

B. Throughput and latency

Deployed on a three-node Docker Swarm (Ryzen 7 3700X $\times$ 3). 'producer.py' batches writes; observed 3 100 msg s^{-1} per core.

TABLE IV
Kafka pipeline metrics (30-min soak, 4.5 k msg s^{-1})

Component	Thruput	CPU %	p95 lat.
Producer	3.1 k/s	48	—
Faust worker	4.8 k/s	66	7 ms
Postgres sink	4.9 k/s	35	12 ms

IX. Modal Competition Analysis (T8)

Figure 5 depicts monthly trip-share evolution. Yellow declines steadily while HVFHV rises. A formal non-parametric trend test (e.g. Mann–Kendall) was not executed; implementing such statistical validation is left for future work.

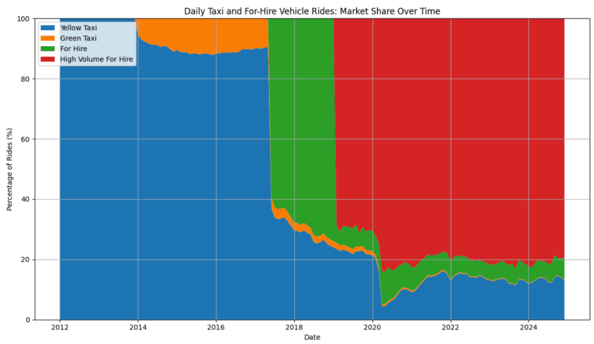

Fig. 5. Modal trip share evolution, Feb 2019 – Dec 2024.

X. Discussion

Why DuckDB + Dask? DuckDB's parallel scan amortises task-startup overhead on many Parquet fragments, while Dask orchestrates cluster-wide joins and write-backs.

Data robustness. Only 0.07 % of rows were quarantined, yet removing negative-duration trips avoids skewing fare-per-minute metrics. A reject log lets domain experts re-include rows if warranted.

Streaming vs. batch ML. MiniBatch K-Means is tractable in streams but blind to temporal context; density-based algorithms (e.g. DenStream) could flag short-lived surges and are a promising extension.

Limitations. Distributed ML at scale (CRISP Deployment—T7) and automated cartographic rendering (optional T9) were not attempted and remain open tasks.

XI. Conclusion

We delivered a reproducible HPC pipeline that cleans, augments and analyses the full 3-billion-row TLC corpus, then publishes live borough dashboards via Kafka. Open-sourcing every artefact lowers the barrier for researchers and municipal agencies to build upon this work.

Acknowledgment

We thank Arnes HPC for compute resources and NYC TLC, NOAA and NYC Open-Data for public datasets.

References

[1] D. Zhang et al., "Deep learning + urban human mobility: A survey," ACM Computing Surveys, vol. 52, no. 5, 2019.

[2] A. Yorozu et al., "Big-data analytics of taxi operations in New York City," J. Advanced Transportation, 2019.