EBA 5005 Practice Module in Specialized Predictive modelling & Forecasting

Optimizing Taxi Economics: A Data-Driven Approach to Pricing and Resource Management in NYC

Date of Report

4th May 2025

Team Name

Group 1

Team Members

Ashish Nagineni (A0297393L)

Koh Zhen Wen Ian (A0033871H)

Li Yingzhen (A0297776B)

Mohammed Muzaffar Ali(A0297946A)

Zhu Wenqing (A0269734N)

Individual Contribution

1. Ashish Nagineni (A0297393L)

Contribution	Corresponding Sections in Report
Price Optimisation: Price Elasticity	9.1.8 Demand Curve Estimation: Modelling Elasticity 9.1.8.1 Interpretation and Business Insights 9.1.8.2 Summary
Price Optimisation: Linear Programming	9.2 Linear Programming 9.2.1 Introduction 9.2.2 Technical Implementation 9.2.2.1 Defining the Objective Function 9.2.2.2 Setting the Constraints 9.2.2.3 Solving the Linear Programming Problem 9.2.3 Results and Analysis 9.2.4 Key Findings 9.2.5 Conclusion

2. Koh Zhen Wen Ian (A0033871H)

Contribution	Corresponding Sections in Report
Driver Allocation Optimisation: Time Series Order Prediction: SARIMAX *(Joint work with Wenqing)*	8.1.1.1 Limitations and considerations using SARIMAX 8.1.1.2 Data Selection 8.1.1.3 ACF and PACF plots 8.1.1.4 Stationarity Testing 8.1.1.5 SARIMAX Model Order 8.1.1.6 Exogenous Variables 8.1.1.7 Modelling Results, Forecast and Metrics 8.1.1.8 Results Summary and Discussion
Driver Allocation Optimisation: Time Series Order Prediction: Long Short-Term Memory (LSTM) Network	8.1.2 Time Series Order Prediction: Long Short-Term Memory (LSTM) Network 8.1.2.1 Preprocessing for LSTM 8.1.2.2 Sequencing for LSTM Next Hour Prediction 8.1.2.3 Training of LSTM Time Series Machine Learning 8.1.2.4 LSTM Time Series Results and Evaluation Metrics 8.1.2.5 LSTM Time Series Conclusions

3. Li Yingzhen (A0297776B)

Contribution	Corresponding Sections in Report
Driver Allocation Optimisation: Monte Carlo Simulation and Linear Programming	8.3 Monte Carlo Simulation and Linear Programming 8.3.1 Data Selection and Assumptions 8.3.2 Order Level Distribution by Hour 8.3.3 "Key-Hour" Selection 8.3.4 Order Level Distribution by Location 8.3.4.1 Distribution Characteristics 8.3.5 Order Level Distribution Monte Carlo Simulation 8.3.5.1 Simulation Design 8.3.5.2 Simulation Results and Interpretability 8.3.6 Lineal Programming – Fleet allocation optimization 8.3.6.1 Objective function 8.3.6.2 Constraints 8.3.6.3 Optimization result
Price Optimisation: Price Elasticity	9.1 Price Elasticity 9.1.1 Variable Construction 9.1.2 Initial Regression and Implications 9.1.2.1 Linear OLS Model 9.1.2.2 Log-OLS Model 9.1.2.3 ARIMAX 9.1.2.4 Model Selection 9.1.3 Grouped Regression Analysis by Hour 9.1.3.1 Regression Result 9.1.3.2 Business Implication 9.1.4 Grouped Regression Analysis by Location

4. Mohammed Muzaffar Ali (A0297946A)

Contribution	Corresponding Sections in Report
Price Optimisation: Price Elasticity	9.1.5 Borough-Wise Variation in Driver Pay. 9.1.6 Monthly Variation in Driver Pay Across Boroughs 9.1.7 Hourly Demand Segmentation via Quantile Analysis 9.1.8 Demand Curve Estimation: Modelling Elasticity 9.1.8.1 Interpretation and Business Insights 9.1.8.2 Summary 9.1.9 Hourly Revenue Sensitivity to Price Adjustments
Price Optimisation: Linear Programming	9.2.2 Technical Implementation 9.2.2.1 Defining the Objective Function 9.2.2.2 Setting the Constraints

5. Zhu Wenqing (A0269734N)

Contribution	Corresponding Sections in Report
Driver Allocation Optimisation: Time Series Order Prediction: SARIMAX *(Joint work with Ian)*	8.1.1.1 Limitations and considerations using SARIMAX 8.1.1.2 Data Selection 8.1.1.3 ACF and PACF plots 8.1.1.4 Stationarity Testing 8.1.1.5 SARIMAX Model Order 8.1.1.6 Exogenous Variables 8.1.1.7 Modelling Results, Forecast and Metrics 8.1.1.8 Results Summary and Discussion
Survival (Wait Time) Analysis	8.2 Survival (Wait Time) Analysis 8.2.1 Introduction 8.2.2 Data preparation: 8.2.3 Methodology: 8.2.4 Cox model evaluation: 8.2.5 Survival analysis Conclusion

Executive Summary

This report details a data analysis project conducted to address Uber's key business objectives: Driver Resource Optimal Allocation and Pricing Optimization.

Driver Resource Optimal Allocation: To improve supply-demand matching, optimize dispatching, and fine-tune fleet size, this project employed time series machine learning (SARIMAX and LSTM) to predict future demand by location and time. Survival analysis was used to estimate wait times and related probabilities. Monte Carlo simulation was then utilized to optimize driver allocation.

Pricing Optimization: To maximize revenue, the project determined demand elasticity using regression analysis (Linear OLS, Log-OLS, and ARIMAX). Linear programming was then applied to optimize pricing strategies.

Key deliverables include models and analyses that provide actionable insights for Uber to:

- Enhance the efficiency of driver allocation.
- Reduce customer wait times.
- Optimize fleet size.
- Strategically adjust pricing to increase revenue.

The results of this project offer data-driven recommendations to enable Uber to improve its operational efficiency and financial performance.

Table of Contents

List of Figures

List of Tables

1. Introduction & Background

The taxi industry has undergone significant transformation in recent years due to the rise of rideshare services and evolving consumer preferences. The NYC Taxi and Limousine Commission (TLC) has engaged us as consultants to assess the current state of the taxi industry and recommend data-driven solutions to address the emerging challenges.

This project addresses the challenges of unoptimized pricing and driver allocation, leading to demand-supply imbalances. We aim to provide a comprehensive overview of the current state and future trajectory of New York City's taxi industry, while also offering data-driven solutions to optimize supply-demand dynamics, improve operational efficiency, and enhance the overall customer experience.

New York City's taxi industry is a complex system comprising yellow medallion taxis, green borough taxis, and the dominant for-hire vehicle (FHV) sector, primarily rideshare companies like Uber and Lyft. Each segment plays a distinct role and faces unique challenges.

The industry faces several key challenges. The rise of FHVs has created intense competition, leading to a sharp decline in yellow cab ridership and a collapse in medallion values, triggering a medallion debt crisis for many owners. The COVID-19 pandemic further compounded these issues, drastically reducing ridership and revenue, forcing some drivers into delivery services.

The NYC taxi industry has undergone a dramatic transformation. While challenges remain, the industry is adapting through innovation, technology, and policy adjustments. The future will likely involve a mix of traditional taxis, rideshare services, and other options, creating a dynamic urban transportation ecosystem.

2. Problem Statement

- Inefficiencies in supply-demand matching in NYC taxi and private for hire vehicle industry
- Unoptimized pricing and driver allocation causes demand-supply imbalances.
- Demand elasticity aids dynamic pricing, while forecasting and simulation can improve fleet
- distribution and wait times.

3. Business and Technical Objectives

UBER has engaged us to conduct data analysis in support of the two business objectives outlined below.

A. Driver Resource Optimal Allocation:

Business Objectives:

- Improve Supply-Demand Matching: Better matching of car availability with predicted demand, minimizing wait times for customers and reducing the number of idle taxis on the road.
- Optimized Dispatching: Streamlining fleet dispatch based on demand forecasts, ensuring more efficient routes and minimizing unnecessary taxi congestion in low-demand areas.
- Fleet Size Optimization: Fine-tune the total number of cars and drivers in the fleet to match demand, ensuring optimal resource utilization and minimizing inefficiencies.

Technical Objectives:

- Use time series machine learning algorithms to predict future demand, thus allowing us to allocate resources for demand based on location and time.
- Perform survival analysis to estimate the wait times and related probabilities. This allows us to understand current situation and improve upon it.
- Finally, we will perform optimisation and simulation. By utilizing Monte Carlo simulation, we can optimise the driver allocation based on time and location.

B. Pricing Optimisation:

Business Objectives:

- Increase Revenue: By strategically adjusting prices based on demand elasticity and market dynamics, we aim to maximize overall revenue generation.

Technical Objectives:

- Find the demand elasticity to understand how price impacts demand.
- Perform linear programming to optimize the price for maximum revenue.

4. Project Design

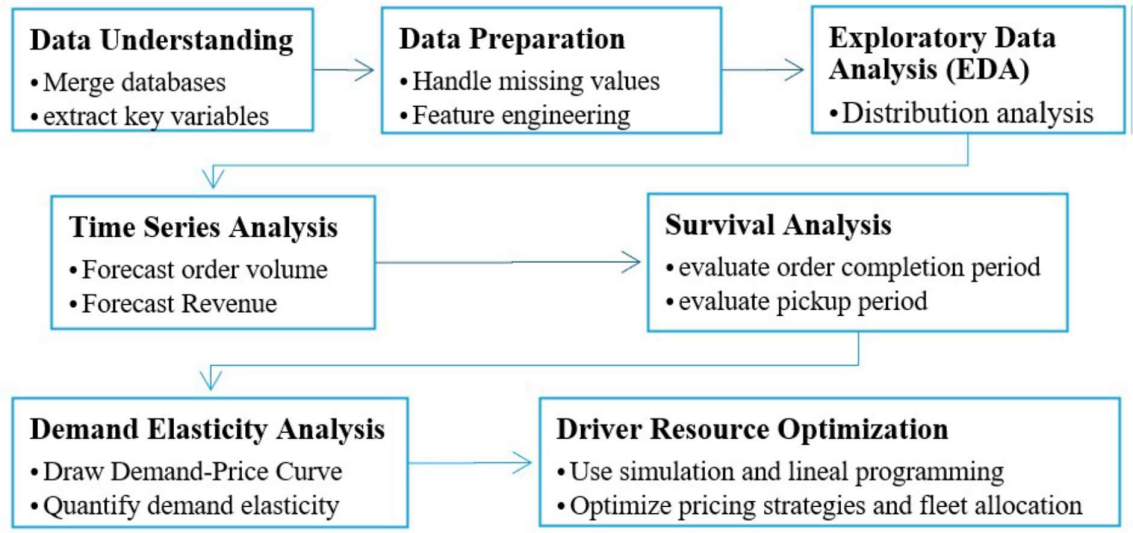


Figure 1 Project Design

5. Scope of Work

A. Driver Allocation Optimisation

Enhancing efficiency and profitability through demand forecasting, wait time prediction, and optimized driver allocation using linear programming and Monte Carlo simulation.

Methodology

1) Forecasting Order Volume, Demand & Taxi Revenue

- **Time Series Forecasting:** Utilizing statistical models like ARIMAX and LSTM networks to predict future demand patterns based on historical data.

✓ **Evaluation criteria:** Accuracy metrics (e.g. MSE, RMSE).

2) Wait time Prediction

- **Survival analysis:** Utilize Kaplan-Meier estimation, Cox Proportional Hazards Model, and Aalen's Additive Model models to predict time to order completion and evaluate influencing factors. Estimate the probability of an order being completed within a given time frame and identifying risks leading to delays.

✓ **Evaluation criteria:** Accuracy metrics (e.g. C-Index).

3) Optimisation and Simulation

- **Simulation**: Use Monte Carlo simulation to model passenger demand across regions, accounting for other factors (demand forecasting, order time predictions and driver locations).
- **Linear Programming:** Optimize driver allocation through linear programming based on regional demand fluctuations and fleets constraints to maximize profit.

✓ **Evaluation criteria:** % increase in revenue, minimised wait time after optimization.

B. Pricing Optimisation

To enhance revenue generation, a robust pricing optimization model will be developed. This model will leverage a dual analytical approach to achieve its objectives.

Methodology

The pricing optimization strategy will be built on two key analytical techniques:

- **Demand Elasticity:** Analyse the relationship between order volume and price per kilometre to understand how changes in price impact demand.
- **Linear Programming:** Optimize the taxi price, maximising revenue.

✓ **Evaluation criteria:** Expected increase in revenue in across board and per geographical zone after price optimization

6. Data Set Used & Data Description

The dataset used in this project is sourced from NYC government website. The datasets were collected and provided to the NYC Taxi and Limousine Commission (TLC) by technology providers authorized under the Taxicab & Livery Passenger Enhancement Programs (TPEP/LPEP). [1]

The dataset provides information about taxi and private for hire trips, with each entry containing price, time and location data of each taxi ride. Dataset is large (~4GB per year), historical and structured.

The high volume private for hire dataset was utilized for this project and filtered for only UBER.

Below is the data dictionary of all the relevant fields that were used in this project.

Variable	Description	Type
Pickup_datetime	The date and time of the trip pick-up	datetime
DropOff_datetime	The date and time of the trip drop-off	datetime
PULocationID	TLC Taxi Zone in which the trip began	int32
DOLocationID	TLC Taxi Zone in which the trip ended	int32
request_datetime	date/time when passenger requested to be picked up	datetime64
on_scene_datetime	date/time when driver arrived at the pick-up location (Accessible Vehicles-only)	datetime64
trip_miles	total miles for passenger trip	float64
trip_time	total time in seconds for passenger trip	int64
driver_pay	total driver pay (not including tolls or tips and net of commission, surcharges, or taxes)	float64

Table 1 TLC Private for hire Data Table

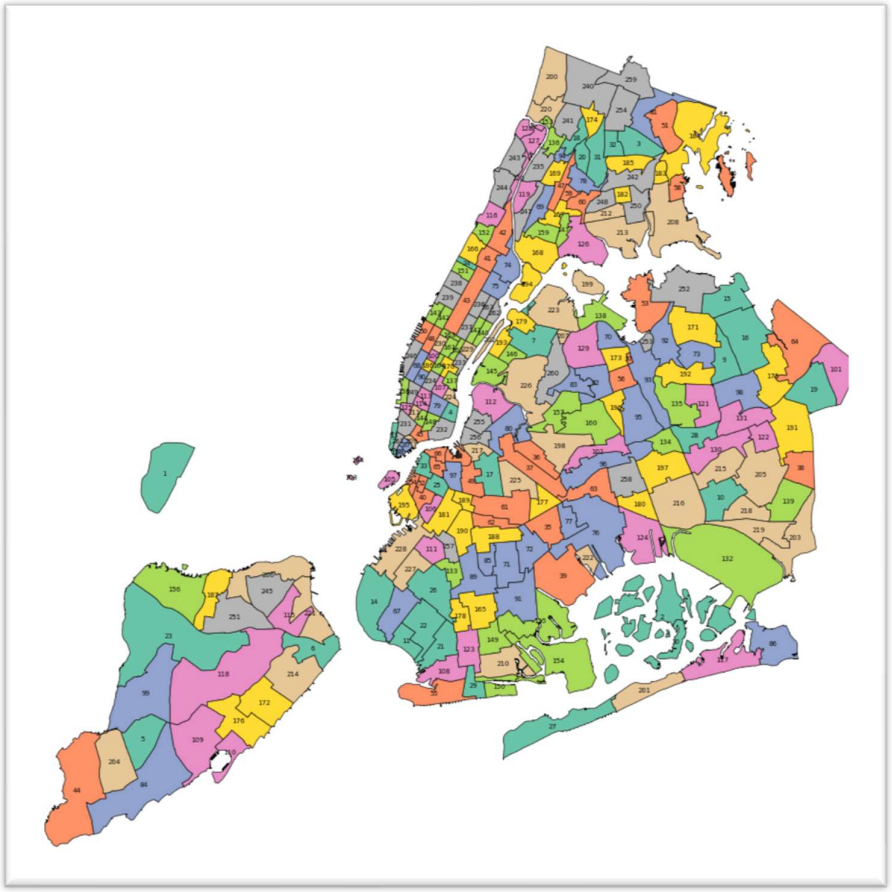


Figure 2 TLC Taxi Zoning map (LocationID)

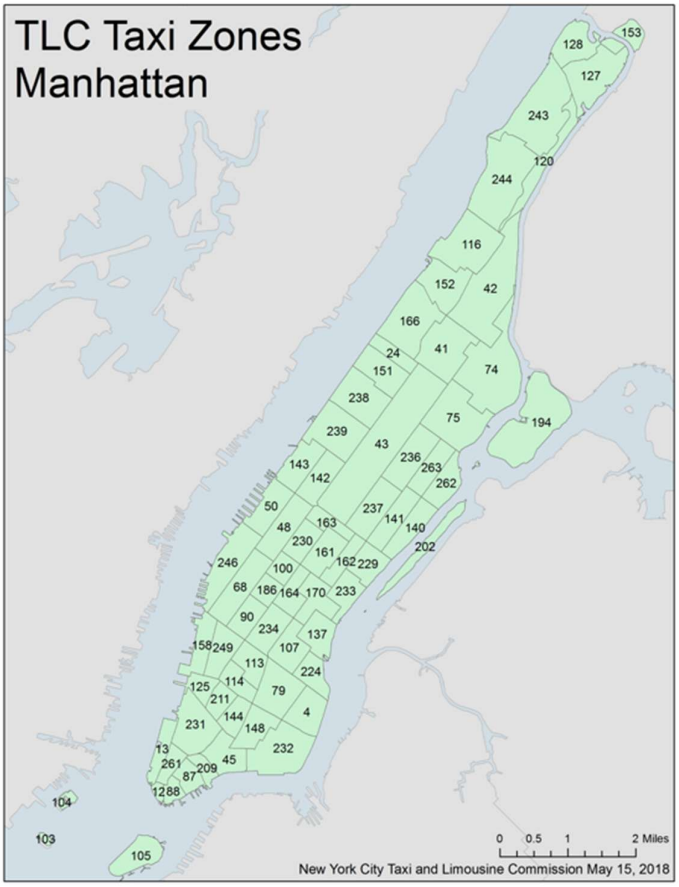


Figure 3TLC Taxi Zones (LocationID) in Manhattan

A lookup table containing Borough and LocationID data was also retrieved from TLC.

Variable	Description	Type
LocationID	TLC Taxi Zone	int32
Borough	5 Boroughs of NYC	string

Table 2 TLC Taxi Zones and Borough Lookup Table

7. Data Understanding & Data Preprocessing

This study uses large-scale trip data from New York City taxis for 2023 and 2024. The raw data, stored monthly in Parquet format, includes hundreds of files and reaches terabyte scale. To process it efficiently, we used the Dask framework with the PyArrow engine for fast, parallel computing.

7.1 Variable Overview and Structure Exploration

The original data contains key fields such as pickup time (pickup_datetime), driver earnings (driver_pay), trip distance (trip_miles), pickup location (PULocationID), and additional charges (e.g., tolls, congestion surcharge, airport fees). After schema validation and type checking, we standardized the time column to an hourly granularity (datetime_hour) to support hour-level aggregation and modelling.

```
hvfhs_license_num       string[pyarrow]
dispatching_base_num    string[pyarrow]
originating_base_num    string[pyarrow]
request_datetime         datetime64[us]
on_scene_datetime        datetime64[us]
pickup_datetime          datetime64[us]
dropoff_datetime         datetime64[us]
PULocationID                     int32
DOLocationID                     int32
trip_miles                     float64
trip_time                        int64
base_passenger_fare            float64
tolls                          float64
bcf                            float64
sales_tax                      float64
congestion_surcharge           float64
airport_fee                    float64
tips                           float64
driver_pay                     float64
shared_request_flag     string[pyarrow]
shared_match_flag       string[pyarrow]
access_a_ride_flag      string[pyarrow]
wav_request_flag        string[pyarrow]
wav_match_flag          string[pyarrow]
dtype: object
```

Figure 4 Data Type Summary

7.2 Aggregation Logic

Data was aggregated along two main dimensions:

1. **Hour Dimension**: All trips were rounded down to the nearest hour to create datetime_hour;
2. **Location Dimension**: Each record was grouped by its pickup location (PULocationID), and an additional variable trip_miles_type was derived to classify trips by distance (e.g., short, medium, long).

Based on our modelling needs, we computed the following metrics:

- **Order Count**: Number of trips per hour and pickup location;
- **Earnings Variables**: Including driver_pay, base_passenger_fare, tips, congestion_surcharge, etc.;
- **Operational Metrics**: Including trip_miles and trip_time;

All aggregations were performed using Dask's distributed compute engine and exported as structured CSV files for downstream regression and visualization.

7.3 Data Cleaning

To ensure stability and economic interpretability in the modelling stage, all continuous variables underwent sanity checks, winsorization to mitigate the impact of outliers, and removal of invalid or missing values. Through this structured preprocessing pipeline, we built a clean, scalable, and reproducible data foundation for the subsequent elasticity estimation and resource optimization tasks.

8. Driver Allocation Optimisation

The following models were built in this section:

1. Time Series models
2. Survival (wait) Time models
3. Monte Carlo and Linear Programming

The scope of all the analysis done in this section was limited to the locations within Manhattan. By limiting to 1 borough, we can shorten the time taken for the models to be built and provide higher accuracy.

Manhattan was chosen as it was the busiest borough in NYC and represented the most complex dataset among the 5 Borough. Modelling for the other 4 Boroughs subsequently would then prove to be a minor task.

8.1 Time Series Order Prediction

Time series prediction is the first step in understanding and forecasting the dynamic nature of ride demand. By examining historical order data, we can uncover critical patterns such as daily and weekly fluctuations, the impact of specific events or holidays, and longer-term growth trends. The insights derived from these predictions helps the client to optimize fleet allocation, managing driver availability, and proactively addressing periods of high demand, ultimately enhancing operational efficiency and improving user experience.

8.1.1 Time Series Order Prediction: SARIMAX

SARIMAX stands for Seasonal Autoregressive Integrated Moving Average with Exogenous Regressors. It's a powerful statistical method used for analysing and forecasting time series data that exhibits both seasonal patterns and is influenced by external factors (exogenous variables).

SARIMAX extends the ARIMA model to handle seasonality directly and incorporates the impact of other relevant time series on the variable being forecasted.

8.1.1.1 Limitations and considerations using SARIMAX

SARIMAX algorithm can only be applied to a single time series at a time. It could be used on a consolidated order count of the whole of NYC or a consolidated order of each borough. However, that was deemed not as useful to the client. Thus, a single location needs to be chosen to test out the predictive accuracy of SARIMAX.

8.1.1.2 Data Selection

1. Consolidate demand per hour per PU location (2023 -2024 data)

This is as detailed in the earlier section. The consolidated demand per hour per PU location was used.

2. Selection of Location

The data was then grouped by the PULocationID, and the total number of orders for each PULocationID was calculated. The location with the highest order count was selected for modelling.

PULocationID	
138	9113542
132	8729054
79	6476941
61	6172485
230	5855430
161	5717682
231	5474438
68	5386163
246	5169061
164	5123424

Top 10 locations by total order count

PULocationID 138 has the highest order count. Below is the times series of the order count for this location.

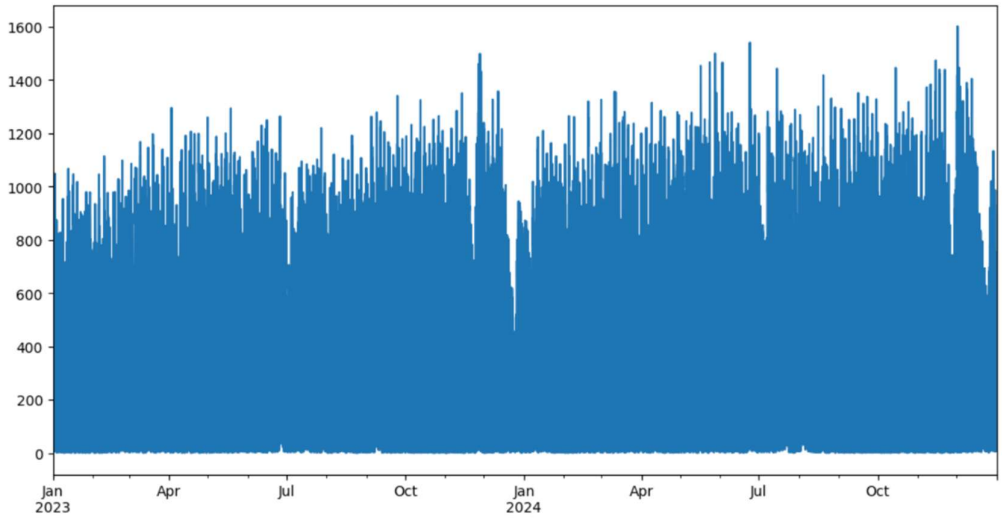


Figure 5 Order Count for Loc_138

8.1.1.3 ACF and PACF plots

The ACF (Autocorrelation Function) and PACF (Partial Autocorrelation Function) were also plotted to identify patterns and relationships within a time series data.

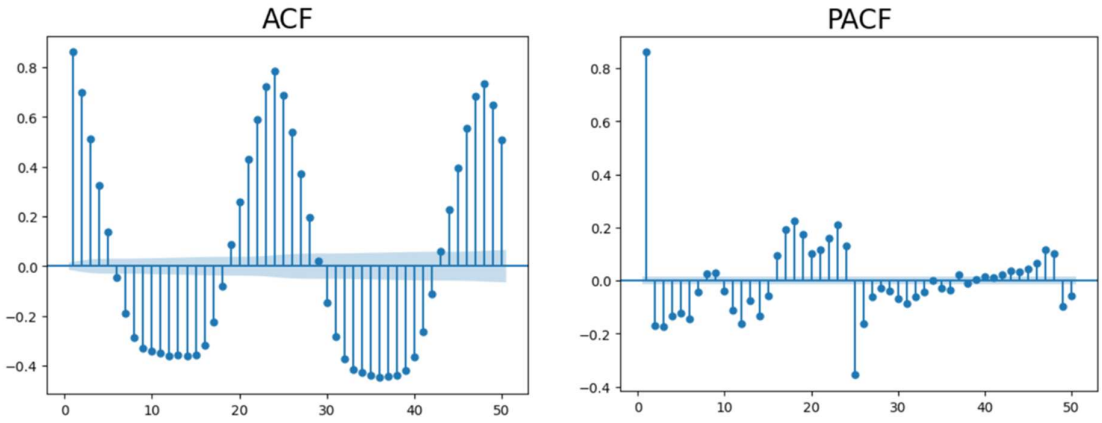


Figure 6 ACF and PACF plots

From ACF plot:

1. Gradual decay, indictive of a MA(1) process.
2. Repeating structure every 24 lags, cycle is every 24 hours or daily, indicting 24 seasonal periods
3. Has strong spikes at seasonal lags → seasonality not stationary → D=1 likely needed for seasonality.
4. Seasonal ACF also shows decay at multiples of 24, indicating seasonal MA(1) structure.

From PACF plot:

1. The PACF plot shows a strong spike at lag 1, and then a rapid drop-off, indicating AR(1) behaviour.
2. The PACF plot shows a strong spike at lag 24, and then a rapid drop-off, indicating seasonal AR(1) behaviour.

8.1.1.4 Stationarity Testing

An Augmented Dickey-Fuller (ADF) test was performed to check for stationarity.

- **ADF p-value**: 2.846e-30 → Stationary, thus no differencing is needed to be performed (d=0)

```
ADF Statistic: -17.947613926209424
p-value: 2.8460760567527773e-30
```

Figure 7 ADF test results

A second ADF test was also performed on seasonal differenced data as shown below:

```
# Seasonal differencing: subtract value from its value 24 steps ago
seasonal_diff = top_loc_df['order_count'] - top_loc_df['order_count'].shift(24)
# Drop NaNs caused by shifting
seasonal_diff = seasonal_diff.dropna()
# Apply ADF test to the seasonally differenced data
stationarity_24 = adfuller(seasonal_diff)
print(f"Augmented Dickey Fuller p-value: {stationarity_24[1]}")
```

```
Augmented Dickey Fuller p-value: 0.0
```

Figure 8 ADF test results for Seasonal differenced data

- **ADF p-value**: 0 → Stationary after differencing, thus seasonal differencing will be beneficial (D=1)

8.1.1.5 SARIMAX Model Order

Thus, the following model order was chosen:

Parameter	Value
p (AR term)	1
d (Differencing)	0
q (MA term)	1
P (Seasonal AR)	1
D (Seasonal differencing)	1
Q (Seasonal MA)	1
s (Seasonal period)	24

Table 3 Sarimax Model Order

8.1.1.6 Exogenous Variables

The following variables were also added as exogenous variables.

1. From datetime_hour time variable:
   - hour
   - dayofweek
   - is_weekend
   - month
   - day
2. From externally sourced holiday data of NYC
   - is_holiday

8.1.1.7 Modelling Results, Forecast and Metrics

Before fitting to the various models, a train test split of 80:20 was done. The model summaries are given below, together with a plot. The plot shows the last 5 days of the train set and the first 30 days of the test set. MSE and RMSE were also calculated.

1. SARIMA Model

First, we build a model without the exogenous variables. The model order parameters are as shown above. As can be seen, all the p-values are < 0.05.

SARIMAX Results

Dep. Variable:	order_count	No. Observations:	13987
Model:	SARIMAX(1, 0, 1)x(1, 1, 1, 24)	Log Likelihood	-88358.160
Date:	Thu, 01 May 2025	AIC	176726.321
Time:	13:25:59	BIC	176764.042
Sample:	0	HQIC	176738.880
	- 13987		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.8194	0.005	151.183	0.000	0.809	0.830
ma.L1	-0.2231	0.009	-25.089	0.000	-0.241	-0.206
ar.S.L24	0.1197	0.008	14.468	0.000	0.103	0.136
ma.S.L24	-0.7917	0.005	-144.670	0.000	-0.802	-0.781
sigma2	1.834e+04	145.861	125.726	0.000	1.81e+04	1.86e+04

Ljung-Box (L1) (Q):	0.00	Jarque-Bera (JB):	4217.70
Prob(Q):	0.95	Prob(JB):	0.00
Heteroskedasticity (H):	1.22	Skew:	-0.26
Prob(H) (two-sided):	0.00	Kurtosis:	5.64

Figure 9 SARIMA Model Summary

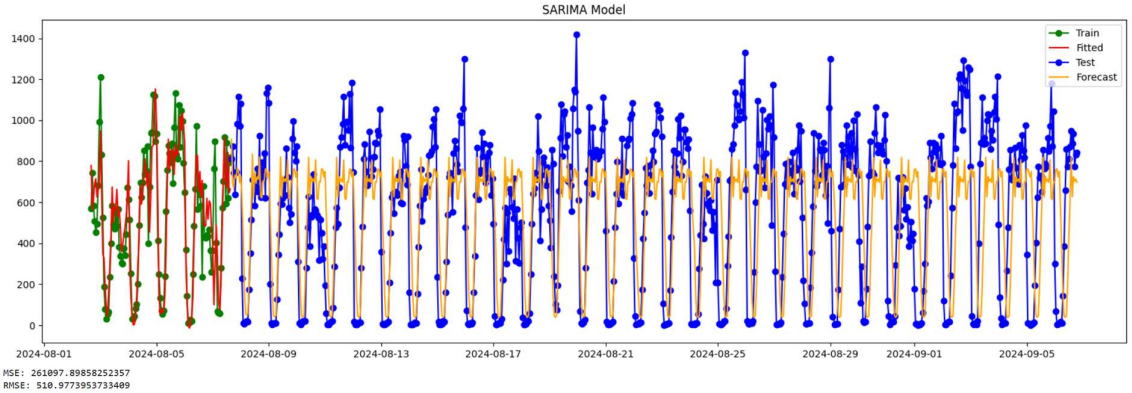


Figure 10 SARIMA Model Plot, MSE/ RMSE

2. SARIMAX Model

SARIMAX model was built next, adding the exogenous variables to the existing model. After some iterations, “is_holiday” and “month” data were dropped as their p-values were > 0.05.

SARIMAX Results

Dep. Variable:	order_count	No. Observations:	13987
Model:	SARIMAX(1, 0, 1)x(1, 1, 1, 24)	Log Likelihood	-88172.176
Date:	Thu, 01 May 2025	AIC	176362.352
Time:	13:29:43	BIC	176430.249
Sample:	0	HQIC	176384.959
	- 13987		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
hour	13.4225	0.263	51.103	0.000	12.908	13.937
dayofweek	28.6315	1.574	18.195	0.000	25.547	31.716
is_weekend	-233.2252	6.895	-33.827	0.000	-246.738	-219.712
day	1.7253	0.426	4.049	0.000	0.890	2.560
ar.L1	0.7916	0.006	127.793	0.000	0.779	0.804
ma.L1	-0.2094	0.009	-22.545	0.000	-0.228	-0.191
ar.S.L24	0.1415	0.008	17.920	0.000	0.126	0.157
ma.S.L24	-0.8302	0.005	-169.039	0.000	-0.840	-0.821
sigma2	1.766e+04	147.693	119.588	0.000	1.74e+04	1.8e+04

Ljung-Box (L1) (Q):	0.08	Jarque-Bera (JB):	4237.65
Prob(Q):	0.77	Prob(JB):	0.00
Heteroskedasticity (H):	1.21	Skew:	-0.04
Prob(H) (two-sided):	0.00	Kurtosis:	5.70

Figure 11 SARIMAX Model Summary

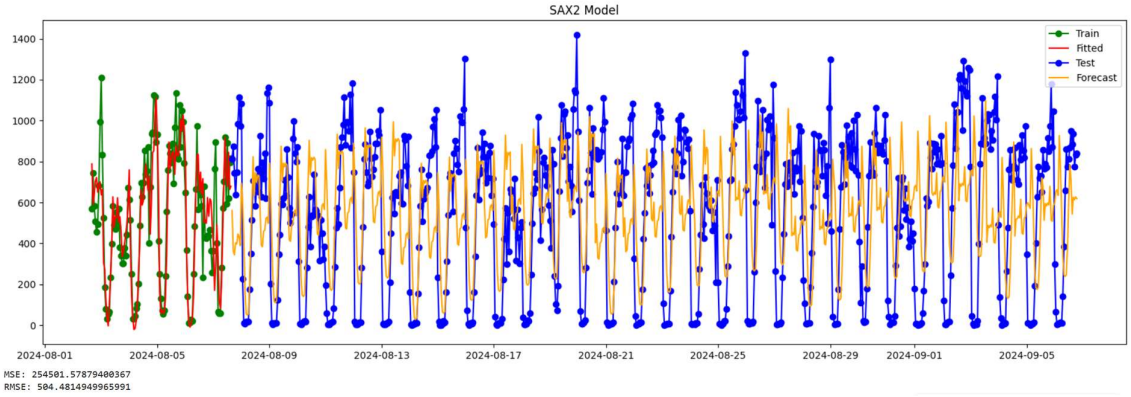


Figure 12 SARIMAX Model Plot, MSE/ RMSE

3. Benchmark Model

Finally, a benchmark model was done for comparison. The order count values were shifted one hour forward. This represents using the last known value as the prediction.

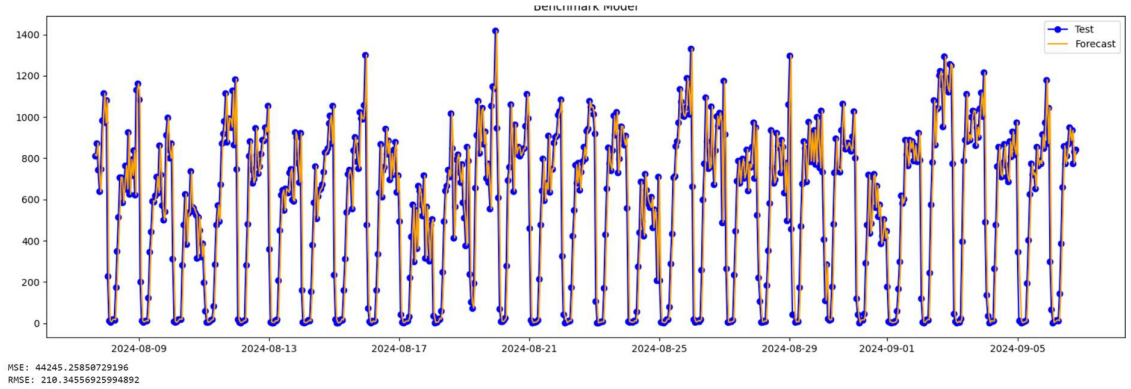


Figure 13 Benchmark Plot, MSE/ RMSE

8.1.1.8 Results Summary and Discussion

	SARIMA	SARIMAX	Benchmark
AIC	176726	176362	N/A
BIC	176764	176430	N/A
MSE	261098	254502	44245
RMSE	511	504	210

Table 4 Models Metric Summary

As seen from the results, the SARIMAX model did marginally better than the SARIMA, but still performed significantly worse than the simple benchmark's performance.

Furthermore, the SARIMAX model is limited to a single time series at a time. It is unable to generalize to all the numerous pickup locations.

Thus, the SARIMAX model is unacceptable for our use case and a more powerful predictive time series model is necessary.

8.1.2 Time Series Order Prediction: Long Short-Term Memory (LSTM) Network

LSTM is a type of Recurrent Neural Network and is well suited for time series predictions. For this section, the scope of the predictions was kept to all the PU locations in Manhattan Borough. The same process can be repeated for the other 4 Boroughs in New York.

A total of 2 models were developed, one that predicts for the all the locations orders in the next hour and one that predicts for 24 hours later

Understanding future order demand through this analysis will allow for proactive fleet allocation.

8.1.2.1 Preprocessing for LSTM

Preprocessing is the first step in preparing data for Time Series Modelling. It ensures the right feature are selected, feature engineer is performed, and data is scaled to the right format and range for the time series machine learning algorithms.

Below details the steps performed:

1. Consolidate demand per hour per PU location (2023 -2024 data)

This is as detailed in the earlier section. The consolidated demand per hour per PU location was used.

2. Filter for only PU locations within Manhattan Borough

As mentioned above this is done to reduce the scope and number of locations predicted by the LSTM algorithm. This will improve the algorithm's predictive powers. The same process described in this section can be done for the other 4 Boroughs.

3. Apply MinMaxScalar (0,1) to the order_count (demand)

datetime_hour	PULocationID	order_count	LocationID	Borough	order_count_mm
2023-01-01 00:00:00	4	190.0	4	Manhattan	0.067592
2023-01-01 00:00:00	12	9.0	12	Manhattan	0.003202
2023-01-01 00:00:00	13	133.0	13	Manhattan	0.047314
2023-01-01 00:00:00	24	87.0	24	Manhattan	0.030950
2023-01-01 00:00:00	41	343.0	41	Manhattan	0.122021
...	...	...	...	...	...
2024-12-31 23:00:00	246	264.0	246	Manhattan	0.093917
2024-12-31 23:00:00	249	418.0	249	Manhattan	0.148702
2024-12-31 23:00:00	261	111.0	261	Manhattan	0.039488
2024-12-31 23:00:00	262	109.0	262	Manhattan	0.038776
2024-12-31 23:00:00	263	255.0	263	Manhattan	0.090715

Figure 14 MinMaxScalar on order_count

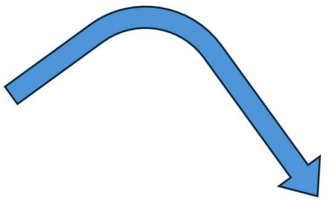
This is necessary for LSTM input as it ensures:

- Equal Contribution of Features
- Gradient Stability and Faster Convergence
- Matching to Input range to all Activation Functions

4. Pivot the table so that the columns are the locations and the values in the table are the order count.

datetime_hour	PULocationID	order_count	LocationID	Borough	order_count_mm
2023-01-01 00:00:00	4	190.0	4	Manhattan	0.067592
2023-01-01 00:00:00	12	9.0	12	Manhattan	0.003202
2023-01-01 00:00:00	13	133.0	13	Manhattan	0.047314
2023-01-01 00:00:00	24	87.0	24	Manhattan	0.030950
2023-01-01 00:00:00	41	343.0	41	Manhattan	0.122021
...	...	...	...	...	...
2024-12-31 23:00:00	246	264.0	246	Manhattan	0.093917
2024-12-31 23:00:00	249	418.0	249	Manhattan	0.148702
2024-12-31 23:00:00	261	111.0	261	Manhattan	0.039488
2024-12-31 23:00:00	262	109.0	262	Manhattan	0.038776
2024-12-31 23:00:00	263	255.0	263	Manhattan	0.090715

1175448 rows × 6 columns



17544 rows × 73 columns

datetime_hour	4	12	13	24	41	42	43	45	48	...	246	249	261	262	263
2023-01-01 00:00:00	0.067592	0.003202	0.047314	0.030950	0.122021	0.117752	0.049449	0.046603	0.170402	...	0.175738	0.221274	0.034863	0.073995	0.119175
2023-01-01 01:00:00	0.070793	0.001067	0.037353	0.023835	0.130203	0.138385	0.039488	0.050516	0.203131	...	0.210601	0.168623	0.044824	0.068659	0.134116
2023-01-01 02:00:00	0.063678	0.000356	0.020633	0.023123	0.110281	0.159730	0.018143	0.051939	0.210245	...	0.187834	0.179651	0.058342	0.048026	0.123088
2023-01-01 03:00:00	0.074351	0.000000	0.017432	0.022056	0.097474	0.157595	0.007471	0.064034	0.195304	...	0.119530	0.163999	0.049449	0.033440	0.083956
2023-01-01 04:00:00	0.052650	0.000356	0.009249	0.016009	0.069370	0.104233	0.004980	0.037353	0.154749	...	0.104233	0.086802	0.040555	0.021345	0.043401

Figure 15 Pivoting the table

By pivoting the table this way, multiple time series can be fed into the LSTM at one time, training for multiple locations simultaneously. This saves time in the training process, and the algorithm can also learn interaction effects between the locations in Manhattan, should they exist.

5. Perform Feature Engineering by

    1. Creating new features from existing ones (hour, day_of_week, is_weekend and month)
    2. Adding exogenous variables (is_holiday)
    3. Perform MinMaxScalar on hour, day_of_week and month

Feature Engineering

datetime_hour	4	12	13	24	41	42	43	45	48	...	246	249	261	262	263	is_weekend	is_holiday	hour	day	month
2023-01-01 00:00:00	0.067592	0.003202	0.047314	0.030950	0.122021	0.117752	0.049449	0.046603	0.170402	...	0.175738	0.221274	0.034863	0.073995	0.119175	1	0	0.000000	1.000000	0.0
2023-01-01 01:00:00	0.070793	0.001067	0.037353	0.023835	0.130203	0.138385	0.039488	0.050516	0.203131	...	0.210601	0.168623	0.044824	0.068659	0.134116	1	0	0.043478	1.000000	0.0
2023-01-01 02:00:00	0.063678	0.000356	0.020633	0.023123	0.110281	0.159730	0.018143	0.051939	0.210245	...	0.187834	0.179651	0.058342	0.048026	0.123088	1	0	0.086957	1.000000	0.0
2023-01-01 03:00:00	0.074351	0.000000	0.017432	0.022056	0.097474	0.157595	0.007471	0.064034	0.195304	...	0.119530	0.163999	0.049449	0.033440	0.083956	1	0	0.130435	1.000000	0.0
2023-01-01 04:00:00	0.052650	0.000356	0.009249	0.016009	0.069370	0.104233	0.004980	0.037353	0.154749	...	0.104233	0.086802	0.040555	0.021345	0.043401	1	0	0.173913	1.000000	0.0

Figure 16 Feature Engineering

Feature engineering provides the algorithm with more information to learn from. To create is_holiday column, public holiday of NYC over 2023 and 2024, was gathered from the web and merged to the table. Finally these columns were also normalised.

8.1.2.2 Sequencing for LSTM Next Hour Prediction

Sequencing is an important step in the preparation for training a Time Series Machine Learning algorithm. It would determine the length of the time series used to do predictions, as well as the target prediction time.

To prepare for the training of next hour as well as next day predictions, 2 sequencing methodologies were utilized.

The diagrams illustrate clearly how the sequencing was performed for both cases.

1. Next hour Order Count Prediction

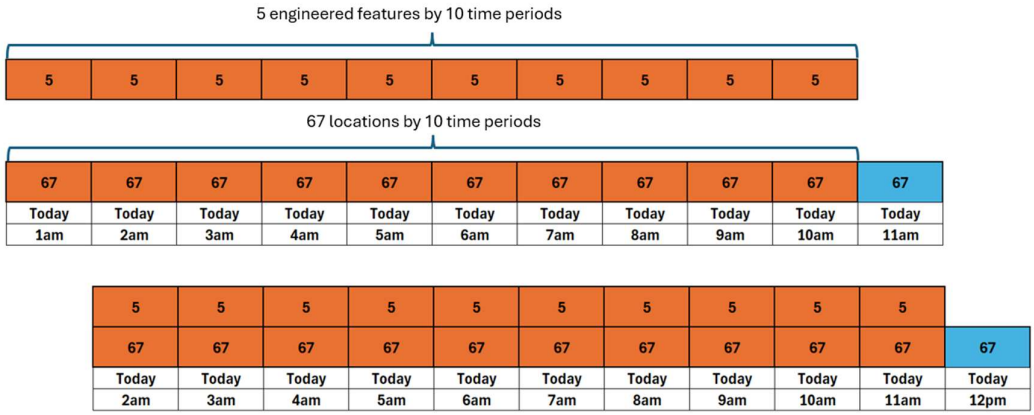


Figure 17 Sequencing for next hour prediction

Input

- Sequence of 10 hourly data of demand of each location (10 x 67)
- Feature engineered variables at each time (10 x 5)
- Total input (10 x 72)

Output

- Predicted demand for each of the 67 locations in Manhattan 1 hour later

2. Next hour Order Count Prediction

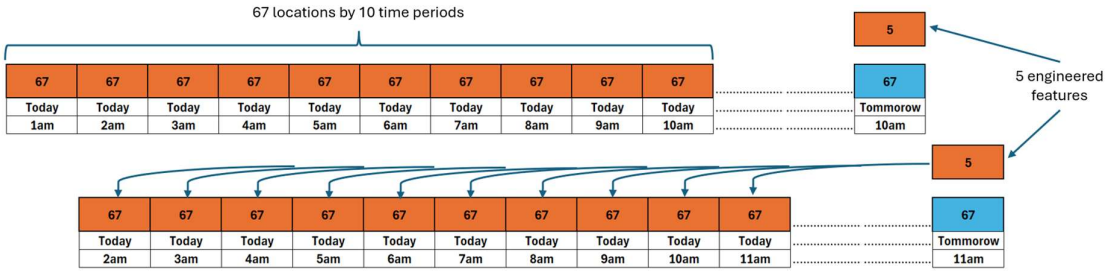


Figure 18 Sequencing for next day prediction

Input

- Sequence of 10 hourly data of demand of each location (10 x 67)
- Feature engineered variables at prediction target time 24 hours later (10 x 5)

- Total input (10 x 72)

Output

- Predicted demand for each of the 67 locations in Manhattan *24 hours later*

Finally, for both sequencing processes, a 80-20 train test split with no shuffle was performed.

8.1.2.3 Training of LSTM Time Series Machine Learning

Below, the architecture and parameters used for the LSTM model is shown.

```
model_f = Sequential([
    LSTM(64, activation="tanh", return_sequences=True, input_shape=(X_train_f.shape[1], X_train_f.shape[2])),
    LSTM(32, activation="tanh"),
    Dense(16, activation="relu"),
    Dense(32, activation="relu"),
    Dense(64, activation="relu"),
    Dropout(0.2),
    Dense(y_train_f.shape[1], activation="sigmoid")
])

model_f.compile(optimizer='adam', loss='mse')

hist= model_f.fit(X_train_f, y_train_f, epochs=1000, batch_size=1024, validation_data=(X_test_f, y_test_f))
```

Figure 19 LSTM architecture and parameters

Layer	Type	Nodes	Activation
1	LSTM	64	tanh
2	LSTM	32	tanh
3	Dense	16	relu
4	Dense	32	relu
5	Dense	64	relu
6	Dense	67	sigmoid

Optimizer	Loss Function	Epochs	Batch Size
Adam	MSE	1000	1024

Table 5 LSTM Architecture and Parameters summary

For the last layer, the activation function used was sigmoid as the target variable has been normalized to (0,1).

The loss function output during training for next day prediction was recorded and plotted as shown below. For next hour, the plot looks almost identical.

```
plot_loss(hist,10,1000)
```

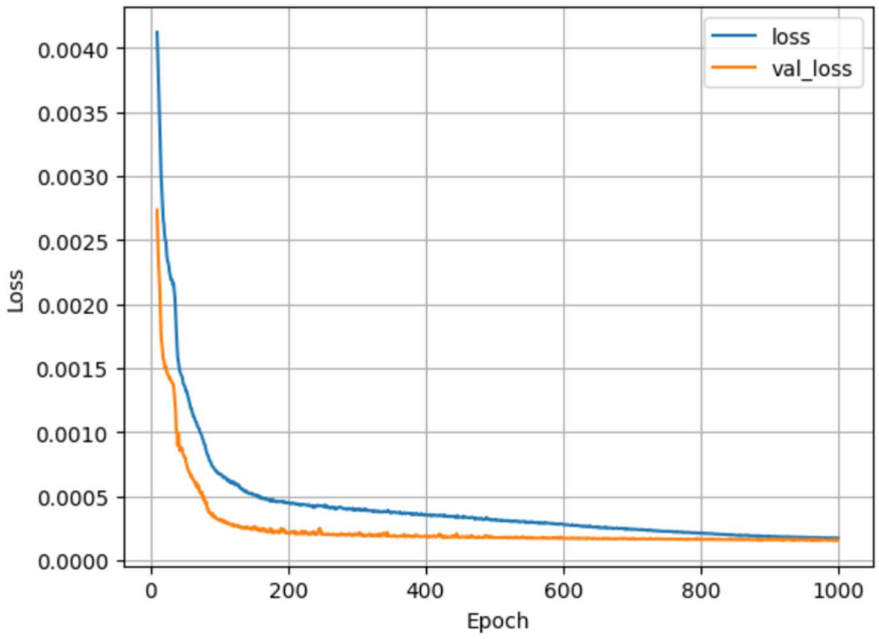


Figure 20 LSTM: Loss vs Epoch plot for next day prediction

As can be seen from the plot above, no overfitting has occurred as the validation loss (using the test set) is still decreasing (upon zooming in) up till the 1000 epoch.

The batch size of 1024 and epoch of 1000 were carefully chosen after many iterations to achieve this plot. A full training batch size is 14008.

Below details the thought process for arriving at these values:

1. A higher batch size will result in a slower loss reduction, which in turn requires more Epochs
2. A batch size of 1024 was chosen as it allowed for a reasonable training time while producing good results
3. The epoch value was carefully chosen to find a low loss without overfitting

The diagram below helps to explain how the batch size affects training.

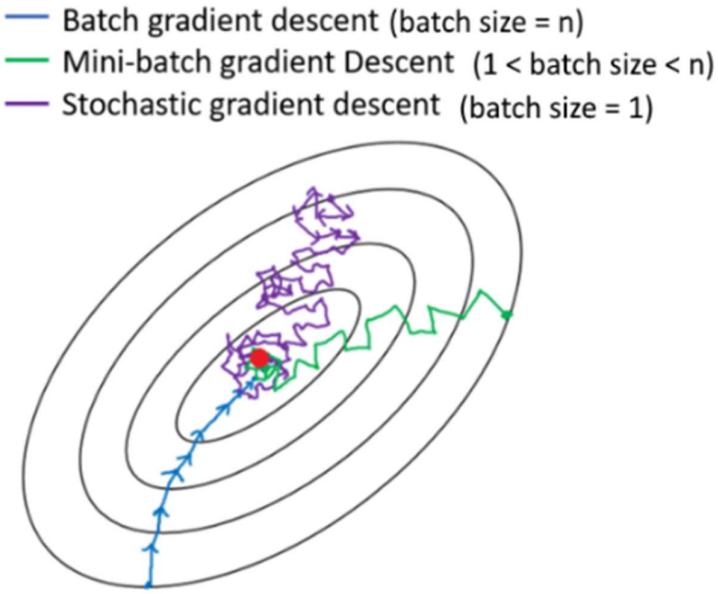


Figure 21 LSTM: Batch, Mini-batch and Stochastic gradient descent

8.1.2.4 LSTM Time Series Results and Evaluation Metrics

After training is completed, the evaluation metrics can be calculated. The order count value needs to be transformed back to their original scale.

Below are the results of our model predictions. The Mean Square and Root Mean Square Error values for both models are as shown.

A simplistic benchmark model was also done for comparison. It takes the last value of the order count for each location, treating it as the prediction value.

	LSTM	Benchmark	Percentage Improvement
MSE	1016.18	1996.3	0.490968291
RMSE	28.83	39.19	0.264353151

Table 6 LSTM: Next Hour Prediction Metrics

	LSTM	Benchmark	Percentage Improvement
MSE	1239.39	4816.94	0.742701798
RMSE	31.53	57.1	0.447810858

Table 7 LSTM: Next Day Prediction Metrics

One trend observed is that the performance of both the benchmark and the model declines as the prediction horizon extends. This is a common occurrence as the uncertainty associated with forecasting increases over longer timeframes.

Notably, the model's performance degrades at a slower rate. This is anticipated as well because while relying on the last known order count becomes increasingly unreliable with time, a model specifically trained for these kinds of longer-term predictions should maintain a more reasonable level of accuracy

Finally, the prediction vs actual order count of every location were plotted for visualization.

Here's an example of the test vs prediction plot of location 4 for next day. The rest of the result plots for both next hour and next day predictions can be found in the jupyter notebook.

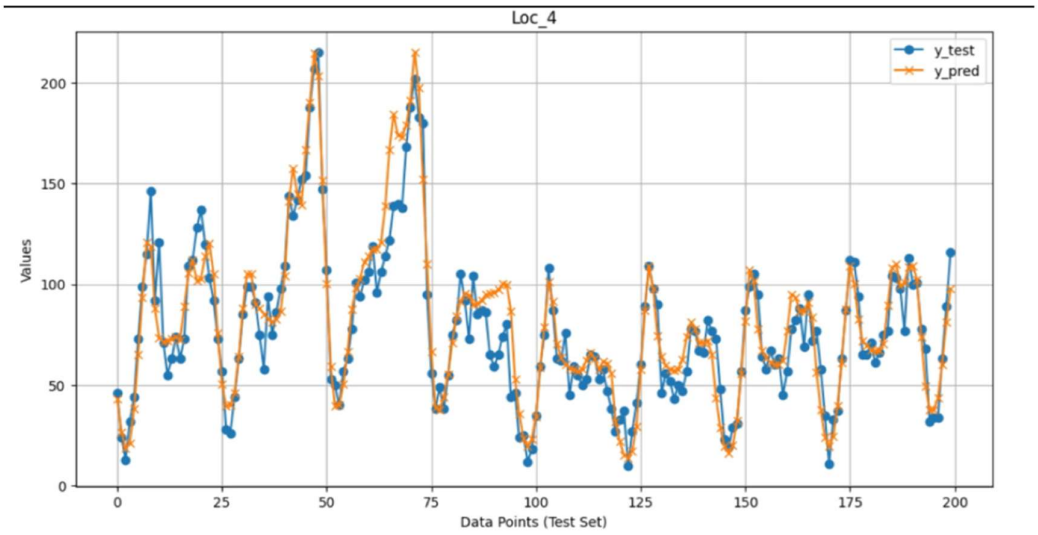


Figure 22 LSTM: Loc_4 Test vs Predicted Plot for next day predictions

8.1.2.5 LSTM Time Series Conclusions

The LSTM model performed very well compared to the benchmark model with > 25% improvement across all the metrics.

By preparing 2 prediction timeframes, the client can be flexible in choosing which to use. Next day predictions could be used to prepare the fleet allocation a day in advance, while next hour prediction can provide more accurate prediction during operation.

8.2 Survival (Wait Time) Analysis

8.2.1 Introduction

Using survival analysis is helpful to predict the probability that Uber ride request is picked up at time t, allowing it to forecast not only which requests are likely to be delayed but also to provide key factors affecting pick up wait time.

The project's objective is to help Uber:

- **Optimize driver allocation and reduce idle time**, improving overall efficiency.
- **Enhance the rider experience** by reducing wait times and improving pickup reliability.
- **Understand operational bottlenecks** unique to high-demand, high-density zones like Manhattan.

This analysis was conducted only for the locations in Manhattan Borough.

8.2.2 Data preparation:

1. Filtering:

The original Uber dataset for New York City in 2024 was filtered out Manhattan area ride request by Excel. The *PULocationID* linked to Manhattan pick up zone IDs.

2. Random Sampling:

The Manhattan filtered dataset was still large and potentially redundant. In order to avoid overfitting from extreme outliers and noise, randomly sampled 100,000 ride requests to help us to Improve computational efficiency during model training and testing.

3. Feature engineering:

Based on sampled dataset variables, *request-datetime,* merged with new features which are *demand-category* (peak, normal and non-peak) and *request day* (weekday and weekend).

4. Handling censoring:

Right Censoring occurs when there is not any recorded wait time for some requests. maybe due to cancellation by driver or rider.

Creating new feature *Event,*

- *event = 1: the rider was picked up eventually*
- *event = 0: the rider was not picked up*

5. Encoding categorical variables:

   survival models require all variables are numeric, our categorical variables PULocationID, demand_category, and request_day, not suitable for it, hence they have to be transformed to numeric variables by one-hot encoding method, which create nonary numeric columns (0 or 1) for every category.

For instance:

Original variable:
PULocationID = 99

demand_category = peak

request_day = Monday

After one-hot encoding:
PULocationID_99 = 1 :in pickup zone 99

demand_category_peak = 1:peak period

request_day_Monday = 0: not monday

6. Create interaction terms:

   generated interaction terms to observe conditional relationships:

- ***PULocationID × request_day***.
- ***PULocationID × demand_category***

8.2.3 Methodology:

There are three survival analysis models to be used:

Kaplan-Meier Estimator: A non-parametric method used to observe data to estimate the conditional probability of confirmed survival at each observed survival time and then multiply them to obtain an estimate of the overall survival function. In this project, we used Colab to show Kaplan-Meier curve, which is used to compare group-level pickup behavior in Manhattan.

1. Weekday vs. Weekend

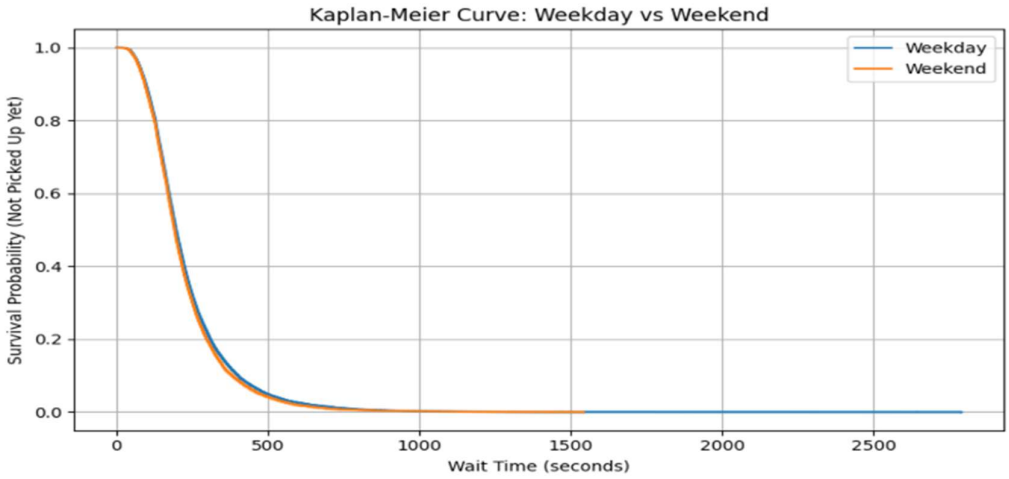


Figure 23 Kaplan-meier Curve for request day comparison

t_0	-1
null_distribution	chi squared
degrees_of_freedom	1
test_name	logrank_test

	test_statistic	p	-log2(p)
0	28.51	<0.005	23.36

Figure 24 Logrank-test for request day

p < 0.005 indicates a **statistically significant difference** in wait times between weekdays and weekends.

2. Peak period vs. non-peak period vs. normal period

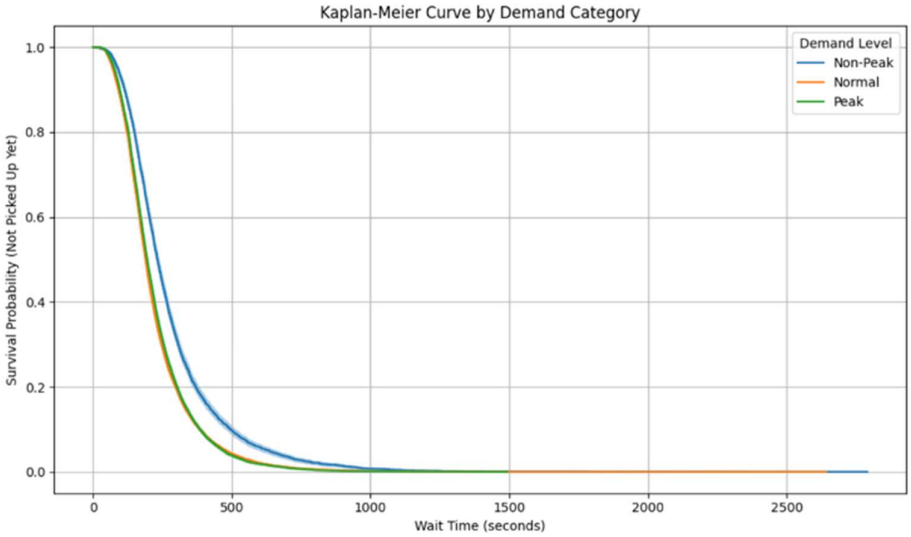


Figure 25 Kaplan-meier Curve for demand category

t_0	-1
null_distribution	chi squared
degrees_of_freedom	2
test_name	multivariate_logrank_test

	test_statistic	p	-log2(p)
0	449.43	<0.005	324.20

Figure 26 Multivariate-logrank-test for demand category

p < 0.005 indicates a **statistically significant difference** in wait times between Normal , peak and non-peak period.

Key findings for 2 curves:

1. **Most pickups occur within the first 8–10 minutes**, regardless of the day or demand time, showing Uber has high operational efficiency.
2. A **longer tail on weekdays** shows a small number of rides experience extended wait times, possibly during peak traffic hours.
3. Peak period have shortest wait times.
4. Other periods have longer tail shows longer wait times, possibly due to fewer available driver.
5. Although at first time we expected weekdays and weekends has significant different pickup behaviour, the KM curves showed **minor difference** — suggesting that **pickup dynamics are more influenced by factors like demand category and zone**, rather than day alone.

Cox Proportional Hazards (CoxPH) Model: A semi-parametric regression model that relates multiple predictors to the hazard (event risk) over time. It assumes **proportional hazards** — the ratio of hazards(HR) show the relative effect of each variable on the hazard. In this project, the hazard is wait times rate, and it helps us to understand how multiple factors influence wait times together.

- **HR > 1**: Faster pickup (shorter wait time)
- **HR < 1**: Slower pickup (longer wait time)

Base Cox proportional hazard model includes *demand_category* and *pickup location*.

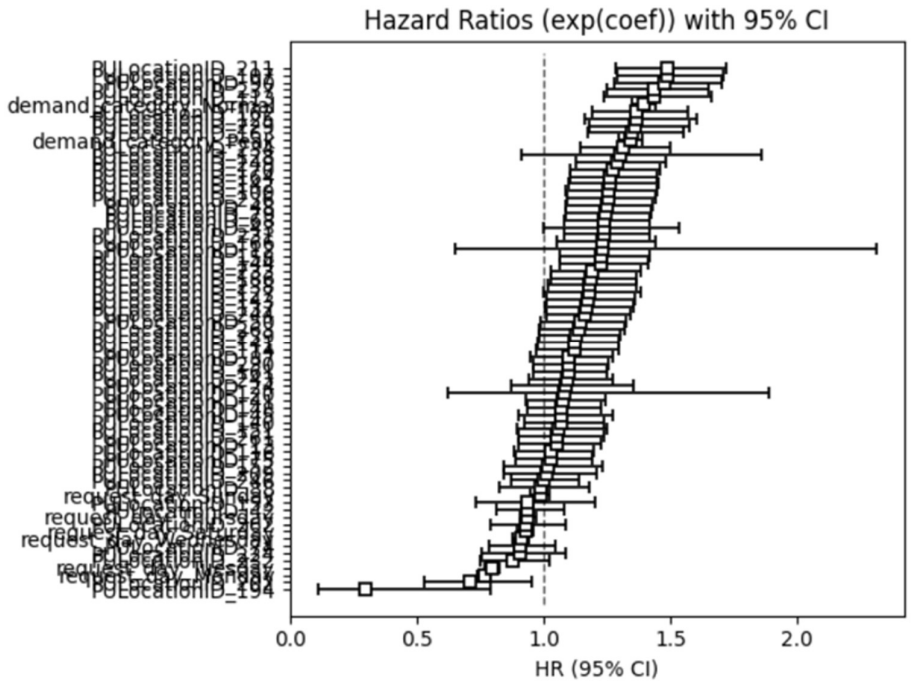


Figure 27 Base Cox model Hazard ratios

there are **39 significant terms** ($p < 0.05$).

```
[ ]  num_significant_vars = significant_vars.shape[0]
     print(f"Number of significant variables: {num_significant_vars}")

     Number of significant variables: 39
```

Figure 28 No. of Significant terms

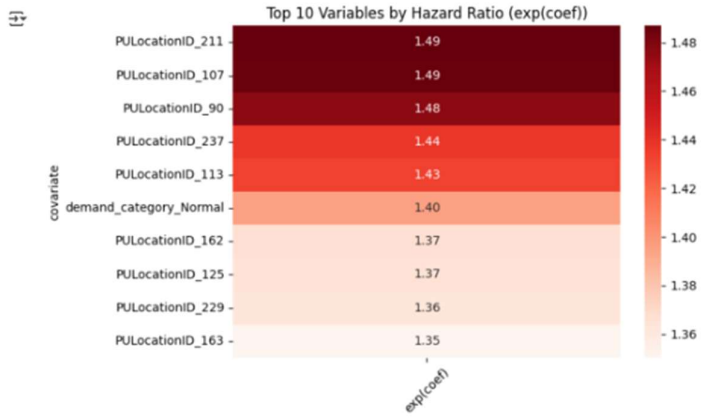


Figure 29 Top 10 (fastest) pickup locations

Summary of CoxPH Top 10 Fastest Pickup Zones:

Rank	Variable	Hazard Ratio (HR)	Interpretation
1	PULocationID_211	1.49	Fastest pickup zone (49% faster)
2	PULocationID_107	1.49	Fastest pickup zone (49% faster)
3	PULocationID_90	1.48	48% faster than baseline
4	PULocationID_237	1.44	44% faster than baseline
5	PULocationID_113	1.43	43% faster than baseline
6	demand_category_Normal	1.40	Normal demand periods improve pickup
7	PULocationID_162	1.37	37% faster than baseline
8	PULocationID_125	1.37	37% faster than baseline
9	PULocationID_229	1.36	36% faster than baseline
10	PULocationID_163	1.35	35% faster than baseline

Table 8 Summary of CoxPH Top 10 Fastest Pickup Zones

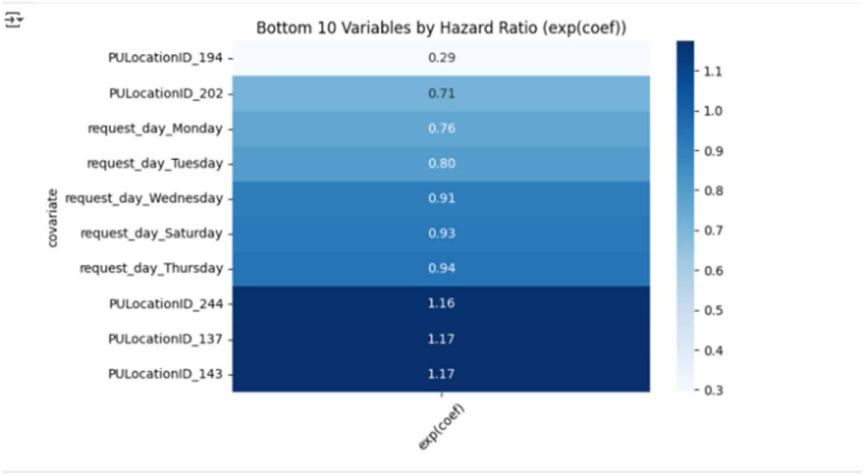


Figure 30 Bottom 10 (lowest) pickup locations

Summary of CoxPH Bottom 10 Slowest Pickup Zones:

Rank	Variable	Hazard Ratio (HR)	Interpretation
1	PULocationID_194	0.29	Slowest pickup zone (71% slower)
2	PULocationID_202	0.71	29% slower than baseline
3	request_day_Monday	0.76	24% slower on Mondays
4	request_day_Tuesday	0.80	20% slower on Tuesdays
5	request_day_Wednesday	0.91	9% slower on Wednesdays
6	request_day_Saturday	0.93	7% slower on Saturdays
7	request_day_Thursday	0.94	6% slower on Thursdays
8	PULocationID_244	1.16	16% faster than baseline

9	PULocationID_137	1.17	17% faster than baseline
10	PULocationID_143	1.17	17% faster than baseline

Table 9 Summary of CoxPH Bottom 10 Slowest Pickup Zones

- Pickup zone like **PULocationID_211**, **PULocationID_107**, and **PULocationID_90** consistently demonstrated faster wait times, indicating high operational efficient.
- In contrast, pick up zones like **PULocationID_194** and **PULocationID_202**, as well as requests on **Mondays and Tuesdays**, showing slower pick up times.

Pickup dynamics in Manhattan are influenced by urban conditions such as traffic jam, surge pricing, and driver rerouting. Hence, we think the hazard (picked up occurred) may change over time rather than remain constant, which Cox proportional assumes constant hazard ratios over time. In constant, **Aalen's Additive Model** is a non-parametric model which variables effect are allowed to vary over time. Thus, through Aalen's additive model, we can identify how pickup zones' performance changes dynamically.

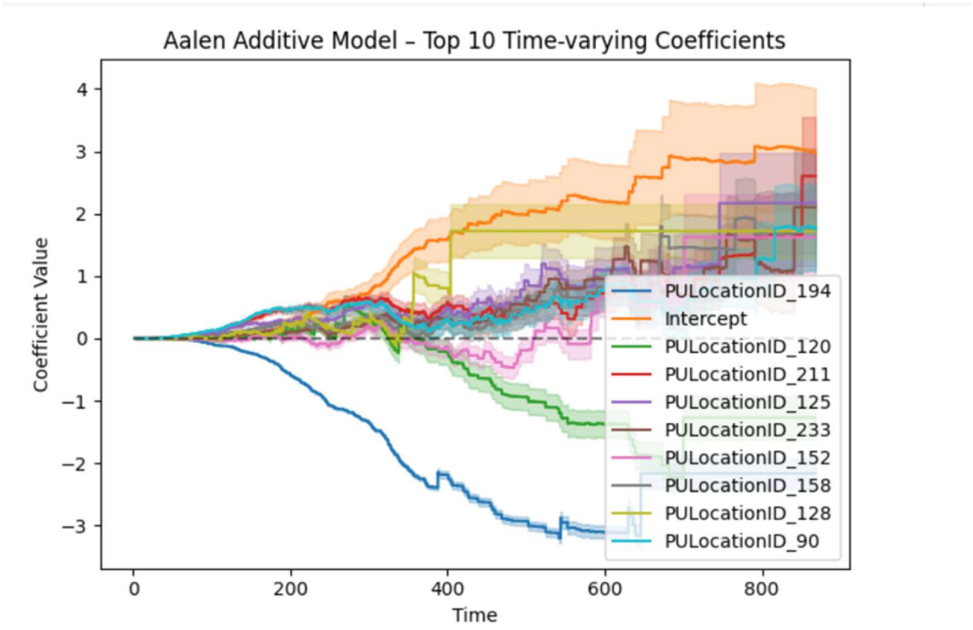


Figure 31 Aalen Additve Model - Top 10 Time-varying Coefficients

```
[22] aaf.score(df_model, scoring_method='concordance_index')

np.float64(0.562056613373766)
```

Figure 32 Concordance index

- **C-index is 0.562, it means** the model correctly ranked pickup times in 56.2% of the comparable rider pairs. although the percentage shows weak, it is still non-random predictive ability. Hence, we can rely on the key findings from Top10 time varying plot:

Rank	Variable	Time-Varying Effect Trend	Interpretation
1	PULocationID_194	Strong negative effect	Worst pickup zone, worsens over time
2	PULocationID_120	Moderate negative effect	Pickup probability worsens over time
3	PULocationID_211	Strong positive effect	Best pickup zone, improves over time
4	PULocationID_125	Positive effect	Pickup efficiency improves over time
5	PULocationID_233	Slight positive effect	Modest pickup improvement over time
6	PULocationID_152	Slight positive effect	Some pickup improvement over time
7	PULocationID_158	Mild positive trend	Pickup efficiency slightly improves
8	PULocationID_128	Mild positive trend	Pickup efficiency slightly improves
9	PULocationID_90	Neutral to mild positive	Pickup performance relatively stable but slightly improving
10	Intercept	Overall baseline increasing trend	Pickup environment improving over time citywide

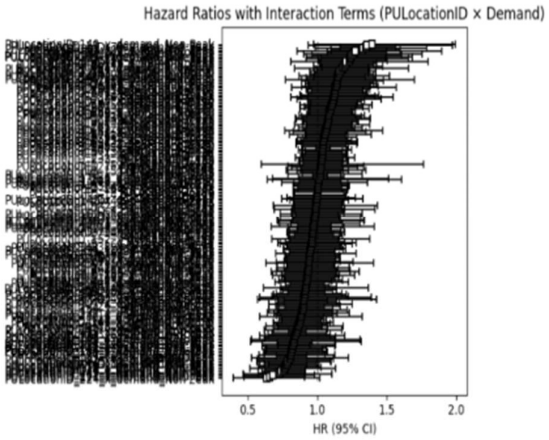
Table 10 Summary of Top 10 Time-varying Coefficients

From Cox base model and Aalen's additive model comparison, there are some findings need to be considered:

1. PUlocationID_194 is flagged by both models as consistently performance poor.
2. PUlocationID_211, 125, 90 in both models are consistently performance good.
3. PUlocationID_120 is not flagged by Cox base model, but Aalen's additive model indicates it gets worse over time.

While the Aalen's Additive Model helps us to capture how individual zone effects evolve over waiting time, real-world pickup behavior is often influenced by a combination of multiple conditions rather than a single factor alone. As KM curve result mentioned before, pickup dynamics are more influenced by factors like demand category and zone**,** rather than day alone. Therefore, to uncover **deeper, context-specific pickup dynamics**, extending our modeling by introducing **interaction terms** into the CoxPH model is necessary.

- *PULocationID × demand_category* to capture how pickup zones performance changes with demand level.




```
num_significant_vars = significant_vars.shape[0]
print(f"Number of significant variables: {num_significant_vars}")
```

```
Number of significant variables: 19
```

Figure 33 Hazard ratio with interactions and no. of significant terms

There are a total of 19 interaction terms:

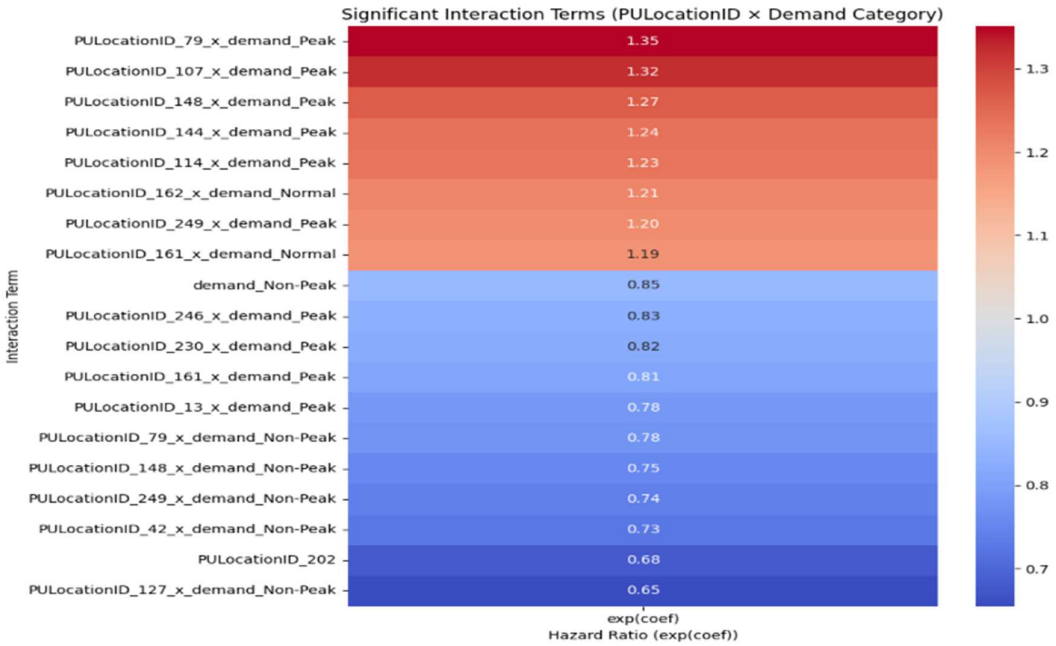


Figure 34 Significant interaction terms (PULocationID X Demand category)

Rank	Interaction Term	Hazard Ratio (HR)	Interpretation
1	PULocationID_79 × demand_Peak	1.35	Pickup much faster during peak at Zone 79
2	PULocationID_107 × demand_Peak	1.32	Pickup faster during peak at Zone 107
3	PULocationID_148 × demand_Peak	1.27	Pickup faster during peak at Zone 148
4	PULocationID_144 × demand_Peak	1.24	Pickup faster during peak at Zone 144
5	PULocationID_114 × demand_Peak	1.23	Pickup faster during peak at Zone 114
6	PULocationID_162 × demand_Normal	1.21	Pickup faster during normal demand at Zone 162

7	PULocationID_249 × demand_Peak	1.20	Pickup faster during peak at Zone 249
8	PULocationID_161 × demand_Normal	1.19	Pickup faster during normal demand at Zone 161
9	Demand_Non-peak	0.85	Pickuo slower durong non-peak period
10	PULocationID_246 × demand_Peak	0.83	Pickup slower during peak at Zone 246
11	PULocationID_230 × demand_Peak	0.82	Pickup slower during peak at Zone 230
12	PULocationID_161 × demand_Peak	0.81	Pickup slower during peak at Zone 161
13	PULocationID_113 × demand_Peak	0.78	Pickup slower during peak at Zone 113
14	PULocationID_79 × demand_NonPeak	0.78	Pickup slower during non-peak at Zone 79
15	PULocationID_148 × demand_NonPeak	0.75	Pickup slower during non-peak at Zone 148
16	PULocationID_249 × demand_NonPeak	0.74	Pickup slower during non-peak at Zone 249
17	PULocationID_42 × demand_NonPeak	0.73	Pickup slower during non-peak at Zone 42
18	PULocationID_202	0.68	Consistently slower pickup at Zone 202 regardless of demand

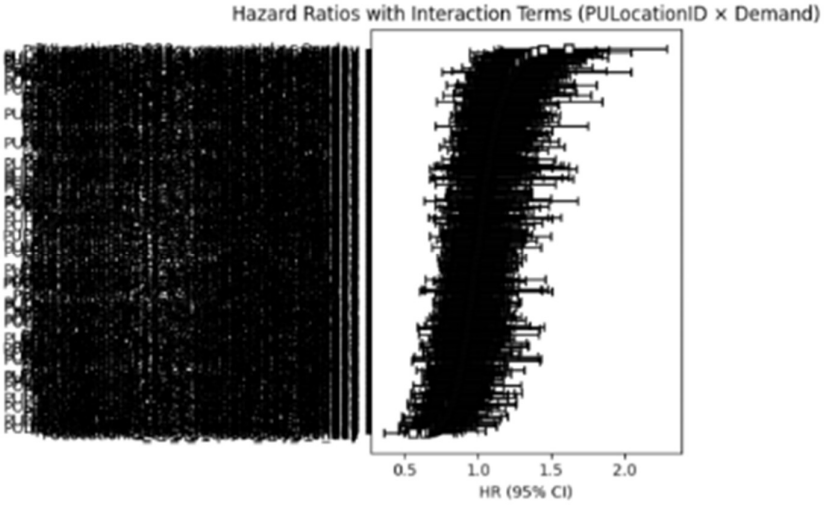
19	PULocationID_127 × demand_NonPeak	0.65	Pickup much slower during non-peak at Zone 127

Table 11 Summary of significant interactions between PULocations and demand

In conclusion, peak period improves pickups speed, and non-peak period slower down speed.

PULocationID_202 is consistently poor no matter demand level.

- *PULocationID × request_day* to show pickup zones perform across weekdays and weekends.




```
num_significant_vars = significant_vars.shape[0]
print(f"Number of significant variables: {num_significant_vars}")
```

```
Number of significant variables: 35
```

Figure 35 Hazard ratio with interactions and no.of significant terms

There are a total of 35 significant interaction terms:

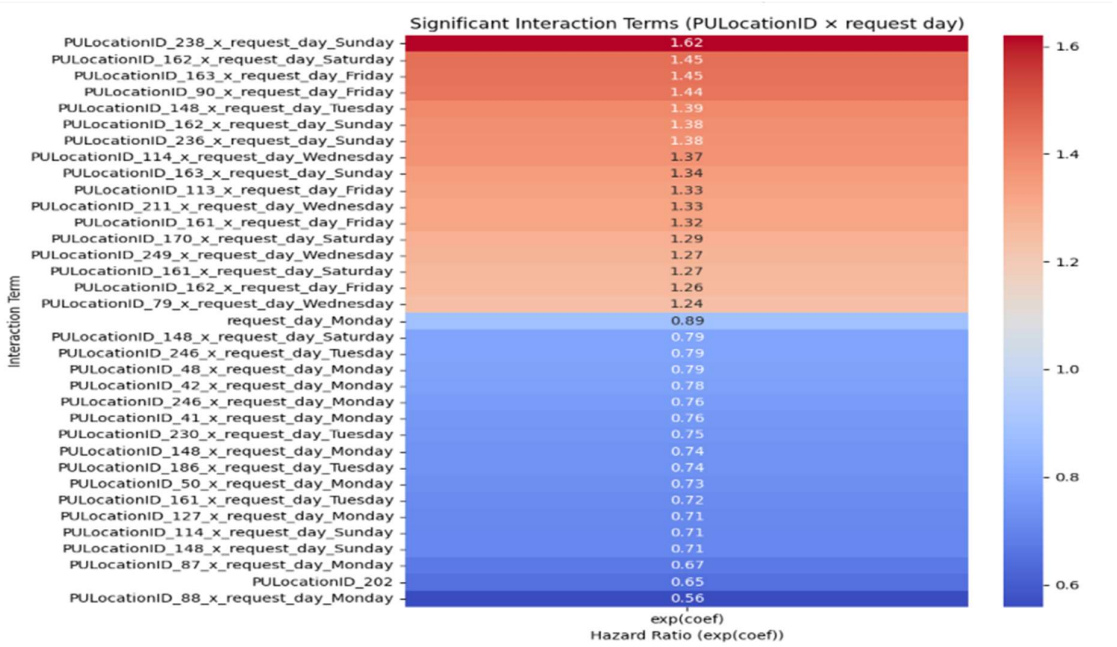


Figure 36 Significant interaction terms (PULocationID X Request day)

Rank	Interaction Term	Hazard Ratio (HR)	Interpretation
1	PULocationID_238 × request_day_Sunday	1.62	Pickup much faster at Zone 238 on Sunday
2	PULocationID_162 × request_day_Saturday	1.45	Faster pickup at Zone 162 on Saturday
3	PULocationID_163 × request_day_Friday	1.45	Faster pickup at Zone 163 on Friday
4	PULocationID_90 × request_day_Friday	1.44	Faster pickup at Zone 90 on Friday
5	PULocationID_148 × request_day_Tuesday	1.39	Faster pickup at Zone 148 on Tuesday
6	PULocationID_162× request_day_Sunday	1.38	Faster pickup at Zone 162 on Sunday

7	PULocationID_236 × request_day_Sunday	1.38	Faster pickup at Zone 236 on Sunday
8	PULocationID_114 × request_day_Wednesday	1.37	Faster pickup at Zone 114 on Wednesday
9	PULocationID_163 × request_day_Sunday	1.34	Faster pickup at Zone 163 on Sunday
10	PULocationID_113 × request_day_Friday	1.33	Faster pickup at Zone 113 on Friday
11	PULocationID_211 × request_day_Wednesday	1.33	Faster pickup at Zone 211 on Wednesday
12	PULocationID_161× request_day_Sunday	1.32	Faster pickup at Zone 161 on Friday
13	PULocationID_170 × request_day_Saturday	1.29	Faster pickup at Zone 170 on Saturday
14	PULocationID_249 × request_day_Wednesday	1.27	Faster pickup at Zone 249 on Wednesday
15	PULocationID_161 × request_day_Saturday	1.27	Faster pickup at Zone 161 on Saturday
16	PULocationID_162 × request_day_ Saturday	1.26	Faster pickup at Zone 162 on Saturday
17	PULocationID_79 × request_day_Wednesday	1.24	Faster pickup at Zone 79 on Wednesday
18	Request_day_Monday	0.89	Slower pickup on Monday

19	PULocationID_148× request_day_Saturday	0.79	Slower pickup at Zone 148 on Saturday
20	PULocationID_246 × request_day_Tuesday	0.79	Slower pickup at Zone 249 on Tuesday
21	PULocationID_48 × request_day_Monday	0.79	Slower pickup at Zone 48 on Monday
22	PULocationID_42 × request_day_Monday	0.78	Slower pickup at Zone 42 on Monday
23	PULocationID_246 × request_day_Monday	0.76	Slower pickup at Zone 246 on Monday
24	PULocationID_41 × request_day_Monday	0.76	Slower pickup at Zone 41 on Monday
25	PULocationID_230 × request_day_Tuesday	0.75	Slower pickup at Zone 230 on Tuesday
26	PULocationID_148 × request_day_Monday	0.74	Slower pickup at Zone 148 on Monday
27	PULocationID_186 × request_day_Tuesday	0.74	Slower pickup at Zone 186 on Tuesday
28	PULocationID_50 × request_day_Monday	0.73	Slower pickup at Zone 50 on Monday
29	PULocationID_161 × request_day_Tuesday	0.72	Slower pickup at Zone 161 on Tuesday
30	PULocationID_127 × request_day_Monday	0.71	Very slow pickup at Zone 127 on Monday

31	PULocationID_114 × request_day_Sunday	0.71	Very slow pickup at Zone 114 on Sunday
32	PULocationID_148 × request_day_Sunday	0.71	Very slow pickup at Zone 148 on Sunday
33	PULocationID_87 × request_day_Monday	0.67	Very slow pickup at Zone 87 on Monday
34	PULocationID_202	0.65	Very slow pickup at Zone 202
35	PULocationID_88 × request_day_Monday	1.14	Very slow pickup at Zone 88 on Monday

Table 12 Summary of significant interactions between PULocations and request day

In conclusion, Sunday and Friday are good for many zones since there are faster pickups. However, Monday is consistently poor performance, shows slower pickups. PULocationID_202 is consistently poor all week.

8.2.4 Cox model evaluation:

Proportional Hazards Assumption Check using Schoenfeld residuals to confirm model validity. [2]

```
cox.check_assumptions(df, p_value_threshold=0.05)
```

```
The ``p_value_threshold`` is set at 0.05. Even under the null hypothesis of no violations, some
covariates will be below the threshold by chance. This is compounded when there are many covariates.
Similarly, when there are lots of observations, even minor deviances from the proportional hazard
assumption will be flagged.
```

Figure 37 Proportional Hazards Assumption Check

According to Wang et al. (2023), even under the null hypothesis of no violation, some covariates may show statistically significant p-values **simply due to the number of variables tested or large sample sizes**. This can lead to **overdiagnosis of violations**, especially in real-world, high-dimensional datasets like this Uber demand dataset.

8.2.5 Survival analysis Conclusion

By conducting survival model, we can provide Uber with actionable insights to **optimize dispatching, reduce wait times**, and **prioritize problem zones**, there are some business implications:

1. Driver allocation:
- increase available drivers for under-performance zones during peak period.
2. Dynamic pricing and incentive strategies:
- specific zones surge pricing, such as slow pickup zone during peak period.
- providing promotion/offer to pickup zone which are worsen over time.

8.3 Monte Carlo Simulation and Linear Programming

8.3.1 Data Selection and Assumptions

The data in this analysis covers the period from January to June 2023. The goal is to identify typical hour-based travel patterns and provide a robust foundation for subsequent simulation and resource optimization. We focus on the hourly distribution of trip orders in the Manhattan area, aiming to capture the commuting dynamics and demand patterns in New York's central business district.

To enhance the data's usability and representativeness, the following filtering and preprocessing steps were applied:

- **Time Range:** Limited **to January 1 to June 30, 2023**, to avoid distortions from holidays or special events throughout the year, ensuring temporal consistency and improving computational efficiency.
- **Geographic Scope:** Focused exclusively on Manhattan, **excluding non-operating zones (e.g., PULocationID = 105).** This area features high commuting intensity and population density, making it both representative and analytically valuable.
- **Time Granularity:** Set at the **hourly level** to allow for clear **comparisons between peak and off-peak demand periods** and to pinpoint critical time slots.
- **Processing Method:** Leveraged Dask for efficient, parallelized aggregation of large-scale datasets, followed **by Kernel Density Estimation (KDE) to explore distributional characteristics** and volatility across hours.

8.3.2 Order Level Distribution by Hour

We aggregated the order counts across all 24 hours in a day and visualized the results using hourly histograms combined with KDE curves. Key insights are as follows:

- **Pronounced Right-Skewed Distributions:** During early morning hours (0–2 AM, 4–6 AM), order volumes tend to cluster at lower levels, occasionally spiking, forming long-tailed distributions.
- **Clear Peak-Hour Patterns:** Morning (7–9 AM) and evening (4–7 PM) peak periods show consistently high order volumes and concentrated KDE curves, indicating stable and dense commuter demand.
- **High Volatility During Night and Off-Peak Hours:** Periods such as 0–5 AM and 9–11 PM exhibit dispersed, multi-modal or heavy-tailed distributions, signalling greater demand uncertainty and modelling risk.

- **Bandwidth Variability Reflects Stability:** For example, the 9 AM KDE bandwidth is approximately 955, indicating a tight, predictable distribution. In contrast, 1 AM and 4 AM bandwidths exceed 1200, showing sparse, unstable demand distributions.

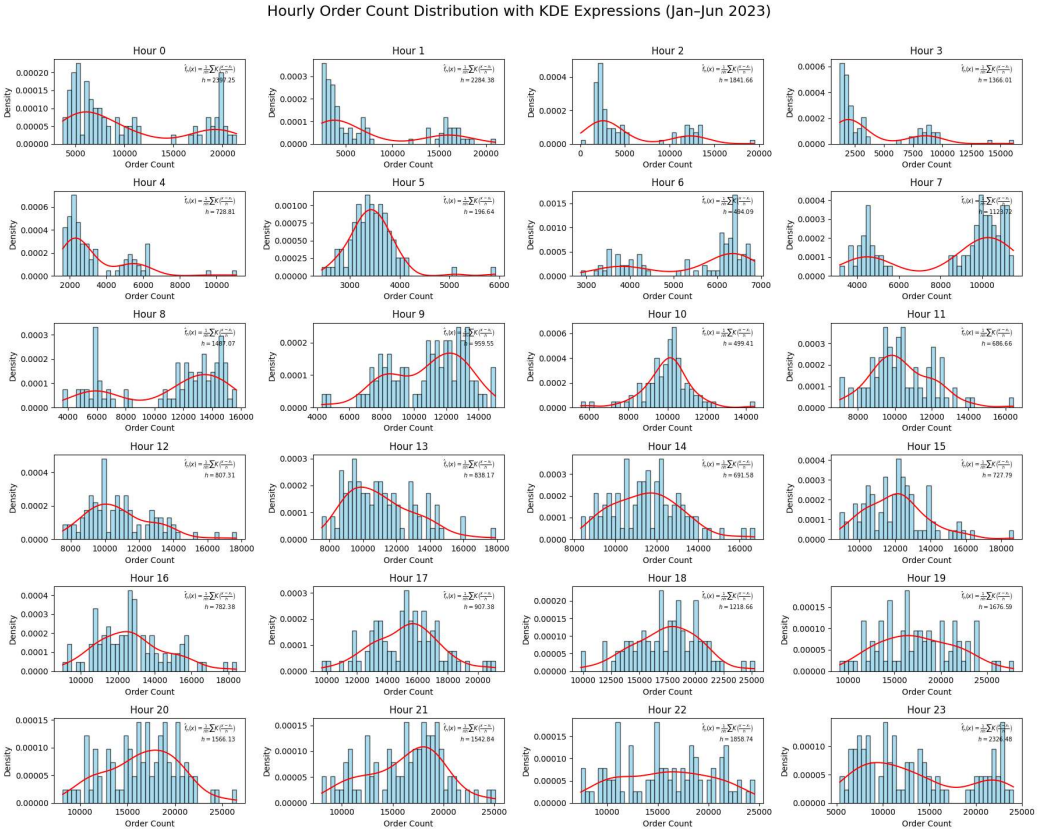


Figure 38 Hourly Order Level Distribution

8.3.3 "Key-Hour" Selection

Considering these features, **9:00 AM emerges as the most balanced and optimal time window in terms of order volume, distribution compactness, and modelling feasibility**. It was therefore selected as the focal point for simulation and linear programming analysis.

It demonstrates strong order density and predictive value but also offers superior statistical stability and strategic business relevance. Whether for demand forecasting, revenue optimization, or resource allocation, this hour represents a critical inflection point where supply-demand tension peaks and data-driven interventions are most impactful. It has therefore been selected as the core analytical unit for this phase of the study.

8.3.4 Order Level Distribution by Location

We conducted a granular distributional analysis of order counts across all active PULocationIDs (pickup locations) within Manhattan 9AM. Each region's historical order data was aggregated and fitted with a Kernel Density Estimation (KDE) model to capture the characteristics of its demand.

8.3.4.1 Distribution Characteristics

1. Strong Heterogeneity Across Locations

Order volumes vary significantly between locations, ranging from single-digit counts (e.g.,

PULocationID 128) to over 400 orders (e.g., PULocationID 170). **High-demand areas** such as 100, 142, 186, 229, 234, 246, and 249 show **unimodal**, concentrated distributions, identifying them as consistent demand sinks. In contrast, **low-traffic or peripheral zones** (e.g., 128, 194, 109) exhibit sparse, Poisson-like distributions with minimal and volatile activity.

2. Diverse Distributional Shapes

Locations like 186, 234, and 237 display symmetric, **bell-shaped curves**, indicating stable fluctuations within a narrow band. Seen in locations like 140 and 170, where most orders concentrate at lower volumes but include occasional spikes. These are well-suited for log-normal modeling. **Volatile patterns** are also found in zones such as 166 and 224, potentially due to event-driven demand (e.g., near schools, convention centers).

3. Variability in Sample Size and KDE Stability

Locations with fewer samples (e.g., 12, 128, 194) yield higher bandwidths and less stable KDE curves, often with jitter or over-smoothing. Conversely, high-volume areas such as 231, 234, and 249 produce smooth, reliable curves ideal for statistical inference and model fitting.

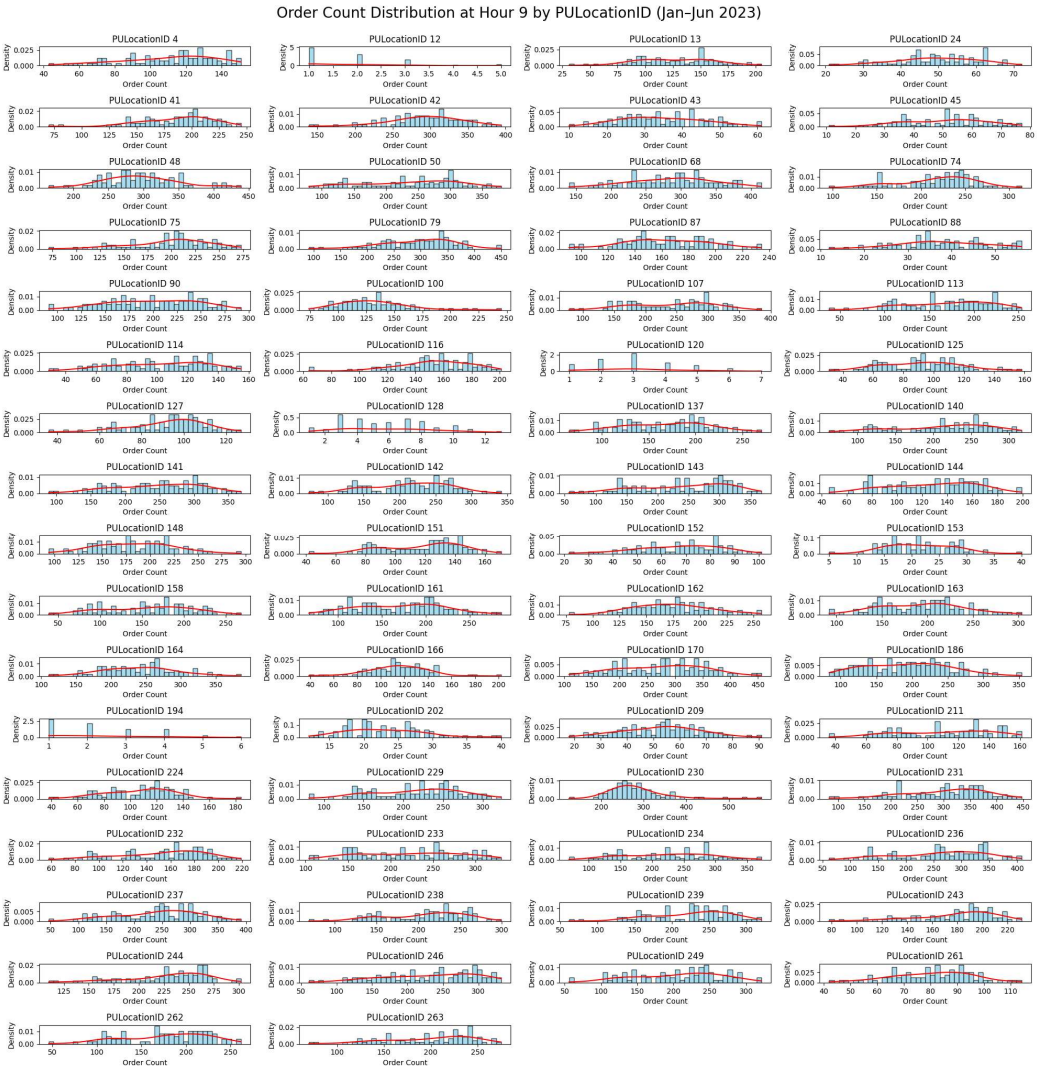


Figure 39 Order Count Distribution at Hour 9 by Location

8.3.5 Order Level Distribution Monte Carlo Simulation

8.3.5.1 Simulation Design

To operationalize this, we implemented a **Monte Carlo simulation** for each PULocationID.

First, we fitted a KDE function to each location's historical 9 AM order data. We then used scipy.stats.gaussian_kde.resample() to draw **10,000 random samples** per location, simulating possible order volumes for a future 9 AM period. The resulting samples were visualized in green histograms and compared against the original KDE curves (in red) to assess alignment.

8.3.5.2 Simulation Results and Interpretability

The simulations generally reproduced historical distributions with high fidelity.

- **In high-volume** regions such as **100, 142, 231, and 234**, the simulated and empirical curves nearly overlapped, validating the KDE fit.
- **In low-volume** or irregular regions like **12, 128, and 194**, slight deviations in tail behavior or

peak sharpness were observed—underscoring the importance of bandwidth calibration and the limits of KDE in sparse data regimes.

Overall, the Monte Carlo-enhanced KDE approach captures the nuanced, location-specific nature of urban mobility demand without requiring explicit parametric assumptions. It lays a robust foundation for downstream applications such as driver allocation optimization, surge pricing simulation, or scenario-based demand planning.

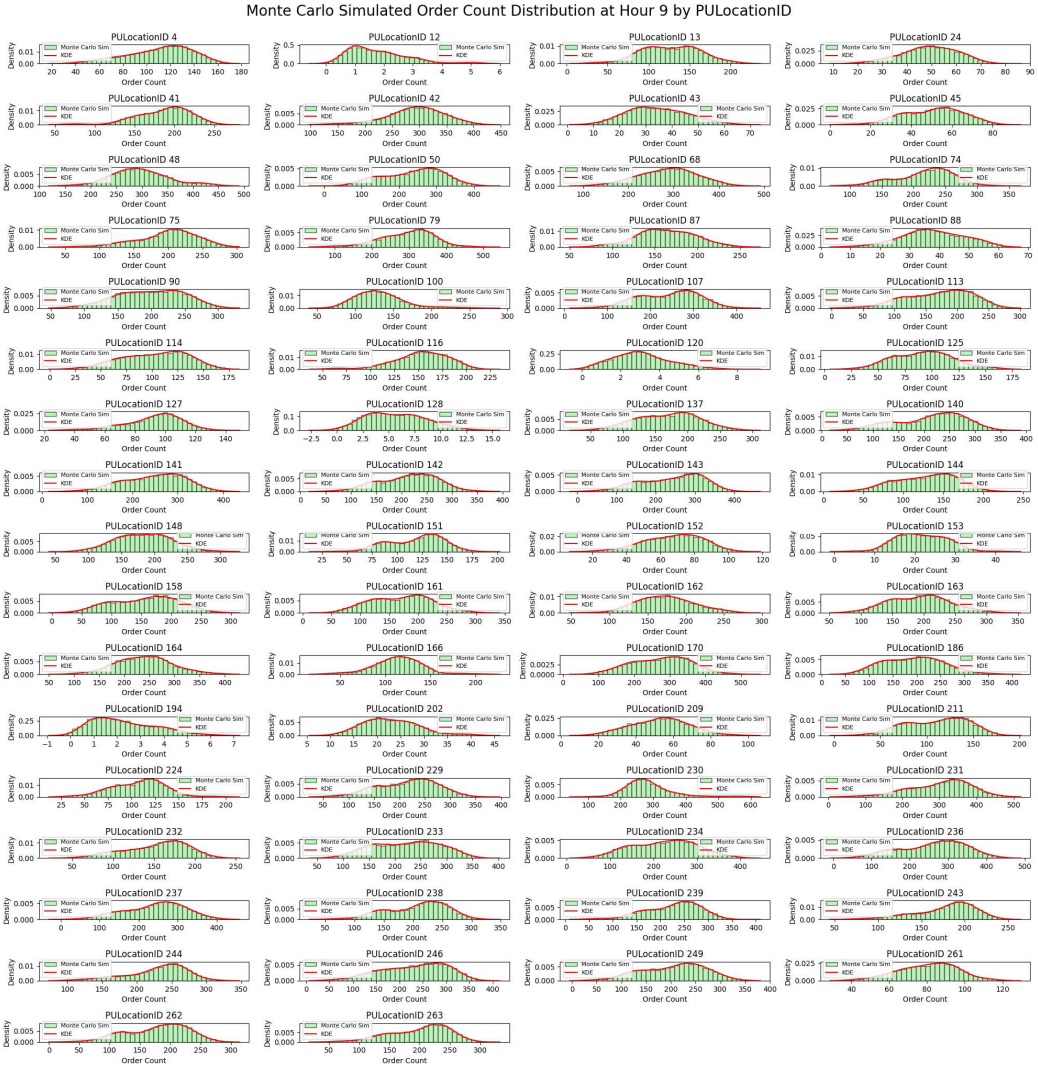


Figure 40 Monte Carlo Simulated Order Count Distribution at Hour 9 by Location

8.3.6 Lineal Programming – Fleet allocation optimization

8.3.6.1 Objective function

Total revenue = $\sum_{location}$ (Expected fulfilled orders × Average trip distance × Earnings per mile)

- **Expected fulfilled orders:** Depends on **the number of drivers X_i allocated to each region** and are estimated via **Monte Carlo simulation** based on historical trip data.
- **Unit revenue**: The average trip distance and earnings per mile are calculated from actual trip and income

records (earnings per mile = driver_pay / trip_miles).

The model optimizes **the number of drivers X_i** allocated to each region to maximize total revenue.

8.3.6.2 Constraints

1. **$\sum X_i \leq$ Average Order Level × (1 + leisure_rate)**

   The total number of drivers allocated across all regions must not exceed the available driver pool. Assume Leisure rate to be 30%.

2. **$d_i\_min \leq X_i \leq d_i\_max$**

   Each region's driver allocation must fall within a valid range.

   d_i_min: Minimum demand level of each borough

   d_i_max: Maximum demand level of each borough

8.3.6.3 Optimization result

Results show a significant revenue increase:

Original Revenue: $42,763.68 **Optimized Revenue:** $52,526.41

Revenue Improvement: $9,762.73 **(+22%)**

The visual comparison confirms that the optimized strategy shifts drivers toward locations with **higher marginal value**, such as PULocationIDs 229, 234, 246, while slightly reducing allocation in lower-efficiency areas like 12, 128, and 153.

Potential Limitation:
This model does not yet account for real-world constraints such as traffic congestion, idle time between trips, driver variability, nor public holidays or city events, which may impact actual fulfillment efficiency and cost. Future iterations could integrate travel-time-adjusted productivity scores or route-level constraints to refine the allocation.

Overall, the optimized allocation strategy demonstrates how data-driven simulation and linear optimization can materially enhance operational efficiency in dynamic ride-hailing environments.

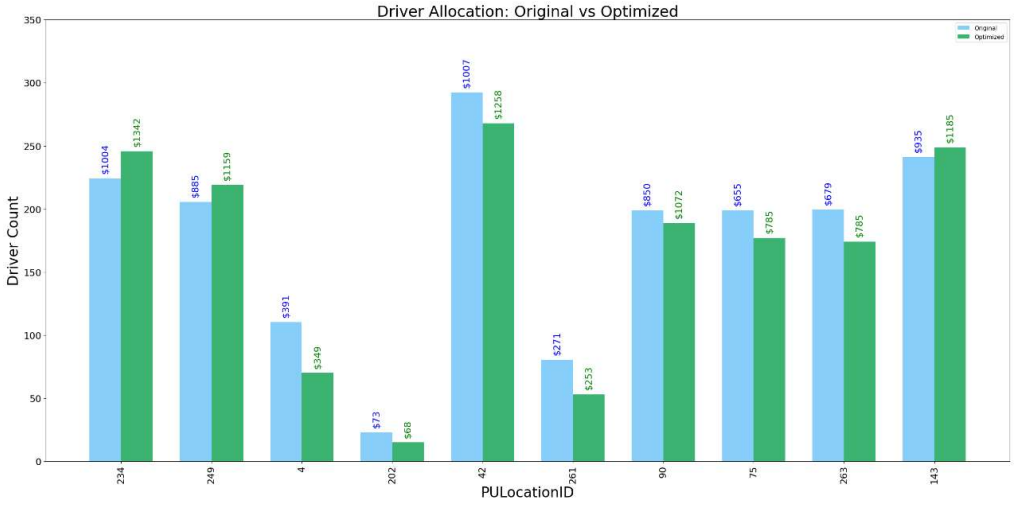


Figure 41 Fleet Optimization Result (10 locations)

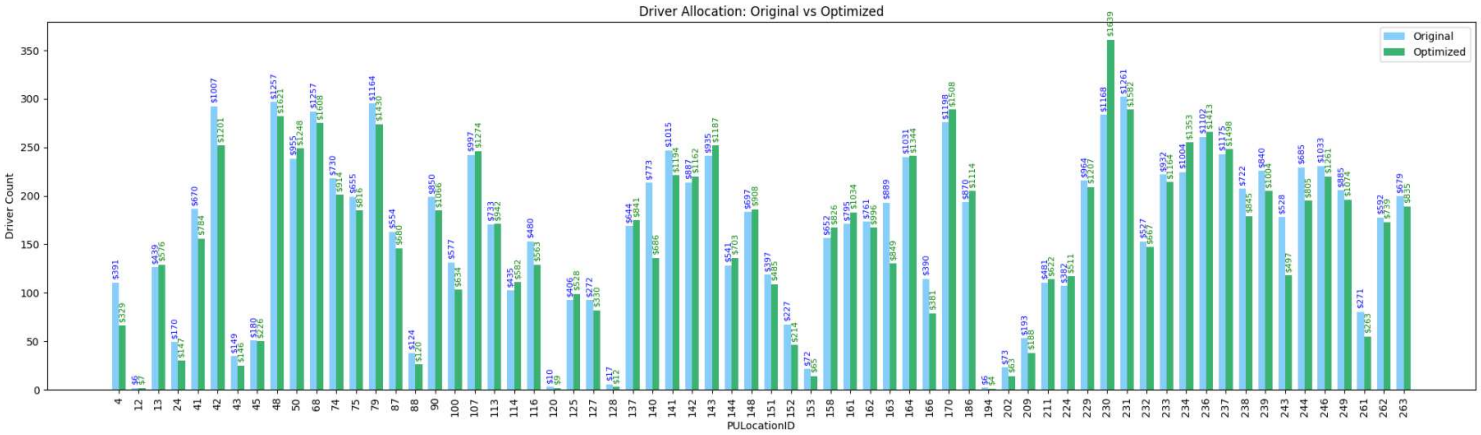


Figure 42 Fleet Optimization Result (Full Sample)

9. Price Optimisation

9.1 Price Elasticity

9.1.1 Variable Construction

To analyse price elasticity, this study centers around two key variables:

- **Driver Pay per Mile = driver_pay / trip_miles.** This serves as a proxy for unit price, representing the incentive level for drivers per mile traveled.
- **Order Count**: **The total number of completed orders** per hour per region, used as the dependent variable.

Given the presence of extreme values in the raw dataset, both pay_per_mile and order_count were winsorized at the 5th and 95th percentiles to reduce the influence of outliers and improve model robustness. In addition, we constructed a log-transformed dependent variable, log_order_count = log(order_count + 1), to support log-linear modeling and mitigate issues of skewness and heteroscedasticity.

9.1.2 Initial Regression and Implications

9.1.2.1 Linear OLS Model

The model specification is as follows:

$$\text{Order Count}_t = \beta_0 + \beta_1 \cdot \text{Pay per Mile}_t + \varepsilon_t$$

OLS Regression Results

Dep. Variable:	order_count	R-squared:	0.603
Model:	OLS	Adj. R-squared:	0.603
Method:	Least Squares	F-statistic:	2.660e+04
Date:	Sun, 30 Mar 2025	Prob (F-statistic):	0.00
Time:	21:13:17	Log-Likelihood:	-1.7915e+05
No. Observations:	17544	AIC:	3.583e+05
Df Residuals:	17542	BIC:	3.583e+05
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-3.115e+04	358.685	-86.835	0.000	-3.18e+04	-3.04e+04
pay_per_mile	1.537e+04	94.256	163.099	0.000	1.52e+04	1.56e+04

Omnibus:	554.702	Durbin-Watson:	0.231
Prob(Omnibus):	0.000	Jarque-Bera (JB):	561.664
Skew:	0.410	Prob(JB):	1.09e-122
Kurtosis:	2.689	Cond. No.	29.3

Figure 44 Demand Curve- OLS regression-1

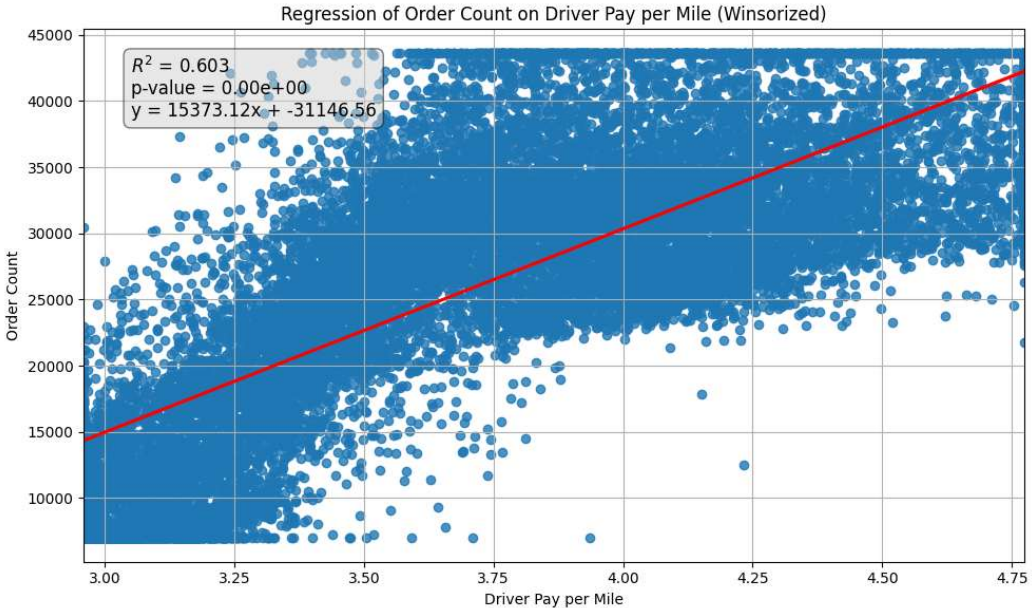


Figure 43 Demand Curve- OLS regression -2

Key findings from the OLS model:

Statistical Significance: The coefficient on pay_per_mile is +15,373.12, with **a p-value effectively zero.** This indicates a **highly significant and positive relationship** between unit price and completed orders.

Model Fit: **R^2 = 0.603**, meaning that over 60% of the variation in order count can be explained by driver pay per mile, demonstrating strong explanatory power.

Business Implication: The result reinforces the intuitive notion that higher unit compensation incentivizes driver participation, improving order fulfilment capacity across the system. But **unlike typical demand curves, we found a positive relationship — possibly because Higher unit price increases driver responsiveness and matching efficiency, leading to more completed orders and greater demand level**

9.1.2.2 Log-OLS Model

The model is specified as:

$$\log(1 + \text{Order Count}_t) = \beta_0 + \beta_1 \cdot \text{Pay per Mile}_t + \varepsilon_t$$

Figure 45 Demand Curve- Log-OLS regression 1

OLS Regression Results

Dep. Variable:	log_order_count	R-squared:	0.576
Model:	OLS	Adj. R-squared:	0.576
Method:	Least Squares	F-statistic:	2.383e+04
Date:	Tue, 29 Apr 2025	Prob (F-statistic):	0.00
Time:	20:17:13	Log-Likelihood:	-5647.4
No. Observations:	17544	AIC:	1.130e+04
Df Residuals:	17542	BIC:	1.131e+04
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	7.3087	0.018	401.982	0.000	7.273	7.344
pay_per_mile	0.7375	0.005	154.364	0.000	0.728	0.747

Omnibus:	468.184	Durbin-Watson:	0.281
Prob(Omnibus):	0.000	Jarque-Bera (JB):	452.451
Skew:	-0.356	Prob(JB):	5.64e-99
Kurtosis:	2.666	Cond. No.	29.3

Figure 46 Demand Curve- Log-OLS regression 2

Key findings from the Log-OLS model:

- **Coefficient Interpretation**: The coefficient on pay_per_mile is **+0.7375**, also highly significant. This suggests that for each additional unit of pay per mile, the log of order count increases by approximately 0.74 units.
- **Model Fit**: R^2 = **0.576**, slightly lower than the linear OLS model, but still strong for behavioural data. Log transformation offers better stability in the presence of right-skewed data.
- **Business Implication:** This log specification enables interpretation in terms of growth rates, indicating that price elasticity is both measurable and positive.

3. Model Comparison and Conclusion

Both the linear OLS and log-transformed OLS models reveal a **strong, statistically significant, and economically meaningful positive relationship** between driver pay per mile and order count. The linear model highlights the absolute effect of unit price on order volume, showing that higher compensation can result in thousands more completed trips per hour. The log-OLS model, on the other hand, provides a more nuanced view of relative responsiveness, particularly useful in interpreting elasticity across differently scaled regions.

While the linear OLS model offers greater interpretability in practical terms, the log-linear model delivers better statistical robustness and is more appropriate when dealing with highly skewed demand distributions. Together, they confirm that **price elasticity is not only present but strategically actionable**, reinforcing the value of using price-based levers to influence driver supply and system-wide efficiency.

9.1.2.3 ARIMAX

To further capture the dynamic relationship between pricing and demand over time, we implemented an ARIMAX (AutoRegressive Integrated Moving Average with Exogenous variables)

model. This approach enables the modelling of temporal autocorrelation in hourly order counts while accounting for **pay-per-mile** as an external regressor.

The model was applied to the **hourly order data from January 1 to January 30, 2023**, using the following specification:

Dependent Variable: order_count **Exogenous Regressor**: pay_per_mile

Model Order: ARIMAX(1, 0, 1)

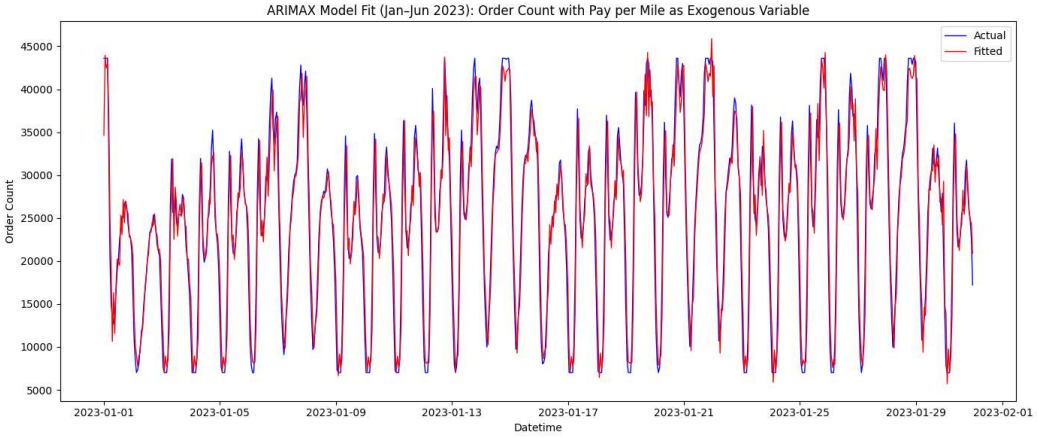


Figure 47 Demand ARIMAX

ARIMAX Result: To incorporate temporal dynamics into our price elasticity analysis, we implemented an ARIMAX(1,0,1) model using hourly order data from January 1 to January 30, 2023, with driver pay per mile as the exogenous variable. The model effectively tracks short-term fluctuations in demand, as shown by the close alignment between actual and fitted values across time.

Business Implication: From a business perspective, the model highlights the operational sensitivity of demand to real-time pricing signals. As pay-per-mile increases, the system consistently fulfils more ride requests, reinforcing that targeted pricing adjustments can directly influence matching efficiency and supply responsiveness on an hourly basis.

9.1.2.4 Model Selection

Among the three models tested, the standard OLS regression was ultimately selected for downstream analysis due to its strong explanatory power ($R^2 = 0.603$), intuitive interpretability, and robustness after winsorization. While the log-transformed model offered marginal variance reduction and ARIMAX captured temporal patterns, the linear OLS best balances accuracy, transparency, and applicability for scenario-based planning.

9.1.3 Grouped Regression Analysis by Hour

To uncover how price elasticity varies across different times of day, we conducted a series of hourly regressions using standardized variables. By estimating the relationship between driver pay per mile and order count separately for each hour, we aim to capture the dynamic responsiveness of demand to pricing signals over a typical 24-hour cycle. This allows us to identify which periods exhibit stronger or weaker elasticity—critical for designing time-sensitive pricing strategies.

9.1.3.1 Regression Result

The results indicate that the positive relationship between pay per mile and order volume is consistently present across all hours, though the strength of this relationship varies. The slope of the fitted regression line—representing standardized elasticity—is generally higher during daytime hours (e.g., 7 AM to 9 PM), reflecting greater driver responsiveness and demand fulfilment sensitivity to price during active business and commuting periods. In contrast, overnight hours (e.g., 0–5 AM) show weaker but still positive associations, likely due to lower baseline activity and fewer available drivers.

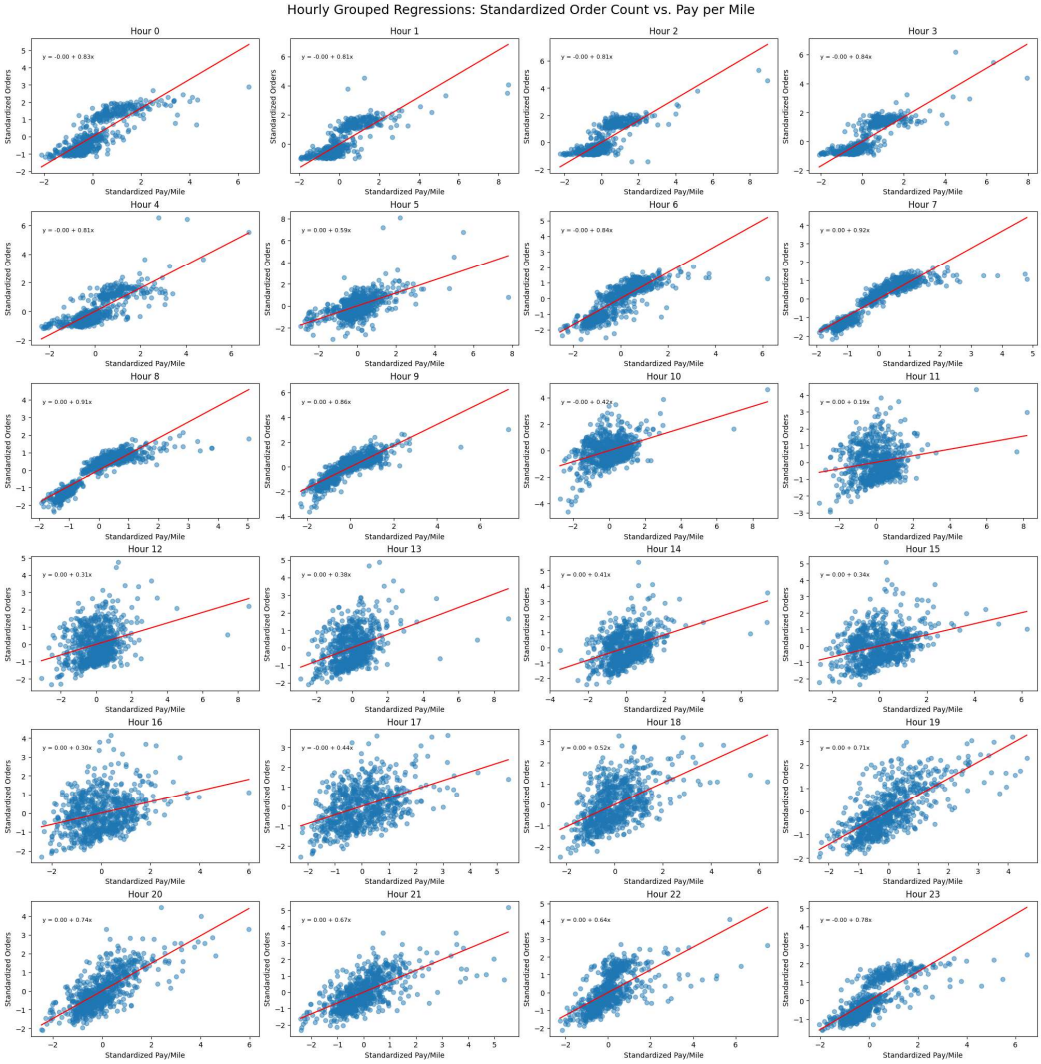


Figure 48 Demand Curve Regression by Hour

9.1.3.2 Business Implication

Hour 8 & 9: High Elasticity (β ≈ 0.9)

Strong positive responsiveness indicates highly effective driver-passenger matching. **Recommended action:** Moderate price increase to capture demand while preserving service quality.

Hour 19 & 21: Moderate Elasticity (β ≈ 0.6)

Pricing still affects demand, though responsiveness is less intense than in peak hours.

Recommended action: Slight upward price adjustment may improve matching without deterring riders.

Hour 11 & 12: Low Elasticity (β ≈ 0.2)
Weak sensitivity to price signals suggests diminishing returns from pricing adjustments.
Recommended action: Maintain stable pricing to avoid unnecessary fluctuations in revenue or supply balance.

9.1.4 Grouped Regression Analysis by Location

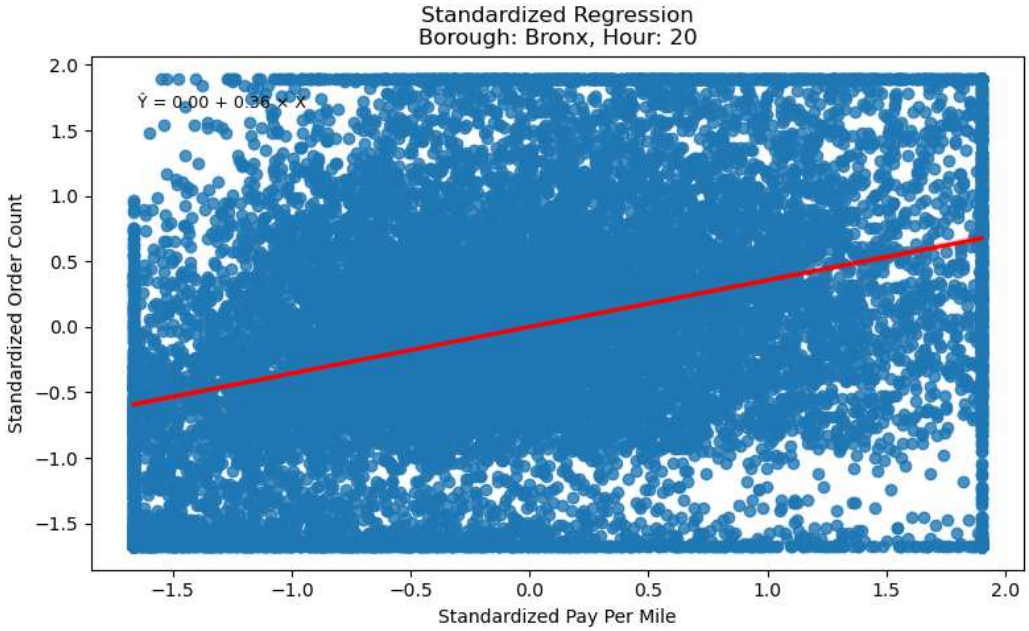


Figure 49 Demand Curve by Location

Regression by location proves impractical not only due to the absence of clear distributional patterns, but also because platforms like Uber do not differentiate pricing based on pickup location alone. From a business standpoint, location-based pricing adjustments are rarely implemented, as demand is influenced more by time and dynamic supply conditions than by static geography. This limits the operational value of location-segmented elasticity analysis.

Instead of forcing a regression framework, alternative analyses such as ANOVA or clustering based on latent demand patterns across locations could provide more actionable insights without over-interpreting weak linear trends.

9.1.5 Borough-Wise Variation in Driver Pay

A one-way ANOVA test was conducted to evaluate whether average driver pay varies significantly across the five NYC boroughs. The analysis confirmed a statistically significant difference (p-value < 0.05), indicating that driver earnings are not uniform across locations. Manhattan reported the highest average pay, while Staten Island had the lowest.

A supporting bar chart visualized these differences, revealing clear disparities in compensation likely driven by variation in trip volume, fare rates, and rider density. These findings suggest that borough-

level dynamics should be considered when designing driver incentives and regional pricing strategies.

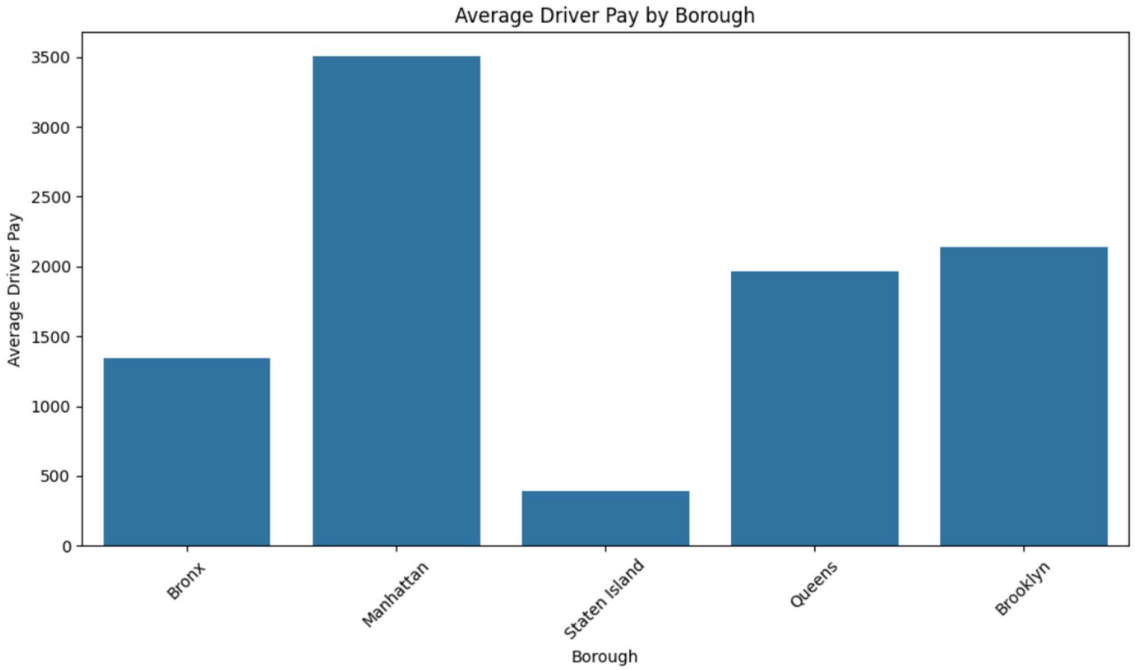


Figure 50 Average Driver Pay by Borough in New York City

9.1.6 Monthly Variation in Driver Pay Across Boroughs

To evaluate how driver pay varied by borough over time, a one-way ANOVA test was conducted separately for each calendar month. The analysis revealed consistent statistical differences in average driver pay between boroughs throughout the year.

A line chart was used to visualize these trends, showing that Manhattan consistently maintained the highest driver pay, with noticeable seasonal variation. Other boroughs, such as Brooklyn and Queens, showed gradual increases over the months, while Staten Island remained the lowest throughout. These findings emphasize the importance of incorporating both spatial and temporal dynamics into driver compensation strategies.

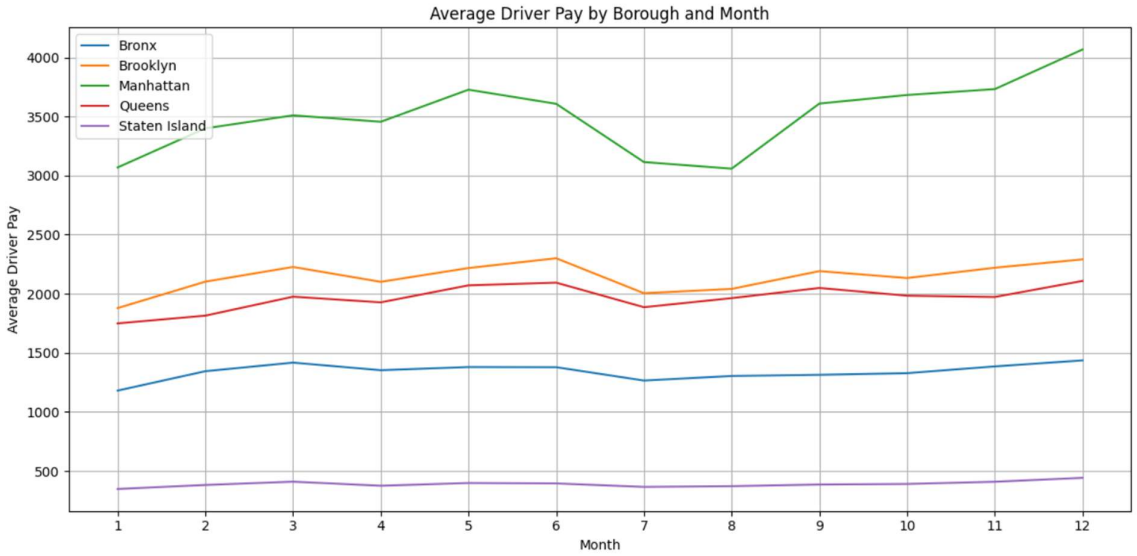


Figure 51 Average Driver Pay by Borough and Month in NYC

9.1.7 Hourly Demand Segmentation via Quantile Analysis

To identify structurally distinct periods of demand throughout the day, we conducted an hourly analysis of average ride orders. This segmentation supports the development of differentiated pricing strategies suited to specific demand environments.

Methodology

We calculated the average number of ride orders (`order_count`) for each hour (0–23). To classify hours based on demand intensity, we applied quantile analysis:

- Off-Peak Hours: Bottom 25% of hourly demand
- Normal Hours: Middle 50%
- Peak Hours: Top 25%

This classification avoids subjective thresholds and is grounded in the statistical distribution of demand values.

Analytical Procedure: Hourly Demand Segmentation

1. Aggregate the dataset by hour to calculate the average ride demand per hour.
2. Determine quantile boundaries using the 25th (Q1) and 75th (Q3) percentiles of demand.
3. Assign hour segments:
    a. If average demand < Q1 → label as Off-Peak
    b. If average demand ≥ Q3 → label as Peak
    c. Otherwise → label as Normal
4. Attach segment labels to each record for downstream modelling.

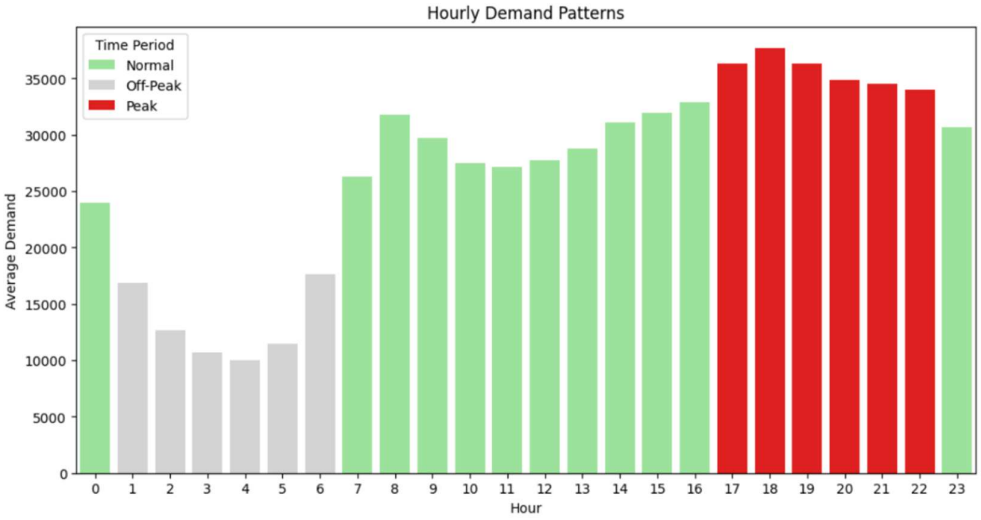


Figure 52 Hourly Demand Classification into Off-Peak, Normal, and Peak Hours using Quantile Analysis

Insights

- Demand follows a clear daily cycle with identifiable low and high periods.
- Off-Peak hours represent an opportunity for incentive-based interventions.
- Peak hours may require operational or non-price-based strategies due to supply saturation.

9.1.8 Demand Curve Estimation: Modelling Elasticity

Following segmentation, we estimated how responsive demand is to driver pay - a measure of price elasticity of supply. Specifically, we examined how changes in `pay_per_mile` affect the number of ride orders fulfilled in each demand period.

Methodology

Separate linear regression models were fitted for each of the three time-based segments (Peak, Normal, Off-Peak). The relationship was modelled as:

$$\text{Order Count} = \alpha + \beta \times \text{Pay per Mile}$$

Where:

- α: Intercept — base order count (theoretically at zero pay)
- β: Slope — elasticity coefficient, showing the change in order count for a unit change in driver pay

Analytical Procedure: Demand Curve Estimation

1. Filter the data for each group: Peak, Normal, and Off-Peak.
2. Define variables:
    a. Independent variable: pay per mile

   b. Dependent variable: order count
3. Fit a linear regression using Ordinary Least Squares (OLS) for each group.
4. Extract model parameters:
   a. Intercept
   b. Slope
   c. R-squared value (for goodness of fit)
5. Store elasticity values for use in pricing optimization.
6. Visualize the estimated demand curves for comparison.

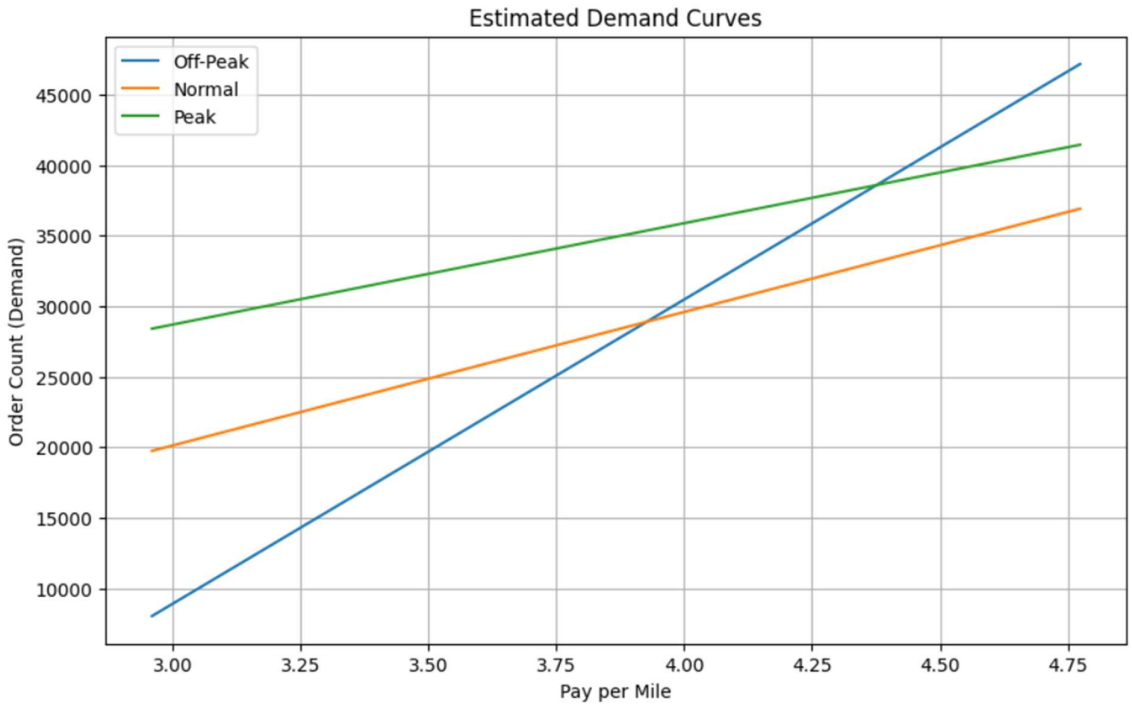


Figure 53 Estimated Demand Curves for Different Time Periods (Off-Peak, Normal, Peak)

9.1.8.1 Interpretation and Business Insights

1. Elasticity Varies Significantly by Time Segment

Time Period	Elasticity (Slope)	Interpretation
Off-Peak	High (Steep slope)	Small increases in pay lead to large increases in accepted orders
Normal	Medium	Pay moderately affects driver responsiveness
Peak	Low (Flat slope)	Additional pay does not significantly increase order fulfilment

Table 13 Elasticity by Time Segment

These differences confirm that one-size-fits-all pricing strategies are suboptimal. Instead, elasticity-informed pricing can better match supply and demand conditions.

2. Role in Optimization

The slope coefficients from the regression models are directly used as weights in the objective function of the Linear Programming model. This ensures that price adjustments are economically justified and targeted to where they are most effective.

3. Operational Interpretation

Positive slopes across all segments confirm a supply-constrained system: higher pay attracts more drivers. However, the return on incentive investment is maximized during Off-Peak hours, where responsiveness is greatest.

9.1.8.2 Summary

- A quantile-based analytical framework was applied to categorize demand into Off-Peak, Normal, and Peak periods.
- Regression modelling identified distinct elasticities for each period, quantifying how sensitive ride acceptance is to driver pay.
- This elasticity information serves as the mathematical input to the Linear Programming optimization in the next phase, allowing the system to recommend prices that are both data-driven and context-specific.

9.1.9 Hourly Revenue Sensitivity to Price Adjustments

To identify time periods with the highest potential for revenue improvement, a regression-based sensitivity analysis was performed across hourly segments. The model estimated the change in revenue resulting from a hypothetical 5% increase in pay-per-mile, using standardized linear regression for each hour.

The analysis revealed that early morning hours (0–2) and the morning commute hour (8 AM) showed the greatest potential for revenue uplift. These hours exhibited high responsiveness to price changes, suggesting that targeted rate increases during these periods could drive significant revenue gains.

The bar chart visualizes the top 10 hours with the highest estimated revenue change, offering clear operational insight into when price adjustments could be most effective.

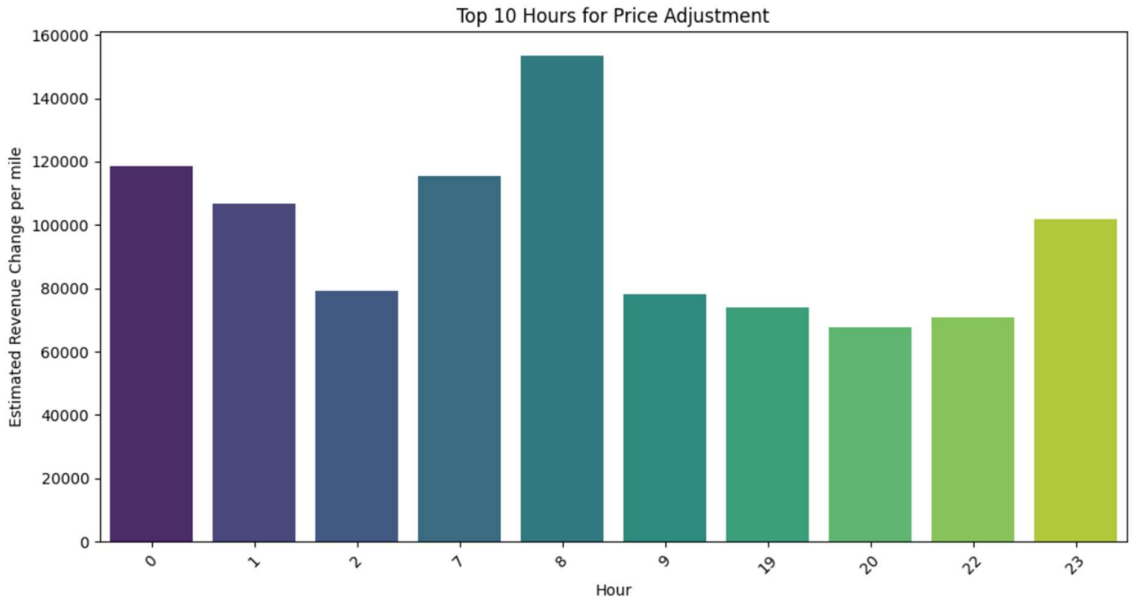


Figure 54 Top 10 Hours with Highest Estimated Revenue Change from Price Adjustment

9.2 Linear Programming

9.2.1 Introduction

In this section, we implement Linear Programming (LP) to optimize taxi driver pay-per-mile pricing. Our objective is to maximize the total hourly revenue by adjusting prices within specific constraint limits. This builds directly on earlier analysis of hourly demand and demand elasticity, allowing for data-driven, operationally feasible price recommendations.

To ensure price stability and fairness, constraints were applied to limit the maximum allowable increase or decrease in pay-per-mile prices.

9.2.2 Technical Implementation

The linear programming (LP) framework for pricing optimization is structured systematically in several stages.

- First, we define an objective function aimed at maximizing total revenue by strategically adjusting driver pay-per-mile rates. Since the optimization solver is designed for minimization tasks, we reformulate the objective to minimize the negative of revenue.
- Next, we establish a series of constraints to ensure that optimized prices stay within acceptable operational limits. These constraints prevent excessive price fluctuations and maintain service stability.
- The demand elasticity coefficients (slopes) derived from earlier regression analysis are directly incorporated into the optimization model, ensuring that price adjustments are sensitive to real-world driver response behaviour.
- Finally, the formulated LP problem is solved using SciPy's `linprog` function with the HiGHS algorithm, which efficiently handles large, sparse linear programming problems. The

resulting optimized prices are compared against initial prices to evaluate the effectiveness of the pricing adjustment under different constraint scenarios (5%, 10%, 15%, and 20%).

9.2.2.1 Defining the Objective Function

- Objective: Maximize total revenue across all hours.
- Since linear programming solvers minimize by default, we minimize the negative of the revenue function.
- Revenue is influenced by demand elasticity (sensitivity of order volume to pay-per-mile).

The **objective function** is mathematically formulated as:

$$\text{Minimize } Z = -\sum_i (\text{slope}_i \times x_i)$$

where:

- x_i = optimized price (pay-per-mile) for time period i
- slope_i = price elasticity coefficient for period i

Here, slopes are derived from the earlier regression analysis performed for each demand segment (Peak, Normal, Off-Peak).

9.2.2.2 Setting the Constraints

To ensure price stability and limit sharp fluctuations:

- **Non-Negativity Constraint:** Prices must be positive.
- **Price Adjustment Constraints:**
  - Upper Bound:

$$x_i \leq \text{initial price}_i \times (1 + \text{max\_increase\_percent})$$

  - Lower Bound:

$$x_i \geq \text{initial price}_i \times (1 - \text{max\_decrease\_percent})$$

Different scenarios were tested by varying the maximum allowable price change limits: 5%, 10%, 15%, and 20%.

9.2.2.3 Solving the Linear Programming Problem

Once the objective function and constraints were formulated, the next critical step was to solve the linear programming (LP) problem efficiently and accurately.

We implemented the solution using the following structured approach:

Step 1: Preparation of Inputs

- **Objective Coefficients:**
  We prepared a list of coefficients representing the negative slopes from the demand elasticity analysis. These coefficients capture the sensitivity of revenue to price changes for each time group (Peak, Normal, Off-Peak).

- **Constraint Matrices:**
  Two sets of inequality constraints were established:
  - **Upper Bound Constraints:** Limiting the maximum increase in price for each group.
  - **Lower Bound Constraints:** Limiting the maximum decrease in price for each group.

  Together, these constraints ensure that the optimized prices remain within an operationally acceptable range.

- **Bounds on Variables:**
  Additionally, each price was constrained to be non-negative, ensuring that no price falls below zero.

Step 2: Problem Formulation

- The problem was formulated as a standard linear programming minimization problem, where:
  - The objective was to minimize the negative weighted sum of the optimized prices.
  - The constraints guaranteed that each optimized price stayed within both the upper and lower limits set around its initial value.
- This precise formulation enables the LP solver to find the optimal price vector that respects all operational rules.

Step 3: Selection of Solver

- To solve the LP problem, we utilized a state-of-the-art solver called the HiGHS algorithm, accessed through the `linprog` functionality of the optimization library.
- HiGHS is a modern optimization engine specifically designed for solving large-scale and complex linear programming problems quickly and accurately.

Step 4: Execution of the Solver

- All inputs - the objective function coefficients, constraint matrices, and bounds, were fed into the HiGHS solver.
- The solver processed these inputs and attempted to find the set of optimized prices that minimize the objective function while satisfying all constraints.
- If the solver successfully converged:
  - The optimized pay-per-mile prices were extracted.
  - These optimized prices were then assigned back to the corresponding hourly time groups.

- If the solver failed to converge (which did not occur in our case), the system would report an error, and no price optimization would be applied.

Step 5: Post-Solution Processing

- After solving, we computed:
  - The initial total revenue based on original prices.
  - The optimized total revenue based on newly optimized prices.
- We then calculated the percentage improvement in revenue, allowing us to quantitatively evaluate the impact of the optimization.
- Finally, we plotted and compared the hourly revenue before and after optimization under different constraint scenarios (5%, 10%, 15%, 20%).

9.2.3 Results and Analysis

The LP model was applied under four different price constraint scenarios:

Constraint	Observations
5%	Revenue optimization was modest. In some hours, optimized revenue was lower than initial revenue.
10%	Similar pattern as 5%, with some negative improvements in a few hours.
15%	Optimized revenue exceeded original revenue across all hours. Larger improvements were observed.
20%	Highest revenue gains. However, price changes might be perceived as aggressive by drivers.

Table 14 Linear Programming Contraints

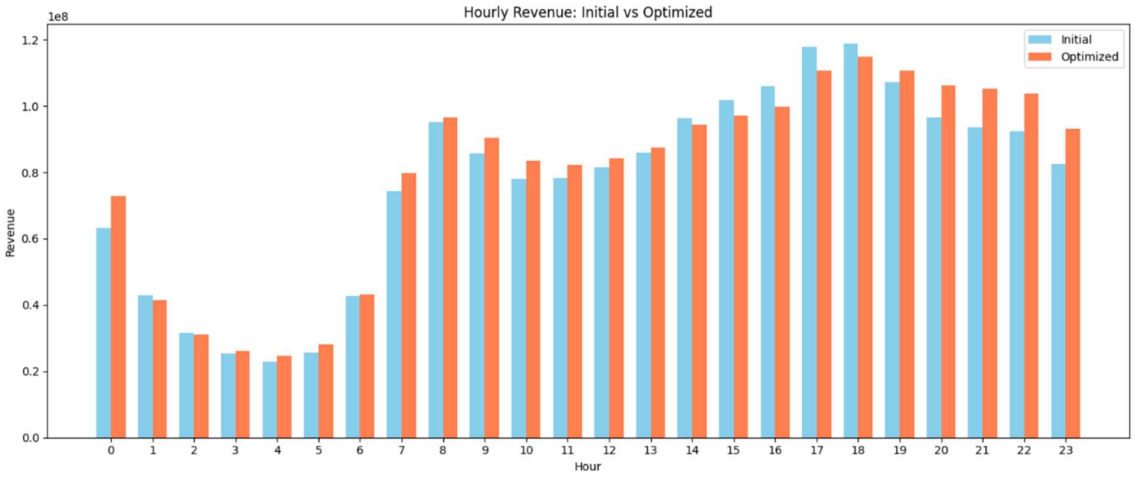


Figure 55 Hourly Revenue Comparison: Initial vs Optimized (5% Constraint)

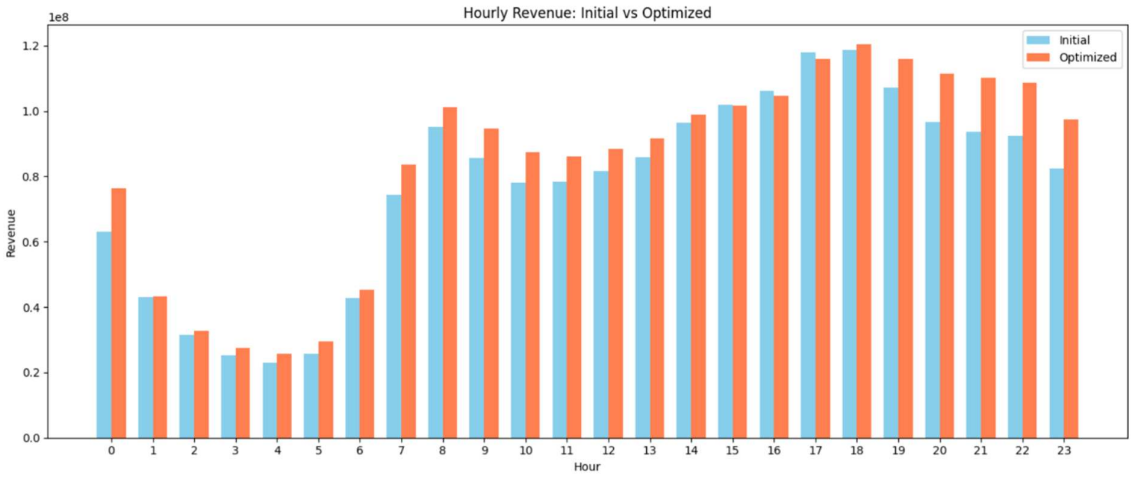


Figure 56 Hourly Revenue Comparison: Initial vs Optimized (10% Constraint)

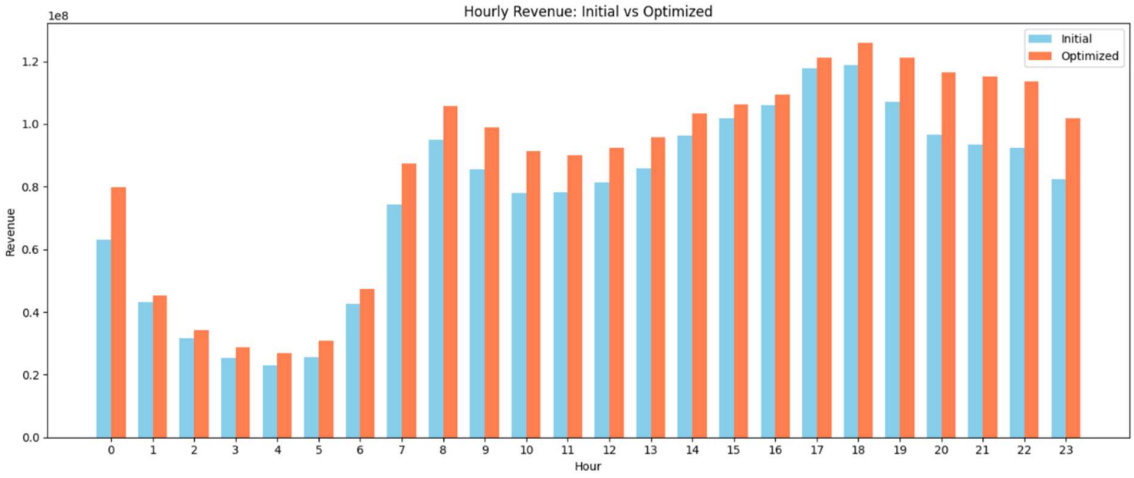


Figure 57 Hourly Revenue Comparison: Initial vs Optimized (15% Constraint)

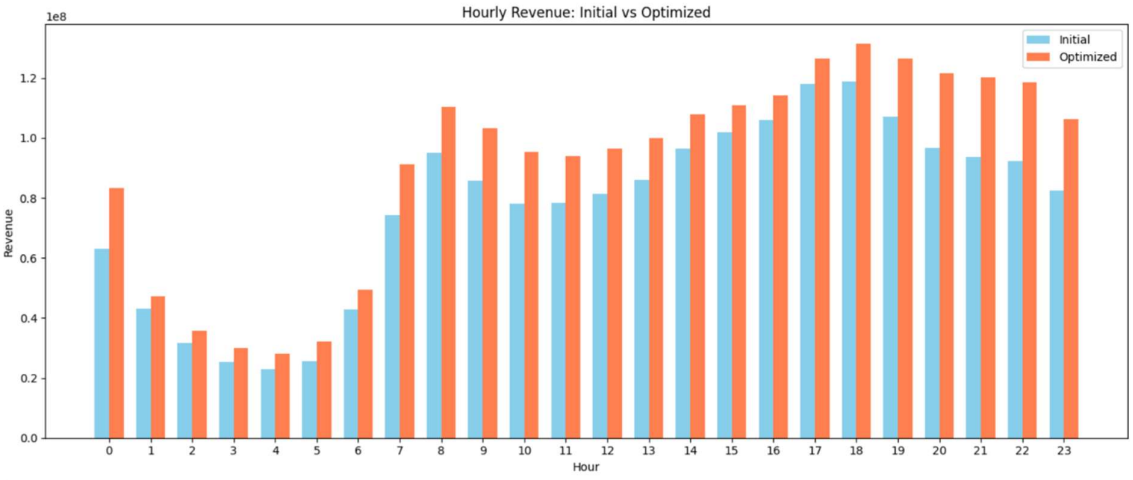


Figure 58 Hourly Revenue Comparison: Initial vs Optimized (20% Constraint)

9.2.4 Key Findings

- 5% and 10% constraints:
  - Too restrictive. Some hours showed a decline in revenue compared to initial pricing.
- 15% constraint:
  - Optimal balance between revenue improvement and pricing stability.
  - Revenue increased consistently across all hours.
- 20% constraint:
  - Higher revenue gains.
  - However, it risks sudden price hikes that may affect driver or rider satisfaction.

Thus, allowing up to a 15% adjustment was identified as the most balanced strategy.

9.2.5 Conclusion

Linear programming successfully identified optimized price points across different hours, resulting in overall revenue improvement.
The elasticity-based objective function ensured that price adjustments were sensitive to real-world driver behaviour and market dynamics.

Key Success Factors:

- Incorporation of real elasticity estimates from demand curve modelling.
- Application of tight but flexible constraints to ensure operational feasibility.
- Use of modern LP solvers (HiGHS algorithm) for efficient and accurate optimization.

This methodology provides a robust framework for dynamic pricing adjustments in the NYC taxi industry, offering significant business value.

10. Key Deliverables

A. Driver Allocation Optimisation:

1. Time Series Predictive Model - Forecast future taxi order volumes and demand across various time periods and locations.
   - Forecasting model results, segmented by time of day and location.
2. Survival analysis model - Predict order completion efficiency, considering location, time periods, and other operational factors.
   - Survival analysis results showing order completion probabilities across various conditions (e.g., time of day, route types).
   - Efficiency prediction metrics, such as expected completion time and probability of successful completion.
3. Driver Allocation Optimization
   - Monte Carlo and linear programming model to determine optimal fleet allocation

B. Pricing Optimisation:

Pricing optimization model – Maximise Revenue by adjusting Taxi fare pricing

- Demand elasticity analysis to determine the relationship between order volume and price per kilometer.
- Linear programming model for optimizing price adjustments based on demand forecasts.

11. Key Outcomes/ Results of Models /Discussion & Analysis

A. Driver Allocation Optimisation:

1. Time Series: SARIMAX
   - Despite thorough analysis and incorporating exogenous factors, the SARIMAX model proved to be unsatisfactory for predicting the orders as it scored worse than the benchmark.
   - The SARIMAX model was also incapable of generalising and predicting for multiple locations.

2. Time Series: LSTM
   - To prepare the dataset for LSTM, careful preprocessing, feature engineering and sequencing were performed.
   - The LSTM model architecture and parameters were carefully chosen after many iterations, improving on the results each time.
   - The LSTM model was successful in predicting orders for next hour as well as next day, for all the locations across Manhattan Borough. Overall, it performed > 25% better than benchmark.

3. Survival Wait Time Analysis
   - Survival analysis model the duration between ride requests and successful pickups, including both completed and right-censored cases since we assume that censored pickups will eventually happen.
   - Kaplan-meier provides non-parametric survival curves to explore group-level differences with significant logrank test statistic result.
   - Cox proportional hazard model observes the relationship between important factors and pickup probabilities.
   - Aalen's additive model provides time-varying effects which are not captured by CoxPH. It uncovers factors' performance changes dynamically. It provides a dynamic perspective and algins with real-world situations.

4. Monte Carlo Simulation and Linear Programming

This study leverages historical 9:00 AM order distributions for each pickup location across Manhattan to simulate future demand via Monte Carlo methods. These simulated order volumes were then integrated into a linear programming model aimed at optimizing driver allocation across regions, with the objective of maximizing total revenue.
The optimization delivered a significant improvement in revenue—from $42,763.68 to $52,526.41, representing a 22% increase. The model allocates more drivers to high-marginal-return zones, while slightly reducing allocation in less efficient areas. Furthermore, the model provided a driver allocation portfolio that yields maximum expected revenue, further enhanced the operational efficiency and resource utilization.

B. Pricing Optimisation:

1. Price Elasticity

Overall OLS Regression: We applied both standard and log-OLS regressions. The standard OLS model showed strong explanatory power ($R^2 = 0.603$), with a significantly positive coefficient, confirming that increasing driver pay effectively boosts order completion.

Hourly Grouped OLS: Hourly regressions on standardized variables reveal that elasticity varies across the day. It peaks during morning rush hours (e.g., 8–9 AM), remains moderate in the evening, and is weakest overnight—offering guidance for time-sensitive pricing strategies.

Location Grouped OLS: Location-level regressions proved insignificant result due to the lack of geographically differentiated pricing in Uber and the absence of strong spatial demand patterns.

Segmented OLS by Demand Periods: Using quantile analysis, we segmented the day into Off-Peak, Normal, and Peak periods and modelled each separately. Results show the highest elasticity during Off-Peak hours—small pay increases lead to large order gains—making it ideal for incentive-based allocation. These coefficients directly informed the objective function of our LP model.

2. Linear Programming

The Linear Programming (LP) model provided a structured, data-driven framework to optimize pricing strategies by incorporating demand elasticity into the objective function. By segmenting demand into Off-Peak, Normal, and Peak hours using quantile analysis, and estimating price sensitivity through regression, we ensured that each pricing decision was grounded in actual behavioural patterns.

Key Insights:

- **Effective Revenue Maximization:** The LP model consistently outperformed baseline revenue figures across all constraint levels. Even under conservative adjustment scenarios, optimized pricing led to notable gains in hourly revenue, highlighting the efficacy of using elasticity-driven adjustments.
- **Elasticity-Informed Adjustments:** The optimization algorithm directed larger pay-per-mile increases during Off-Peak hours, where demand was most responsive to price. Conversely, it applied more conservative changes during Peak periods, where elasticity was low. This validated the model's alignment with supply-side responsiveness.
- **Operational Practicality:** The model ensured that pricing changes remained within realistic operational bounds, avoiding abrupt pay fluctuations. This is crucial for maintaining driver satisfaction and avoiding adverse market reactions.
- **Scalability and Flexibility:** By using a modular setup that integrates seamlessly with previous elasticity analyses, the LP model can be updated easily with new elasticity estimates or business constraints, making it adaptable to real-time pricing needs.

Overall, the LP approach demonstrates a robust and interpretable way to optimize ride-hailing pricing dynamically, ensuring revenue growth while respecting operational constraints and driver behaviour patterns.

12. Conclusions

A. Driver Allocation Optimisation

1. Successful built 3 models that can help in Driver Allocation Optimisation
   - Time Series model to predict next hour and next day order.
   - Wait time analysis to understand factors that influence wait time.
   - Monte Carlo Simulation and Linear programming model to optimize fleet allocation for revenue across a single hour.
2. These models can potentially be used to:
   - Maximize demand fulfillment and overall revenue by matching order volumes across locations
   - Relocate drivers at the right time to locations with longer waiting times, improving customer satisfaction
   - Serve locations with higher demand, thus increasing total revenue

B. Pricing Optimisation

1. The analysis shows that optimizing prices can increase revenue, but the amount of increase depends on how much prices are allowed to change.
2. Finding the right balance is key: It is essential to identify an optimal pricing portfolio, adjusting prices in high-margin segments while maintaining stability in low-margin ones, to maximize driver responsiveness and unlock greater revenue potential.
3. Price optimization affects different times of day differently, so it's important to consider time-based pricing strategies.

13. Future Recommendations

A. Driver Allocation Optimisation:

1. Consider including other factors like weather and traffic conditions to help predict future order amounts
2. Increasing scope of analysis
   - The Time Series LSTM model, as well as Survival Time analysis, did not cover the whole of NYC. The other 4 Boroughs can be similarly analyzed for a more complete analysis.
   - Mote Carlo Simulation can similarly cover the whole of NYC, as well as analyze more time periods of interest to provide more comprehensive coverage.
   - Survival models should add on more real-time traffic and weather data to have more accurate results.
3. Build a real-time optimized dispatch system based on models built.
   - Anticipating orders volume in each location
   - Relocate drivers to reduce waiting times, and serve locations with higher demand
   - Using Monte Carlo Simulation to prove effectiveness

B. Pricing Optimisation:

1. Develop a Real-Time Optimized Demand Monitoring System

   - Capture all attempted ride requests within the app—not just completed orders—to obtain a more comprehensive view of demand patterns.

- Monitor demand elasticity at a minute-level granularity and implement dynamic pricing adjustments based on real-time variations across different times and locations.

2. Enhance Elasticity-Aware Pricing Models

- Regularly update elasticity coefficients using recent data to ensure that the LP model adapts to shifts in driver responsiveness and demand conditions.

3. Enable Real-Time Optimization Deployment

- Integrate the LP model into a live pricing system to dynamically adjust hourly pay-per-mile rates based on current demand patterns and operational constraints.

14. Acknowledgements

We would like to extend our sincere gratitude to our lecturers for their dedication and the generous amount of time they have invested in our education. Their patient guidance and the wealth of real-life industry examples they shared have significantly enriched our learning experience, helping us understand complex concepts and their practical applications.

15. References

[1] NYC Taxi and Limousine Commission. (n.d.). Taxicab & Livery Passenger Enhancement Programs, NYC Taxi and Limousine Commission (TLC) Trip Record Data. Retrieved from https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page

[2] Schoenfeld, D. (2022). Survival Analysis: Part II – Multivariate Data Analysis. Indian Journal of Dermatology, Venereology and Leprology, 88(4), 435-444. https://doi.org/PMC11579528