Optimizing Urban Mobility Predictive Analytics for Ride-Hailing Demand Allocation

Contents

List of Figures

1 Introduction

A deep-rooted issue in the ride-hailing industry in Egypt is the mismatch between supply and demand. Ride-hailing platforms like Uber, InDrive, etc. offer their services of connecting drivers with customers; within the connection process arises the challenge of balancing the supply of drivers with the fluctuating demand, especially during rush hours, vacations, and public events. This project delves into Manhattan's bustling taxi activity to uncover patterns in passenger pickups using a data-driven approach. By leveraging key dataset attributes—pickup timestamps, longitude, and latitude—we aimed to identify temporal and spatial trends in taxi movement across the city. Data cleaning and preprocessing ensured the dataset's reliability, enabling the generation of geospatial aggregations and dynamic visualizations that highlight hotspots and flow trends. Our Python-based solution transforms raw data into actionable insights by creating a video visualization of passenger movement over time. The process involves filtering the dataset by user-defined time intervals, aggregating pickup locations through coordinate rounding for precision, and compiling the results into a time-evolved representation. These visual narratives offer valuable perspectives for urban planning, transportation optimization, and understanding Manhattan's dynamic rhythms.

The primary objective of this project is to uncover spatial and temporal patterns in NYC's taxi activy, analyze passenger movement trends, and develop predictive models for ride demand to enhance urban planning and transportation systems to simulate the optimization of ride-hailing applications. The goal is to predict areas including creating dynamic visualizations of passenger demand enhancing predictive accuracy through machine learning and evaluate how these tools can improve driver efficiency and customer experience without developing a full system.

2 Uncovering Manhattan's Taxi Patterns: A Data-Driven Exploration of Passenger Movement

To analyze and visualize the data we acquired about Manhattan's taxi pickups, we adopted a systematic approach centered on the dataset's key attributes: pickup timestamps, longitude, and latitude. By cleaning and preprocessing the data to ensure consistency and reliability, we aimed to uncover temporal and spatial patterns in passenger movement across Manhattan. Through geospatial aggregation and dynamic visualizations, this project transforms raw data into actionable insights, highlighting taxi activity hotspots and passenger flow trends. These visual narratives provide valuable perspectives for urban planning, transportation management, and understanding the city's dynamic rhythms in both space and time.

To achieve the dynamic visualization of Manhattan's taxi pickups, we developed a Python-based solution that processes the dataset, aggregates the data, and generates a video showcasing the evolution of passenger movement over time. The process begins by sorting the dataset based on timestamps and then extracting a specified time interval, which is defined by the user. Within this interval, the data is used to generate one frame of the video. For this, we implemented the following approach:

First, the dataset is filtered for a specific time interval, which is determined by the user. In this case, the time interval is one hour, but this can be adjusted as needed. The latitude and longitude values of the filtered data are extracted and stored in lists.

To improve the precision of the data aggregation, we applied a technique of rounding the coordinates. This is done by multiplying the latitude and longitude values by a factor 10^{LVL}, where LVL represents the level of accuracy desired. For example, setting $\text{LVL} = 3$ results in multiplying the coordinates by 1,000. Then, the `math.floor()` function is used to round down the multiplied values, and the values are divided by 1,000 to return them to their original scale but with reduced precision. This aggregation step allows us to group pickups occurring within small geographic regions.

The next step involves creating a dictionary, res, where each key is a tuple of latitude and longitude plus AVG_STEP, and the value represents the number of pickups that occurred at those coordinates. An average step value AVG_STEP is calculated as:

$$\text{AVG\_STEP} = \frac{0.5}{10^{\text{LVL}}}$$

This value is used to slightly offset the coordinates for better visualization. If a coordinate pair has been encountered previously, its count is incremented; otherwise, it is added to the dictionary with an initial count 1.

Example:

Latitude = 41.24565756, ________________________ **Longitude** = -73.**998032**, _____ **LVL** = 3:

- Latitude = $41.245657 \times 10^3 = 41245.657$, ______**Longitude** $= -73.998032 \times 10^3 = -73998.032$
- Latitude = $\text{floor}(41245.65756) = 41245$, _______**Longitude** $= \text{floor}(-73998.032) = -73998$
- Latitude = $41245/10^3 = 41.245$, _____________**Longitude** $= -73998/10^3 = -73.998$
- AVG_STEP = $0.5/10^3 = 0.0005$
- Latitude = $41.245 + 0.0005 = 41.2455$, ________**Longitude** $= -73.998 + 0.0005 = -73.9985$
- res = $\{(41.2455, -73.9985) : 1\}$

Where these values represent all values that lies between Lat [41.245, 41.246[and Lon [-73.998, -73.999[and 1 is the number of pickups in this area.

Finally, the results are stored in a DataFrame, results_df, which contains the aggregated pickup data ready for visualization. This cleaned and aggregated data will be used to create a single frame for the dynamic density maps, where the density of taxi pickups is visualized on a map of Manhattan.

To visualize this data, we utilize Plotly and OpenStreetMap to generate a density map shown in figure 1 for each time interval. Each map reflects the aggregated data, with color gradients that indicate the intensity of taxi pickups in different areas of Manhattan. This process repeats for each defined time window, creating a sequence of frames that represent the evolving spatial and temporal patterns of taxi activity across the city.

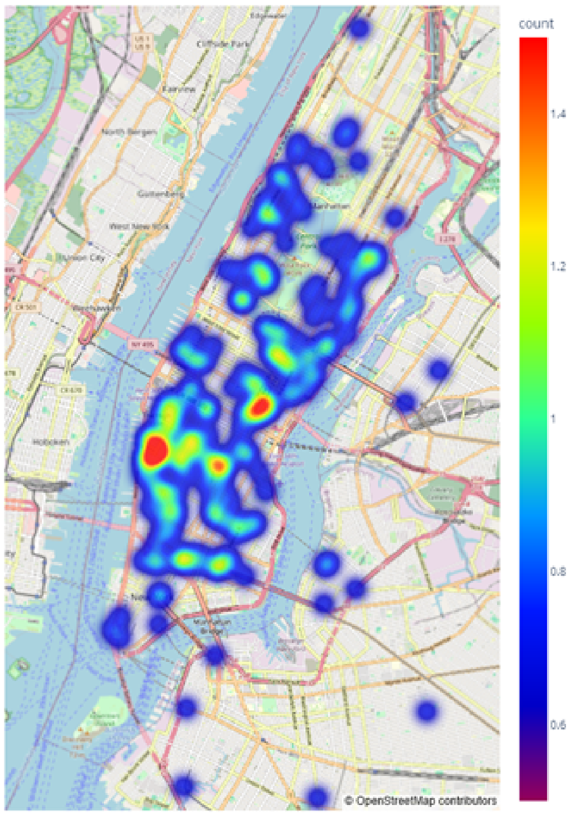


Figure 1: Density map

Once all frames are generated, they are compiled into a video using OpenCV, resulting in a compelling visual representation of how Manhattan's taxi pickups change over time. This video offers valuable insights into passenger movement and transportation trends across the city, making it an effective tool for urban planning, transportation analysis, and more. The video generation process concludes when the last frame is created, based on the end time specified by the user.

3 Data Preparation for Machine Learning and Descriptive Analysis

This section outlines the data preprocessing steps undertaken in Python using `pandas` to prepare the dataset for advanced analysis and machine learning. The focus was on cleaning, structuring, aggregating, and enriching the data to ensure its suitability for descriptive and predictive analytics.

1. **Data Selection and Transformation:**

   Data for each geographical region was selected from a dictionary and processed independently. The timestamps were standardized by converting the `pickup_datetime` column into a datetime format, enabling extraction of key components such as month, day, hour, and minutes for temporal analysis. Sorting the data ensured chronological consistency.

2. **Data Cleaning:**

   Unnecessary columns, including those related to drop-off details and passenger counts, were removed to streamline the dataset. This step focused the analysis on relevant features, such as temporal attributes and order counts.

3. **Aggregation and Grouping:**

   The data was grouped by month, day, and hour to calculate the total number of orders for each interval. This aggregation simplified temporal trend analysis and provided a basis for identifying patterns in order activity.

4. **Feature Engineering:**

   New features were added to enhance the dataset:

   - **Temporal Features:** A `date` column was created by combining month and day information, with the year set to a fixed value. The day of the week was extracted from this date for weekly trend analysis.
   - **Indicator Columns:** One-hot encoded columns were added for each day of the week and each hour of the day, enabling machine learning models to capture temporal patterns effectively.

5. **Missing Data Handling:**

   Missing time intervals were identified by generating a complete range of hourly timestamps within the dataset's timeframe. The existing data was merged with this comprehensive timestamp range, ensuring no time intervals were missing. Missing values in the order counts were filled with zeros, maintaining data integrity for subsequent analysis.

6. **Alignment with External Datasets:**

   The data was aligned with an external dataset (e.g., temperature data) to incorporate additional features. This alignment ensured consistency and enriched the dataset with external context, such as weather conditions.

7. **Encoding Temporal and Categorical Features:**

   After merging, redundant date columns were removed. The table was then sorted by month, day, and hour to ensure a logical sequence, and the dataset was reset for clean indexing. The processed data was stored back into a dictionary for each cell, enabling efficient access during subsequent analyses.

The preprocessing steps transformed raw, unstructured data into a structured and enriched dataset optimized for machine learning and descriptive analytics. The resulting dataset captures temporal and categorical trends, handles missing values comprehensively, and integrates external information to support robust and insightful analyses.

4 Regional Classification for Granular Analysis

To extend the analysis, the dataset was classified into distinct regions based on predefined geographic boundaries. By aggregating and analyzing data at the regional level, we aimed to uncover unique patterns and behaviors specific to each area, offering a more granular and actionable perspective.

4.1 Dividing Manhattan using QGIS Software

The goal is to divide Manhattan into its original regions to separate the data prediction to fit for every region of Manhattan. Dividing Manhattan into its regions required a few steps using the QGIS software and it goes as follows:

1. **Start a New Project:** Open QGIS and create a new project.
2. **Create Shapefile Layer:** Navigate to `Layer → Create Layer → Create Shapefile Layer`, name the file, and set the geometry type to "Line String" for delineating regions.
3. **Set CRS:** Choose `EPSG:4326-WGS 84` as the coordinate reference system (CRS) to include Manhattan.
4. **Draw Features:** Enable editing, use the "Add Line Feature" tool, and manually draw lines around Manhattan's regions for precise boundaries.
5. **Assign IDs:** Label each region with a unique ID.
6. **Export as GeoJSON:** Save the edited file and export it in the GeoJSON format.

Manhattan was divided into 5 different regions with their respective IDs where the IDs were numbered from 1 to 5 for the regions "Financial District", "Mid Town", "Upper East Side", "Upper West Side" and "Harlem" respectively. The regions were divided as shown in the following figure.

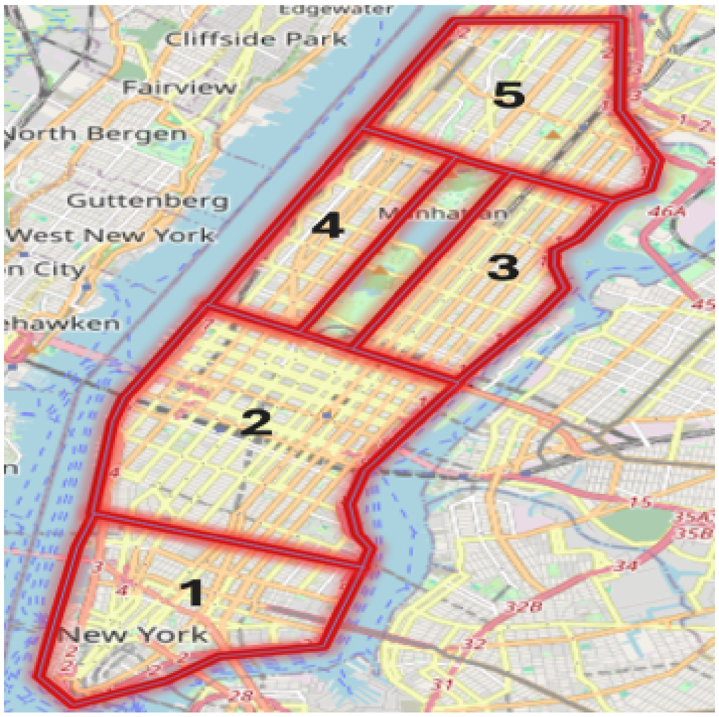


Figure 2: Division of Manhattan into regions

4.2 Transforming LineString to MultiPolygon Geometries and Assigning Regions

After dividing Manhattan into distinct regions using QGIS, we took the analysis a step further by transforming the initial GeoJSON files (which contained the district boundaries as LineString geometries) into MultiPolygon geometries. This transformation was crucial to ensure that the regions could be properly utilized for spatial analysis. Using Python, we employed the geopandas library to convert the LineString geometries into MultiPolygon shapes, which represent the actual areas of each district more accurately. The Python code utilizes the GeoJSON data, first converting it into a GeoDataFrame containing the district boundaries. Then, it uses a spatial join to determine which district each pickup point belongs to by checking if the pickup coordinates lie within the boundaries of each region. The resulting dataset now includes a new column identifying the district for each pickup point. To verify the accuracy of the region classifications, we used a density map visualization. The code generates a density map using **plotly** to visualize the distribution of pickup points across Manhattan's regions. This step allowed us to confirm whether the boundaries defined in the GeoJSON file were correctly applied and whether the regions contained appropriate data points.

4.3 Verifying Region Classifications

To verify the accuracy of the district classifications, a density map was created using `Plotly` to visualize the distribution of pickup points across Manhattan's regions. The density map allowed us to confirm whether the boundaries from the GeoJSON file were correctly applied and whether the pickup points were accurately assigned to their respective regions. By examining the distribution, we ensured that the predefined regions contained the appropriate data points.

4.4 Refining the Dataset Based on Regional Segmentation

After performing the spatial analysis and assigning each pickup point to its respective district, we proceeded to refine the dataset by handling any missing values and segmenting it based on district IDs. The following steps were carried out using R to manage the regions and prepare the data for further analysis:

1. **Loading and Inspecting the Data**
   First, we loaded the dataset containing the district IDs (`districts.csv`), which was generated during the spatial join. We used basic functions such as `View` and `summary` to inspect the dataset, understanding its structure and identifying any potential issues. Additionally, we checked the distribution of district IDs with the `table` function to confirm how well the data was classified into regions.

2. **Handling Missing Values**
   After inspecting the dataset, we identified some missing values in the `district_id` column. To address this, we replaced all missing district IDs with a default value (6), which was considered as a placeholder for undefined or unclassified districts.

3. **Segmentation by District**
   The dataset was then segmented into separate subsets based on the district IDs. This allowed us to focus on the analysis of each region individually, enabling more localized insights. We created separate dataframes for each district (Financial District, Midtown, Upper East Side, Upper West Side, and Harlem) by filtering the dataset for each district's ID.

4. **Saving the Results**
   Finally, we exported all of the region-specific datasets to a CSV file for further analysis or modeling.

By segmenting the data based on district IDs and handling missing values, we ensured that the dataset was clean, well-structured, and regionally specific. This segmentation lays the foundation for more granular, district-level analysis and ensures that any further machine learning models or descriptive analyses can be performed with greater accuracy and precision for each Manhattan district.

5 Descriptive Analysis: Visualization and Insights

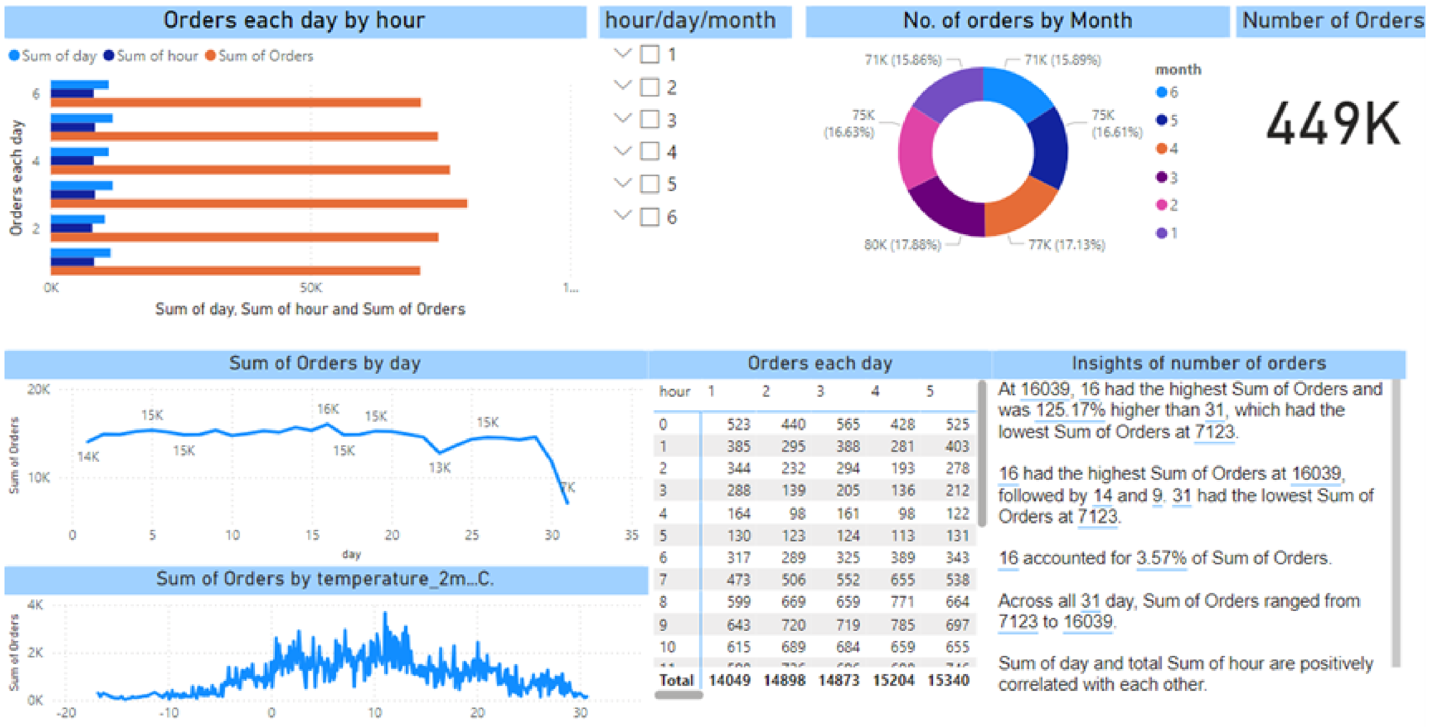
There were two approaches to create dashboards to visualize the data and obtain insights using Power BI and Tableau.

5.1 Dashboard Representation using Power BI

Developed interactive dashboards for each city in the dataset: Harlem, Financial District, Midtown, Upper West Side, and Upper East Side. Every dashboard has been carefully developed to offer comprehensive insights into the ride-hailing data over a six-month period.

Key Features:

- The number of orders for each of the six months is displayed on the dashboards, broken down by day and then by hour. Using a slicer can dynamically filter and examine data by month, day, or hour.
- A table-format matrix displays precise hourly statistics, making it simple to comprehend and compare the quantity of orders over time.
- A doughnut chart shows an aggregated picture of order distributions over six months. When a specific month is selected, the chart automatically refreshes to provide a more in-depth view of monthly patterns and insights.
- A card element clearly displays the total number of orders, giving consumers quick access to this essential data.
- A line chart shows the association between temperature changes and the number of orders. This visualization aids in understanding how weather conditions affect ride-hailing demand.
- Power BI's AI function generates narrative insights based on the dataset. This functionality provides short, actionable observations, allowing stakeholders to draw relevant conclusions directly from the data without requiring further manual interpretation.



Orders each day

hour	1	2	3	4	5
0	523	440	565	428	525
1	385	295	388	281	403
2	344	232	294	193	278
3	288	139	205	136	212
4	164	98	161	98	122
5	130	123	124	113	131
6	317	289	325	389	343
7	473	506	552	655	538
8	599	669	659	771	664
9	643	720	719	785	697
10	615	689	684	659	655
Total	14049	14898	14873	15204	15340

Insights of number of orders

At 16039, 16 had the highest Sum of Orders and was 125.17% higher than 31, which had the lowest Sum of Orders at 7123.

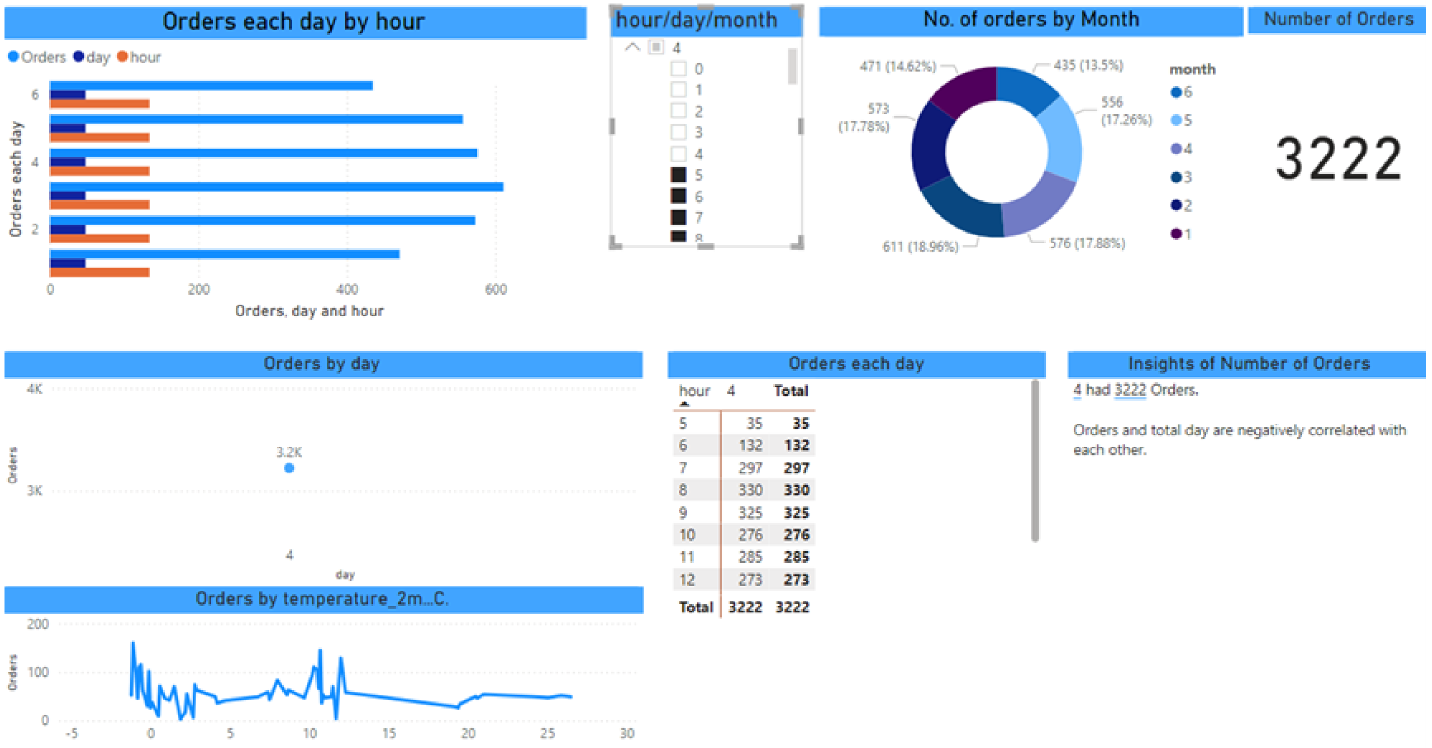
16 had the highest Sum of Orders at 16039, followed by 14 and 9. 31 had the lowest Sum of Orders at 7123.

16 accounted for 3.57% of Sum of Orders.

Across all 31 day, Sum of Orders ranged from 7123 to 16039.

Sum of day and total Sum of hour are positively correlated with each other.

1- Here is a dashboard of Mid-Town representing all six month results and their insights. We can show the insights of whatever day we need.



Orders each day

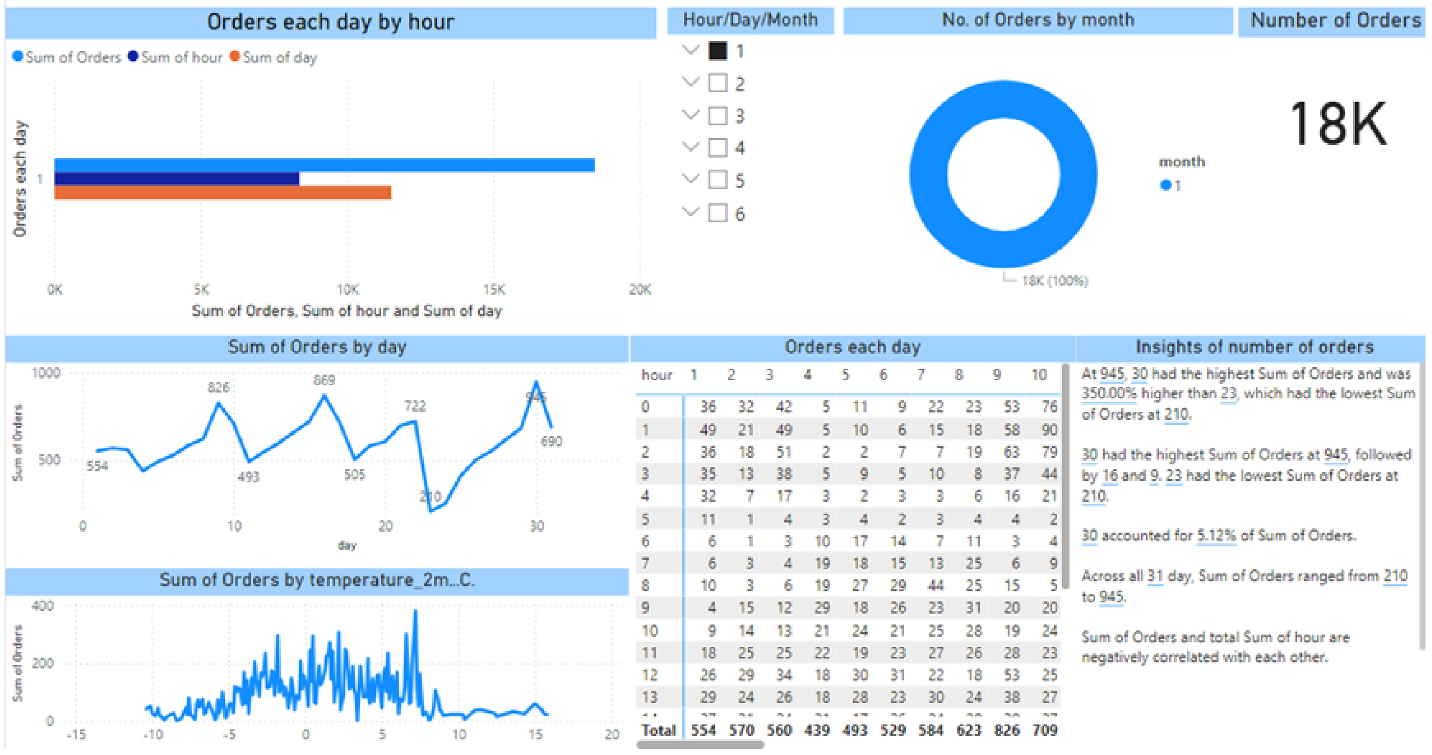
hour	4	Total
5	35	35
6	132	132
7	297	297
8	330	330
9	325	325
10	276	276
11	285	285
12	273	273
Total	3222	3222

Insights of Number of Orders

4 had 3222 Orders.

Orders and total day are negatively correlated with each other.

2- Dashboard of Upper East Side orders in day 4 all months in certain hours.



Orders each day

hour	1	2	3	4	5	6	7	8	9	10
0	36	32	42	5	11	9	22	23	53	76
1	49	21	49	5	10	6	15	18	58	90
2	36	18	51	2	2	7	7	19	63	79
3	35	13	38	5	9	5	10	8	37	44
4	32	7	17	3	2	3	3	6	16	21
5	11	1	4	3	4	2	3	4	4	2
6	6	1	3	10	17	14	7	11	3	4
7	6	3	4	19	18	15	13	25	6	9
8	10	3	6	19	27	29	44	25	15	5
9	4	15	12	29	18	26	23	31	20	20
10	9	14	13	21	24	21	25	28	19	24
11	18	25	25	22	19	23	27	26	28	23
12	26	29	34	18	30	31	22	18	53	25
13	29	24	26	18	28	23	30	24	38	27
Total	554	570	560	439	493	529	584	623	826	709

Insights of number of orders

At 945, 30 had the highest Sum of Orders and was 350.00% higher than 23, which had the lowest Sum of Orders at 210.

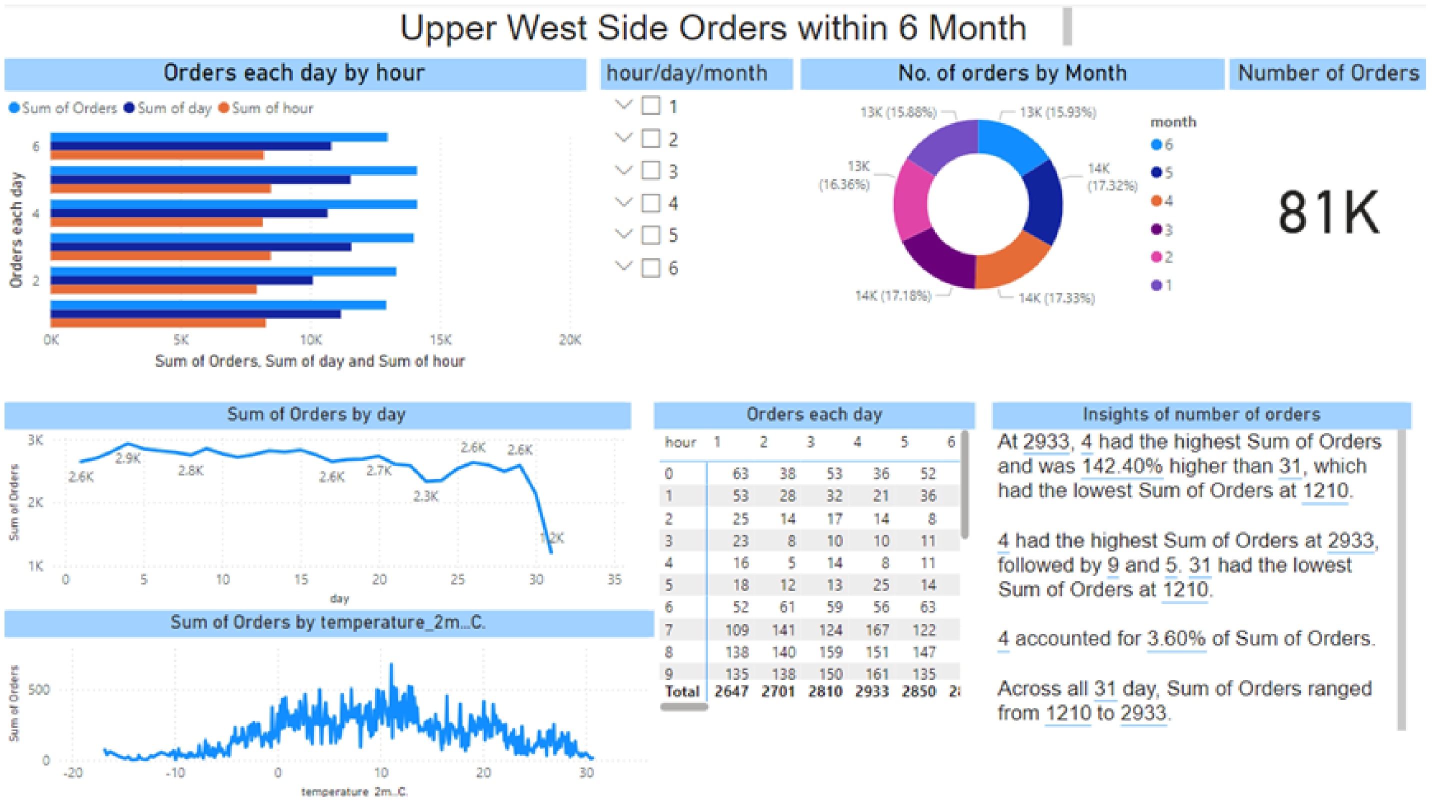
30 had the highest Sum of Orders at 945, followed by 16 and 9. 23 had the lowest Sum of Orders at 210.

30 accounted for 5.12% of Sum of Orders.

Across all 31 day, Sum of Orders ranged from 210 to 945.

Sum of Orders and total Sum of hour are negatively correlated with each other.

3- Financial District's first month over the days and hours.



Orders each day

hour	1	2	3	4	5	6
0	63	38	53	36	52	
1	53	28	32	21	36	
2	25	14	17	14	8	
3	23	8	10	10	11	
4	16	5	14	8	11	
5	18	12	13	25	14	
6	52	61	59	56	63	
7	109	141	124	167	122	
8	138	140	159	151	147	
9	135	138	150	161	135	
Total	2647	2701	2810	2933	2850	2

Insights of number of orders

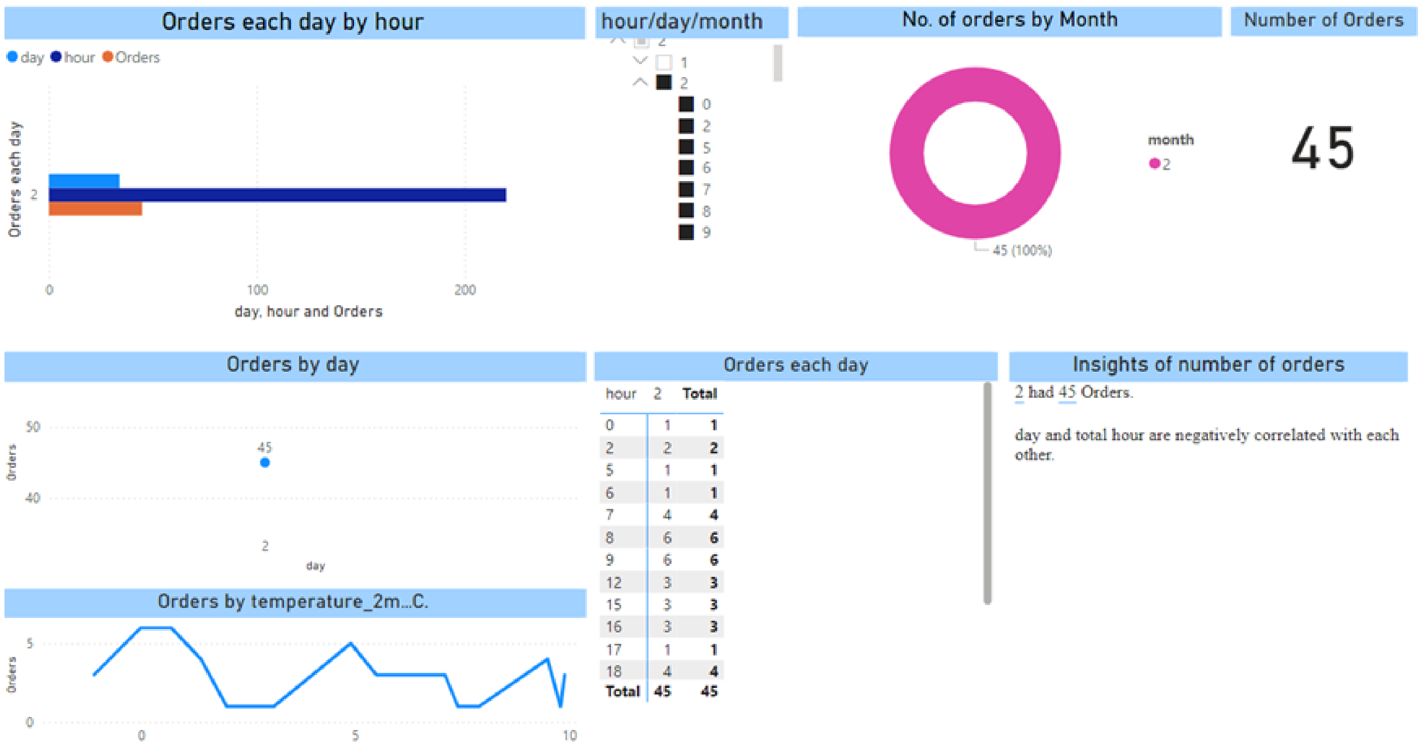
At 2933, 4 had the highest Sum of Orders and was 142.40% higher than 31, which had the lowest Sum of Orders at 1210.

4 had the highest Sum of Orders at 2933, followed by 9 and 5. 31 had the lowest Sum of Orders at 1210.

4 accounted for 3.60% of Sum of Orders.

Across all 31 day, Sum of Orders ranged from 1210 to 2933.

4- Upper West Side all months



Orders each day

hour	2	Total
0	1	1
2	2	2
5	1	1
6	1	1
7	4	4
8	6	6
9	6	6
12	3	3
15	3	3
16	3	3
17	1	1
18	4	4
Total	45	45

Insights of number of orders

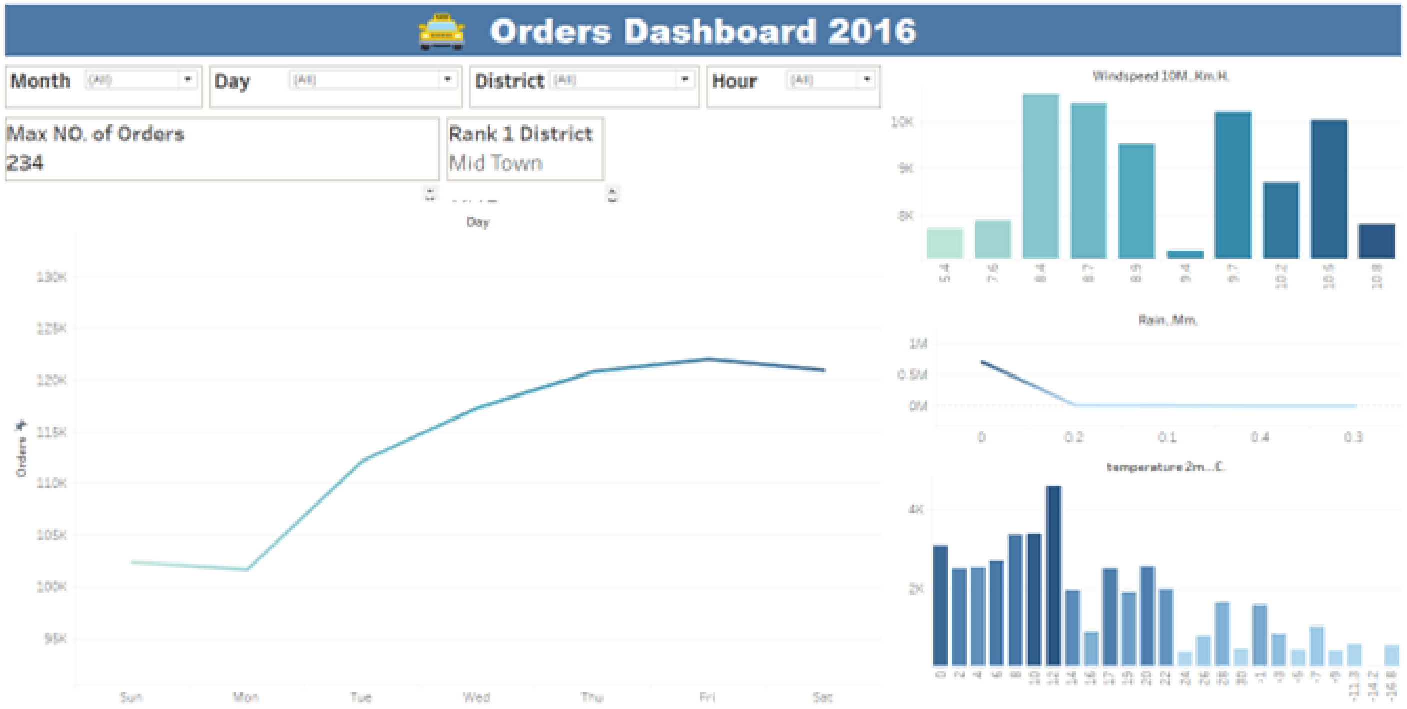
2 had 45 Orders.

day and total hour are negatively correlated with each other.

5- Finally, Harlem City dashboard, but representing 2nd month in day 2

5.2 Dashboard using Tableau

The Dashboard was designed to provide insights into the factors affecting orders in various districts over time.



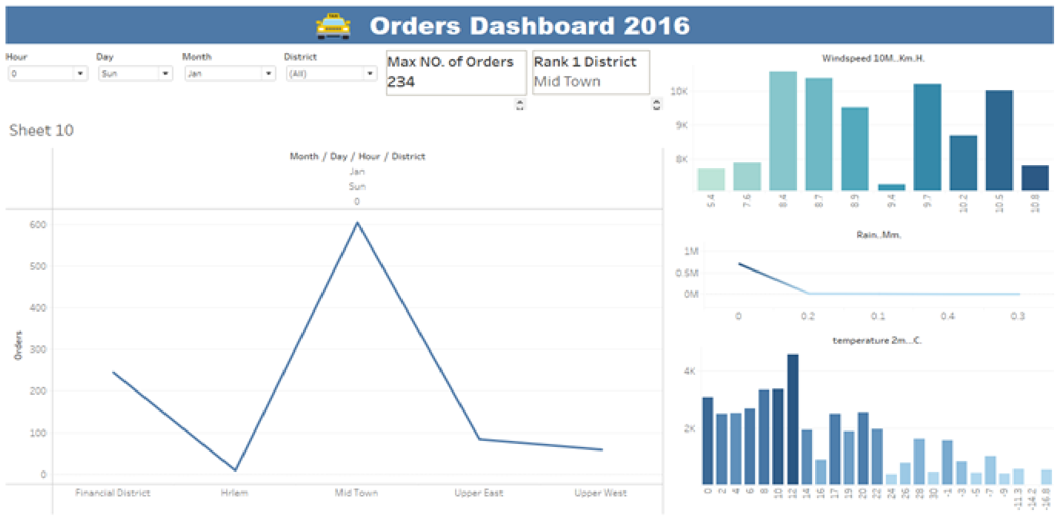
Orders Dashboard 2016

Month (All) | Day (All) | District (All) | Hour (All)

Max NO. of Orders 234

Rank 1 District Mid Town

Key Features of the Dashboard:

1. **Filters for Dynamic Analysis:**
   - The dashboard includes filters for Month, Day, District, and Hour, enabling users to customize the view and explore specific time periods, regions, or time slots.
   - Users can now know the number of orders per month, per day, per hour, and in each district.
   - 

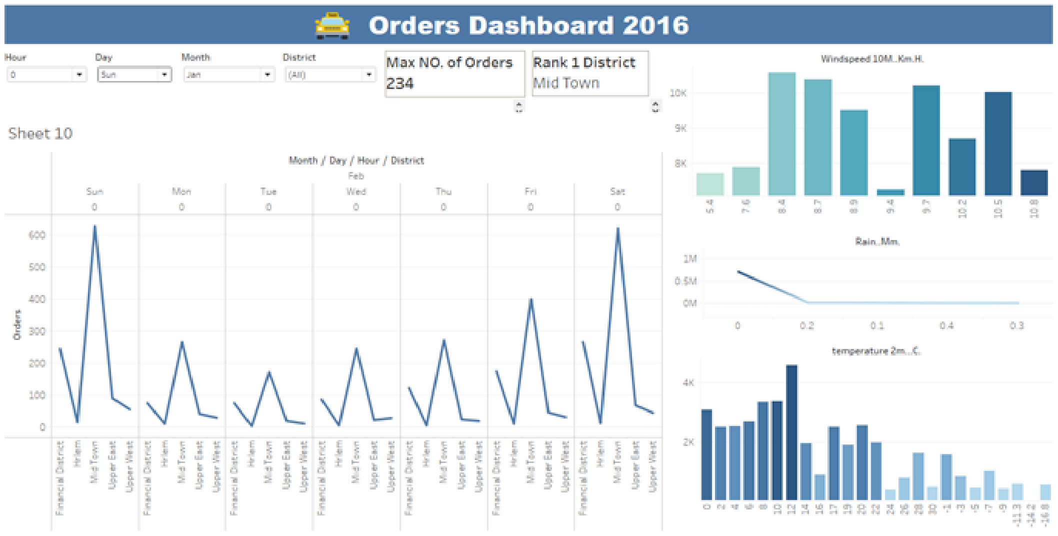
Orders Dashboard 2016

Hour 0 | Day Sun | Month Jan | District (All)

Max NO. of Orders 234

Rank 1 District Mid Town

Sheet 10



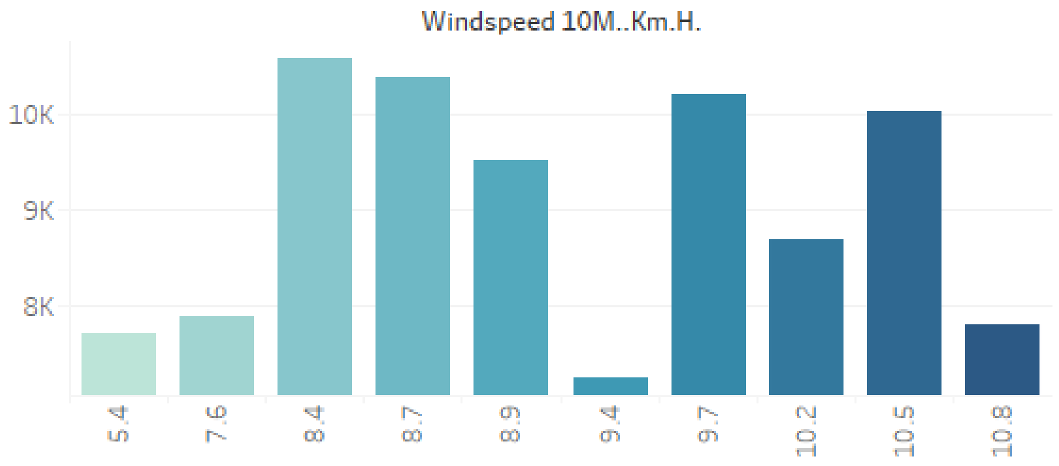
-

2. **Max Number of Orders:**
   - A summary section displays the maximum number of orders (234) and the top-performing district (Mid Town).
   - This information provides insights into the best-performing areas, which we can use to direct drivers to.
3. **Visual Representation of Trends:**
   - A line chart displays the variation in the number of orders throughout the week, offering a clear view of daily performance trends.
   - Histogram-like bar charts show correlations between environmental factors such as wind speed, rain levels, and temperature with the number of orders.

5.3 Insights and Interactive Functionality

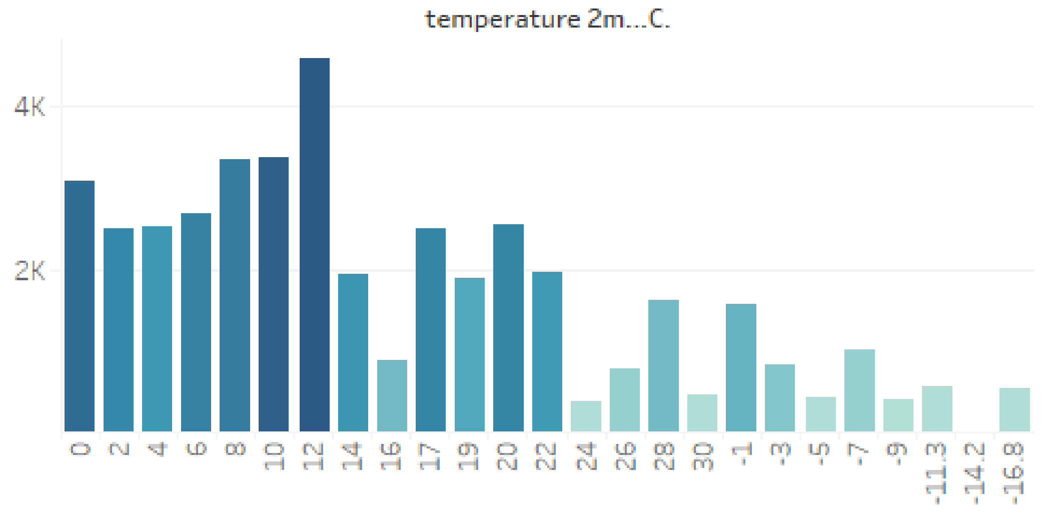
The dashboard's interactivity makes it an excellent tool for decision-making:

- **Understanding Trends by Day:** The line chart helps identify patterns, such as peak order days during the week, enabling businesses to optimize resources.
- **Weather Impact Analysis:**
  - **Wind Speed:** The bar chart demonstrates the effect of varying wind speeds on the volume of orders.

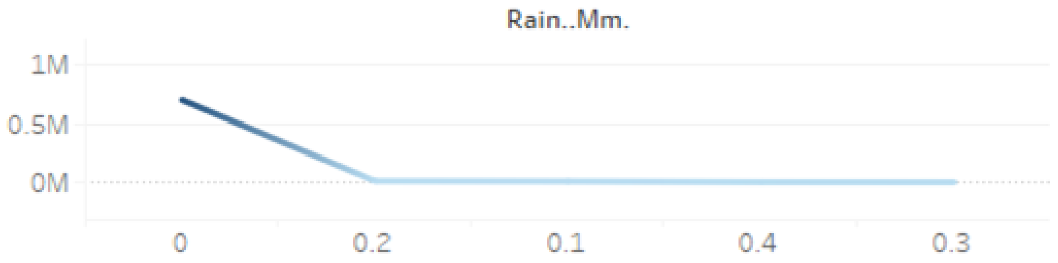


  -
  - **Rain and Temperature:** The rain and temperature charts provide insights into how these factors influence customer behavior and order trends.

- * In the temperature chart, we can see that orders tend to be higher at moderate temperatures and tend to be lower when the temperature is extremely high or low.



- *
- * In the rain chart, we can see that there are almost no orders when it's raining, regardless of the intensity of rain.



- *

- **Interactive Exploration:** The filters allow users to dynamically adjust parameters to explore specific queries, such as how orders change during hours in a district or how weather conditions affect order volumes.

6 Machine Learning: Random Forest Model

The goal is to predict demand for car ride orders using historical data, employing a Random Forest model.

6.1 Methodology

1. **Data Preprocessing:**
   - **Feature Selection:** Variables were selected to capture the most influential factors affecting car ride demand. These include weather-related variables and time-of-day or day-of-week features.
   - **Time-based Variables:** Variables representing hourly data (`hr_0` to `hr_23`) and days of the week (Friday, Saturday, Sunday, Monday, Tuesday, Wednesday, Thursday) were included to capture patterns related to time.
   - **Weather Features:** Environmental variables like `temperature_2m...C.`, `precipitation..mm.`, and `rain..mm.` were used to account for external factors that may influence ride demand.
2. **Train-Test Split:**
   - The dataset `df` was split into training and testing sets, with 75% of the data used for training and 25% for testing.
   - Given that the total number of rows in `df` is 4346, 75% of the dataset is approximately 3268.5 rows, which is rounded to 3269 rows for the training set.

3. **Model Development:**
    - A Random Forest model was implemented using the formula `Orders ~.` with all selected features. The model predicts the number of orders (target variable) based on the weather and time-based features.
4. **Seed Optimization:**
    - The model was run using 200 different seeds to account for variability in the random sampling process inherent in Random Forest.
    - Seed 78 produced the best performance, as determined by evaluating the model using metrics such as RMSE and R^2. This seed was used for all further analyses.
5. **Training Configuration:**
    - The Random Forest model was trained using the provided training data set, consisting of weather and temporal variables.
    - The model leverages the ensemble learning method, combining multiple decision trees to provide an average prediction, which helps to reduce overfitting and improve prediction accuracy.

6.2 Evaluation Metrics

- **R^2 Score:** This measures the proportion of variance in the actual ride demand explained by the model. A higher R^2 indicates that the model is a better fit for the data.
- $R^2 = 0.8772$ (for the test data)

6.3 Visualization

Actual vs Predicted Demand: The plot shown in Figure 3 compares the actual demand with predictions from each model for the first 200 time steps.

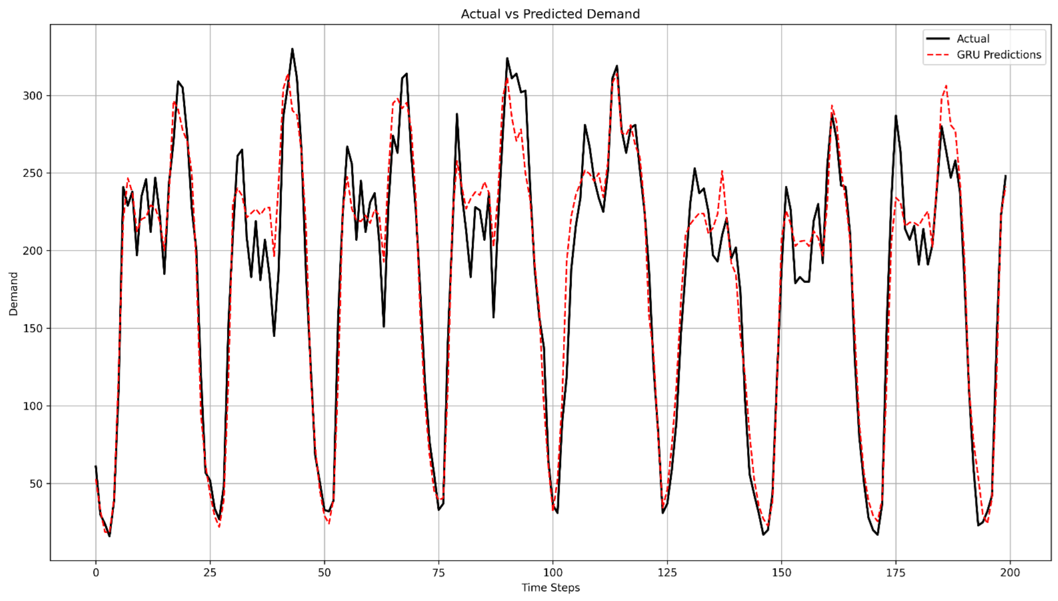


Figure 3: Actual vs Predicted Demand

7 Machine Learning: Neural Networks Models

The aim of building these models is to predict demand for car ride orders using historical data, choosing the best model after comparing the performance of RNN, LSTM, and GRU models.

7.1 Methodology

1. **Data Preprocessing:**
   - Feature selection was applied to include relevant variables.
   - Data normalization using `StandardScaler` ensured all features were scaled appropriately.
   - Sequences were generated using a sliding window approach with a step size of 4 hours to predict the demand of the next hour.
2. **Model Development:**
   - Three models were implemented: RNN, LSTM, and GRU.
   - Early stopping was applied during training to prevent overfitting.
   - Models were trained for up to 50 epochs with a batch size of 32.

7.2 Evaluation Metrics

1. **Mean Absolute Error (MAE):** Measures the average absolute difference between predicted and actual values.
2. **Root Mean Squared Error (RMSE):** Measures the square root of the average squared differences.
3. **R^2 Score:** Evaluates the proportion of variance explained by the model.

7.3 Results

Model	Loss (MSE)	MAE	RMSE	R^2
RNN	0.0552	0.1735	0.2350	0.9319
LSTM	0.0559	0.1713	0.2364	0.9311
GRU	0.0519	0.1673	0.2278	0.9360

Overall Results:

- The GRU model achieved the best results with the lowest loss, MAE, RMSE, and the highest R^2 score, indicating superior predictive power.
- LSTM performed closely behind GRU and can be considered a viable alternative.
- RNN lagged due to its inability to capture long-term dependencies effectively.

Visualization

Actual vs Predicted Demand: The plot below compares the actual demand with predictions from each model for the first 100 time steps.

The plot shown in figure 4 compares the actual demand with predictions from each model for the first 100 time steps.

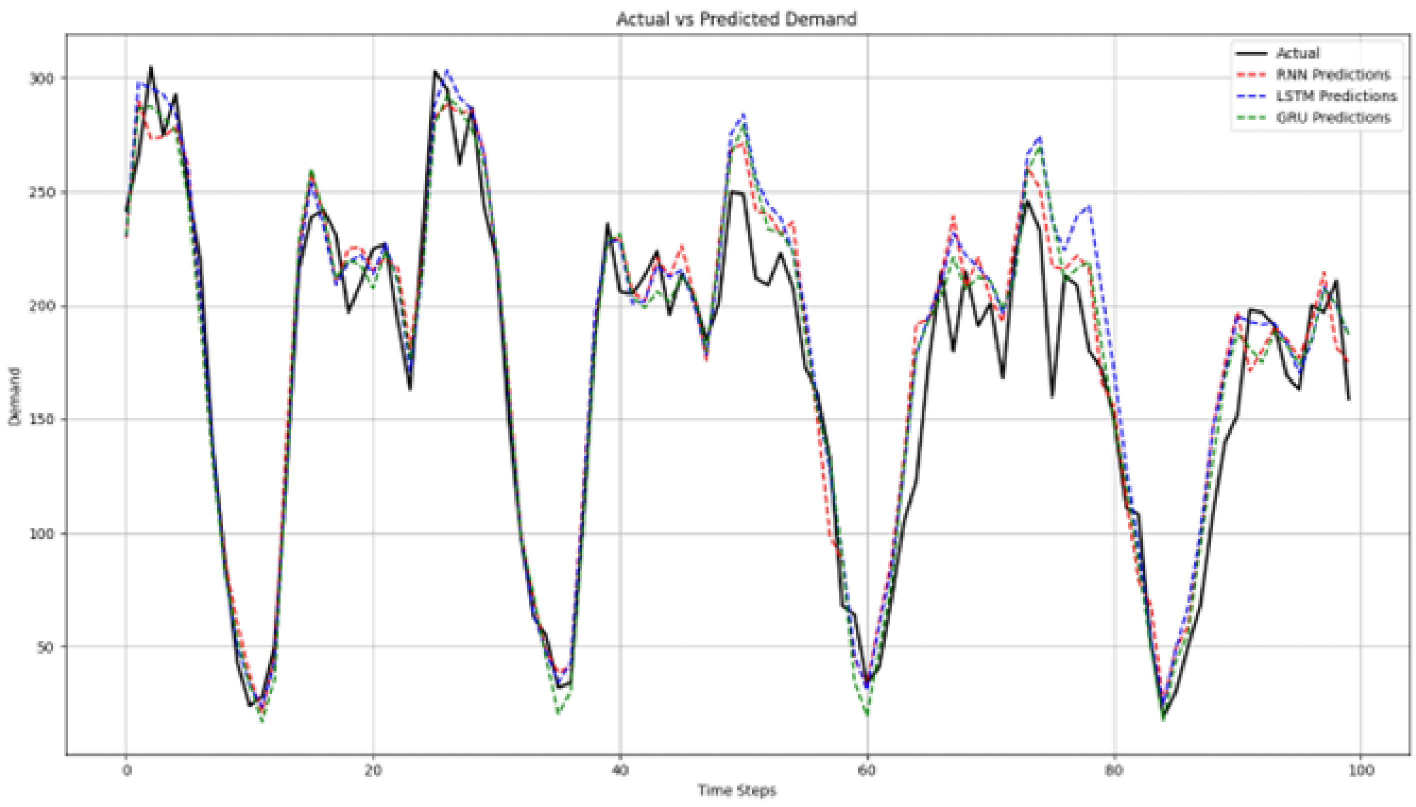


Figure 4: Actual vs Predicted Demand

7.4 Validation MAE Trends Across Epochs

The plot shown in figure 5 illustrates the validation MAE trends for each model during training.

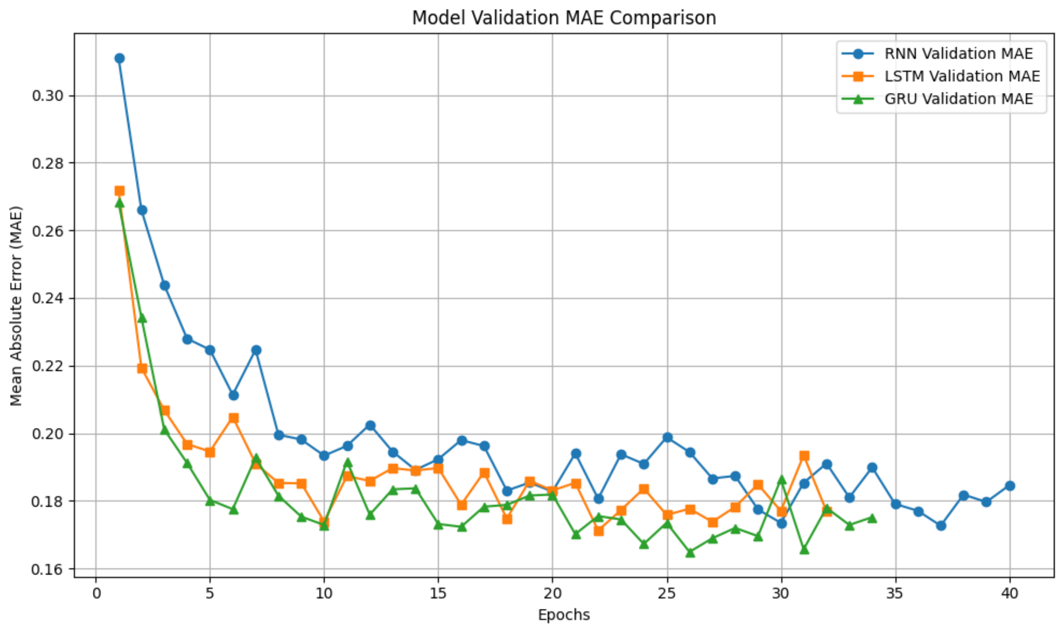


Figure 5: Validation MAE Trends

8 Prediction Models for All Regions

Prediction models were developed to forecast car ride demand across five distinct regions of Manhattan (as discussed in Section 4). Each region was independently modeled using its specific dataset, enabling the creation of models tailored to the unique characteristics and trends within each area. Among the various approaches tested, GRU was chosen as the primary model due to its superior predictive performance, particularly in terms of R² score.

The data preprocessing pipeline incorporated the same relevant features used in the initial GRU model, including weather conditions, temporal patterns, and demand history. GRUs were selected for their capability to effectively capture sequential dependencies in time-series data while being computationally efficient. The training process involved splitting each region's data into training and testing sets, ensuring robust evaluation and minimizing overfitting. Each model's performance was assessed using key metrics such as RMSE and R² score, providing a consistent framework for comparison across regions.

By developing dedicated models for each region—Financial District, Midtown, Upper East Side, Upper West Side, and Harlem—the analysis captured localized variations and trends in ride demand.

8.1 Financial District Prediction Model

The Financial District Prediction Model demonstrates a strong predictive capability as shown in Figure 6, as evidenced by an ***RMSE of 0.3491*** and an ***R² of 0.8481***. The graph illustrates the model's ability to closely follow the actual demand trends over time, with the GRU predictions aligning well with the observed data. This high R² value indicates that the model explains a significant portion of the variance in demand, making it a reliable tool for forecasting ride-hailing needs in the Financial District. The relatively low RMSE further supports the model's accuracy, suggesting minimal deviation between predicted and actual values. Overall, the model effectively captures the temporal patterns of demand, providing valuable insights for strategic planning and resource allocation.

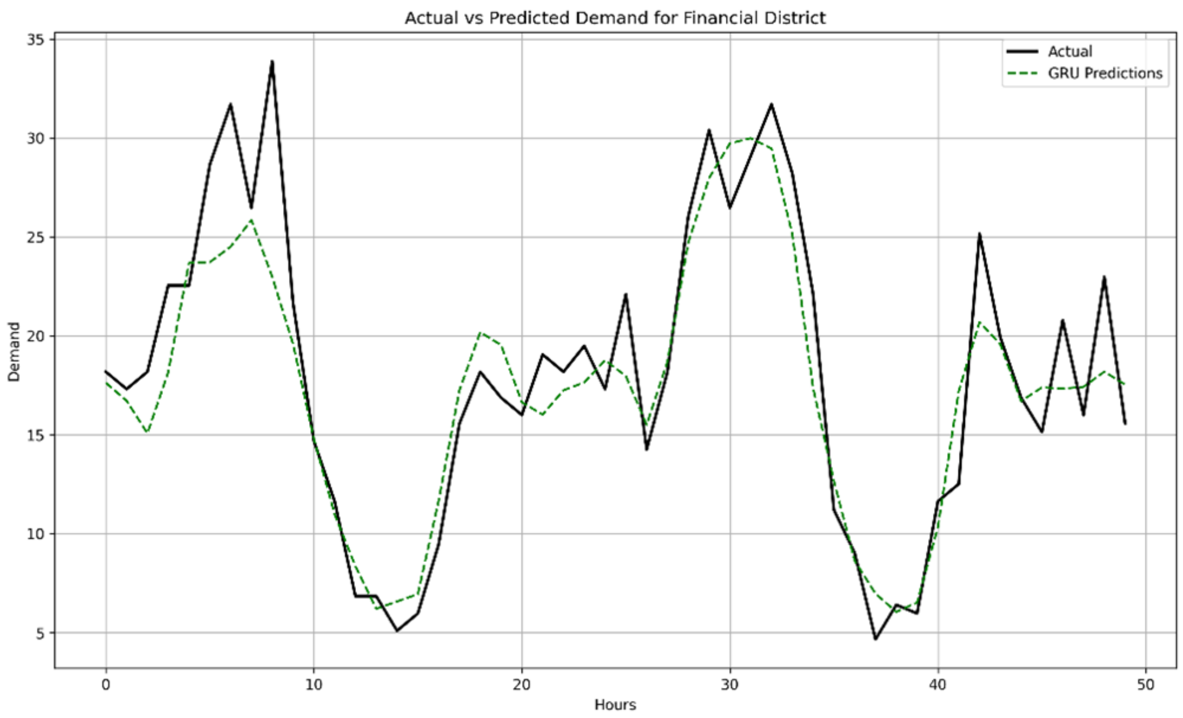


Figure 6: Financial District Prediction Model Performance

8.2 Midtown Prediction Model

The Midtown Prediction Model exhibits excellent predictive performance as shown in Figure 7, with an ***RMSE of 0.2748*** and an ***R² of 0.9059***. The graph shows that the GRU model's predictions closely match the actual demand patterns, indicating a high level of accuracy. The R² value suggests that the model accounts for over 90% of the variance in demand, making it a robust tool for forecasting in the Midtown area.

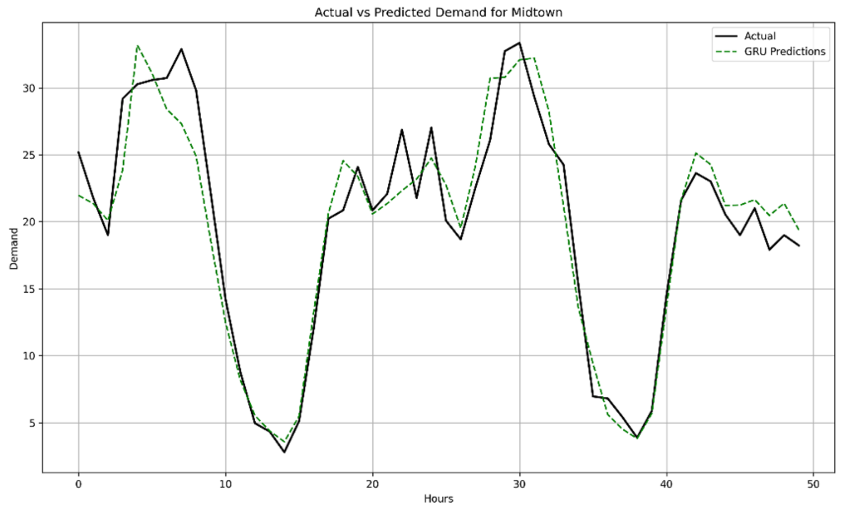


Figure 7: Midtown Prediction Model Performance

8.3 Upper East Side Prediction Model

The Upper East Side Prediction Model effectively forecasts demand, achieving an ***RMSE of 0.3234*** and an ***R² of 0.8863***. The model's predictions closely mirror the actual demand trends as illustrated in Figure 8, demonstrating its reliability. With a high R², the model captures most of the demand variability, making it a valuable asset for planning and decision-making in the Upper East Side. The low RMSE indicates precise predictions, ensuring accurate insights for managing resources and operations in this region.

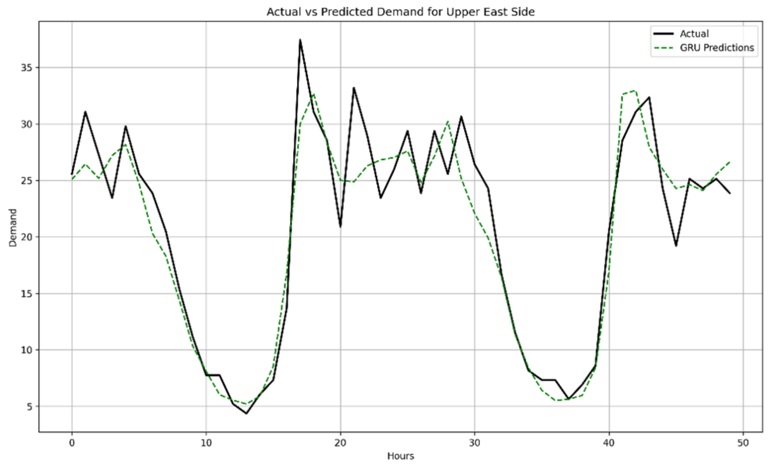


Figure 8: Upper East Side Prediction Model Performance

8.4 Upper West Side Prediction Model

The Upper West Side Prediction Model shows moderate predictive performance as presented in Figure 9, with an ***RMSE of 0.4457*** and an ***R^2 of 0.7678.*** While the model captures the general demand trends, there is room for improvement in precision. The R^2 value indicates that the model accounts for a substantial portion of demand variability, but some fluctuations remain unexplained.

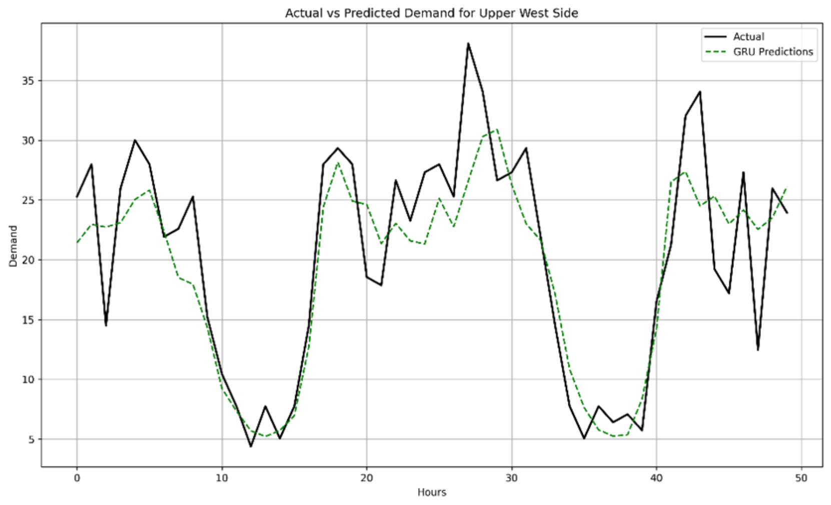


Figure 9: Upper West Side Prediction Model Performance

8.5 Harlem Prediction Model

The Harlem Prediction Model shows limited predictive accuracy as shown in Figure 10, with an ***RMSE of 0.8443*** and an ***R^2 of 0.2152***. This indicates that the model struggles to capture the variability in demand, explaining only a small portion of it. Several factors could contribute to these results. Harlem's dynamic and diverse community, with its mix of residential, cultural, and commercial influences, may lead to complex and fluctuating demand patterns.

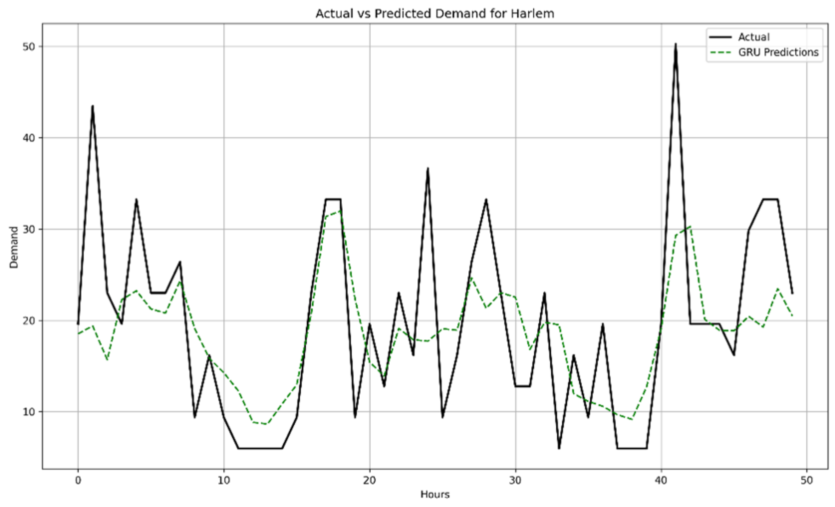


Figure 10: Harlem Prediction Model Performance

9 Geospatial Grid Analysis and Point-Based Representation

After predicting regions in Manhattan, we extended our analysis by dividing the map into a uniform grid for more granular geospatial insights. Each grid square represents an area with a side length of approximately 850 meters. To achieve this, Python was used to calculate the actual distance between coordinates based on the Haversine formula. This ensured accuracy in measuring distances and establishing grid dimensions.

9.1 Grid Creation and Coverage

The grid covered the Manhattan region and extended slightly beyond it, with grid cells defined using latitude and longitude boundaries. However, some data points from the original dataset fell outside this grid and were removed to maintain consistency and relevance. This refinement ensured the grid accurately represented Manhattan's geospatial structure.

Assigning and Filtering Data Based on Geospatial Cells

To integrate ride data into this grid, a Python-based methodology was employed:

1. **Cell Assignment:**
    - Pickup latitude and longitude values were used to assign a grid cell for each data point.
    - The grid cells were identified by rounding coordinates to predefined resolution levels (back in Section 2) and mapping them to unique identifiers using a dictionary. This dictionary facilitated the efficient assignment of geospatial cells to ride data points.
2. **Data Filtering:**
    - Data points not associated with valid grid cells were marked as invalid and excluded.
    - Columns unrelated to geospatial analysis were dropped to streamline the dataset.
3. **Indexing and Sorting:**
    - The filtered data was sorted by assigned grid cells and indexed sequentially to prepare for further analysis.

Unique Cell Identification and Point Representation

The next step involved identifying unique cells within the grid and ensuring each cell was represented by a single realistic point. This was done by:

1. Extracting unique cell identifiers from the dataset.
2. Removing cells with no associated data points, such as those located in water areas outside Manhattan.
3. Assigning realistic points to grid cells by finding the closest data point to the center of each cell. This ensured that the representative point is on a street and reflected actual ride activity.

The result was a cleaned, geospatially-aware dataset with each grid cell represented by a single point, providing a detailed and actionable framework for further analyses and visualization. This granularity facilitates targeted insights into passenger pickup patterns and helps bridge the gap between spatial precision and data usability.

9.2 Aggregation and District Assignment of Data for Each Cell

For each unique cell, data aggregation was conducted similar to the process described in Section 3 but tailored to the spatial granularity of individual cells. The cleaned and structured data for each cell was stored in a dedicated dictionary `Each_Cell_data_dict`, enabling efficient access and further analysis.

Once the data for each cell was prepared, we assigned each cell to its corresponding district using the same method described in Section 4.2. The spatial boundaries defined earlier ensured that each cell was accurately linked to a specific district. This step allowed for district-level analysis of aggregated grid data. By combining the grid-based approach with district assignments, the resulting dataset offered a robust framework for analyzing ride activity with both spatial precision and district-level context as represented in Figure11.

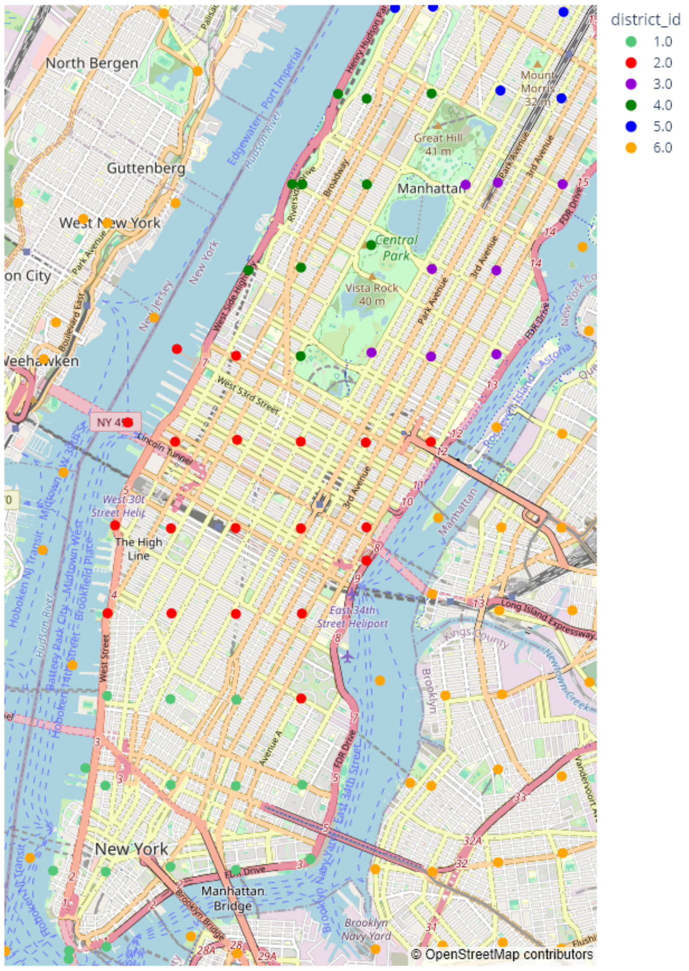


Figure 11: Data Aggregation Map

10 Exploring Alternative Approaches: Probability Distributions for Demand Allocation

When we applied the GRU machine learning model, which had shown promise in previous analyses, the results for this grid-based approach were unsatisfactory. The model struggled to capture the complex relationships between variables at the cell level, yielding unreliable and inconsistent predictions. These shortcomings highlighted the limitations of traditional machine learning models in scenarios requiring fine-grained spatial predictions over multiple cells.

10.1 A Shift to Probability Distributions for Demand Allocation

Given the challenges with GRU, we adopted a fundamentally different approach: creating probability distributions for each region, mapped to individual cells for every hour of the day. This method allowed us to estimate the likelihood of demand occurring in specific cells within a district, bypassing the direct reliance on predictive models at the cell level.

The new framework leveraged district-level predictions obtained from our earlier models and distributed these predictions among the cells within each district. For each cell, we calculated its share of the hourly district-wide demand based on historical data, using ratios derived from observed demand patterns.

To enable real-time insights and dynamic visualization in our map application, we interpolated the hourly predictions to estimate demand at the minute level. This interpolation smoothed the transition of demand over time, ensuring a continuous and realistic representation of activity.

10.2 Simulation and Distribution

Using the calculated probability distributions, we simulated the allocation of demand to specific cells. A weighted random selection process determined the number of requests allocated to each cell, guided by the probability values. This simulation effectively replicated real-world demand distributions while maintaining alignment with district-level predictions.

The final output provided a granular, probabilistic representation of demand across the map, enabling enhanced visualization and strategic insights for urban planning and resource allocation. This shift in approach proved to be both adaptable and practical, offering a viable solution to the limitations faced in our earlier modeling attempts.

11 Database Design and Implementation for the Application

11.1 System Architecture and Purpose

The database system designed for the application represents an optimized architecture that integrates weather patterns, geospatial distributions, and probabilistic allocation to manage and predict taxi requests efficiently. It is structured to facilitate real-time demand distribution while maintaining data integrity and scalability. The core objective of this system is to provide a robust backend for managing geographically distributed demand, ensuring precision and adaptability in data operations.

The system's architecture is centered around six fundamental entities: `Weather`, `District`, `Request`, `Cell`, `Distribution`, and `Allocation`. These entities are interconnected through a carefully normalized structure, eliminating redundancy and ensuring seamless data flow. At its core, the database facilitates probabilistic allocation of requests across geospatial cells, enriched by weather data, and dynamically adjusts predictions based on real-time inputs.

11.2 Entities and Database Schema

The system is built around six fundamental entities as shown on Figure 12, each serving a distinct purpose while maintaining interconnectivity. The `Weather` entity functions as a comprehensive meteorological data repository, tracking environmental metrics (Temperature, Precipitation, Rain) with a one-to-many relationship to `Request`. The `Cell` entity manages geographical coordinates through Longitude and Latitude, establishing a crucial foundation for location-based services. The `Distribution` entity handles probability calculations and hourly data, maintaining a one-to-many relationship with `Cell` to enable precise geographical distribution mapping. The `Request` entity serves as the transactional cornerstone, recording request counts and predicted requests while maintaining relationships with `Weather` and `District` entities. The `District` entity manages regional segmentation, while the `Allocation` entity acts as a sophisticated junction point, connecting `Distribution` and `Request` entities while tracking minute-level request allocation counts.

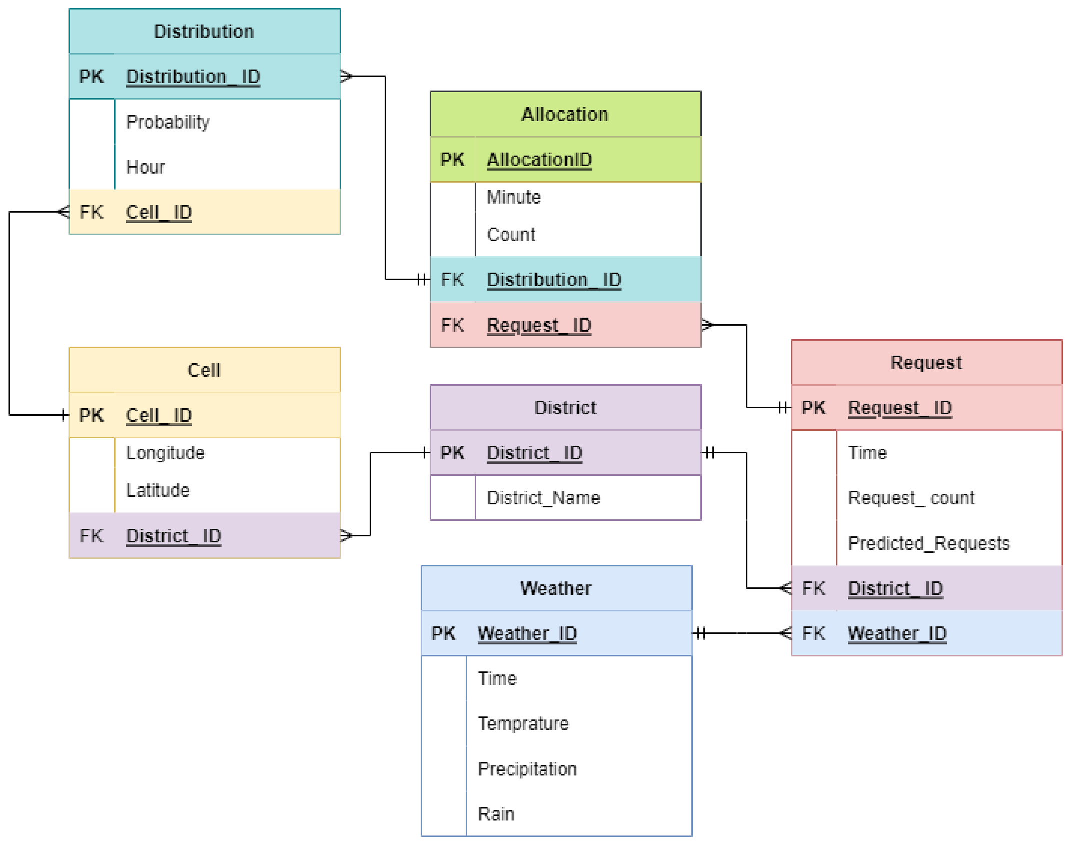


Figure 12: Database Schema

11.3 Design Benefits and Flexibility

The normalized structure of this database offers several significant advantages. By clearly defining relationships and segregating data into distinct entities, redundancy is minimized, ensuring that the system remains lean and efficient. The ID-based relationship system ensures robust connectivity while supporting geographical precision through the integration of the Cells table.

The design's flexibility allows for seamless expansion. For instance, new attributes can be added to track additional metrics such as wind speed or traffic density without disrupting existing relationships. The architecture's modular nature ensures that each entity operates independently, allowing for updates, insertions, or deletions without compromising data integrity.

11.4 Operational Efficiency and Scalability

Operational efficiency is a core strength of the database design. The interconnected tables enable efficient querying and data manipulation, even as the volume of data grows. For instance, probability calculations for geospatial cells are seamlessly linked to hourly demand predictions, which are then interpolated into minute-level allocations for real-time application updates. The structured relationships also facilitate rapid analysis, making the system highly effective for handling large datasets and evolving business requirements.

11.5 Future-Proofing and Maintenance

The database design incorporates a forward-thinking approach that anticipates future needs. Its flexibility allows for the addition of new entities or attributes with minimal restructuring, ensuring that the system evolves seamlessly as requirements change. For instance, incorporating real-time traffic data or additional weather metrics can be achieved without disrupting existing data flows.

By combining geographical precision, operational flexibility, and scalability, this database architecture provides a robust foundation for managing and analyzing taxi demand patterns. Its integration of weather-based predictions and cell-based demand allocation ensures a sophisticated framework capable of supporting advanced visualization and strategic planning tools.

12 Dynamic Prediction and Allocation in Action

The final application represents the culmination of advanced database architecture, machine learning predictions, and geospatial visualizations, delivering a comprehensive solution for real-time taxi demand management. Its core objective is to empower taxi drivers and fleet managers with actionable insights, ensuring optimal resource allocation and enhanced user experience.

12.1 Real-Time Demand Predictions and Distribution

At the heart of the application is its ability to dynamically predict taxi demand at a district level using machine learning models. These models process historical and real-time data, such as weather conditions and previous demand patterns, to forecast upcoming demand. The application then distributes this district-level prediction into minute-level estimates and allocates it across smaller geographical cells, ensuring a high resolution of demand mapping.

The probabilistic allocation system leverages historical probability distributions stored in the database to determine where demand is likely to occur within each district. Using weighted random sampling, the system simulates real-world taxi requests, ensuring realistic and precise allocation results. These demand allocations are stored in the database and updated every minute, enabling a truly dynamic workflow.

12.2 Interactive Visualization for Taxi Drivers

A critical feature of the application is its real-time demand map, created using the Folium library in Python. The map provides an intuitive, interactive visualization of demand hotspots across Manhattan, allowing taxi drivers to adapt their routes efficiently. The map, as shown in the figure 13, displays clusters representing aggregated demand levels, with larger or higher-density clusters indicating areas of higher predicted demand. Each marker corresponds to a geographical cell, providing precise locational data for drivers. The clusters are updated every minute, ensuring the information remains current and actionable.By providing this visual guidance, the application transforms static predictions into tangible, driver-centric insights. Taxi drivers can prioritize high-demand areas, minimizing idle time and improving their earnings, while customers benefit from faster response times.

12.3 Enhanced Operational Efficiency

This seamless integration of prediction, allocation, and visualization significantly enhances operational efficiency. Taxi fleets can maximize their coverage in real-time, adapting to changing weather conditions or fluctuating demand patterns without manual intervention. The map also serves as a powerful decision-making tool for dispatchers, allowing them to monitor demand across the city and make data-driven adjustments as needed. The application's modular architecture ensures it is ready to scale alongside growing data volumes and evolving requirements. The system can easily integrate additional features, such as demand trend overlays, traffic conditions, or customized alerts for drivers. The database's robust design supports these expansions without disrupting its existing functionality. In conclusion, the application exemplifies a sophisticated, real-time demand management system that blends predictive analytics with intuitive geospatial tools. By delivering minute-level updates and actionable visualizations, it sets a new standard for efficiency and responsiveness in urban mobility systems. Taxi drivers, dispatchers, and passengers alike benefit from its streamlined functionality, making it an indispensable tool for modern transportation networks.

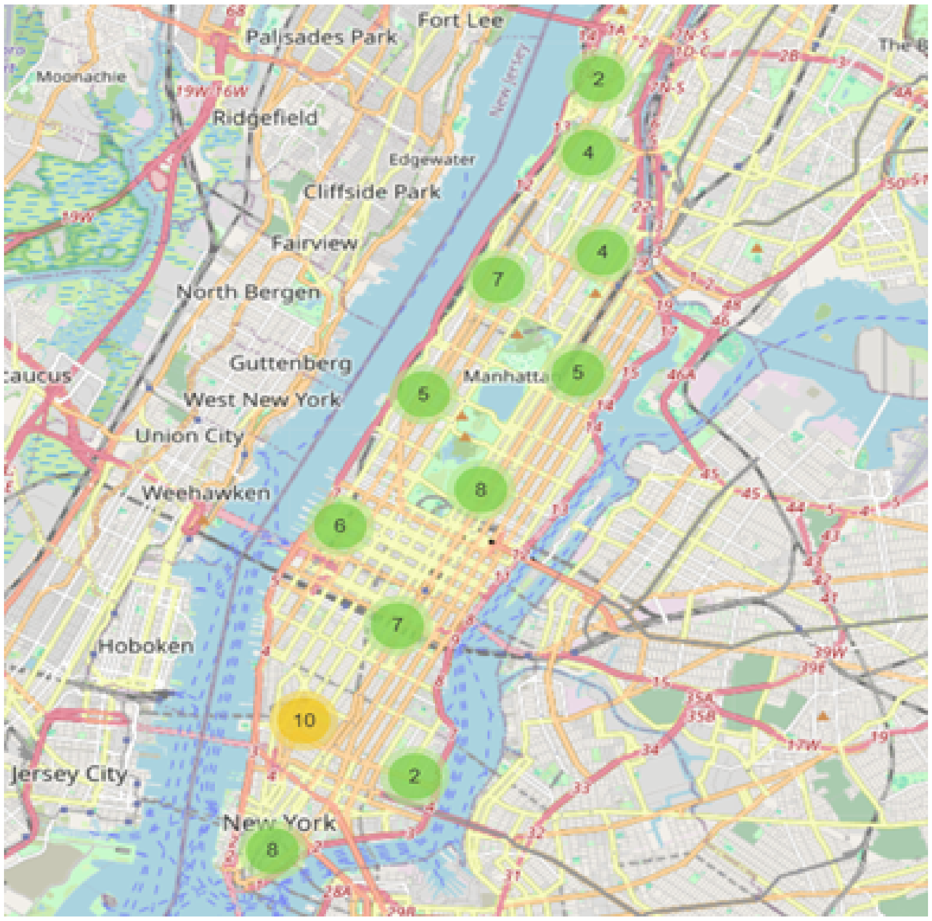


Figure 13: Dynamic Map

13 Conclusion

The analysis revealed distinct spatial and temporal patterns in Manhattan's taxi demand, highlighting significant spatial and temporal trends. Midtown consistently emerged as the highest-demand area. Moderate weather was related with peak ride quantities, but severe temperatures and rainfall considerably reduced demand. Temporally, bike activity peaked during weekday mornings and evenings, reflecting commuter trends, while weekends were more evenly distributed. These findings highlight the role of environmental conditions and human behavior in determining urban mobility. Machine learning algorithms, particularly GRU, displayed high predicted accuracy, with Midtown and the Upper East Side doing best. GRU models accurately captured temporal and spatial demand dynamics, with good R^2 scores and low RMSE values. However, demand predictability differed by region, with Harlem having the lowest predictability due to uncertain demand patterns. To overcome these issues, the study used geospatial probability distributions, which allowed for exact demand mapping at both the district and grid levels, identifying localized hotspots that corresponded to urban activity centers. The integration of geospatial research, dynamic visualization, and predictive modeling yields useful insights for optimizing taxi allocation and urban transportation planning. Real-time demand maps and dashboards enabled more efficient resource allocation, which reduced driver waiting time and improved passenger service. These findings not only improve operational efficiency, but also lay foundations for long-term and adaptive transportation solutions in cities. This framework establishes a benchmark for using data-driven approaches to address mobility issues and improve urban planning. Future work could explore integrating additional data sources such as traffic patterns and socioeconomic factors to further enhance predictions and planning.