1. Optimizing Logistics Center Location

Background & Objective

In logistics, the location of a distribution center can significantly impact both cost efficiency and delivery performance. If the current location doesn't align well with vehicle movement patterns, unnecessary transportation costs pile up and delivery times stretch out.

In this project, I used GPS tracking data and road network information to recommend the **optimal location for a logistics center**. The two key goals were:

- **Cut logistics costs** by optimizing delivery routes
- **Boost efficiency** by reducing the distance between the center and delivery vehicles

Data Overview

- The dataset was provided in CSV format, with key variables summarized below:

Column	Description
`vehicle_id`	Unique identifier for each delivery vehicle
`timestamp`	Timestamp of GPS record
`move_x`	X coordinate of vehicle movement (longitude)
`move_y`	Y coordinate of vehicle movement (latitude)
`speed_km_s`	Vehicle speed (km/s)
`center_x`	Current logistics center's X coordinate
`center_y`	Current logistics center's Y coordinate

- 1. GPS Tracking Data
    - Collected at regular intervals, allowing for analysis by time of day and day of week
    - Timestamped data enabled time-specific analysis of vehicle movement patterns.

- 2. Logistics Center Coordinates
    - The current logistics center's coordinates were used to evaluate its efficiency.

- 3. External Data
    - External road network data from OpenStreetMap (OSM) was utilized to analyze the surrounding infrastructure of each cluster center. This enabled an objective evaluation of accessibility, which supported the selection of the most optimal location for the logistics center.

Data Preprocessing

1. Handling Missing Values

- Columns containing missing values in the GPS dataset were identified and handled accordingly—either by removing the columns or applying appropriate imputation techniques. For instance, columns labeled as 'Unnamed' were determined to be irrelevant and were removed from the dataset.

2. Outlier Removal

- Outliers were identified and removed from the coordinate and speed columns. The Interquartile Range (IQR) method was applied to detect and filter out extreme values based on distribution spread.

```
# Remove 'Unnamed' columns (irrelevant columns)
df_cleaned = df.dropna(axis=1, how='all')

# Calculate IQR for outlier removal
Q1_x = df_cleaned['move_x'].quantile(0.25)
Q3_x = df_cleaned['move_x'].quantile(0.75)
IQR_x = Q3_x - Q1_x
lower_bound_x = Q1_x - 1.5 * IQR_x
upper_bound_x = Q3_x + 1.5 * IQR_x

Q1_y = df_cleaned['move_y'].quantile(0.25)
Q3_y = df_cleaned['move_y'].quantile(0.75)
IQR_y = Q3_y - Q1_y
lower_bound_y = Q1_y - 1.5 * IQR_y
upper_bound_y = Q3_y + 1.5 * IQR_y

# Remove outliers
df_filtered = df_cleaned[
    (df_cleaned['move_x'] >= lower_bound_x) & (df_cleaned['move_x'] <= upper_bound_x) &
    (df_cleaned['move_y'] >= lower_bound_y) & (df_cleaned['move_y'] <= upper_bound_y)
]
```

3. Key Insights Extracted from GPS Data

- **Route Pattern Analysis**: Vehicle movement data was used to trace delivery routes and identify frequently traveled paths. This helped uncover recurring patterns in vehicle behavior.
- **Speed Analysis**: Speed data was analyzed to evaluate transportation efficiency. Time periods with sudden drops in vehicle speed were identified as potential congestion points.
- **Time-of-Day Segmentation**: The dataset was segmented by time slots to pinpoint hours during which vehicle activity was heavily concentrated. This provided insight into peak delivery periods.

Analytical Methodology

1. Route Mapping & Cost Estimation

- GPS data was used to visualize delivery routes and calculate travel distances. These distance metrics served as the foundation for estimating logistics-related transportation costs.

2. Distance-Based Clustering: Identifying Central Points

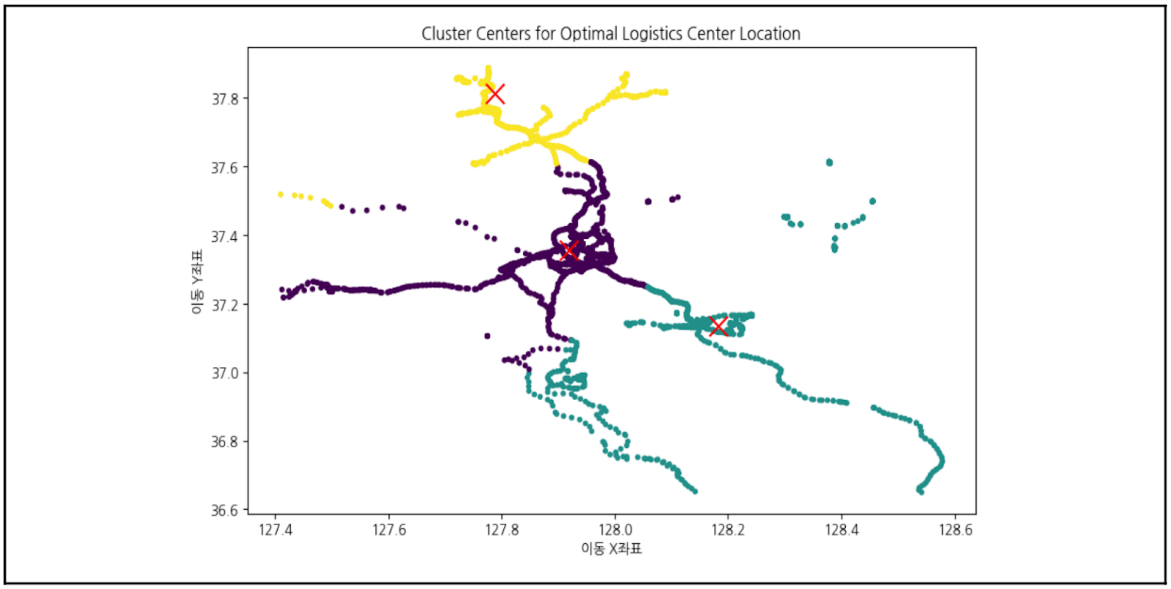
- K-Means Clustering: The K-Means algorithm was applied to group frequently visited locations based on GPS coordinates. Each cluster's centroid was evaluated as a potential logistics hub.
- Elbow Method: To determine the optimal number of clusters, the Elbow Method was used. This helped identify a balanced cluster count that minimized within-cluster variance.

3. Incorporating External Data: Traffic and Accessibility Considerations

- Road Network Analysis via OpenStreetMap: Road infrastructure surrounding each cluster center was analyzed using OpenStreetMap (OSM) data. This enabled an evaluation of accessibility, which informed the selection of the most strategic logistics center location.

4. Optimization Using Clustering Algorithms

- K-Means for Candidate Selection: Vehicle movement data was clustered using the K-Means algorithm, and the centroid of each cluster was treated as a potential candidate location for the logistics center.
- Accessibility Evaluation: The surrounding road network of each candidate site was further assessed to measure accessibility. The final location was chosen based on the highest overall access efficiency.



Final Site Selection Process

- **Road Accessibility Assessment:** Using OpenStreetMap data, road accessibility for each candidate location was analyzed. The cluster center with the shortest average travel distance was selected as the optimal logistics center site.

```
# Accessibility evaluation: Calculate the average distance from each cluster center to major road nodes
accessibility_scores = []
for i, G in enumerate(road_networks):
    try:
        center_node = ox.distance.nearest_nodes(G, cluster_centers[i][0], cluster_centers[i][1])

        # Use networkx's shortest_path_length
        lengths = nx.single_source_dijkstra_path_length(G, center_node, weight='length')

        # Use average distance as an accessibility score
        mean_distance = np.mean(list(lengths.values()))
        accessibility_scores.append(mean_distance)

    except Exception as e:
        print(f"Accessibility evaluation error: {e}. Skipping evaluation for cluster {i+1}.")
        accessibility_scores.append(float('inf'))  # Treat as infinitely inaccessible
```

Conclusion

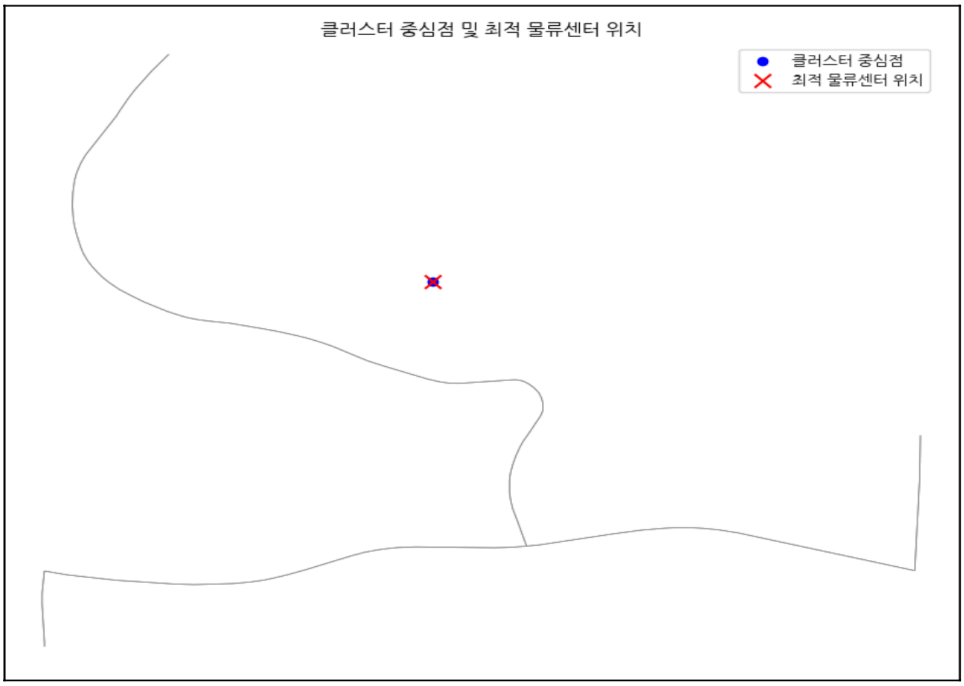
1. Optimal Location Result

- Selected Site: The optimal logistics center location was determined to be the center of Cluster 2, which showed the highest accessibility among all candidates.
- Coordinates:
    - Longitude (X): 128.18263091269648
    - Latitude (Y): 37.134549694830596

```
# Select the cluster center with the highest accessibility as the optimal location
if accessibility_scores:
    best_cluster_idx = np.argmin(accessibility_scores)
    best_location = cluster_centers[best_cluster_idx]

    print(f"The optimal logistics center location is the center of Cluster {best_cluster_idx + 1}.")
    print(f"Optimal coordinates: X = {best_location[0]}, Y = {best_location[1]}")
```

```
The optimal logistics center location is the center of Cluster 2
Optimal coordinates: X = 128.18263091269648, Y = 37.134549694830596
```



2. Future Improvements & Considerations

- **Granular Time-Based Analysis**: Further segmentation of the data by time of day and day of week could enable a more detailed understanding of vehicle movement patterns.
- **Incorporating Traffic Congestion Data**: Beyond static road networks, integrating real-time or historical traffic congestion data could lead to more accurate evaluations of accessibility and logistics center placement.
- **Considering Environmental Factors**: In addition to demand and accessibility, future analyses could include environmental variables—such as weather conditions or disaster risk—to ensure more robust and resilient site selection.

2. Demand Forecasting Model

Overview

Accurate demand forecasting is essential for effective logistics and inventory management. Poor forecasts can result in overstock or stockouts, both of which lead to increased logistics costs or missed sales opportunities.

This project aims to build tailored demand forecasting models for each client to help optimize inventory operations. The focus is on **short-term forecasting**, with the goal of minimizing cost and improving operational efficiency.

Data Description

1. Dataset Overview
- The dataset used in this project is time-series logistics data, including variables such as inbound volume, outbound volume, and inventory levels for each client.
- The dataset was provided in CSV format, with key variables summarized below:

Column	Description
`date`	Year/Month of order
`vendor_id`	Vendor ID
`device_type`	Type of logistics device
`quantity`	Units ordered

2. Potential Use of External Data

- Although external data was not used in this project, future enhancements could incorporate additional sources such as weather data, economic indicators, or market trends from competitors to further improve forecasting accuracy.
- These external factors can be especially helpful in capturing demand fluctuations caused by economic volatility or specific events (e.g., promotional periods or seasonal sales).

Data Preprocessing

1. Handling Missing Values

- Missing values are common in time-series datasets. However, no missing entries were found in this dataset, so no additional imputation was required.
- If missing values arise in future datasets, they can be handled using interpolation techniques such as linear interpolation or moving average smoothing.

2. Outlier Removal

- Outliers can significantly affect model performance. In this project, outliers were removed using both the Interquartile Range (IQR) method and the Z-score method.

    - IQR Method:

```
Q1 = grouped_data['quantity'].quantile(0.25)
Q3 = grouped_data['quantity'].quantile(0.75)
IQR = Q3 - Q1
grouped_data = grouped_data[
    (grouped_data['quantity'] >= (Q1 - 1.5 * IQR)) &
    (grouped_data['quantity'] <= (Q3 + 1.5 * IQR))
]
```

- Z-score Method:

```
from scipy.stats import zscore

grouped_data['zscore'] = zscore(grouped_data['quantity'])
grouped_data_clean = grouped_data[(grouped_data['zscore'].abs() <= 3)]
grouped_data_clean = grouped_data_clean.drop('zscore', axis=1)
```

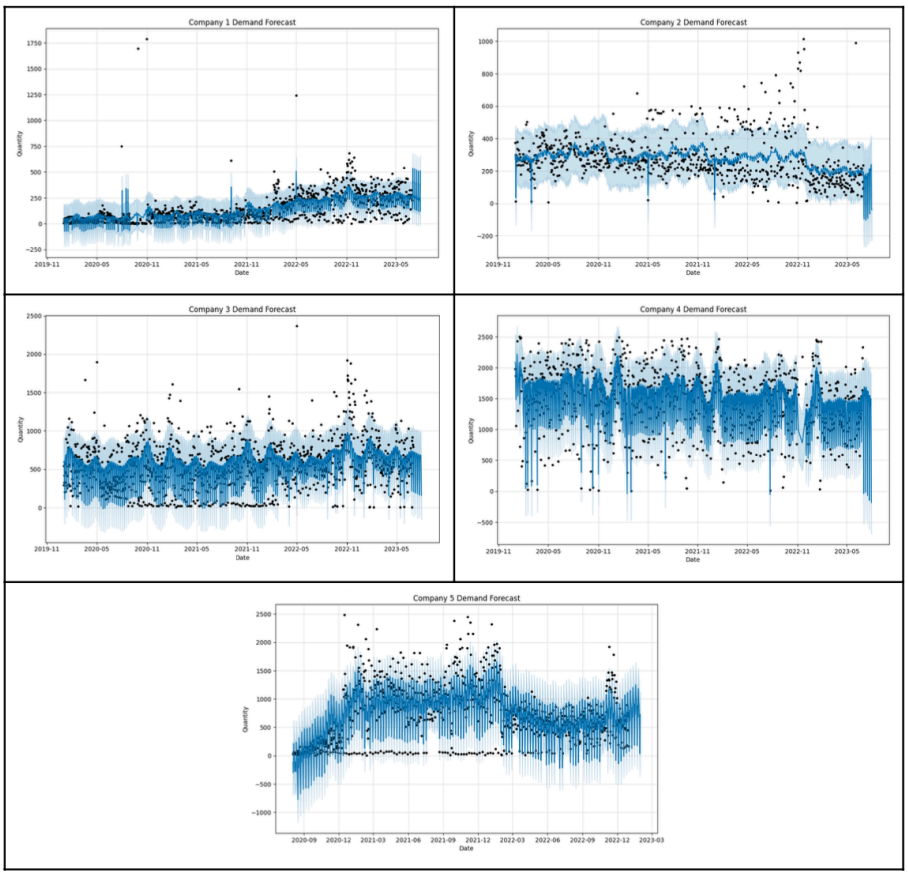
Model Development

This section presents demand forecasting using both Prophet and ARIMA models. Each model was selected to capture the time-series characteristics of the data and generate the most accurate short-term predictions.

1. Prophet Model

   Prophet, developed by Facebook, is a time-series forecasting model that accounts for seasonality, trend, and periodic fluctuations in the data. One of its main advantages is its ability to deliver fast and accurate forecasts without the need for extensive hyperparameter tuning.

   In this project, the Prophet model was used to generate 30-day short-term demand forecasts for each vendor. The parameter daily_seasonality=True was enabled to capture daily seasonal patterns in the data.

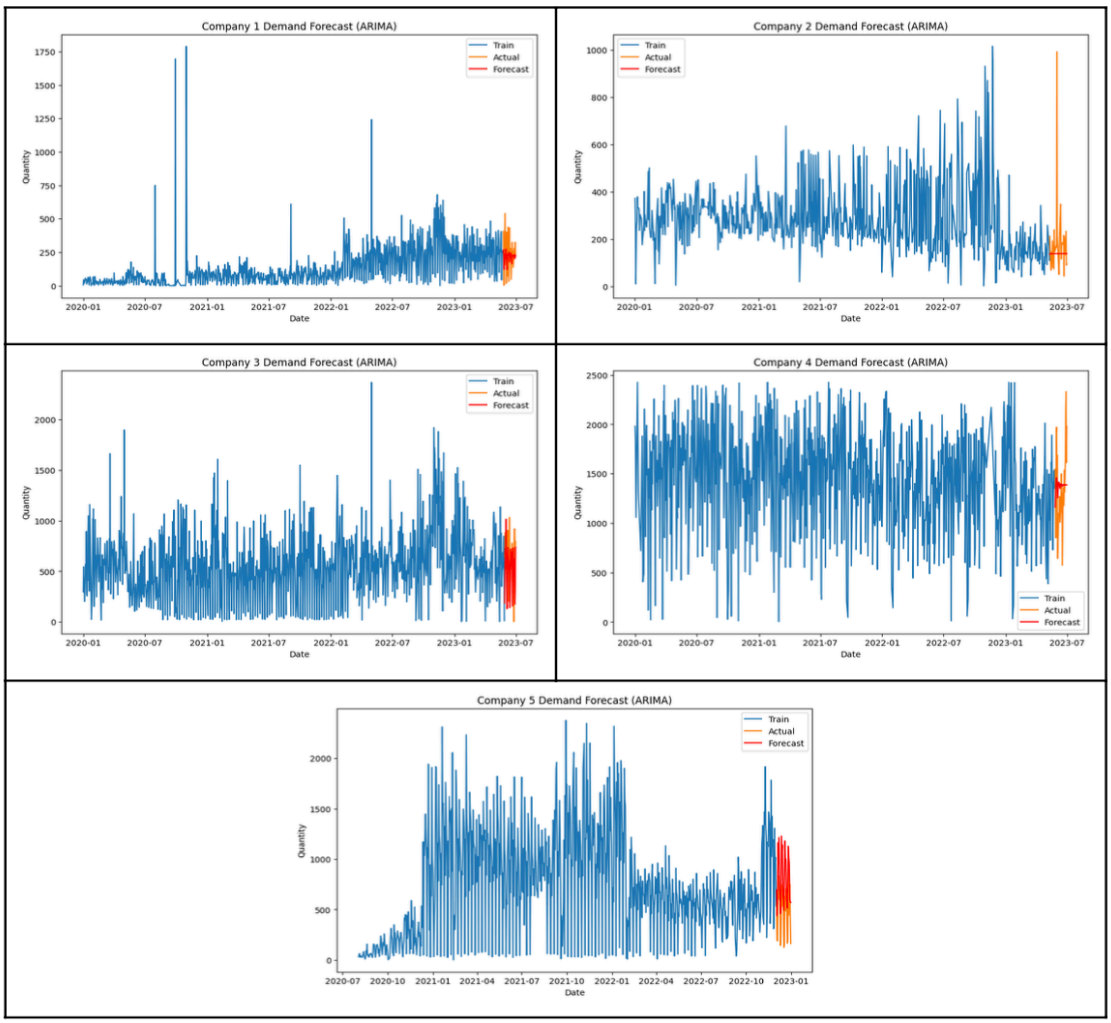


2. Prophet Forecast Results Analysis

- The Prophet model produced forecasts tailored to each vendor's demand pattern, effectively capturing both long-term trends and seasonal fluctuations.
- In the forecast visualizations, the blue line represents the predicted demand, while the shaded area indicates the 95% confidence interval.
- In most cases, the predicted values closely matched the actual demand. For vendors with sudden changes in demand, the model responded by widening the forecast interval to accommodate the increased variability.

3. ARIMA Model

- The ARIMA model, which combines autoregression, differencing, and moving average components, was also applied in this project.
- Before fitting the model, a log transformation was applied to normalize the data distribution and reduce the impact of extreme values.
- Additionally, monthly seasonality was incorporated to improve overall forecasting performance.



4. ARIMA Forecast Results Analysis

- **Log Transformation**: A log transformation was applied to normalize the distribution of the data. This technique was particularly effective in handling datasets with sharp fluctuations in demand, allowing the model to operate in a more stable and robust manner.
- **Monthly Seasonality**: Monthly seasonality was incorporated by setting m=12, which proved especially beneficial for vendors with strong seasonal demand patterns.
- **Forecast Performance**: After inverse transforming the predictions back to the original scale, the ARIMA model produced results that closely aligned with actual demand in most cases.

- The log transformation reduced data volatility and consequently enhanced model stability and accuracy.
- However, some vendors with abrupt demand shifts still showed noticeable forecasting errors.
- In the prediction graphs for each vendor, the **blue line** represents the training data, the **orange line** shows the actual demand, and the **red line** indicates the forecasted demand.
- Overall, the ARIMA model's predictions closely matched the actual values, and the log transformation helped the model remain robust even in the presence of large fluctuations. However, in cases with abrupt and irregular demand changes, the model's accuracy tended to decrease.

Conclusion

1. Short-Term Forecasting Results

- To improve short-term planning, both Prophet and a modified ARIMA model were applied to forecast vendor-specific demand.
- The ARIMA model, in particular, demonstrated improved performance through **log transformation** during preprocessing, playing a key role in enhancing overall forecast accuracy.

[Prophet Model]

- The Prophet model was used to forecast 30-day short-term demand for each vendor, incorporating both long-term trends and seasonal effects.
- Its ability to automatically capture complex patterns in time-series data made it a convenient and effective tool for forecasting.

- **Graph Interpretation:**
  In the Prophet forecast plots, the **blue line** represents the predicted demand, and the **shaded area** indicates the confidence interval.

- In many cases, the forecasts closely aligned with actual demand, effectively reflecting long-term patterns and seasonal behavior.

[ARIMA Model]

- The ARIMA model was trained after applying a log transformation during preprocessing, which normalized the data distribution and improved model performance.
- Additionally, monthly seasonality was incorporated to generate more accurate forecasts, particularly for vendors with strong periodic patterns.

- Graph Interpretation:
  - The ARIMA forecasts, after applying log transformation, appeared significantly more stable and closely aligned with actual demand.
  - In the visualization, the blue line represents the training data, the orange line indicates the actual demand, and the red line shows the forecasted values.
  - The ARIMA model performed particularly well for vendors with relatively stable demand, and the log transformation helped it better handle fluctuations in more volatile datasets.

2. Final System Outcome & Future Improvements

- The demand forecasting system developed in this project has the potential to significantly improve logistics and inventory efficiency.
- By leveraging both Prophet and a modified ARIMA model, the system provided **vendor-specific forecasts**, helping to prevent stockouts and overstock situations. This ultimately supports cost reduction and more efficient supply chain operations.

Future improvements may include the following approaches:

1. Incorporating External Data: Adding external factors such as **economic indicators**, **weather data**, or market signals can enhance forecast accuracy. These variables provide additional explanatory power to the models and help capture demand fluctuations more effectively.

2. Applying Ensemble Techniques: Building an **ensemble model** that combines the strengths of Prophet and the modified ARIMA model could lead to improved forecasting performance. This approach can enhance reliability and better adapt to diverse demand patterns.

3. Developing a Real-Time Forecasting System: Implementing a system that supports **real-time data processing** would enable more agile responses to dynamic market conditions and improve the operational usefulness of the forecasting tool.