


Business Case : Delhivery¹

Table of Contents

1. Problem statement
2. Data Exploration
3. Data cleaning
 - a. Handling Missing values
 - b. Handling Outliers
 - c. Handling categorical Variables
 - d. Normalization / Standardization
4. Feature Extraction and Feature Engineering
5. Visual Analysis
6. Hypothesis Test : In Depth analysis and correlation study
7. Business Insights
8. Recommendations

[Link to Colab](#)

¹  Buisness_case_Delhivery.ipynb

1. Problem Statement

Delhivery, India's leading logistics and supply chain company, aims to improve its data processing pipeline to enhance operational efficiency. The task is to clean, sanitize, and analyze the provided data, which includes trip and delivery information, to build meaningful features and insights. This processed data will support the data science team in building forecasting models, identifying trends, and offering actionable business recommendations.

Key objectives include:

1. **Data Cleaning and Exploration:** Handle missing values, remove outliers, and analyze relationships between fields.
2. **Feature Engineering:** Extract useful features from timestamps, geographic information, and trip data.
3. **Aggregation:** Merge trip-level data into comprehensive summaries for analysis.
4. **Hypothesis Testing:** Compare and validate relationships between different time and distance metrics.
5. **Outlier Treatment and Scaling:** Handle anomalies and normalize numerical features for consistency.
6. **Actionable Insights:** Identify patterns in routes, delivery times, and distances to optimize operations.

The ultimate goal is to provide the company with insights and recommendations to improve the quality, efficiency, and profitability of their logistics operations.

2. Data Exploration

1. Dataset loading

```
df=pd.read_csv('delhivery_data.csv')
df.head()
```

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_center	source_name	destination_center	destination_name	od_start_time	...	c
0	training	2018-09-20 02:35:36.476840	thanos:sroute:eb7bfc78- b351-4c0e-a951- fa3d5c3...	Carting	153741093647649320	trip- IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND388620AAB	Khambhat_MotvdDPP_D (Gujarat)	2018-09-20 03:21:32.418600	...	c
1	training	2018-09-20 02:35:36.476840	thanos:sroute:eb7bfc78- b351-4c0e-a951- fa3d5c3...	Carting	153741093647649320	trip- IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND388620AAB	Khambhat_MotvdDPP_D (Gujarat)	2018-09-20 03:21:32.418600	...	c
2	training	2018-09-20 02:35:36.476840	thanos:sroute:eb7bfc78- b351-4c0e-a951- fa3d5c3...	Carting	153741093647649320	trip- IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND388620AAB	Khambhat_MotvdDPP_D (Gujarat)	2018-09-20 03:21:32.418600	...	c
3	training	2018-09-20 02:35:36.476840	thanos:sroute:eb7bfc78- b351-4c0e-a951- fa3d5c3...	Carting	153741093647649320	trip- IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND388620AAB	Khambhat_MotvdDPP_D (Gujarat)	2018-09-20 03:21:32.418600	...	c

2. The dataset has **144,867 rows** and **24 columns**. Here's a breakdown of what this means:

- **Rows (144,867)**: Each row represents a unique record of a delivery trip or segment of a trip.
- **Columns (24)**: These include various attributes related to the trip, such as timestamps, locations, times, distances, and other details.

This size suggests a comprehensive dataset that can provide significant insights into delivery patterns, efficiency, and potential optimization areas.

3. Here is a brief summary of the key statistics for the numerical columns of the dataset:

- **Row Count**: 144,867
- **Key Metrics**:
 - **start_scan_to_end_scan**: Average ~961 (likely seconds), with a wide range (20 to 7,898), indicating highly varied delivery times.
 - **actual_distance_to_destination**: Average ~234 km, ranging from ~9 km to ~1,927 km.
 - **actual_time**: Average ~417 (seconds), with a large spread, maxing out at 4,532.

- **osrm_time**: Predicted routing engine time averages ~214, shorter than actual times, suggesting delays.
- **osrm_distance**: Predicted routing distances average ~285 km, aligning closely with actual distances.

Other columns, like **cutoff_factor** and **segment_actual_time**, also show variability, requiring further exploration to understand their distribution and relationships. The large ranges and outliers are apparent, indicating the need for handling and normalization.

4. Column Profiling (Key Features):

data: Indicates whether the row pertains to training or testing data.

trip_creation_time: Timestamp of when the trip was created.

route_schedule_uuid: Unique identifier for a specific route schedule.

route_type: Specifies transportation type:

- **FTL (Full Truck Load)**: Trucks carry goods directly to the destination without intermediate stops.
- **Carting**: Involves handling systems with small vehicles (carts).

trip_uuid: Unique identifier for a specific trip, which may span multiple source and destination centers.

source_center: Identifier for the origin center of the trip.

source_name: Full name of the origin center, including city, region, and state.

destination_center: Identifier for the destination center of the trip.

destination_name: Full name of the destination center, including city, region, and state.

od_start_time: Timestamp indicating when the trip started.

od_end_time: Timestamp indicating when the trip ended.

start_scan_to_end_scan: Time taken for delivery from source to destination.

is_cutoff: Binary field (True/False), purpose unclear, possibly indicating if a trip was marked for cutoff.

cutoff_factor: Numerical value; purpose not explicitly defined.

cutoff_timestamp: Timestamp for a cutoff event.

actual_distance_to_destination: Distance (in kilometers) between the source and destination warehouses.

actual_time: Total time (in seconds) taken to complete the delivery, aggregated over all trip segments.

osrm_time: Predicted delivery time (in seconds) calculated by the OSRM (Open Source Routing Machine), which factors in typical traffic and optimized routes.

osrm_distance: Predicted delivery distance (in kilometers) calculated by OSRM.

factor: Unexplained numerical value; requires further exploration to understand its role.

segment_actual_time: Actual delivery time (in seconds) for a specific segment of the trip.

segment_osrm_time: Predicted delivery time (in seconds) for a specific segment calculated by OSRM.

segment_osrm_distance: Predicted delivery distance (in kilometers) for a specific segment calculated by OSRM.

segment_factor: Numerical value; unclear purpose, possibly related to segment efficiency or quality.

3. Data Cleaning

3.a) Handling Missing values

The dataset contains missing values in two columns:

1. **source_name**: 293 missing values.
2. **destination_name**: 261 missing values.

All other columns have no missing values.

These missing entries in **source_name** and **destination_name** should be addressed during data preprocessing, as they might affect subsequent feature extraction and analysis.

Numerical Columns:

- For columns with numerical data types (**np.number**), missing values are replaced with the **mean** of the respective column. This ensures the overall distribution remains consistent while filling gaps.

Categorical Columns:

- For columns with object data types (**'object'**), missing values are replaced with the **most frequent value** (mode) of the column. This is a common strategy to handle missing values in categorical fields.

Result:

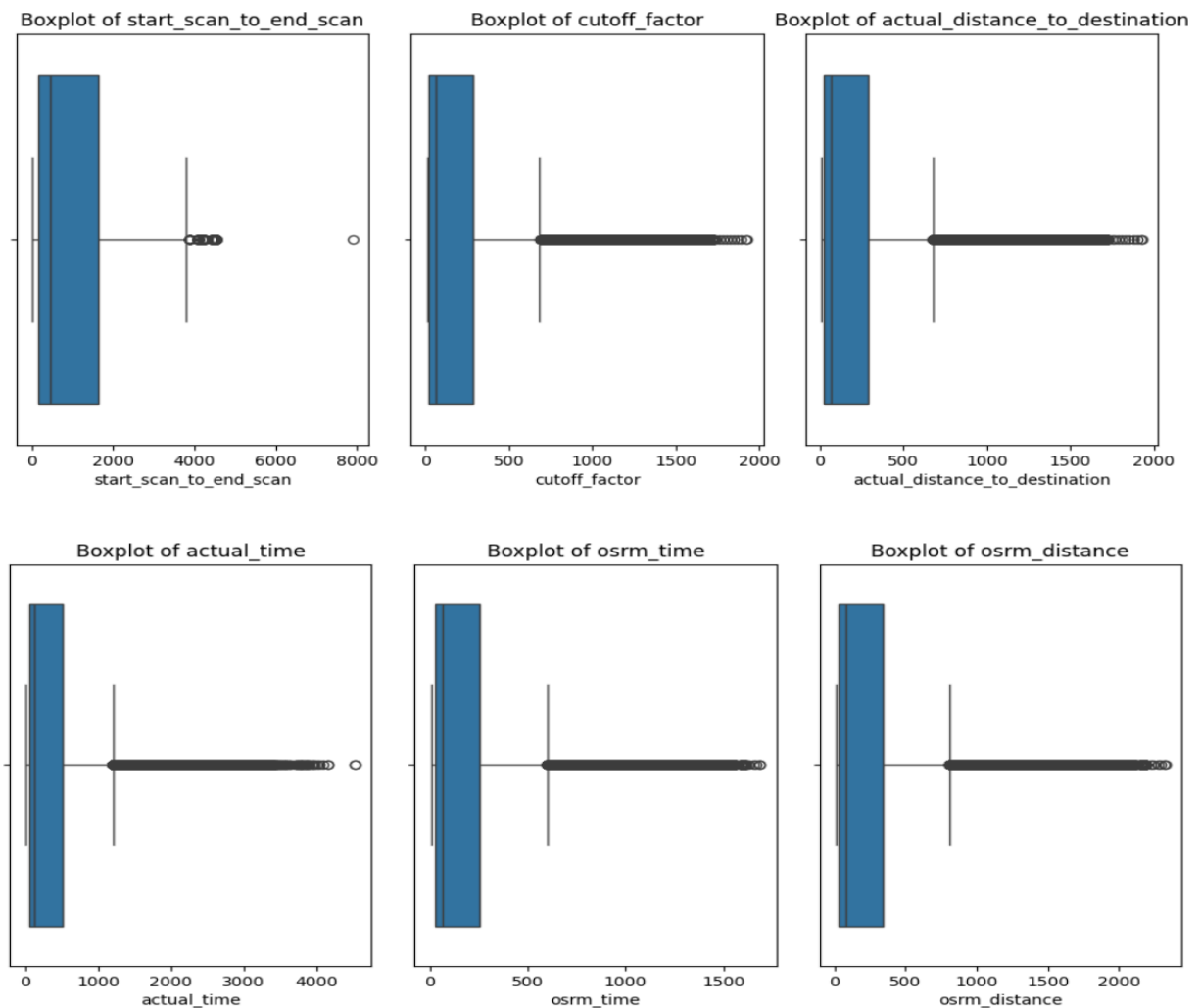
- After applying this approach, all missing values in the dataset are addressed, as confirmed by checking **df.isnull().sum()**.

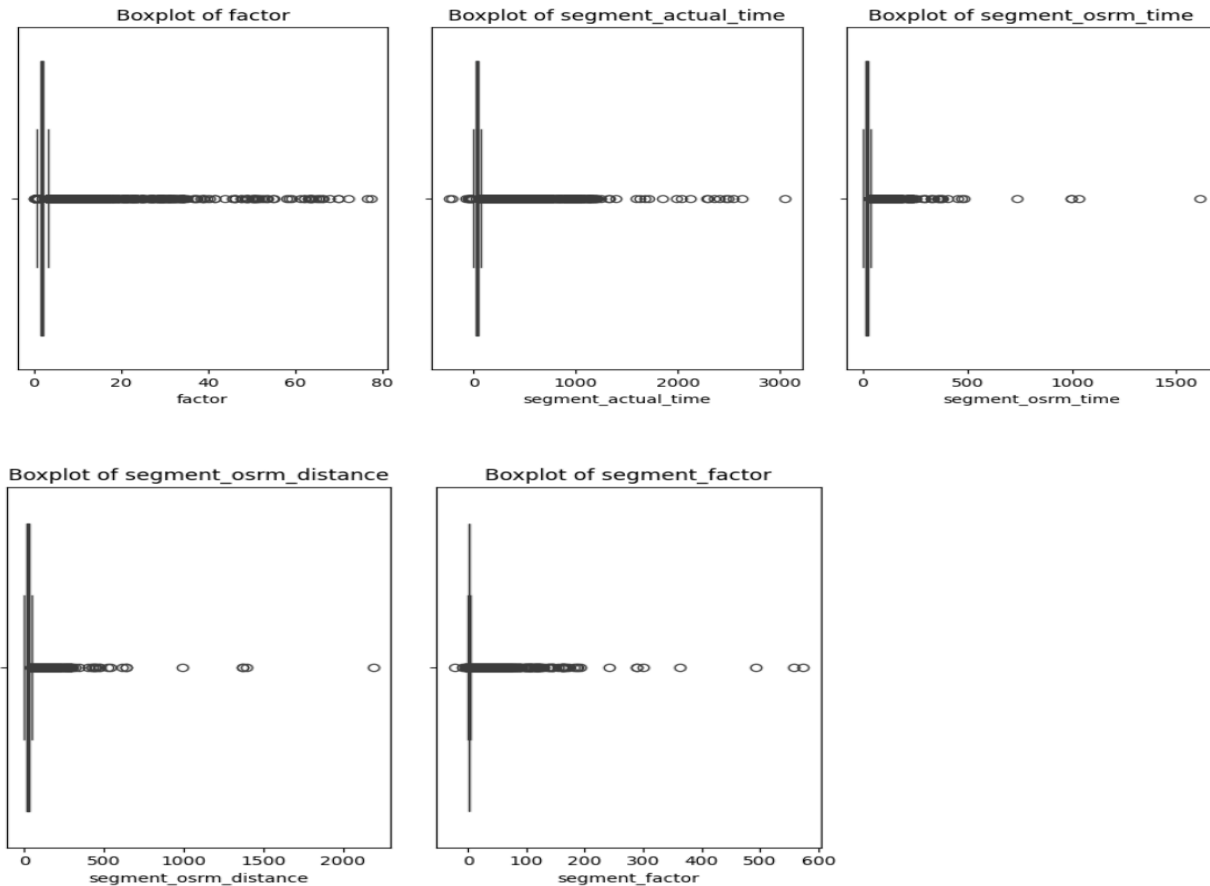
3.b) Handling Outliers

In this analysis, boxplots were used to visualize the distribution of numerical features within the dataset. For each numerical column in the DataFrame (**df**), a boxplot was generated to assess the spread, central tendency, and potential outliers of the data.

The code iterates through the list of numeric columns (`numeric_cols`), and for each column, a boxplot is created using Seaborn's `boxplot` function. This visualization provides insights into the following:

- **Median:** The central line within the box indicates the median value, offering a quick view of the data's central tendency.
- **Interquartile Range (IQR):** The box represents the range between the first and third quartiles, indicating the middle 50% of the data.
- **Outliers:** Points outside the whiskers (typically 1.5 times the IQR) are considered potential outliers, which could warrant further investigation or preprocessing.





After handling outliers using the IQR method, the key results are:

1. **Outlier Removal:** The dataset was cleaned by removing data points that fell outside the calculated bounds ($1.5 \times \text{IQR}$ above Q3 or below Q1) for each numerical column.
2. **Dataset Shape:** The resulting dataset, after outlier removal, has a shape of `(130891, 24)`, indicating that the number of rows and columns has been maintained while outliers were removed from the numerical columns.
3. **Improved Data Quality:** With the outliers removed, the dataset is more robust, leading to more accurate analysis and modeling, free from the influence of extreme values.

These results suggest that the dataset is now cleaner, with outliers properly handled to ensure more reliable insights from the analysis.


3.c) Handling Categorical Variables

One-hot encoding is a technique used to convert categorical variables into a numerical format that can be used in machine learning models. It transforms each category into a new binary (0 or 1) column.

For example, if the categorical variable `route_type` has three categories (e.g., "A", "B", and "C"), one-hot encoding would create three new binary columns:

- `route_type_A`: 1 if the `route_type` is "A", otherwise 0.
- `route_type_B`: 1 if the `route_type` is "B", otherwise 0.
- `route_type_C`: 1 if the `route_type` is "C", otherwise 0.

3.d) Normalization/Standardization of Columns

```
 normalized_df=outlier_cleaned_df  
scaler=MinMaxScaler()  
normalized_df[numeric_cols]=scaler.fit_transform(outlier_cleaned_df[numeric_cols])
```

The code normalizes the numerical features in the cleaned dataset (`outlier_cleaned_df`) using Min-Max scaling. It scales the values of each numeric column in the range [0, 1], ensuring all features contribute equally to machine learning models. The result is stored in `normalized_df`.

4. Feature Extraction and Feature Engineering

1. Extracting City and State from `source_name`:

- `source_city`: Extracts the city name from `source_name` before the first underscore (`_`).
- `source_state`: Extracts the state enclosed in parentheses from `source_name`.

2. Extracting City and State from `destination_name`:

- `destination_city`: Extracts the city name from `destination_name` before the first underscore (`_`).
- `destination_state`: Extracts the state enclosed in parentheses from `destination_name`.

3. Date and Time Features from `trip_creation_time`:

- `trip_creation_time`: Converts the `trip_creation_time` column to datetime format.
- `trip_month`: Extracts the month from `trip_creation_time`.
- `trip_day`: Extracts the day from `trip_creation_time`.
- `trip_hour`: Extracts the hour from `trip_creation_time`.

4. Calculating Delivery Times and Discrepancies:

- `total_delivery_time`: Calculates the total delivery time in seconds by subtracting `od_start_time` from `od_end_time`.
- `time_discrepancy`: Computes the discrepancy between `start_scan_to_end_scan` and `total_delivery_time`.
- Converts `total_delivery_time` from seconds to days for easier comparison with `start_scan_to_end_scan`.

5. Aggregated Features:

- Grouping by `trip_uuid`:

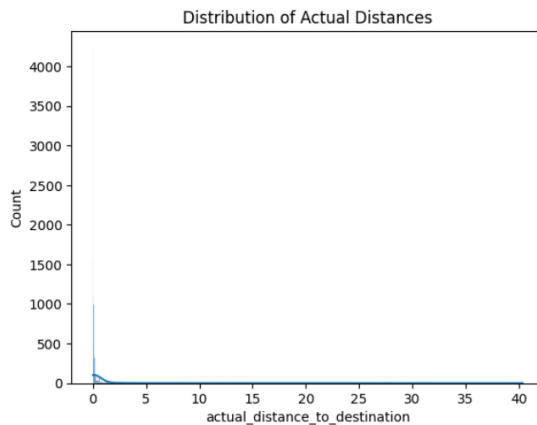
- Aggregates data at the `trip_uuid` level to sum various time and distance metrics (e.g., `actual_distance_to_destination`, `actual_time`, `osrm_time`, etc.).
- Retains the first value of categorical features like `route_type`, `source_city`, `destination_city`, etc., for each `trip_uuid`.

These feature engineering steps help in enriching the dataset, creating more insightful and structured features for analysis or modeling.

5. Visual analysis

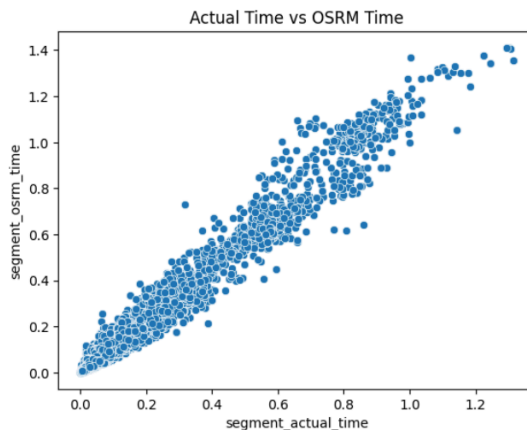
Distribution of Distances:

- A histogram with a Kernel Density Estimate (KDE) overlay is used to visualize the distribution of the `actual_distance_to_destination` in the dataset. This plot helps in understanding the spread and central tendency of actual distances, and the KDE line provides a smoother estimate of the distribution.



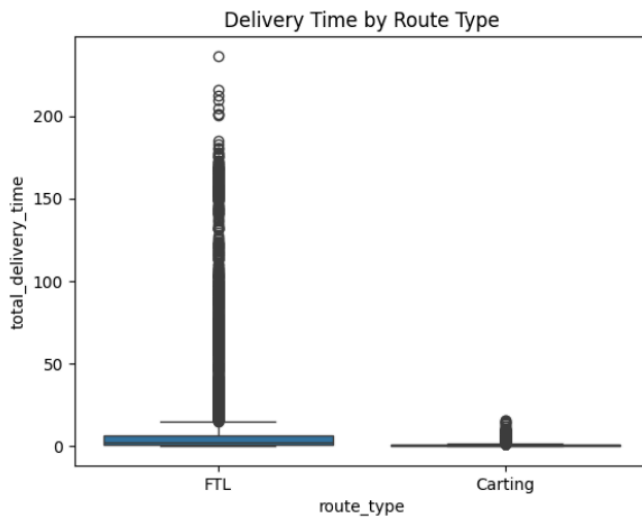
Comparison of Times:

- A scatter plot is created to compare `segment_actual_time` against `segment_osrm_time`. This plot allows for visual identification of any relationship between the actual segment time and the predicted time based on the OSRM model. Points that deviate significantly from the line of equality might indicate discrepancies between actual and predicted times.



Delivery Time by Route Type:

- A boxplot is used to compare the **total_delivery_time** across different **route_type** categories. The plot visualizes the median, interquartile range, and any outliers, providing insight into how delivery times vary by route type.



6. Hypothesis Test : In Depth analysis and correlation study

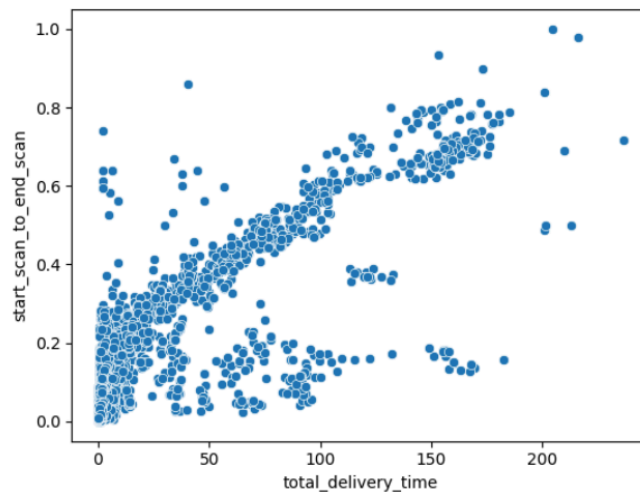
6.1) Comparison between **total_delivery_time** and **start_scan_to_end_scan**

- **Hypotheses:**

- **H₀:** There is no significant difference between **total_delivery_time** and **start_scan_to_end_scan**. ($\mu_1 = \mu_2$)
- **H₁:** There is a significant difference between **total_delivery_time** and **start_scan_to_end_scan**. ($\mu_1 \neq \mu_2$)

- **Test Method:**

- **Pearson Correlation:** The correlation coefficient is 0.8547, with a p-value of 0.0, indicating a strong positive correlation between the two variables.



- **T-test:** The t-statistic is 33.15 with a very low p-value (1.58e-236), suggesting a significant difference between the two distributions.

- **Conclusion:**

- Since the **p-value for both the Pearson correlation and T-test is less than 0.05**, we reject the null hypothesis, concluding that there is a significant correlation and difference between **total_delivery_time** and **start_scan_to_end_scan**.
-

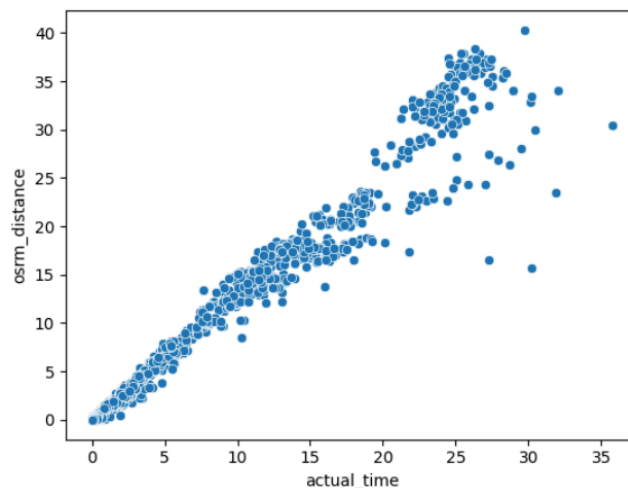
6.2) Comparison between **actual_time** and **osrm_distance**

- **Hypotheses:**

- **H₀:** There is no significant correlation between **actual_time** and **osrm_distance**. ($\rho = 0$)
- **H₁:** There is a significant correlation between **actual_time** and **osrm_distance**. ($\rho \neq 0$)

- **Test Method:**

- **Pearson Correlation:** The correlation coefficient is 0.9906, with a p-value of 0.0, indicating a very strong positive correlation between **actual_time** and **osrm_distance**.



- **T-test:** The t-statistic is -4.60, with a p-value of 4.19e-06, indicating a significant difference between the two variables.

- **Conclusion:**

- The **p-value is less than 0.05**, so we reject the null hypothesis. There is a significant correlation and difference between **actual_time** and **osrm_distance**.
-

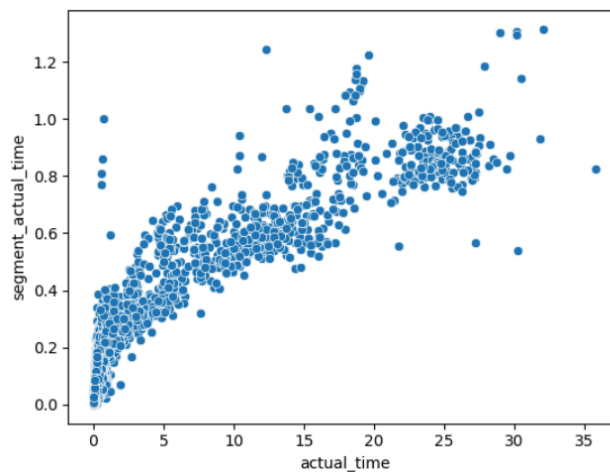
6.3) Comparison between **actual_time** and **segment_actual_time**

- **Hypotheses:**

- **H₀:** There is no significant correlation between **actual_time** and **segment_actual_time**. ($\rho = 0$)
- **H₁:** There is a significant correlation between **actual_time** and **segment_actual_time**. ($\rho \neq 0$)

- **Test Method:**

- **Pearson Correlation:** The correlation coefficient is 0.8936, with a p-value of 0.0, indicating a strong positive correlation between **actual_time** and **segment_actual_time**.



- **T-test:** The t-statistic is 28.25 with a p-value of 3.44e-173, indicating a significant difference between the two distributions.

- **Conclusion:**

- The **p-value is less than 0.05**, so we reject the null hypothesis. There is a significant correlation and difference between **actual_time** and **segment_actual_time**.
-

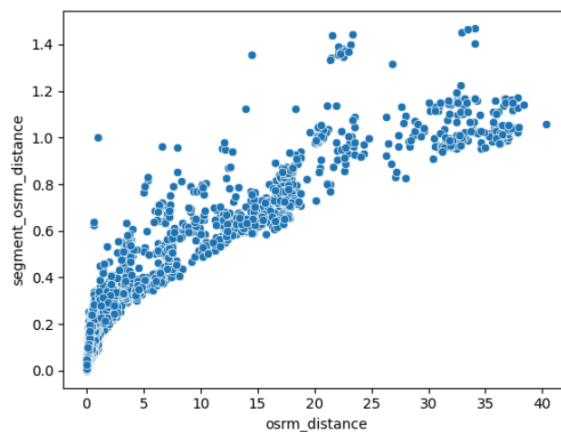
6.4) Comparison between **osrm_distance** and **segment_osrm_distance**

- **Hypotheses:**

- **H₀:** There is no significant correlation between **osrm_distance** and **segment_osrm_distance**. ($\rho = 0$)
- **H₁:** There is a significant correlation between **osrm_distance** and **segment_osrm_distance**. ($\rho \neq 0$)

- **Test Method:**

- **Pearson Correlation:** The correlation coefficient is 0.9090, with a p-value of 0.0, indicating a very strong positive correlation between **osrm_distance** and **segment_osrm_distance**.



- **T-test:** The t-statistic is 27.95 with a p-value of 1.21e-169, suggesting a significant difference between the two variables.

- **Conclusion:**

- The **p-value is less than 0.05**, so we reject the null hypothesis. There is a significant correlation and difference between **osrm_distance** and **segment_osrm_distance**.

Summary of Conclusions:

- For all the comparisons made between different time and distance variables (e.g., **total_delivery_time** vs **start_scan_to_end_scan**, **actual_time** vs **osrm_distance**, etc.), the p-values were all less than 0.05, indicating strong evidence to **reject the null hypothesis**. This suggests significant correlations and differences exist between the variables being analyzed.

7. **Business Insights**

1. Most Common Source-Destination Pairs (City-Wise):
 - The most common corridor at the city level is between Bengaluru to Bengaluru, indicating a highly frequent local transportation route within the city.
2. Most Common Source-Destination Pairs (State-Wise):
 - The most common corridor at the state level is between Maharashtra to Maharashtra, highlighting frequent intra-state deliveries within Maharashtra.
3. Most Common Source-Destination Pairs (City-State-Wise):
 - The most common corridor at the city-state level is between Bengaluru (Karnataka) to Bengaluru (Karnataka), confirming the importance of local city-to-city connections within the same state.
4. Average Delivery Time:
 - The average delivery time across all corridors is 6.26 days, suggesting relatively fast delivery times across the dataset.

These insights provide a better understanding of the transportation flow, with a focus on local and intra-state corridors, and suggest that delivery times are generally efficient.

8. Recommendation

Optimize Local and Intra-State Routes:

- Focus on high-frequency city-to-city and state-to-state corridors for resource allocation and efficiency improvements.

Improve Delivery Time Efficiency:

- Address discrepancies between `total_delivery_time` and `start_scan_to_end_scan` to streamline delivery processes.

Monitor Time Discrepancies:

- Reduce differences between actual time and segment actual time for better route consistency and planning.

Enhance Route Planning:

- Use the strong correlation between `actual_time` and `osrm_distance` to improve predictive accuracy and route planning.

Further Optimize Delivery Times:

- Investigate opportunities to reduce the average delivery time of 6.26 seconds, focusing on high-volume routes.

Invest in Automation and Technology:

- Utilize automation to address time discrepancies and optimize delivery schedules on high-correlation routes.

Allocate Resources to High-Traffic Routes:

- Assign more resources to the most common corridors to improve operational capacity and reduce delivery time.

