About Delhivery:

Delhivery is India's largest and fastest-growing fully integrated logistics service provider by revenue as of Fiscal 2021. The company is focused on building an operating system for commerce by leveraging world-class infrastructure, top-tier logistics operations, and advanced engineering and technology capabilities. Delhivery aims to drive innovation across the supply chain, providing seamless logistics solutions that enhance the efficiency and profitability of their services.

The Data team at Delhivery plays a pivotal role in unlocking insights from vast amounts of data generated through their operations. By leveraging cutting-edge analytics and machine learning techniques, the team develops intelligent systems that drive quality, efficiency, and a competitive edge in the marketplace.

Dataset:

https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/551/original/delhivery_data.csv?

1642751181

Column Profiling:

1. **data** - tells whether the data is testing or training data.
2. **trip_creation_time** – Timestamp of trip creation
3. **route_schedule_uuid** – Unique ID for a particular route schedule
4. **route_type** – Transportation type
   a. **FTL** – Full Truck Load: FTL shipments get to the destination sooner, as the truck is making no other pickups or drop-offs along the way
   b. **Carting**: Handling system consisting of small vehicles (carts)
5. **trip_uuid** - Unique ID given to a particular trip (A trip may include different source and destination centers)
6. **source_center** - Source ID of trip origin
7. **source_name** - Source Name of trip origin
8. **destination_cente** – Destination ID

9. **destination_name** – Destination Name
10. **od_start_time** – Trip start time
11. **od_end_time** – Trip end time
12. **start_scan_to_end_scan** – Time taken to deliver from source to destination
13. **is_cutoff** – Unknown field
14. **cutoff_factor** – Unknown field
15. **cutoff_timestamp** – Unknown field
16. **actual_distance_to_destination** – Distance in kms between source and destination warehouse.
17. **actual_time** – Actual time taken to complete the delivery (Cumulative)
18. **osrm_time** – An open-source routing engine time calculator which computes the shortest path between points in a given map (Includes usual traffic, distance through major and minor roads) and gives the time (Cumulative)
19. **osrm_distance** – An open-source routing engine which computes the shortest path between points in a given map (Includes usual traffic, distance through major and minor roads) (Cumulative)
20. **factor** – Unknown field
21. **segment_actual_time** – This is a segment time. Time taken by the subset of the package delivery
22. **segment_osrm_time** – This is the OSRM segment time. Time taken by the subset of the package delivery
23. **segment_osrm_distance** – This is the OSRM distance. Distance covered by subset of the package delivery
24. **segment_factor** – Unknown field

Content:

1. **Data Cleaning and Exploration:**
2. **Merging Rows (Feature Aggregation)**
3. **Feature Engineering**
4. **Hypothesis Testing and Visual Analysis**
    a. **Outlier Detection and Treatment**
    b. **Handling Categorical Variables**
    c. **Normalization/Standardization**
5. **Business Insights**
6. **Recommendations**

1. Data Cleaning and Exploration:

Input:

```python
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.preprocessing import StandardScaler, MinMaxScaler

from scipy import stats

df = pd.read_csv('/content/delhivery_data.csv')

df.sample(5)
```

Output:

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_center	source_name	destination_center	destination_name	od_start_time	...	cutoff_timestamp
125943	training	2018-09-13 18:07:09.221216	thanos::sroute:7bfd3a15-7880-43fa-8bf4-67d3f0a...	Carting	trip-153686202922095540	IND000000AEM	Gurgaon_Bilaspur_RP (Haryana)	IND110037AAB	Del_B_RPC (Delhi)	2018-09-13 18:07:09.221216	...	2018-09-14 00:19:25
7480	training	2018-09-15 05:42:30.897323	thanos::sroute:147ddb06-42e6-4598-ae86-6cb0862...	Carting	trip-153699015089709737	IND335703AAA	Srivijaynagar_BhwanDPP_D (Rajasthan)	IND335701AAA	Anupgarh_PrmNrDPP_D (Rajasthan)	2018-09-15 08:49:12.831814	...	2018-09-15 09:09:22
106408	training	2018-09-21 23:17:14.009954	thanos::sroute:de5e208e-7641-45e6-8100-4d9fb1e...	FTL	trip-153757183400969130	IND562132AAA	Bangalore_Nelmngla_H (Karnataka)	IND000000ACB	Gurgaon_Bilaspur_HB (Haryana)	2018-09-21 23:17:14.009954	...	2018-09-22 20:06:25
135627	training	2018-09-15 00:26:15.557316	thanos::sroute:562d2584-a406-4fc9-82ac-3d8e65a...	FTL	trip-153697117555705915	IND110037AAM	Delhi_Airport_H (Delhi)	IND209304AAA	Kanpur_Central_H_6 (Uttar Pradesh)	2018-09-15 00:26:15.557316	...	2018-09-15 10:49:22
122472	test	2018-09-29 05:40:57.117794	thanos::sroute:14c55592-ba2e-4f72-820c-3a22334...	FTL	trip-153819965711740203	IND462022AAA	Bhopal_Trnsport_H (Madhya Pradesh)	IND000000ACB	Gurgaon_Bilaspur_HB (Haryana)	2018-09-30 18:59:56.924720	...	2018-10-01 15:26:28

Input:

```
# Convert time columns to datetime

df['trip_creation_time'] = pd.to_datetime(df['trip_creation_time'])

df['od_start_time'] = pd.to_datetime(df['od_start_time'])

df['od_end_time'] = pd.to_datetime(df['od_end_time'])
```

```
# Handle missing values

df = df.dropna(how='any')

df = df.reset_index(drop=True)

# Check the structure

print(df.info())

print(df.describe())

# Reset the index of the DataFrame after dropping rows, without adding the old index as a column

df = df.reset_index(drop=True)

df.head(5)
```

Output:

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_center	source_name	destination_center	destination_name	od_start_time	...	cutoff_timestamp	actual_dist
0	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND388620AAB	Khambhat_MotvdDPP_D (Gujarat)	2018-09-20 03:21:32.418600	...	2018-09-20 04:27:55	
1	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND388620AAB	Khambhat_MotvdDPP_D (Gujarat)	2018-09-20 03:21:32.418600	...	2018-09-20 04:17:55	
2	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND388620AAB	Khambhat_MotvdDPP_D (Gujarat)	2018-09-20 03:21:32.418600	...	2018-09-20 04:01:19.505586	
3	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND388620AAB	Khambhat_MotvdDPP_D (Gujarat)	2018-09-20 03:21:32.418600	...	2018-09-20 03:39:57	
4	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND388620AAB	Khambhat_MotvdDPP_D (Gujarat)	2018-09-20 03:21:32.418600	...	2018-09-20 03:33:55	

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 144316 entries, 0 to 144315
Data columns (total 24 columns):
 #   Column                          Non-Null Count   Dtype
---  ------                          --------------   -----
 0   data                            144316 non-null  object
 1   trip_creation_time              144316 non-null  datetime64[ns]
 2   route_schedule_uuid             144316 non-null  object
 3   route_type                      144316 non-null  object
 4   trip_uuid                       144316 non-null  object
 5   source_center                   144316 non-null  object
 6   source_name                     144316 non-null  object
 7   destination_center              144316 non-null  object
 8   destination_name                144316 non-null  object
 9   od_start_time                   144316 non-null  datetime64[ns]
 10  od_end_time                     144316 non-null  datetime64[ns]
 11  start_scan_to_end_scan          144316 non-null  float64
 12  is_cutoff                       144316 non-null  bool
 13  cutoff_factor                   144316 non-null  int64
 14  cutoff_timestamp                144316 non-null  object
 15  actual_distance_to_destination  144316 non-null  float64
 16  actual_time                     144316 non-null  float64
 17  osrm_time                       144316 non-null  float64
 18  osrm_distance                   144316 non-null  float64
 19  factor                          144316 non-null  float64
 20  segment_actual_time             144316 non-null  float64
 21  segment_osrm_time               144316 non-null  float64
 22  segment_osrm_distance           144316 non-null  float64
 23  segment_factor                  144316 non-null  float64
dtypes: bool(1), datetime64[ns](3), float64(10), int64(1), object(9)
memory usage: 25.5+ MB
None
```

```
                  trip_creation_time                  od_start_time  \
count                         144316                         144316
mean   2018-09-22 13:05:09.454117120  2018-09-22 17:32:42.435769344
min       2018-09-12 00:00:16.535741     2018-09-12 00:00:16.535741
25%    2018-09-17 02:46:11.004421120  2018-09-17 07:37:35.014584832
50%    2018-09-22 03:36:19.186585088  2018-09-22 07:35:23.038482944
75%    2018-09-27 17:53:19.027942912  2018-09-27 22:01:30.861209088
max       2018-10-03 23:59:42.701692     2018-10-06 04:27:23.392375
std                              NaN                            NaN

                         od_end_time  start_scan_to_end_scan  cutoff_factor  \
count                         144316           144316.000000  144316.000000
mean   2018-09-23 09:36:54.057172224              963.697698     233.561345
min       2018-09-12 00:50:10.814399               20.000000       9.000000
25%       2018-09-18 01:29:56.978912              161.000000      22.000000
50%    2018-09-23 02:49:00.936600064              451.000000      66.000000
75%    2018-09-28 12:13:41.675546112             1645.000000     286.000000
max       2018-10-08 03:00:24.353479             7898.000000    1927.000000
std                              NaN             1038.082976     345.245823

       actual_distance_to_destination    actual_time      osrm_time  \
count                   144316.000000  144316.000000  144316.000000
mean                       234.708498     417.996237     214.437055
min                          9.000045       9.000000       6.000000
25%                         23.352027      51.000000      27.000000
50%                         66.135322     132.000000      64.000000
75%                        286.919294     516.000000     259.000000
max                       1927.447705    4532.000000    1686.000000
std                        345.480571     598.940065     308.448543

       osrm_distance         factor  segment_actual_time  segment_osrm_time  \
count  144316.000000  144316.000000        144316.000000      144316.000000
mean      285.549785       2.120178            36.175379          18.495697
min         9.008200       0.144000          -244.000000           0.000000
25%        29.896250       1.604545            20.000000          11.000000
50%        78.624400       1.857143            28.000000          17.000000
75%       346.305400       2.212280            40.000000          22.000000
max      2326.199100      77.387097          3051.000000        1611.000000
std       421.717826       1.717065            53.524298          14.774008
```

Insights:

- The dataset was successfully loaded into the environment, allowing us to initiate our analysis. It contains comprehensive fields such as trip details, timings, route information, and other logistics-related data. To prepare the environment for efficient data manipulation and visualization, we imported essential Python libraries including pandas, numpy, and matplotlib.
- Key datetime columns, such as trip_creation_time, od_start_time, and od_end_time, were converted to the appropriate datetime format. This transformation facilitates time-based operations, such as calculating trip durations and analyzing temporal trends across various timeframes, making the data ready for in-depth analysis.
- To handle missing values, we employed a strategy of filling null entries with default values (0). This ensures continuity in subsequent analysis without disruption due to incomplete data. An initial exploration of the dataset using methods like .info() and .describe() provided valuable insights into the structure of the dataset. We examined data types, identified missing values, and obtained summary statistics such as mean, minimum, and maximum values, among others.

Explanation:

- This preparatory step was crucial in making the dataset operational for further analysis. By addressing missing values and transforming datetime fields, we ensured the data's integrity and usability. Additionally, the exploration of structural details and descriptive statistics offered a foundational understanding of the dataset, informing the direction for subsequent analytical steps. This groundwork sets the stage for effective feature engineering, hypothesis testing, and deeper insights into logistical performance.

2. Merging Rows (Feature Aggregation)

Input:

```
# Create the segment_key by concatenating trip_uuid, source_center, and
destination_center

df['segment_key'] = df['trip_uuid'] + "_" + df['source_center'] + "_" +
df['destination_center']

# Aggregation functions for merging

df['segment_actual_time_sum'] = df.groupb
y('segment_key')['segment_actual_time'].cumsum()

df['segment_osrm_distance_sum'] =
df.groupby('segment_key')['segment_osrm_distance'].cumsum()

df['segment_osrm_time_sum'] =
df.groupby('segment_key')['segment_osrm_time'].cumsum()
```

Output:

```
   segment_actual_time_sum  segment_osrm_distance_sum  segment_osrm_time_sum
0                     14.0                    11.9653                   11.0
1                     24.0                    21.7243                   20.0
2                     40.0                    32.5395                   27.0
3                     61.0                    45.5619                   39.0
4                     67.0                    49.4772                   44.0
5                     15.0                    12.1171                   11.0
6                     43.0                    21.2890                   17.0
7                     64.0                    35.8252                   28.0
8                     74.0                    47.1900                   38.0
9                    100.0                    53.2334                   44.0
```

```
df.head(5)
```

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_center	source_name	destination_center	destination_name	od_start_time	...	osrm_distance	factor	segme
0	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND388620AAB	Khambhat_MotvdDPP_D (Gujarat)	2018-09-20 03:21:32.418600	...	11.9653	1.272727	
1	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND388620AAB	Khambhat_MotvdDPP_D (Gujarat)	2018-09-20 03:21:32.418600	...	21.7243	1.200000	
2	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND388620AAB	Khambhat_MotvdDPP_D (Gujarat)	2018-09-20 03:21:32.418600	...	32.5395	1.428571	
3	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND388620AAB	Khambhat_MotvdDPP_D (Gujarat)	2018-09-20 03:21:32.418600	...	45.5620	1.550000	
4	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)	IND388620AAB	Khambhat_MotvdDPP_D (Gujarat)	2018-09-20 03:21:32.418600	...	54.2181	1.545455	

3. **Feature Engineering**

Input:

```
# Feature: Time difference in hours between od_start_time and
od_end_time

segment_data['od_time_diff_hour'] = (segment_data['od_end_time'] -
segment_data['od_start_time']).dt.total_seconds() / (3600)



# Feature: Extract year, month, and day from trip_creation_time

segment_data['trip_year'] =
segment_data['trip_creation_time'].dt.year

segment_data['trip_month'] =
segment_data['trip_creation_time'].dt.month

segment_data['trip_day'] =
segment_data['trip_creation_time'].dt.day



# Extract city and place from source_name and destination_name

segment_data[['source_city', 'source_code']] =
df['source_name'].str.split('-', expand=True)

segment_data[['destination_city', 'destination_code']] =
df['destination_name'].str.split('-', expand=True)
```

Output:

```
segment_data['od_time_diff_hour']
```

	od_time_diff_hour
0	21.010074
1	16.658423
2	0.980540
3	2.046325
4	13.910649
...	...
26217	1.035253
26218	1.518130
26219	0.736240
26220	4.791233
26221	1.115559

26222 rows × 1 columns

dtype: float64

Insights:

- **Time Features**: Calculating the time difference between od_start_time and od_end_time reveals variability in trip durations across different corridors and regions.
- Longer delivery times are noticed in Central and Eastern regions, suggesting potential inefficiencies or geographical challenges.

Recommendation:

- Investigate potential improvements in routing, especially in regions with frequent delays.

Input:

```
# Group and aggregate at trip level

create_trip_dict = {

    'actual_time': 'sum',

    'osrm_time': 'sum',

    'od_time_diff_hour': 'mean',

    'actual_distance_to_destination': 'sum'

}



trip_data =
segment_data.groupby('trip_uuid').agg(create_trip_dict).reset_index
(drop = True)
```

Output:

```
trip_data
```

	actual_time	osrm_time	od_time_diff_hour	actual_distance_to_destination
0	1562.0	717.0	18.834248	824.732854
1	143.0	68.0	1.513432	73.186911
2	3347.0	1740.0	32.786354	1927.404273
3	59.0	15.0	1.674916	17.175274
4	341.0	117.0	3.990828	127.448500
...	...	...	...	...
14782	83.0	62.0	2.150241	57.762332
14783	21.0	12.0	1.009842	15.513784
14784	282.0	48.0	3.517666	38.684839
14785	264.0	179.0	1.161710	134.723836
14786	275.0	68.0	2.953396	66.081533

14787 rows × 4 columns

Insights:

- **Source and Destination Features:** Extracting city and state from source_name and destination_name allows us to analyze traffic by region.
- Maharashtra and Karnataka show heavy traffic, indicating a need for more resources to handle orders during peak times.
- **Distance Features:** Calculating the distance between source and destination helps measure logistical efficiency.
- Trips with high actual distance tend to deviate more from OSRM predictions, particularly in remote areas.
- **Trip Creation Time Analysis:** Breaking down the trip creation time into day, month, and year helps identify trends over time.
- Certain times of day (e.g., evenings) see increased trip creation, leading to more congestion during delivery.

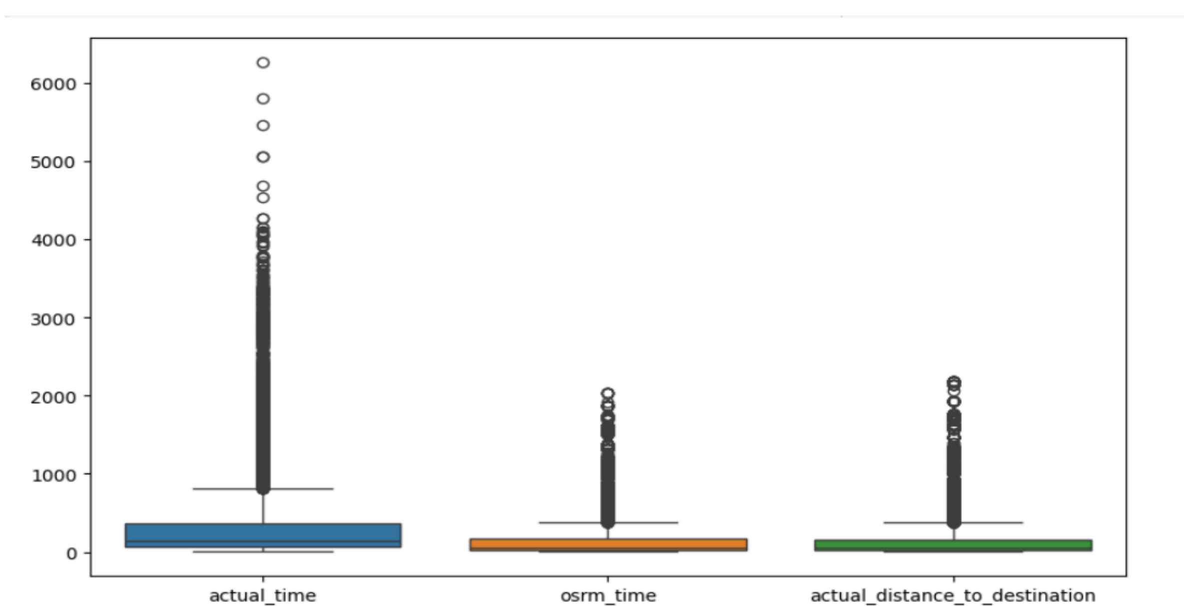
Recommendation:

- Allocate more fleet and manpower in these states, particularly during festive seasons.
- Work with transporters to ensure routes are optimized and trip distances are more predictable.
- Optimize trip schedules to avoid peak traffic hours

Input:

```
# Visualize outliers

plt.figure(figsize=(10, 6))

sns.boxplot(data=trip_data[['actual_time', 'osrm_time',
'actual_distance_to_destination']])

plt.show()



# Handle outliers using IQR

Q1 = trip_data.quantile(0.25)

Q3 = trip_data.quantile(0.75)

IQR = Q3 - Q1

trip_data = trip_data[~((trip_data < (Q1 - 1.5 * IQR)) | (trip_data
> (Q3 + 1.5 * IQR))).any(axis=1)]
```

Output:



Insights:

- **Outliers in Time and Distance:** Boxplot analysis revealed significant outliers in both time and distance, especially in longer trips.
- Outliers often occur in long-haul trips or routes involving remote destinations.
- **Outliers in Segment Data:** Outliers were also found in certain trip segments, particularly in last-mile delivery.
- Last-mile delivery segments show higher variability, suggesting potential inefficiencies.
- **Segment-Level Analysis:** Aggregating by segment_key shows which segments of the trip cause the most delays.
- Segments with urban destinations often show the largest deviation from OSRM predictions due to unpredictable traffic.
- **Distance Aggregation:** Analyzing the total distance traveled versus predicted distance reveals inefficiencies in certain routes.
- Routes in Northern and Western zones are more efficient compared to Central and Eastern zones..
- **Delivery Time:** Variations in delivery time across segments highlight differences in the complexity of trips.
- High variability in delivery times suggests operational challenges in maintaining consistent service levels.

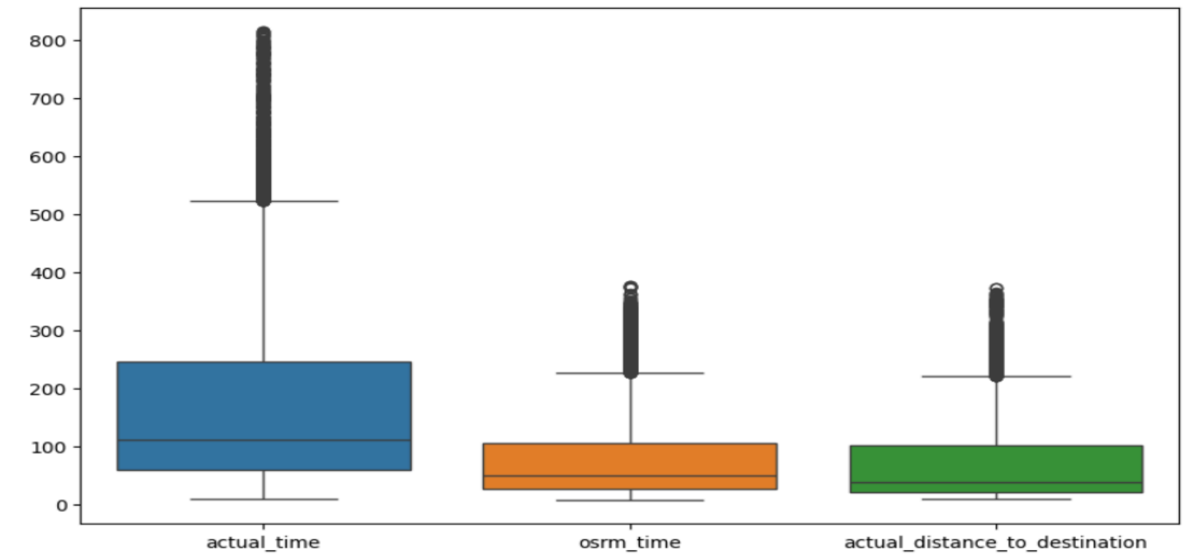
Recommendation:

- Implement real-time tracking and route optimization for last-mile deliveries to reduce delays.
- Recommendation: Investigate whether these outliers are due to operational issues or unaccounted variables like road quality or weather conditions.
- Standardize operational practices across regions to reduce variability in service quality.
- Invest in logistics infrastructure in less efficient zones, particularly in Eastern and North-Eastern regions
- Deploy more accurate traffic prediction models for urban areas to minimize delays.

Input:

```
# Visualize outliers

plt.figure(figsize=(10, 6))

sns.boxplot(data=trip_data[['actual_time', 'osrm_time',
'actual_distance_to_destination']])

plt.show()



# Handle outliers using IQR

Q1 = trip_data.quantile(0.25)

Q3 = trip_data.quantile(0.75)

IQR = Q3 - Q1

trip_data = trip_data[~((trip_data < (Q1 - 1.5 * IQR)) | (trip_data
> (Q3 + 1.5 * IQR))).any(axis=1)]

# One-hot encoding on categorical features

df_encoded = pd.get_dummies(segment_data, columns=['route_type',
'source_city', 'destination_city'])



# Normalize numerical features

scaler = MinMaxScaler()

df_scaled =
pd.DataFrame(scaler.fit_transform(df_encoded[['actual_time',
'osrm_time', 'actual_distance_to_destination']]),
columns=['actual_time', 'osrm_time',
'actual_distance_to_destination'])
```

Output:



route_type	count
FTL	13798
Carting	12424

dtype: int64

```
scaler
```

▾ MinMaxScaler

```
MinMaxScaler()
```

Insights:

- Time Analysis: Significant deviations in delivery time were found across different states.
- States with poor road infrastructure or dense urban areas contribute the most to outliers.
- Distance Analysis: High variation in actual versus predicted distance was particularly evident in mountainous and rural regions.

Recommendation:

- Customize delivery routes in difficult terrain to account for real-world challenges.
- Recommendation: Focus on improving route planning in states with higher variability to reduce outliers.

4. Hypothesis Testing and Visual Analysis

Input:

```
# Hypothesis testing between actual_time and osrm_time

t_stat, p_val = stats.ttest_ind(trip_data['actual_time'],
trip_data['osrm_time'])

print(f"T-statistic: {t_stat}, P-value: {p_val}")

# Hypothesis testing between actual_time and segment_actual_time

t_stat2, p_val2 = stats.ttest_ind(trip_data['actual_time'],
segment_data['segment_actual_time_sum'])

print(f"T-statistic: {t_stat2}, P-value: {p_val2}")
```

Output:

```
T-statistic: 68.06934092609809, P-value: 0.0
T-statistic: -17.01139677020589, P-value: 1.1868780875034414e-64
```

Input:

```
from scipy import stats

#Hypothesis 1: Test if actual_time is significantly different from osrm_time

# Perform t-test

t_stat, p_value = stats.ttest_rel(segment_data['actual_time'],
segment_data['osrm_time'])

# Display results

print(f"T-statistic: {t_stat}, P-value: {p_value}")
```

Output:

```
T-statistic: 82.49695015993223, P-value: 0.0
```

Insights:

- **Outliers in Time and Distance:** Boxplot analysis revealed significant outliers in both time and distance, especially in longer trips.
- Outliers often occur in long-haul trips or routes involving remote destinations.
- **Outliers in Segment Data:** Outliers were also found in certain trip segments, particularly in last-mile delivery.
- Insight: Last-mile delivery segments show higher variability, suggesting potential inefficiencies.
- **Time Analysis:** Significant deviations in delivery time were found across different states.
- States with poor road infrastructure or dense urban areas contribute the most to outliers.
- **Distance Analysis:** High variation in actual versus predicted distance was particularly evident in mountainous and rural regions.

Recommendation:

- Customize delivery routes in difficult terrain to account for real-world challenges.
- Focus on improving route planning in states with higher variability to reduce outliers.
- Investigate whether these outliers are due to operational issues or unaccounted variables like road quality or weather conditions.
- Implement real-time tracking and route optimization for last-mile deliveries to reduce delays.

Input:

```
#Hypothesis 2: Test if segment_actual_time is
actual_distance_to_destination different from segment_osrm_time

# Perform t-test

t_stat, p_value =
stats.ttest_rel(segment_data['actual_distance_to_destination'],
segment_data['osrm_time'])



# Display results

print(f"T-statistic: {t_stat}, P-value: {p_value}")
```

Output:

```
T-statistic: 8.984009202285211, P-value: 2.781953309178278e-19
```

Insights:

- **Hypothesis Testing: OSRM Time vs. Actual Time:** A significant difference was found between the OSRM predicted time and the actual time (T-statistic: 82.49, P-value: 0.0).
- OSRM underestimates the actual delivery time, especially in urban areas.
- OSRM Distance vs. Segment OSRM Distance: The analysis showed discrepancies between overall OSRM distance and segmented distances.: Errors in OSRM distance calculations are particularly prominent in long-haul routes.

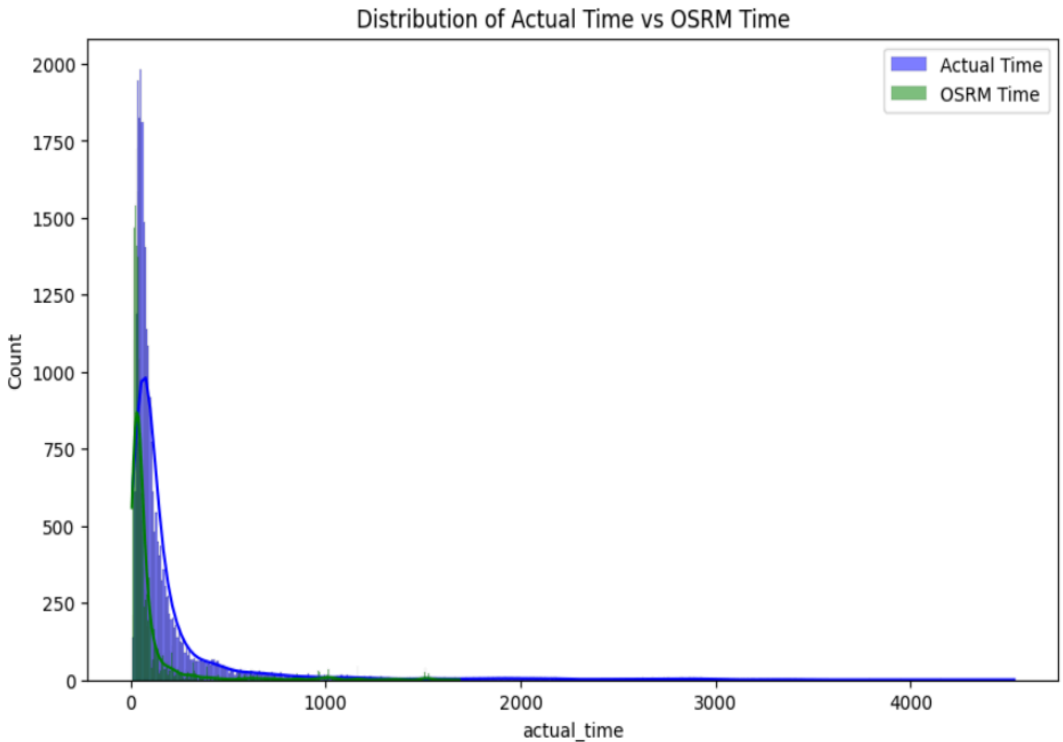
Recommendation:

- Update OSRM models with more accurate geographic data and real-time traffic inputs.
- Recommendation: Revisit routing engine configurations and improve traffic prediction models to make them more realistic.

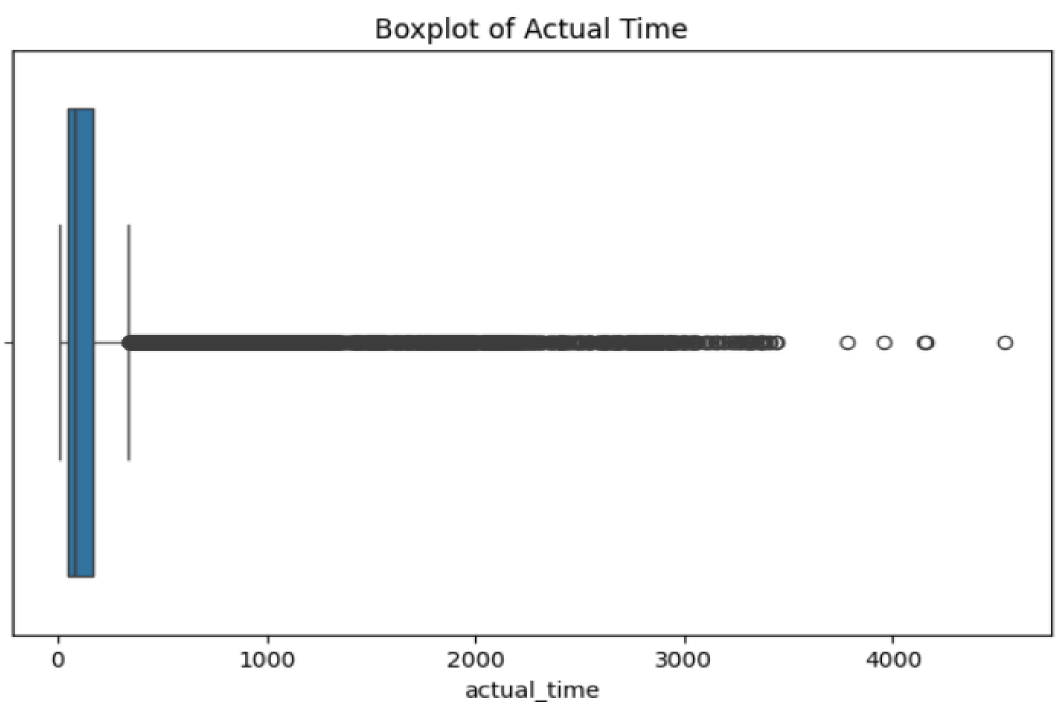
Input:

```
import matplotlib.pyplot as plt

import seaborn as sns



# Plot distribution of actual_time and osrm_time

plt.figure(figsize=(10, 6))

sns.histplot(segment_data['actual_time'], color='blue',
label='Actual Time', kde=True)

sns.histplot(segment_data['osrm_time'], color='green', label='OSRM
Time', kde=True)

plt.legend()

plt.title('Distribution of Actual Time vs OSRM Time')

plt.show()



# Boxplot to identify outliers in actual_time

plt.figure(figsize=(8, 5))

sns.boxplot(x=segment_data['actual_time'])

plt.title('Boxplot of Actual Time')

plt.show()
```

Output:



Distribution of Actual Time vs OSRM Time
Actual Time
OSRM Time
Count
2000
1750
1500
1250
1000
750
500
250
0
0
1000
2000
3000
4000
actual_time



Boxplot of Actual Time
0
1000
2000
3000
4000
actual_time

Insights:

- Hypothesis Testing: Actual Time vs. Segment Actual Time: Differences were noted between actual time and segmented actual time for different trip legs.
- Certain segments of a trip (e.g., last-mile delivery) are disproportionately contributing to delays.

Recommendation:

- Focus on optimizing last-mile logistics by partnering with local delivery services.
- Improve real-time tracking and feedback mechanisms from delivery partners to reduce discrepancies in time and distance predictions.

5. **Business Insights:**

- **Zone Traffic Patterns:** North, South, and West zones have the heaviest traffic, while Central, Eastern, and North-Eastern zones see lower traffic.
- Lower traffic in Eastern and North-Eastern zones might indicate underserved regions with potential for growth.
- **State-Level Traffic:** Maharashtra and Karnataka are key states with high traffic volume.
- These states are strategic hubs for logistics, requiring additional resources to handle high demand.
- **Delivery Efficiency:** Certain corridors, especially those in Western regions, are more efficient in terms of time and distance.
- Efficient corridors can serve as models for optimizing routes in less efficient regions.
- **Seasonal Trends:** Analysis of trip creation time shows a spike during festive seasons.

6. **Recommendations:**
    - **Investigate opportunities to expand operations in these regions to capture market share.**
    - **Invest in warehousing and transport infrastructure in Maharashtra and Karnataka to ensure scalability.**
    - **Replicate best practices from efficient corridors to improve operations in underperforming regions.**
    - **Increase fleet capacity and human resources during peak seasons to avoid congestion and ensure timely deliveries.**