Delhivery - Feature Engineering

Delhivery is the largest and fastest-growing fully integrated player in India by revenue in Fiscal 2021. They aim to build the operating system for commerce, through a combination of world-class infrastructure, logistics operations of the highest quality, and cutting-edge engineering and technology capabilities.

The Data team builds intelligence and capabilities using this data that helps them to widen the gap between the quality, efficiency, and profitability of their business versus their competitors.

The company wants to understand and process the data coming out of data engineering pipelines:

• Clean, sanitize and manipulate data to get useful features out of raw fields

• Make sense out of the raw data and help the data science team to build forecasting models on it

```
#importing necessary packages for the analyses
import numpy as np
import pandas as pd
from scipy import stats
import matplotlib.pyplot as plt
import seaborn as sns
```

```
#downloading the .csv file
df = pd.read_csv("https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/551/original/delhivery_data.csv?1642751181")
```

Data checking

```
#initial data check
df.head()
```

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid
0	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320

```
df.tail()
```

	data	trip_creation_time	route_schedule_uuid	route_type	trip_u
144862	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	153746066843555
144863	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	153746066843555
144864	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	153746066843555
144865	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	153746066843555
144866	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	153746066843555

5 rows × 24 columns

```
# no of rows amd columns in dataset
print(f"# rows: {df.shape[0]} \n# columns: {df.shape[1]}")
```

```
# rows: 144867
# columns: 24
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 144867 entries, 0 to 144866
Data columns (total 24 columns):
 #   Column                          Non-Null Count   Dtype  
---  ------                          --------------   -----  
 0   data                            144867 non-null  object 
 1   trip_creation_time              144867 non-null  object 
 2   route_schedule_uuid             144867 non-null  object 
 3   route_type                      144867 non-null  object 
 4   trip_uuid                       144867 non-null  object 
 5   source_center                   144867 non-null  object 
 6   source_name                     144574 non-null  object 
 7   destination_center              144867 non-null  object 
 8   destination_name                144606 non-null  object 
 9   od_start_time                   144867 non-null  object 
 10  od_end_time                     144867 non-null  object 
 11  start_scan_to_end_scan          144867 non-null  float64
 12  is_cutoff                       144867 non-null  bool   
 13  cutoff_factor                   144867 non-null  int64  
 14  cutoff_timestamp                144867 non-null  object 
 15  actual_distance_to_destination  144867 non-null  float64
 16  actual_time                     144867 non-null  float64
 17  osrm_time                       144867 non-null  float64
 18  osrm_distance                   144867 non-null  float64
 19  factor                          144867 non-null  float64
 20  segment_actual_time             144867 non-null  float64
 21  segment_osrm_time               144867 non-null  float64
 22  segment_osrm_distance           144867 non-null  float64
 23  segment_factor                  144867 non-null  float64
dtypes: bool(1), float64(10), int64(1), object(12)
memory usage: 25.6+ MB
```

```
df.describe()
```

	start_scan_to_end_scan	cutoff_factor	actual_distance_to_destination	actua
count	144867.000000	144867.000000	144867.000000	144867.
mean	961.262986	232.926567	234.073372	416.

```
df.iloc[:, 1:].describe(include='all')
```

	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	so
count	144867	144867	144867	144867	
unique	14817	1504	2	14817	
top	2018-09-28 05:23:15.359220	thanos::sroute:4029a8a2-6c74-4b7e-a6d8-f9e069f...	FTL	trip-153811219535896559	IN
freq	101	1812	99660	101	
mean	NaN	NaN	NaN	NaN	
std	NaN	NaN	NaN	NaN	
min	NaN	NaN	NaN	NaN	
25%	NaN	NaN	NaN	NaN	
50%	NaN	NaN	NaN	NaN	
75%	NaN	NaN	NaN	NaN	
max	NaN	NaN	NaN	NaN	

11 rows × 23 columns

```
# detecting missing values in the dataset
df.isnull().sum()
```

```
data                              0
trip_creation_time                0
route_schedule_uuid               0
route_type                        0
trip_uuid                         0
source_center                     0
source_name                     293
destination_center                0
destination_name                261
od_start_time                     0
od_end_time                       0
start_scan_to_end_scan            0
is_cutoff                         0
cutoff_factor                     0
cutoff_timestamp                  0
actual_distance_to_destination    0
actual_time                       0
osrm_time                         0
osrm_distance                     0
factor                            0
segment_actual_time               0
segment_osrm_time                 0
segment_osrm_distance             0
segment_factor                    0
dtype: int64
```

There are some missing values in the source_name and destination_name column. However, these don't even constitute 1% of the total data. Hence we can ignore the same. Thus remove or drop null values from the data.

```
#removing null values
df = df.dropna(how='any')
df = df.reset_index(drop=True)
```

```
#rechecking data for null values
df.isnull().sum()
```

```
data                            0
trip_creation_time              0
route_schedule_uuid             0
route_type                      0
trip_uuid                       0
source_center                   0
source_name                     0
destination_center              0
destination_name                0
```

```
od_start_time                    0
od_end_time                      0
start_scan_to_end_scan           0
is_cutoff                        0
cutoff_factor                    0
cutoff_timestamp                 0
actual_distance_to_destination   0
actual_time                      0
osrm_time                        0
osrm_distance                    0
factor                           0
segment_actual_time              0
segment_osrm_time                0
segment_osrm_distance            0
segment_factor                   0
dtype: int64
```

```
#converting datetime to pandas datetime

df["od_start_time"] = pd.to_datetime(df["od_start_time"])
df["od_end_time"] = pd.to_datetime(df["od_end_time"])
```

```
#using max columns to display all the columns
pd.set_option('display.max_columns', None)
df.head()
```

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid
0	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320
1	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320
2	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320
3	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320
4	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320

```
#column 9 and 10 shows dtype as datetime64 after change
df.info()
```

```
<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  object
 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](2), float64(10), int64(1), object(10)
memory usage: 25.5+ MB
```

```
# grouping by sub-journey in the trip
df['segment_key'] = df['trip_uuid'] + df['source_center'] + df['destination_center']


segment_cols = ['segment_actual_time', 'segment_osrm_distance', 'segment_osrm_time']


#after the loop completes, df will have new columns for each original column specified in segment_cols, where the values represent the cu
for col in segment_cols:
    df[col + '_sum'] = df.groupby('segment_key')[col].cumsum()

df[[col + '_sum' for col in segment_cols]]
```

	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
...	...	...	...
144311	92.0	65.3487	94.0
144312	118.0	82.7212	115.0
144313	138.0	103.4265	149.0
144314	155.0	122.3150	176.0
144315	423.0	131.1238	185.0

144316 rows × 3 columns

```
#aggregating at sub-journey level

create_segment_dict = {

    'data' : 'first',
    'trip_creation_time' : 'first',
    'route_schedule_uuid' : 'first',
    'route_type' : 'first',
    'trip_uuid' : 'first',
    'source_center' : 'first',
    'source_name' : 'first',

    'destination_center' : 'last',
    'destination_name' : 'last',

    'od_start_time' : 'first',
    'od_end_time' : 'first',
    'start_scan_to_end_scan' : 'first',

    'actual_distance_to_destination' : 'last',
    'actual_time' : 'last',

    'osrm_time' : 'last',
    'osrm_distance' : 'last',

    'segment_actual_time_sum' : 'last',
    'segment_osrm_distance_sum' : 'last',
    'segment_osrm_time_sum' : 'last',

}


# grouping mini-trips, sorting by time
segment = df.groupby('segment_key').agg(create_segment_dict).reset_index()
segment = segment.sort_values(by=['segment_key','od_end_time'], ascending=True).reset_index()
segment
```

	index	segment_key	data	trip_creation_tir
0	0	trip-153671041653548748IND209304AAAIND000000ACB	training	2018-09-1 00:00:16.53574
1	1	trip-153671041653548748IND462022AAAIND209304AAA	training	2018-09-1 00:00:16.53574
2	2	trip-153671042288605164IND561203AABIND562101AAA	training	2018-09-1 00:00:22.88643
3	3	trip-153671042288605164IND572101AAAIND561203AAB	training	2018-09-1 00:00:22.88643
4	4	trip-153671043369099517IND000000ACBIND160002AAC	training	2018-09-1 00:00:33.69128
...	...	...	...	
26217	26217	trip-153861115439069069IND628204AAAIND627657AAA	test	2018-10-0 23:59:14.39095
26218	26218	trip-153861115439069069IND628613AAAIND627005AAA	test	2018-10-0 23:59:14.39095
26219	26219	trip-153861115439069069IND628801AAAIND628204AAA	test	2018-10-0 23:59:14.39095

```
#example of filtering segment by trip_uuid
segment[segment['trip_uuid']=='trip-153671041653548748']
```

	index	segment_key	data	trip_creation_time	
0	0	trip-153671041653548748IND209304AAAIND000000ACB	training	2018-09-12 00:00:16.535741	tl
1	1	trip-153671041653548748IND462022AAAIND209304AAA	training	2018-09-12 00:00:16.535741	tl

```
segment.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26222 entries, 0 to 26221
Data columns (total 21 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   index                           26222 non-null  int64
 1   segment_key                     26222 non-null  object
 2   data                            26222 non-null  object
 3   trip_creation_time              26222 non-null  object
 4   route_schedule_uuid             26222 non-null  object
 5   route_type                      26222 non-null  object
 6   trip_uuid                       26222 non-null  object
 7   source_center                   26222 non-null  object
 8   source_name                     26222 non-null  object
 9   destination_center              26222 non-null  object
 10  destination_name                26222 non-null  object
 11  od_start_time                   26222 non-null  datetime64[ns]
 12  od_end_time                     26222 non-null  datetime64[ns]
 13  start_scan_to_end_scan          26222 non-null  float64
 14  actual_distance_to_destination  26222 non-null  float64
 15  actual_time                     26222 non-null  float64
 16  osrm_time                       26222 non-null  float64
 17  osrm_distance                   26222 non-null  float64
 18  segment_actual_time_sum         26222 non-null  float64
 19  segment_osrm_distance_sum       26222 non-null  float64
 20  segment_osrm_time_sum           26222 non-null  float64
dtypes: datetime64[ns](2), float64(8), int64(1), object(10)
memory usage: 4.2+ MB
```

```
#calculating time taken between od_start_time and od_end_time and keep it as a feature
segment['time_diff_hours'] = (segment['od_end_time'] - segment['od_start_time']).dt.total_seconds()/60
segment['time_diff_hours']
```

```
0        1260.604421
1         999.505379
2          58.832388
```

```
3        122.779486
4        834.638929
            ...
26217     62.115193
26218     91.087797
26219     44.174403
26220    287.474007
26221     66.933565
Name: time_diff_hours, Length: 26222, dtype: float64
```

```
segment
```

	index	segment_key	data	trip_creation_tir
0	0	trip-153671041653548748IND209304AAAIND000000ACB	training	2018-09-' 00:00:16.5357
1	1	trip-153671041653548748IND462022AAAIND209304AAA	training	2018-09-' 00:00:16.5357
2	2	trip-153671042288605164IND561203AABIND562101AAA	training	2018-09-' 00:00:22.8864
3	3	trip-153671042288605164IND572101AAAIND561203AAB	training	2018-09-' 00:00:22.8864
4	4	trip-153671043369099517IND000000ACBIND160002AAC	training	2018-09-' 00:00:33.6912
...	...	...	...	
26217	26217	trip-153861115439069069IND628204AAAIND627657AAA	test	2018-10-(23:59:14.3909
26218	26218	trip-153861115439069069IND628613AAAIND627005AAA	test	2018-10-(23:59:14.3909
26219	26219	trip-153861115439069069IND628801AAAIND628204AAA	test	2018-10-(23:59:14.3909
26220	26220	trip-153861118270144424IND583119AAAIND583101AAA	test	2018-10-(23:59:42.7016
26221	26221	trip-153861118270144424IND583201AAAIND583119AAA	test	2018-10-(23:59:42.7016

26222 rows × 22 columns

```
create_trip_dict = {

    'data' : 'first',
    'trip_creation_time' : 'first',
    'route_schedule_uuid' : 'first',
    'route_type' : 'first',
    'trip_uuid' : 'first',

    'source_center' : 'first',
    'source_name' : 'first',

    'destination_center' : 'last',
    'destination_name' : 'last',

    'start_scan_to_end_scan' : 'sum',
    'time_diff_hours' : 'sum',

    'actual_distance_to_destination' : 'sum',
    'actual_time' : 'sum',
    'osrm_time' : 'sum',
    'osrm_distance' : 'sum',

    'segment_actual_time_sum' : 'sum',
    'segment_osrm_distance_sum' : 'sum',
    'segment_osrm_time_sum' : 'sum',

}
```

```
trip = segment.groupby("trip_uuid").agg(create_trip_dict).reset_index(drop=True)
trip
```

	data	trip_creation_time	route_schedule_uuid	route_type	trip_u
0	training	2018-09-12 00:00:16.535741	thanos::sroute:d7c989ba-a29b-4a0b-b2f4-288cdc6...	FTL	153671041653548
1	training	2018-09-12 00:00:22.886430	thanos::sroute:3a1b0ab2-bb0b-4c53-8c59-eb2a2c0...	Carting	153671042288605
2	training	2018-09-12 00:00:33.691250	thanos::sroute:de5e208e-7641-45e6-8100-4d9fb1e...	FTL	153671043369099
3	training	2018-09-12 00:01:00.113710	thanos::sroute:f0176492-a679-4597-8332-bbd1c7f...	Carting	153671046011330
4	training	2018-09-12 00:02:09.740725	thanos::sroute:d9f07b12-65e0-4f3b-bec8-df06134...	FTL	153671052974046
...	...	...	...	...	
14782	test	2018-10-03 23:55:56.258533	thanos::sroute:8a120994-f577-4491-9e4b-b7e4a14...	Carting	153861095625827
14783	test	2018-10-03 23:57:23.863155	thanos::sroute:b30e1ec3-3bfa-4bd2-a7fb-3b75769...	Carting	153861104386292
14784	test	2018-10-03 23:57:44.429324	thanos::sroute:5609c268-e436-4e0a-8180-3db4a74...	Carting	153861106442901
14785	test	2018-10-03 23:59:14.390954	thanos::sroute:c5f2ba2c-8486-4940-8af6-d1d2a6a...	Carting	153861115439069
14786	test	2018-10-03 23:59:42.701692	thanos::sroute:412fea14-6d1f-4222-8a5f-a517042...	FTL	153861118270144

14787 rows × 18 columns

```
trip[['actual_time','segment_actual_time_sum']]
```

	actual_time	segment_actual_time_sum
0	1562.0	1548.0
1	143.0	141.0
2	3347.0	3308.0
3	59.0	59.0
4	341.0	340.0
...	...	...
14782	83.0	82.0
14783	21.0	21.0
14784	282.0	281.0
14785	264.0	258.0
14786	275.0	274.0

14787 rows × 2 columns

```
#example of filtering by trip_uuid
trip[trip['trip_uuid'].isin(['trip-153671042288605164','trip-153671052974046625'])]
```

data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid

```
trip[['actual_distance_to_destination','osrm_distance']]
```

	actual_distance_to_destination	osrm_distance
0	824.732854	991.3523
1	73.186911	85.1110
2	1927.404273	2354.0665
3	17.175274	19.6800
4	127.448500	146.7918
...	...	...
14782	57.762332	73.4630
14783	15.513784	16.0882
14784	38.684839	58.9037
14785	134.723836	171.1103
14786	66.081533	80.5787

14787 rows × 2 columns

Hypothesis testing

```
trip.head()
```

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid
0	training	2018-09-12 00:00:16.535741	thanos::sroute:d7c989ba-a29b-4a0b-b2f4-288cdc6...	FTL	trip-153671041653548748
1	training	2018-09-12 00:00:22.886430	thanos::sroute:3a1b0ab2-bb0b-4c53-8c59-eb2a2c0...	Carting	trip-153671042288605164
2	training	2018-09-12 00:00:33.691250	thanos::sroute:de5e208e-7641-45e6-8100-4d9fb1e...	FTL	trip-153671043369099517
3	training	2018-09-12 00:01:00.113710	thanos::sroute:f0176492-a679-4597-8332-bbd1c7f...	Carting	trip-153671046011330457
4	training	2018-09-12 00:02:09.740725	thanos::sroute:d9f07b12-65e0-4f3b-bec8-df06134...	FTL	trip-153671052974046625

```
trip['destination_name'] = trip['destination_name'].str.lower() # lowering all columns
trip['source_name'] = trip['source_name']
```

```
def place2state(x):
    # transform "gurgaon_bilaspur_hb (haryana)" into "haryana"
    state = x.split('(')[1]

    return state[:-1] #removing ')' from ending

def place2city(x):
    #we remove state in this step
    city = x.split(' (')[0]

    city = city.split('_')[0]

    # now dealing with edge cases

    if city == 'pnq vadgaon sheri dpc': return 'vadgaonsheri'

    # ['PNQ Pashan DPC', 'Bhopal MP Nagar', 'HBR Layout PC',
    #  'PNQ Rahatani DPC', 'Pune Balaji Nagar', 'Mumbai Antop Hill']

    if city in ['pnq pashan dpc','pnq rahatani dpc', 'pune balaji nagar']:
        return 'pune'

    if city == 'hbr layout pc' :
        return 'bengaluru'
```

```
    if city == 'bhopal mp nagar':
        return 'bhopal'
    if city == 'mumbai antop hill':
        return 'mumbai'

    return city

def place2city_place(x):

    # we will remove state
    x = x.split('(')[0]

    len_ = len(x.split('_'))

    if len_ >= 3:
        return x.split('_')[1]

    # small cities have same city and place name
    if len_ == 2:
        return x.split('_')[0]

    # now we need to deal with edge cases or improper name convention

    # if len(x.split('_')) == 2:

    return x.split(' ')[0]

def place2code(x):
    # we will remove state
    x = x.split('(')[0]

    if len(x.split('_')) >= 3:
        return x.split('_')[-1]

    return 'none'
```

```
trip['destination_state'] = trip['destination_name'].apply(lambda x: place2state(x))
trip['destination_city']  = trip['destination_name'].apply(lambda x: place2city(x))
trip['destination_place'] = trip['destination_name'].apply(lambda x: place2city_place(x))
trip['destination_code']  = trip['destination_name'].apply(lambda x: place2code(x))
```

```
trip['destination_name'].head()
```

```
0    kanpur_central_h_6 (uttar pradesh)
1     doddablpur_chikadpp_d (karnataka)
2         gurgaon_bilaspur_hb (haryana)
3        mumbai_mirard_ip (maharashtra)
4         sandur_wrdn1dpp_d (karnataka)
Name: destination_name, dtype: object
```

```
trip[['destination_state','destination_city','destination_place','destination_code']]
```

	destination_state	destination_city	destination_place	destination_code
0	uttar pradesh	kanpur	central	6
1	karnataka	doddablpur	chikadpp	d
2	haryana	gurgaon	bilaspur	hb
3	maharashtra	mumbai	mirard	ip
4	karnataka	sandur	wrdn1dpp	d
...	...	...	...	...
14782	punjab	chandigarh	mehmdpur	h
14783	haryana	faridabad	blbgarh	dc
14784	uttar pradesh	kanpur	govndngr	dc
14785	tamil nadu	tirchchndr	shnmgprm	d
14786	karnataka	sandur	wrdn1dpp	d

14787 rows × 4 columns

```
trip['trip_creation_time'] = pd.to_datetime(trip['trip_creation_time'])

trip['trip_year'] = trip['trip_creation_time'].dt.year
trip['trip_month'] = trip['trip_creation_time'].dt.month
trip['trip_day'] = trip['trip_creation_time'].dt.day
trip['trip_hour'] = trip['trip_creation_time'].dt.hour
```

```
trip['trip_week'] = trip['trip_creation_time'].dt.isocalendar().week
trip['day_of_week'] = trip['trip_creation_time'].dt.dayofweek
```

```
trip[['trip_year','trip_month','trip_day','trip_hour','trip_week','day_of_week']]
```

	trip_year	trip_month	trip_day	trip_hour	trip_week	day_of_week
0	2018	9	12	0	37	2
1	2018	9	12	0	37	2
2	2018	9	12	0	37	2
3	2018	9	12	0	37	2
4	2018	9	12	0	37	2
...	...	...	...	...	...	...
14782	2018	10	3	23	40	2
14783	2018	10	3	23	40	2
14784	2018	10	3	23	40	2
14785	2018	10	3	23	40	2
14786	2018	10	3	23	40	2

14787 rows × 6 columns

```
trip.head()
```

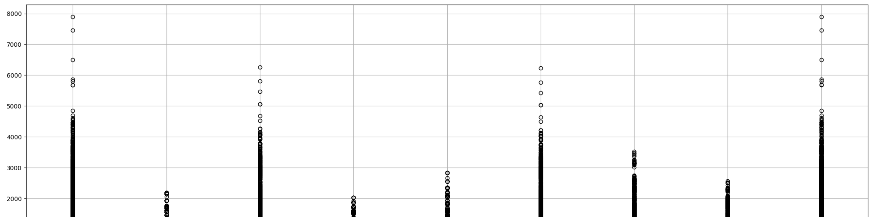
	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid
0	training	2018-09-12 00:00:16.535741	thanos::sroute:d7c989ba-a29b-4a0b-b2f4-288cdc6...	FTL	trip-153671041653548748
1	training	2018-09-12 00:00:22.886430	thanos::sroute:3a1b0ab2-bb0b-4c53-8c59-eb2a2c0...	Carting	trip-153671042288605164
2	training	2018-09-12 00:00:33.691250	thanos::sroute:de5e208e-7641-45e6-8100-4d9fb1e...	FTL	trip-153671043369099517
3	training	2018-09-12 00:01:00.113710	thanos::sroute:f0176492-a679-4597-8332-bbd1c7f...	Carting	trip-153671046011330457
4	training	2018-09-12 00:02:09.740725	thanos::sroute:d9f07b12-65e0-4f3b-bec8-df06134...	FTL	trip-153671052974046625

```
num_cols = ['start_scan_to_end_scan','actual_distance_to_destination','actual_time','osrm_time',
            'osrm_distance','segment_actual_time_sum','segment_osrm_distance_sum',
           'segment_osrm_time_sum', 'time_diff_hours']
```

Find outliers in numericals variable and visualize it using visual analysis

```
trip[num_cols].boxplot(rot=25, figsize=(25,8))
```

```
<Axes: >
```



Handle the outliers using IQR method

```
Q1 = trip[num_cols].quantile(0.25)
Q3 = trip[num_cols].quantile(0.75)

IQR = Q3 - Q1
IQR
```

```
start_scan_to_end_scan             483.000000
actual_distance_to_destination     140.814159
actual_time                        300.000000
osrm_time                          139.000000
osrm_distance                      175.887300
segment_actual_time_sum            298.000000
segment_osrm_distance_sum          183.981750
segment_osrm_time_sum              154.000000
time_diff_hours                    483.839201
dtype: float64
```

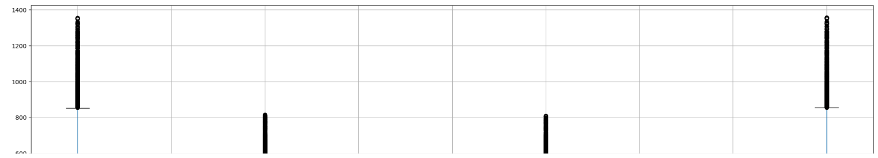
```
trip = trip[-((trip[num_cols] < (Q1 - 1.5 * IQR)) | (trip[num_cols] > (Q3 + 1.5 * IQR))).any(axis=1)]
trip = trip.reset_index(drop=True)
```

```
trip.head()
```

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid
0	training	2018-09-12 00:00:22.886430	thanos::sroute:3a1b0ab2-bb0b-4c53-8c59-eb2a2c0...	Carting	trip-153671042288605164
1	training	2018-09-12 00:01:00.113710	thanos::sroute:f0176492-a679-4597-8332-bbd1c7f...	Carting	trip-153671046011330457
2	training	2018-09-12 00:02:09.740725	thanos::sroute:d9f07b12-65e0-4f3b-bec8-df06134...	FTL	trip-153671052974046625
3	training	2018-09-12 00:02:34.161600	thanos::sroute:9bf03170-d0a2-4a3f-aa4d-9aaab3d...	Carting	trip-153671055416136166
4	training	2018-09-12 00:04:22.011653	thanos::sroute:a97698cc-846e-41a7-916b-88b1741...	Carting	trip-153671066201138152

```
trip[num_cols].boxplot(rot=25, figsize=(25,8))
```

```
<Axes: >
```



Handling Categorical Variables

```
trip['route_type'].value_counts()
```

```
Carting    8812
FTL        3911
Name: route_type, dtype: int64
```

```
trip['route_type'] = trip['route_type'].map({'FTL':0, 'Carting':1})
```

Normalize/Standarize the numerical features using MinMaxScaler or StandardScaler

```
from sklearn.preprocessing import StandardScaler
```

```
scaler = StandardScaler()
scaler.fit(trip[num_cols])
```

```
▾ StandardScaler
StandardScaler()
```

```
trip[num_cols] = scaler.transform(trip[num_cols])
```

```
trip[num_cols]
```

	start_scan_to_end_scan	actual_distance_to_destination	actual_time	osrm_time
0	-0.548546	0.012060	-0.217856	-0.144341
1	-0.861602	-0.765152	-0.749015	-0.877085
2	1.552838	0.764988	1.034163	0.533102
3	-0.513328	-0.662169	-0.736369	-0.766482
4	-0.869428	-0.877197	-0.970332	-0.904736
...	...	...	...	..
12718	-0.247231	-0.201970	-0.597255	-0.227293
12719	-1.018130	-0.788207	-0.989302	-0.918561
12720	0.394533	-0.466688	0.661086	-0.420848
12721	0.104957	0.865940	0.547267	1.390274
12722	0.128436	-0.086534	0.616823	-0.144341

12723 rows × 9 columns

```
trip[num_cols].describe()
```

	start_scan_to_end_scan	actual_distance_to_destination	actual_time	osrm_time	osrm_distance	segment_actual_time_sum
count	1.272300e+04	1.272300e+04	1.272300e+04	1.272300e+04	1.272300e+04	1.272300e+04
mean	-1.619566e-17	-7.371818e-17	-8.041983e-17	4.467769e-17	3.797603e-17	-3.127438e-17
std	1.000039e+00	1.000039e+00	1.000039e+00	1.000039e+00	1.000039e+00	1.000039e+00
min	-1.162918e+00	-8.785574e-01	-1.065181e+00	-1.001514e+00	-9.229378e-01	-1.061764e+00
25%	-7.207269e-01	-7.065920e-01	-7.363685e-01	-7.111809e-01	-7.077649e-01	-7.371165e-01
50%	-3.411472e-01	-4.689012e-01	-4.012322e-01	-3.931975e-01	-4.836339e-01	-3.997380e-01
75%	4.023595e-01	4.073375e-01	4.650634e-01	4.224989e-01	4.419548e-01	4.596223e-01
max	4.049455e+00	4.178358e+00	4.031419e+00	4.113871e+00	4.150641e+00	4.037107e+00

Recommendation:

- Routing Engine Discrepancies:

The data indicates a noticeable variance between the output from the Open Source Routing Machine (OSRM) and the actual parameters. It is imperative to review the input information provided to the routing engine for trip planning. Additionally, a thorough examination of potential disparities with transporters is recommended to ensure the routing engine is configured for optimal results.

- Traffic Disparities Across Zones:

Analyzing the traffic patterns reveals significant order flow in the North, South, and West Zones, while the Central, Eastern, and North-Eastern zones exhibit comparatively lower activity. While this observation may need validation over a more extended period, it suggests an opportunity to explore and enhance our presence in these less-explored regions.

- Strategic Resource Allocation:

State-wise analysis highlights Maharashtra as a region with heavy order traffic, closely followed by Karnataka. This insight serves as a strategic indicator, emphasizing the need for prioritized resource planning and on-ground presence in these two states, particularly during festive seasons.