Feature Engineering - Delhivery

Objective of this notebook is:

1. To understand the patterns in the data.
2. How to Handle the categorical features.
3. How to deal with missing data.
4. Feature Engineering
5. Finding the most important features
6. Understanding the Normalization and standardisation of the data.

Load data and libraries

In [481]:
```
import numpy as np
import pandas as pd
from scipy import stats

import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings('ignore')
```

In [482]:
```
delhivery_df = pd.read_csv("https://d2beiqkhq929f0.cloudfront.net/public_assets/a
```

In [483]:
```
delhivery_df.shape
```

Out[483]:
```
(144867, 24)
```

In [484]:
```
delhivery_df.columns
```

Out[484]:
```
Index(['data', 'trip_creation_time', 'route_schedule_uuid', 'route_type',
       'trip_uuid', 'source_center', 'source_name', 'destination_center',
       'destination_name', 'od_start_time', 'od_end_time',
       'start_scan_to_end_scan', 'is_cutoff', 'cutoff_factor',
       'cutoff_timestamp', 'actual_distance_to_destination', 'actual_time',
       'osrm_time', 'osrm_distance', 'factor', 'segment_actual_time',
       'segment_osrm_time', 'segment_osrm_distance', 'segment_factor'],
      dtype='object')
```

In [485]:
```
delhivery_df.head()
```

Out[485]:

e	destination_center	destination_name	od_start_time	...	cutoff_timestamp	actual_distance_t
C t)	IND388620AAB	Khambhat_MotvdDPP_D (Gujarat)	2018-09-20 03:21:32.418600	...	2018-09-20 04:27:55	
C t)	IND388620AAB	Khambhat_MotvdDPP_D (Gujarat)	2018-09-20 03:21:32.418600	...	2018-09-20 04:17:55	
C t)	IND388620AAB	Khambhat_MotvdDPP_D (Gujarat)	2018-09-20 03:21:32.418600	...	2018-09-20 04:01:19.505586	
C t)	IND388620AAB	Khambhat_MotvdDPP_D (Gujarat)	2018-09-20 03:21:32.418600	...	2018-09-20 03:39:57	
C t)	IND388620AAB	Khambhat_MotvdDPP_D (Gujarat)	2018-09-20 03:21:32.418600	...	2018-09-20 03:33:55	

In [486]:
```
delhivery_df.dtypes
```

Out[486]:
```
data                            object
trip_creation_time              object
route_schedule_uuid             object
route_type                      object
trip_uuid                       object
source_center                   object
source_name                     object
destination_center              object
destination_name                object
od_start_time                   object
od_end_time                     object
start_scan_to_end_scan         float64
is_cutoff                         bool
cutoff_factor                    int64
cutoff_timestamp                object
actual_distance_to_destination  float64
actual_time                     float64
osrm_time                       float64
osrm_distance                   float64
factor                          float64
segment_actual_time             float64
segment_osrm_time               float64
segment_osrm_distance           float64
segment_factor                  float64
dtype: object
```

In [487]:
```
delhivery_df.nunique()
```

Out[487]:
```
data                                2
trip_creation_time              14817
route_schedule_uuid              1504
route_type                          2
trip_uuid                       14817
source_center                    1508
source_name                      1498
destination_center               1481
destination_name                 1468
od_start_time                   26369
od_end_time                     26369
start_scan_to_end_scan           1915
is_cutoff                           2
cutoff_factor                     501
cutoff_timestamp                93180
actual_distance_to_destination  144515
actual_time                      3182
osrm_time                        1531
osrm_distance                  138046
factor                           45641
segment_actual_time                747
segment_osrm_time                  214
segment_osrm_distance           113799
segment_factor                    5675
dtype: int64
```

Basic Data Exploration

In [488]:
```
df_trip = delhivery_df [delhivery_df['trip_uuid'] == 'trip-153741093647649320']
df_trip
```

Out[488]:

name	destination_center	destination_name	od_start_time	...	cutoff_timestamp	actual_distan
ır_DC ijarat)	IND388620AAB	Khambhat_MotvdDPP_D (Gujarat)	2018-09-20 03:21:32.418600	...	2018-09-20 04:27:55	
ır_DC ijarat)	IND388620AAB	Khambhat_MotvdDPP_D (Gujarat)	2018-09-20 03:21:32.418600	...	2018-09-20 04:17:55	
ır_DC ijarat)	IND388620AAB	Khambhat_MotvdDPP_D (Gujarat)	2018-09-20 03:21:32.418600	...	2018-09-20 04:01:19.505586	
ır_DC ijarat)	IND388620AAB	Khambhat_MotvdDPP_D (Gujarat)	2018-09-20 03:21:32.418600	...	2018-09-20 03:39:57	
ır_DC ijarat)	IND388620AAB	Khambhat_MotvdDPP_D (Gujarat)	2018-09-20 03:21:32.418600	...	2018-09-20 03:33:55	
PP_D ijarat)	IND388320AAA	Anand_Vaghasi_IP (Gujarat)	2018-09-20 04:47:45.236797	...	2018-09-20 06:15:58	
PP_D ijarat)	IND388320AAA	Anand_Vaghasi_IP (Gujarat)	2018-09-20 04:47:45.236797	...	2018-09-20 05:47:29	
PP_D ijarat)	IND388320AAA	Anand_Vaghasi_IP (Gujarat)	2018-09-20 04:47:45.236797	...	2018-09-20 05:25:58	
PP_D ijarat)	IND388320AAA	Anand_Vaghasi_IP (Gujarat)	2018-09-20 04:47:45.236797	...	2018-09-20 05:15:56	
PP_D ijarat)	IND388320AAA	Anand_Vaghasi_IP (Gujarat)	2018-09-20 04:47:45.236797	...	2018-09-20 04:49:20	

In [489]:
```
delhivery_df.describe().T
# only numeric features
```

Out[489]:

	count	mean	std	min	25%	5
start_scan_to_end_scan	144867.0	961.262986	1037.012769	20.000000	161.000000	449.000
cutoff_factor	144867.0	232.926567	344.755577	9.000000	22.000000	66.000
actual_distance_to_destination	144867.0	234.073372	344.990009	9.000045	23.355874	66.126
actual_time	144867.0	416.927527	598.103621	9.000000	51.000000	132.000
osrm_time	144867.0	213.868272	308.011085	6.000000	27.000000	64.000
osrm_distance	144867.0	284.771297	421.119294	9.008200	29.914700	78.525
factor	144867.0	2.120107	1.715421	0.144000	1.604264	1.857
segment_actual_time	144867.0	36.196111	53.571158	-244.000000	20.000000	29.000
segment_osrm_time	144867.0	18.507548	14.775960	0.000000	11.000000	17.000
segment_osrm_distance	144867.0	22.829020	17.860660	0.000000	12.070100	23.513
segment_factor	144867.0	2.218368	4.847530	-23.444444	1.347826	1.684

In [490]:
```
# catgeorical features
delhivery_df.describe(include = ['object']).T
```

Out[490]:

	count	unique	top	freq
data	144867	2	training	104858
trip_creation_time	144867	14817	2018-09-28 05:23:15.359220	101
route_schedule_uuid	144867	1504	thanos::sroute:4029a8a2-6c74-4b7e-a6d8-f9e069f...	1812
route_type	144867	2	FTL	99660
trip_uuid	144867	14817	trip-153811219535896559	101
source_center	144867	1508	IND000000ACB	23347
source_name	144574	1498	Gurgaon_Bilaspur_HB (Haryana)	23347
destination_center	144867	1481	IND000000ACB	15192
destination_name	144606	1468	Gurgaon_Bilaspur_HB (Haryana)	15192
od_start_time	144867	26369	2018-09-21 18:37:09.322207	81
od_end_time	144867	26369	2018-09-24 09:59:15.691618	81
cutoff_timestamp	144867	93180	2018-09-24 05:19:20	40

In [491]:
```
#missing values
delhivery_df.isna().sum()
```

Out[491]:
```
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
```

In [492]:
```
# catgeorical and numerical columns
cat_cols = delhivery_df.dtypes =='object'
cat_cols = list(cat_cols[cat_cols].index)

num_cols = delhivery_df.dtypes !='object'
num_cols = list(num_cols[num_cols].index)
```

In [493]:
```
cat_cols
```

Out[493]:
```
['data',
 'trip_creation_time',
 'route_schedule_uuid',
 'route_type',
 'trip_uuid',
 'source_center',
 'source_name',
 'destination_center',
 'destination_name',
 'od_start_time',
 'od_end_time',
 'cutoff_timestamp']
```

In [494]:
```
num_cols
```

Out[494]:
```
['start_scan_to_end_scan',
 'is_cutoff',
 'cutoff_factor',
 'actual_distance_to_destination',
 'actual_time',
 'osrm_time',
 'osrm_distance',
 'factor',
 'segment_actual_time',
 'segment_osrm_time',
 'segment_osrm_distance',
 'segment_factor']
```

In [495]:
```
delhivery_df[cat_cols].head()
```

Out[495]:

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_cent
0	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AA
1	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AA
2	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AA
3	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AA
4	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AA

In [496]:
```
delhivery_df[num_cols].head()
```

Out[496]:

	start_scan_to_end_scan	is_cutoff	cutoff_factor	actual_distance_to_destination	actual_time	osrr
0	86.0	True	9	10.435660	14.0	
1	86.0	True	18	18.936842	24.0	
2	86.0	True	27	27.637279	40.0	
3	86.0	True	36	36.118028	62.0	
4	86.0	False	39	39.386040	68.0	

In [497]:
```
delhivery_df.route_type.value_counts()
```

Out[497]:
```
FTL        99660
Carting    45207
Name: route_type, dtype: int64
```

Handle Missing value

Source name and destination name missing in some rows, replacing with 'others', don't want to loss any info from available data

In [498]:
```
delhivery_df['source_name'] = delhivery_df['source_name'].fillna('Others')
delhivery_df['destination_name'] = delhivery_df['destination_name'].fillna('Other
delhivery_df
```

Out[498]:

name	destination_center	destination_name	od_start_time	...	cutoff_timestamp	actual_distan
ar_DC ujarat)	IND388620AAB	Khambhat_MotvdDPP_D (Gujarat)	2018-09-20 03:21:32.418600	...	2018-09-20 04:27:55	
ar_DC ujarat)	IND388620AAB	Khambhat_MotvdDPP_D (Gujarat)	2018-09-20 03:21:32.418600	...	2018-09-20 04:17:55	
ar_DC ujarat)	IND388620AAB	Khambhat_MotvdDPP_D (Gujarat)	2018-09-20 03:21:32.418600	...	2018-09-20 04:01:19.505586	
ar_DC ujarat)	IND388620AAB	Khambhat_MotvdDPP_D (Gujarat)	2018-09-20 03:21:32.418600	...	2018-09-20 03:39:57	
ar_DC ujarat)	IND388620AAB	Khambhat_MotvdDPP_D (Gujarat)	2018-09-20 03:21:32.418600	...	2018-09-20 03:33:55	
...	...	...	...	...	...	
ndli_H yana)	IND000000ACB	Gurgaon_Bilaspur_HB (Haryana)	2018-09-20 16:24:28.436231	...	2018-09-20 21:57:20	
ndli_H yana)	IND000000ACB	Gurgaon_Bilaspur_HB (Haryana)	2018-09-20 16:24:28.436231	...	2018-09-20 21:31:18	
ndli_H yana)	IND000000ACB	Gurgaon_Bilaspur_HB (Haryana)	2018-09-20 16:24:28.436231	...	2018-09-20 21:11:18	
ndli_H yana)	IND000000ACB	Gurgaon_Bilaspur_HB (Haryana)	2018-09-20 16:24:28.436231	...	2018-09-20 20:53:19	
ndli_H yana)	IND000000ACB	Gurgaon_Bilaspur_HB (Haryana)	2018-09-20 16:24:28.436231	...	2018-09-20 16:24:28.436231	

In [499]:
```
#missing values
delhivery_df.isna().sum()
```

Out[499]:
```
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
```

Basic Data Engineering

In [398]:
```
def getState(value):
    try:
        float(value)
        return value
    except ValueError:
        return value[value.find('(')+1: -1]
```

Createing city, place, code & state from source and destination name

In [399]:
```
# Feature Creation from existing feature(spliting city, locality, category and st

delhivery_df['source_city'] = delhivery_df['source_name'].str.split('_').str[0]
delhivery_df['source_place'] = delhivery_df['source_name'].str.split('_').str[1]
delhivery_df['source_code'] = delhivery_df['source_name'].str.split('_').str[2].s
delhivery_df['source_state'] = delhivery_df['source_name'].apply(lambda x: getSta
delhivery_df.drop('source_name', axis=1, inplace=True)
delhivery_df
```

Out[399]:

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source
0	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388
1	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388
2	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388
3	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388
4	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388
...	...	...	...	...	...	
144862	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	IND131
144863	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	IND131
144864	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	IND131
144865	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	IND131
144866	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	IND131

144867 rows × 27 columns

In [400]:
```
# Feature Creation from existing feature(spliting city, locality, category and st

delhivery_df['destination_city'] = delhivery_df['destination_name'].str.split('_'
delhivery_df['destination_place'] = delhivery_df['destination_name'].str.split('_
delhivery_df['destination_code'] = delhivery_df['destination_name'].str.split('_'
delhivery_df['destination_state'] = delhivery_df['destination_name'].apply(lambda
delhivery_df.drop('destination_name', axis=1, inplace=True)
delhivery_df
```

Out[400]:

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source
0	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388
1	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388
2	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388
3	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388
4	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388
...	...	...	...	...	...	
144862	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	IND131
144863	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	IND131
144864	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	IND131
144865	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	IND131
144866	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	IND131

144867 rows × 30 columns

Trip creating day, month & year from trip_creation_time

In [407]:
```
temp= pd.to_datetime(delhivery_df['trip_creation_time'], format='%Y-%m-%d %H:%M:%
delhivery_df['trip_creation_date'] = temp.dt.strftime('%d')
delhivery_df['trip_creation_month'] = temp.dt.strftime('%M')
delhivery_df['trip_creation_year'] = temp.dt.strftime('%Y')
delhivery_df['trip_creation_hour'] = temp.dt.strftime('%H')
delhivery_df['trip_creation_minute'] = temp.dt.strftime('%M')
delhivery_df['trip_creation_second'] = temp.dt.strftime('%S')
delhivery_df.drop('trip_creation_time', axis=1, inplace=True)
delhivery_df
# temp
# df['Time'] = df['DateTime'].dt.strftime('%H:%M')
```

Out[407]:

	data	route_schedule_uuid	route_type	trip_uuid	source_center	destinatior
0	training	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	IND388
1	training	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	IND388
2	training	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	IND388
3	training	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	IND388
4	training	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	IND388
...	...	...	...	...	...	
144862	training	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	IND131028AAB	IND000
144863	training	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	IND131028AAB	IND000
144864	training	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	IND131028AAB	IND000
144865	training	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	IND131028AAB	IND000
144866	training	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	IND131028AAB	IND000

144867 rows × 35 columns

Overall time taken (od_end_time - od_start_time)

In [421]:
```
od_start_time = pd.to_datetime(delhivery_df['od_start_time'], format='%Y-%m-%d %H
od_end_time = pd.to_datetime(delhivery_df['od_end_time'], format='%Y-%m-%d %H:%M:
diff = od_end_time - od_start_time
delhivery_df['time_difference'] = diff.dt.total_seconds() / 60
delhivery_df.drop('od_start_time', axis=1, inplace=True)
delhivery_df.drop('od_end_time', axis=1, inplace=True)
delhivery_df
```

Out[421]:

	data	route_schedule_uuid	route_type	trip_uuid	source_center	destination_cen
0	training	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	IND388620A
1	training	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	IND388620A
2	training	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	IND388620A
3	training	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	IND388620A
4	training	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	IND388620A
...	...	...	...	...	...	
862	training	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	IND131028AAB	IND000000A
863	training	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	IND131028AAB	IND000000A
864	training	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	IND131028AAB	IND000000A
865	training	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	IND131028AAB	IND000000A
866	training	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	IND131028AAB	IND000000A

867 rows × 34 columns

GroupBy based on trip id, source center & destination center

In [477]:
```
df = pd.DataFrame(delhivery_df.groupby(['trip_uuid', 'source_center', 'destinatic

df
```

Out[477]:

tate	start_scan_to_end_scan	time_difference	actual_time	osrm_time	segment_osrm_time	segment_a
ana	1260.0	1260.604421	732.0	349.0	534.0	
esh	999.0	999.505379	830.0	394.0	474.0	
aka	58.0	58.832388	47.0	26.0	26.0	
aka	122.0	122.779486	96.0	42.0	39.0	
ıjab	834.0	834.638929	611.0	212.0	231.0	
...	...	...	...	...	...	
adu	62.0	62.115193	51.0	41.0	42.0	
adu	91.0	91.087797	90.0	48.0	77.0	
adu	44.0	44.174403	30.0	14.0	14.0	
aka	287.0	287.474007	233.0	42.0	42.0	
aka	66.0	66.933565	42.0	26.0	25.0	

Categorical to Numerical encoding

1. Label encoding

In [478]:
```
from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()
cols=['trip_uuid', 'source_center', 'destination_center', 'route_type', 'source_c
for col in cols:
    df[col] = label_encoder.fit_transform(df[col].astype(str))
df
```

Out[478]:

ıte	start_scan_to_end_scan	time_difference	actual_time	osrm_time	segment_osrm_time	segment_ac
11	1260.0	1260.604421	732.0	349.0	534.0	
30	999.0	999.505379	830.0	394.0	474.0	
15	58.0	58.832388	47.0	26.0	26.0	
15	122.0	122.779486	96.0	42.0	39.0	
25	834.0	834.638929	611.0	212.0	231.0	
...	...	...	...	...	...	
27	62.0	62.115193	51.0	41.0	42.0	
27	91.0	91.087797	90.0	48.0	77.0	
27	44.0	44.174403	30.0	14.0	14.0	
15	287.0	287.474007	233.0	42.0	42.0	
15	66.0	66.933565	42.0	26.0	25.0	

In [475]:
```
df['start_scan_to_end_scan'][0]
```

Out[475]:
```
array([1260.])
```

Column Standarization and Normalization

- Mean centering and Variance scaling (Standard Scaling)
- MinMax Scaling

In [479]:
```
from sklearn.preprocessing import StandardScaler, MinMaxScaler

scaler = StandardScaler()
std_data = scaler.fit_transform(df)
std_data = pd.DataFrame(std_data, columns=df.columns)
std_data.head()
```

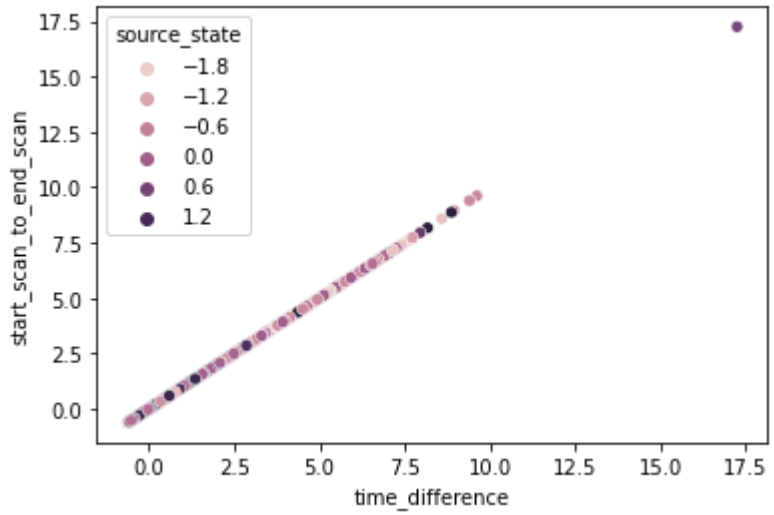
Out[479]:

te	start_scan_to_end_scan	time_difference	actual_time	osrm_time	segment_osrm_time	segment_ac
32	2.183046	2.183288	1.380576	1.388293	2.004753	
25	1.590603	1.590619	1.635224	1.630505	1.726520	
31	-0.545370	-0.544614	-0.399355	-0.350255	-0.350950	
31	-0.400097	-0.399460	-0.272031	-0.264135	-0.290667	
23	1.216070	1.216389	1.066165	0.650890	0.599678	

Compare time difference & start_scan_to_end_scan

In [502]:
```
sns.scatterplot(data=std_data, x="time_difference", y="start_scan_to_end_scan", h
```

Out[502]:
```
<AxesSubplot:xlabel='time_difference', ylabel='start_scan_to_end_scan'>
```



With visual analysis unable to see difference between time_difference and start_scan_to_end_scan.

Need to check using statistical methods

Step 1: Define the null and alternate hypotheses

H_0: Calculated delivery time (time difference) is equal to start_scan_to_end_scan.

H_a: Calculated delivery time (time difference) is not equal to start_scan_to_end_scan

Let μ_1 and μ_2 be the mean calculated delivery time (time difference) and start_scan_to_end_scan respectively.

Mathematically, the above formulated hypotheses can be written as:

$$H_0 : \mu_1 = \mu_2$$

$$H_a : \mu_1 <> \mu_2$$

Step 2: Select Appropriate test

This is a two-tailed test concerning two population means from two independent populations. As the population standard deviations are unknown, the two sample independent t-test will be the appropriate test for this problem.

Step 3: Decide the significance level

As given in the problem statement, we select α = 0.05.

Step 4: Collect and prepare data

In [513]:
```
start_scan_to_end_scan = df['start_scan_to_end_scan']
time_difference = df['time_difference']
print('The sample standard deviation of the start scan to end scan time :', start
print('The sample standard deviation of the calculated delivery time :', time_dif
```

```
The sample standard deviation of the start scan to end scan time : <bound metho
d NDFrame._add_numeric_operations.<locals>.std of 0        1260.0
1         999.0
2          58.0
3         122.0
4         834.0
          ...
26363      62.0
26364      91.0
26365      44.0
26366     287.0
26367      66.0
Name: start_scan_to_end_scan, Length: 26368, dtype: float64>
The sample standard deviation of the calculated delivery time : <bound method N
DFrame._add_numeric_operations.<locals>.std of 0        1260.604421
1         999.505379
2          58.832388
3         122.779486
4         834.638929
            ...
26363      62.115193
26364      91.087797
26365      44.174403
26366     287.474007
26367      66.933565
Name: time_difference, Length: 26368, dtype: float64>
```

As the sample standard deviations closer to each other, the population standard deviations may be assumed to be closer.

Step 5: Calculate the p-value

In [517]:
```
# import the required function
from scipy.stats import ttest_ind
# find the p-value
test_stat, p_value = ttest_ind(start_scan_to_end_scan, time_difference, equal_var
print('The p-value is', p_value)

# print the conclusion based on p-value
if p_value < 0.05:
    print(f'As the p-value {p_value} is less than the level of significance, we r
else:
    print(f'As the p-value {p_value} is greater than the level of significance, w
```

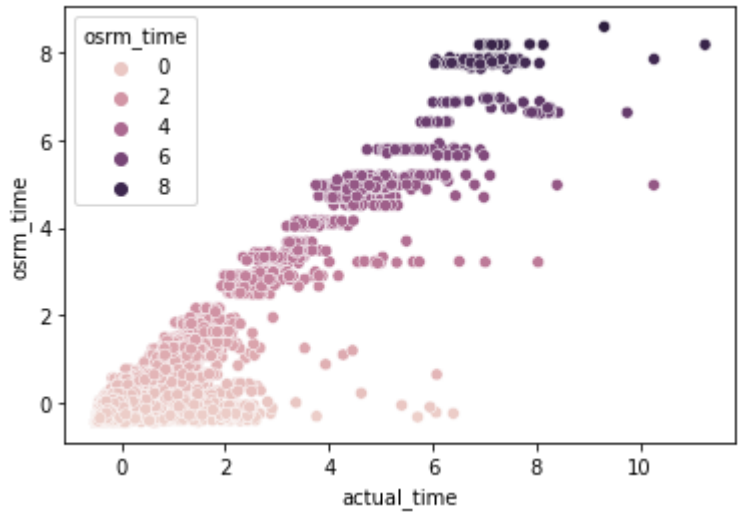
```
The p-value is 0.8965583786340379
As the p-value 0.8965583786340379 is greater than the level of significance, we
accepting H-0
```

As the p-value 0.8965583786340379 is greater than the level of significance, we accepting the H-0

Compare actual_time aggregated OSRM time aggregated value

In [520]:
```
sns.scatterplot(data=std_data, x="actual_time", y="osrm_time", hue='osrm_time')
```

Out[520]:
```
<AxesSubplot:xlabel='actual_time', ylabel='osrm_time'>
```



With visual analysis data scattered across there may be difference between actual time aggregaed and OSRM time aggregated

Need to check using statistical methods

Step 1: Define the null and alternate hypotheses

H_0: actual_time aggregated is equal to OSRM time aggregated values.

H_a: actual_time aggregated is not equal to OSRM time aggregated values.

Let μ_1 and μ_2 be the mean actual_time aggregated and mean OSRM time aggregated values respectively.

Mathematically, the above formulated hypotheses can be written as:

$$H_0 : \mu_1 = \mu_2$$

$$H_a : \mu_1 <> \mu_2$$

Step 2: Select Appropriate test

This is a two-tailed test concerning two population means from two independent populations. As the population standard deviations are unknown, the two sample independent t-test will be the appropriate test for this problem.

Step 3: Decide the significance level

As given in the problem statement, we select α = 0.05.

Step 4: Collect and prepare data

In [521]:
```
actual_time = df['actual_time']
osrm_time = df['osrm_time']
print('The sample standard deviation of the actual_time aggregated value is :', a
print('The sample standard deviation of the osrm_time aggregated value is :', osr
```

```
The sample standard deviation of the actual_time aggregated value is : <bound m
ethod NDFrame._add_numeric_operations.<locals>.std of 0        732.0
1        830.0
2         47.0
3         96.0
4        611.0
         ...
26363     51.0
26364     90.0
26365     30.0
26366    233.0
26367     42.0
Name: actual_time, Length: 26368, dtype: float64>
The sample standard deviation of the osrm_time aggregated value is : <bound met
hod NDFrame._add_numeric_operations.<locals>.std of 0        349.0
1        394.0
2         26.0
3         42.0
4        212.0
         ...
26363     41.0
26364     48.0
26365     14.0
26366     42.0
26367     26.0
Name: osrm_time, Length: 26368, dtype: float64>
```

As the sample standard deviations are different, the population standard deviations may be assumed to be different.

Step 5: Calculate the p-value

In [524]:
```
# import the required function
from scipy.stats import ttest_ind
# find the p-value
test_stat, p_value = ttest_ind(actual_time, osrm_time, equal_var = True, alternat
print('The p-value is', p_value)

# print the conclusion based on p-value
if p_value < 0.05:
    print(f'As the p-value {p_value} is less than the level of significance, we r
else:
    print(f'As the p-value {p_value} is greater than the level of significance, w
```

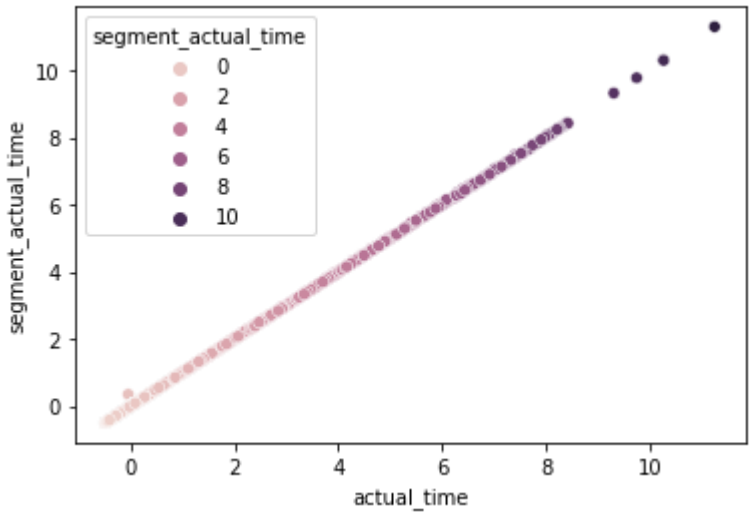
```
The p-value is 0.0
As the p-value 0.0 is less than the level of significance, we rejecting H-0
```

As the p-value 0.0 is less than the level of significance, we rejecting the H-0

Compare actual_time aggregated value and segment actual time aggregated value

In [525]:
```
sns.scatterplot(data=std_data, x="actual_time", y="segment_actual_time", hue='seg
```

Out[525]:
```
<AxesSubplot:xlabel='actual_time', ylabel='segment_actual_time'>
```



With visual analysis unable to see difference between actual_time and segment_actual_time.

Need to check using statistical methods

Step 1: Define the null and alternate hypotheses

H_0: actual_time is equal to segment_actual_time.

H_a: actual_time is not equal to segment_actual_time.

Let μ_1 and μ_2 be the mean actual_time and segment_actual_time respectively.

Mathematically, the above formulated hypotheses can be written as:

$H_0 : \mu_1 = \mu_2$

$H_a : \mu_1 <> \mu_2$

Step 2: Select Appropriate test

This is a two-tailed test concerning two population means from two independent populations. As the population standard deviations are unknown, the two sample independent t-test will be the appropriate test for this problem.

Step 3: Decide the significance level

As given in the problem statement, we select α = 0.05.

Step 4: Collect and prepare data

In [526]:
```
actual_time = df['actual_time']
segment_actual_time = df['segment_actual_time']
print('The sample standard deviation of the actual_time :', actual_time.std)
print('The sample standard deviation of the segment_actual_time :', segment_actua
```

```
The sample standard deviation of the actual_time : <bound method NDFrame._add_n
umeric_operations.<locals>.std of 0        732.0
1        830.0
2         47.0
3         96.0
4        611.0
         ...
26363     51.0
26364     90.0
26365     30.0
26366    233.0
26367     42.0
Name: actual_time, Length: 26368, dtype: float64>
The sample standard deviation of the segment_actual_time : <bound method NDFram
e._add_numeric_operations.<locals>.std of 0        728.0
1        820.0
2         46.0
3         95.0
4        608.0
         ...
26363     49.0
26364     89.0
26365     29.0
26366    233.0
26367     41.0
Name: segment_actual_time, Length: 26368, dtype: float64>
```

As the sample standard deviations closer to each other, the population standard deviations may be assumed to be closer.

Step 5: Calculate the p-value

In [527]:
```
# import the required function
from scipy.stats import ttest_ind
# find the p-value
test_stat, p_value = ttest_ind(actual_time, segment_actual_time, equal_var = True
print('The p-value is', p_value)

# print the conclusion based on p-value
if p_value < 0.05:
    print(f'As the p-value {p_value} is less than the level of significance, we r
else:
    print(f'As the p-value {p_value} is greater than the level of significance, w
```

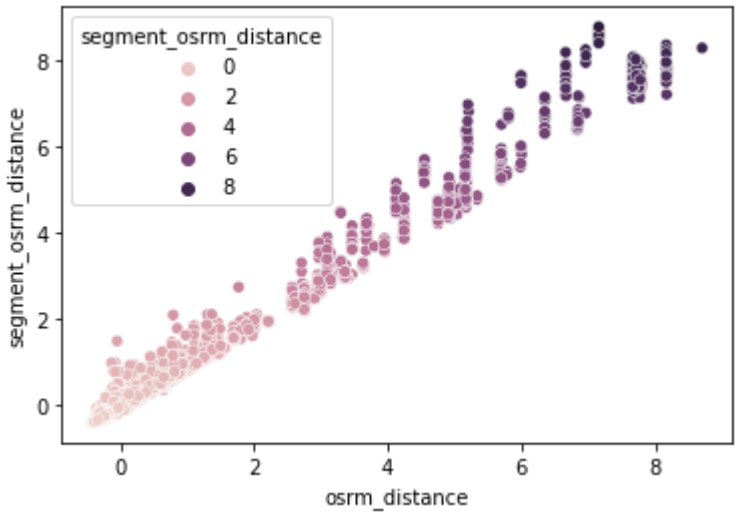
```
The p-value is 0.5839328464797933
As the p-value 0.5839328464797933 is greater than the level of significance, we
accepting H-0
```

As the p-value 0.5839328464797933 is greater than the level of significance, we accepting the H-0

Compare osrm distance aggregated value and segment osrm distance aggregated value

In [528]:
```
sns.scatterplot(data=std_data, x="osrm_distance", y="segment_osrm_distance", hue=
```

Out[528]:
```
<AxesSubplot:xlabel='osrm_distance', ylabel='segment_osrm_distance'>
```



With visual analysis data scattered across there may be difference between osrm distance aggregated value and segment osrm distance aggregated value Need to check using statistical methods

Step 1: Define the null and alternate hypotheses

H_0: osrm distance aggregated value is equal to segment osrm distance aggregated values.

H_a: osrm distance aggregated value is not equal to segment osrm distance aggregated values.

Let μ_1 and μ_2 be the mean osrm distance aggregated value and mean segment osrm distance aggregated value respectively.

Mathematically, the above formulated hypotheses can be written as:

$$H_0 : \mu_1 = \mu_2$$

$$H_a : \mu_1 <> \mu_2$$

Step 2: Select Appropriate test

This is a two-tailed test concerning two population means from two independent populations. As the population standard deviations are unknown, the two sample independent t-test will be the appropriate test for this problem.

Step 3: Decide the significance level

As given in the problem statement, we select α = 0.05.

Step 4: Collect and prepare data

In [533]:
```
osrm_distance = df['osrm_distance']
segment_osrm_distance = df['segment_osrm_distance']
print('The sample standard deviation of the osrm_distance aggregated value is :',
print('The sample standard deviation of the segment_osrm_distance aggregated valu
```

```
The sample standard deviation of the osrm_distance aggregated value is : <bound
method NDFrame._add_numeric_operations.<locals>.std of 0        446.5496
1        544.8027
2         28.1994
3         56.9116
4        281.2109
           ...
26363     42.5213
26364     40.6080
26365     16.0185
26366     52.5303
26367     28.0484
Name: osrm_distance, Length: 26368, dtype: float64>
The sample standard deviation of the segment_osrm_distance aggregated value is
: <bound method NDFrame._add_numeric_operations.<locals>.std of 0        670.62
05
1        649.8528
2         28.1995
3         55.9899
4        317.7408
           ...
26363     42.1431
26364     78.5869
26365     16.0184
26366     52.5303
26367     28.0484
Name: segment_osrm_distance, Length: 26368, dtype: float64>
```

As the sample standard deviations are different, the population standard deviations may be assumed to be different.

Step 5: Calculate the p-value

In [530]:
```
# import the required function
from scipy.stats import ttest_ind
# find the p-value
test_stat, p_value = ttest_ind(osrm_distance, segment_osrm_distance, equal_var =
print('The p-value is', p_value)

# print the conclusion based on p-value
if p_value < 0.05:
    print(f'As the p-value {p_value} is less than the level of significance, we r
else:
    print(f'As the p-value {p_value} is greater than the level of significance, w
```

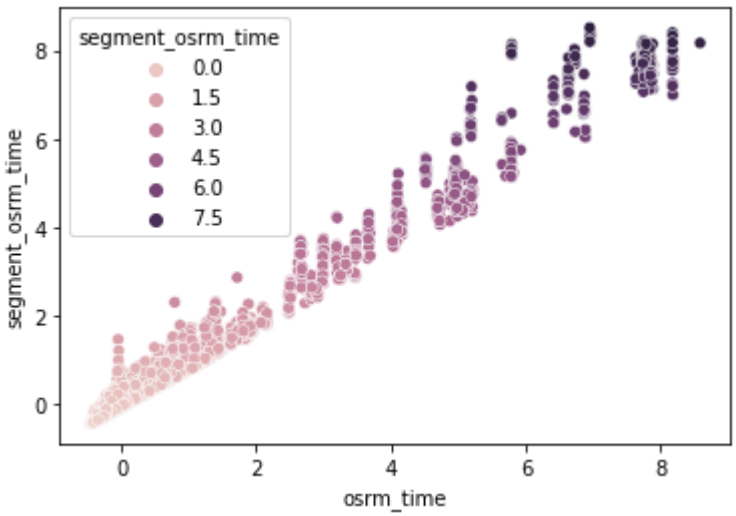
```
The p-value is 1.57892754563369e-05
As the p-value 1.57892754563369e-05 is less than the level of significance, we
rejecting H-0
```

As the p-value 1.57892754563369e-05 is less than the level of significance, we rejecting the H-0

Compare osrm time aggregated value and segment osrm time aggregated value

In [532]:
```
sns.scatterplot(data=std_data, x="osrm_time", y="segment_osrm_time", hue='segment
```

Out[532]:
```
<AxesSubplot:xlabel='osrm_time', ylabel='segment_osrm_time'>
```



With visual analysis data scattered across there may be difference between osrm distance aggregated value and segment osrm distance aggregated value Need to check using statistical methods

Step 1: Define the null and alternate hypotheses

H_0: osrm_time aggregated value is equal to segment_osrm_time aggregated values.

H_a: osrm_time aggregated value is not equal to segment_osrm_time aggregated values.

Let μ_1 and μ_2 be the mean osrm_time aggregated value and mean segment osrm_time aggregated value respectively.

Mathematically, the above formulated hypotheses can be written as:

$$H_0 : \mu_1 = \mu_2$$

$$H_a : \mu_1 <> \mu_2$$

Step 2: Select Appropriate test

This is a two-tailed test concerning two population means from two independent populations. As the population standard deviations are unknown, the two sample independent t-test will be the appropriate test for this problem.

Step 3: Decide the significance level

As given in the problem statement, we select α = 0.05.

Step 4: Collect and prepare data

In [535]:
```
osrm_time = df['osrm_time']
segment_osrm_time = df['segment_osrm_time']
print('The sample standard deviation of the osrm_time aggregated value is :', osr
print('The sample standard deviation of the segment_osrm_time aggregated value is
```

```
The sample standard deviation of the osrm_time aggregated value is : <bound met
hod NDFrame._add_numeric_operations.<locals>.std of 0        349.0
1        394.0
2         26.0
3         42.0
4        212.0
         ...
26363     41.0
26364     48.0
26365     14.0
26366     42.0
26367     26.0
Name: osrm_time, Length: 26368, dtype: float64>
The sample standard deviation of the segment_osrm_time aggregated value is : <b
ound method NDFrame._add_numeric_operations.<locals>.std of 0        534.0
1        474.0
2         26.0
3         39.0
4        231.0
         ...
26363     42.0
26364     77.0
26365     14.0
26366     42.0
26367     25.0
Name: segment_osrm_time, Length: 26368, dtype: float64>
```

As the sample standard deviations are different, the population standard deviations may be assumed to be different.

Step 5: Calculate the p-value

In [536]:
```
# import the required function
from scipy.stats import ttest_ind
# find the p-value
test_stat, p_value = ttest_ind(osrm_time, segment_osrm_time, equal_var = True, al
print('The p-value is', p_value)

# print the conclusion based on p-value
if p_value < 0.05:
    print(f'As the p-value {p_value} is less than the level of significance, we r
else:
    print(f'As the p-value {p_value} is greater than the level of significance, w
```

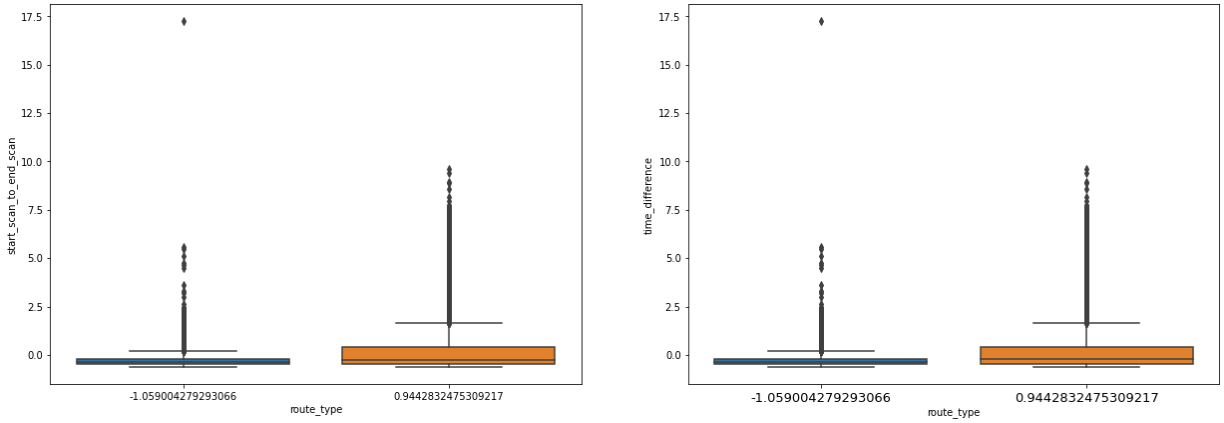
```
The p-value is 1.441878590900604e-09
As the p-value 1.441878590900604e-09 is less than the level of significance, we
rejecting H-0
```

As the p-value 1.441878590900604e-09 is less than the level of significance, we rejecting the H-0

Outlier detection

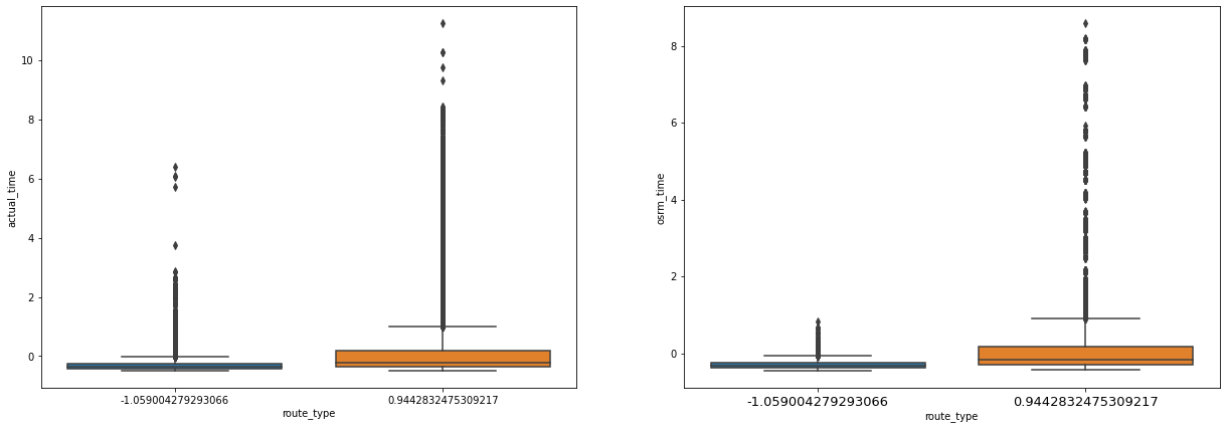
In [538]:
```
fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=(21,7))
sns.boxplot(x = 'route_type', y = 'start_scan_to_end_scan', data=std_data, ax=ax1
sns.boxplot(x = 'route_type', y = 'time_difference', data=std_data, ax=ax2)
# sns.boxplot(x = 'type', y = 'Season', data=netflix_df, ax=ax3)
plt.xticks(fontsize= 13)
# plt.title('Box plot of numerical columns', fontsize=16);
```

Out[538]:
```
(array([0, 1]),
 [Text(0, 0, '-1.059004279293066'), Text(1, 0, '0.9442832475309217')])
```



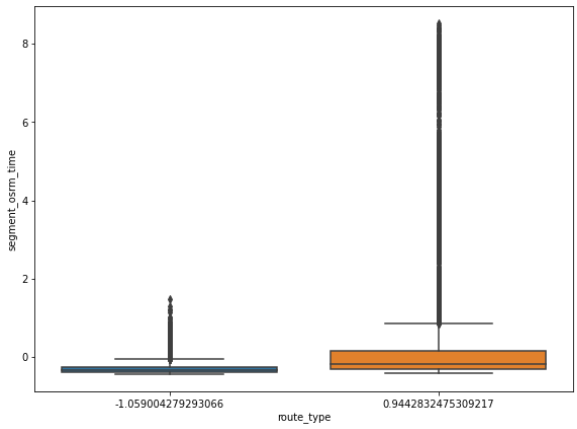
In [539]:
```
fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=(21,7))
sns.boxplot(x = 'route_type', y = 'actual_time', data=std_data, ax=ax1)
sns.boxplot(x = 'route_type', y = 'osrm_time', data=std_data, ax=ax2)
# sns.boxplot(x = 'type', y = 'Season', data=netflix_df, ax=ax3)
plt.xticks(fontsize= 13)
# plt.title('Box plot of numerical columns', fontsize=16);
```

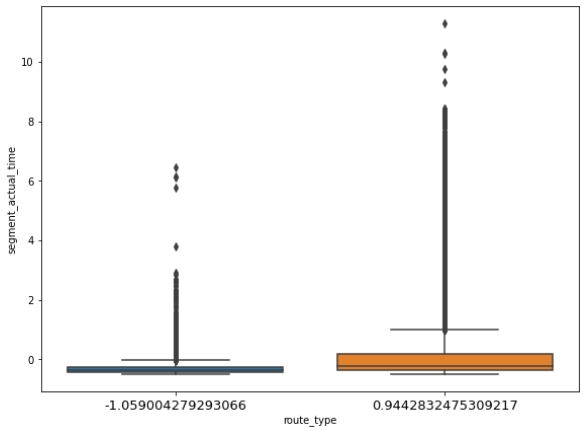
Out[539]:
```
(array([0, 1]),
 [Text(0, 0, '-1.059004279293066'), Text(1, 0, '0.9442832475309217')])
```



In [540]:
```
fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=(21,7))
sns.boxplot(x = 'route_type', y = 'segment_osrm_time', data=std_data, ax=ax1)
sns.boxplot(x = 'route_type', y = 'segment_actual_time', data=std_data, ax=ax2)
# sns.boxplot(x = 'type', y = 'Season', data=netflix_df, ax=ax3)
plt.xticks(fontsize= 13)
# plt.title('Box plot of numerical columns', fontsize=16);
```

Out[540]:
```
(array([0, 1]),
 [Text(0, 0, '-1.059004279293066'), Text(1, 0, '0.9442832475309217')])
```





Recommendation

From the above analysis,

there is difference between actual time & OSRM time (OSRM time is less than the actual time)

Delhivery can try to use the OSRM path it may lead to better result in delivery