Problem Statement : Delhivery is the largest and fastest-growing fully integrated player in India by revenue in Fiscal 2021. 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. In this case study we will clean, sanitize and manipulate data to get useful and meaningful features out of the raw data given to us. We will use feature engineering to deal with the above functionalities we want to perform on the raw data given to us. This will help the data science team to build forecasting models on it

In [1]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [2]:

```
from scipy import stats
from scipy.stats import ttest_ind # Numeric Vs categorical
from scipy.stats import ttest_1samp
from scipy.stats import ranksums
from sklearn import preprocessing
```

In [3]:

```
from sklearn.preprocessing import OneHotEncoder
```

In [4]:

```
delhivery = pd.read_csv('C:/Prakruthi/DSML/Delhivery - Case study/delhivery_data.csv')
```

In [5]:

```
delhivery.shape
```

Out[5]:

```
(144867, 24)
```

In [6]:

```
delhivery.columns
```

Out[6]:

```
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 [7]:

```
print(delhivery['trip_uuid'].nunique())
print(delhivery['source_center'].nunique())
print(delhivery['destination_center'].nunique())
delhivery.groupby('route_type')['route_type'].value_counts().sort_values(ascending=False)
```

```
14817
1508
1481
```

Out[7]:

```
route_type  route_type
FTL         FTL           99660
Carting     Carting       45207
Name: route_type, dtype: int64
```

```
Basic metrics: We have a total of 144867 rows which represent the number of sub
trips(there will be multiple rows for the same trip id) and 24 attributes/columns which
are going to use for our data cleanzing. These basic metrics will keep changing as and
when we keep cleanzing our data

1. There are a total of 14817 trip IDs available in the dataset
2. A total of 1508 source centers are present
3. A total of 1481 destination centers are present
4. FTL route type is used much more than the carting type. This could since it is faster.
```

In [8]:

```
delhivery.head()
```

Out[8]:

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_c
0	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND38812
1	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND38812
2	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND38812
3	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND38812
4	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND38812

5 rows × 24 columns

In [9]:

```
delhivery.dtypes
```

Out[9]:

```
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
```

Missing values treatment

In [10]:

```
delhivery.isna().sum()
```

Out[10]:

```
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 [11]:

```
def missing_to_df(df):
    #Number and percentage of missing data in training data set for each column
    total_missing_df = df.isnull().sum().sort_values(ascending =False)
    percent_missing_df = (df.isnull().sum()/df.isnull().count()*100).sort_values(ascending=
    missing_data_df = pd.concat([total_missing_df, percent_missing_df], axis=1, keys=['Tota
    return missing_data_df
```

In [12]:

```
missing_df = missing_to_df(delhivery)
missing_df[missing_df['Total'] > 0]
```

Out[12]:

	Total	Percent
source_name	293	0.202254
destination_name	261	0.180165

In [13]:

```
delhivery['source_name'] = delhivery['source_name'].fillna('Other')
delhivery['destination_name'] = delhivery['destination_name'].fillna('Other')
```

Statistical Summary

In [14]:

```
delhivery[['start_scan_to_end_scan','actual_distance_to_destination','actual_time','osrm_ti
```

Out[14]:

	start_scan_to_end_scan	actual_distance_to_destination	actual_time	osrm_time	os
count	144867.000000	144867.000000	144867.000000	144867.000000	14
mean	961.262986	234.073372	416.927527	213.868272	
std	1037.012769	344.990009	598.103621	308.011085	
min	20.000000	9.000045	9.000000	6.000000	
25%	161.000000	23.355874	51.000000	27.000000	
50%	449.000000	66.126571	132.000000	64.000000	
75%	1634.000000	286.708875	513.000000	257.000000	
max	7898.000000	1927.447705	4532.000000	1686.000000	

Visual Analysis

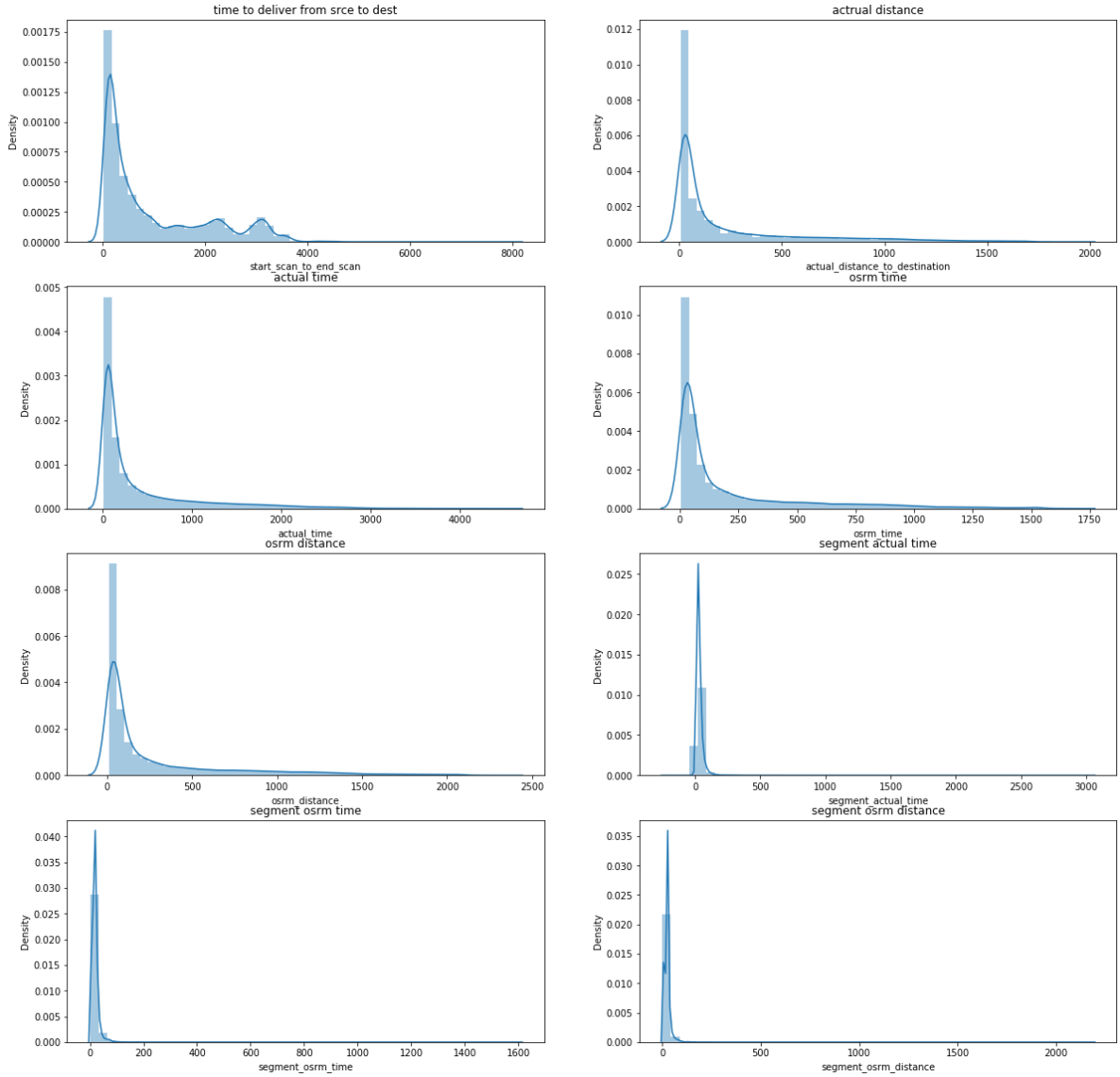
In [15]:

```
# Distribution plot for continous variable to visualize their distributions, outlier check
fig, axis = plt.subplots(nrows=4, ncols=2, figsize=(20, 20))
sns.distplot(delhivery['start_scan_to_end_scan'],ax=axis[0,0]).set_title('time to deliver f
sns.distplot(delhivery['actual_distance_to_destination'],ax=axis[0,1]).set_title('actrual d
sns.distplot(delhivery['actual_time'],ax=axis[1,0]).set_title('actual time')
sns.distplot(delhivery['osrm_time'],ax=axis[1,1]).set_title('osrm time')
sns.distplot(delhivery['osrm_distance'],ax=axis[2,0]).set_title('osrm distance')
sns.distplot(delhivery['segment_actual_time'],ax=axis[2,1]).set_title('segment actual time'
sns.distplot(delhivery['segment_osrm_time'],ax=axis[3,0]).set_title('segment osrm time')
sns.distplot(delhivery['segment_osrm_distance'],ax=axis[3,1]).set_title('segment osrm dista
```

```
C:\Users\sanke\Anaconda3\lib\site-packages\seaborn\distributions.py:2551: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\sanke\Anaconda3\lib\site-packages\seaborn\distributions.py:2551: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\sanke\Anaconda3\lib\site-packages\seaborn\distributions.py:2551: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\sanke\Anaconda3\lib\site-packages\seaborn\distributions.py:2551: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\sanke\Anaconda3\lib\site-packages\seaborn\distributions.py:2551: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\sanke\Anaconda3\lib\site-packages\seaborn\distributions.py:2551: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\sanke\Anaconda3\lib\site-packages\seaborn\distributions.py:2551: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\sanke\Anaconda3\lib\site-packages\seaborn\distributions.py:2551: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

Out[15]:

```
Text(0.5, 1.0, 'segment osrm distance')
```



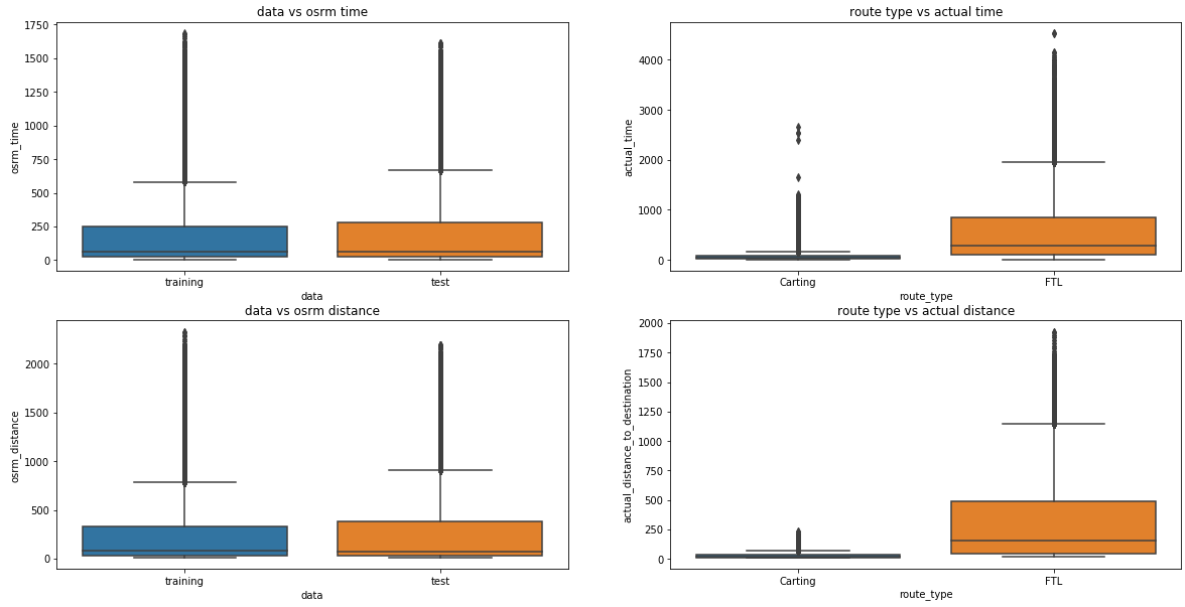
Here, we can clearly see that all the time and distance fields are right skewed and do not have a normal distribution

In [16]:

```
# Visual analysis using boxplots for categorical variables
fig, axis = plt.subplots(nrows=2, ncols=2, figsize=(20, 10))
sns.boxplot(data=delhivery,x='data',y='osrm_time',ax=axis[0,0]).set_title('data vs osrm tim
sns.boxplot(data=delhivery,x='route_type',y='actual_time',ax=axis[0,1]).set_title('route ty
sns.boxplot(data=delhivery,x='data',y='osrm_distance',ax=axis[1,0]).set_title('data vs osrm
sns.boxplot(data=delhivery,x='route_type',y='actual_distance_to_destination',ax=axis[1,1]).
```

Out[16]:

Text(0.5, 1.0, 'route type vs actual distance')



Observations based on the above visual analysis
1. Both training and test data have the same range of osrm time recorded
2. FTL route type has more actual time compared to Carting. This can also be since FTL is used a lot more than carting in the data available to us
3. Both training and test data have the same range of osrm distance recorded
4. FTL route type has more actual distance compared to Carting. This can also be since FTL is used a lot more than carting in the data available to us

Merging of rows and aggregation of fields

In [17]:

```
#Grouping of sub-journey in the trip
delhivery['segment_key'] = delhivery['trip_uuid'] + delhivery['source_center'] + delhivery[
segment_cols = ['segment_actual_time', 'segment_osrm_distance', 'segment_osrm_time']

for col in segment_cols:
    delhivery[col + '_sum'] = delhivery.groupby('segment_key')[col].cumsum()

delhivery
```

segment_key	segment_actual_time_sum	segment_osrm_distance_sum	segment_os
trip-1093647649320IND388121AAAIND388620AAB	14.0	11.9653	
trip-1093647649320IND388121AAAIND388620AAB	24.0	21.7243	
trip-1093647649320IND388121AAAIND388620AAB	40.0	32.5395	
trip-1093647649320IND388121AAAIND388620AAB	61.0	45.5619	
trip-	67.0	49.4772	

In [18]:

```
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'

}
```

In [19]:

```
#Grouping by each mini trip
segment = delhivery.groupby('segment_key').agg(segment_dict).reset_index()
segment = segment.sort_values(by=['segment_key','od_end_time'],ascending=True).reset_index(
```

In [20]:

```
# Finding the trip duration by using the od start time and end time
segment['od_start_time'] = segment['od_start_time'].astype('datetime64')
segment['od_end_time'] = segment['od_end_time'].astype('datetime64')
segment['od_time_diff_hour'] = (segment['od_end_time'] - segment['od_start_time']) / pd.Tim
segment.drop(['od_start_time', 'od_end_time'], axis=1, inplace=True)
```

In [21]:

```
segment
```

rm_time	osrm_distance	segment_actual_time_sum	segment_osrm_distance_sum	segment_osrm_time_sum	od_
329.0	446.5496	728.0	670.6205	534.0	
388.0	544.8027	820.0	649.8528	474.0	
26.0	28.1994	46.0	28.1995	26.0	
42.0	56.9116	95.0	55.9899	39.0	
212.0	281.2109	608.0	317.7408	231.0	

In [22]:

```
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',
    'od_time_diff_hour' : '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'

}
```

In [23]:

```
trip = segment.groupby('segment_key').agg(trip_dict).reset_index(drop=True)
```

In [25]:

```
trip
```

estination	actual_time	osrm_time	osrm_distance	segment_actual_time_sum	segment_osrm_distance_sum	seg
3.759164	732.0	329.0	446.5496	728.0	670.6205	
0.973689	830.0	388.0	544.8027	820.0	649.8528	
4.644021	47.0	26.0	28.1994	46.0	28.1995	
8.542890	96.0	42.0	56.9116	95.0	55.9899	
7.439610	611.0	212.0	281.2109	608.0	317.7408	

In [ ]:

```
```

Now we have the data ready for us at the TRIP ID level. This is much more intuitive and accurate for training or testing the model

Feature Creation

In [42]:

```
#Splitting and extracting the source_name to get the city, place and code as separate featu

trip["source_city"] = ""
trip["source_place"] = ""
trip["source_code"] = ""
trip["source_state"] = ""
for i in range(len(trip['source_name'])):
    split_string = trip['source_name'][i].split("(")
    if len(split_string) > 1:
        split_state = split_string[1].split(")")
        trip["source_state"][i] = split_state[0]
    else:
        trip["source_state"][i] = ''
    split_string_2 = split_string[0].split("_")
    trip["source_city"][i] = split_string_2[0]
    if len(split_string_2) == 2:
        trip['source_place'][i] = split_string_2[1]
        trip['source_code'][i] = ''
    if len(split_string_2) == 3:
        trip['source_place'][i] = split_string_2[1]
        trip['source_code'][i] = split_string_2[2]
    if len(split_string_2) == 4:
        trip['source_place'][i] = split_string_2[1]
        trip['source_code'][i] = split_string_2[2] + split_string_2[3]
```

```
C:\Users\sanke\Anaconda3\lib\site-packages\ipykernel_launcher.py:11: Setting
WithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/s
table/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pand
as-docs/stable/indexing.html#indexing-view-versus-copy)
  # This is added back by InteractiveShellApp.init_path()
C:\Users\sanke\Anaconda3\lib\site-packages\ipykernel_launcher.py:15: Setting
WithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/s
table/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pand
as-docs/stable/indexing.html#indexing-view-versus-copy)
  from ipykernel import kernelapp as app
C:\Users\sanke\Anaconda3\lib\site-packages\ipykernel_launcher.py:23: Setting
WithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/s
table/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pand
as-docs/stable/indexing.html#indexing-view-versus-copy)
C:\Users\sanke\Anaconda3\lib\site-packages\ipykernel_launcher.py:24: Setting
WithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/s
table/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pand
as-docs/stable/indexing.html#indexing-view-versus-copy)
C:\Users\sanke\Anaconda3\lib\site-packages\ipykernel_launcher.py:20: Setting
WithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame
```

```
See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/s
table/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pand
as-docs/stable/indexing.html#indexing-view-versus-copy)
C:\Users\sanke\Anaconda3\lib\site-packages\ipykernel_launcher.py:21: Setting
WithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/s
table/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pand
as-docs/stable/indexing.html#indexing-view-versus-copy)
C:\Users\sanke\Anaconda3\lib\site-packages\ipykernel_launcher.py:17: Setting
WithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/s
table/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pand
as-docs/stable/indexing.html#indexing-view-versus-copy)
C:\Users\sanke\Anaconda3\lib\site-packages\ipykernel_launcher.py:18: Setting
WithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/s
table/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pand
as-docs/stable/indexing.html#indexing-view-versus-copy)
C:\Users\sanke\Anaconda3\lib\site-packages\ipykernel_launcher.py:13: Setting
WithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/s
table/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pand
as-docs/stable/indexing.html#indexing-view-versus-copy)
  del sys.path[0]
```

In [59]:

```
#Splitting and extracting the destination_name to get the city, place and code as separate

trip["destination_city"] = ""
trip["destination_place"] = ""
trip["destination_code"] = ""
trip["destination_state"] = ""
for i in range(len(trip['destination_name'])):
    split_string = trip['destination_name'][i].split("(")
    if len(split_string) > 1:
        split_state = split_string[1].split(")")
        trip["destination_state"][i] = split_state[0]
    else:
        trip["destination_state"][i] = ''
    split_string_2 = split_string[0].split("_")
    trip["destination_city"][i] = split_string_2[0]
    if len(split_string_2) == 2:
        trip['destination_place'][i] = split_string_2[1]
        trip['destination_code'][i] = ''
    if len(split_string_2) == 3:
        trip['destination_place'][i] = split_string_2[1]
        trip['destination_code'][i] = split_string_2[2]
    if len(split_string_2) == 4:
        trip['destination_place'][i] = split_string_2[1]
        trip['destination_code'][i] = split_string_2[2] + split_string_2[3]
```

```
C:\Users\sanke\Anaconda3\lib\site-packages\ipykernel_launcher.py:11: Setti
ngWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-doc
s/stable/indexing.html#indexing-view-versus-copy (http://pandas.pydata.or
g/pandas-docs/stable/indexing.html#indexing-view-versus-copy)
  # This is added back by InteractiveShellApp.init_path()
C:\Users\sanke\Anaconda3\lib\site-packages\ipykernel_launcher.py:15: Setti
ngWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-doc
s/stable/indexing.html#indexing-view-versus-copy (http://pandas.pydata.or
g/pandas-docs/stable/indexing.html#indexing-view-versus-copy)
  from ipykernel import kernelapp as app
C:\Users\sanke\Anaconda3\lib\site-packages\ipykernel_launcher.py:20: Setti
ngWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-doc
s/stable/indexing.html#indexing-view-versus-copy (http://pandas.pydata.or
g/pandas-docs/stable/indexing.html#indexing-view-versus-copy)
C:\Users\sanke\Anaconda3\lib\site-packages\ipykernel_launcher.py:21: Setti
ngWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-doc
s/stable/indexing.html#indexing-view-versus-copy (http://pandas.pydata.or
g/pandas-docs/stable/indexing.html#indexing-view-versus-copy)
C:\Users\sanke\Anaconda3\lib\site-packages\ipykernel_launcher.py:23: Setti
ngWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame
```

```
See the caveats in the documentation: http://pandas.pydata.org/pandas-doc
s/stable/indexing.html#indexing-view-versus-copy (http://pandas.pydata.or
g/pandas-docs/stable/indexing.html#indexing-view-versus-copy)
C:\Users\sanke\Anaconda3\lib\site-packages\ipykernel_launcher.py:24: Setti
ngWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-doc
s/stable/indexing.html#indexing-view-versus-copy (http://pandas.pydata.or
g/pandas-docs/stable/indexing.html#indexing-view-versus-copy)
C:\Users\sanke\Anaconda3\lib\site-packages\ipykernel_launcher.py:17: Setti
ngWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-doc
s/stable/indexing.html#indexing-view-versus-copy (http://pandas.pydata.or
g/pandas-docs/stable/indexing.html#indexing-view-versus-copy)
C:\Users\sanke\Anaconda3\lib\site-packages\ipykernel_launcher.py:18: Setti
ngWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-doc
s/stable/indexing.html#indexing-view-versus-copy (http://pandas.pydata.or
g/pandas-docs/stable/indexing.html#indexing-view-versus-copy)
C:\Users\sanke\Anaconda3\lib\site-packages\ipykernel_launcher.py:13: Setti
ngWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-doc
s/stable/indexing.html#indexing-view-versus-copy (http://pandas.pydata.or
g/pandas-docs/stable/indexing.html#indexing-view-versus-copy)
  del sys.path[0]
```

In [60]:

```
# create a new column corridor to help draw business insights
trip['corridor'] = trip['source_city'] + '_' + trip['destination_city']
```

In [61]:

```
trip['trip_creation_year'] = pd.to_datetime(trip['trip_creation_time']).dt.year
trip['trip_creation_month'] = pd.to_datetime(trip['trip_creation_time']).dt.month
trip['trip_creation_day'] = pd.to_datetime(trip['trip_creation_time']).dt.day
```

In [62]:

```
trip
```

tination_place	destination_code	destination_state	corridor	trip_creation_year	trip_creation_mont
Bilaspur	HB	Haryana	Kanpur_Gurgaon	2018	
Central	H6	Uttar Pradesh	Bhopal_Kanpur	2018	
ShntiSgr	D	Karnataka	Doddablpur_Chikblapur	2018	
ChikaDPP	D	Karnataka	Tumkur_Doddablpur	2018	
Mehmdpur	H	Punjab	Gurgaon_Chandigarh	2018	

Now that we have both places and time columns, we can use them to draw different insights and recommendations by doing some analysis around them. This will help in making some important business decisions

Comparison & Visualization of time and distance fields

In [63]:

```
trip.columns
```

Out[63]:

```
Index(['data', 'trip_creation_time', 'route_schedule_uuid', 'route_type',
       'trip_uuid', 'source_center', 'source_name', 'destination_center',
       'destination_name', 'start_scan_to_end_scan', 'od_time_diff_hour',
       'actual_distance_to_destination', 'actual_time', 'osrm_time',
       'osrm_distance', 'segment_actual_time_sum', 'segment_osrm_distance_su
m',
       'segment_osrm_time_sum', 'source_city', 'source_place', 'source_cod
e',
       'source_state', 'destination_city', 'destination_place',
       'destination_code', 'destination_state', 'corridor',
       'trip_creation_year', 'trip_creation_month', 'trip_creation_day'],
      dtype='object')
```

In [64]:

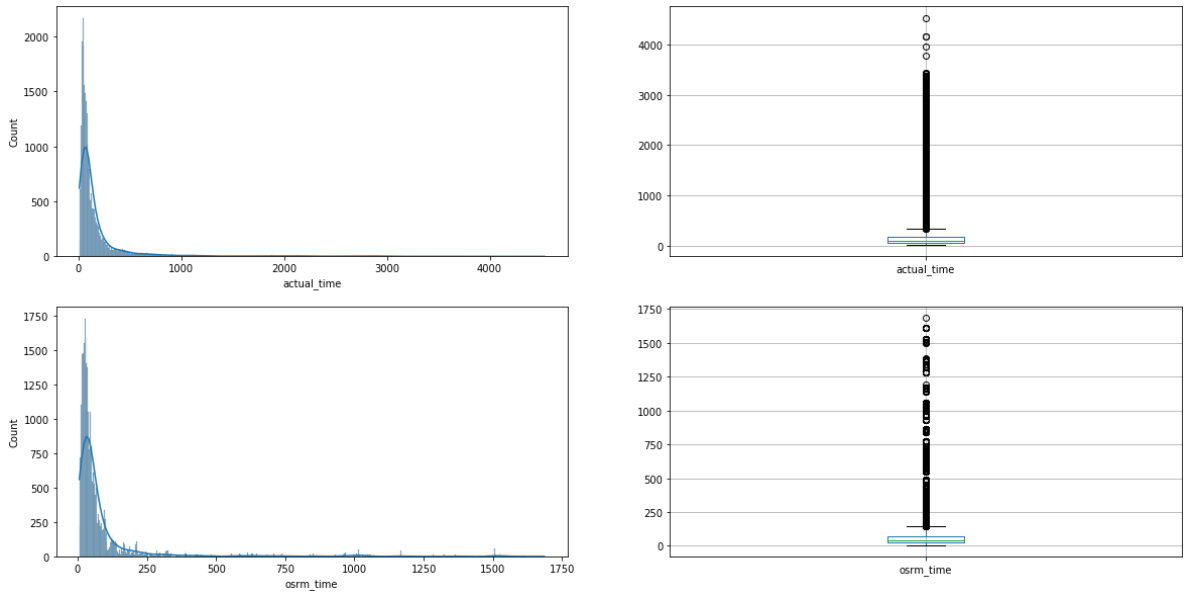
```
#comparing actual time with osrm time to see if they're different

fig, axis = plt.subplots(nrows=2, ncols=2, figsize=(20, 10))
sns.histplot(trip["actual_time"], ax=axis[0,0], kde=True)
trip.boxplot(column='actual_time', ax=axis[0,1])

sns.histplot(trip["osrm_time"], ax=axis[1,0], kde=True)
trip.boxplot(column='osrm_time', ax=axis[1,1])
```

Out[64]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x21628277d30>
```



Both actual time and osrm time are heavily right-skewed with a lot of outliers. We can also observe that actual time taken is much more than the osrm time.
We will further see how to handle the outliers using IQR method

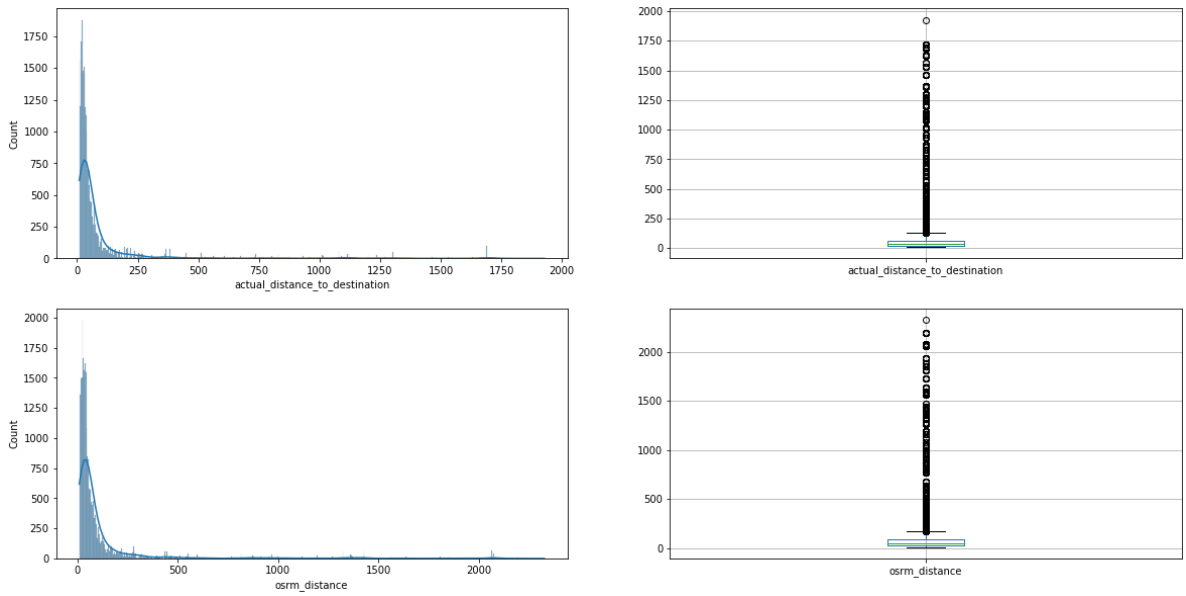
In [65]:

```
fig, axis = plt.subplots(nrows=2, ncols=2, figsize=(20, 10))
sns.histplot(trip["actual_distance_to_destination"], ax=axis[0,0], kde=True)
trip.boxplot(column='actual_distance_to_destination', ax=axis[0,1])

sns.histplot(trip["osrm_distance"], ax=axis[1,0], kde=True)
trip.boxplot(column='osrm_distance', ax=axis[1,1])
```

Out[65]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2162e4ebb70>
```



Both actual distance and osrm distance are heavily right-skewed with a lot of outliers.
We can also observe that actual distance is a little similar to the osrm distance.
We will further see how to handle the outliers using IQR method

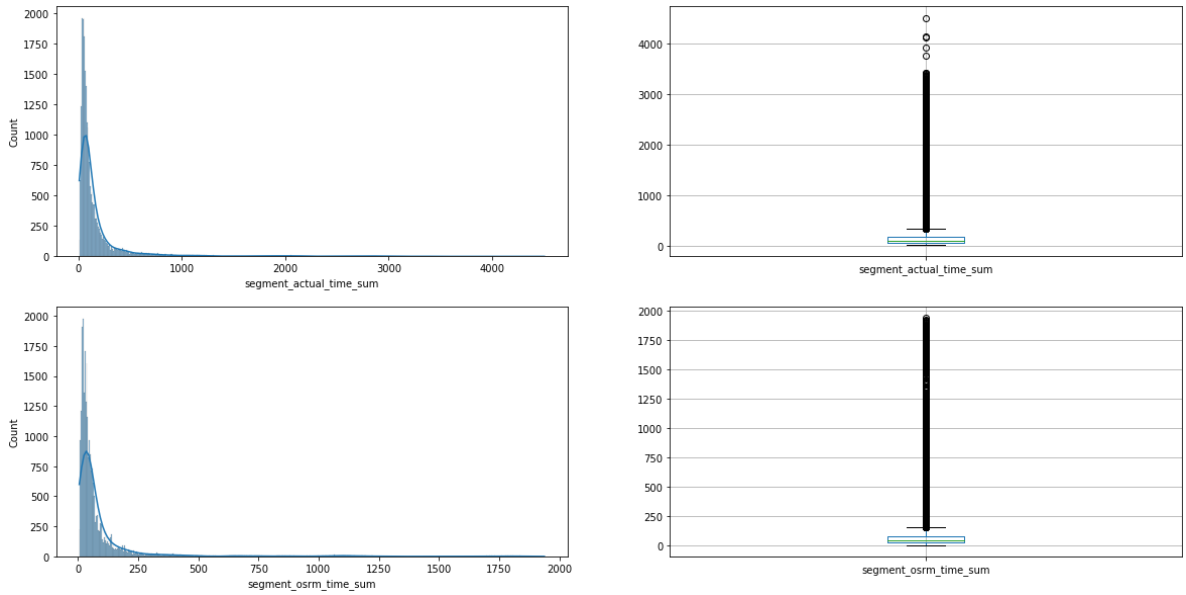
In [66]:

```
fig, axis = plt.subplots(nrows=2, ncols=2, figsize=(20, 10))
sns.histplot(trip["segment_actual_time_sum"], ax=axis[0,0], kde=True)
trip.boxplot(column='segment_actual_time_sum', ax=axis[0,1])

sns.histplot(trip["segment_osrm_time_sum"], ax=axis[1,0], kde=True)
trip.boxplot(column='segment_osrm_time_sum', ax=axis[1,1])
```

Out[66]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x216305aeb00>
```



Both aggregated segment actual time and aggreagated segment osrm time are heavily right-skewed with a lot of outliers. We can also observe that actual segment time taken is much more than the segment osrm time.
We will further see how to handle the outliers using IQR method

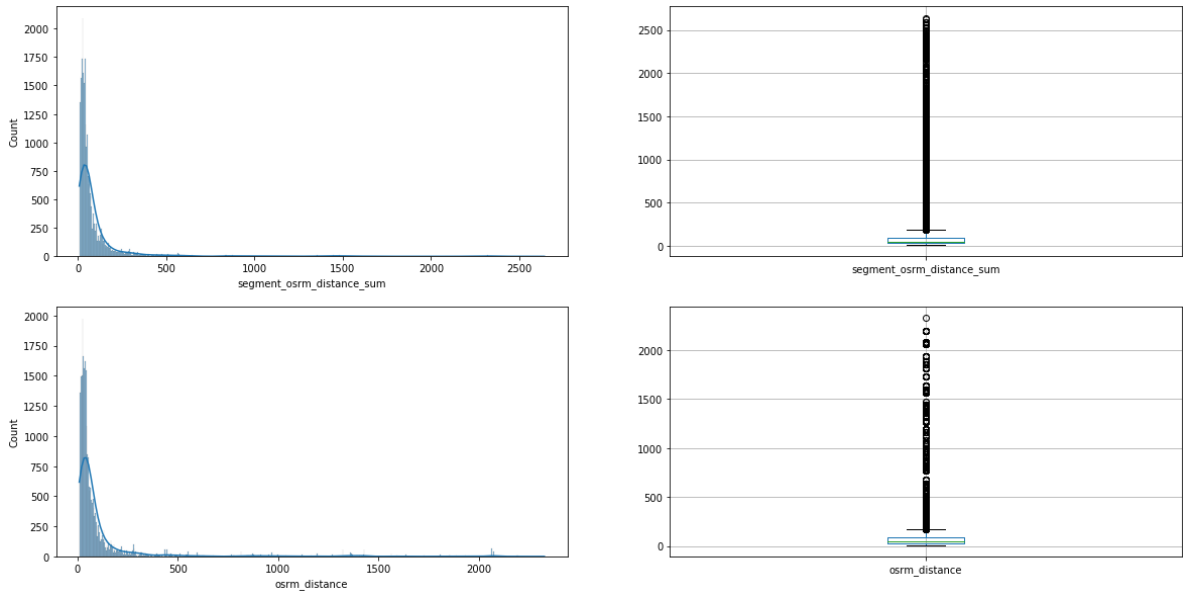
In [67]:

```
fig, axis = plt.subplots(nrows=2, ncols=2, figsize=(20, 10))
sns.histplot(trip["segment_osrm_distance_sum"], ax=axis[0,0], kde=True)
trip.boxplot(column='segment_osrm_distance_sum', ax=axis[0,1])

sns.histplot(trip["osrm_distance"], ax=axis[1,0], kde=True)
trip.boxplot(column='osrm_distance', ax=axis[1,1])
```

Out[67]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2163159a518>
```



Both aggregated segment actual distance and aggreagated osrm distance are heavily right-skewed with a lot of outliers. We can also observe that osrm segment distance is similar to the aggregated osrm distance(as one would expect)
We will further see how to handle the outliers using IQR method

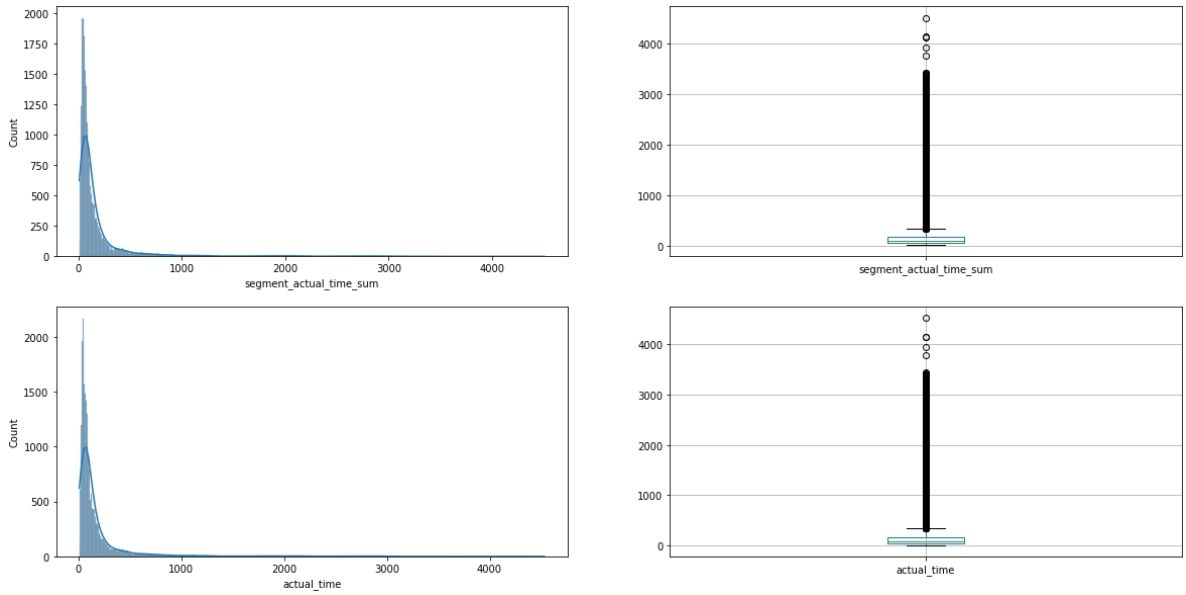
In [68]:

```
fig, axis = plt.subplots(nrows=2, ncols=2, figsize=(20, 10))
sns.histplot(trip["segment_actual_time_sum"], ax=axis[0,0], kde=True)
trip.boxplot(column='segment_actual_time_sum', ax=axis[0,1])

sns.histplot(trip["actual_time"], ax=axis[1,0], kde=True)
trip.boxplot(column='actual_time', ax=axis[1,1])
```

Out[68]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2163371b978>
```



Both aggregated segment actual time and aggreagated actual time are heavily right-skewed with a lot of outliers. We can also observe that both are similar in distribution(as one would expect)
We will further see how to handle the outliers using IQR method

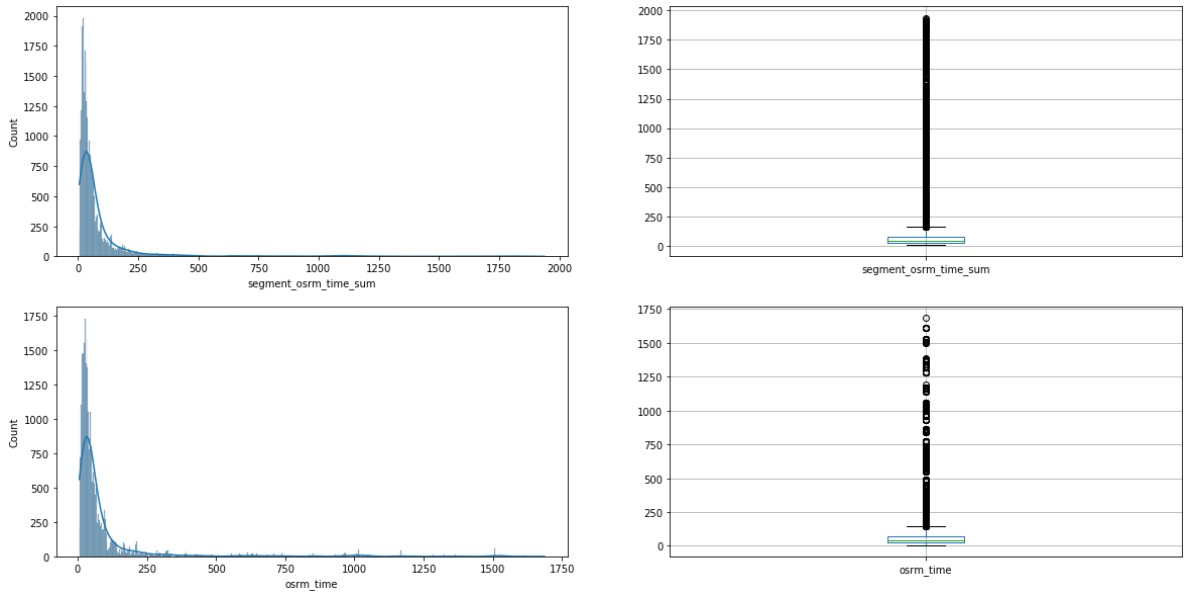
In [69]:

```
fig, axis = plt.subplots(nrows=2, ncols=2, figsize=(20, 10))
sns.histplot(trip["segment_osrm_time_sum"], ax=axis[0,0], kde=True)
trip.boxplot(column='segment_osrm_time_sum', ax=axis[0,1])

sns.histplot(trip["osrm_time"], ax=axis[1,0], kde=True)
trip.boxplot(column='osrm_time', ax=axis[1,1])
```

Out[69]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2163472b860>
```



Both aggregated segment osrm time and aggreagated osrm time are heavily right-skewed with a lot of outliers. We can also observe that both are similar in distribution(as one would expect)
We will further see how to handle the outliers using IQR method

Outliers treatment

In [70]:

```
# We will be using the IQR method to treat the outliers. Using the IQR methos, we will be r
#with mean of the respective columns
def impute_outliers_IQR(df):
    q1=df.quantile(0.25)
    q3=df.quantile(0.75)
    IQR=q3-q1
    upper = df[~(df>(q3+1.5*IQR))].max()
    lower = df[~(df<(q1-1.5*IQR))].min()
    df = np.where(df > upper,df.mean(),np.where(df < lower,df.mean(),df))
    return df
```

In [71]:

```
trip['actual_distance_to_destination'] = impute_outliers_IQR(trip['actual_distance_to_desti
```

In [72]:

```
trip['actual_time'] = impute_outliers_IQR(trip['actual_time'])
```

In [73]:

```
trip['osrm_time'] = impute_outliers_IQR(trip['osrm_time'])
```

In [74]:

```
trip['osrm_distance'] = impute_outliers_IQR(trip['osrm_distance'])
```

In [75]:

```
trip['segment_actual_time_sum'] = impute_outliers_IQR(trip['segment_actual_time_sum'])
```

In [76]:

```
trip['segment_osrm_distance_sum'] = impute_outliers_IQR(trip['segment_osrm_distance_sum'])
```

In [77]:

```
trip['segment_osrm_time_sum'] = impute_outliers_IQR(trip['segment_osrm_time_sum'])
```

Checking relationship between aggregated fields

```
We have already visualized and checked the relationship between different aggregated time
and distance fields.
Next we are going to use hypothesis testing check the relationship between the same
fields and draw conclusions
```

In [78]:

```
# hypothesis testing between actual_time aggregated value and OSRM time aggregated value
```

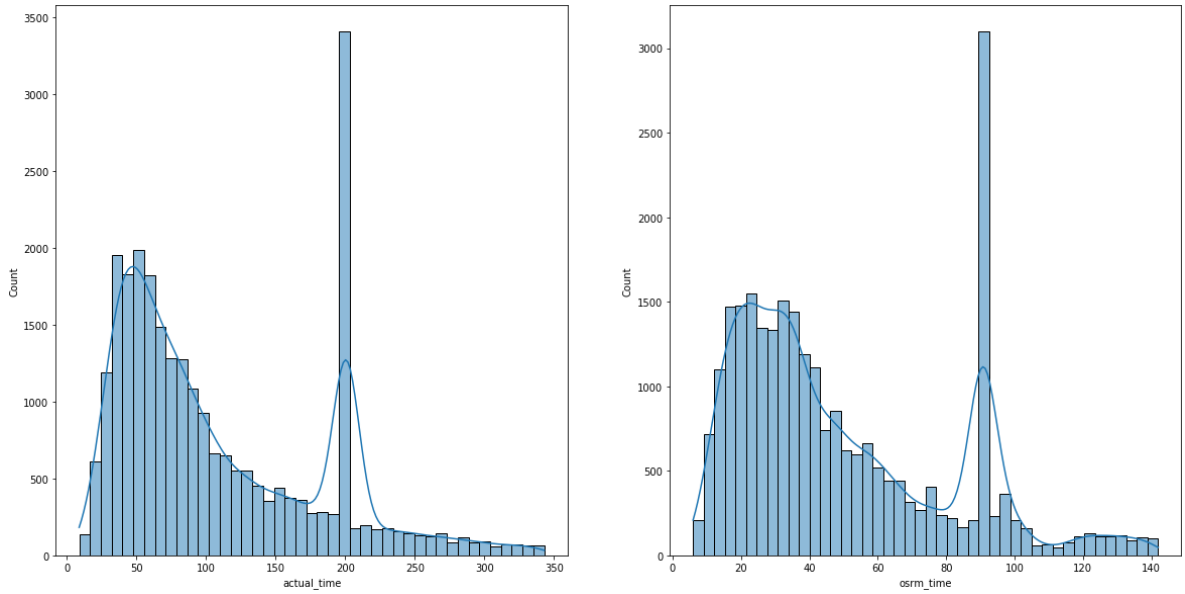
```
Lets visualize the data fields we are going to test to make sure we choose the right test
```

In [79]:

```
fig, axis = plt.subplots(ncols=2, figsize=(20, 10))
sns.histplot(trip["actual_time"], ax=axis[0], kde=True)
sns.histplot(trip["osrm_time"], ax=axis[1], kde=True)
```

Out[79]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2163456bc88>
```



here we can observe that the data is not normally distributed for both actual time aggregated value and osrm time aggregated value. Hence we are going to use Wilcoxon Rank-Sum test for our hypothesis testing

In [80]:

```
# H0 : aggregated actual time is same as aggregated osrm time
# Ha : aggregated actual time is more than the aggregated osrm time
alpha = 0.05 #testing at 95% confidence
test_stat , p_value = ranksums(trip['actual_time'], trip['osrm_time'],alternative='greater'
print("test statistic:",test_stat)
print("p value :",p_value)
if p_value < alpha:
    print("Reject H0")
else:
    print("Fail to reject H0")
```

```
test statistic: 109.89729905559884
p value : 0.0
Reject H0
```

As we can clearly observe from our test, aggregated actual time is more than the aggregated osrm time and the alternate hypothesis stands True and we reject the Null hypothesis

In [81]:

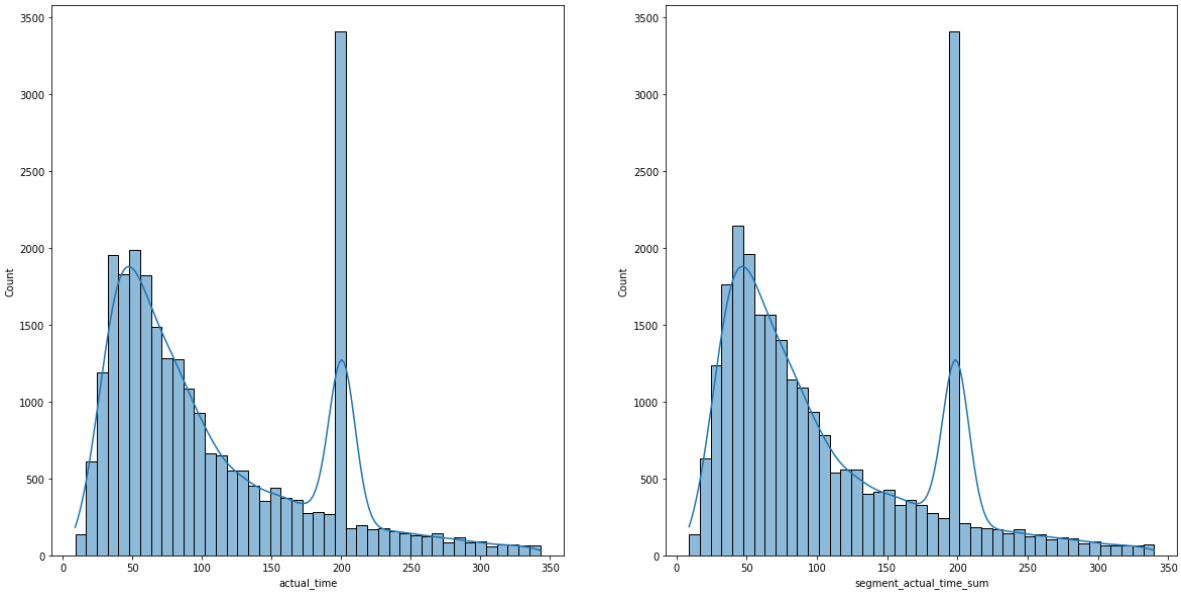
```
# Hypothesis testing between actual time aggregated value and segment actual time aggregate
```

In [82]:

```
fig, axis = plt.subplots(ncols=2, figsize=(20, 10))
sns.histplot(trip["actual_time"], ax=axis[0], kde=True)
sns.histplot(trip["segment_actual_time_sum"], ax=axis[1], kde=True)
```

Out[82]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2163521bf60>
```



here we can observe that the data is not normally distributed we are going to use Wilcoxon Rank-Sum test for our hypothesis testing

In [83]:

```
# H0 : aggregated actual time is same as aggregated segment actual time
# Ha : aggregated actual time is more than the aggregated segment actual time
alpha = 0.05 #testing at 95% confidence
test_stat , p_value = ranksums(trip['actual_time'], trip['segment_actual_time_sum'],alterna
print("test statistic:",test_stat)
print("p value :",p_value)
if p_value < alpha:
    print("Reject H0")
else:
    print("Fail to reject H0")
```

```
test statistic: 4.408578057488138
p value : 5.20257593791673e-06
Reject H0
```

As we can clearly observe from our test, aggregated actual time is more than the aggregated segment actual time and the alternate hypothesis stands True and we reject the Null hypothesis

In [84]:

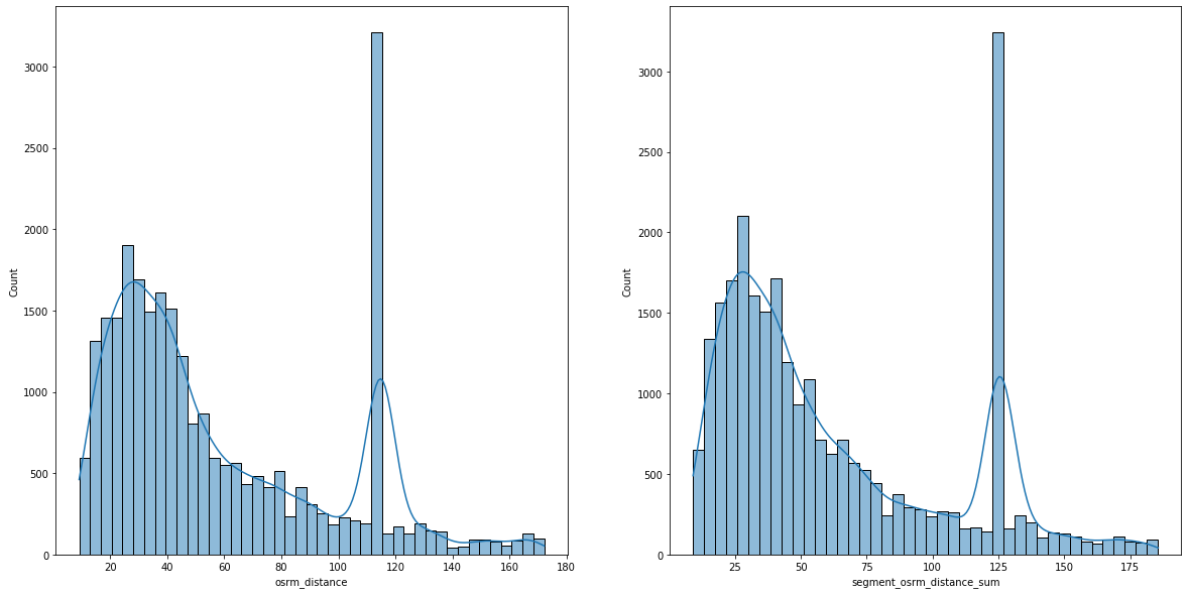
```
# Hypothesis testing between osrm distance aggregated value and segment osrm distance aggre
```

In [85]:

```
fig, axis = plt.subplots(ncols=2, figsize=(20, 10))
sns.histplot(trip["osrm_distance"], ax=axis[0], kde=True)
sns.histplot(trip["segment_osrm_distance_sum"], ax=axis[1], kde=True)
```

Out[85]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x21635b0c860>
```



here we can observe that the data is not normally distributed we are going to use Wilcoxon Rank-Sum test for our hypothesis testing

In [86]:

```
# H0 : aggregated segment osrm distance is same as aggregated osrm distance
# Ha : aggregated segment osrm distance is more than the aggregated osrm distance
alpha = 0.05 #testing at 95% confidence
test_stat , p_value = ranksums(trip['segment_osrm_distance_sum'],trip['osrm_distance'],alte
print("test statistic:",test_stat)
print("p value :",p_value)
if p_value < alpha:
    print("Reject H0")
else:
    print("Fail to reject H0")
```

```
test statistic: 10.276208900025038
p value : 4.510316896570302e-25
Reject H0
```

As we can clearly observe from our test, aggregated segment osrm distance is more than the aggregated osrm distance and the alternate hypothesis stands True and we reject the Null hypothesis

In [87]:

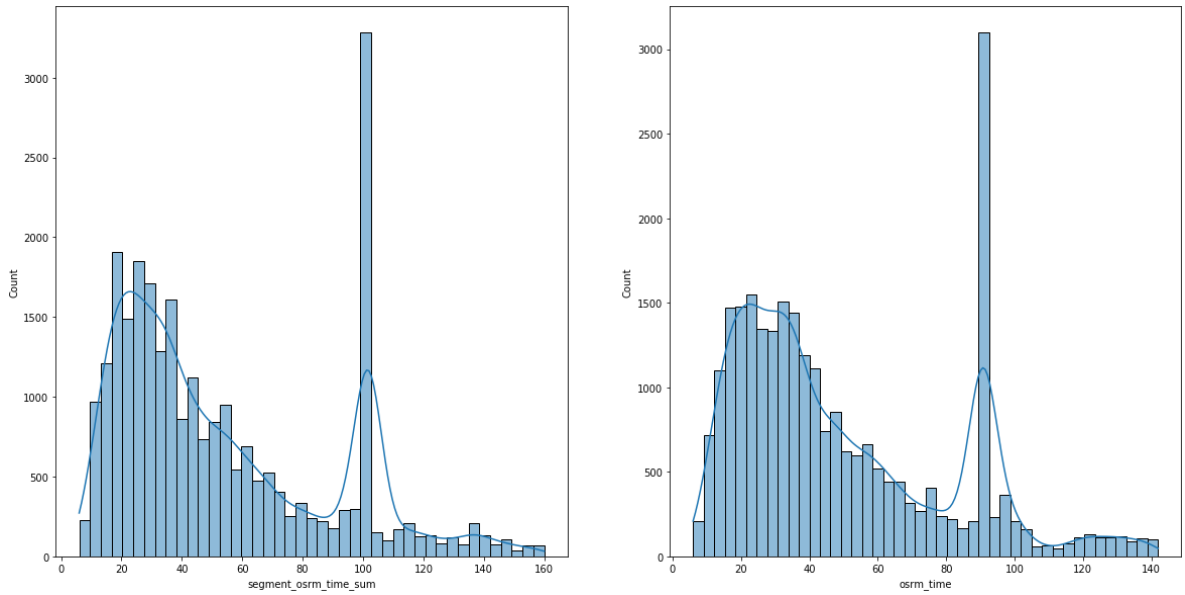
```
# Hypothesis testing between osrm time aggregated value and segment osrm time aggregated va
```

In [88]:

```
fig, axis = plt.subplots(ncols=2, figsize=(20, 10))
sns.histplot(trip["segment_osrm_time_sum"], ax=axis[0], kde=True)
sns.histplot(trip["osrm_time"], ax=axis[1], kde=True)
```

Out[88]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x21635cc6198>
```



here we can observe that the data is not normally distributed we are going to use Wilcoxon Rank-Sum test for our hypothesis testing

In [89]:

```
# H0 : aggregated segment osrm time is same as aggregated osrm time
# Ha : aggregated segment osrm time is more than the aggregated osrm time
alpha = 0.05 #testing at 95% confidence
test_stat , p_value = ranksums(trip['segment_osrm_time_sum'],trip['osrm_time'],alternative=
print("test statistic:",test_stat)
print("p value :",p_value)
if p_value < alpha:
    print("Reject H0")
else:
    print("Fail to reject H0")
```

```
test statistic: 11.212478501454225
p value : 1.7706786165328236e-29
Reject H0
```

As we can clearly observe from our test, aggregated segment osrm time is more than the aggregated osrm time and the alternate hypothesis stands True and we reject the Null hypothesis

In [90]:

```
# Hypothesis testing to Compare the difference between Point a. and start_scan_to_end_scan
```

In [91]:

```
#Compare the difference between Point a. and start_scan_to_end_scan. We can use the 1 sampl
# H0 : the average actual time taken to deliver is the same as osrm time
# Ha : the average actual time taken to deliver is more than the osrm time
alpha = 0.05 #testing at 95% confidence
point_a = np.mean(trip['osrm_time'])
test_stat , p_value = ttest_1samp(trip['start_scan_to_end_scan'], popmean=point_a)
print("test statistic:",test_stat)
print("p value :",p_value)
if p_value < alpha:
    print("Reject H0")
else:
    print("Fail to reject H0")
```

```
test statistic: 91.73431091314349
p value : 0.0
Reject H0
```

As we can clearly observe from our test, the average actual time taken to deliver is more than the osrm time and the alternate hypothesis stands True and we reject the Null hypothesis

Handling categorical values

We are going to use one-hot encoder to make our categorical values more expressive. Many machine learning algorithms cannot handle categorical values, hence this way can be used while sanitizing our data

In [92]:

```
encoder = OneHotEncoder(handle_unknown='ignore')
encoder_df = pd.DataFrame(encoder.fit_transform(trip[['route_type']]).toarray())
```

```
C:\Users\sanke\Anaconda3\lib\site-packages\sklearn\preprocessing\_encoders.p
y:110: DeprecationWarning: `np.int` is a deprecated alias for the builtin `i
nt`. To silence this warning, use `int` by itself. Doing this will not modif
y any behavior and is safe. When replacing `np.int`, you may wish to use e.
g. `np.int64` or `np.int32` to specify the precision. If you wish to review
your current use, check the release note link for additional information.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/d
evdocs/release/1.20.0-notes.html#deprecations (https://numpy.org/devdocs/rel
ease/1.20.0-notes.html#deprecations)
  X_int = np.zeros((n_samples, n_features), dtype=np.int)
C:\Users\sanke\Anaconda3\lib\site-packages\sklearn\preprocessing\_encoders.p
y:111: DeprecationWarning: `np.bool` is a deprecated alias for the builtin `
bool`. To silence this warning, use `bool` by itself. Doing this will not mo
dify any behavior and is safe. If you specifically wanted the numpy scalar t
ype, use `np.bool_` here.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/d
evdocs/release/1.20.0-notes.html#deprecations (https://numpy.org/devdocs/rel
ease/1.20.0-notes.html#deprecations)
  X_mask = np.ones((n_samples, n_features), dtype=np.bool)
```

In [93]:

```
trip = trip.join(encoder_df)
trip.drop('route_type', axis=1, inplace=True)
trip.drop(0, axis=1, inplace=True)
trip.rename(columns = {1:'route_type'}, inplace = True)
trip
```

	destination_code	destination_state	corridor	trip_creation_year	trip_creation_month	trip_creat
	HB	Haryana	Kanpur_Gurgaon	2018	9	
	H6	Uttar Pradesh	Bhopal_Kanpur	2018	9	
	D	Karnataka	Doddablpur_Chikblapur	2018	9	
	D	Karnataka	Tumkur_Doddablpur	2018	9	
	H	Punjab	Gurgaon_Chandigarh	2018	9	

Same can be done to 'data' field which has either type training or test

In [94]:

```
encoder = OneHotEncoder(handle_unknown='ignore')
encoder_df = pd.DataFrame(encoder.fit_transform(trip[['data']]).toarray())
```

```
C:\Users\sanke\Anaconda3\lib\site-packages\sklearn\preprocessing\_encoders.p
y:110: DeprecationWarning: `np.int` is a deprecated alias for the builtin `i
nt`. To silence this warning, use `int` by itself. Doing this will not modif
y any behavior and is safe. When replacing `np.int`, you may wish to use e.
g. `np.int64` or `np.int32` to specify the precision. If you wish to review
your current use, check the release note link for additional information.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/d
evdocs/release/1.20.0-notes.html#deprecations (https://numpy.org/devdocs/rel
ease/1.20.0-notes.html#deprecations)
  X_int = np.zeros((n_samples, n_features), dtype=np.int)
C:\Users\sanke\Anaconda3\lib\site-packages\sklearn\preprocessing\_encoders.p
y:111: DeprecationWarning: `np.bool` is a deprecated alias for the builtin `
bool`. To silence this warning, use `bool` by itself. Doing this will not mo
dify any behavior and is safe. If you specifically wanted the numpy scalar t
ype, use `np.bool_` here.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/d
evdocs/release/1.20.0-notes.html#deprecations (https://numpy.org/devdocs/rel
ease/1.20.0-notes.html#deprecations)
  X_mask = np.ones((n_samples, n_features), dtype=np.bool)
```

In [95]:

```
trip = trip.join(encoder_df)
trip.drop('data', axis=1, inplace=True)
trip.drop(0, axis=1, inplace=True)
trip.rename(columns = {1:'data'}, inplace = True)
trip
```

ation_code	destination_state	corridor	trip_creation_year	trip_creation_month	trip_creation_day
HB	Haryana	Kanpur_Gurgaon	2018	9	12
H6	Uttar Pradesh	Bhopal_Kanpur	2018	9	12
D	Karnataka	Doddablpur_Chikblapur	2018	9	12
D	Karnataka	Tumkur_Doddablpur	2018	9	12
H	Punjab	Gurgaon_Chandigarh	2018	9	12

Column Normalization /Column Standardization

Machine learning algorithms tend to perform better or converge faster when the different features are on a smaller scale. Therefore it is common practice to normalize the data before training machine learning models on it.
Normalization also makes the training process less sensitive to the scale of the features. This results in getting better coefficients after training. This process of making features more suitable for training by rescaling is called feature scaling

In [96]:

```
trip.dtypes
```

Out[96]:

```
trip_creation_time                object
route_schedule_uuid               object
trip_uuid                         object
source_center                     object
source_name                       object
destination_center                object
destination_name                  object
start_scan_to_end_scan           float64
od_time_diff_hour                float64
actual_distance_to_destination   float64
actual_time                      float64
osrm_time                        float64
osrm_distance                    float64
segment_actual_time_sum          float64
segment_osrm_distance_sum        float64
segment_osrm_time_sum            float64
source_city                       object
source_place                      object
source_code                       object
source_state                      object
destination_city                  object
destination_place                 object
destination_code                  object
destination_state                 object
corridor                          object
trip_creation_year                 int64
trip_creation_month                int64
trip_creation_day                  int64
route_type                       float64
data                             float64
dtype: object
```

In [97]:

```
trip1 = trip[['start_scan_to_end_scan','od_time_diff_hour','actual_distance_to_destination'
scaler = preprocessing.MinMaxScaler()
names = trip1.columns
d = scaler.fit_transform(trip1)
scaled_df = pd.DataFrame(d, columns=names)
trip[['start_scan_to_end_scan','od_time_diff_hour','actual_distance_to_destination','actual
trip.head()
```

Out[97]:

nation_name	start_scan_to_end_scan	od_time_diff_hour	actual_distance_to_destination	...	destin
_Bilaspur_HB (Haryana)	0.157400	0.157391	0.679328	...	
_Central_H_6 Jttar Pradesh)	0.124270	0.124247	0.679328	...	
ır_ShntiSgr_D (Karnataka)	0.004824	0.004840	0.127380	...	
_ChikaDPP_D (Karnataka)	0.012947	0.012957	0.321990	...	[
.Mehmdpur_H (Punjab)	0.103326	0.103320	0.679328	...	(

Business insights

In [98]:

```
# Checking from which State most orders are coming from
trip['source_state'].value_counts().sort_values(ascending=False).head(1)
```

Out[98]:

```
Maharashtra    3565
Name: source_state, dtype: int64
```

In [99]:

```
#Checking to which state most orders are going to
trip['destination_state'].value_counts().sort_values(ascending=False).head(1)
```

Out[99]:

```
Karnataka    3505
Name: destination_state, dtype: int64
```

In [100]:

```
# Checking from which city most orders are coming from
trip['source_city'].value_counts().sort_values(ascending=False).head(1)
```

Out[100]:

```
Gurgaon    1141
Name: source_city, dtype: int64
```

In [101]:

```
# Checking to which city most orders are going to
trip['destination_city'].value_counts().sort_values(ascending=False).head(1)
```

Out[101]:

```
Bengaluru    1180
Name: destination_city, dtype: int64
```

In [102]:

```
# Checking which is the busiest corridor
trip['corridor'].value_counts().sort_values(ascending=False).head(1)
```

Out[102]:

```
Bengaluru_Bengaluru    565
Name: corridor, dtype: int64
```

In [109]:

```
trip2 = trip
trip2[['start_scan_to_end_scan','od_time_diff_hour','actual_distance_to_destination','actua
```

In [118]:

```
# Checking the average time taken between the busiest corridor
df_corridor = trip2.loc[trip2["corridor"] == 'Bengaluru_Bengaluru']
df_corridor["actual_time"].mean()
```

Out[118]:

```
79.2778761061947
```

In [119]:

```
# Checking the average distance between the busiest corridor
df_corridor_distance = trip2.loc[trip2["corridor"] == 'Bengaluru_Bengaluru']
df_corridor_distance["actual_distance_to_destination"].mean()
```

Out[119]:

```
29.075026481569214
```

In [122]:

```
# Checking which month had the highest no. of trips
trip['trip_creation_month'].value_counts().sort_values(ascending=False).head()
```

Out[122]:

```
9     23159
10     3209
Name: trip_creation_month, dtype: int64
```

In [124]:

```
# Checking which month had the highest no. of trips
trip['trip_creation_day'].value_counts().sort_values(ascending=False)
```

Out[124]:

```
18    1373
21    1361
25    1330
20    1322
13    1320
14    1296
15    1293
17    1293
12    1290
22    1280
26    1274
24    1177
27    1169
19    1155
3     1128
16    1098
23    1085
28    1078
1     1050
29    1034
2     1031
30     931
Name: trip_creation_day, dtype: int64
```

In [ ]:

```

```