Delhivery :Feature Engineering - Feature Engineering

About Delhivery

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

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

How can you help here?

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

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

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

Column Profiling:

data - tells whether the data is testing or training data

trip_creation_time – Timestamp of trip creation

route_schedule_uuid – Unique Id for a particular route schedule

route_type – Transportation type

FTL – Full Truck Load: FTL shipments get to the destination sooner, as the truck is making no other pickups or drop-offs along the way

Carting: Handling system consisting of small vehicles (carts)

trip_uuid - Unique ID given to a particular trip (A trip may include different source and destination centers)

source_center - Source ID of trip origin

source_name - Source Name of trip origin

destination_cente – Destination ID

destination_name – Destination Name

od_start_time – Trip start time

od_end_time – Trip end time

start_scan_to_end_scan – Time taken to deliver from source to destination

is_cutoff – Unknown field

cutoff_factor – Unknown field

cutoff_timestamp – Unknown field

actual_distance_to_destination – Distance in Kms between source and destination warehouse

actual_time – Actual time taken to complete the delivery (Cumulative)

osrm_time – An open-source routing engine time calculator which computes the shortest path between points in a given map (Includes usual traffic, distance through major and minor roads) and gives the time (Cumulative)

osrm_distance – An open-source routing engine which computes the shortest path between points in a given map (Includes usual traffic, distance through major and minor roads) (Cumulative)

factor – Unknown field

segment_actual_time – This is a segment time. Time taken by the subset of the package delivery

segment_osrm_time – This is the OSRM segment time. Time taken by the subset of the package delivery

segment_osrm_distance – This is the OSRM distance. Distance covered by subset of the package delivery

segment_factor – Unknown field

Concept Used:

Feature Creation Relationship between Features Column Normalization /Column Standardization Handling categorical values Missing values - Outlier treatment / Types of outliers

```
!wget "https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/551/original/delhivery_data.csv?1642751181"
```

```
--2023-04-23 15:01:17--  https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/551/original/delhivery_data.csv?1642751
Resolving d2beiqkhq929f0.cloudfront.net (d2beiqkhq929f0.cloudfront.net)... 54.230.209.222, 54.230.209.118, 54.230.209.194, ...
```

```
Connecting to d2beiqkhq929f0.cloudfront.net (d2beiqkhq929f0.cloudfront.net)|54.230.209.222|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 55617130 (53M) [text/plain]
Saving to: ‘delhivery_data.csv?1642751181.4’

delhivery_data.csv? 100%[===================>]  53.04M  86.3MB/s    in 0.6s

2023-04-23 15:01:18 (86.3 MB/s) - ‘delhivery_data.csv?1642751181.4’ saved [55617130/55617130]
```

```
# Importing required libraries -
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from scipy import stats
from scipy.stats import ttest_ind # T-test for independent samples
from scipy.stats import shapiro # Shapiro-Wilk’s test for Normality
from scipy.stats import levene # Levene's test for Equality of Variance
from scipy.stats import f_oneway # One-way ANOVA
from scipy.stats import chi2_contingency # Chi-square test of independence

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

```
df = pd.read_csv("delhivery_data.csv?1642751181")
```

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

	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	IND3881
1	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND3881
2	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND3881
3	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND3881
4	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND3881

5 rows × 24 columns

```
df.info()
```

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

```
 20  segment_actual_time             144867 non-null  float64
 21  segment_osrm_time               144867 non-null  float64
 22  segment_osrm_distance           144867 non-null  float64
 23  segment_factor                  144867 non-null  float64
dtypes: bool(1), float64(10), int64(1), object(12)
memory usage: 25.6+ MB
```

```
df.isna().sum()
```

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

```
df.isna().sum()/len(df)*100
```

```
data                              0.000000
trip_creation_time                0.000000
route_schedule_uuid               0.000000
route_type                        0.000000
trip_uuid                         0.000000
source_center                     0.000000
source_name                       0.202254
destination_center                0.000000
destination_name                  0.180165
od_start_time                     0.000000
od_end_time                       0.000000
start_scan_to_end_scan            0.000000
is_cutoff                         0.000000
cutoff_factor                     0.000000
cutoff_timestamp                  0.000000
actual_distance_to_destination    0.000000
actual_time                       0.000000
osrm_time                         0.000000
osrm_distance                     0.000000
factor                            0.000000
segment_actual_time               0.000000
segment_osrm_time                 0.000000
segment_osrm_distance             0.000000
segment_factor                    0.000000
dtype: float64
```

1) Here the percentage of missing values of very low(0.2/0.1 percent) compared to actual.

So dropping the missing data.

```
df = df.dropna(how='any')
df = df.reset_index(drop=True)   ## As the index values are not in order reset it
df
```

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	so
0	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IN
1	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IN
2	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IN
3	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IN
4	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IN
...	...	...	...	...	...	
144311	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	IN
144312	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-67b86d5...	Carting	trip-153746066843555182	IN
144313	training	2018-09-20 16:24:28.436231	thanos::sroute:f0569d2f-4e20-4c31-8542-	Carting	trip-153746066843555182	IN

```
## Converting object type to datatime type for required fields

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

144315	training		4e20-4c31-8542-	Carting		IN

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 144316 entries, 0 to 144315
Data columns (total 24 columns):
 #   Column                          Non-Null Count   Dtype
---  ------                          --------------   -----
 0   data                            144316 non-null  object
 1   trip_creation_time              144316 non-null  datetime64[ns]
 2   route_schedule_uuid             144316 non-null  object
 3   route_type                      144316 non-null  object
 4   trip_uuid                       144316 non-null  object
 5   source_center                   144316 non-null  object
 6   source_name                     144316 non-null  object
 7   destination_center              144316 non-null  object
 8   destination_name                144316 non-null  object
 9   od_start_time                   144316 non-null  datetime64[ns]
 10  od_end_time                     144316 non-null  datetime64[ns]
 11  start_scan_to_end_scan          144316 non-null  float64
 12  is_cutoff                       144316 non-null  bool
 13  cutoff_factor                   144316 non-null  int64
 14  cutoff_timestamp                144316 non-null  object
 15  actual_distance_to_destination  144316 non-null  float64
 16  actual_time                     144316 non-null  float64
 17  osrm_time                       144316 non-null  float64
 18  osrm_distance                   144316 non-null  float64
 19  factor                          144316 non-null  float64
 20  segment_actual_time             144316 non-null  float64
 21  segment_osrm_time               144316 non-null  float64
 22  segment_osrm_distance           144316 non-null  float64
 23  segment_factor                  144316 non-null  float64
dtypes: bool(1), datetime64[ns](3), float64(10), int64(1), object(9)
memory usage: 25.5+ MB
```

Grouping / Merging the data for each Trip

```
df['grp_trip_source_dest'] = df['trip_uuid'] + df['source_center'] + df['destination_center']

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

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

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

	segment_actual_time_sum	segment_osrm_time_sum	segment_osrm_distance_sum
0	14.0	11.0	11.9653
1	24.0	20.0	21.7243
2	40.0	27.0	32.5395
3	61.0	39.0	45.5619
4	67.0	44.0	49.4772
...	...	...	...
144311	92.0	94.0	65.3487
144312	118.0	115.0	82.7212
144313	138.0	149.0	103.4265
144314	155.0	176.0	122.3150
144315	423.0	185.0	131.1238

144316 rows × 3 columns

```
## df.groupby("grp_trip_source_dest").first()  --> This gives all the first values of grouped rows but some need first and some need last
```

```
df[['grp_trip_source_dest','start_scan_to_end_scan','actual_distance_to_destination','actual_time','osrm_time','osrm_distance','segment_a
```

	grp_trip_source_dest	start_scan_to_end_scan	actual_distance_to_
0	trip-153741093647649320IND388121AAAIND388620AAB	86.0	
1	trip-153741093647649320IND388121AAAIND388620AAB	86.0	
2	trip-153741093647649320IND388121AAAIND388620AAB	86.0	
3	trip-153741093647649320IND388121AAAIND388620AAB	86.0	
4	trip-153741093647649320IND388121AAAIND388620AAB	86.0	
5	trip-153741093647649320IND388620AABIND388320AAA	109.0	
6	trip-153741093647649320IND388620AABIND388320AAA	109.0	
7	trip-153741093647649320IND388620AABIND388320AAA	109.0	
8	trip-153741093647649320IND388620AABIND388320AAA	109.0	
9	trip-153741093647649320IND388620AABIND388320AAA	109.0	

```
row_values_f_l = {
'data' : 'first',
'trip_creation_time' : 'first',
'route_schedule_uuid' : 'first',
'route_type' : 'first',
'trip_uuid' : 'first',
'source_center' : 'first',
'source_name' : 'first',
'destination_center' : 'first',
'destination_name' : 'first',
'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_time_sum' : 'last',
```

```
    'segment_osrm_distance_sum' : 'last'
}

sub_grp = df.groupby('grp_trip_source_dest').agg(row_values_f_l).reset_index()

sub_grp = sub_grp.sort_values(by=['grp_trip_source_dest','od_end_time'], ascending=True).reset_index(drop=True)
sub_grp
```

	grp_trip_source_dest	data	trip_creation_time	route_sch
0	trip-153671041653548748IND209304AAAIND000000ACB	training	2018-09-12 00:00:16.535741	thanos::srout a29t
1	trip-153671041653548748IND462022AAAIND209304AAA	training	2018-09-12 00:00:16.535741	thanos::srout a29t
2	trip-153671042288605164IND561203AABIND562101AAA	training	2018-09-12 00:00:22.886430	thanos::srout bb0b
3	trip-153671042288605164IND572101AAAIND561203AAB	training	2018-09-12 00:00:22.886430	thanos::srout bb0b
4	trip-153671043369099517IND000000ACBIND160002AAC	training	2018-09-12 00:00:33.691250	thanos::srout 7641
...	...	...	...	
26217	trip-153861115439069069IND628204AAAIND627657AAA	test	2018-10-03 23:59:14.390954	thanos::srou 848(
26218	trip-153861115439069069IND628613AAAIND627005AAA	test	2018-10-03 23:59:14.390954	thanos::srou 848(
26219	trip-153861115439069069IND628801AAAIND628204AAA	test	2018-10-03 23:59:14.390954	thanos::srou 848(
26220	trip-153861118270144424IND583119AAAIND583101AAA	test	2018-10-03 23:59:42.701692	thanos::srout 6d1
26221	trip-153861118270144424IND583201AAAIND583119AAA	test	2018-10-03 23:59:42.701692	thanos::srout 6d1

26222 rows × 20 columns

Calculate the time taken between od_start_time and od_end_time and keep it as a feature. Drop the original columns, if required

Now get the trp duration:

od_trip_duration_hr = (od_end_time - od_start_time).dt.total_seconds()/(60)

```
sub_grp['od_trip_duration_hr'] = (sub_grp['od_end_time']-sub_grp['od_start_time']).dt.total_seconds()/(60)

sub_grp.drop(['od_end_time','od_start_time'], axis=1)

sub_grp['od_trip_duration_hr']
```

```
0        1260.604421
1         999.505379
2          58.832388
3         122.779486
4         834.638929
            ...
26217      62.115193
26218      91.087797
26219      44.174403
26220     287.474007
26221      66.933565
Name: od_trip_duration_hr, Length: 26222, dtype: float64
```

```
sub_grp[sub_grp['trip_uuid'] == 'trip-153741093647649320']
```

	grp_trip_source_dest	data	trip_creation_time	route_sche
10370	trip-153741093647649320IND388121AAAIND388620AAB	training	2018-09-20 02:35:36.476840	thanos::sroute b351-
10371	trip-153741093647649320IND388620AABIND388320AAA	training	2018-09-20 02:35:36.476840	thanos::sroute b351-

2 rows × 21 columns

Now as we can see from above specific trip we have 2 rows specified and we have to sum it up to source start to destination end and middle hub should can exclude.

So now we have to get 1 values for particular trip

```
final_row_values_f_l = {
'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_trip_duration_hr':'sum',
'start_scan_to_end_scan' : 'sum',
'actual_distance_to_destination' : 'sum',
'actual_time' : 'sum',
'osrm_time' : 'sum',
'osrm_distance' : 'sum',
'segment_actual_time_sum' : 'sum',
'segment_osrm_time_sum' : 'sum',
'segment_osrm_distance_sum' : 'sum'
}
```

```
data = sub_grp.groupby('trip_uuid').agg(final_row_values_f_l).reset_index(drop=True)
data
```

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

```
data[['actual_time', 'segment_actual_time_sum']]
```

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

14787 rows × 2 columns

Build some features to prepare the data for actual analysis.

Build some features to prepare the data for actual analysis. Extract features from the below fields:

Destination Name: Split and extract features out of destination. City-place-code (State)

Source Name: Split and extract features out of destination. City-place-code (State)

```
def state(x):
  state = x.split('(')[1]

  return state[:-1]

data['dest_state'] = data['destination_name'].apply(lambda x:state(x))
data['source_state'] = data['source_name'].apply(lambda x:state(x))


def city(x):
  city = x.split('_')

  if len(city) <=1:
        return x.split(' ')[0]


  return city[0]

data['dest_city'] = data['destination_name'].apply(lambda x:city(x))
data['source_city'] = data['source_name'].apply(lambda x:city(x))


def Place(x):
  Place = x.split('_')
  if len(Place) ==2:
    return Place[0]
```

```
  if len(Place) >=3:
    return Place[1]

  if len(Place) <=1:
        return x.split(' ')[0]

data['dest_Place'] = data['destination_name'].apply(lambda x:Place(x))
data['source_Place'] = data['source_name'].apply(lambda x:Place(x))


def code(x):
  x = x.split('(')[0]
  code = x.split('_')
  if len(code) ==2:
    return code[1]

  if len(code) ==3:
    return code[2]

  if len(code) <=1:
        return x.split(' ')[-1]

data['dest_code'] = data['destination_name'].apply(lambda x:code(x))
data['source_code'] = data['source_name'].apply(lambda x:code(x))


data[['source_state','dest_state','source_city','dest_city','source_Place','dest_Place','source_code','dest_code']]
```

	source_state	dest_state	source_city	dest_city	source_Place	dest_Place	source_cc
0	Uttar Pradesh	Uttar Pradesh	Kanpur	Kanpur	Central	Central	Nc
1	Karnataka	Karnataka	Doddablpur	Doddablpur	ChikaDPP	ChikaDPP	
2	Haryana	Haryana	Gurgaon	Gurgaon	Bilaspur	Bilaspur	I
3	Maharashtra	Maharashtra	Mumbai	Mumbai	Mumbai	MiraRd	
4	Karnataka	Karnataka	Bellary	Sandur	Bellary	WrdN1DPP	
...	...	...	...	...	...	...	
14782	Punjab	Punjab	Chandigarh	Chandigarh	Mehmdpur	Mehmdpur	
14783	Haryana	Haryana	FBD	Faridabad	Balabhgarh	Blbgarh	DI
14784	Uttar Pradesh	Uttar Pradesh	Kanpur	Kanpur	GovndNgr	GovndNgr	I
14785	Tamil Nadu	Tamil Nadu	Tirunelveli	Tirchchndr	VdkkuSrt	Shnmgprm	
14786	Karnataka	Karnataka	Sandur	Sandur	WrdN1DPP	WrdN1DPP	

```
data['source_state'].value_counts()
```

```
Maharashtra               2308
Karnataka                 2025
Haryana                   1365
Tamil Nadu                1032
Telangana                  701
Delhi                      658
Gujarat                    656
Uttar Pradesh              619
West Bengal                551
Punjab                     472
Rajasthan                  431
Andhra Pradesh             378
Bihar                      267
Kerala                     261
Madhya Pradesh             238
Assam                      220
Jharkhand                  123
Orissa                      94
Chandigarh                  93
Uttarakhand                 93
Chhattisgarh                42
Goa                         34
Jammu & Kashmir             16
Dadra and Nagar Haveli      15
Pondicherry                 12
Himachal Pradesh            12
Nagaland                     4
Arunachal Pradesh            3
Name: source_state, dtype: int64
```

```
data['dest_state'].value_counts()
```

```
Maharashtra               2285
Karnataka                 2070
Haryana                   1333
Tamil Nadu                1040
Telangana                  682
Gujarat                    653
Uttar Pradesh              620
Delhi                      579
West Bengal                559
Punjab                     549
Rajasthan                  477
Andhra Pradesh             376
Bihar                      267
Madhya Pradesh             255
Kerala                     250
Assam                      193
Jharkhand                  123
Uttarakhand                 92
Orissa                      89
Chandigarh                  65
Chhattisgarh                42
Goa                         31
Himachal Pradesh            25
Arunachal Pradesh           22
Dadra and Nagar Haveli      17
Jammu & Kashmir             16
Meghalaya                    8
Mizoram                      2
Nagaland                     1
Tripura                      1
Daman & Diu                  1
Name: dest_state, dtype: int64
```

```
data['source_Place'].value_counts()
```

```
Central     728
Bilaspur    691
Nelmngla    544
Mankoli     540
Bomsndra    440
           ...
Ymunpurm      1
Greenmkt      1
ShbdnDPP      1
ITICollg      1
WrdN1DPP      1
Name: source_Place, Length: 748, dtype: int64
```

```
data['dest_Place'].value_counts()
```

```
Central     673
Bilaspur    597
Nelmngla    386
Mankoli     377
Bomsndra    331
           ...
KetyDPP       1
MahmurGj      1
Mainrd        1
Khenewa       1
VrdhriRD      1
Name: dest_Place, Length: 847, dtype: int64
```

Trip_creation_time: Extract features like month, year and day etc

```
data['year']= data['trip_creation_time'].dt.year
data['month'] = data['trip_creation_time'].dt.month
data['day']= data['trip_creation_time'].dt.day
data['hour'] = data['trip_creation_time'].dt.hour
```

```
data
```

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	so
0	training	2018-09-12 00:00:16.535741	thanos::sroute:d7c989ba-a29b-4a0b-b2f4-288cdc6...	FTL	trip-153671041653548748	INI
1	training	2018-09-12 00:00:22.886430	thanos::sroute:3a1b0ab2-bb0b-4c53-8c59-eb2a2c0...	Carting	trip-153671042288605164	INI
2	training	2018-09-12 00:00:33.691250	thanos::sroute:de5e208e-7641-45e6-8100-4d9fb1e...	FTL	trip-153671043369099517	INI
3	training	2018-09-12 00:01:00.113710	thanos::sroute:f0176492-a679-4597-8332-bbd1c7f...	Carting	trip-153671046011330457	INI
4	training	2018-09-12 00:02:09.740725	thanos::sroute:d9f07b12-65e0-4f3b-bec8-df06134...	FTL	trip-153671052974046625	INI
...	...	...	...	...	...	
14782	test	2018-10-03 23:55:56.258533	thanos::sroute:8a120994-f577-4491-9e4b-b7e4a14...	Carting	trip-153861095625827784	INI
14783	test	2018-10-03 23:57:23.863155	thanos::sroute:b30e1ec3-3bfa-4bd2-a7fb-3b75769...	Carting	trip-153861104386292051	INI
14784	test	2018-10-03 23:57:44.429324	thanos::sroute:5609c268-e436-4e0a-8180-3db4a74...	Carting	trip-153861106442901555	INI
14785	test	2018-10-03 23:59:14.390954	thanos::sroute:c5f2ba2c-8486-4940-8af6-d1d2a6a...	Carting	trip-153861115439069069	INI
14786	test	2018-10-03 23:59:42.701692	thanos::sroute:412fea14-6d1f-4222-8a5f-	FTL	trip-153861118270144424	IN

```
data['month'].value_counts()
```

```
9     11172
10     1551
Name: month, dtype: int64
```

```
data['hour'].value_counts()
```

```
23    990
22    969
20    933
0     899
21    755
19    677
1     637
2     607
18    587
3     574
4     554
6     551
17    512
16    477
7     441
5     407
15    406
14    310
8     265
9     260
13    254
12    233
11    218
10    207
Name: hour, dtype: int64
```

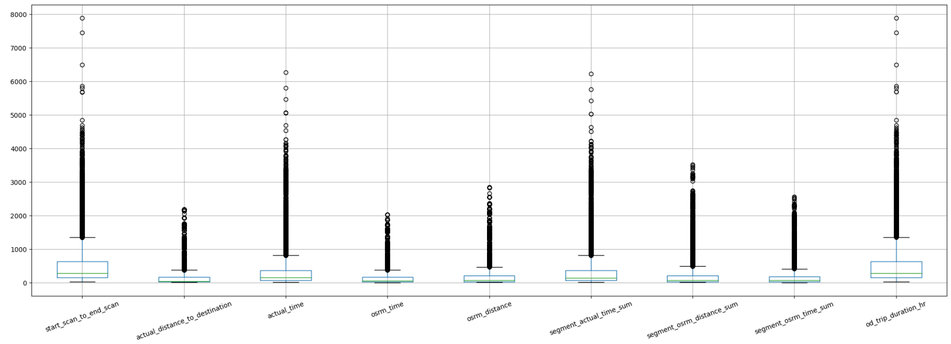
Find outliers in the numerical variables (you might find outliers in almost all the variables), and check it using visual analysis

```
num_values = ['start_scan_to_end_scan','actual_distance_to_destination','actual_time','osrm_time','osrm_distance','segment_actual_time_su
```

```
            'segment_osrm_time_sum', 'od_trip_duration_hr']
```

```
data[num_values].boxplot(rot=20, figsize=(25,8))
```

```
<Axes: >
```



Handle the outliers using the IQR method.

```
Quat1 = data[num_values].quantile(0.25)
Quat3 = data[num_values].quantile(0.75)

IQR = Quat3 - Quat1
IQR
```

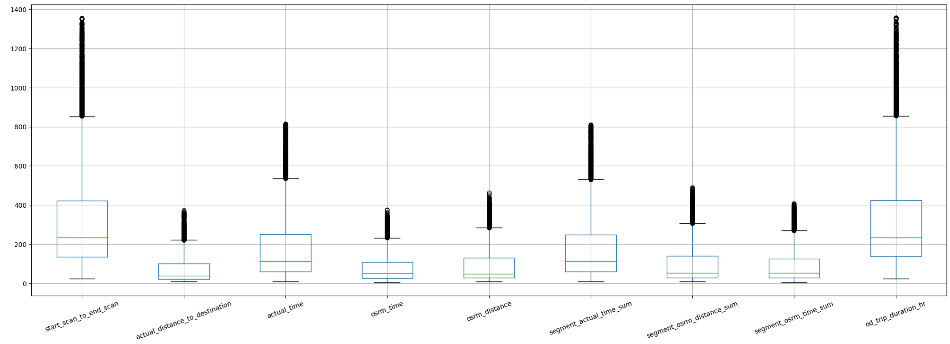
```
start_scan_to_end_scan            483.000000
actual_distance_to_destination    140.814159
actual_time                       300.000000
osrm_time                         139.000000
osrm_distance                     175.887300
segment_actual_time_sum           298.000000
segment_osrm_distance_sum 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

```
data = data[-((data[num_values] < (Quat1 - 1.5 * IQR)) | (data[num_values] > (Quat3 + 1.5 * IQR))).any(axis=1)]
data.reset_index(drop=True)
```

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	so
0	training	2018-09-12 00:00:22.886430	thanos::sroute:3a1b0ab2-bb0b-4c53-8c59-eb2a2c0...	Carting	trip-153671042288605164	INI
1	training	2018-09-12 00:01:00.113710	thanos::sroute:f0176492-a679-4597-8332-bbd1c7f...	Carting	trip-153671046011330457	INI
2	training	2018-09-12 00:02:09.740725	thanos::sroute:d9f07b12-65e0-4f3b-bec8-df06134...	FTL	trip-153671052974046625	INI
3	training	2018-09-12 00:02:34.161600	thanos::sroute:9bf03170-d0a2-4a3f-aa4d-9aaab3d...	Carting	trip-153671055416136166	INI
4	training	2018-09-12 00:04:22.011653	thanos::sroute:a97698cc-846e-41a7-916b-88b1741...	Carting	trip-153671066201138152	INI
...	...	...	...	...	...	
12718	test	2018-10-03 23:55:56.258533	thanos::sroute:8a120994-f577-4491-9e4b-b7e4a14...	Carting	trip-153861095625827784	INI
12719	test	2018-10-03 23:57:23.863155	thanos::sroute:b30e1ec3-3bfa-4bd2-a7fb-3b75769...	Carting	trip-153861104386292051	INI
12720	test	2018-10-03 23:57:44.429324	thanos::sroute:5609c268-e436-4e0a-8180-3db4a74...	Carting	trip-153861106442901555	INI

```
data[num_values].boxplot(rot=20, figsize=(25,8))
```

```
<Axes: >
```



In-depth analysis and feature engineering:

Do hypothesis testing/ visual analysis between actual_time aggregated value and OSRM time aggregated value (aggregated values are the values you'll get after merging the rows on the basis of trip_uuid)

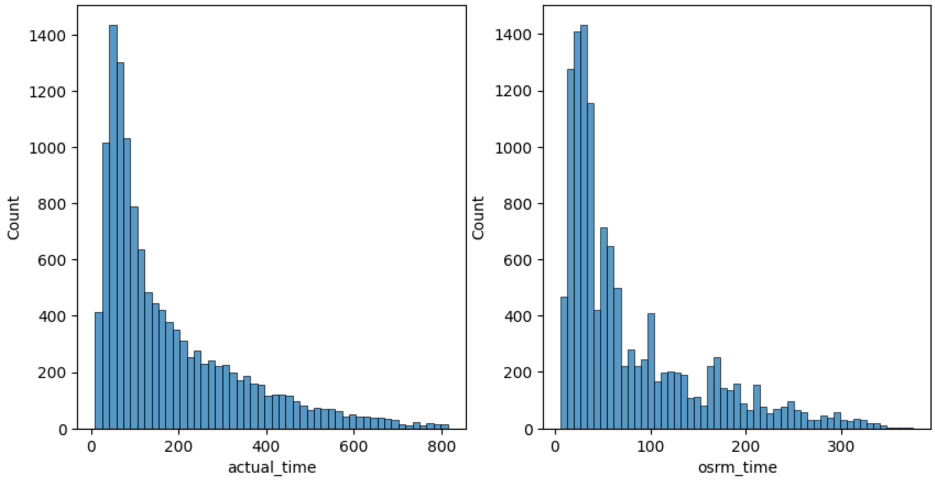
```
data['actual_time'].mean()
```

```
177.4527234142891
```

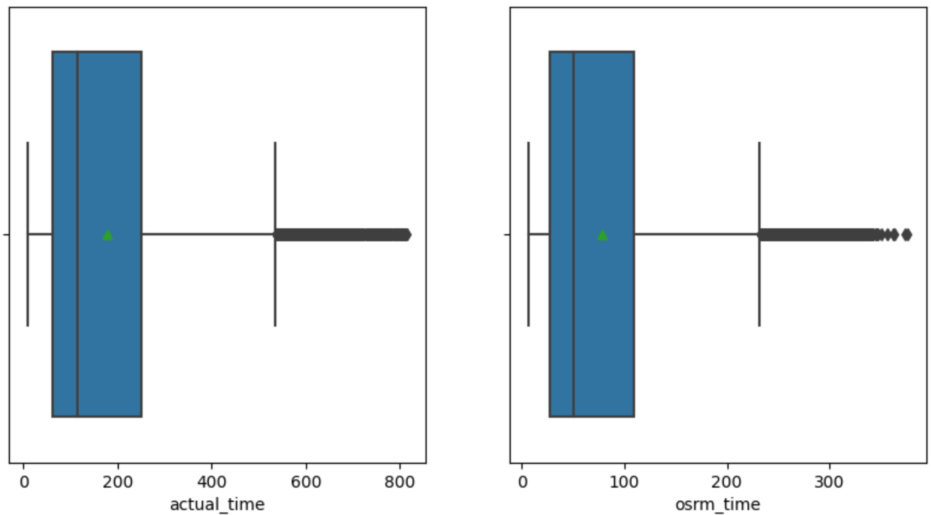
```
data['osrm_time'].mean()
```

    78.44030495952212

```
plt.figure(figsize=(10,5))
plt.subplot(121)
sns.histplot(data['actual_time'])

plt.subplot(122)
sns.histplot(data['osrm_time'])

plt.show()
```




```
plt.figure(figsize=(10,5))
plt.subplot(121)
sns.boxplot(data = data , x = 'actual_time',showmeans=True)

plt.subplot(122)
sns.boxplot(data = data , x = 'osrm_time',showmeans=True)

plt.show()
```



Do hypothesis testing/ visual analysis between actual_time aggregated value and segment actual time aggregated value (aggregated values are the values you'll get after merging the rows on the basis of trip_uuid)

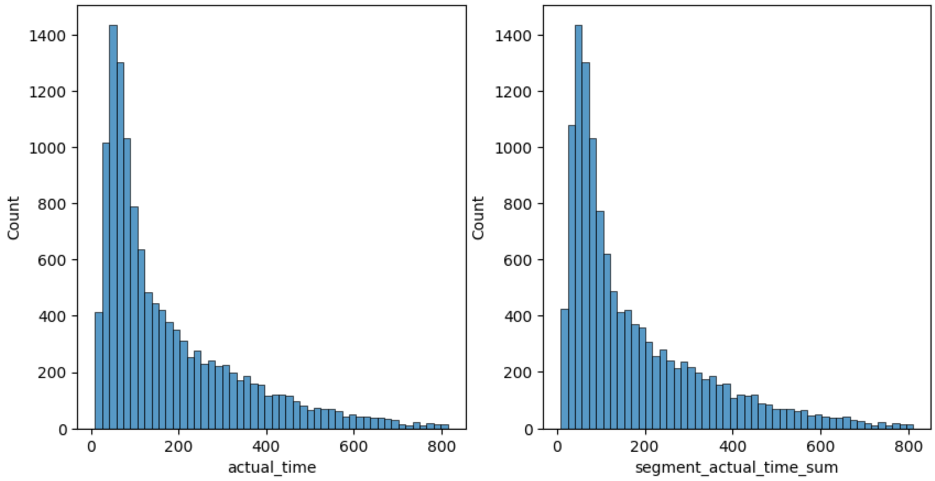
```
data['actual_time'].mean()
```

```
177.4527234142891
```

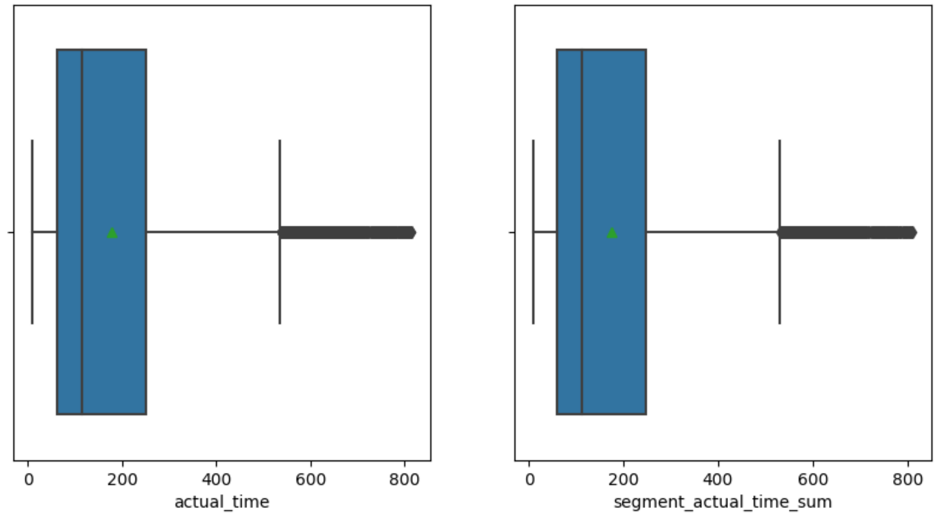
```
data['segment_actual_time_sum'].mean()
```

```
175.79627446356992
```

```
plt.figure(figsize=(10,5))
plt.subplot(121)
sns.histplot(data['actual_time'])

plt.subplot(122)
sns.histplot(data['segment_actual_time_sum'])

plt.show()
```




```
plt.figure(figsize=(10,5))
plt.subplot(121)
sns.boxplot(data = data , x = 'actual_time',showmeans=True)

plt.subplot(122)
sns.boxplot(data = data , x = 'segment_actual_time_sum',showmeans=True)

plt.show()
```



Do hypothesis testing/ visual analysis between osrm distance aggregated value and segment osrm distance aggregated value

```
data['osrm_distance'].mean()
```

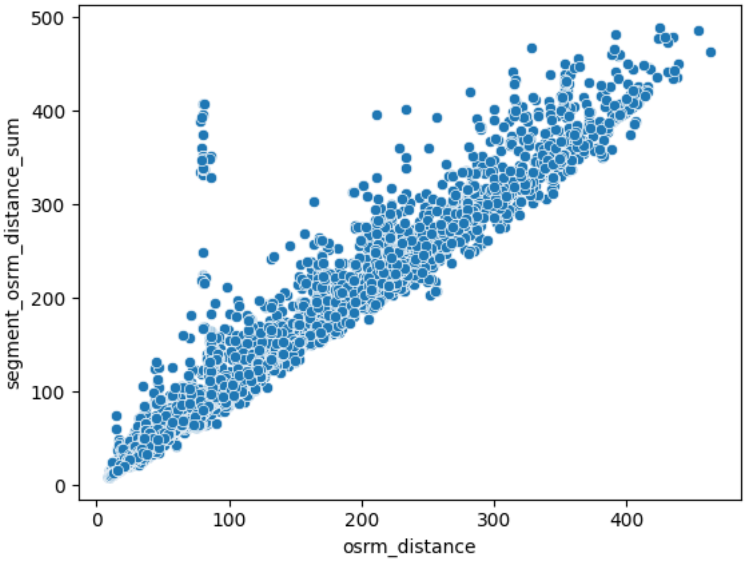
    91.73403030731745

```
data['segment_osrm_distance_sum'].mean()
```

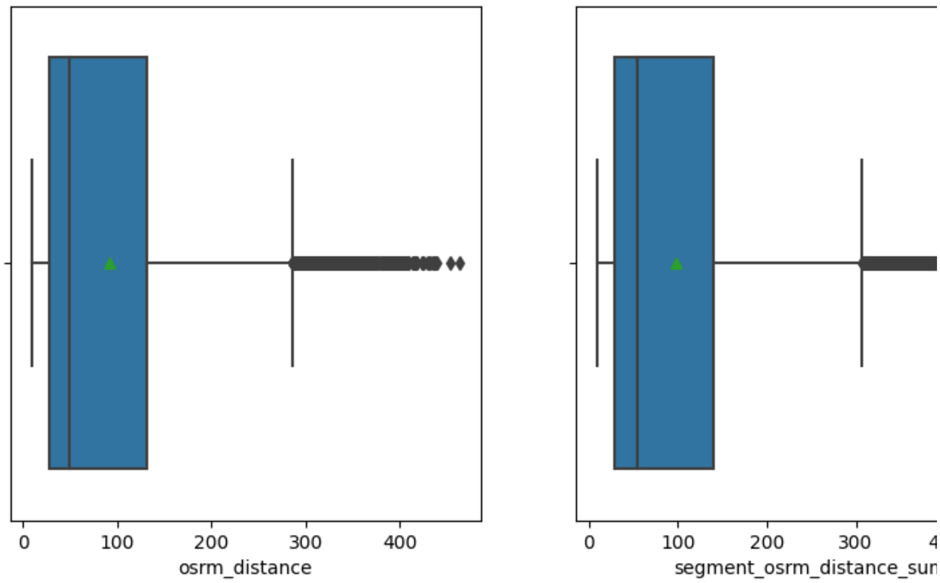
    97.97155838245698

```
sns.scatterplot(x=data['osrm_distance'],y =data['segment_osrm_distance_sum'])
```

    <Axes: xlabel='osrm_distance', ylabel='segment_osrm_distance_sum'>




```
plt.figure(figsize=(10,5))
plt.subplot(121)
sns.boxplot(data = data , x = 'osrm_distance',showmeans=True)

plt.subplot(122)
sns.boxplot(data = data , x = 'segment_osrm_distance_sum',showmeans=True)

plt.show()
```



Do hypothesis testing/ visual analysis between osrm time aggregated value and segment osrm time aggregated value (aggregated values are the values you'll get after merging the rows on the basis of trip_uuid)

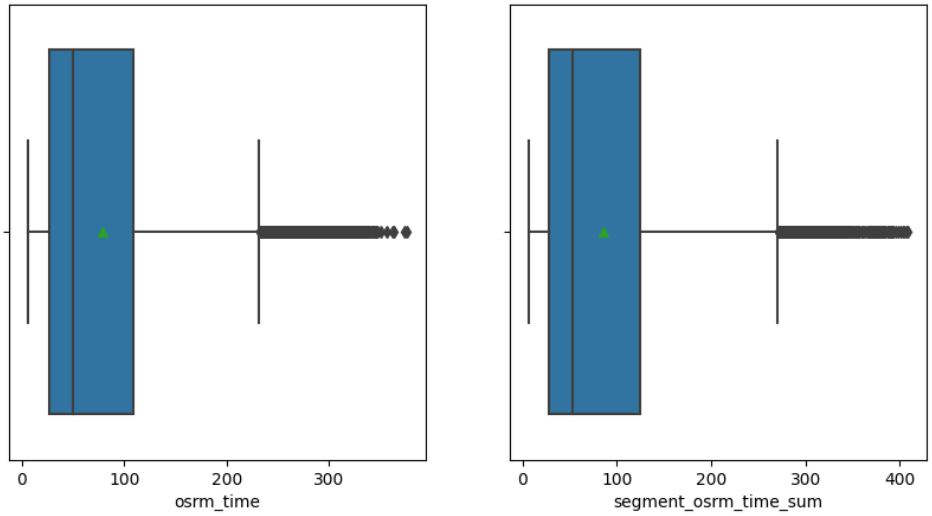
```
data['osrm_time'].mean()
```

    78.44030495952212

```
data['segment_osrm_time_sum'].mean()
```

```
85.90835494773245
```

```
plt.figure(figsize=(10,5))
plt.subplot(121)
sns.boxplot(data = data , x = 'osrm_time',showmeans=True)

plt.subplot(122)
sns.boxplot(data = data , x = 'segment_osrm_time_sum',showmeans=True)

plt.show()
```



Do one-hot encoding of categorical variables (like route_type)

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

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

```
enc = OneHotEncoder()

enc_data = pd.DataFrame(enc.fit_transform(data[['route_type']]).toarray())
enc_data
```

	0	1
0	1.0	0.0
1	1.0	0.0
2	0.0	1.0
3	1.0	0.0
4	1.0	0.0
...	...	...
12718	1.0	0.0
12719	1.0	0.0
12720	1.0	0.0
12721	1.0	0.0
12722	0.0	1.0

12723 rows × 2 columns

```
# Merge with main
data_new = data.join(enc_data)
data_new
```

trip_uuid	source_center	source_name	destination_center	destinati
trip-153671042288605164	IND561203AAB	Doddablpur_ChikaDPP_D (Karnataka)	IND561203AAB	Doddablpur_Chik (Ka
trip-153671046011330457	IND400072AAB	Mumbai Hub (Maharashtra)	IND401104AAA	Mumbai_M (Mah
trip-153671052974046625	IND583101AAA	Bellary_Dc (Karnataka)	IND583119AAA	Sandur_WrdN (Ka
trip-153671055416136166	IND600056AAA	Chennai_Poonamallee (Tamil Nadu)	IND600056AAA	Chennai_Poo (Tar
trip-153671066201138152	IND600044AAD	Chennai_Chrompet_DPC (Tamil Nadu)	IND600048AAA	Chennai_Van (Tar
...	...	...	...	
trip-153861095625827784	IND160002AAC	Chandigarh_Mehmdpur_H (Punjab)	IND160002AAC	Chandigarh_Meh
trip-153861104386292051	IND121004AAB	FBD_Balabhgarh_DPC (Haryana)	IND121004AAA	Faridabad_Blb (
trip-153861106442901555	IND208006AAA	Kanpur_GovndNgr_DC (Uttar Pradesh)	IND208006AAA	Kanpur_Govn (Uttar
trip-153861115439069069	IND627005AAA	Tirunelveli_VdkkuSrt_I (Tamil Nadu)	IND628204AAA	Tirchchndr_Shnr (Tar
trip-153861118270144424	IND583119AAA	Sandur_WrdN1DPP_D (Karnataka)	IND583119AAA	Sandur_WrdN (Ka

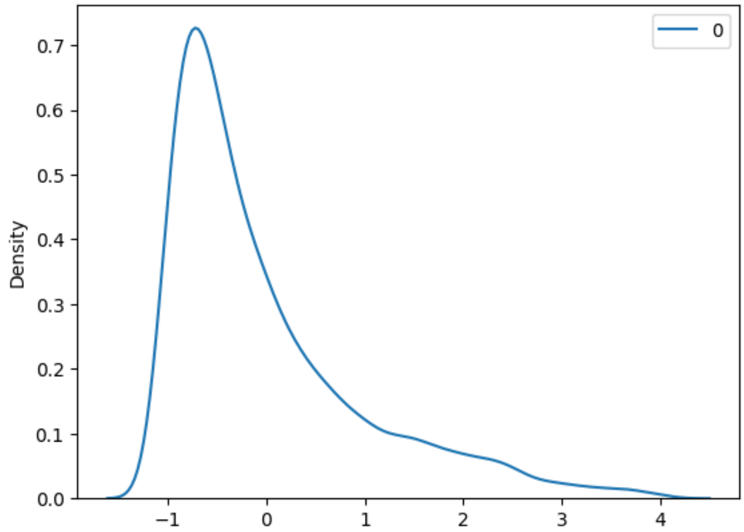
Normalize/ Standardize the numerical features using MinMaxScaler or StandardScaler.

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 12723 entries, 1 to 14786
Data columns (total 30 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   data                            12723 non-null  object
 1   trip_creation_time              12723 non-null  datetime64[ns]
 2   route_schedule_uuid             12723 non-null  object
 3   route_type                      12723 non-null  object
 4   trip_uuid                       12723 non-null  object
 5   source_center                   12723 non-null  object
 6   source_name                     12723 non-null  object
 7   destination_center              12723 non-null  object
 8   destination_name                12723 non-null  object
 9   od_trip_duration_hr             12723 non-null  float64
 10  start_scan_to_end_scan          12723 non-null  float64
 11  actual_distance_to_destination  12723 non-null  float64
 12  actual_time                     12723 non-null  float64
 13  osrm_time                       12723 non-null  float64
 14  osrm_distance                   12723 non-null  float64
 15  segment_actual_time_sum         12723 non-null  float64
 16  segment_osrm_time_sum           12723 non-null  float64
 17  segment_osrm_distance_sum       12723 non-null  float64
 18  dest_state                      12723 non-null  object
 19  source_state                    12723 non-null  object
 20  dest_city                       12723 non-null  object
 21  source_city                     12723 non-null  object
 22  dest_Place                      12723 non-null  object
 23  source_Place                    12723 non-null  object
 24  dest_code                       11816 non-null  object
 25  source_code                     11693 non-null  object
 26  year                            12723 non-null  int64
 27  month                           12723 non-null  int64
 28  day                             12723 non-null  int64
 29  hour                            12723 non-null  int64
dtypes: datetime64[ns](1), float64(9), int64(4), object(16)
memory usage: 3.5+ MB
```

```
x1 =[['actual_time']]
x2 =[['segment_actual_time_sum']]


stan = preprocessing.StandardScaler()
standard_df = pd.DataFrame(stan.fit_transform(data[['od_trip_duration_hr']]))

sns.kdeplot(standard_df, color ='black')
```

```
<Axes: ylabel='Density'>
```



Insights:

1)Maharastra State is the most fequently used Source and destination states of all.

2)As a source location Arunachal Pradesh is the least and as destination location Dam & Diu is the least.

3) Most no.of trip has been created in September month and on hourly basis it is at 9 - 11 PM.

4)There are lots of outliers detetced in numerical columns.

5) Actual Average time and osrm time is having difference of almost 60 seconds.(assuming data given in seconds)

6) Actual and segment actual average and median time is almost same.

7) osrm and segment osrm are higly correlated to each other.

Recommendations:

1) North, South Zones corridors have significant traffic of orders.

2) We have a smaller presence in Central, Eastern and North-Eastern zone. However it would be difficult to conclude this, by looking at just 2 months data. It is worth investigating and increasing our presence in these regions.

3) From state point of view, we have heavy traffic in Mahrashtra followed by Karnataka. This is a good indicator that we need to plan for resources on ground in these 2 states on priority. Especially, during festive seasons.

4) And Arunachal pradesh and dam & diu has the least traffic . So we can reduce the resources on these 2 states as per the records given.

5) Trips are heavy at the time of 9 - 11 PM so resources at that period of time should be kept high as emergency.

2) Actual time and segment actual time is almost same so keep that up.

0s completed at 23:50