Business Case Study : Delhivery

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.

Business Problem -

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

In [134…

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import math

from scipy.stats import ttest_1samp, ttest_ind
from scipy.stats import ttest_ind # Numeric vs categorical
from scipy.stats import ks_2samp
from statsmodels.graphics.gofplots import qqplot
from scipy.stats import spearmanr, ttest_rel

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

In [3]:

```
df = pd.read_csv("/Users/bose/Downloads/delhivery_data.csv")
```

In [4]:

```
df.head()
```

Out[4]:

	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

5 rows × 24 columns

In [7]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 144316 entries, 0 to 144315
Data columns (total 24 columns):
 #   Column                          Non-Null Count   Dtype  
---  ------                          --------------   -----  
 0   data                            144316 non-null  object 
 1   trip_creation_time              144316 non-null  object 
 2   route_schedule_uuid             144316 non-null  object 
 3   route_type                      144316 non-null  object 
 4   trip_uuid                       144316 non-null  object 
 5   source_center                   144316 non-null  object 
 6   source_name                     144316 non-null  object 
 7   destination_center              144316 non-null  object 
 8   destination_name                144316 non-null  object 
 9   od_start_time                   144316 non-null  object 
 10  od_end_time                     144316 non-null  object 
 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), float64(10), int64(1), object(12)
memory usage: 25.5+ MB
```

Observations on :

- Shape of data
- Datatypes
- Statistical Summary

In [15]:

```
#Shape of data
df.shape
```

Out[15]:

```
(144316, 24)
```

In [16]:

```
#Datatypes of all attributes
df.dtypes
```

Out[16]:

```
data                                       object
trip_creation_time                 datetime64[ns]
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                      datetime64[ns]
od_end_time                        datetime64[ns]
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 [17]:

```
#Statistical Summary
df.describe()
```

Out[17]:

	start_scan_to_end_scan	cutoff_factor	actual_distance_to_destination	actual_tim
count	144316.000000	144316.000000	144316.000000	144316.00000
mean	963.697698	233.561345	234.708498	417.99623
std	1038.082976	345.245823	345.480571	598.94006
min	20.000000	9.000000	9.000045	9.00000
25%	161.000000	22.000000	23.352027	51.00000
50%	451.000000	66.000000	66.135322	132.00000
75%	1645.000000	286.000000	286.919294	516.00000
max	7898.000000	1927.000000	1927.447705	4532.00000

In [18]:

```
df.describe(include = 'object')
```

Out[18]:

	data	route_schedule_uuid	route_type	trip_uuid	source_center
count	144316	144316	144316	144316	144316
unique	2	1497	2	14787	1496
top	training	thanos::sroute:4029a8a2-6c74-4b7e-a6d8-f9e069f...	FTL	trip-153837029526866991	IND000000ACB
freq	104632	1812	99132	101	23267

In [170…

```
df[df.duplicated()]
```

Out[170]:

data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_center	sour

0 rows × 37 columns

No duplicate records found

Removing Null Values -

In [6]:

```
df = df.dropna(how='any')
df = df.reset_index(drop=True)
```

Changing the datatype of attributes -

In [8]:

```
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"])
```

In [9]:

```
df["trip_creation_time"].dt.month_name().value_counts()
```

Out[9]:

```
September    126932
October       17384
Name: trip_creation_time, dtype: int64
```

In [10]:

```
df["trip_creation_time"].dt.year.value_counts()
```

Out[10]:

```
2018    144316
Name: trip_creation_time, dtype: int64
```

In [11]:

```
df["trip_creation_time"].dt.day_name().value_counts()
```

Out[11]:

```
Wednesday    26634
Thursday     20422
Friday       20177
Saturday     19874
Tuesday      19858
Monday       19540
Sunday       17811
Name: trip_creation_time, dtype: int64
```

In [174…

```
df.nunique()
```

Out[174]:

```
data                                   2
trip_creation_time                 14787
route_schedule_uuid                 1497
route_type                             2
trip_uuid                          14787
source_center                       1496
source_name                         1496
destination_center                  1466
destination_name                    1466
od_start_time                      26223
od_end_time                        26223
start_scan_to_end_scan              1914
is_cutoff                              2
cutoff_factor                        501
cutoff_timestamp                   92894
actual_distance_to_destination    143965
actual_time                         3182
osrm_time                           1531
osrm_distance                     137544
factor                             45588
segment_actual_time                  746
segment_osrm_time                    214
segment_osrm_distance             113497
segment_factor                      5663
segment_key                        26222
segment_actual_time_sum             3153
segment_osrm_distance_sum         138589
segment_osrm_time_sum               1870
Diff_betw_odstart_odend_1          26223
source_city                         1240
source_state                          54
destination_city                    1237
destination_state                     52
source_pincode                      1384
destination_pincode                 1374
source_city_state                   1248
destination_city_state              1240
dtype: int64
```

In [19]:

```
df['segment_key'] = df['trip_uuid'] + df['source_center'] + df['destination_

segment_cols = ['segment_actual_time', 'segment_osrm_distance', 'segment_os

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

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

Out[19]:

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

144316 rows × 3 columns

In [20]:
```
df.head(5)
```

Out[20]:

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

5 rows × 28 columns

In [21]:
```
create_segment_dict = {

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

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

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

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

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

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

}
```

In [22]:
```
segment = df.groupby('segment_key').agg(create_segment_dict).reset_index()
segment = segment.sort_values(by=['segment_key','od_end_time'], ascending=T
```

In [23]:
```
segment
```

Out[23]:

	index	segment_key	data	trip_creation_time
0	0	trip-153671041653548748IND209304AAAIND000000ACB	training	2018-09-12 00:00:16.535741
1	1	trip-153671041653548748IND462022AAAIND209304AAA	training	2018-09-12 00:00:16.535741
2	2	trip-153671042288605164IND561203AABIND562101AAA	training	2018-09-12 00:00:22.886430
3	3	trip-153671042288605164IND572101AAAIND561203AAB	training	2018-09-12 00:00:22.886430
4	4	trip-153671043369099517IND000000ACBIND160002AAC	training	2018-09-12 00:00:33.691250
...	...	...	...	...
26217	26217	trip-153861115439069069IND628204AAAIND627657AAA	test	2018-10-03 23:59:14.390954
26218	26218	trip-153861115439069069IND628613AAAIND627005AAA	test	2018-10-03 23:59:14.390954
26219	26219	trip-153861115439069069IND628801AAAIND628204AAA	test	2018-10-03 23:59:14.390954
26220	26220	trip-153861118270144424IND583119AAAIND583101AAA	test	2018-10-03 23:59:42.701692
26221	26221	trip-153861118270144424IND583201AAAIND583119AAA	test	2018-10-03 23:59:42.701692

26222 rows × 21 columns

In [24]:
```
segment.info()
```

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

Calculating Time taken between od_start_time and od_end_time

In [25]:

```
segment['od_time_diff'] = (segment['od_end_time'] - segment['od_start_time'
segment['od_time_diff']
```

Out[25]:

```
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_time_diff, Length: 26222, dtype: float64
```

In [27]:

```
segment.head(5)
```

Out[27]:

	index	segment_key	data	trip_creation_time	r
0	0	trip-153671041653548748IND209304AAAIND000000ACB	training	2018-09-12 00:00:16.535741	than
1	1	trip-153671041653548748IND462022AAAIND209304AAA	training	2018-09-12 00:00:16.535741	than
2	2	trip-153671042288605164IND561203AABIND562101AAA	training	2018-09-12 00:00:22.886430	than
3	3	trip-153671042288605164IND572101AAAIND561203AAB	training	2018-09-12 00:00:22.886430	than
4	4	trip-153671043369099517IND000000ACBIND160002AAC	training	2018-09-12 00:00:33.691250	than

5 rows × 22 columns

In [30]:

```
create_trip_dict = {

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

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

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

    'start_scan_to_end_scan' : 'sum',
    'od_time_diff' : '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 [31]:

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

In [32]:

```
trip.head(5)
```

Out[32]:

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

Extracting City, States and Pincodes -

In [164…

```
trip["source_city"] = trip["source_name"].str.split(" ", n = 1, expand = Tru
trip["source_state"] = trip["source_name"].str.split(" ", n = 1, expand = T
```

In [165…

```
trip["destination_city"] = trip["destination_name"].str.split(" ", n = 1,exp
trip["destination_state"] = trip["destination_name"].str.split(" ",n = 1,exp
```

In [41]:

```
trip["source_pincode"] = trip["source_center"].apply(lambda x : x[3:9] )
trip["destination_pincode"] = trip["destination_center"].apply(lambda x : x
```

In [42]:

```
trip.head()
```

Out[42]:

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

5 rows × 24 columns

In [72]:

```
df["Diff_betw_odstart_odend_1"] = (df["od_end_time"] - df["od_start_time"])
```

In [45]:

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

trip['trip_year'] = trip['trip_creation_time'].dt.year
```

```
trip['trip_month'] = trip['trip_creation_time'].dt.month
trip['trip_hour'] = trip['trip_creation_time'].dt.hour
trip['trip_day'] = trip['trip_creation_time'].dt.day
trip['trip_week'] = trip['trip_creation_time'].dt.isocalendar().week
trip['trip_dayofweek'] = trip['trip_creation_time'].dt.dayofweek
```

Extracting features like Destination name, Source name, Trip creation time -

In [46]:

```
trip[['destination_city','destination_state','source_city','source_state','
```

Out[46]:

	destination_city	destination_state	source_city	source_state	trip_day	trip_month
0	Kanpur	Uttar Pradesh	Kanpur	Uttar Pradesh	12	9
1	Doddablpur	Karnataka	Doddablpur	Karnataka	12	9
2	Gurgaon	Haryana	Gurgaon	Haryana	12	9
3	Mumbai	Maharashtra	Mumbai	Hub Maharashtra	12	9
4	Sandur	Karnataka	Bellary	Karnataka	12	9
...	...	...	...	...	...	...
14782	Chandigarh	Punjab	Chandigarh	Punjab	3	10
14783	Faridabad	Haryana	FBD	Haryana	3	10
14784	Kanpur	Uttar Pradesh	Kanpur	Uttar Pradesh	3	10
14785	Tirchchndr	Tamil Nadu	Tirunelveli	Tamil Nadu	3	10
14786	Sandur	Karnataka	Sandur	Karnataka	3	10

14787 rows × 7 columns

In [166…

```
df["source_city"] = df["source_name"].str.split(" ", n = 1, expand = True)[
df["source_state"] = df["source_name"].str.split(" ", n = 1, expand = True)
df["destination_city"] = df["destination_name"].str.split(" ", n = 1,expand
df["destination_state"] = df["destination_name"].str.split(" ",n = 1,expand
df["source_pincode"] = df["source_center"].apply(lambda x : x[3:9] )
df["destination_pincode"] = df["destination_center"].apply(lambda x : x[3:9
```

In [97]:

```
df["start_scan_to_end_scan"] = df["start_scan_to_end_scan"] / 60
df["actual_time"] = df["actual_time"] / 60
df["osrm_time"] = df["osrm_time"] / 60
df["segment_actual_time"] = df["segment_actual_time"] / 60
df["segment_osrm_time"] = df["segment_osrm_time"] / 60
```

Data Cleaning -

In [106…

```
df["source_state"].unique()
```

Out[106]:

```
array(['Gujarat', 'Maharashtra', 'Karnataka', 'Punjab', 'Haryana',
       'Uttarakhand', 'Tamil Nadu', 'Rajasthan', 'Telangana',
       'Madhya Pradesh', 'Uttar Pradesh', 'Himachal Pradesh', 'Kerala',
       'Andhra Pradesh', 'Bihar', 'Jharkhand', 'Hub Maharashtra', 'Assam',
       'West Bengal', 'Orissa', 'Delhi', 'Nagar_DC Rajasthan',
       'Jammu & Kashmir', 'Alipore_DPC West Bengal', 'Chandigarh',
       'Chhattisgarh', 'Vadgaon Sheri DPC Maharashtra', 'Goa',
       '02_DPC Uttar Pradesh', 'MP Nagar Madhya Pradesh', 'Road Punjab',
       'Pondicherry', 'Layout PC Karnataka', 'Mandakni Madhya Pradesh',
       'Dadra and Nagar Haveli', 'DC Maharashtra', 'Arunachal Pradesh',
       'Antop Hill Maharashtra', 'City Madhya Pradesh',
       'Pashan DPC Maharashtra', 'Nagaland', 'Meghalaya', 'DC Rajasthan',
       'West _Dc Maharashtra', 'Nagar Uttar Pradesh',
       '_NAD Andhra Pradesh', 'Avenue_DPC West Bengal', 'Tripura',
       'Mizoram', 'Rahatani DPC Maharashtra', 'Balaji Nagar Maharashtra',
       'Goa Goa', 'Kothanur_L Karnataka', 'Mahim Maharashtra'],
      dtype=object)
```

In [108…

```
df["source_state"] = df["source_state"].replace(
{ "Goa Goa":"Goa","Layout PC Karnataka":"Karnataka", "Vadgaon Sheri DPC Maha

"Pashan DPC Maharashtra":"Maharashtra", "City Madhya Pradesh":"Madhya Prades
"02_DPC Uttar Pradesh":"Uttar Pradesh", "Nagar_DC Rajasthan":"Rajasthan",
"Alipore_DPC West Bengal":"West Bengal", "Mandakni Madhya Pradesh":"Madhya P
"West _Dc Maharashtra":"Maharashtra", "DC Rajasthan":"Rajasthan",
"MP Nagar Madhya Pradesh":"Madhya Pradesh", "Antop Hill Maharashtra":"Mahara
"Avenue_DPC West Bengal":"West Bengal", "Nagar Uttar Pradesh":"Uttar Prades
"Balaji Nagar Maharashtra":"Maharashtra", "Kothanur_L Karnataka":"Karnataka"
"Rahatani DPC Maharashtra":"Maharashtra", "Mahim Maharashtra":"Maharashtra",
"DC Maharashtra":"Maharashtra", "_NAD Andhra Pradesh":"Andhra Pradesh" } )
```

In [109…

```
df["destination_state"].unique()
```

Out[109]:

```
array(['Gujarat', 'Maharashtra', 'Karnataka', 'Kerala', 'Punjab',
       'Uttarakhand', 'Tamil Nadu', 'Haryana', 'Rajasthan', 'Telangana',
       'Uttar Pradesh', 'Delhi', 'Himachal Pradesh', 'Hub Maharashtra',
       'Andhra Pradesh', 'Bihar', 'Jharkhand', 'Assam', 'Orissa',
       'West Bengal', 'Pashan DPC Maharashtra', 'Jammu & Kashmir',
       'Madhya Pradesh', 'Avenue_DPC West Bengal', 'Chandigarh',
       'Chhattisgarh', 'Vadgaon Sheri DPC Maharashtra',
       '02_DPC Uttar Pradesh', 'Goa', 'MP Nagar Madhya Pradesh',
       'Pondicherry', 'Layout PC Karnataka', 'Mandakni Madhya Pradesh',
       'Arunachal Pradesh', 'Dadra and Nagar Haveli',
       'Nagar_DC Rajasthan', 'West _Dc Maharashtra',
       'Alipore_DPC West Bengal', 'Meghalaya', 'Rahatani DPC Maharashtra',
       'Nagar Uttar Pradesh', 'Kothanur_L Karnataka',
       'City Madhya Pradesh', 'Balaji Nagar Maharashtra', 'Tripura',
       'Mizoram', 'Daman & Diu', 'Nagaland', 'Goa Goa',
       'Antop Hill Maharashtra', 'West_Dc Maharashtra', 'Delhi Delhi'],
      dtype=object)
```

In [110…

```
df["destination_state"] = df["destination_state"].replace(
{ "Goa Goa":"Goa", "Layout PC Karnataka":"Karnataka", "Vadgaon Sheri DPC Mal

"Pashan DPC Maharashtra":"Maharashtra", "City Madhya Pradesh":"Madhya Prades
"02_DPC Uttar Pradesh":"Uttar Pradesh", "Nagar_DC Rajasthan":"Rajasthan",
"Alipore_DPC West Bengal":"West Bengal", "Mandakni Madhya Pradesh":"Madhya P
"West _Dc Maharashtra":"Maharashtra", "DC Rajasthan":"Rajasthan",
"MP Nagar Madhya Pradesh":"Madhya Pradesh", "Antop Hill Maharashtra":"Mahara
"Avenue_DPC West Bengal":"West Bengal", "Nagar Uttar Pradesh":"Uttar Pradesh
"Balaji Nagar Maharashtra":"Maharashtra", "Kothanur_L Karnataka":"Karnataka"
"Rahatani DPC Maharashtra":"Maharashtra", "Mahim Maharashtra":"Maharashtra",
"DC Maharashtra":"Maharashtra", "_NAD Andhra Pradesh":"Andhra Pradesh",
"Delhi Delhi":"Delhi", "West_Dc Maharashtra":"Maharashtra", "Hub Maharashtra
```

In [111…

```
df["source_city"].unique()[:100]
```

Out[111]:

```
array(['Anand', 'Khambhat', 'Bhiwandi', 'LowerParel', 'Bangalore',
       'Bengaluru', 'Ludhiana', 'Jagraon', 'Raikot', 'Junagadh',
       'Veraval', 'Kodinar', 'Una', 'Talala', 'Sonipat', 'Roorkee',
       'Haridwar', 'MAA', 'Jalandhar', 'Gurgaon', 'Jaipur', 'Ajmer',
       'Pali', 'Jodhpur', 'Hyderabad', 'Bhopal', 'Kanpur', 'Auraiya',
       'Etawah', 'Ahmedabad', 'Surat', 'Nanded', 'Loha', 'Gangakher',
       'Parli', 'Ambajogai', 'Mumbai', 'Loharu', 'ChrkhiDdri', 'Boisar',
       'Dahanu', 'Hapur', 'Bangana', 'Nadaun', 'Balotra', 'Pokhran',
       'Phalodi', 'Mehsana', 'Unjha', 'Patan', 'Bhabhar', 'AMD', 'Aluva',
       'Cochin', 'Pune', 'Solapur', 'Kakinada', 'Tuni', 'Purnia',
       'Supaul', 'Saharsa', 'Madhepura', 'Triveninganj', 'Visakhapatnam',
       'Anakapalle', 'Narsiptnm', 'Ranchi', 'Ramgarh', 'Hazaribag',
       'JhumriTlya', 'Beawar', 'Bilara', 'Bijainagar', 'Kekri',
       'Nasirabad', 'Bhuvanagiri', 'Mothkur', 'Thirumalagiri', 'Madhupur',
       'Khammam', 'Kodad', 'Guwahati', 'Morbi', 'Wankaner', 'BLR',
       'Kolkata', 'Bhubaneshwar', 'Alwar', 'Bharatpur', 'Weir', 'Kherli',
       'Bagnan', 'Kolaghat', 'Delhi', 'Puttaprthi', 'Hindupur',
       'GreaterThane', 'Patancheru', 'Del', 'Muzaffrpur'], dtype=object)
```

In [112…

```
df["source_city"] = df["source_city"].replace(
{ "del":"Delhi", "Bangalore":"Bengaluru", "AMD":"Ahmedabad", "Amdavad":"Ahme
```

In [113…

```
df["destination_city"] = df["destination_city"].replace(
{ "del":"Delhi", "Bangalore":"Bengaluru", "AMD":"Ahmedabad", "Amdavad":"Ahme
```

In [114…

```
df["source_city_state"] = df["source_city"] + " " + df["source_state"]
df["destination_city_state"] = df["destination_city"] + " " + df["destinatio
```

In [ ]:

```

```

Visual Analysis -

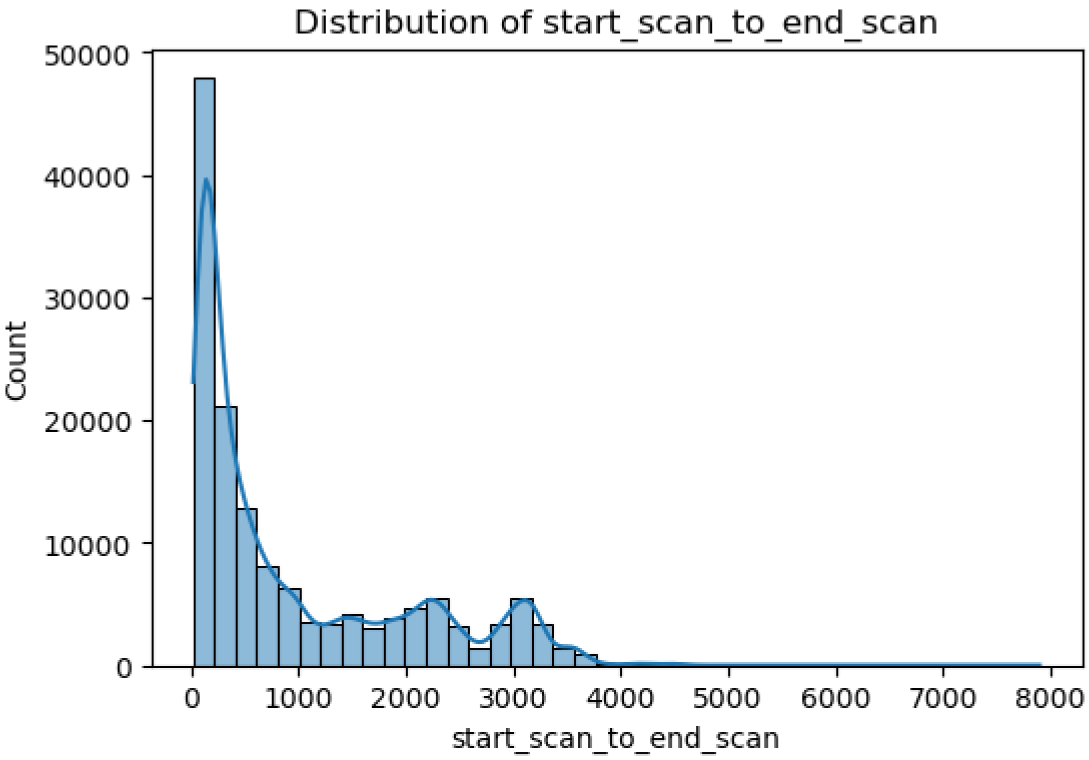
Univariate Analysis -

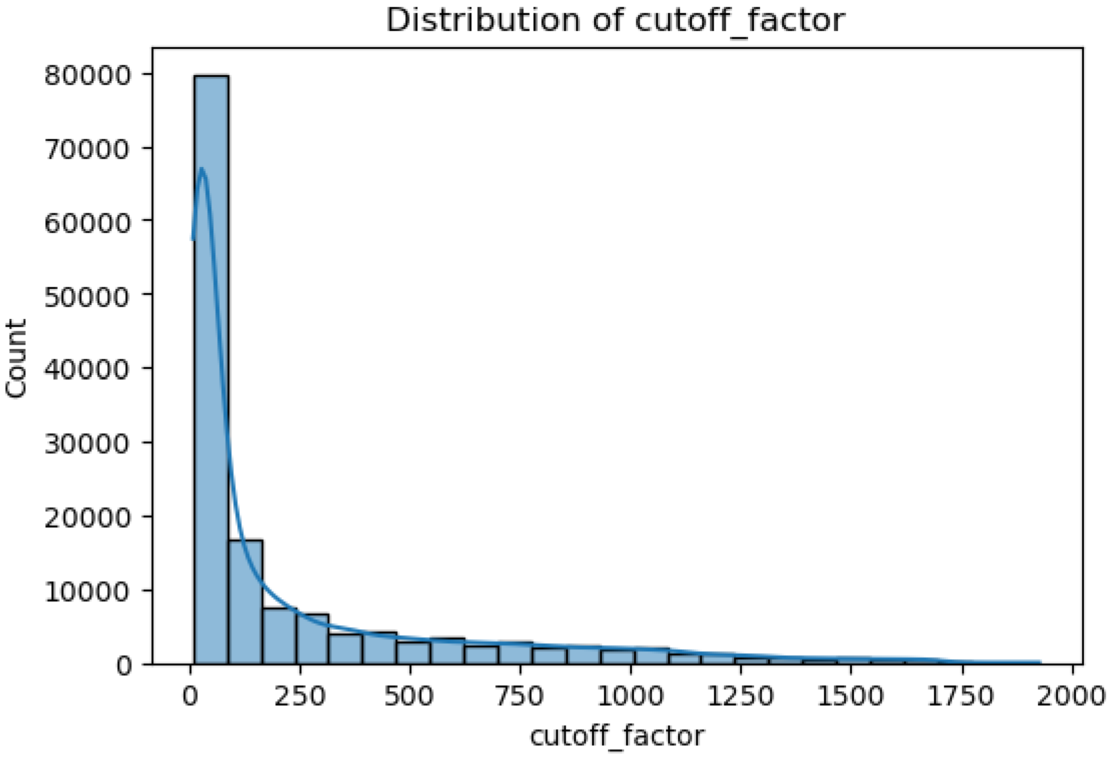
Histograms -

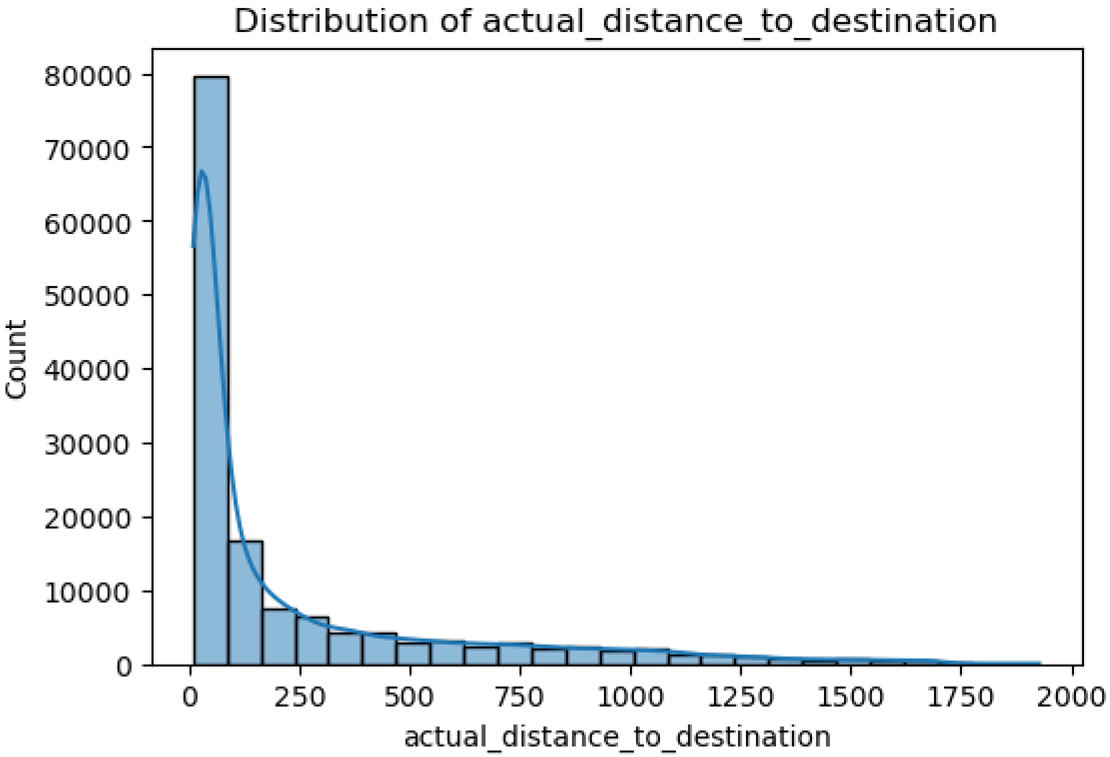
In [57]:

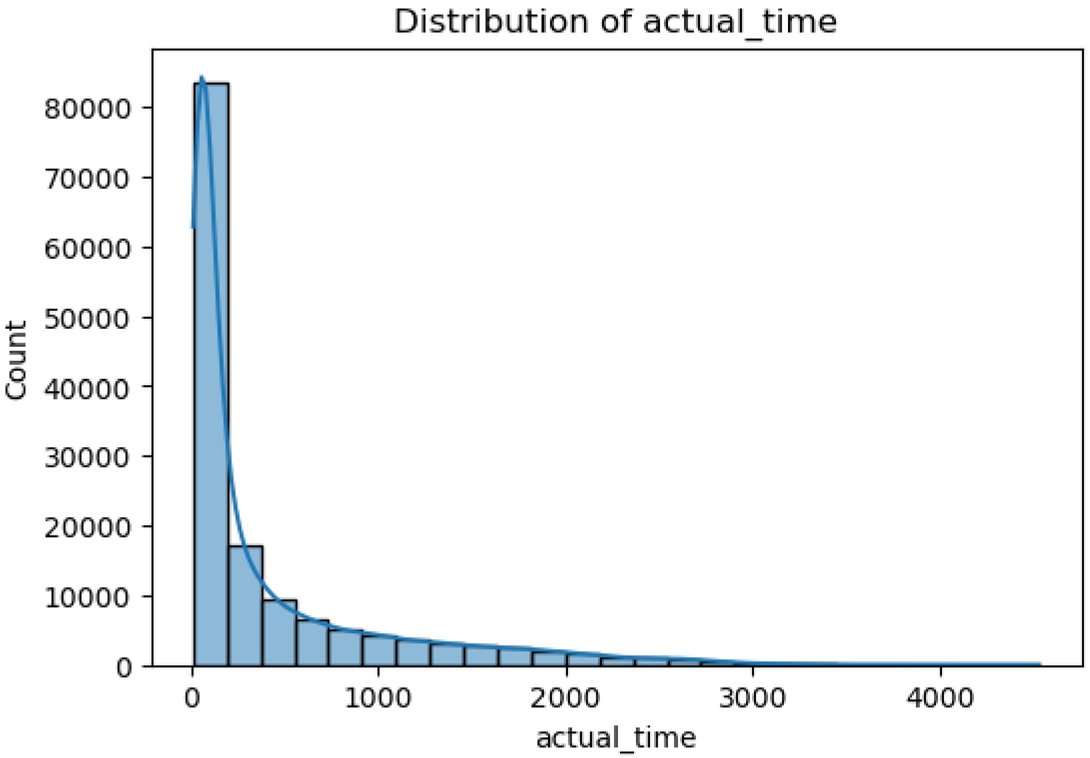
```
# start_scan_to_end_scan
plt.figure(figsize = (6, 4))
sns.histplot(x = "start_scan_to_end_scan", data = df, bins = 40, kde = True
plt.title("Distribution of start_scan_to_end_scan")
plt.show()

# cutoff_factor
plt.figure(figsize = (6, 4))
```

```
sns.histplot(x = "cutoff_factor", data = df, bins = 25, kde = True)
plt.title("Distribution of cutoff_factor")
plt.show()

# actual_distance_to_destination
plt.figure(figsize = (6, 4))
sns.histplot(x = "actual_distance_to_destination", data = df, bins = 25, kd
plt.title("Distribution of actual_distance_to_destination")
plt.show()

# actual_time
plt.figure(figsize = (6, 4))
sns.histplot(x = "actual_time", data = df, bins = 25, kde = True)
plt.title("Distribution of actual_time")
plt.show()
```



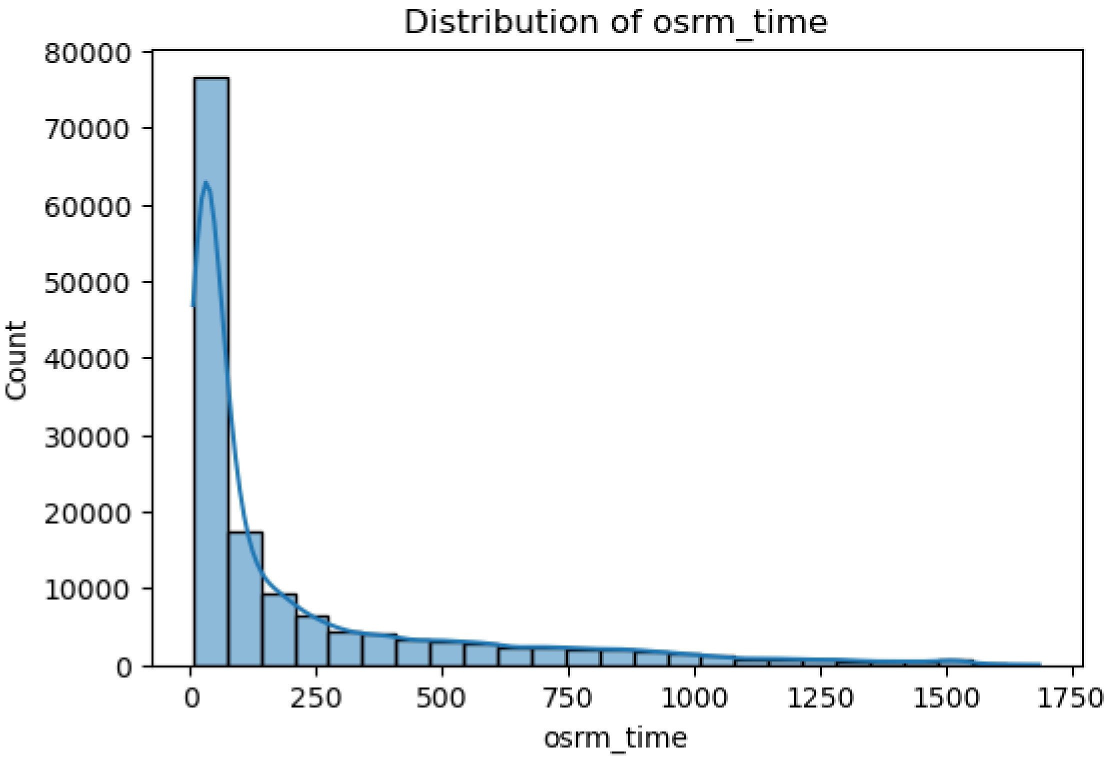


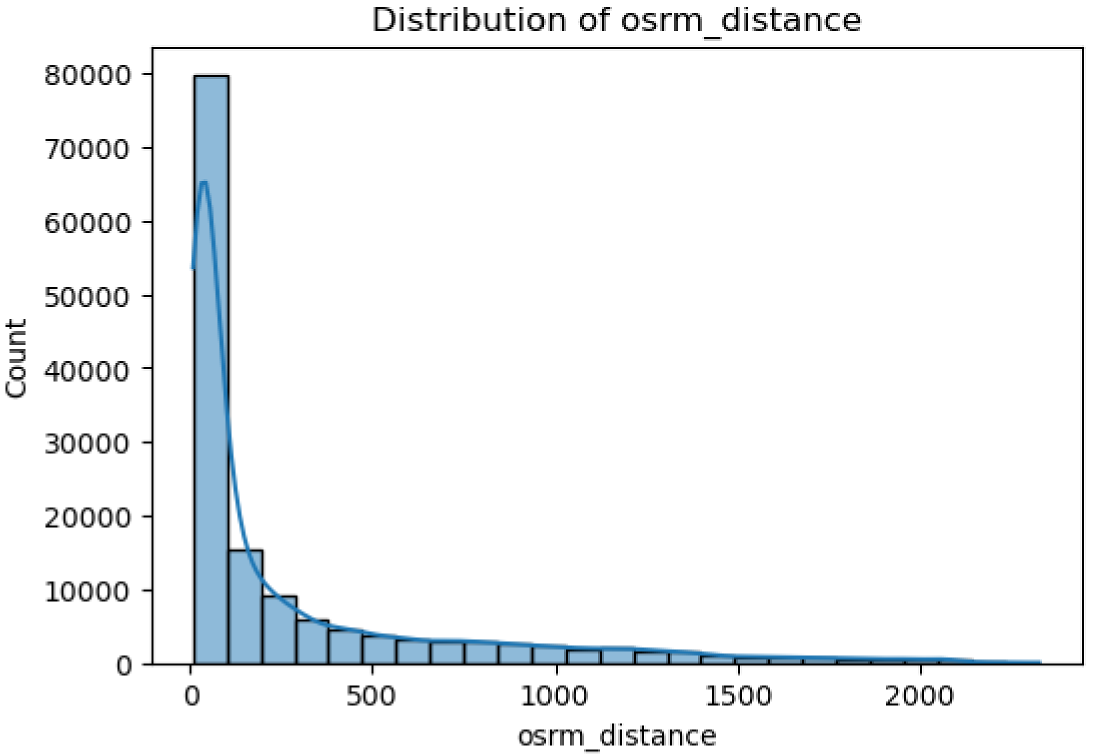


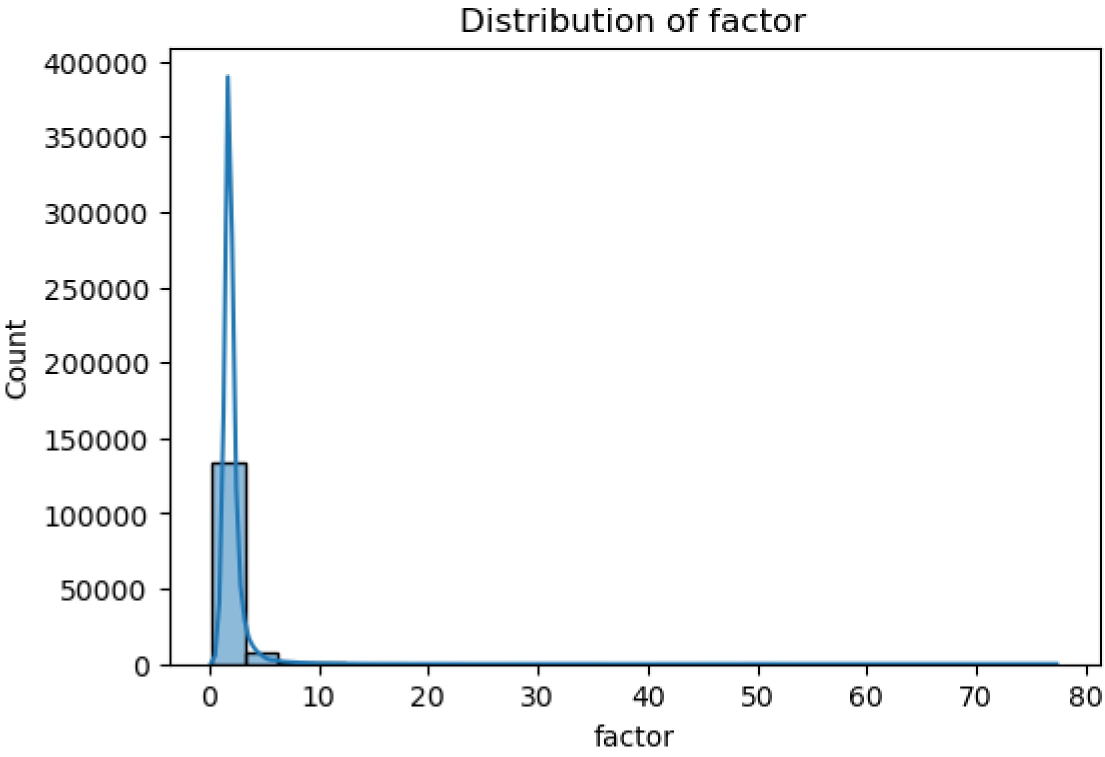


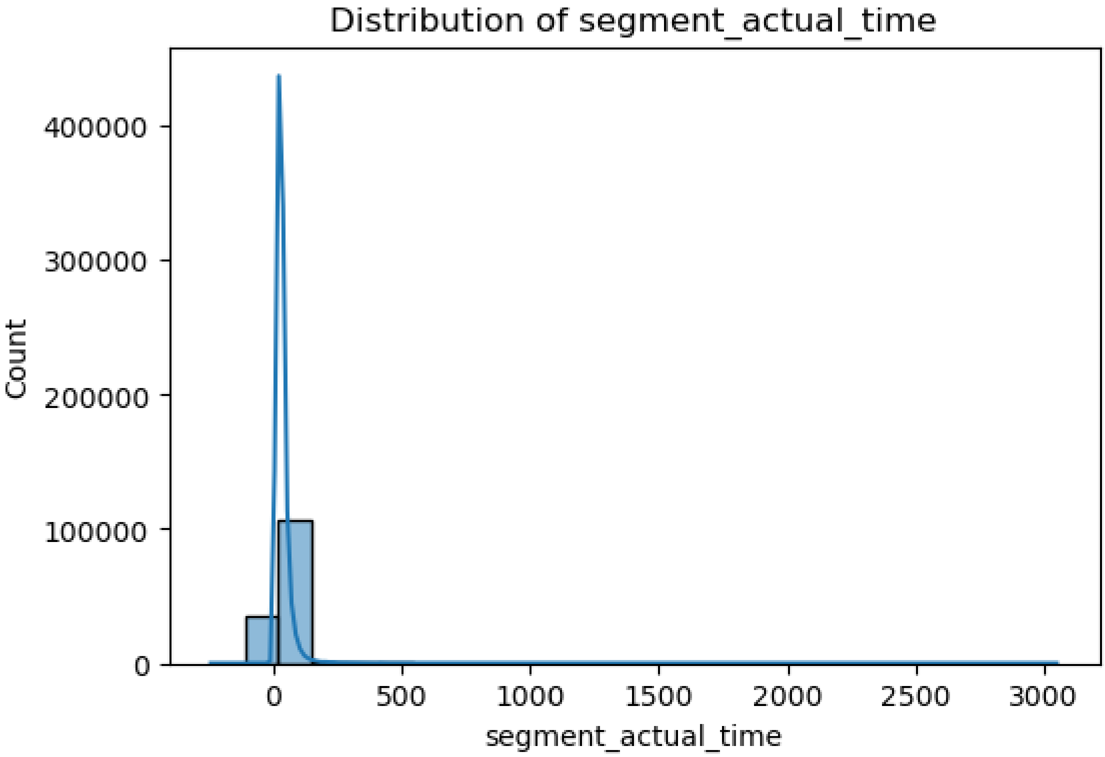
In [56]:
```
# osrm_time
plt.figure(figsize = (6, 4))
sns.histplot(x = "osrm_time", data = df, bins = 25, kde = True)
plt.title("Distribution of osrm_time")
plt.show()

# osrm_distance
plt.figure(figsize = (6, 4))
sns.histplot(x = "osrm_distance", data = df, bins = 25, kde = True)
plt.title("Distribution of osrm_distance")
plt.show()

# factor
plt.figure(figsize = (6, 4))
```

```
sns.histplot(x = "factor", data = df, bins = 25, kde = True)
plt.title("Distribution of factor")
plt.show()

# segment_actual_time
plt.figure(figsize = (6, 4))
sns.histplot(x = "segment_actual_time", data = df, bins = 25, kde = True)
plt.title("Distribution of segment_actual_time")
plt.show()
```



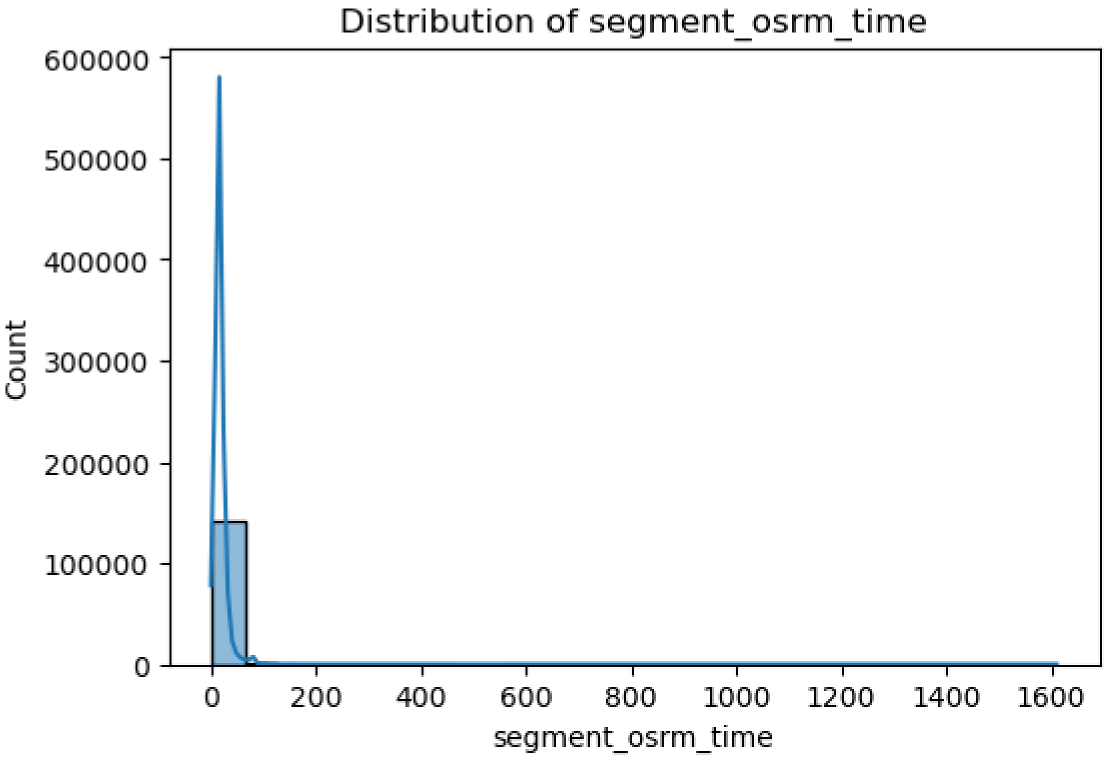


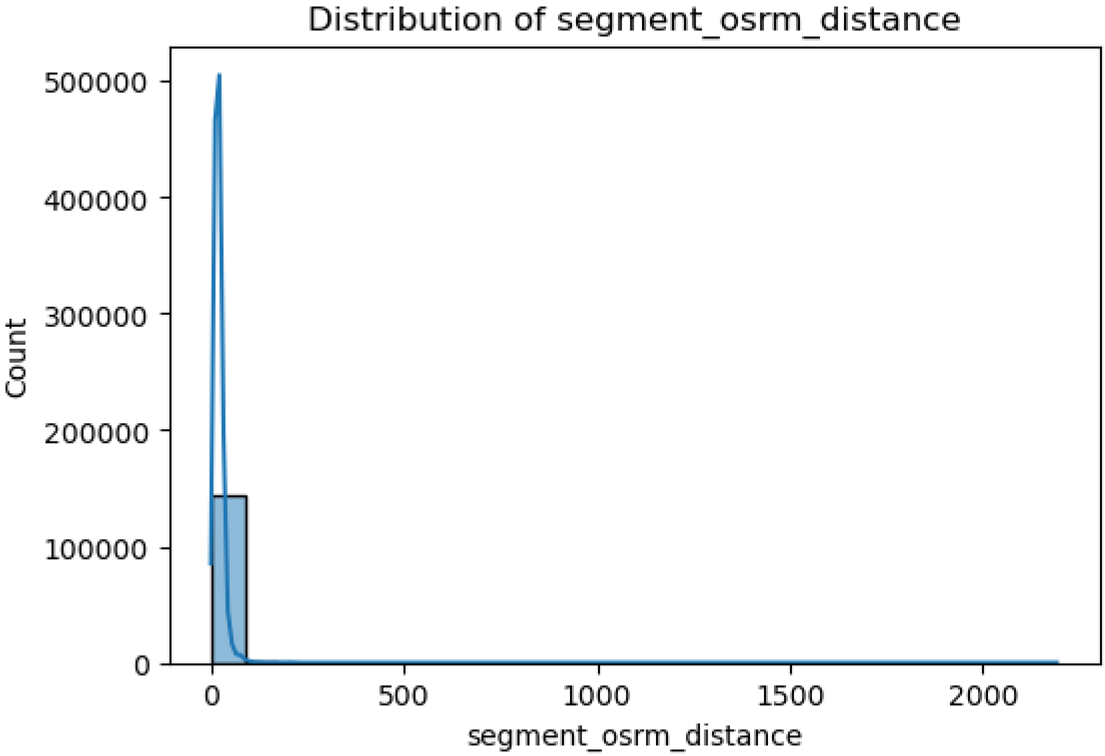


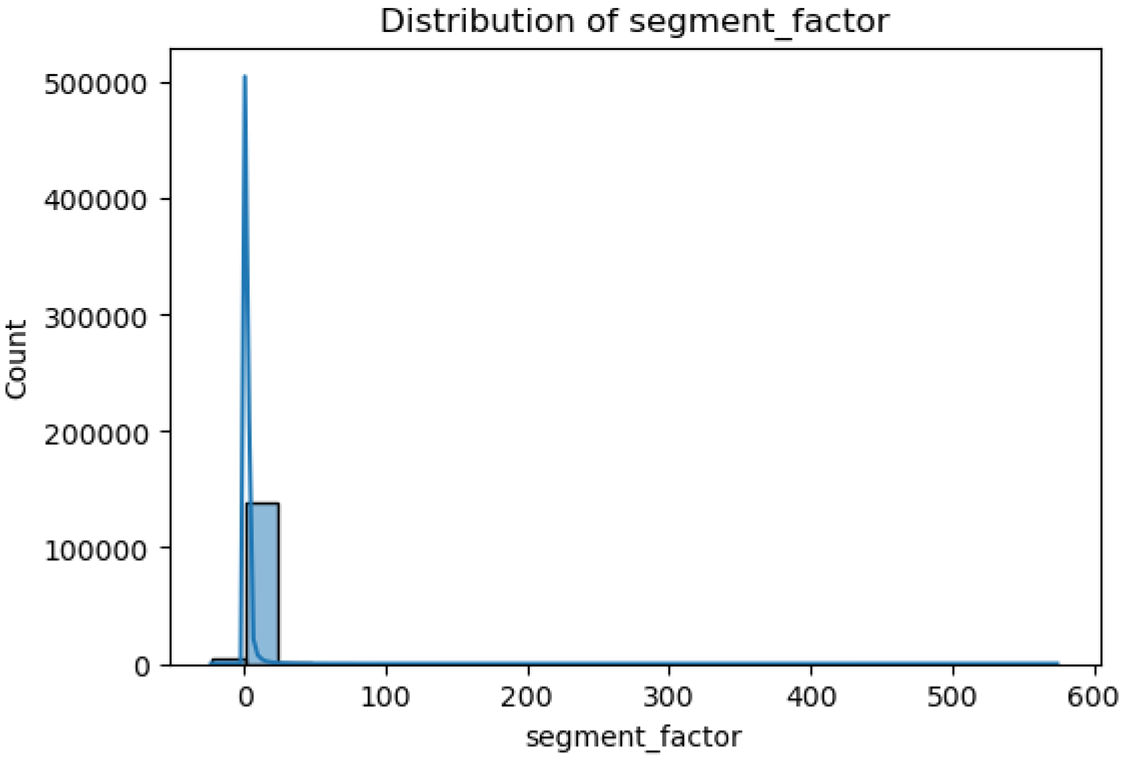


In [55]:
```
# segment_osrm_time
plt.figure(figsize = (6, 4))
sns.histplot(x = "segment_osrm_time", data = df, bins = 25, kde = True)
plt.title("Distribution of segment_osrm_time")
plt.show()

# segment_osrm_distance
plt.figure(figsize = (6, 4))
sns.histplot(x = "segment_osrm_distance", data = df, bins = 25, kde = True)
plt.title("Distribution of segment_osrm_distance")
plt.show()

# segment_factor
plt.figure(figsize = (6, 4))
```

```
sns.histplot(x = "segment_factor", data = df, bins = 25, kde = True)
plt.title("Distribution of segment_factor")
plt.show()
```



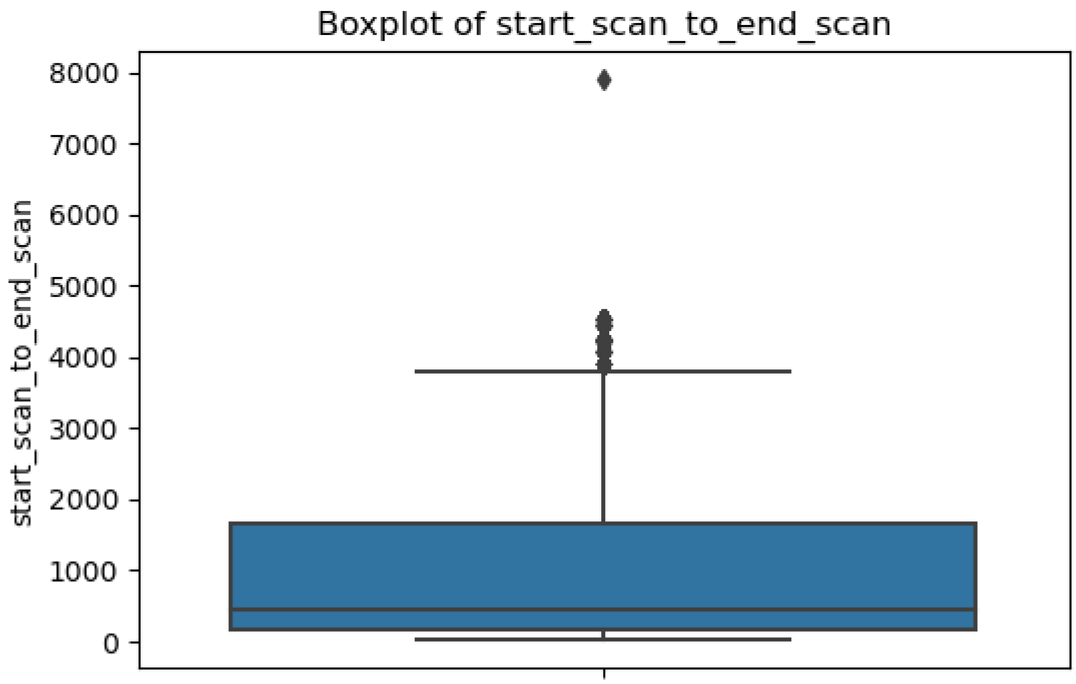


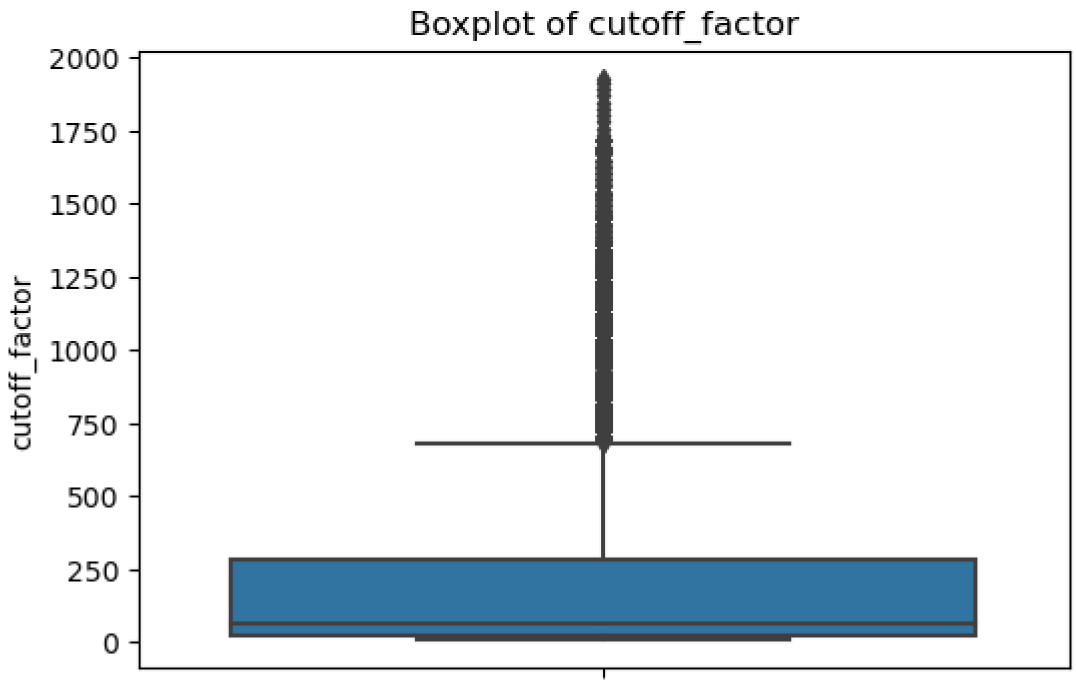


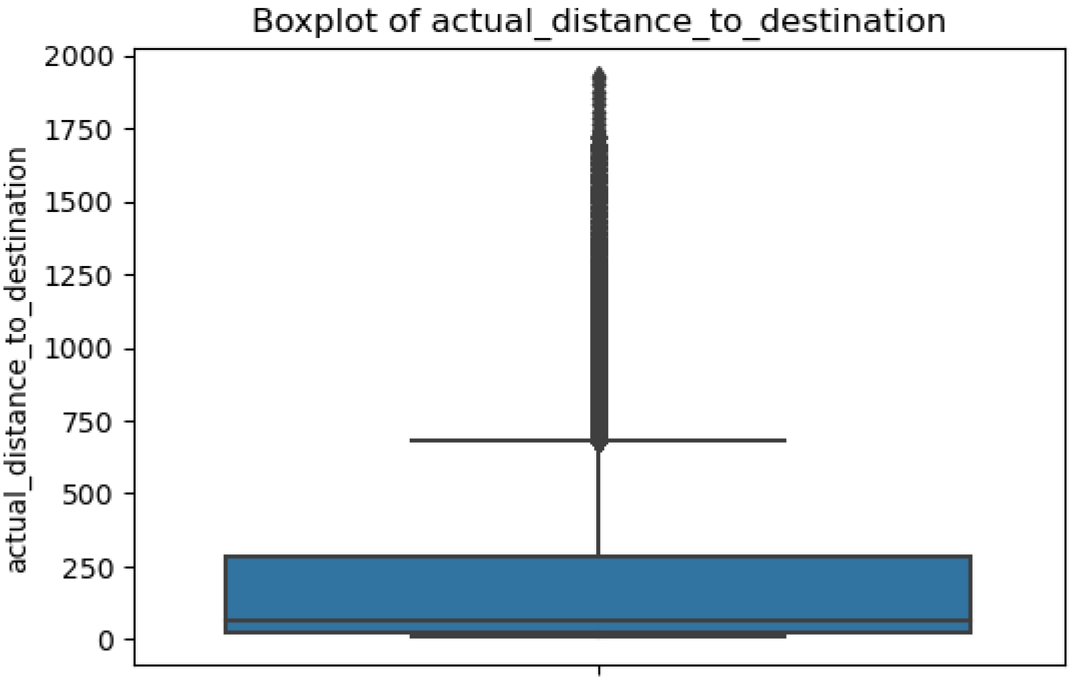
Boxplots -

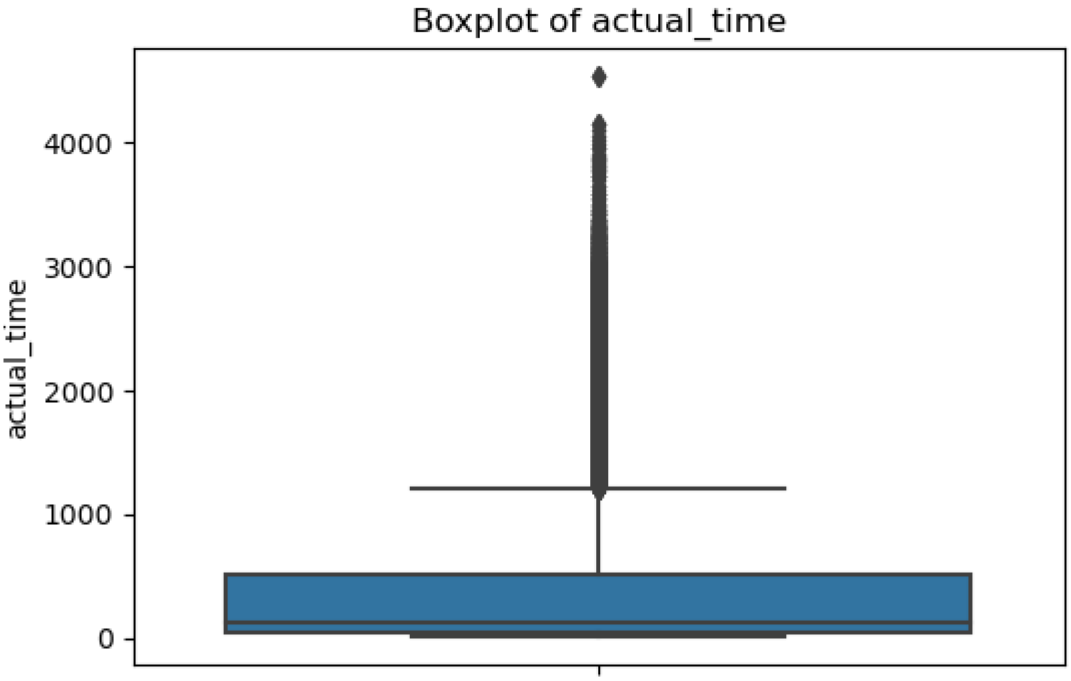
In [53]:

```
# start_scan_to_end_scan
plt.figure(figsize = (6, 4))
sns.boxplot(y = "start_scan_to_end_scan", data = df)
plt.title("Boxplot of start_scan_to_end_scan")
plt.show()

# cutoff_factor
plt.figure(figsize = (6, 4))
sns.boxplot(y = "cutoff_factor", data = df)
plt.title("Boxplot of cutoff_factor")
plt.show()

# actual_distance_to_destination
plt.figure(figsize = (6, 4))
sns.boxplot(y = "actual_distance_to_destination", data = df)
plt.title("Boxplot of actual_distance_to_destination")
plt.show()

# actual_time
plt.figure(figsize = (6, 4))
sns.boxplot(y = "actual_time", data = df)
plt.title("Boxplot of actual_time")
plt.show()
```


Boxplot of start_scan_to_end_scan
start_scan_to_end_scan
8000
7000
6000
5000
4000
3000
2000
1000
0


Boxplot of cutoff_factor
cutoff_factor
2000
1750
1500
1250
1000
750
500
250
0

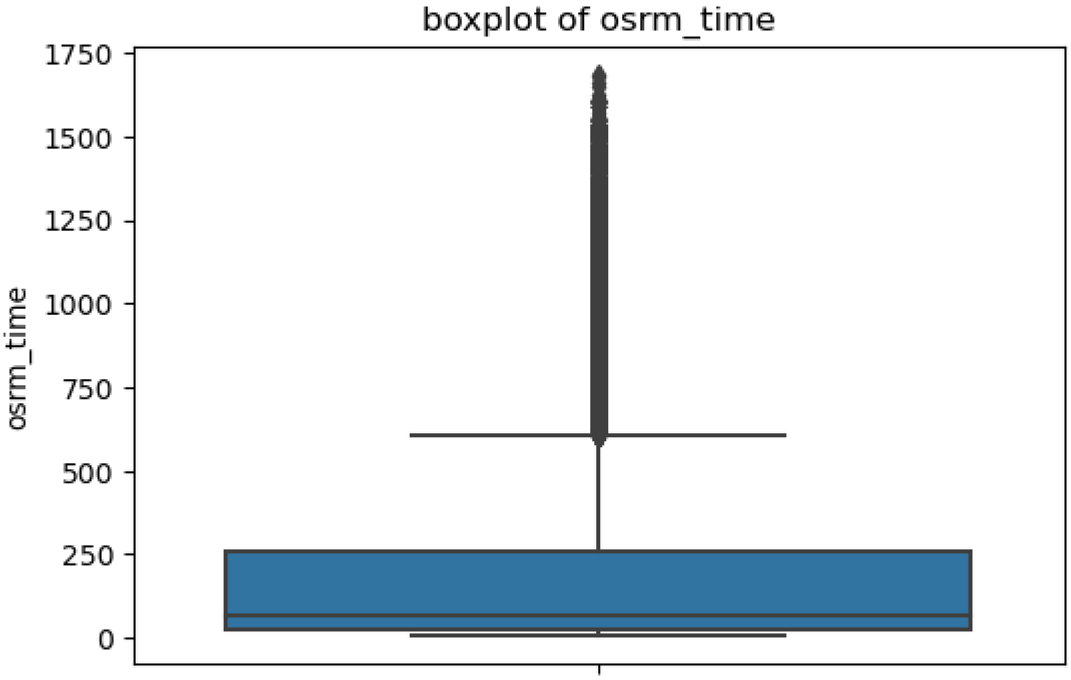


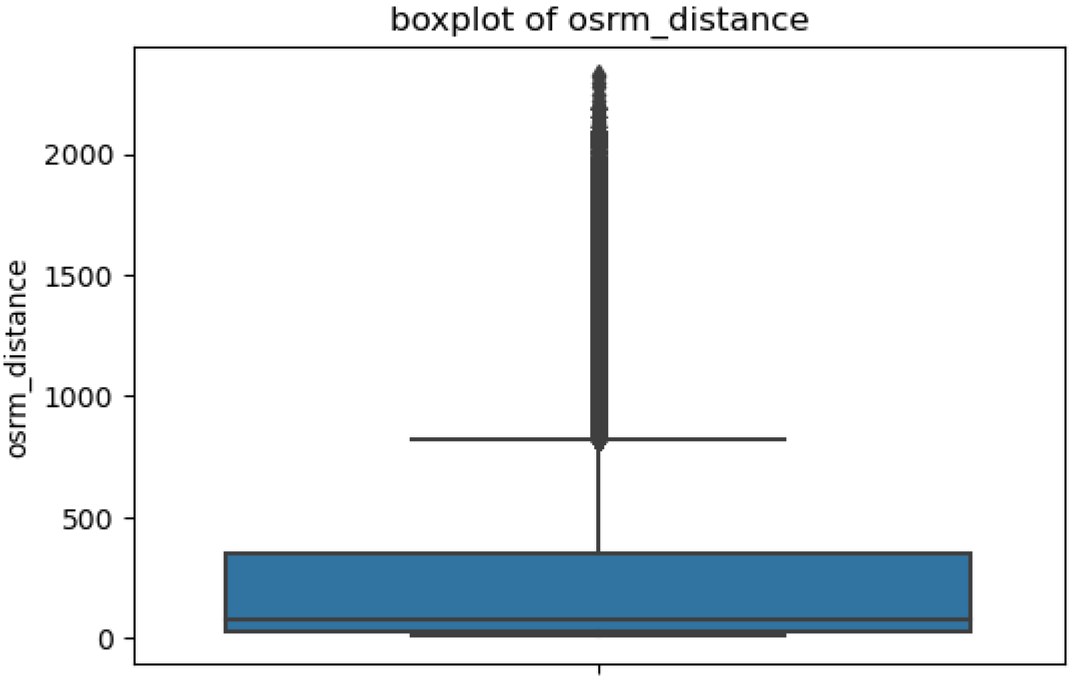


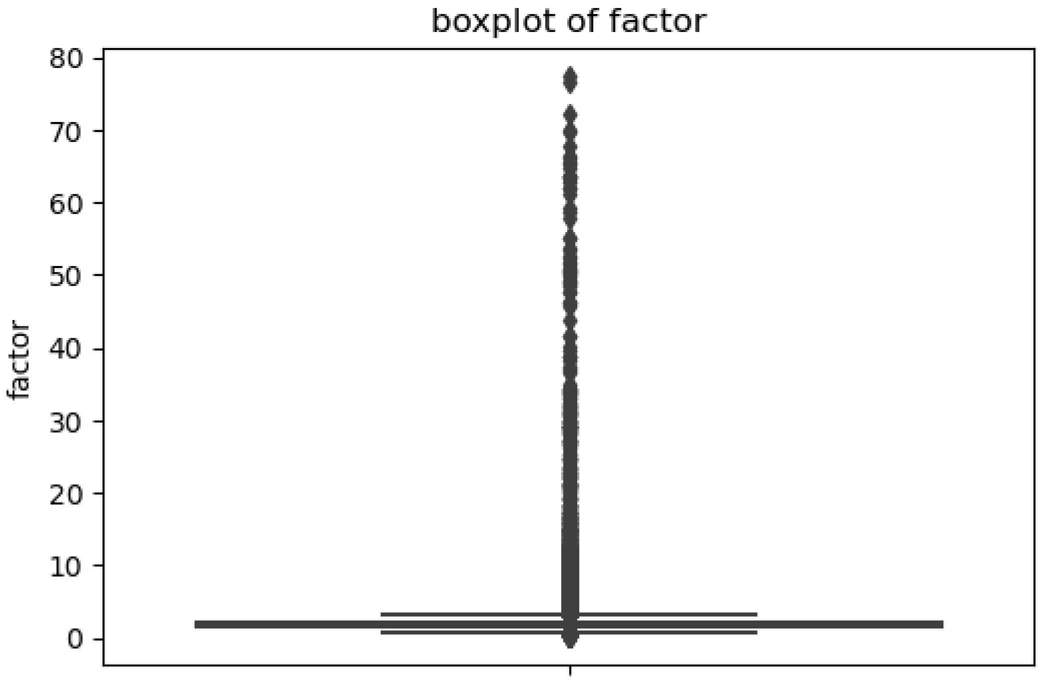
In [54]:

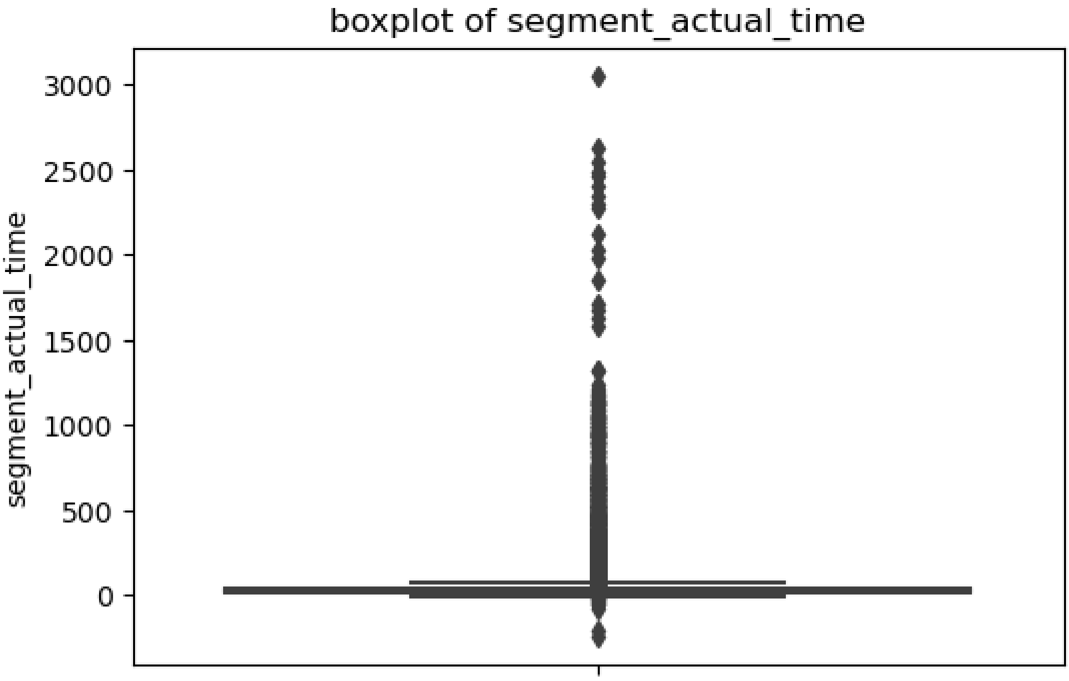
```
# osrm_time
plt.figure(figsize = (6, 4))
sns.boxplot(y = "osrm_time", data = df)
plt.title("boxplot of osrm_time")
plt.show()

# osrm_distance
plt.figure(figsize = (6, 4))
sns.boxplot(y = "osrm_distance", data = df)
plt.title("boxplot of osrm_distance")
plt.show()

# factor
plt.figure(figsize = (6, 4))
sns.boxplot(y = "factor", data = df)
plt.title("boxplot of factor")
plt.show()

# segment_actual_time
```

```
plt.figure(figsize = (6, 4))
sns.boxplot(y = "segment_actual_time", data = df)
plt.title("boxplot of segment_actual_time")
plt.show()
```



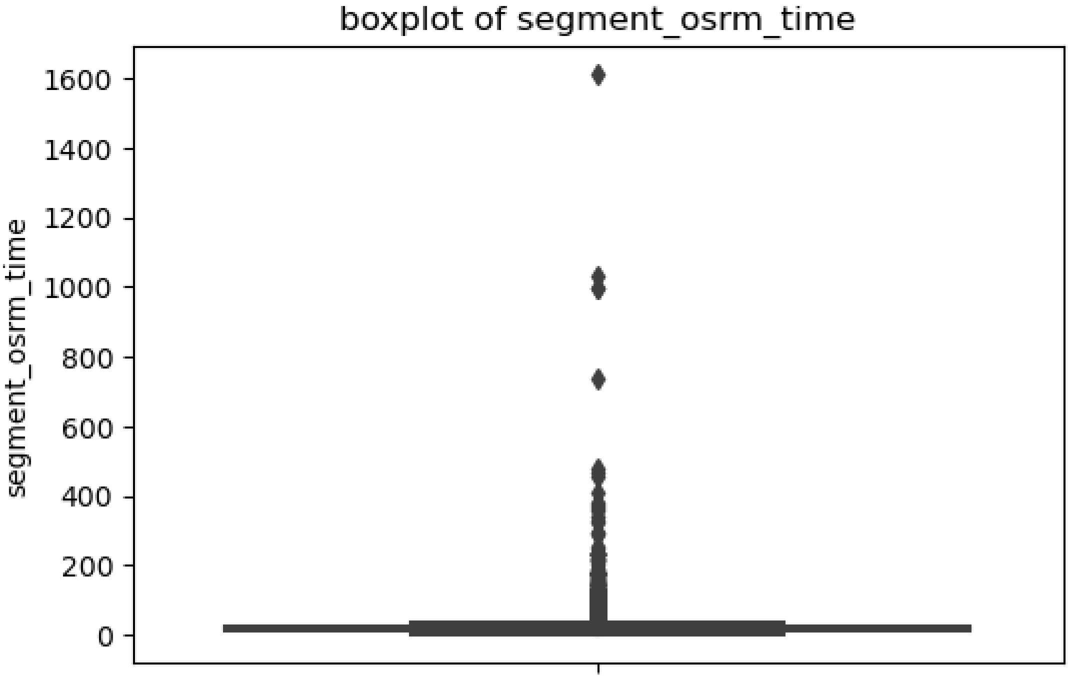


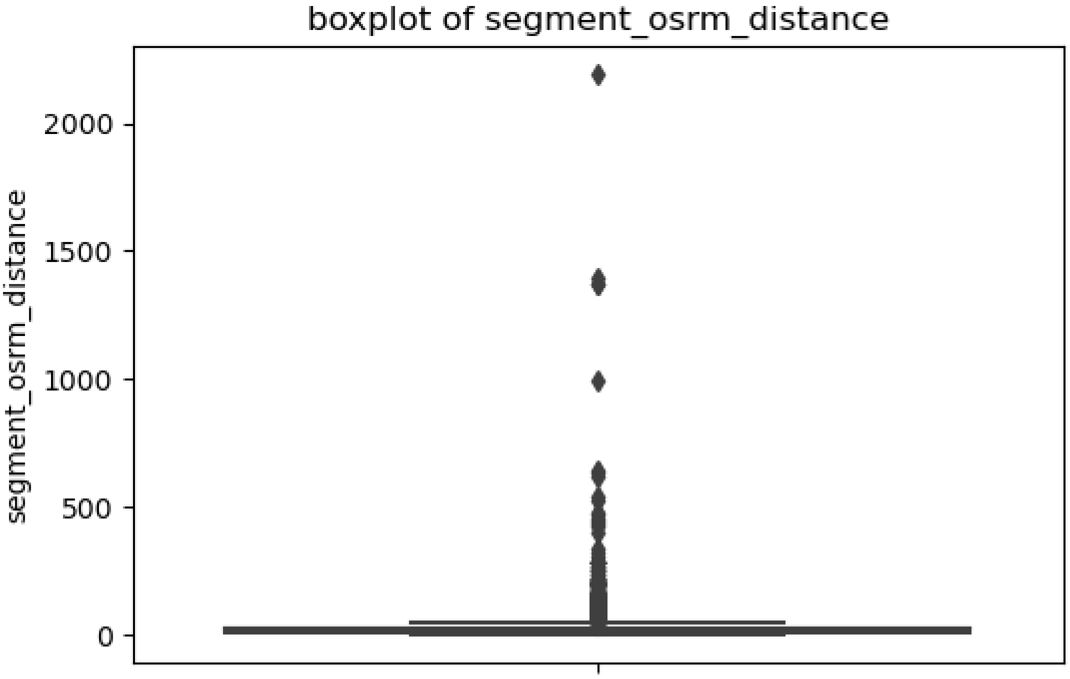


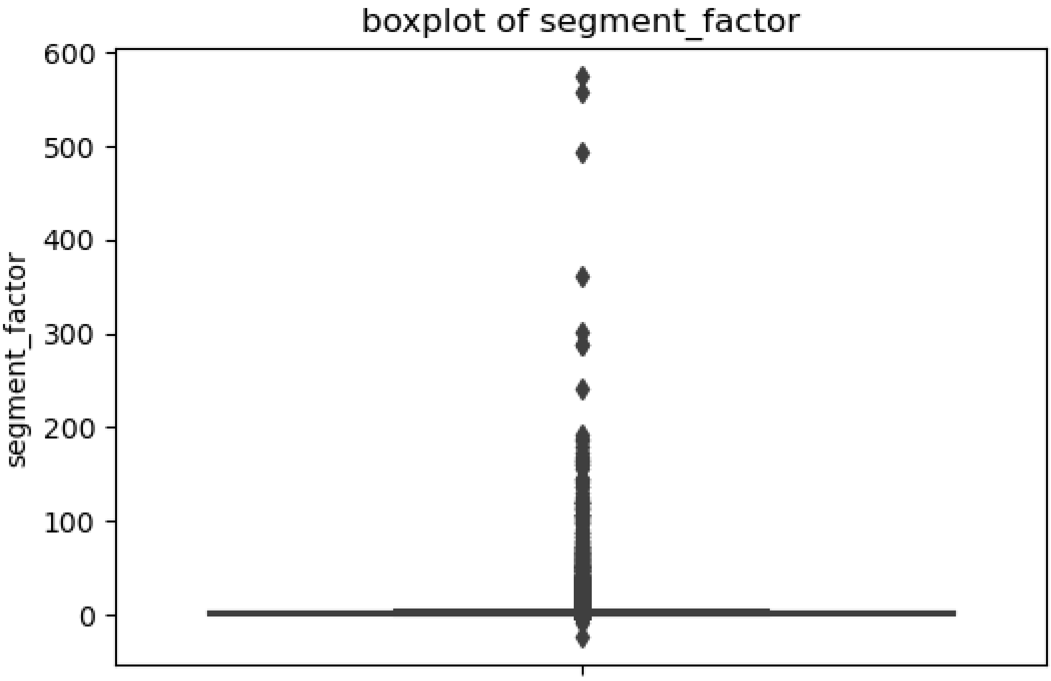


In [58]:

```
# segment_osrm_time
plt.figure(figsize = (6, 4))
sns.boxplot(y = "segment_osrm_time", data = df)
plt.title("boxplot of segment_osrm_time")
plt.show()

# segment_osrm_distance
plt.figure(figsize = (6, 4))
sns.boxplot(y = "segment_osrm_distance", data = df)
plt.title("boxplot of segment_osrm_distance")
plt.show()

# segment_factor
plt.figure(figsize = (6, 4))
sns.boxplot(y = "segment_factor", data = df)
plt.title("boxplot of segment_factor")
plt.show()
```


boxplot of segment_osrm_time
segment_osrm_time
1600
1400
1200
1000
800
600
400
200
0


boxplot of segment_osrm_distance
segment_osrm_distance
2000
1500
1000
500
0



Merging of rows and aggregation field -

In [177…
```
df_copy = df.copy()
```

In [178…
```
df_copy.shape
```

Out[178]:
```
(144316, 37)
```

In [179…
```
df_copy.drop( ['source_center',"source_name","destination_center",

"destination_name","cutoff_timestamp", "od_end_time","od_start_time"], axis
```

In [180…
```
df_copy.columns
```

Out[180]:
```
Index(['data', 'trip_creation_time', 'route_schedule_uuid', 'route_type',
       'trip_uuid', 'start_scan_to_end_scan', 'is_cutoff', 'cutoff_facto
r',
       'actual_distance_to_destination', 'actual_time', 'osrm_time',
       'osrm_distance', 'factor', 'segment_actual_time', 'segment_osrm_tim
e',
       'segment_osrm_distance', 'segment_factor', 'segment_key',
       'segment_actual_time_sum', 'segment_osrm_distance_sum',
       'segment_osrm_time_sum', 'Diff_betw_odstart_odend_1', 'source_cit
y',
       'source_state', 'destination_city', 'destination_state',
       'source_pincode', 'destination_pincode', 'source_city_state',
       'destination_city_state'],
      dtype='object')
```

In [181…
```
df_copy.shape
```

Out[181]:
```
(144316, 30)
```

In [182…
```
actual_time = (df_copy.groupby(["trip_uuid", "start_scan_to_end_scan"])["act
.groupby("trip_uuid")["actual_time"].sum().reset_index())
actual_time.head()
```

Out[182]:

	trip_uuid	actual_time
0	trip-153671041653548748	26.033333
1	trip-153671042288605164	2.383333
2	trip-153671043369099517	55.783333
3	trip-153671046011330457	0.983333
4	trip-153671052974046625	5.683333

In [183…

```
segment_osrm_time = df_copy[["trip_uuid","segment_osrm_time"]].groupby("trip
segment_osrm_time.head()
```

Out[183]:

	trip_uuid	segment_osrm_time
0	trip-153671041653548748	16.800000
1	trip-153671042288605164	1.083333
2	trip-153671043369099517	32.350000
3	trip-153671046011330457	0.266667
4	trip-153671052974046625	1.916667

In [184…

```
segment_actual_time = df_copy.groupby("trip_uuid")["segment_actual_time"].su
segment_actual_time.head()
```

Out[184]:

	trip_uuid	segment_actual_time
0	trip-153671041653548748	25.800000
1	trip-153671042288605164	2.350000
2	trip-153671043369099517	55.133333
3	trip-153671046011330457	0.983333
4	trip-153671052974046625	5.666667

In [185…

```
osrm_time = (df_copy.groupby(["trip_uuid", "start_scan_to_end_scan"])["osrm_
.groupby("trip_uuid")["osrm_time"].sum().reset_index())
osrm_time.head()
```

Out[185]:

	trip_uuid	osrm_time
0	trip-153671041653548748	12.383333
1	trip-153671042288605164	1.133333
2	trip-153671043369099517	29.016667
3	trip-153671046011330457	0.250000
4	trip-153671052974046625	1.950000

In [186…

```
time_btwn_odstart_and_od_end = df_copy.groupby("trip_uuid")["Diff_betw_odsta
time_btwn_odstart_and_od_end.head()
```

Out[186]:

	trip_uuid	Diff_betw_odstart_odend_1
0	trip-153671041653548748	[16.65842298, 21.0100736875]
1	trip-153671042288605164	[2.0463247669444447, 0.9805397955555556]
2	trip-153671043369099517	[51.662059856388886, 13.910648811388889]
3	trip-153671046011330457	[1.6749155866666667]
4	trip-153671052974046625	[2.5335485744444446, 1.3423885633333332, 8.096...

In [187…

```
time_btwn_odstart_and_od_end["time_taken_btwn_odstart_and_od_end"] = (
    time_btwn_odstart_and_od_end["Diff_betw_odstart_odend_1"].apply(sum))
time_btwn_odstart_and_od_end.head()
```

Out[187]:

	trip_uuid	Diff_betw_odstart_odend_1	time_taken_btwn_odstart_and_od_end
0	trip-153671041653548748	[16.65842298, 21.0100736875]	37.668497
1	trip-153671042288605164	[2.0463247669444447, 0.9805397955555556]	3.026865
2	trip-153671043369099517	[51.662059856388886, 13.910648811388889]	65.572709
3	trip-153671046011330457	[1.6749155866666667]	1.674916
4	trip-153671052974046625	[2.5335485744444446, 1.3423885633333332, 8.096...	11.972484

In [188…

```
start_scan_to_end_scan = ((df_copy.groupby("trip_uuid")["start_scan_to_end_s
start_scan_to_end_scan.head()
```

Out[188]:

	trip_uuid	start_scan_to_end_scan
0	trip-153671041653548748	[16.65, 21.0]
1	trip-153671042288605164	[2.033333333333333, 0.9666666666666667]
2	trip-153671043369099517	[51.65, 13.9]
3	trip-153671046011330457	[1.6666666666666667]
4	trip-153671052974046625	[2.533333333333333, 1.3333333333333333, 8.0833...

In [189…

```
start_scan_to_end_scan["start_scan_to_end_scan"] = start_scan_to_end_scan["s
start_scan_to_end_scan.head()
```

Out[189]:

	trip_uuid	start_scan_to_end_scan
0	trip-153671041653548748	37.650000
1	trip-153671042288605164	3.000000
2	trip-153671043369099517	65.550000
3	trip-153671046011330457	1.666667
4	trip-153671052974046625	11.950000

In [190…

```
osrm_distance = (df_copy.groupby(["trip_uuid", "start_scan_to_end_scan"])["o
.groupby("trip_uuid")["osrm_distance"].sum().reset_index())
```

```
osrm_distance.head()
```

Out[190]:

	trip_uuid	osrm_distance
0	trip-153671041653548748	991.3523
1	trip-153671042288605164	85.1110
2	trip-153671043369099517	2372.0852
3	trip-153671046011330457	19.6800
4	trip-153671052974046625	146.7918

In [191…

```
actual_distance_to_destination = (df_copy.groupby(["trip_uuid", "start_scan_
["actual_distance_to_destination"].max().reset_index()
.groupby("trip_uuid")["actual_distance_to_destination"].sum().reset_index()
actual_distance_to_destination.head()
```

Out[191]:

	trip_uuid	actual_distance_to_destination
0	trip-153671041653548748	824.732854
1	trip-153671042288605164	73.186911
2	trip-153671043369099517	1932.273969
3	trip-153671046011330457	17.175274
4	trip-153671052974046625	127.448500

In [192…

```
segment_osrm_distance = ( df_copy[["trip_uuid", "segment_osrm_distance"]]
.groupby("trip_uuid")["segment_osrm_distance"].sum().reset_index() )
segment_osrm_distance.head()
```

Out[192]:

	trip_uuid	segment_osrm_distance
0	trip-153671041653548748	1320.4733
1	trip-153671042288605164	84.1894
2	trip-153671043369099517	2545.2678
3	trip-153671046011330457	19.8766
4	trip-153671052974046625	146.7919

Comparison of distance and time fields -

In [193…

```
distance = segment_osrm_distance.merge(

actual_distance_to_destination.merge(osrm_distance, on = "trip_uuid"),on = '
distance.head()
```

Out[193]:

	trip_uuid	segment_osrm_distance	actual_distance_to_destination	osrm_dist
0	trip-153671041653548748	1320.4733	824.732854	991.
1	trip-153671042288605164	84.1894	73.186911	85
2	trip-153671043369099517	2545.2678	1932.273969	2372.
3	trip-153671046011330457	19.8766	17.175274	19.
4	trip-153671052974046625	146.7919	127.448500	146

In [194…

```
time = segment_osrm_time.merge(
    osrm_time.merge(
        segment_actual_time.merge(
            actual_time.merge(
                time_btwn_odstart_and_od_end.merge(
                    start_scan_to_end_scan, on="trip_uuid"),
                on="trip_uuid"),
            on="trip_uuid"),
        on="trip_uuid"),
    on="trip_uuid")
time.head()
```

Out[194]:

	trip_uuid	segment_osrm_time	osrm_time	segment_actual_time	actual_tim
0	trip-153671041653548748	16.800000	12.383333	25.800000	26.03333
1	trip-153671042288605164	1.083333	1.133333	2.350000	2.38333
2	trip-153671043369099517	32.350000	29.016667	55.133333	55.78333
3	trip-153671046011330457	0.266667	0.250000	0.983333	0.98333
4	trip-153671052974046625	1.916667	1.950000	5.666667	5.68333

In [195…

```
merge_data = time.merge(distance,on="trip_uuid")
merge_data.head()
```

Out[195]:

	trip_uuid	segment_osrm_time	osrm_time	segment_actual_time	actual_tim
0	trip-153671041653548748	16.800000	12.383333	25.800000	26.03333
1	trip-153671042288605164	1.083333	1.133333	2.350000	2.38333
2	trip-153671043369099517	32.350000	29.016667	55.133333	55.78333
3	trip-153671046011330457	0.266667	0.250000	0.983333	0.98333
4	trip-153671052974046625	1.916667	1.950000	5.666667	5.68333

In [196…

```
city = df_copy.groupby("trip_uuid")[["source_city", "destination_city"]].ag
{ "source_city":pd.unique,
"destination_city":pd.unique })

city.head()
```

Out[196]:

	source_city	destination_city
trip_uuid		
trip-153671041653548748	[Bhopal, Kanpur]	[Kanpur, Gurgaon]
trip-153671042288605164	[Tumkur, Doddablpur]	[Doddablpur, Chikblapur]
trip-153671043369099517	[Bangalore, Gurgaon]	[Gurgaon, Chandigarh]
trip-153671046011330457	[Mumbai]	[Mumbai]
trip-153671052974046625	[Bellary, Hospet, Sandur]	[Hospet, Sandur, Bellary]

In [197…

```
state = df_copy.groupby("trip_uuid")[["source_state", "destination_state"]]
{"source_state":pd.unique,
"destination_state":pd.unique })

state.head()
```

Out[197]:

	source_state	destination_state
trip_uuid		
trip-153671041653548748	[Madhya Pradesh, Uttar Pradesh]	[Uttar Pradesh, Haryana]
trip-153671042288605164	[Karnataka]	[Karnataka]
trip-153671043369099517	[Karnataka, Haryana]	[Haryana, Punjab]
trip-153671046011330457	[Hub Maharashtra]	[Maharashtra]
trip-153671052974046625	[Karnataka]	[Karnataka]

In [198…

```
city_state = df_copy.groupby("trip_uuid")[["source_city_state", "destinatio
{ "source_city_state":pd.unique,
"destination_city_state":pd.unique })

city_state.head()
```

Out[198]:

	source_city_state	destination_city_state
trip_uuid		
trip-153671041653548748	[Bhopal Madhya Pradesh, Kanpur Uttar Pradesh]	[Kanpur Uttar Pradesh, Gurgaon Haryana]
trip-153671042288605164	[Tumkur Karnataka, Doddablpur Karnataka]	[Doddablpur Karnataka, Chikblapur Karnataka]
trip-153671043369099517	[Bengaluru Karnataka, Gurgaon Haryana]	[Gurgaon Haryana, Chandigarh Punjab]
trip-153671046011330457	[Mumbai Hub Maharashtra]	[Mumbai Maharashtra]
trip-153671052974046625	[Bellary Karnataka, Hospet Karnataka, Sandur K...	[Hospet Karnataka, Sandur Karnataka, Bellary K...

In [199…

```
locations = city.merge(

city_state.merge(state, on = "trip_uuid", how = "outer"),
on = "trip_uuid", how = "outer")
locations.head()
```

Out[199]:

	source_city	destination_city	source_city_state	destination_city_sta
trip_uuid				
trip-153671041653548748	[Bhopal, Kanpur]	[Kanpur, Gurgaon]	[Bhopal Madhya Pradesh, Kanpur Uttar Pradesh]	[Kanpur Uttar Prade Gurgaon Haryar
trip-153671042288605164	[Tumkur, Doddablpur]	[Doddablpur, Chikblapur]	[Tumkur Karnataka, Doddablpur Karnataka]	[Doddablpur Karnatal Chikblapur Karnatal
trip-153671043369099517	[Bangalore, Gurgaon]	[Gurgaon, Chandigarh]	[Bengaluru Karnataka, Gurgaon Haryana]	[Gurgaon Haryar Chandigarh Punja
trip-153671046011330457	[Mumbai]	[Mumbai]	[Mumbai Hub Maharashtra]	[Mumbai Maharasht
trip-153671052974046625	[Bellary, Hospet, Sandur]	[Hospet, Sandur, Bellary]	[Bellary Karnataka, Hospet Karnataka, Sandur K...	[Hospet Karnatal Sandur Karnatal Bellary K

In [200…

```
route_type = df_copy.groupby("trip_uuid")["route_type"].unique().reset_inde
route_type.head()
```

Out[200]:

	trip_uuid	route_type
0	trip-153671041653548748	[FTL]
1	trip-153671042288605164	[Carting]
2	trip-153671043369099517	[FTL]
3	trip-153671046011330457	[Carting]
4	trip-153671052974046625	[FTL]

In [201…

```
merged_route = route_type.merge(
locations.merge(
merge_data, on = "trip_uuid", how = "outer"),
on="trip_uuid", how = "outer")

merged_route.head()
```

Out[201]:

	trip_uuid	route_type	source_city	destination_city	source_city_state	desti
0	trip-153671041653548748	[FTL]	[Bhopal, Kanpur]	[Kanpur, Gurgaon]	[Bhopal Madhya Pradesh, Kanpur Uttar Pradesh]	[Kan
1	trip-153671042288605164	[Carting]	[Tumkur, Doddablpur]	[Doddablpur, Chikblapur]	[Tumkur Karnataka, Doddablpur Karnataka]	[Dodc Chik
2	trip-153671043369099517	[FTL]	[Bangalore, Gurgaon]	[Gurgaon, Chandigarh]	[Bengaluru Karnataka, Gurgaon Haryana]	[C
3	trip-153671046011330457	[Carting]	[Mumbai]	[Mumbai]	[Mumbai Hub Maharashtra]	[Mur
4	trip-153671052974046625	[FTL]	[Bellary, Hospet, Sandur]	[Hospet, Sandur, Bellary]	[Bellary Karnataka, Hospet Karnataka, Sandur K...	[

In [202…

```
trips = merged_route.copy()
trips["route_type"] = trips["route_type"].apply(lambda x:x[0])
route_to_merge = df_copy.groupby("trip_uuid")["route_schedule_uuid"].unique
trips = trips.merge(route_to_merge, on = "trip_uuid", how = "outer")
trips["route_schedule_uuid"] = trips["route_schedule_uuid"].apply(lambda x:
trips.head()
```

Out[202]:

	trip_uuid	route_type	source_city	destination_city	source_city_state	desti
0	trip-153671041653548748	FTL	[Bhopal, Kanpur]	[Kanpur, Gurgaon]	[Bhopal Madhya Pradesh, Kanpur Uttar Pradesh]	[Kan
1	trip-153671042288605164	Carting	[Tumkur, Doddablpur]	[Doddablpur, Chikblapur]	[Tumkur Karnataka, Doddablpur Karnataka]	[Dodc Chik
2	trip-153671043369099517	FTL	[Bangalore, Gurgaon]	[Gurgaon, Chandigarh]	[Bengaluru Karnataka, Gurgaon Haryana]	[C
3	trip-153671046011330457	Carting	[Mumbai]	[Mumbai]	[Mumbai Hub Maharashtra]	[Mur
4	trip-153671052974046625	FTL	[Bellary, Hospet, Sandur]	[Hospet, Sandur, Bellary]	[Bellary Karnataka, Hospet Karnataka, Sandur K...	[

In [203…

```
trips["source_city"] = trips["source_city"].astype("str").str.strip("[]").s
trips["destination_city"] = trips["destination_city"].astype("str").str.str
trips["source_city_state"] = trips["source_city_state"].astype("str").str.s
trips["destination_city_state"] = trips["destination_city_state"].astype("s
trips["source_state"] = trips["source_state"].astype("str").str.strip("[]")
trips["destination_state"] = trips["destination_state"].astype("str").str.s
```

In [204…

```
trips.head()
```

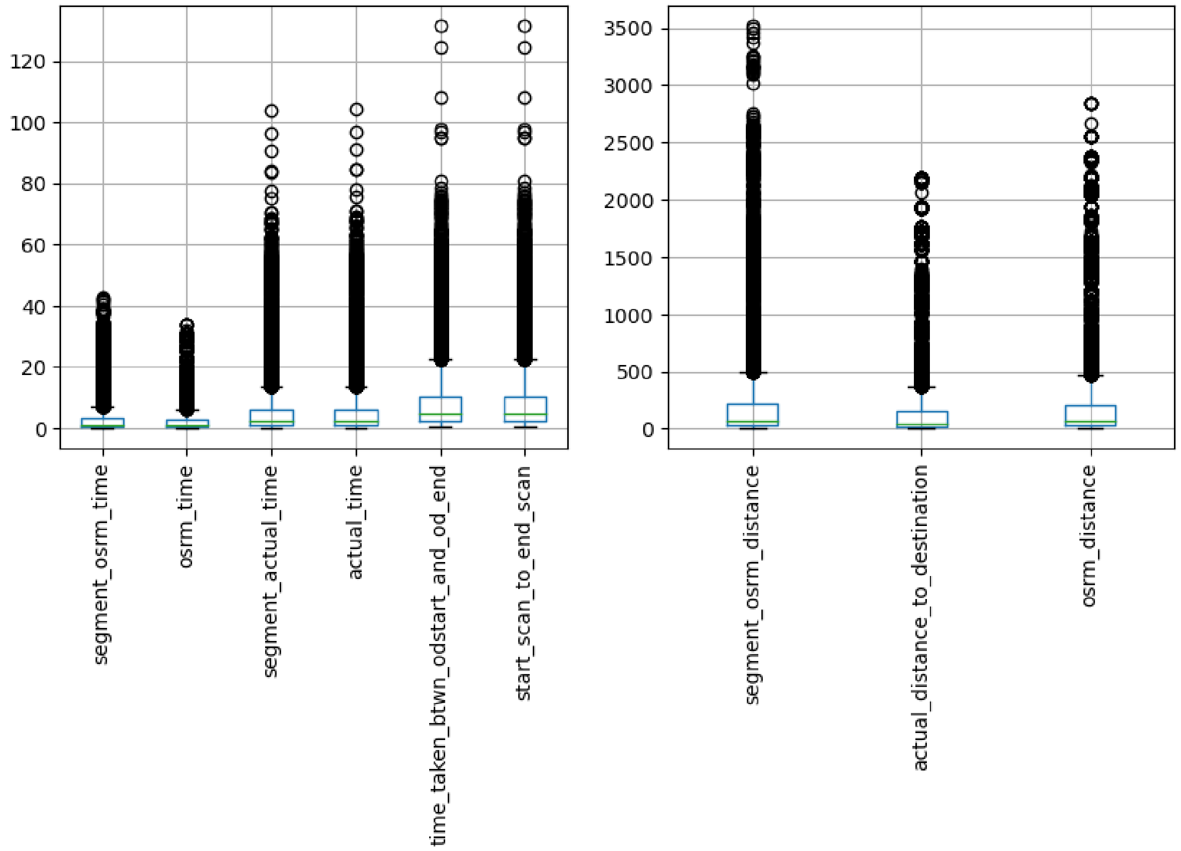
Out[204]:

	trip_uuid	route_type	source_city	destination_city	source_city_state	desti
0	trip-153671041653548748	FTL	Bhopal Kanpur	Kanpur Gurgaon	Bhopal Madhya Pradesh Kanpur Uttar Pradesh	Kar
1	trip-153671042288605164	Carting	Tumkur Doddablpur	Doddablpur Chikblapur	Tumkur Karnataka Doddablpur Karnataka	Dod Chi
2	trip-153671043369099517	FTL	Bangalore Gurgaon	Gurgaon Chandigarh	Bengaluru Karnataka Gurgaon Haryana	(
3	trip-153671046011330457	Carting	Mumbai	Mumbai	Mumbai Hub Maharashtra	Mu
4	trip-153671052974046625	FTL	Bellary Hospet Sandur	Hospet Sandur Bellary	Bellary Karnataka Hospet Karnataka Sandur Karn...	

Outlier Detection and Treatment -

In [205…

```
plt.figure(figsize = (10,4))
plt.subplot(121)
trips[['segment_osrm_time', 'osrm_time', 'segment_actual_time', 'actual_time
'time_taken_btwn_odstart_and_od_end', 'start_scan_to_end_scan']].boxplot()
plt.xticks(rotation = 90)
plt.subplot(122)
trips[['segment_osrm_distance', 'actual_distance_to_destination', 'osrm_dist
plt.xticks(rotation =90)
plt.show()
```



In [206…

```
outliers = trips.copy()
```

In [207…

```
num_col = outliers[['segment_osrm_time', 'osrm_time', 'segment_actual_time',
'time_taken_btwn_odstart_and_od_end', 'start_scan_to_end_scan',
'segment_osrm_distance', 'actual_distance_to_destination', 'osrm_distance']]
```

In [208…

```
num_col.describe()
```

Out[208]:

	segment_osrm_time	osrm_time	segment_actual_time	actual_time	time_taken_
count	14787.000000	14787.000000	14787.000000	14787.000000	
mean	3.008527	2.690634	5.884320	5.931238	
std	5.244655	4.539335	9.272765	9.357566	
min	0.100000	0.100000	0.150000	0.150000	
25%	0.500000	0.483333	1.100000	1.116667	
50%	1.083333	1.000000	2.450000	2.466667	
75%	3.066667	2.800000	6.066667	6.116667	
max	42.733333	33.866667	103.833333	104.416667	

In [209…

```
Q1 = num_col.quantile(0.25)
Q3 = num_col.quantile(0.75)

IQR = Q3 - Q1
```

In [210…

```
outlier_free = num_col[~( (num_col < (Q1 - 1.5 * IQR)) | (num_col > (Q3 + 1.
outlier_free = outlier_free.reset_index(drop=True)
```

In [211…

```
outlier_free
```

Out[211]:

	segment_osrm_time	osrm_time	segment_actual_time	actual_time	time_taken_btw
0	1.083333	1.133333	2.350000	2.383333	
1	0.266667	0.250000	0.983333	0.983333	
2	1.916667	1.950000	5.666667	5.683333	
3	0.383333	0.383333	1.000000	1.016667	
4	0.216667	0.216667	0.400000	0.400000	
...	...	...	...	...	
12718	1.033333	1.033333	1.366667	1.383333	
12719	0.183333	0.200000	0.350000	0.350000	
12720	1.466667	0.900000	4.683333	4.700000	
12721	3.683333	3.066667	4.300000	4.400000	
12722	1.116667	1.133333	4.566667	4.583333	

12723 rows × 9 columns

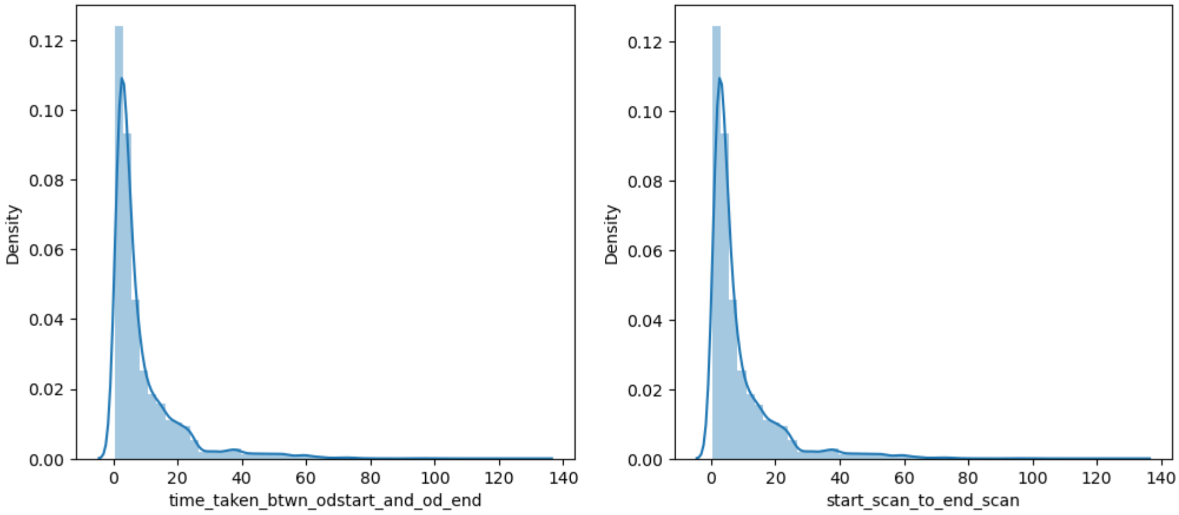
Hypothesis Testing -

1) Hypothesis Test Between (Difference between od_start_time & od_end_time) vs (start_scan_to_end_scan)

- H0 : Mean of time taken is same
- Ha : Mean of time taken is different

In [212…

```
plt.figure(figsize=(12,5))
plt.subplot(121)
sns.distplot(time_btwn_odstart_and_od_end["time_taken_btwn_odstart_and_od_en
plt.subplot(122)
sns.distplot(start_scan_to_end_scan["start_scan_to_end_scan"])
plt.show()
```



In [213…

```
# Ks Test -
# H0 : The distribution are same
# Ha : The distribution are different
ks_2samp(time_btwn_odstart_and_od_end["time_taken_btwn_odstart_and_od_end"]
 ,start_scan_to_end_scan["start_scan_to_end_scan"])
```

Out[213]:

```
KstestResult(statistic=0.004192872117400437, pvalue=0.9994227103139145)
```

In [214…

```
ttest_ind( (time_btwn_odstart_and_od_end["time_taken_btwn_odstart_and_od_enc
```

Out[214]:

```
Ttest_indResult(statistic=0.6804118848245841, pvalue=0.496283097331083)
```

Significance Level = 0.05 (5%)

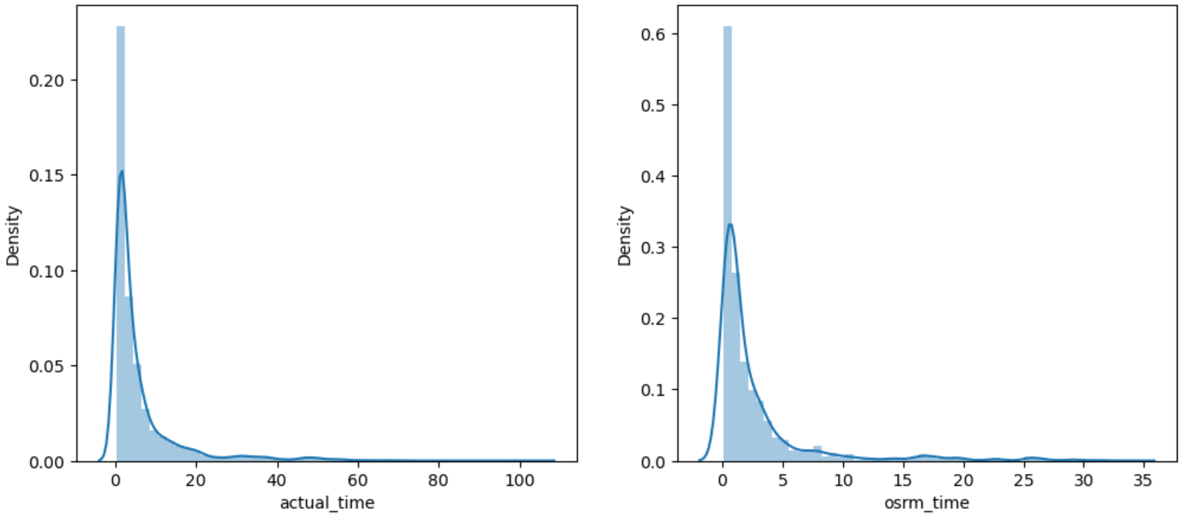
As p_value is greater than significance level, so we **fail to reject null hypothesis** which means Mean of time is same.

2) Hypothesis Test Between (Actual Time) vs (OSRM Time)

- H0 : Mean of OSRM time >= Mean of Actual Time
- Ha : Mean OSRM time < Mean of Actual time

In [215…

```
plt.figure(figsize=(12,5))
plt.subplot(121)
sns.distplot(actual_time["actual_time"])
plt.subplot(122)
sns.distplot(osrm_time["osrm_time"])
plt.show()
```



In [216…

```
# Ks Test -
# H0 : The distribution are same
# Ha : The distribution are different
ks_2samp(actual_time["actual_time"], osrm_time["osrm_time"])
```

Out[216]:

```
KstestResult(statistic=0.2953946033678231, pvalue=0.0)
```

In [217…

```
ttest_ind(actual_time["actual_time"].sample(2000), osrm_time["osrm_time"].sa
```

Out[217]:

```
Ttest_indResult(statistic=14.380884437785868, pvalue=4.642360968682377e-4
6)
```

Significance Level = 0.05 (5%)

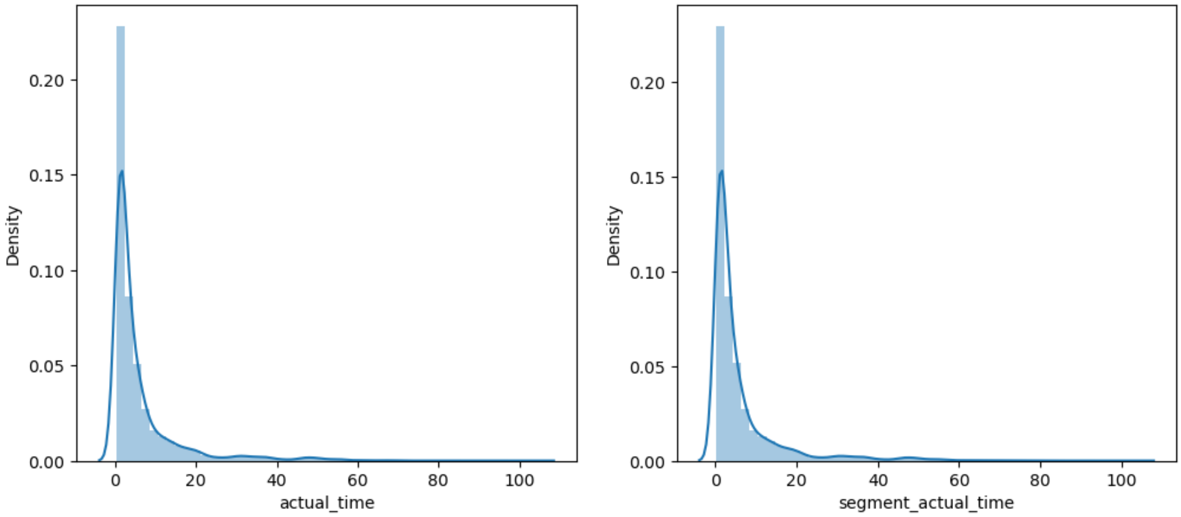
As p_value is lesser than significance level, so we **reject null hypothesis** which means Mean of OSRM time is less than actual time.

3) Hypothesis Test Between (Actual Time) vs (Segment Actual Time)

- H0 : Mean of both times taken are same
- Ha : Mean Segment Actual time is different than Mean of Actual time

In [218…

```
plt.figure(figsize=(12,5))
plt.subplot(121)
sns.distplot(actual_time["actual_time"])
plt.subplot(122)
sns.distplot(segment_actual_time["segment_actual_time"])
plt.show()
```



In [219…

```
# Ks Test -
# H0 : The distribution are same
# Ha : The distribution are different
ks_2samp(actual_time["actual_time"], segment_actual_time["segment_actual_tir
```

Out[219]:

```
KstestResult(statistic=0.00865625211334281, pvalue=0.6334572060090173)
```

In [220…

```
ttest_ind(actual_time["actual_time"].sample(2000), segment_actual_time["segr
```

Out[220]:

```
Ttest_indResult(statistic=-0.994038231131465, pvalue=0.3202644124717193)
```

Significance Level = 0.05 (5%)

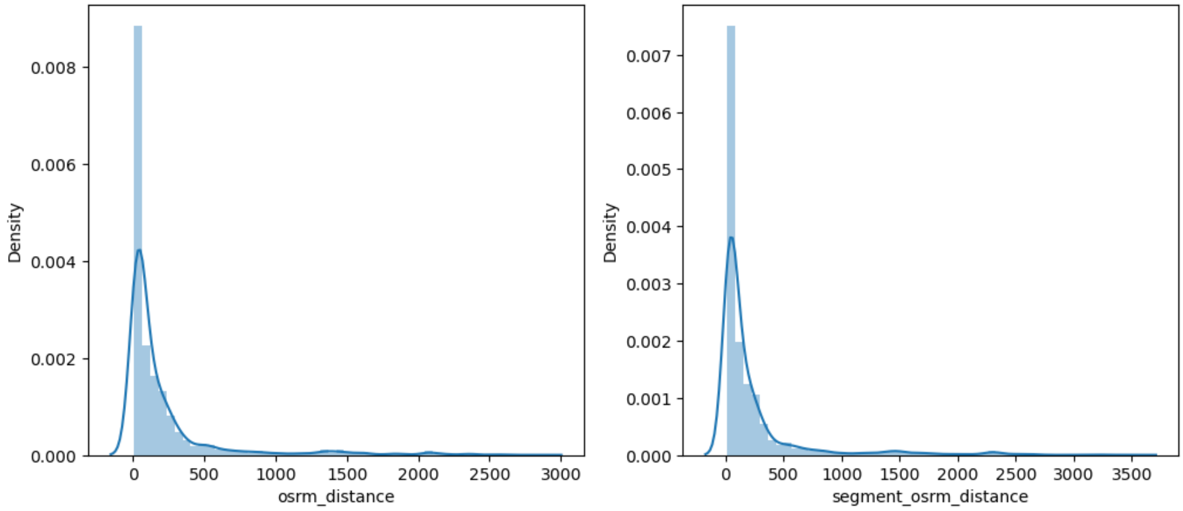
As p_value is greater than significance level, so we **fail to reject null hypothesis** which means Mean of both times are same.

4) Hypothesis Test Between (OSRM Distance) vs (Segment OSRM Distance)

- H0 : Both distances are same
- Ha : Segment OSRM Distance > OSRM Distance

In [221…

```
plt.figure(figsize=(12,5))
plt.subplot(121)
sns.distplot(osrm_distance["osrm_distance"])
plt.subplot(122)
sns.distplot(segment_osrm_distance["segment_osrm_distance"])
plt.show()
```



In [222…

```
# Ks Test -
# H0 : The distribution are same
# Ha : The distribution are different
ks_2samp(osrm_distance["osrm_distance"],segment_osrm_distance["segment_osrm_
```

Out[222]:

```
KstestResult(statistic=0.03949415026712649, pvalue=1.8574496022694056e-10)
```

In [223…

```
ttest_ind(osrm_distance["osrm_distance"].sample(3000), segment_osrm_distance
```

Out[223]:

```
Ttest_indResult(statistic=-0.7858120887052126, pvalue=0.21600431207822052)
```

Significance Level = 0.05 (5%)

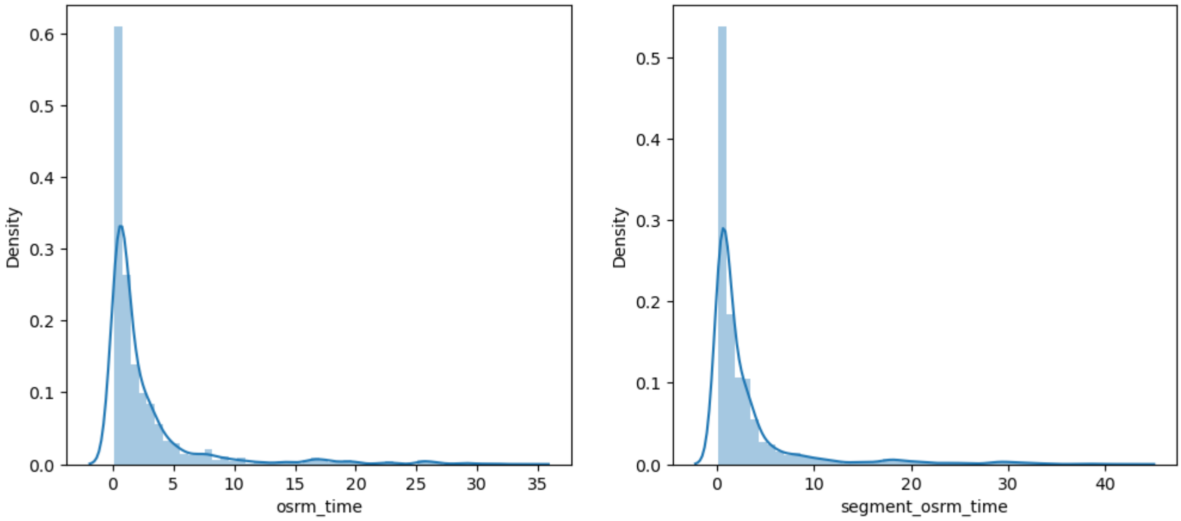
As p_value is lesser than significance level, so we **reject null hypothesis** which means Mean of OSRM distance is less than mean of segment OSRM distance.

5) Hypothesis Test Between (OSRM Time) vs (Segment OSRM Time)

- H0 : Mean time of both are same
- Ha : Mean of Segment OSRM time > Mean of OSRM Time

In [224…

```
plt.figure(figsize=(12,5))
plt.subplot(121)
sns.distplot(osrm_time["osrm_time"])
plt.subplot(122)
sns.distplot(segment_osrm_time["segment_osrm_time"])
plt.show()
```



In [225…

```
# Ks Test -
# H0 : The distribution are same
# Ha : The distribution are different
ks_2samp(osrm_time["osrm_time"],segment_osrm_time["segment_osrm_time"])
```

Out[225]:

```
KstestResult(statistic=0.03482788936227765, pvalue=3.15417156438414e-08)
```

In [226…

```
ttest_ind( (osrm_time["osrm_time"].sample(2000)), (segment_osrm_time["segmen
```

Out[226]:

```
Ttest_indResult(statistic=-0.7769755773829506, pvalue=0.2186095736107081)
```

Significance Level = 0.05 (5%)

As p_value is lesser than significance level, so we **reject null hypothesis** which means Mean of OSRM time is less than mean of segment OSRM time.

Handling Categorical Values -

In [227…

```
trips["trip_uuid"].nunique()
```

Out[227]:

```
14787
```

In [228…

```
trips["route_type"].value_counts()
```

Out[228]:

```
Carting    8906
FTL        5881
Name: route_type, dtype: int64
```

In [229…

```
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
col = 'route_type'
trips[col] = label_encoder.fit_transform(trips[col])
```

In [230…

```
trips["route_type"].value_counts()
```

Out[230]:

```
0    8906
1    5881
Name: route_type, dtype: int64
```

Column Normalization/Standardization -

In [231…

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
```

```
std_data = scaler.fit_transform(num_col)
std_data = pd.DataFrame(std_data, columns = num_col.columns)
std_data
```

Out[231]:

	segment_osrm_time	osrm_time	segment_actual_time	actual_time	time_taken_btw
0	2.629714	2.135341	2.147833	2.148291	
1	-0.367090	-0.343080	-0.381163	-0.379161	
2	5.594737	5.799732	5.311326	5.327644	
3	-0.522809	-0.537681	-0.528553	-0.528778	
4	-0.208192	-0.163165	-0.023473	-0.026493	
...	...	...	...	...	
14782	-0.376623	-0.365110	-0.487212	-0.486030	
14783	-0.538699	-0.548697	-0.596856	-0.596461	
14784	-0.293997	-0.394484	-0.129522	-0.131581	
14785	0.128670	0.082842	-0.170863	-0.163642	
14786	-0.360734	-0.343080	-0.142104	-0.144049	

14787 rows × 9 columns

In [232…

```
Number_of_trips_between_cities = (df_copy.groupby(["source_city_state", "des
Number_of_trips_between_cities
```

Out[232]:

	source_city_state	destination_city_state	trip_uuid
0	Bengaluru Karnataka	Bengaluru Karnataka	1369
1	Bhiwandi Maharashtra	Mumbai Maharashtra	512
2	Mumbai Maharashtra	Mumbai Maharashtra	361
3	Hyderabad Telangana	Hyderabad Telangana	308
4	Mumbai Maharashtra	Bhiwandi Maharashtra	282
...	...	...	...
2298	Jamui Bihar	Munger Bihar	1
2299	Shahjhnpur Uttar Pradesh	Tilhar Uttar Pradesh	1
2300	Nashik Maharashtra	Shrirampur Maharashtra	1
2301	Jamui Bihar	KharagpurBR Bihar	1
2302	Abohar Punjab	Malout Punjab	1

2303 rows × 3 columns

Insights from Data - 1) 14,817 Trips happened between source and destination.

2) The data belongs to September and October months from the year 2018.

3) 60% of the trips routes are Carting & remaining 40% consists of FTL

4) From above table, we can observe that Mumbai Maharashtra, Delhi, Gurgaon(Haryana), Bengaluru Karnataka, Hyderabad Telangana, Chennai Tamil Nadu,

Ahmedabad Gujarat, Pune Maharashtra, Chandigarh Chandigarh and Kolkata West Bengal are some cities have higest amount of trips happening states with in the city.

Insights from Hypothesis Testing - 1) Average time_taken_btwn_odstart_and_od_end for population is equal to Average start_scan_to_end_scan for population.

2) Mean of actual time is higher than Mean of the OSRM estimated time for delivery.

3) Population average for Actual Time taken to complete delivery trip and segment actual time are same.

4) Average of OSRM distance for population is less than average of segment OSRM distance.

5) Population Mean OSRM time is less than Population Mean segment OSRM time.

Recommnedations -

1) As we can see the difference in OSRM distance & time with respect to actual distance & time, we need to check the configuration settings of routing engines.

2) With states & cities having heavy traffics, company needs to allocate extra resources so that delivery will be on time in any seasons.

3) Carting should be provided within city range and Heavy Trucks should be assigned for inter-state delivery, so that we can optimize the delivery time.

4) Optimization can be carried out in scanning time of both ends which is start scanning time and end scanning time so that the delivery time can be equated to the OSRM time.

5) Company should increase connectivity between different tier cities so that it will reduce the delivery time and there by increase revenue.

```
In [ ]:
```