Delhivery-Dataset@Dhanureddy

June 2, 2024

1 Delhivery - 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

Problem statement

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

- I Cleaned, sanitized and manipulated data to get useful features out of raw fields.
- Made a sense out of the raw data to help the data science team to build forecasting models on it.

scope:

- This dataset focuses on feature engineering to transform the data into a suitable format for effective analysis and machine learning predictions.

[131]:
```
import numpy as np
import pandas as pd
```

1. Loading and reading the dataset

[132]:
```
df = pd.read_csv("delhivery_data.csv")
```

[133]:
```
df.head()
```

[133]:

	data	trip_creation_time	\
0	training	2018-09-20 02:35:36.476840	
1	training	2018-09-20 02:35:36.476840	
2	training	2018-09-20 02:35:36.476840	
3	training	2018-09-20 02:35:36.476840	
4	training	2018-09-20 02:35:36.476840	

```
                                route_schedule_uuid route_type  \
0  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
1  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
2  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
3  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
4  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting

                 trip_uuid source_center                source_name  \
0  trip-153741093647649320  IND388121AAA  Anand_VUNagar_DC (Gujarat)
1  trip-153741093647649320  IND388121AAA  Anand_VUNagar_DC (Gujarat)
2  trip-153741093647649320  IND388121AAA  Anand_VUNagar_DC (Gujarat)
3  trip-153741093647649320  IND388121AAA  Anand_VUNagar_DC (Gujarat)
4  trip-153741093647649320  IND388121AAA  Anand_VUNagar_DC (Gujarat)

  destination_center             destination_name  \
0       IND388620AAB  Khambhat_MotvdDPP_D (Gujarat)
1       IND388620AAB  Khambhat_MotvdDPP_D (Gujarat)
2       IND388620AAB  Khambhat_MotvdDPP_D (Gujarat)
3       IND388620AAB  Khambhat_MotvdDPP_D (Gujarat)
4       IND388620AAB  Khambhat_MotvdDPP_D (Gujarat)

               od_start_time  …            cutoff_timestamp  \
0  2018-09-20 03:21:32.418600  …         2018-09-20 04:27:55
1  2018-09-20 03:21:32.418600  …         2018-09-20 04:17:55
2  2018-09-20 03:21:32.418600  …  2018-09-20 04:01:19.505586
3  2018-09-20 03:21:32.418600  …         2018-09-20 03:39:57
4  2018-09-20 03:21:32.418600  …         2018-09-20 03:33:55

   actual_distance_to_destination  actual_time  osrm_time osrm_distance  \
0                       10.435660         14.0       11.0       11.9653
1                       18.936842         24.0       20.0       21.7243
2                       27.637279         40.0       28.0       32.5395
3                       36.118028         62.0       40.0       45.5620
4                       39.386040         68.0       44.0       54.2181

     factor  segment_actual_time  segment_osrm_time  segment_osrm_distance  \
0  1.272727                 14.0               11.0                11.9653
1  1.200000                 10.0                9.0                 9.7590
2  1.428571                 16.0                7.0                10.8152
3  1.550000                 21.0               12.0                13.0224
4  1.545455                  6.0                5.0                 3.9153

   segment_factor
0        1.272727
1        1.111111
2        2.285714
3        1.750000
```

```
4        1.200000

[5 rows x 24 columns]
```

2. Finding the shape of the *dataset*

[134]:
```
df.shape
```

[134]:
```
(144867, 24)
```

3. Checking for presence of null values

[135]:
```
df.isna().sum()
```

[135]:
```
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
```

4. Imputing "Unknown" in place of null values present in columns of the dataset.

[136]:
```
df = df.fillna("Unknown")
```

5. Verifying the presence of null values after imputation to ensure they have been properly handled.

[137]:
```
df.isna().sum()
```

[137]:
```
data                                0
trip_creation_time                  0
route_schedule_uuid                 0
route_type                          0
trip_uuid                           0
source_center                       0
source_name                         0
destination_center                  0
destination_name                    0
od_start_time                       0
od_end_time                         0
start_scan_to_end_scan              0
is_cutoff                           0
cutoff_factor                       0
cutoff_timestamp                    0
actual_distance_to_destination      0
actual_time                         0
osrm_time                           0
osrm_distance                       0
factor                              0
segment_actual_time                 0
segment_osrm_time                   0
segment_osrm_distance               0
segment_factor                      0
dtype: int64
```

6. Merging three unique categorical columns to create a new feature out of it.

[138]:
```
df['segment_key'] = df['trip_uuid'] + '_' + df['source_center'] + '_' +␣
↪df['destination_center']
```

7. Calculating cumsum() of key columns using group by "Segment_key" operation.

[139]:
```
df['segment_actual_time_sum'] = df.
↪groupby('segment_key')['segment_actual_time'].cumsum()
df['segment_osrm_distance_sum'] = df.
↪groupby('segment_key')['segment_osrm_distance'].cumsum()
df['segment_osrm_time_sum'] = df.groupby('segment_key')['segment_osrm_time'].
↪cumsum()
```

[140]:
```
df.head()
```

[140]:

	data	trip_creation_time	\
0	training	2018-09-20 02:35:36.476840	
1	training	2018-09-20 02:35:36.476840	
2	training	2018-09-20 02:35:36.476840	
3	training	2018-09-20 02:35:36.476840	

```
4  training  2018-09-20 02:35:36.476840

                               route_schedule_uuid route_type  \
0  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
1  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
2  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
3  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting
4  thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3…    Carting

                trip_uuid source_center                source_name  \
0  trip-153741093647649320  IND388121AAA  Anand_VUNagar_DC (Gujarat)
1  trip-153741093647649320  IND388121AAA  Anand_VUNagar_DC (Gujarat)
2  trip-153741093647649320  IND388121AAA  Anand_VUNagar_DC (Gujarat)
3  trip-153741093647649320  IND388121AAA  Anand_VUNagar_DC (Gujarat)
4  trip-153741093647649320  IND388121AAA  Anand_VUNagar_DC (Gujarat)

  destination_center              destination_name  \
0       IND388620AAB  Khambhat_MotvdDPP_D (Gujarat)
1       IND388620AAB  Khambhat_MotvdDPP_D (Gujarat)
2       IND388620AAB  Khambhat_MotvdDPP_D (Gujarat)
3       IND388620AAB  Khambhat_MotvdDPP_D (Gujarat)
4       IND388620AAB  Khambhat_MotvdDPP_D (Gujarat)

               od_start_time  … osrm_distance    factor  \
0  2018-09-20 03:21:32.418600  …       11.9653  1.272727
1  2018-09-20 03:21:32.418600  …       21.7243  1.200000
2  2018-09-20 03:21:32.418600  …       32.5395  1.428571
3  2018-09-20 03:21:32.418600  …       45.5620  1.550000
4  2018-09-20 03:21:32.418600  …       54.2181  1.545455

   segment_actual_time  segment_osrm_time segment_osrm_distance  \
0                 14.0               11.0               11.9653
1                 10.0                9.0                9.7590
2                 16.0                7.0               10.8152
3                 21.0               12.0               13.0224
4                  6.0                5.0                3.9153

   segment_factor                                      segment_key  \
0        1.272727  trip-153741093647649320_IND388121AAA_IND388620AAB
1        1.111111  trip-153741093647649320_IND388121AAA_IND388620AAB
2        2.285714  trip-153741093647649320_IND388121AAA_IND388620AAB
3        1.750000  trip-153741093647649320_IND388121AAA_IND388620AAB
4        1.200000  trip-153741093647649320_IND388121AAA_IND388620AAB

   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

[5 rows x 28 columns]
```

8. Checking dataset info like number of columns present and their datatypes.

[141]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 144867 entries, 0 to 144866
Data columns (total 28 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                     144867 non-null  object
 7   destination_center              144867 non-null  object
 8   destination_name                144867 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
 24  segment_key                     144867 non-null  object
 25  segment_actual_time_sum         144867 non-null  float64
 26  segment_osrm_distance_sum       144867 non-null  float64
 27  segment_osrm_time_sum           144867 non-null  float64
dtypes: bool(1), float64(13), int64(1), object(13)
memory usage: 30.0+ MB
```

9. Creating a dictionary named 'create_segment_dict' for running an operation to specify how to aggregate multiple rows at Segment_level .

[142]:
```
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': 'first',
    'destination_name': 'first',
    'od_start_time': 'first',
    'od_end_time': 'last',
    'start_scan_to_end_scan': 'sum',
    'is_cutoff': 'first',
    'cutoff_factor': 'first',
    'cutoff_timestamp': 'first',
    'actual_distance_to_destination': 'sum',
    'actual_time': 'sum',
    'osrm_time': 'sum',
    'osrm_distance': 'sum',
    'factor': 'mean',
    'segment_actual_time': 'sum',
    'segment_osrm_time': 'sum',
    'segment_osrm_distance': 'sum',
    'segment_factor': 'mean',
    'segment_actual_time_sum': 'last',
    'segment_osrm_distance_sum': 'last',
    'segment_osrm_time_sum': 'last'
}
```

10. Aggregating rows based on the parameters provided to each column

[143]:
```
segment_df = df.groupby('segment_key').agg(create_segment_dict).reset_index()
segemt_df = segment_df.sort_values(by = ['segment_key', 'od_end_time'])
```

[144]:
```
segment_df.head(5)
```

[144]:

	segment_key	data	trip_creation_time
0	trip-153671041653548748_IND209304AAA_IND000000ACB	training	2018-09-12 00:00:16.535741
1	trip-153671041653548748_IND462022AAA_IND209304AAA	training	2018-09-12 00:00:16.535741
2	trip-153671042288605164_IND561203AAB_IND562101AAA	training	
3	trip-153671042288605164_IND572101AAA_IND561203AAB	training	
4	trip-153671043369099517_IND000000ACB_IND160002AAC	training	

```
2  2018-09-12 00:00:22.886430
3  2018-09-12 00:00:22.886430
4  2018-09-12 00:00:33.691250

                                route_schedule_uuid route_type  \
0  thanos::sroute:d7c989ba-a29b-4a0b-b2f4-288cdc6…        FTL
1  thanos::sroute:d7c989ba-a29b-4a0b-b2f4-288cdc6…        FTL
2  thanos::sroute:3a1b0ab2-bb0b-4c53-8c59-eb2a2c0…    Carting
3  thanos::sroute:3a1b0ab2-bb0b-4c53-8c59-eb2a2c0…    Carting
4  thanos::sroute:de5e208e-7641-45e6-8100-4d9fb1e…        FTL

                 trip_uuid source_center                          source_name  \
0  trip-153671041653548748  IND209304AAA  Kanpur_Central_H_6 (Uttar Pradesh)
1  trip-153671041653548748  IND462022AAA  Bhopal_Trnsport_H (Madhya Pradesh)
2  trip-153671042288605164  IND561203AAB   Doddablpur_ChikaDPP_D (Karnataka)
3  trip-153671042288605164  IND572101AAA       Tumkur_Veersagr_I (Karnataka)
4  trip-153671043369099517  IND000000ACB       Gurgaon_Bilaspur_HB (Haryana)

  destination_center                   destination_name  … osrm_time  \
0       IND000000ACB       Gurgaon_Bilaspur_HB (Haryana)  …    3464.0
1       IND209304AAA  Kanpur_Central_H_6 (Uttar Pradesh)  …    4323.0
2       IND562101AAA   Chikblapur_ShntiSgr_D (Karnataka)  …      55.0
3       IND561203AAB   Doddablpur_ChikaDPP_D (Karnataka)  …     155.0
4       IND160002AAC      Chandigarh_Mehmdpur_H (Punjab)  …    1427.0

  osrm_distance    factor  segment_actual_time  segment_osrm_time  \
0     4540.1261  1.741964                728.0              534.0
1     6037.6386  2.150702                820.0              474.0
2       60.3157  1.746424                 46.0               26.0
3      209.1151  1.875977                 95.0               39.0
4     1975.7409  1.737898                608.0              231.0

  segment_osrm_distance  segment_factor  segment_actual_time_sum  \
0              670.6205        1.893007                    728.0
1              649.8528        2.134213                    820.0
2               28.1995        1.795767                     46.0
3               55.9899        2.912963                     95.0
4              317.7408        2.326577                    608.0

   segment_osrm_distance_sum  segment_osrm_time_sum
0                   670.6205                  534.0
1                   649.8528                  474.0
2                    28.1995                   26.0
3                    55.9899                   39.0
4                   317.7408                  231.0

[5 rows x 28 columns]
```

11. Formatting key columns to extract new columns, and splitting existing columns into more effective way for draining patterns out of it.

[145]:
```
# Changing key numeric columns to datatime to extact new columns out of it.

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

[146]:
```
# Calculating differnece betwen od_end_time and od_start_time

segment_df['od_time_diff_hr'] = (segment_df['od_end_time'] -␣
 ↪segment_df['od_start_time']).dt.total_seconds()/3600
segment_df = segment_df.drop(columns = ['od_start_time', 'od_end_time'])
```

[147]:
```
# Extract features from destination_name
destination_split = segment_df['destination_name'].str.extract(r'(?:(?
 ↪P<destination_city>[^_]+)_)?(?:(?P<destination_place_code>[^_]+)_)?(?:(?
 ↪P<destination_code>[^ ]+))? \((?P<destination_state>[^)]+)\)')
segment_df = pd.concat([segment_df, destination_split], axis=1)
segment_df = segment_df.drop(columns=['destination_name'])
```

[148]:
```
# Extract features from source_name
source_split = segment_df['source_name'].str.extract(
    r'(?:(?P<source_city>[^_]+)_)?(?:(?P<source_place_code>[^_]+)_)?(?:(?
 ↪P<source_code>[^ ]+))? \((?P<source_state>[^)]+)\)'
)
segment_df = pd.concat([segment_df, source_split], axis=1)
segment_df = segment_df.drop(columns=['source_name'])
```

[149]:
```
# Exracting new columns out of existing column

segment_df['trip_creation_year'] = segment_df['trip_creation_time'].dt.year
segment_df['trip_creation_month'] = segment_df['trip_creation_time'].dt.month
segment_df['trip_creation_day'] = segment_df['trip_creation_time'].dt.day
segment_df['trip_creation_hour'] = segment_df['trip_creation_time'].dt.hour
segment_df['trip_creation_minute'] = segment_df['trip_creation_time'].dt.minute
segment_df['trip_creation_second'] = segment_df['trip_creation_time'].dt.second
segment_df = segment_df.drop(columns=['trip_creation_time'])
```

[150]:
```
segment_df.head()
```

[150]:

	segment_key	data	\
0	trip-153671041653548748_IND209304AAA_IND000000ACB	training	
1	trip-153671041653548748_IND462022AAA_IND209304AAA	training	
2	trip-153671042288605164_IND561203AAB_IND562101AAA	training	

```
3  trip-153671042288605164_IND572101AAA_IND561203AAB  training
4  trip-153671043369099517_IND000000ACB_IND160002AAC  training

                              route_schedule_uuid route_type  \
0  thanos::sroute:d7c989ba-a29b-4a0b-b2f4-288cdc6…        FTL
1  thanos::sroute:d7c989ba-a29b-4a0b-b2f4-288cdc6…        FTL
2  thanos::sroute:3a1b0ab2-bb0b-4c53-8c59-eb2a2c0…    Carting
3  thanos::sroute:3a1b0ab2-bb0b-4c53-8c59-eb2a2c0…    Carting
4  thanos::sroute:de5e208e-7641-45e6-8100-4d9fb1e…        FTL

                trip_uuid source_center destination_center  \
0  trip-153671041653548748  IND209304AAA       IND000000ACB
1  trip-153671041653548748  IND462022AAA       IND209304AAA
2  trip-153671042288605164  IND561203AAB       IND562101AAA
3  trip-153671042288605164  IND572101AAA       IND561203AAB
4  trip-153671043369099517  IND000000ACB       IND160002AAC

   start_scan_to_end_scan  is_cutoff  cutoff_factor  … source_city  \
0                 22680.0       True             22  …      Kanpur
1                 20979.0       True             22  …      Bhopal
2                   174.0       True              9  …  Doddablpur
3                   732.0       True              9  …      Tumkur
4                 10008.0       True             22  …     Gurgaon

   source_place_code  source_code    source_state  trip_creation_year  \
0            Central          H_6   Uttar Pradesh                2018
1          Trnsport            H  Madhya Pradesh                2018
2          ChikaDPP            D       Karnataka                2018
3          Veersagr            I       Karnataka                2018
4          Bilaspur           HB         Haryana                2018

   trip_creation_month  trip_creation_day  trip_creation_hour  \
0                    9                 12                   0
1                    9                 12                   0
2                    9                 12                   0
3                    9                 12                   0
4                    9                 12                   0

   trip_creation_minute  trip_creation_second
0                     0                    16
1                     0                    16
2                     0                    22
3                     0                    22
4                     0                    33

[5 rows x 38 columns]
```

12. Creating a new dictionary named 'create_trip_dict' for running an operation to specify how to aggregate multiple rows at trip_level (trip_uuid).

[151]:
```
create_trip_dict = {
    'segment_key': 'first',                     # Keep the first segment key
    'data': 'first',                            # Keep the first data entry
    'route_schedule_uuid': 'first',             # Keep the first route schedule␣
↪UUID
    'route_type': 'first',                      # Keep the first route type
    'source_center': 'first',                   # Keep the first source center
    'destination_center': 'last',               # Keep the last destination␣
↪center
    'start_scan_to_end_scan': 'sum',            # Sum of scan times
    'is_cutoff': 'first',                       # Keep the first is_cutoff value
    'cutoff_factor': 'sum',                     # Sum of cutoff factors
    'cutoff_timestamp': 'first',                # Keep the first cutoff timestamp
    'actual_distance_to_destination': 'sum',    # Sum of actual distances
    'actual_time': 'sum',                       # Sum of actual times
    'osrm_time': 'sum',                         # Sum of OSRM times
    'osrm_distance': 'sum',                     # Sum of OSRM distances
    'factor': 'mean',                           # Mean of factors
    'segment_actual_time': 'sum',               # Sum of segment actual times
    'segment_osrm_time': 'sum',                 # Sum of segment OSRM times
    'segment_osrm_distance': 'sum',             # Sum of segment OSRM distances
    'segment_factor': 'mean',                   # Mean of segment factors
    'segment_actual_time_sum': 'sum',           # Sum of segment actual times
    'segment_osrm_distance_sum': 'sum',         # Sum of segment OSRM distances
    'segment_osrm_time_sum': 'sum',             # Sum of segment OSRM times
    'od_time_diff_hr': 'sum',                   # Sum of od_time_diff_hr
    'destination_city': 'first',                # Keep the first destination city
    'destination_place_code': 'first',          # Keep the first destination␣
↪place code
    'destination_code': 'first',                # Keep the first destination code
    'destination_state': 'first',               # Keep the first destination␣
↪state
    'source_city': 'first',                     # Keep the first source city
    'source_place_code': 'first',               # Keep the first source place␣
↪code
    'source_code': 'first',                     # Keep the first source code
    'source_state': 'first',                    # Keep the first source state
    'trip_creation_year': 'first',              # Keep the first trip creation␣
↪year
    'trip_creation_month': 'first',             # Keep the first trip creation␣
↪month
    'trip_creation_day': 'first',               # Keep the first trip creation␣
↪day
```

```
    'trip_creation_hour': 'first',              # Keep the first trip creation␣
 ↪hour
    'trip_creation_minute': 'first',            # Keep the first trip creation␣
 ↪minute
    'trip_creation_second': 'first'             # Keep the first trip creation␣
 ↪second
}
```

13. Performing aggregation based on defined parameters under trip_level

[152]:
```
trip_df = segment_df.groupby('trip_uuid').agg(create_trip_dict).reset_index()
```

[153]:
```
trip_df.head()
```

[153]:

	trip_uuid	segment_key	\
0	trip-153671041653548748	trip-153671041653548748_IND209304AAA_IND000000ACB	
1	trip-153671042288605164	trip-153671042288605164_IND561203AAB_IND562101AAA	
2	trip-153671043369099517	trip-153671043369099517_IND000000ACB_IND160002AAC	
3	trip-153671046011330457	trip-153671046011330457_IND400072AAB_IND401104AAA	
4	trip-153671052974046625	trip-153671052974046625_IND583101AAA_IND583201AAA	

	data	route_schedule_uuid	route_type	\
0	training	thanos::sroute:d7c989ba-a29b-4a0b-b2f4-288cdc6…	FTL	
1	training	thanos::sroute:3a1b0ab2-bb0b-4c53-8c59-eb2a2c0…	Carting	
2	training	thanos::sroute:de5e208e-7641-45e6-8100-4d9fb1e…	FTL	
3	training	thanos::sroute:f0176492-a679-4597-8332-bbd1c7f…	Carting	
4	training	thanos::sroute:d9f07b12-65e0-4f3b-bec8-df06134…	FTL	

	source_center	destination_center	start_scan_to_end_scan	is_cutoff	\
0	IND209304AAA	IND209304AAA	43659.0	True	
1	IND561203AAB	IND561203AAB	906.0	True	
2	IND000000ACB	IND000000ACB	248631.0	True	
3	IND400072AAB	IND401104AAA	200.0	True	
4	IND583101AAA	IND583119AAA	1586.0	True	

	cutoff_factor	…	source_city	source_place_code	source_code	\
0	44	…	Kanpur	Central	H_6	
1	18	…	Doddablpur	ChikaDPP	D	
2	44	…	Gurgaon	Bilaspur	HB	
3	9	…	None	None	Hub	
4	66	…	Bellary	WrdN1DPP	Dc	

	source_state	trip_creation_year	trip_creation_month	trip_creation_day	\
0	Uttar Pradesh	2018	9	12	
1	Karnataka	2018	9	12	
2	Haryana	2018	9	12	
3	Maharashtra	2018	9	12	

```
4      Karnataka                2018                    9                  12

   trip_creation_hour  trip_creation_minute  trip_creation_second
0                   0                     0                    16
1                   0                     0                    22
2                   0                     0                    33
3                   0                     1                     0
4                   0                     2                     9

[5 rows x 38 columns]
```

14. Statistical summary of trip_df after doing some significant feature engineering to columns in the datset.

[154]:
```
trip_df_summary = trip_df.describe()
trip_df_summary
```

[154]:
```
       start_scan_to_end_scan  cutoff_factor  actual_distance_to_destination  \
count            14817.000000   14817.000000                    14817.000000
mean              9398.345482      30.183438                     2288.554169
std              33701.706672      38.216430                     8798.110164
min                 26.000000       9.000000                        9.002461
25%                408.000000       9.000000                       49.597866
50%                985.000000      18.000000                      134.059655
75%               2826.000000      44.000000                      463.956888
max             396800.000000    1722.000000                    85110.885093

         actual_time     osrm_time  osrm_distance        factor  \
count   14817.000000  14817.000000   14817.000000  14817.000000
mean     4076.333941   2091.007289    2784.231856      2.434336
std     15216.870041   7956.882351   10759.101819      1.872130
min         9.000000      6.000000       9.072900      0.610398
25%       142.000000     62.000000      65.738600      1.644608
50%       348.000000    167.000000     173.593600      1.994536
75%      1063.000000    516.000000     607.677400      2.580882
max    167920.000000  76953.000000  102415.868000     70.000000

       segment_actual_time  segment_osrm_time  segment_osrm_distance  …  \
count         14817.000000       14817.000000           14817.000000  …
mean            353.892286         180.949787             223.201161  …
std             556.247965         314.542047             416.628374  …
min               9.000000           6.000000               9.072900  …
25%              66.000000          31.000000              32.654500  …
50%             147.000000          65.000000              70.154400  …
75%             367.000000         185.000000             218.802400  …
max            6230.000000        2564.000000            3523.632400  …
```

```
       segment_actual_time_sum  segment_osrm_distance_sum  \
count             14817.000000               14817.000000
mean                353.892286                 223.201161
std                 556.247965                 416.628374
min                   9.000000                   9.072900
25%                  66.000000                  32.654500
50%                 147.000000                  70.154400
75%                 367.000000                 218.802400
max                6230.000000                3523.632400

       segment_osrm_time_sum  od_time_diff_hr  trip_creation_year  \
count           14817.000000     14817.000000             14817.0
mean              180.949787         8.863253              2018.0
std               314.542047        10.986409                 0.0
min                 6.000000         0.391024              2018.0
25%                31.000000         2.498843              2018.0
50%                65.000000         4.679427              2018.0
75%               185.000000        10.636651              2018.0
max              2564.000000       131.642533              2018.0

       trip_creation_month  trip_creation_day  trip_creation_hour  \
count         14817.000000       14817.000000        14817.000000
mean              9.120672          18.370790           12.449821
std               0.325757           7.893275            7.986553
min               9.000000           1.000000            0.000000
25%               9.000000          14.000000            4.000000
50%               9.000000          19.000000           14.000000
75%               9.000000          25.000000           20.000000
max              10.000000          30.000000           23.000000

       trip_creation_minute  trip_creation_second
count          14817.000000          14817.000000
mean              29.853614             29.854626
std               17.398730             17.329229
min                0.000000              0.000000
25%               15.000000             15.000000
50%               30.000000             30.000000
75%               45.000000             45.000000
max               59.000000             59.000000

[8 rows x 21 columns]
```

15. Using “IQR” method to calculate upper and lower boundaries inorder to find outliers.

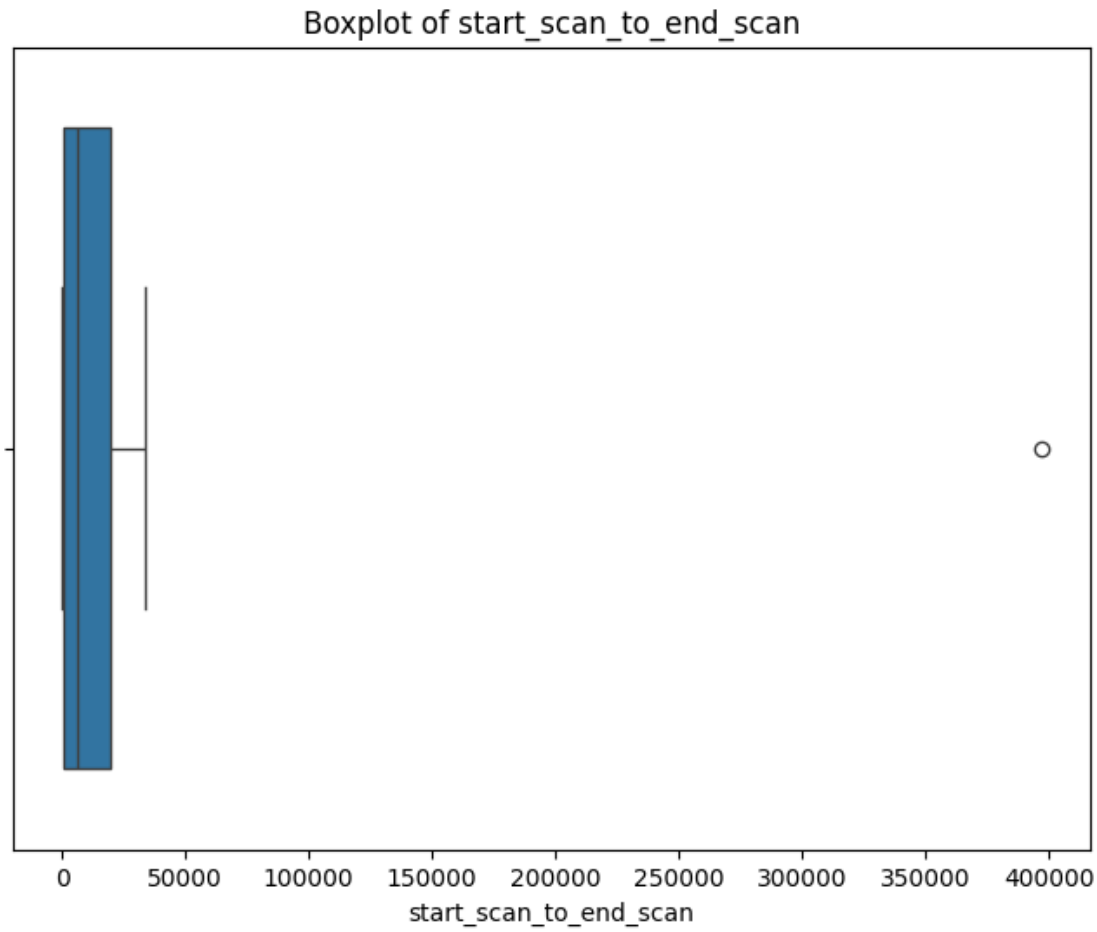
[155]:
```
Q1 = trip_df_summary.loc["25%"]
Q3 = trip_df_summary.loc["75%"]
```

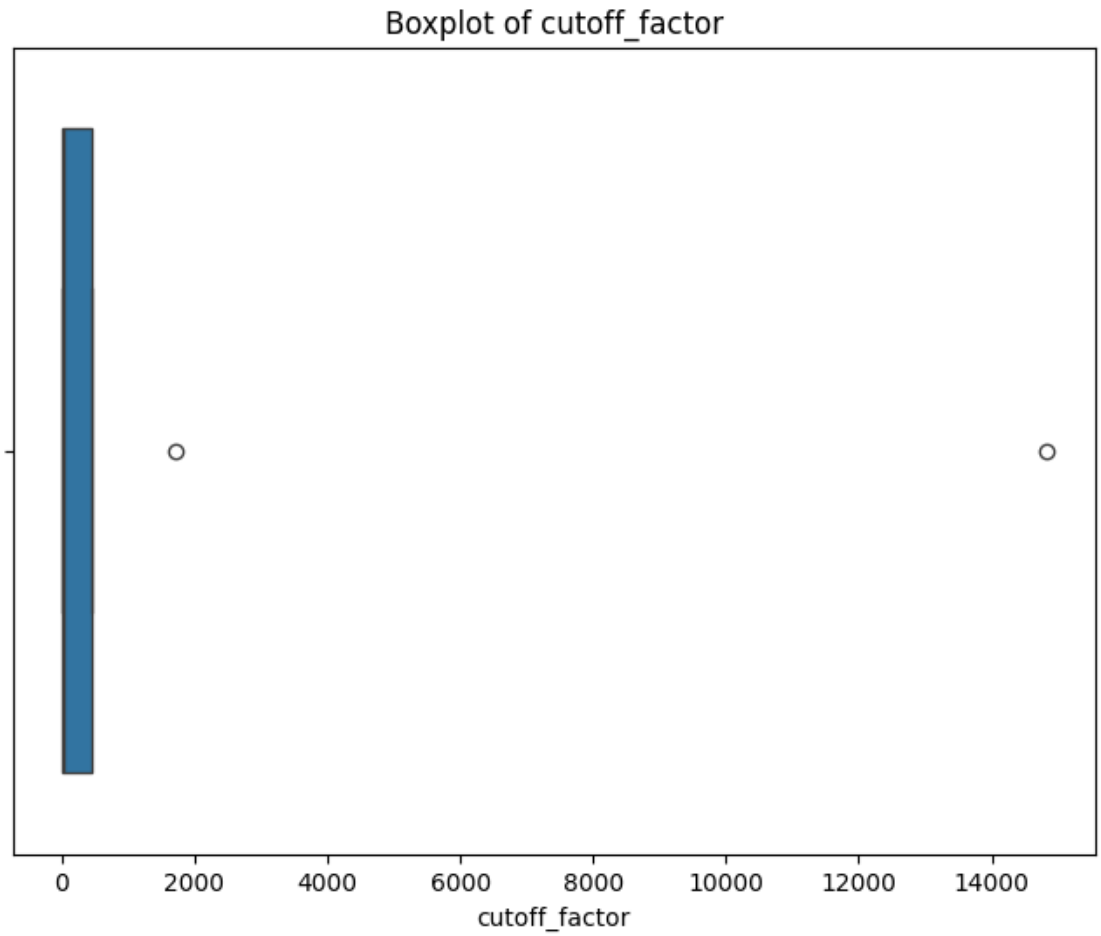
```
IQR = Q3 - Q1
print(IQR)
```

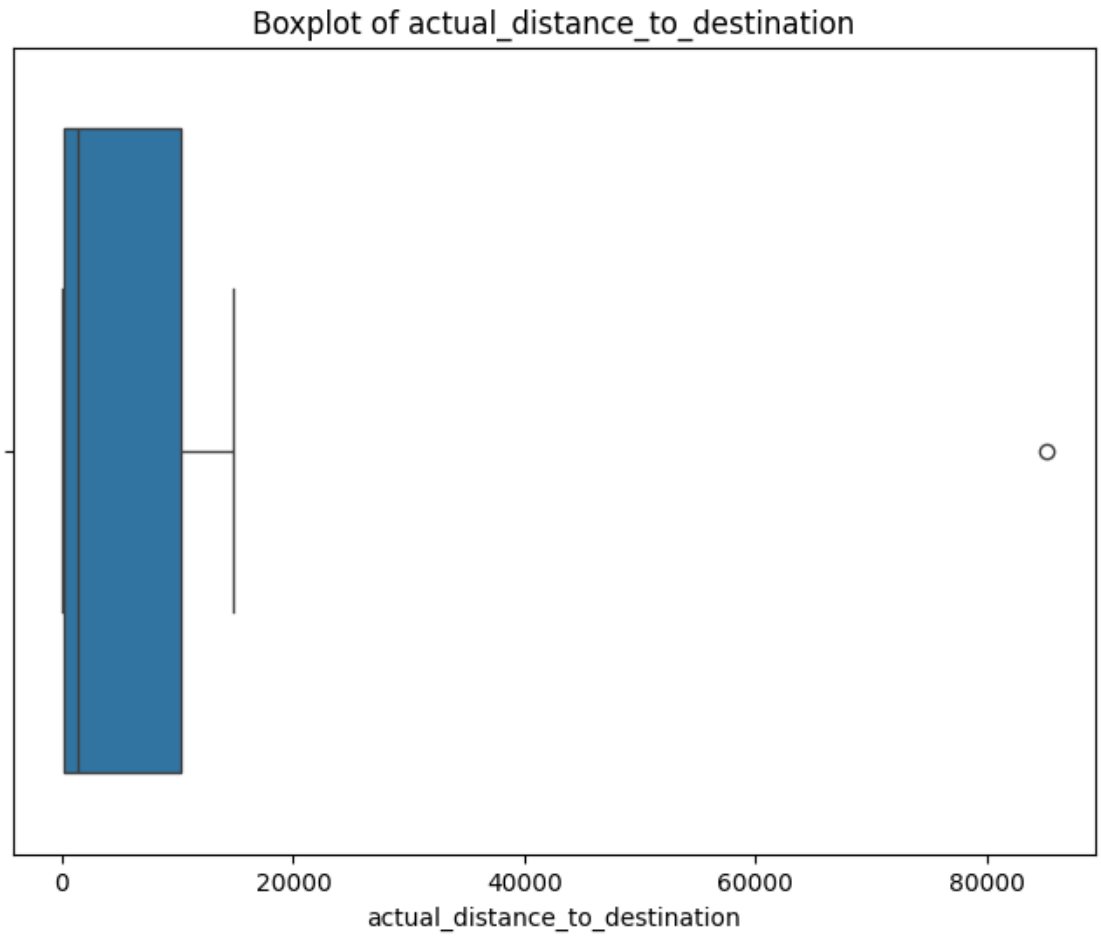
```
start_scan_to_end_scan            2418.000000
cutoff_factor                       35.000000
actual_distance_to_destination     414.359022
actual_time                        921.000000
osrm_time                          454.000000
osrm_distance                      541.938800
factor                               0.936275
segment_actual_time                301.000000
segment_osrm_time                  154.000000
segment_osrm_distance              186.147900
segment_factor                       1.042132
segment_actual_time_sum            301.000000
segment_osrm_distance_sum          186.147900
segment_osrm_time_sum              154.000000
od_time_diff_hr                      8.137808
trip_creation_year                   0.000000
trip_creation_month                  0.000000
trip_creation_day                   11.000000
trip_creation_hour                  16.000000
trip_creation_minute                30.000000
trip_creation_second                30.000000
dtype: float64
```

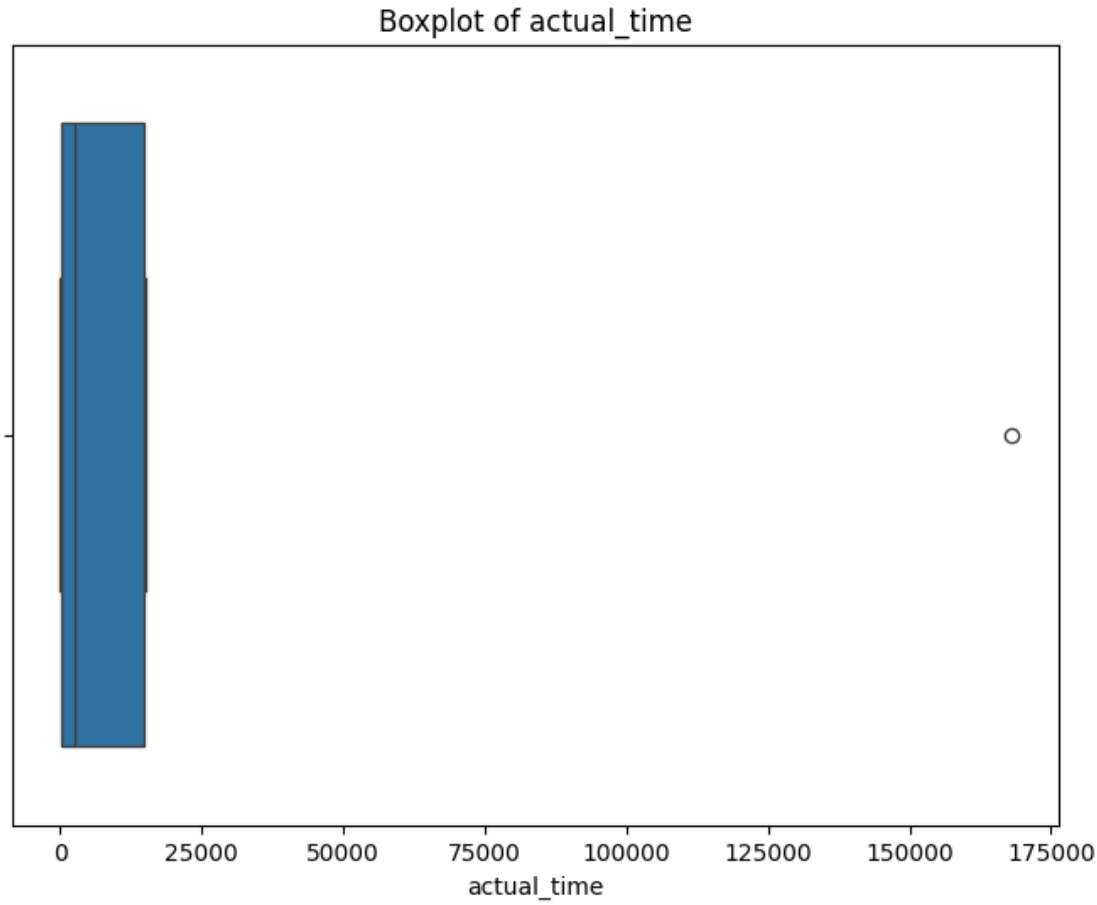
16. Visualization of Outliers presence in every column using Box plots

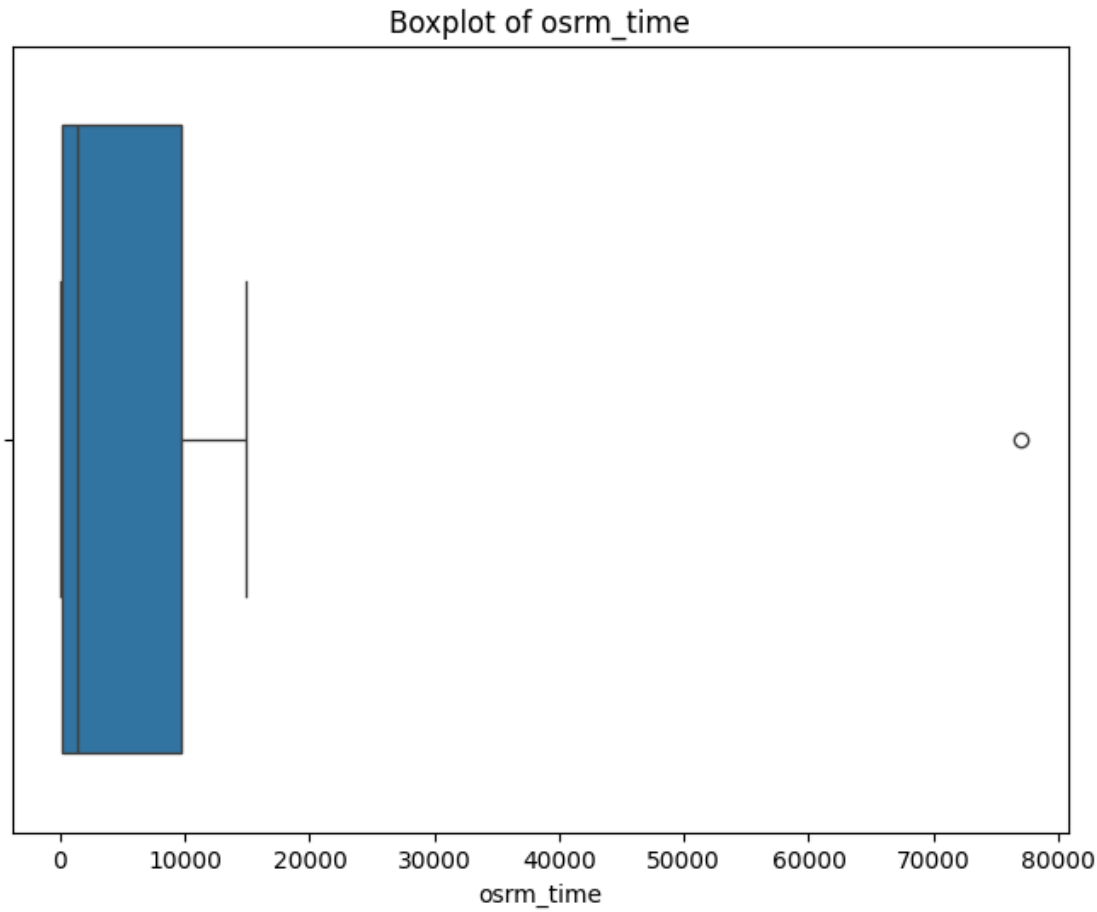
[107]:
```
import seaborn as sns
import matplotlib.pyplot as plt

for col in trip_df_summary:
  plt.figure(figsize = (8, 6))
  sns.boxplot(x = trip_df_summary[col])
  plt.title(f"Boxplot of {col}")
  plt.xlabel(col)
  plt.show()
```

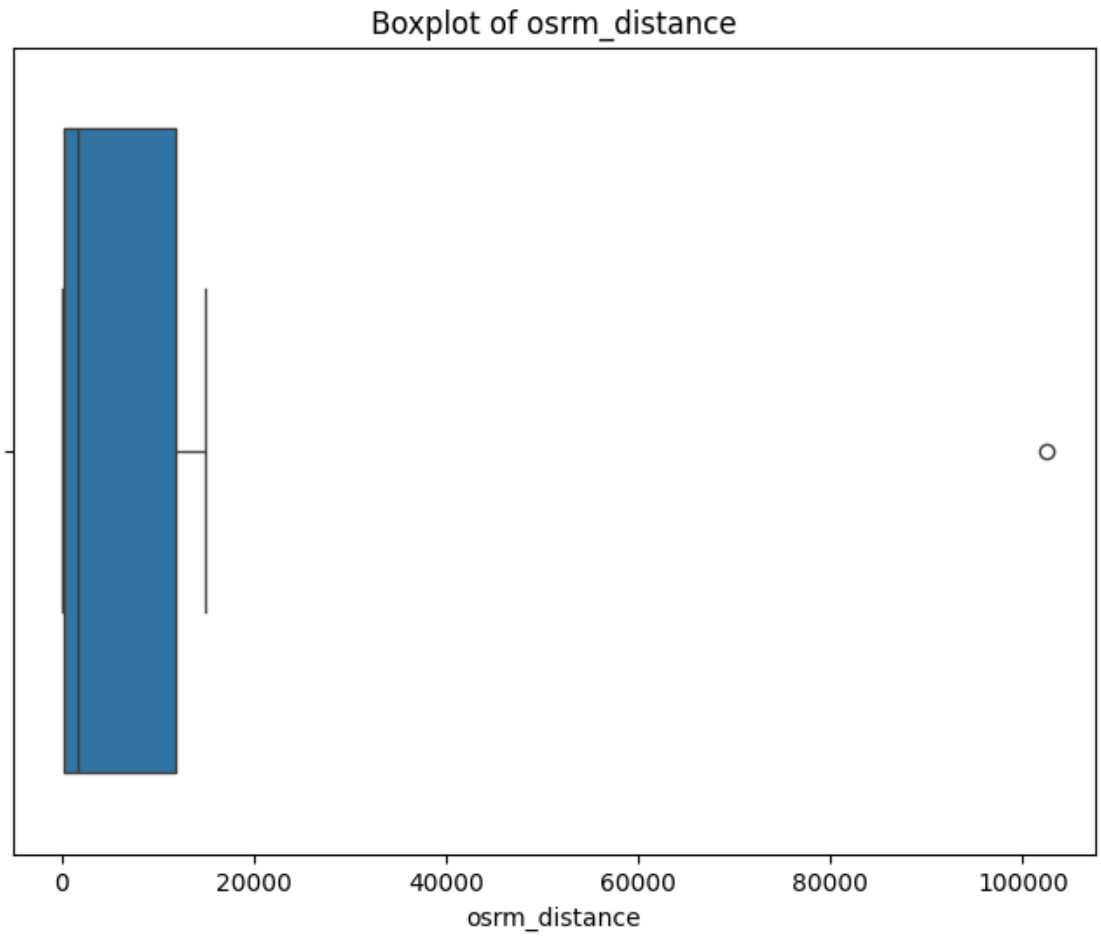

Boxplot of start_scan_to_end_scan
0
50000
100000
150000
200000
250000
300000
350000
400000
start_scan_to_end_scan

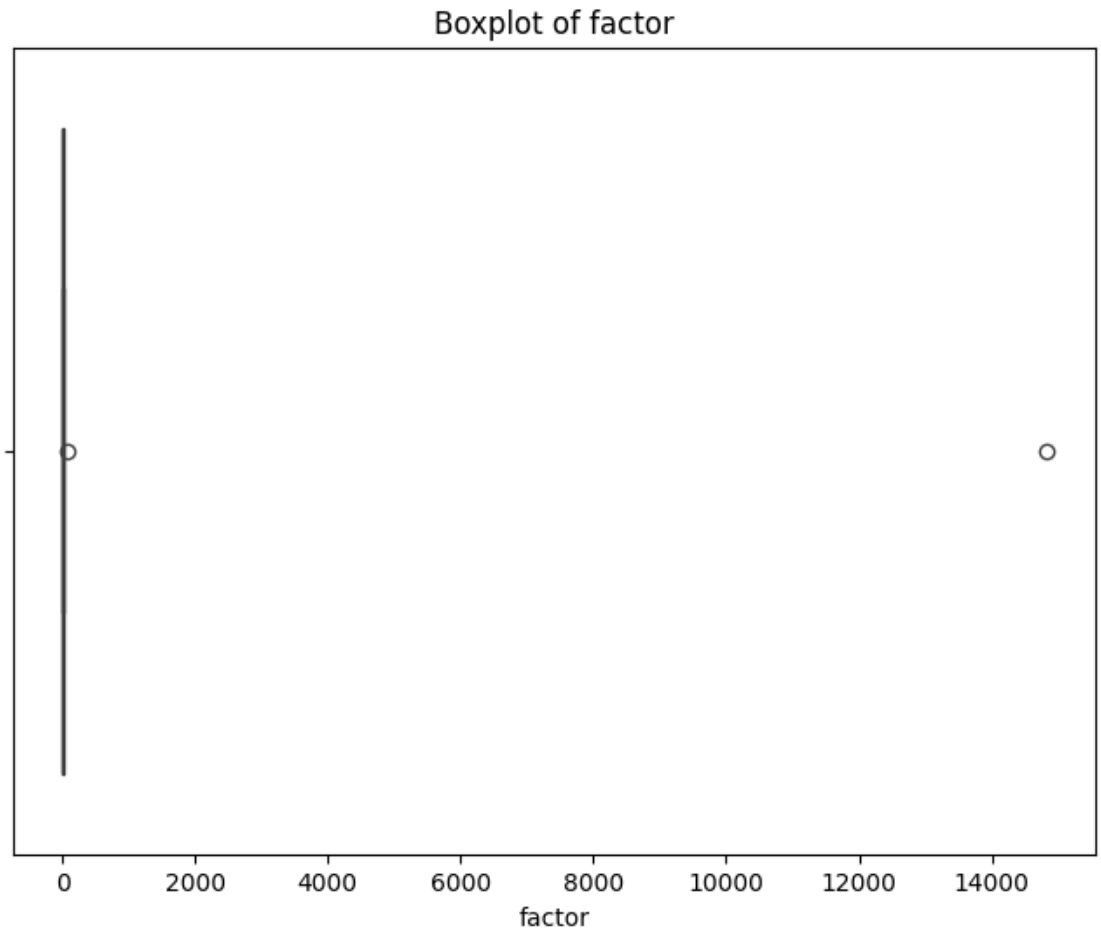

Boxplot of cutoff_factor
0
2000
4000
6000
8000
10000
12000
14000
cutoff_factor

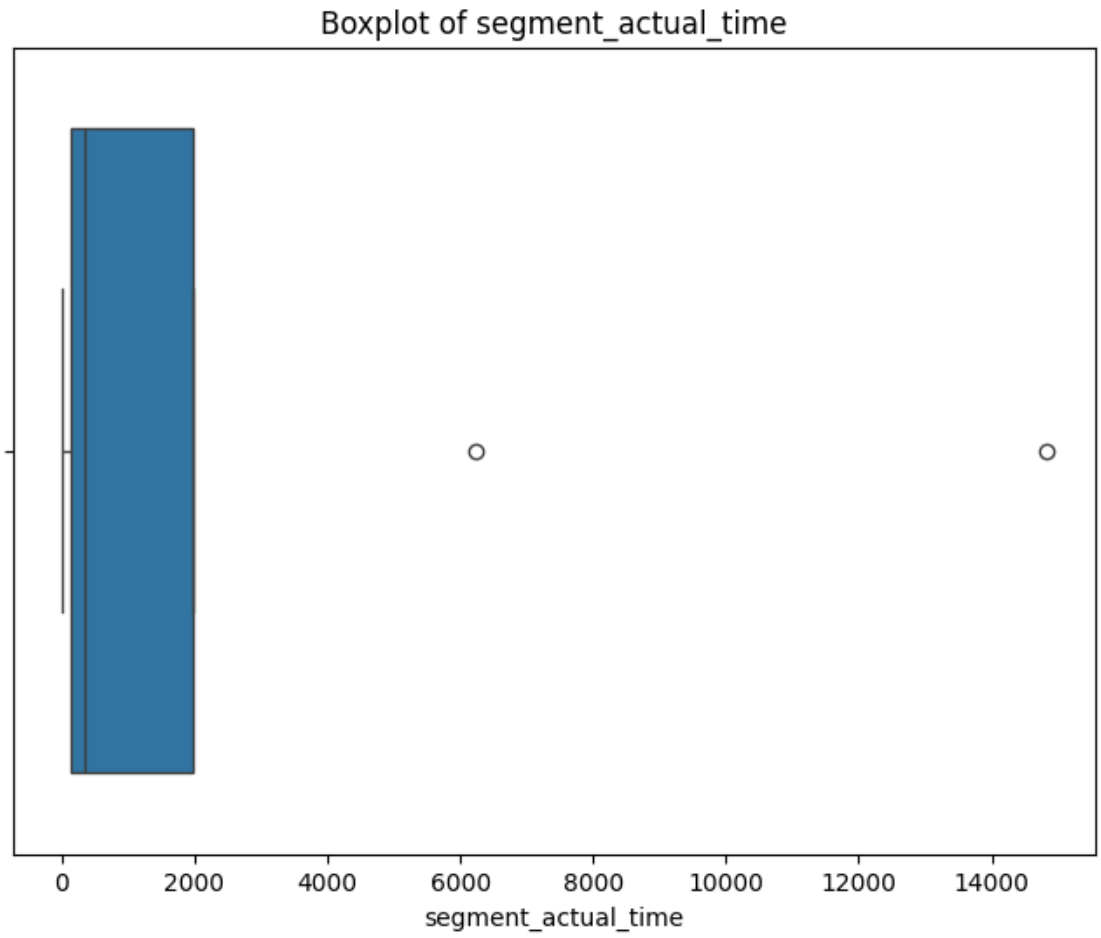

Boxplot of actual_distance_to_destination
0
20000
40000
60000
80000
actual_distance_to_destination

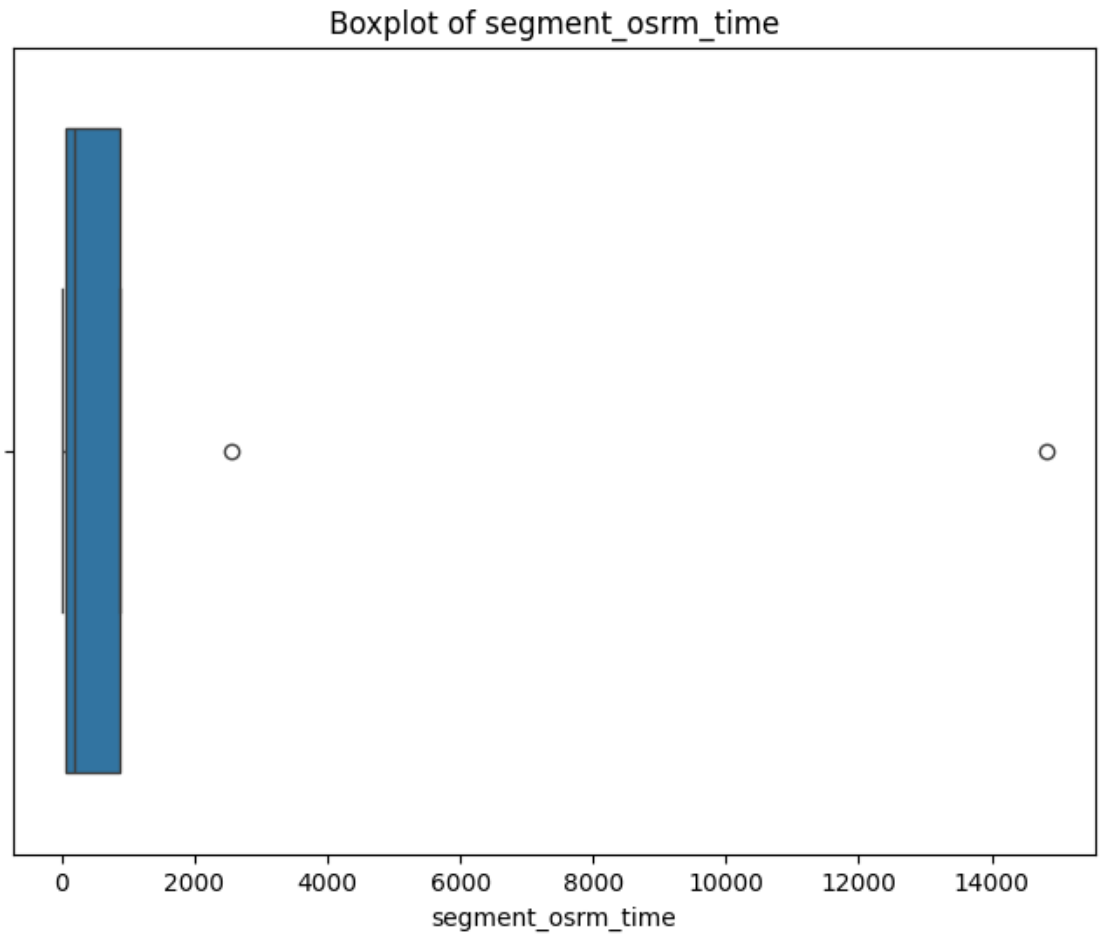

Boxplot of actual_time
0
25000
50000
75000
100000
125000
150000
175000
actual_time

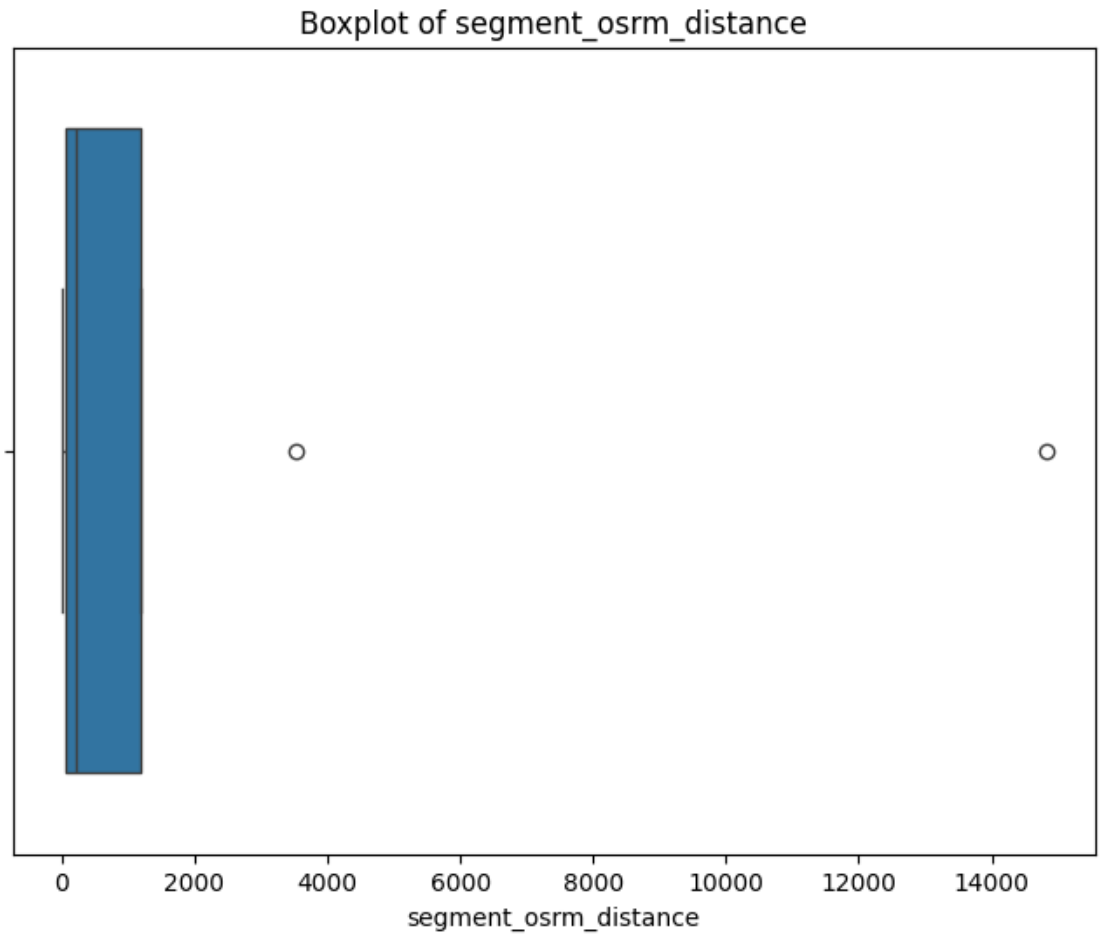

Boxplot of osrm_time
0
10000
20000
30000
40000
50000
60000
70000
80000
osrm_time

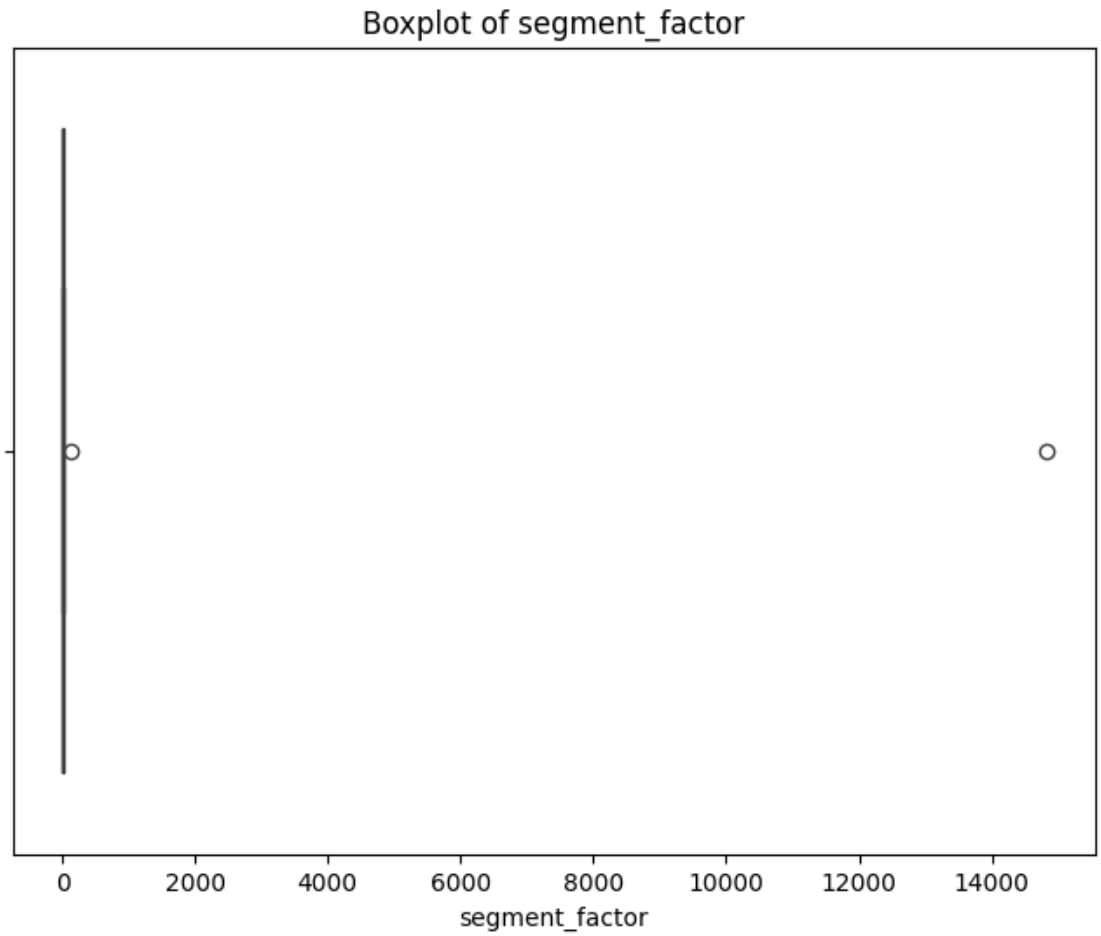

Boxplot of osrm_distance
0
20000
40000
60000
80000
100000
osrm_distance

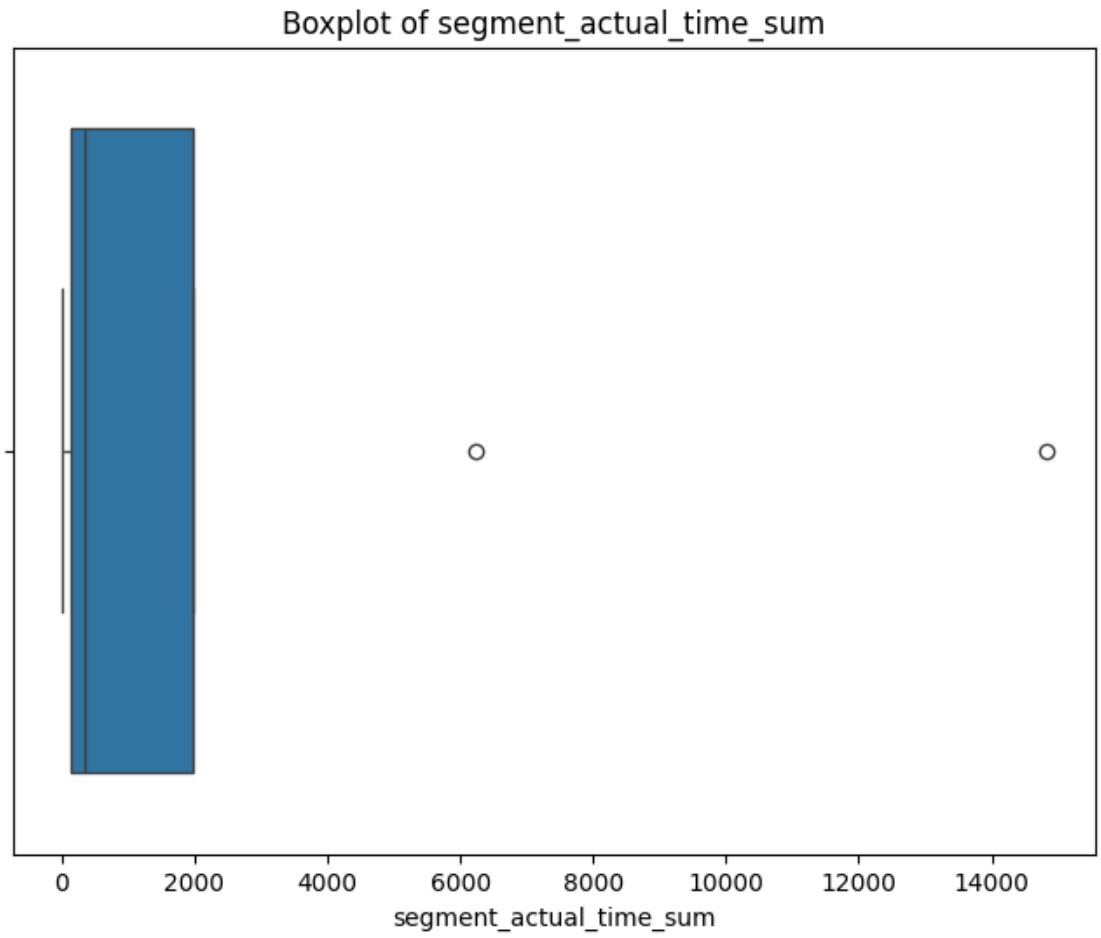

Boxplot of factor
0
2000
4000
6000
8000
10000
12000
14000
factor

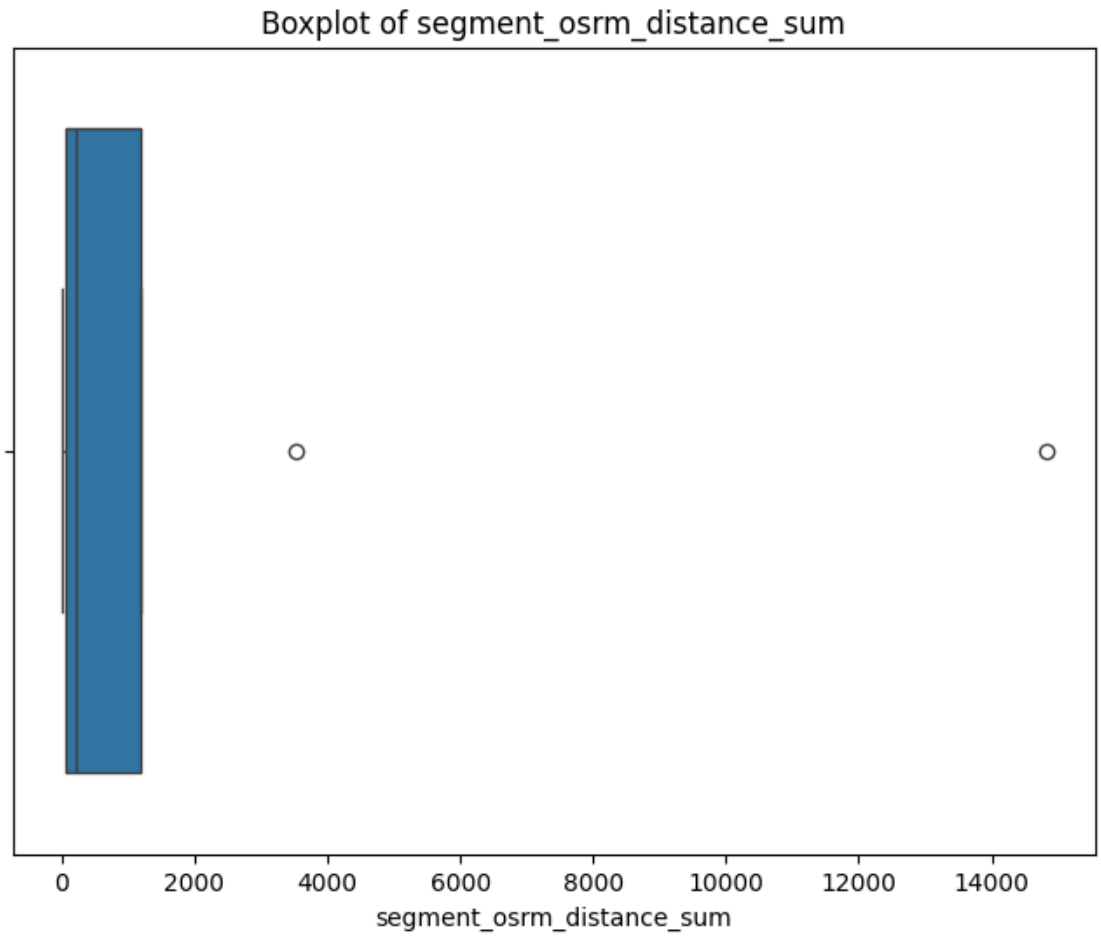

Boxplot of segment_actual_time
0
2000
4000
6000
8000
10000
12000
14000
segment_actual_time

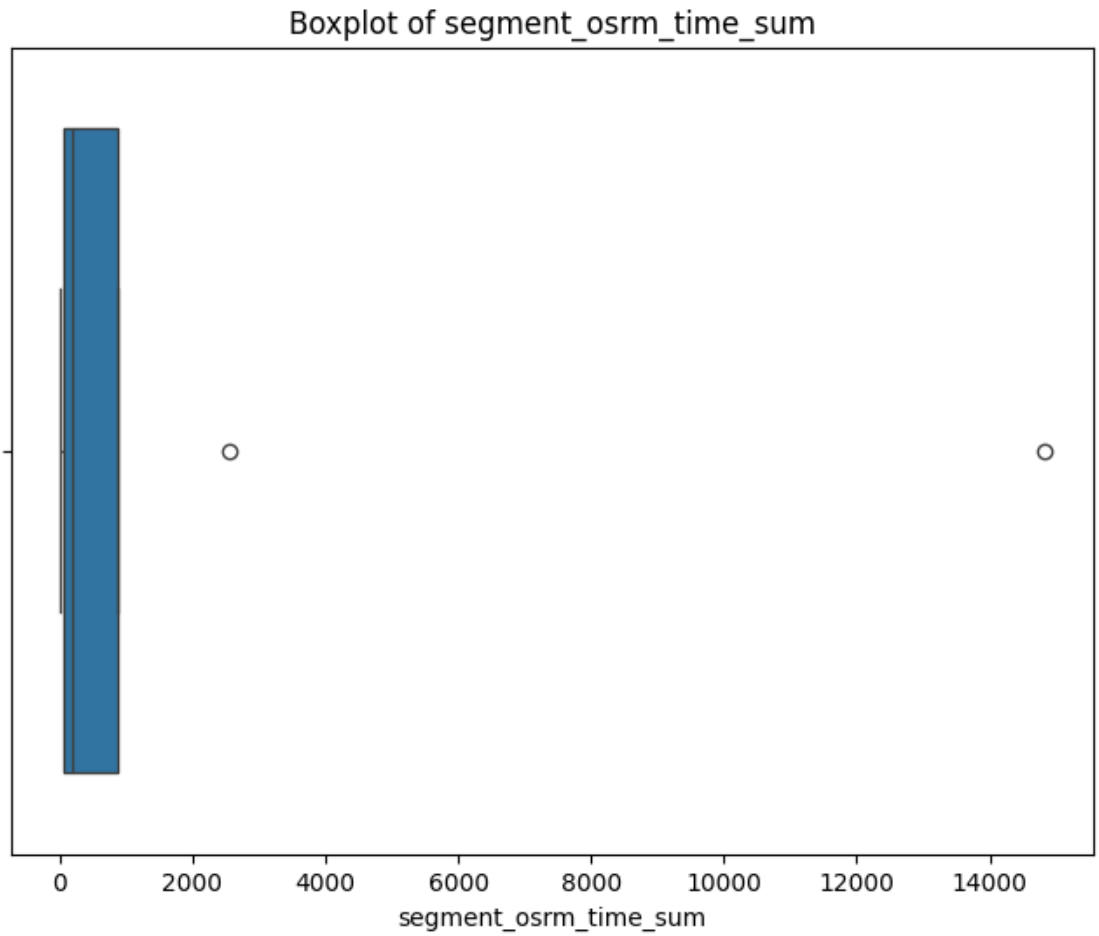

Boxplot of segment_osrm_time
0
2000
4000
6000
8000
10000
12000
14000
segment_osrm_time

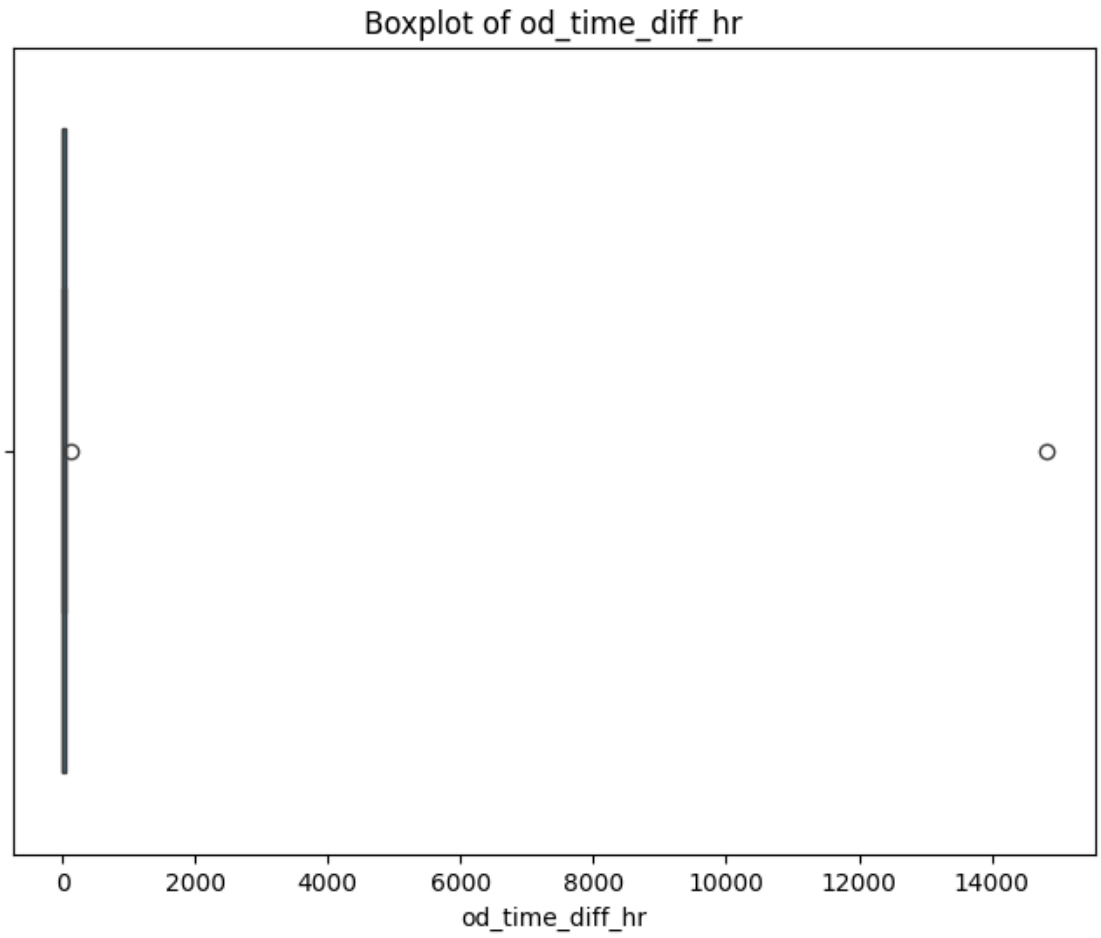

Boxplot of segment_osrm_distance
0
2000
4000
6000
8000
10000
12000
14000
segment_osrm_distance

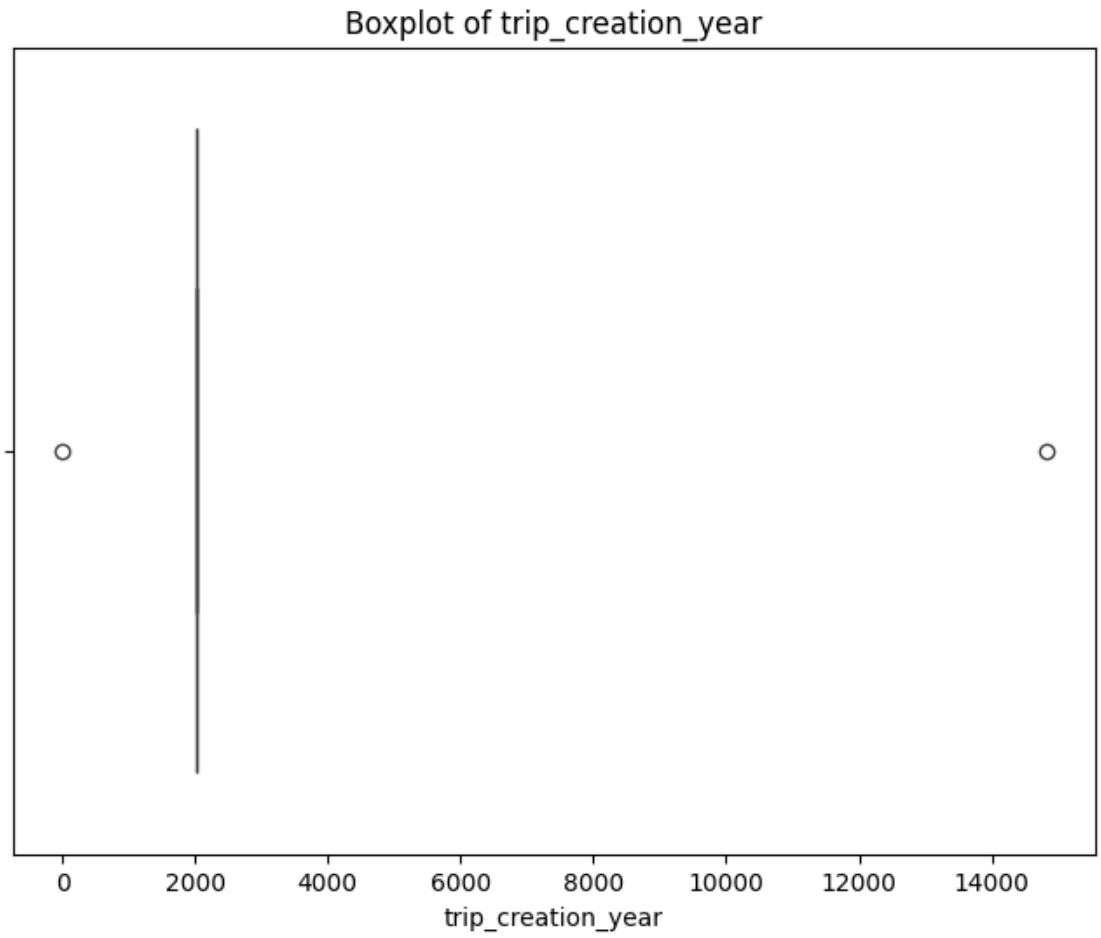

Boxplot of segment_factor
0
2000
4000
6000
8000
10000
12000
14000
segment_factor

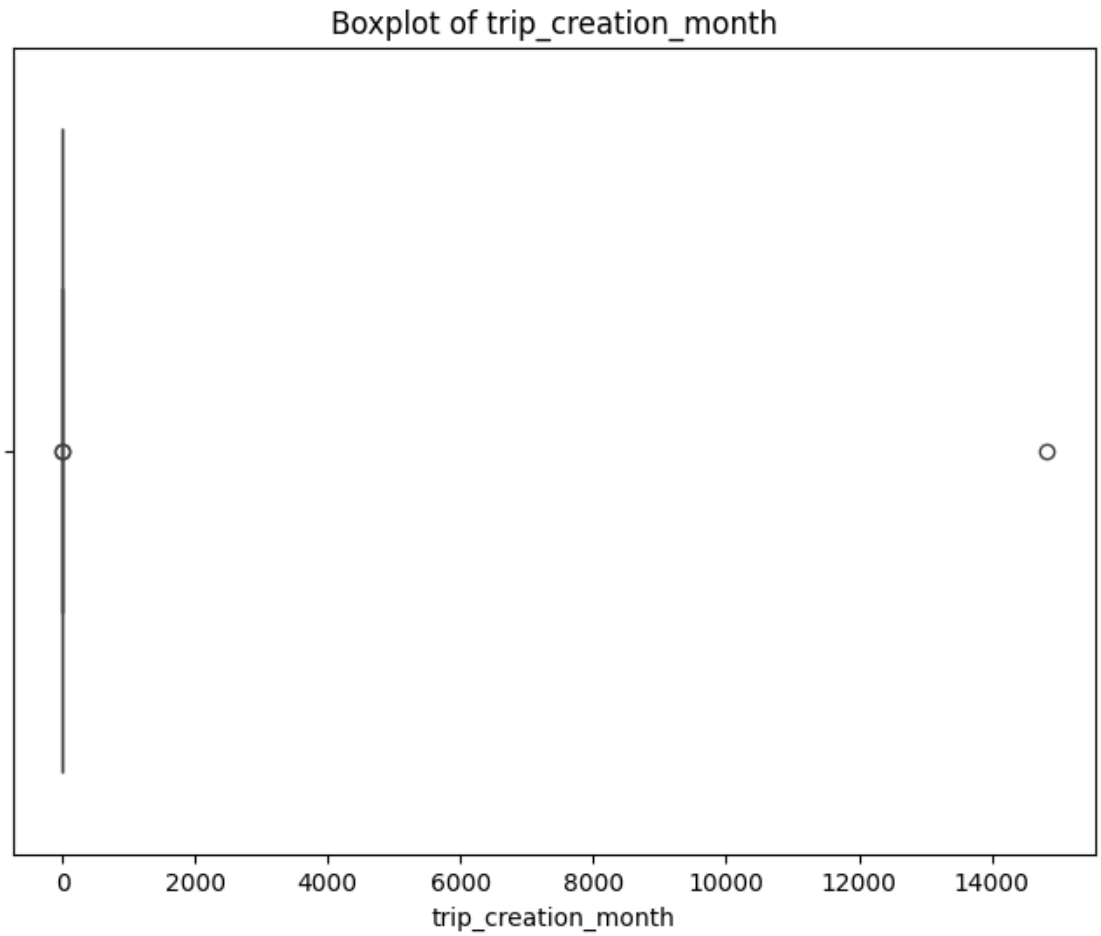

Boxplot of segment_actual_time_sum
0
2000
4000
6000
8000
10000
12000
14000
segment_actual_time_sum

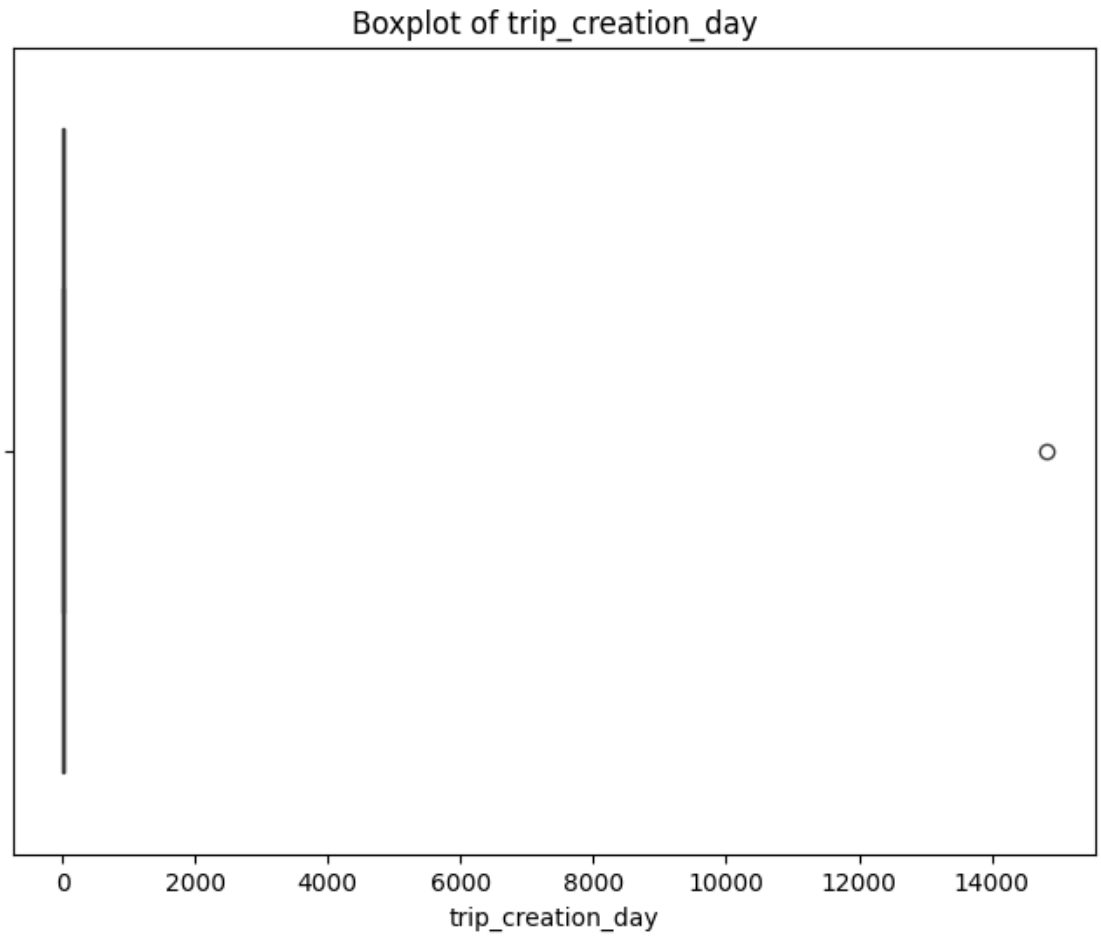

Boxplot of segment_osrm_distance_sum
0
2000
4000
6000
8000
10000
12000
14000
segment_osrm_distance_sum

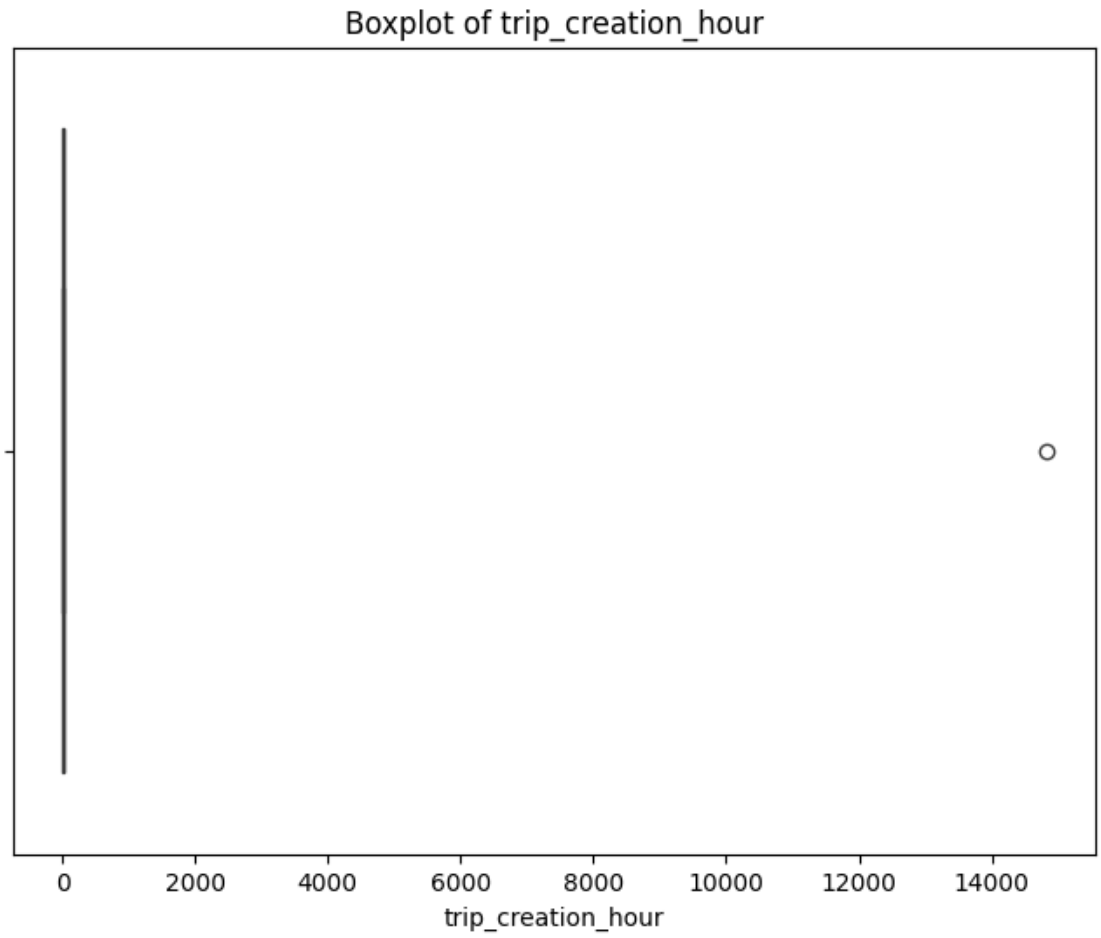

Boxplot of segment_osrm_time_sum
0
2000
4000
6000
8000
10000
12000
14000
segment_osrm_time_sum

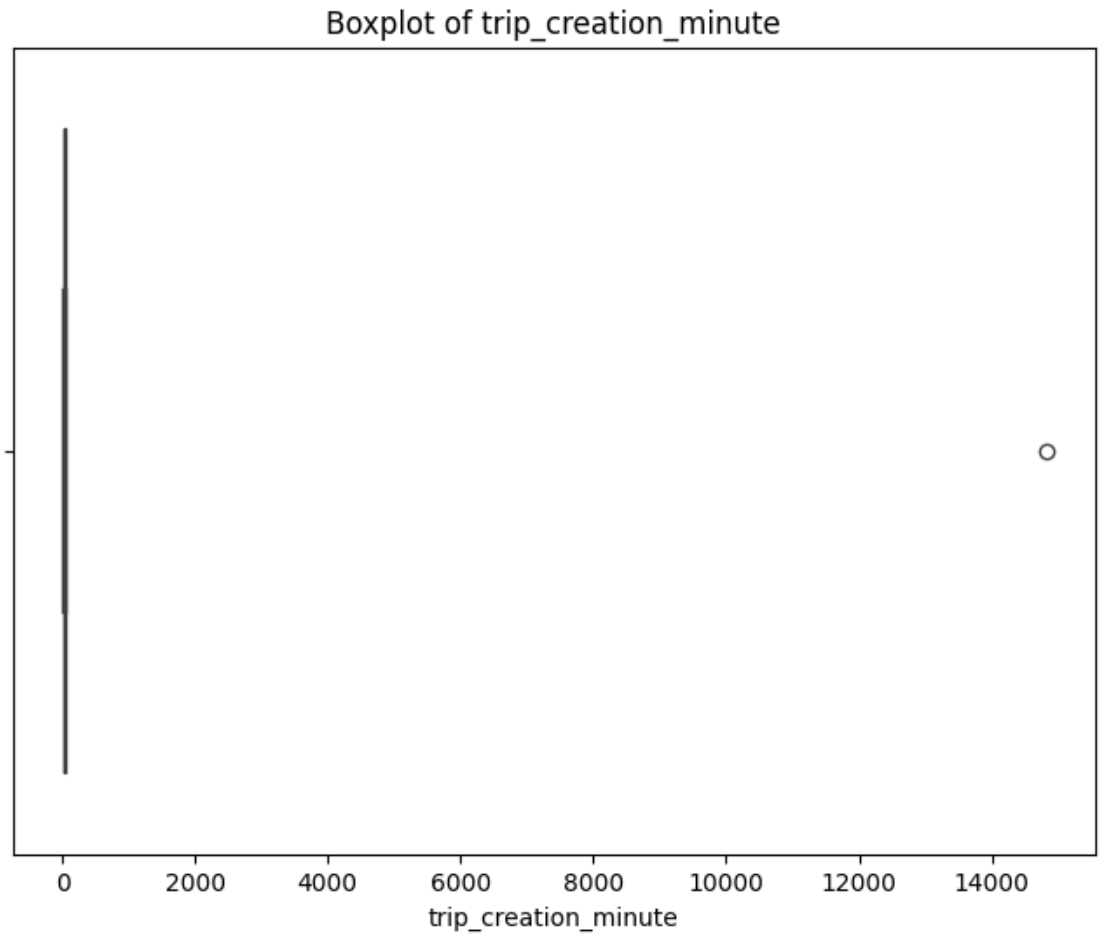

Boxplot of od_time_diff_hr
0
2000
4000
6000
8000
10000
12000
14000
od_time_diff_hr

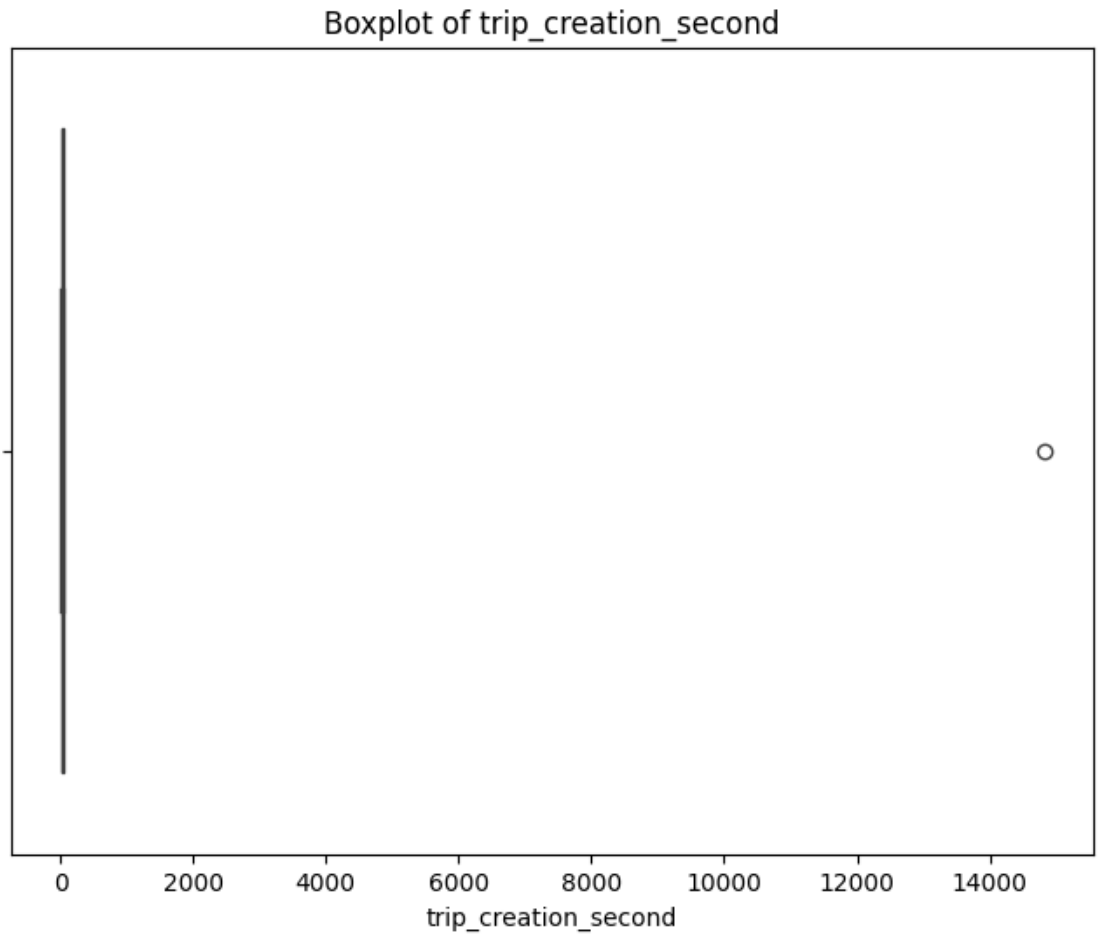

Boxplot of trip_creation_year
0
2000
4000
6000
8000
10000
12000
14000
trip_creation_year


Boxplot of trip_creation_month
0
2000
4000
6000
8000
10000
12000
14000
trip_creation_month


Boxplot of trip_creation_day
0
2000
4000
6000
8000
10000
12000
14000
trip_creation_day


Boxplot of trip_creation_hour
0
2000
4000
6000
8000
10000
12000
14000
trip_creation_hour


Boxplot of trip_creation_minute
0
2000
4000
6000
8000
10000
12000
14000
trip_creation_minute



[156]:
```
Lower_bound = Q1-1.5*IQR
Upper_bound = Q3+1.5*IQR

bounds_df = pd.DataFrame({"LowerBound" :Lower_bound,"UpperBound" :Upper_bound})
print(bounds_df)
```

```
                               LowerBound   UpperBound
start_scan_to_end_scan      -3219.000000  6453.000000
cutoff_factor                 -43.500000    96.500000
actual_distance_to_destination -571.940666  1085.495420
actual_time                  -1239.500000  2444.500000
osrm_time                     -619.000000  1197.000000
osrm_distance                 -747.169600  1420.585600
factor                           0.240196     3.985294
segment_actual_time           -385.500000   818.500000
segment_osrm_time             -200.000000   416.000000
segment_osrm_distance         -246.567350   498.024250
segment_factor                   0.031480     4.200008
```

```
segment_actual_time_sum         -385.500000   818.500000
segment_osrm_distance_sum       -246.567350   498.024250
segment_osrm_time_sum           -200.000000   416.000000
od_time_diff_hr                   -9.707869    22.843363
trip_creation_year              2018.000000  2018.000000
trip_creation_month                9.000000     9.000000
trip_creation_day                 -2.500000    41.500000
trip_creation_hour               -20.000000    44.000000
trip_creation_minute             -30.000000    90.000000
trip_creation_second             -30.000000    90.000000
```

17. Total number of outliers present in every column

[157]:
```
outliers_lower= (trip_df_summary<Lower_bound).sum()
outliers_upper= (trip_df_summary>Upper_bound).sum()
total_outliers= outliers_lower + outliers_upper

ouliers_count_df = pd.DataFrame({"LowerBound_outliers" :outliers_lower,␣
 ↪"UpperBound_outliers":outliers_upper, "Total":total_outliers})

print(ouliers_count_df)
```

```
                              LowerBound_outliers  UpperBound_outliers  \
start_scan_to_end_scan                          0                    4
cutoff_factor                                   0                    2
actual_distance_to_destination                  0                    4
actual_time                                     0                    4
osrm_time                                       0                    4
osrm_distance                                   0                    4
factor                                          0                    2
segment_actual_time                             0                    2
segment_osrm_time                               0                    2
segment_osrm_distance                           0                    2
segment_factor                                  1                    2
segment_actual_time_sum                         0                    2
segment_osrm_distance_sum                       0                    2
segment_osrm_time_sum                           0                    2
od_time_diff_hr                                 0                    2
trip_creation_year                              1                    1
trip_creation_month                             1                    3
trip_creation_day                               0                    1
trip_creation_hour                              0                    1
trip_creation_minute                            0                    1
trip_creation_second                            0                    1

                              Total
start_scan_to_end_scan            4
cutoff_factor                     2
```

```
actual_distance_to_destination      4
actual_time                         4
osrm_time                           4
osrm_distance                       4
factor                              2
segment_actual_time                 2
segment_osrm_time                   2
segment_osrm_distance               2
segment_factor                      3
segment_actual_time_sum             2
segment_osrm_distance_sum           2
segment_osrm_time_sum               2
od_time_diff_hr                     2
trip_creation_year                  2
trip_creation_month                 4
trip_creation_day                   1
trip_creation_hour                  1
trip_creation_minute                1
trip_creation_second                1
```

18. Removing outliers data to make the datset more manageable without any bias.

[158]:
```
for column in trip_df.select_dtypes(include='number').columns:
    trip_df = trip_df[(trip_df[column] >= Lower_bound[column]) &␣
 ↪(trip_df[column] <= Upper_bound[column])]
```

19. Rechecking the max values of each columns after removing the existing outliers with 'describe' function.

[159]:
```
trip_df.describe()
```

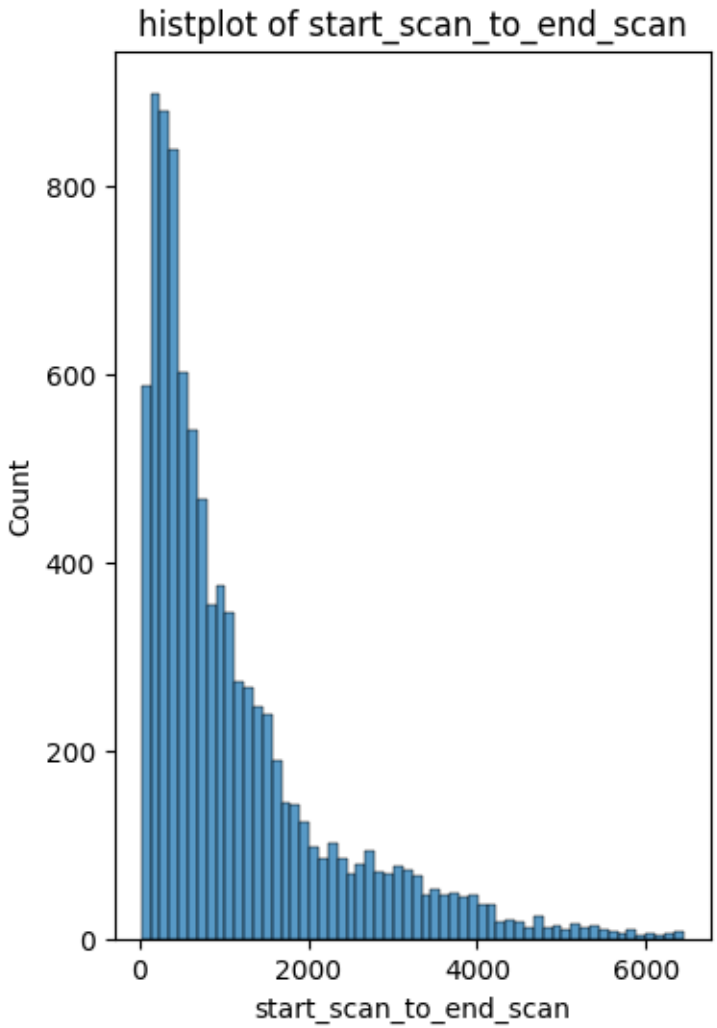
[159]:
```
       start_scan_to_end_scan  cutoff_factor  actual_distance_to_destination  \
count             9050.000000    9050.000000                     9050.000000
mean              1139.941326      22.993591                      189.607805
std               1130.092076      21.683030                      210.227068
min                 26.000000       9.000000                        9.002461
25%                348.000000       9.000000                       48.670468
50%                730.000000       9.000000                       96.842566
75%               1510.750000      27.000000                      258.933178
max               6436.000000      96.000000                     1082.991668

       actual_time    osrm_time  osrm_distance       factor  \
count  9050.000000  9050.000000    9050.000000  9050.000000
mean    403.818564   207.270608     239.226923     2.034572
std     414.376040   209.391446     258.005519     0.626715
min       9.000000     7.000000       9.072900     0.610398
25%     115.000000    57.000000      63.185750     1.580995
50%     239.000000   121.000000     128.075000     1.921249
```

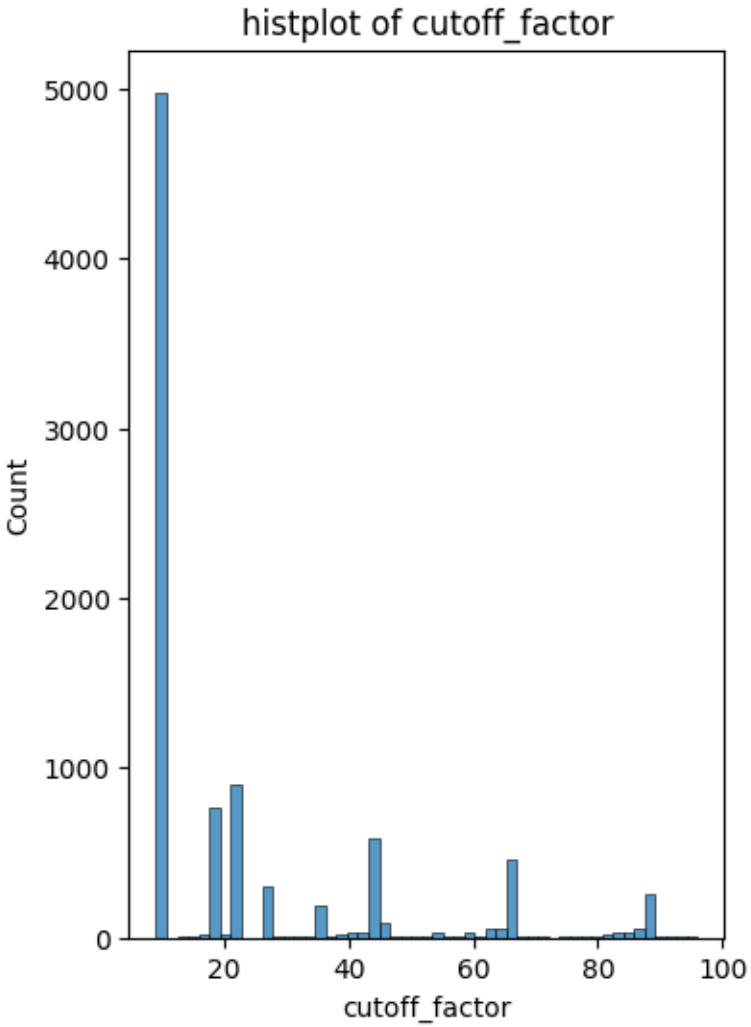
```
75%     552.750000   290.000000     323.595100     2.398786
max    2444.000000  1194.000000    1390.708100     3.980096

       segment_actual_time  segment_osrm_time  segment_osrm_distance  …  \
count          9050.000000        9050.000000            9050.000000  …
mean            146.027293          78.568066              87.823524  …
std             126.968333          67.886323              79.189409  …
min               9.000000           7.000000               9.072900  …
25%              55.000000          29.000000              29.940400  …
50%              96.000000          53.000000              53.472500  …
75%             203.000000         113.000000             129.513050  …
max             798.000000         403.000000             461.225700  …

       segment_actual_time_sum  segment_osrm_distance_sum  \
count              9050.000000                9050.000000
mean                146.027293                  87.823524
std                 126.968333                  79.189409
min                   9.000000                   9.072900
25%                  55.000000                  29.940400
50%                  96.000000                  53.472500
75%                 203.000000                 129.513050
max                 798.000000                 461.225700

       segment_osrm_time_sum  od_time_diff_hr  trip_creation_year  \
count            9050.000000      9050.000000              9050.0
mean               78.568066         4.733798              2018.0
std                67.886323         3.710843                 0.0
min                 7.000000         0.391024              2018.0
25%                29.000000         2.130834              2018.0
50%                53.000000         3.540827              2018.0
75%               113.000000         6.144104              2018.0
max               403.000000        22.789610              2018.0

       trip_creation_month  trip_creation_day  trip_creation_hour  \
count               9050.0        9050.000000         9050.000000
mean                   9.0          20.658122           12.563094
std                    0.0           5.385998            8.036870
min                    9.0          12.000000            0.000000
25%                    9.0          16.000000            4.000000
50%                    9.0          21.000000           15.000000
75%                    9.0          25.000000           20.000000
max                    9.0          30.000000           23.000000

       trip_creation_minute  trip_creation_second
count           9050.000000           9050.000000
mean              29.713702             29.692044
std               17.369621             17.300171
```

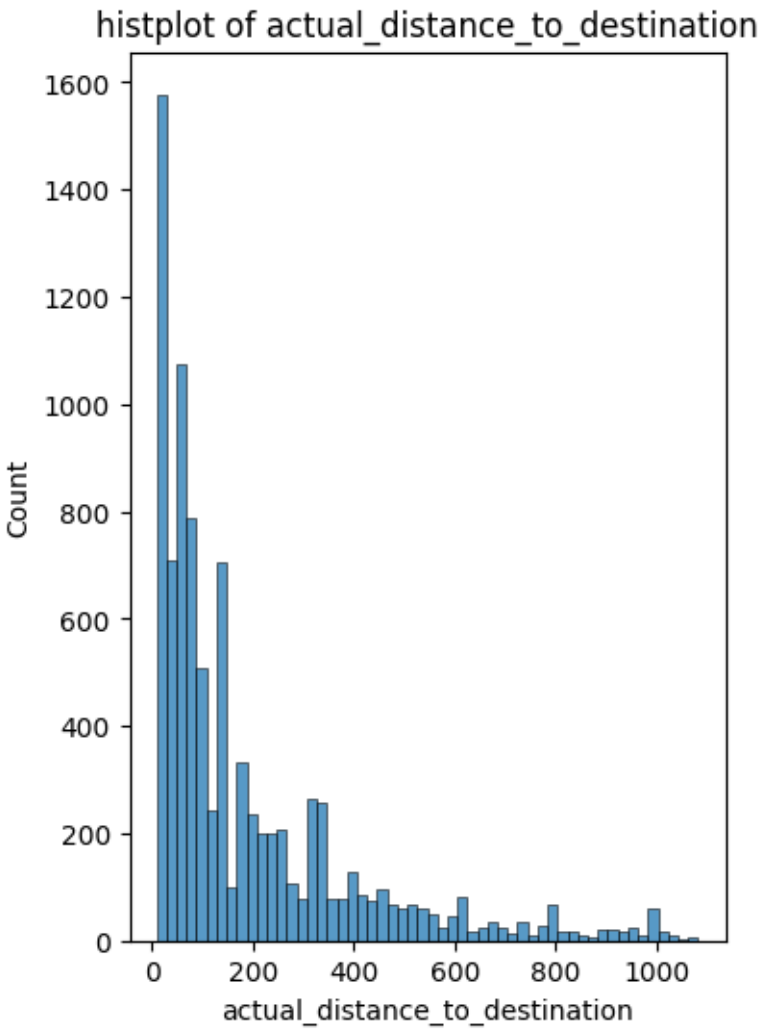
```
min              0.000000              0.000000
25%             15.000000             15.000000
50%             30.000000             30.000000
75%             45.000000             44.000000
max             59.000000             59.000000

[8 rows x 21 columns]
```

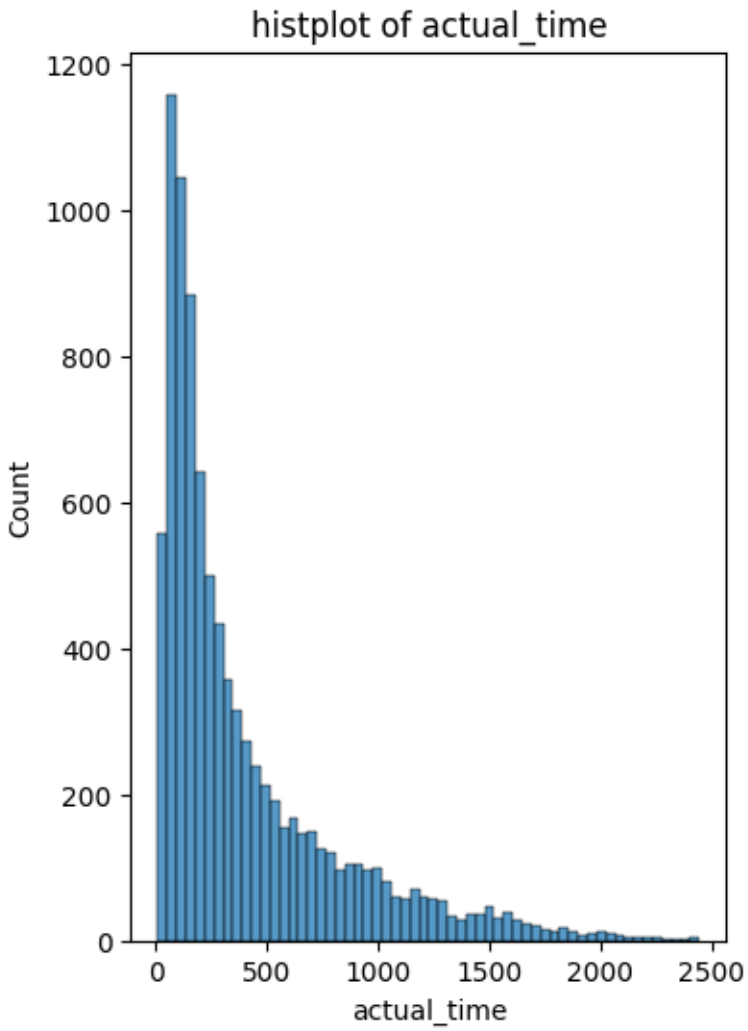
20. Histogram Visualization of every numerical column to identify the skewness and pattern of dstribution of datapoints.

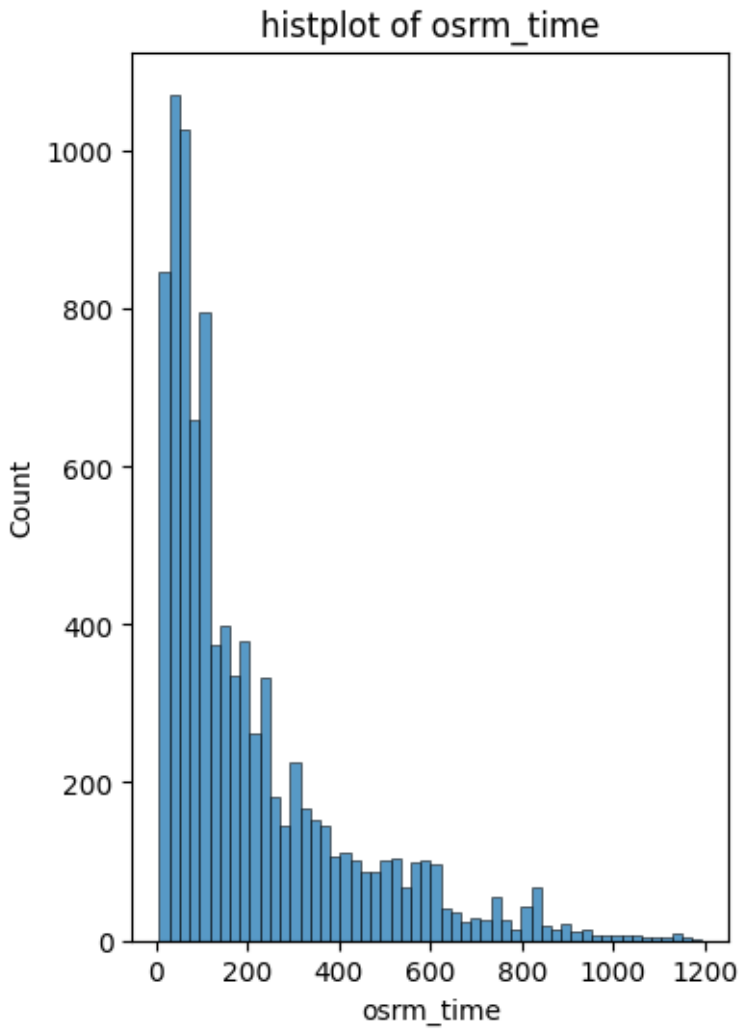
[112]:
```
for col in trip_df.select_dtypes(include = 'number'):
  plt.figure(figsize = (4, 6))
  sns.histplot(x = trip_df[col])
  plt.title(f"histplot of {col}")
  plt.xlabel(col)
  plt.show()
```

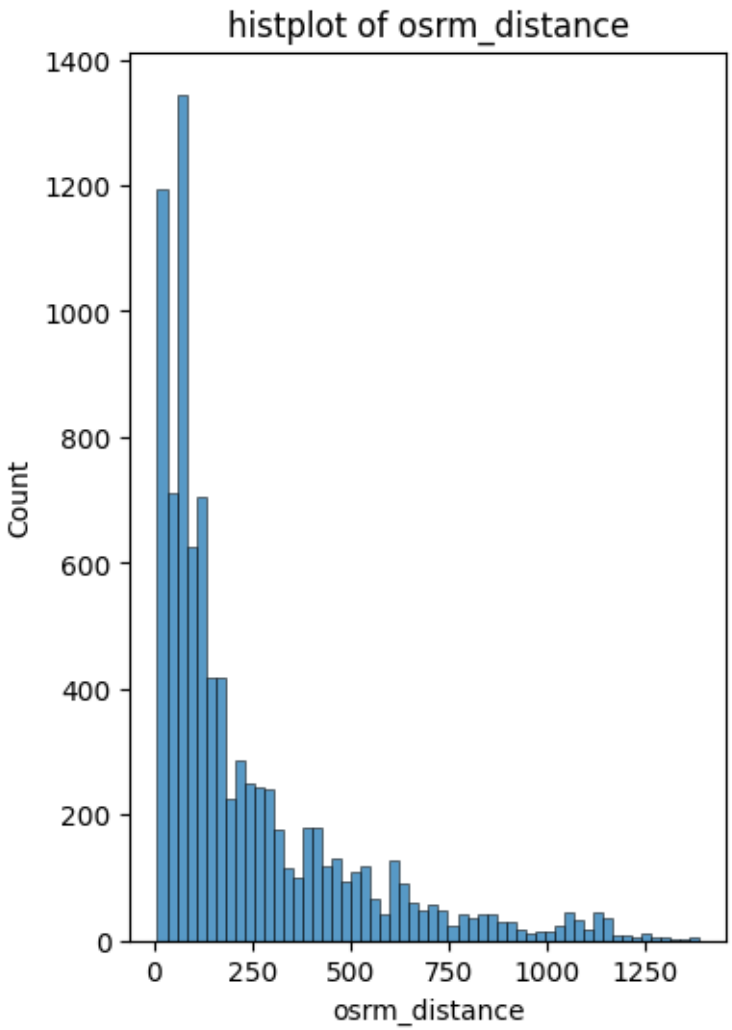


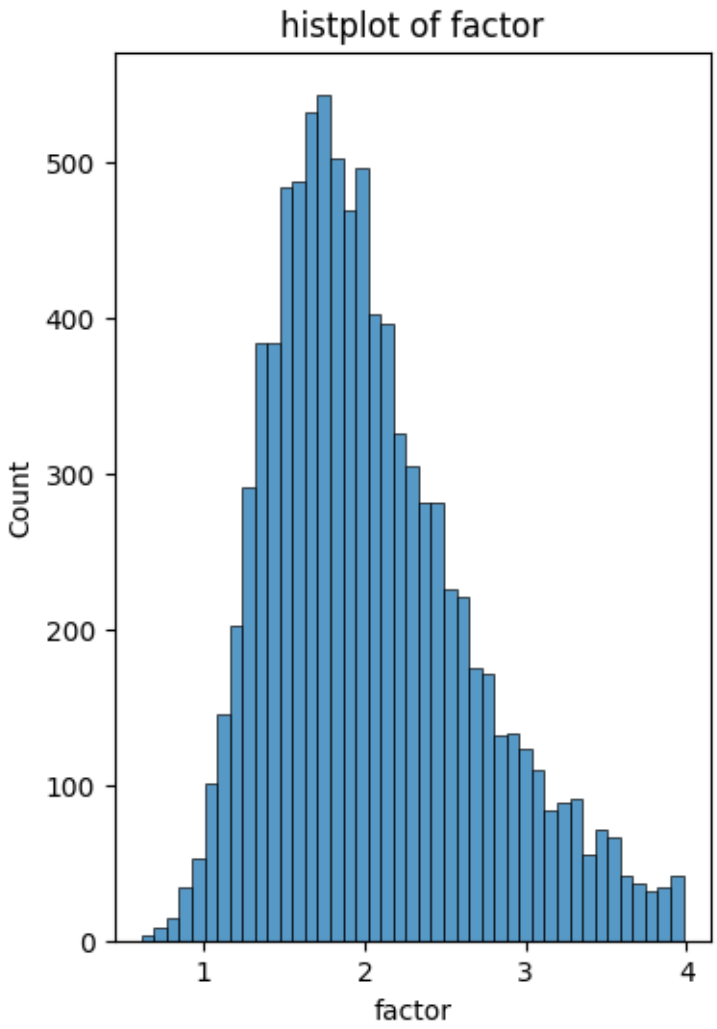

histplot of cutoff_factor
Count
cutoff_factor
0
1000
2000
3000
4000
5000
20
40
60
80
100

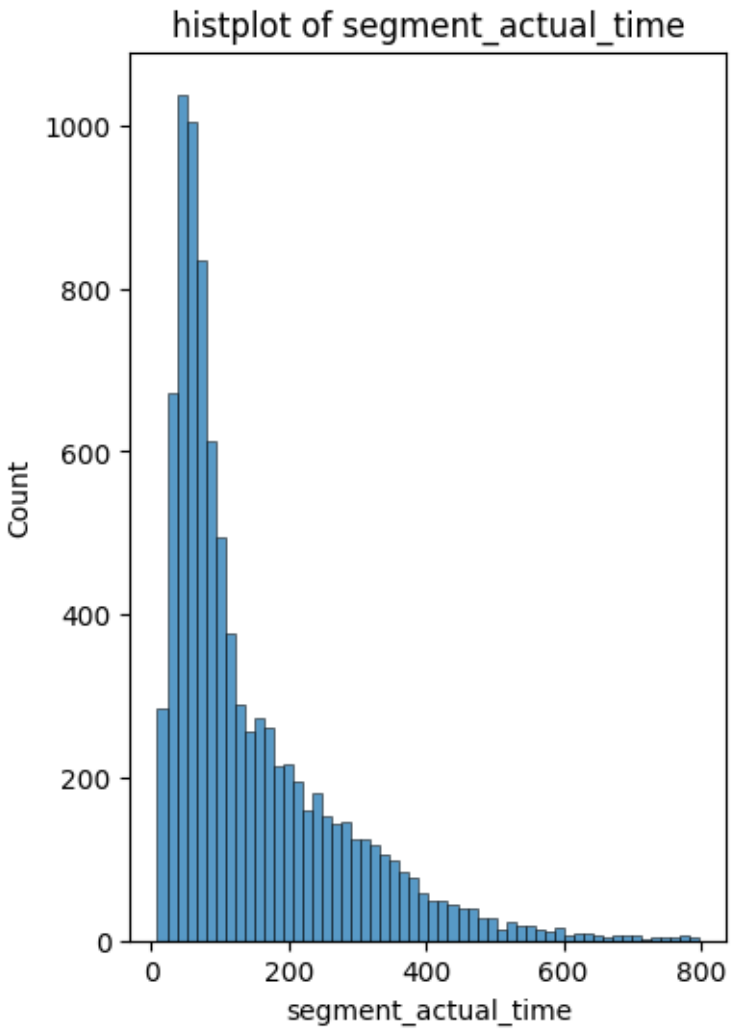

histplot of actual_distance_to_destination
Count
actual_distance_to_destination
0
200
400
600
800
1000
1200
1400
1600

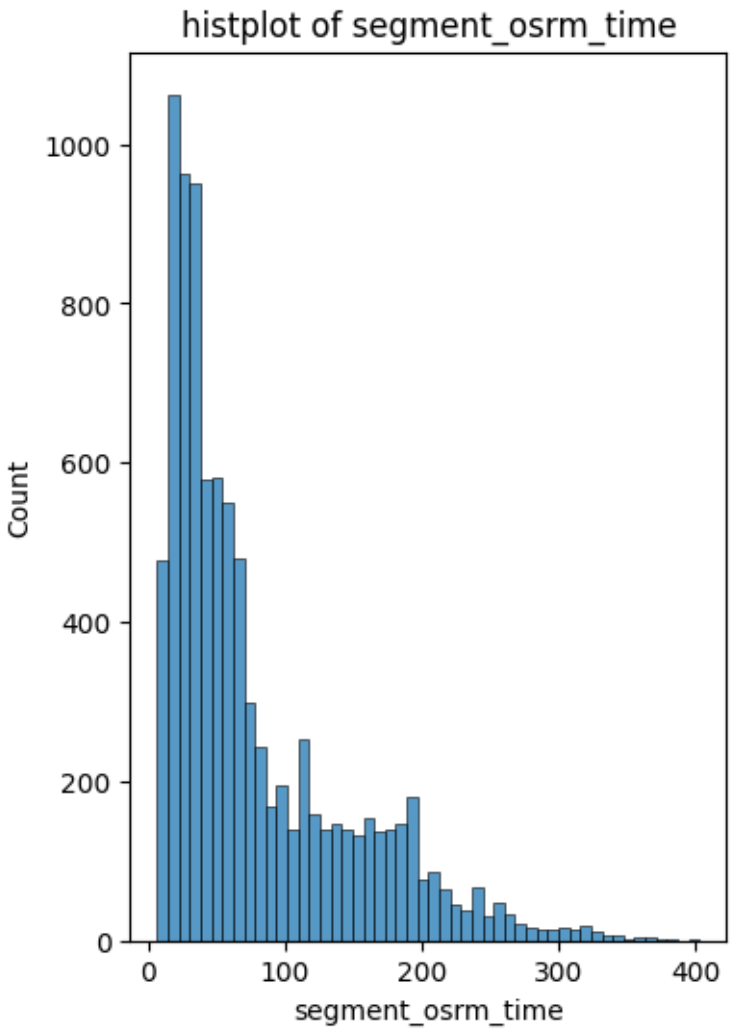

histplot of actual_time
1200
1000
800
600
400
200
0
Count
0
500
1000
1500
2000
2500
actual_time

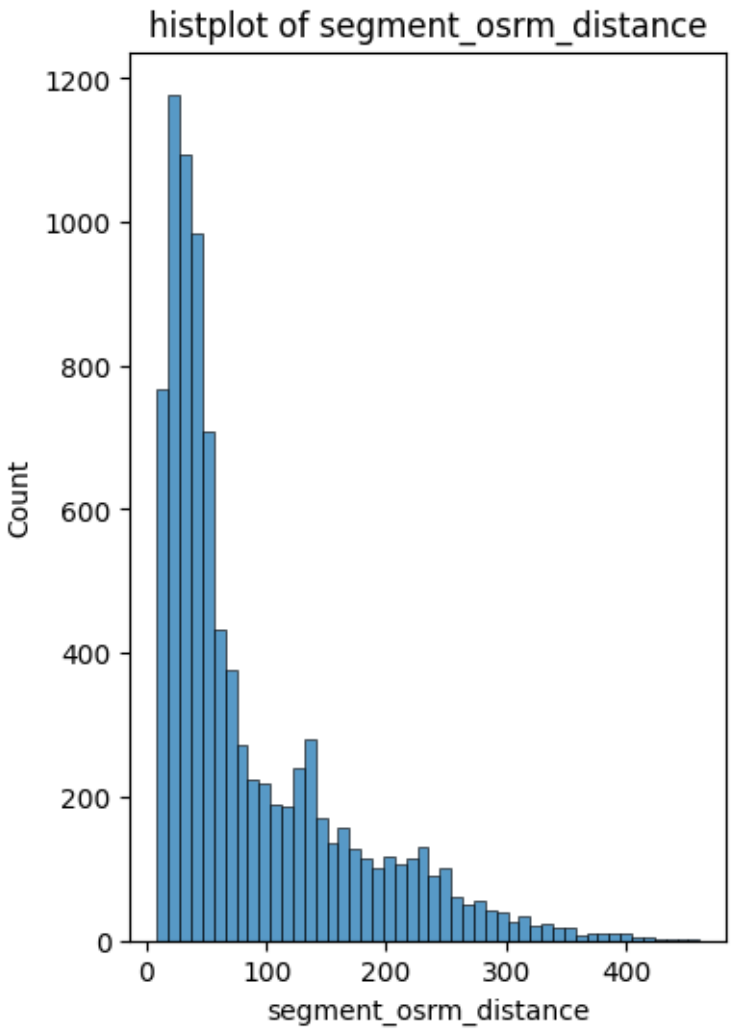

histplot of osrm_time
Count
osrm_time
0
200
400
600
800
1000
1200

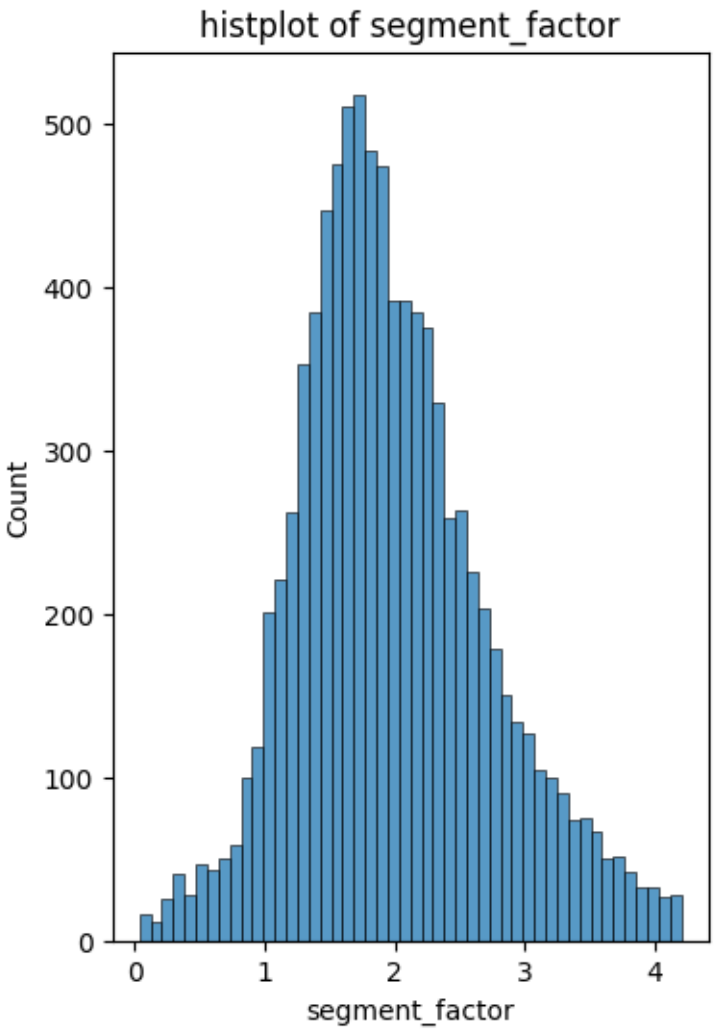

histplot of osrm_distance
Count
osrm_distance
0
200
400
600
800
1000
1200
1400
0
250
500
750
1000
1250

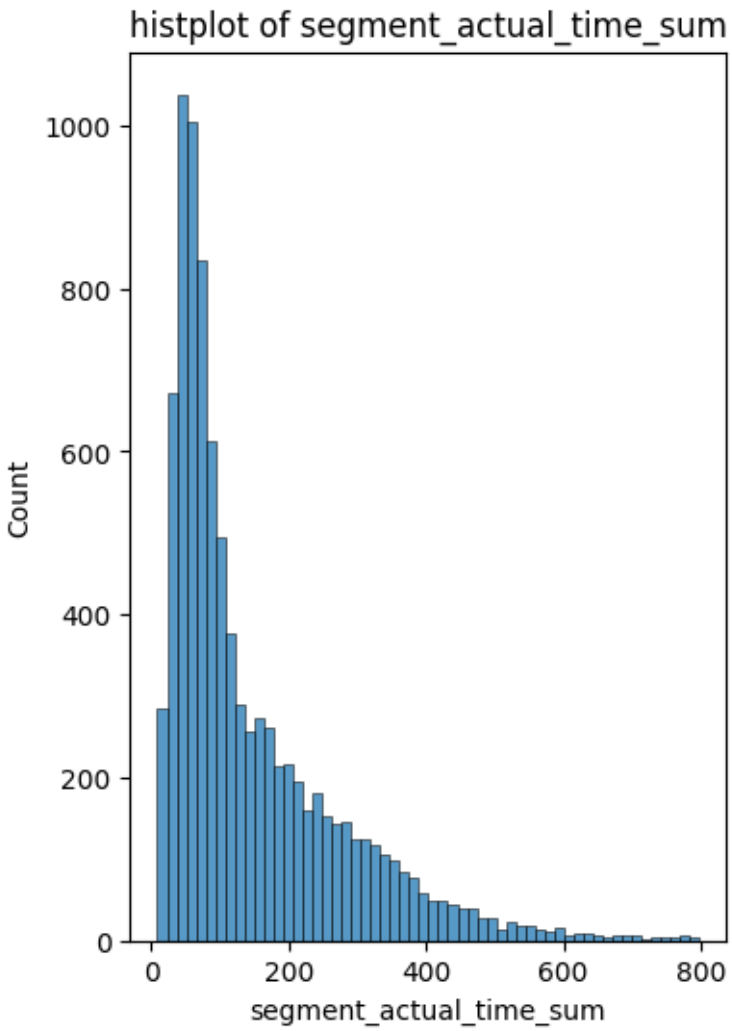

histplot of factor
Count
0
100
200
300
400
500
1
2
3
4
factor

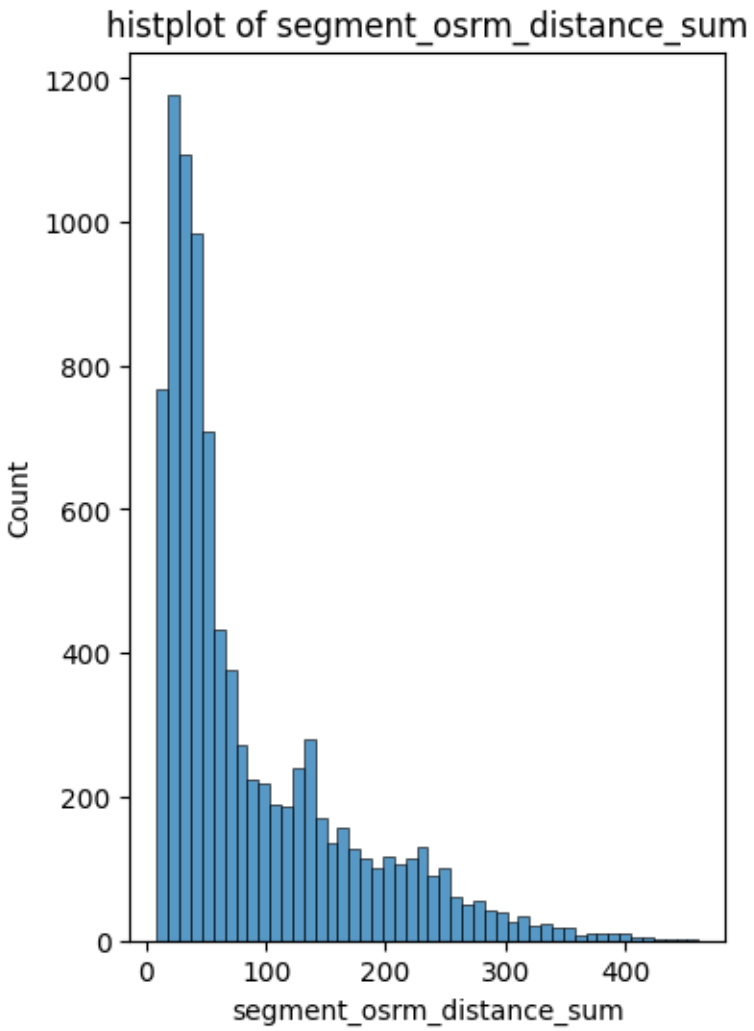

histplot of segment_actual_time
Count
1000
800
600
400
200
0
0
200
400
600
800
segment_actual_time

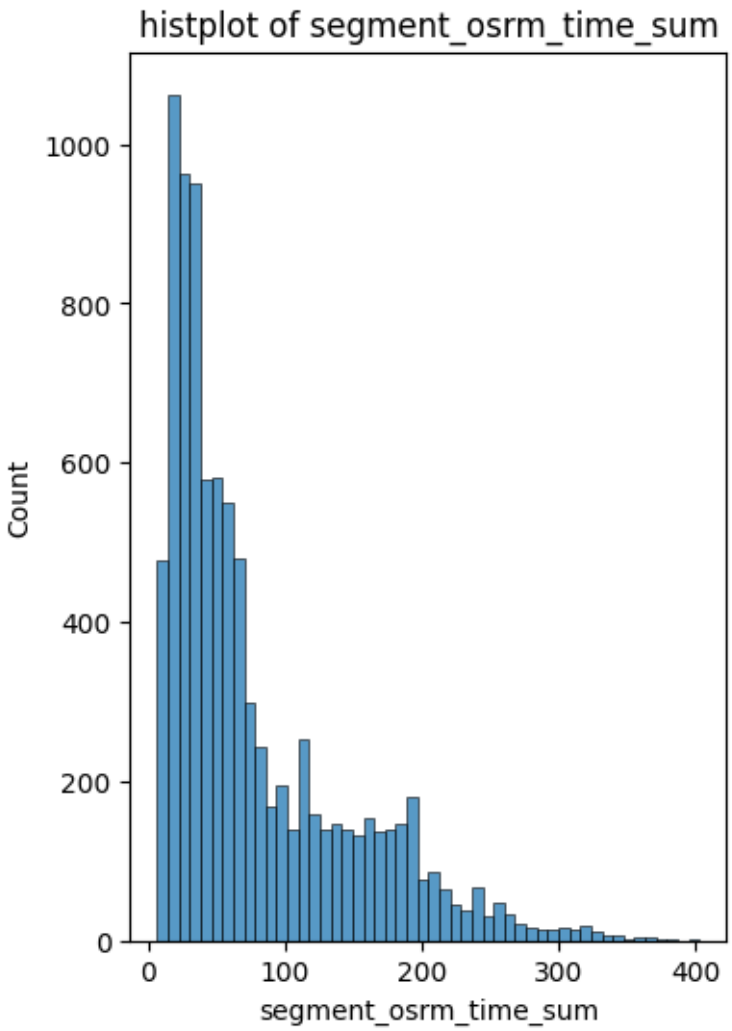

histplot of segment_osrm_time
Count
1000
800
600
400
200
0
0
100
200
300
400
segment_osrm_time

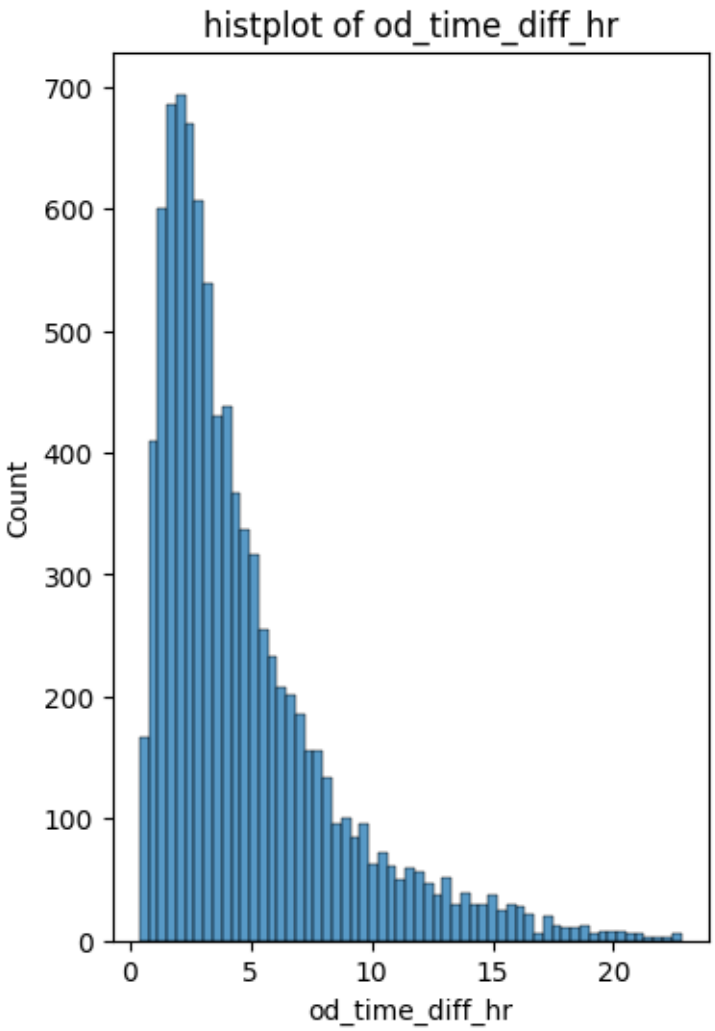

histplot of segment_osrm_distance
Count
segment_osrm_distance
0
200
400
600
800
1000
1200
0
100
200
300
400

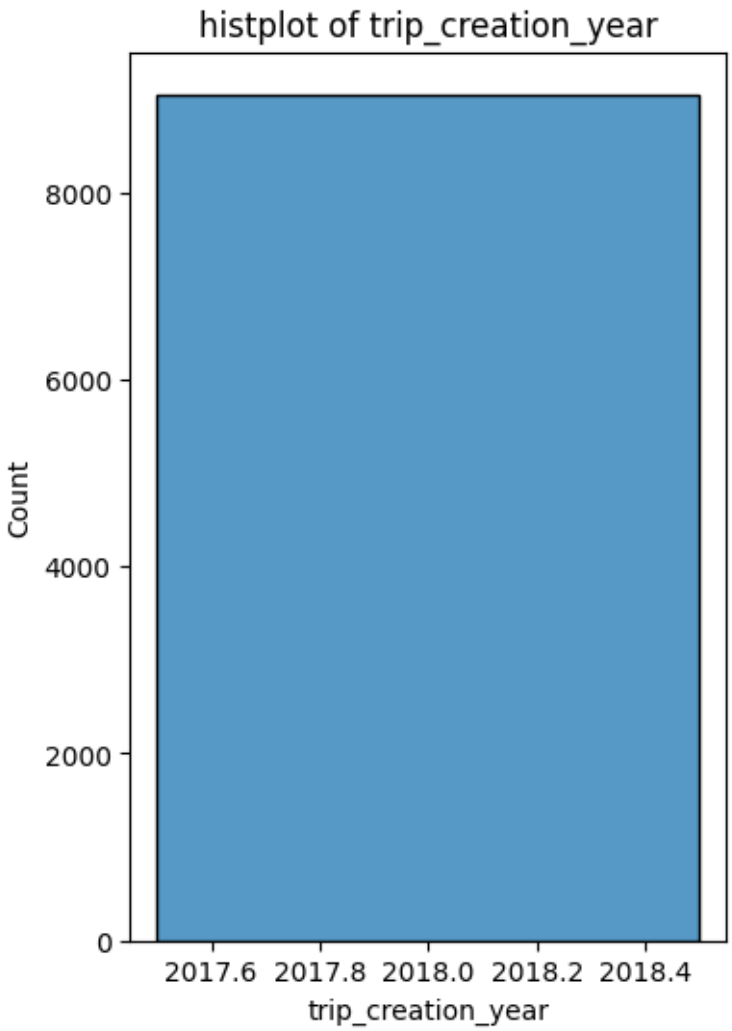

histplot of segment_factor
Count
segment_factor
0
100
200
300
400
500
0
1
2
3
4

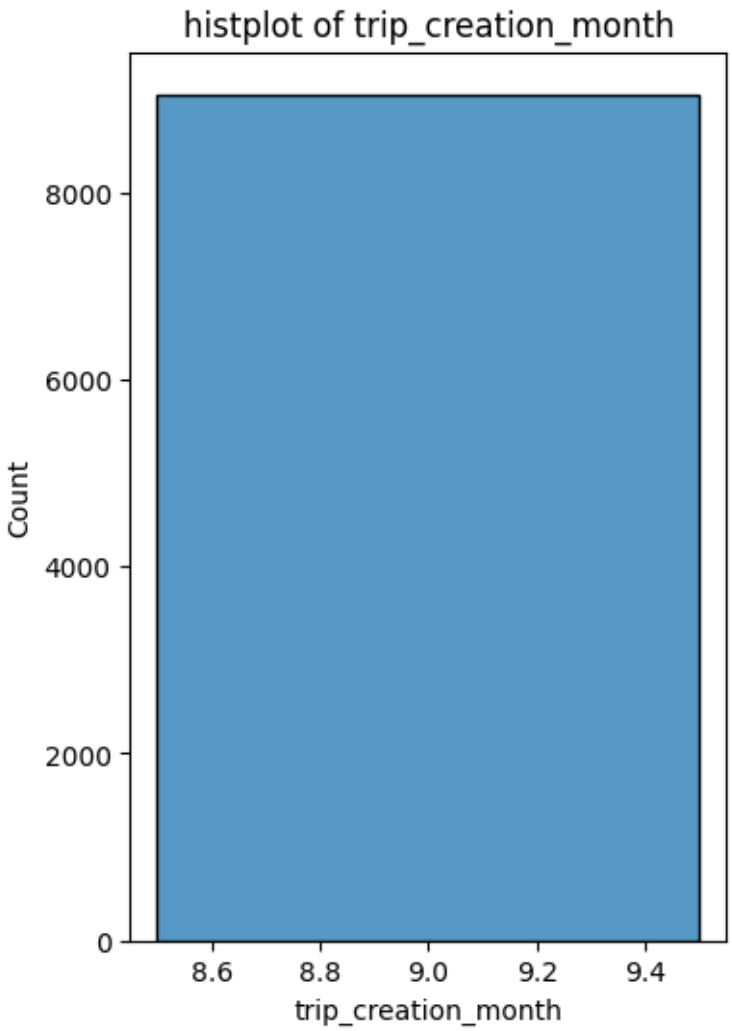

histplot of segment_actual_time_sum
Count
1000
800
600
400
200
0
0
200
400
600
800
segment_actual_time_sum

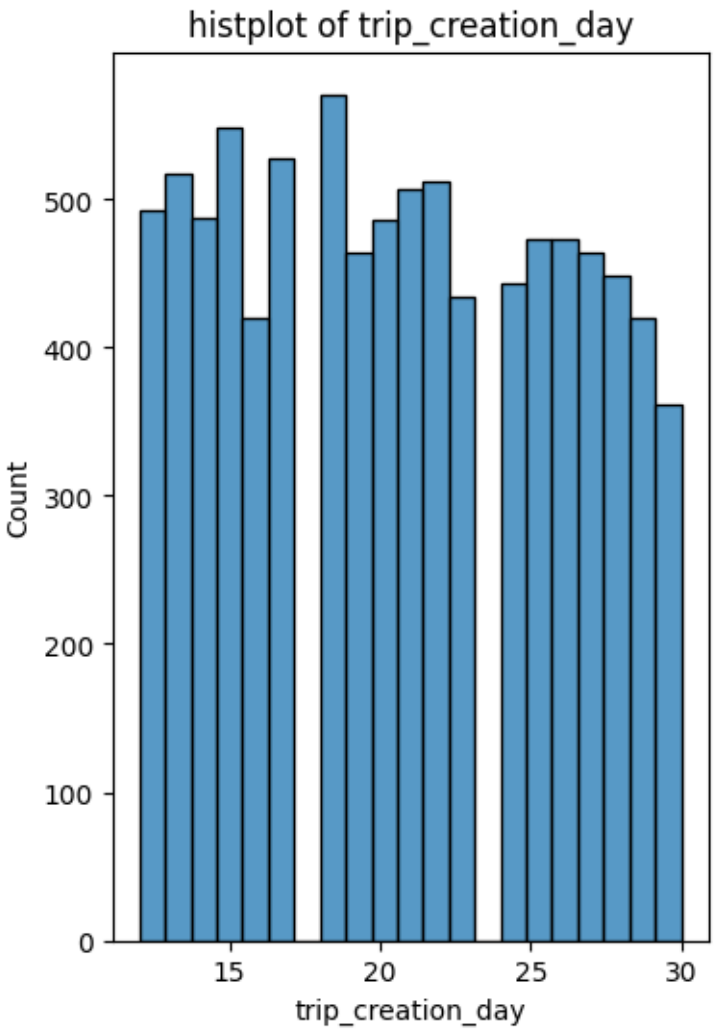

histplot of segment_osrm_distance_sum
Count
segment_osrm_distance_sum
0
200
400
600
800
1000
1200
0
100
200
300
400

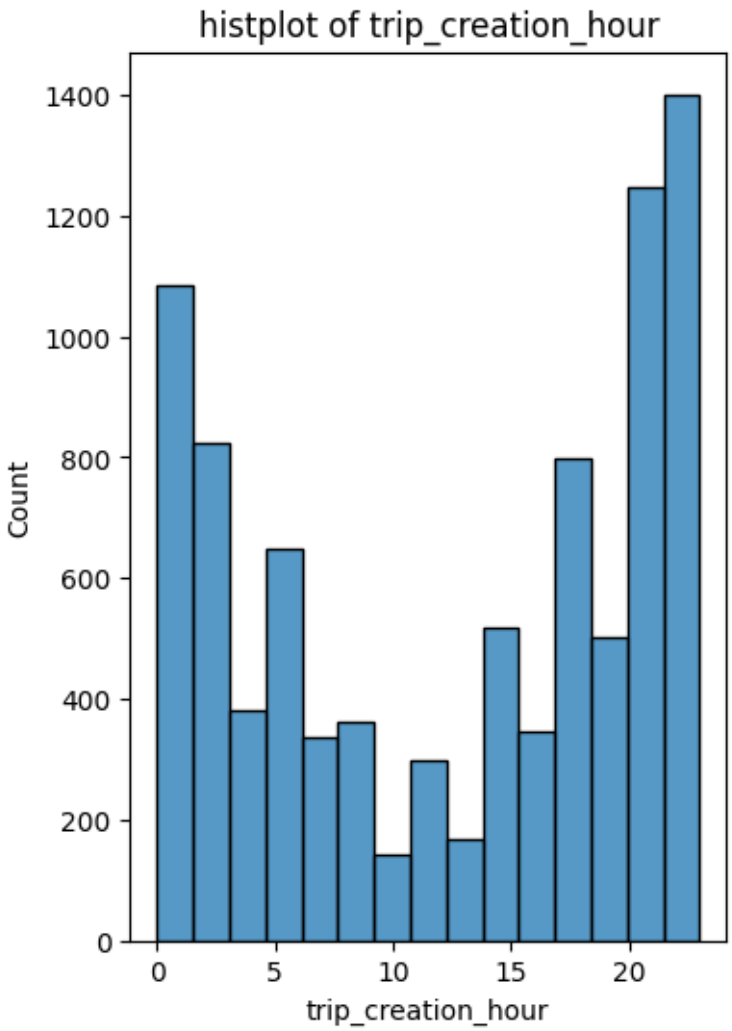

histplot of segment_osrm_time_sum
Count
segment_osrm_time_sum
0
200
400
600
800
1000
0
100
200
300
400

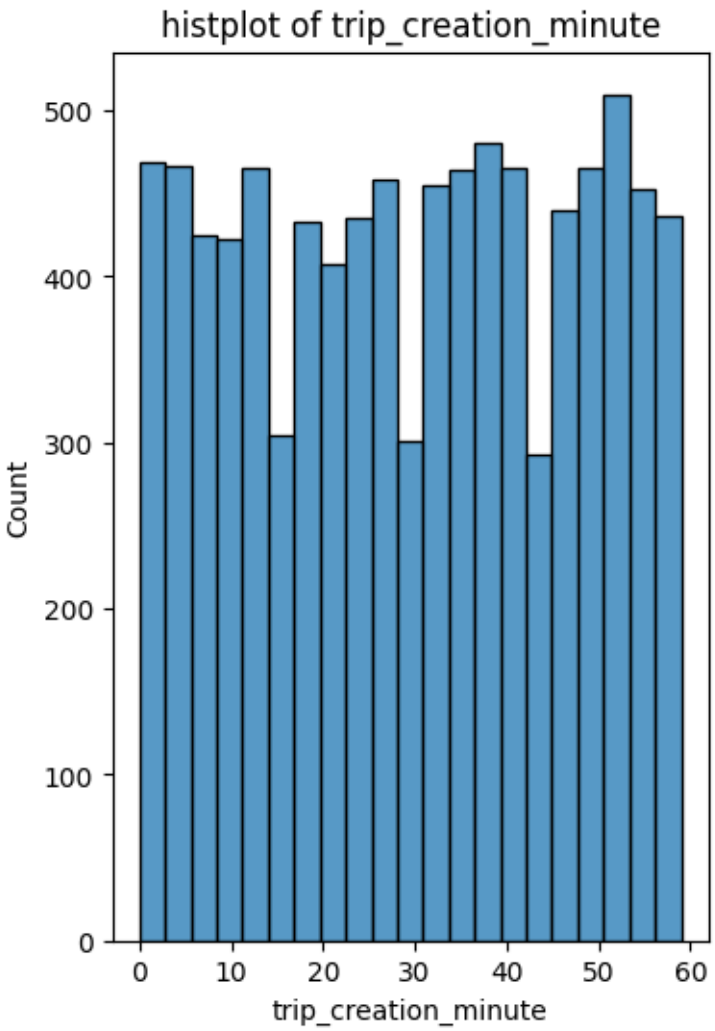

histplot of od_time_diff_hr
Count
od_time_diff_hr
0
100
200
300
400
500
600
700
0
5
10
15
20

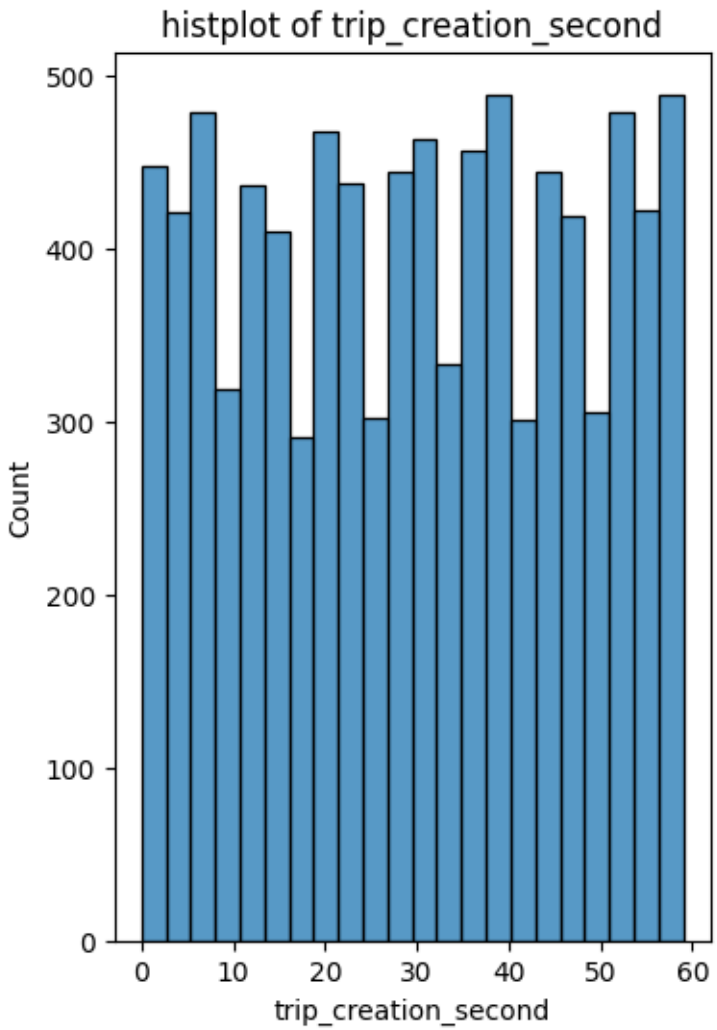

histplot of trip_creation_year
8000
6000
4000
2000
0
Count
2017.6
2017.8
2018.0
2018.2
2018.4
trip_creation_year


histplot of trip_creation_month
8000
6000
4000
2000
0
Count
8.6
8.8
9.0
9.2
9.4
trip_creation_month


histplot of trip_creation_day
Count
0
100
200
300
400
500
15
20
25
30
trip_creation_day


histplot of trip_creation_hour
Count
trip_creation_hour
0
200
400
600
800
1000
1200
1400
0
5
10
15
20


histplot of trip_creation_minute
Count
trip_creation_minute
0
100
200
300
400
500
0
10
20
30
40
50
60



21. Using Label Encoder to convert some key categorical columns to number format for better identification, interpreation and analysis.

[160]:
```
from sklearn.preprocessing import OneHotEncoder
OneHot_encoder = OneHotEncoder(sparse = False)
import warnings
warnings.filterwarnings("ignore")

categorical_columns = ['data']

for column in categorical_columns:
  trip_df[column] = OneHot_encoder.fit_transform(trip_df[[column]])
```

22. Applying Log transformation and then scaling operation to numerical columns to increase the normality and reduce the variance among values inside each column.

[161]:
```
from sklearn.preprocessing import StandardScaler, MinMaxScaler

log_transform_columns = [
    'start_scan_to_end_scan', 'actual_distance_to_destination', 'actual_time',
    'osrm_time', 'osrm_distance', 'segment_actual_time', 'segment_osrm_time',
    'segment_osrm_distance', 'segment_actual_time_sum',␣
 ↪'segment_osrm_distance_sum',
    'segment_osrm_time_sum', 'od_time_diff_hr'
]

for column in log_transform_columns:
    trip_df[column] = np.log1p(trip_df[column])

scaler = StandardScaler()
trip_df[log_transform_columns] = scaler.
 ↪fit_transform(trip_df[log_transform_columns])

minmax_scaler_columns = ['factor', 'segment_factor']
minmax_scaler = MinMaxScaler()
trip_df[minmax_scaler_columns] = minmax_scaler.
 ↪fit_transform(trip_df[minmax_scaler_columns])
```

[162]:
```
trip_df.head()
```

[162]:

	trip_uuid	segment_key	\
1	trip-153671042288605164	trip-153671042288605164_IND561203AAB_IND562101AAA	
3	trip-153671046011330457	trip-153671046011330457_IND400072AAB_IND401104AAA	
4	trip-153671052974046625	trip-153671052974046625_IND583101AAA_IND583201AAA	
5	trip-153671055416136166	trip-153671055416136166_IND600056AAA_IND602105AAB	
6	trip-153671066201138152	trip-153671066201138152_IND600044AAD_IND600048AAA	

	data	route_schedule_uuid	route_type	\
1	0.0	thanos::sroute:3a1b0ab2-bb0b-4c53-8c59-eb2a2c0…	Carting	
3	0.0	thanos::sroute:f0176492-a679-4597-8332-bbd1c7f…	Carting	
4	0.0	thanos::sroute:d9f07b12-65e0-4f3b-bec8-df06134…	FTL	
5	0.0	thanos::sroute:9bf03170-d0a2-4a3f-aa4d-9aaab3d…	Carting	
6	0.0	thanos::sroute:a97698cc-846e-41a7-916b-88b1741…	Carting	

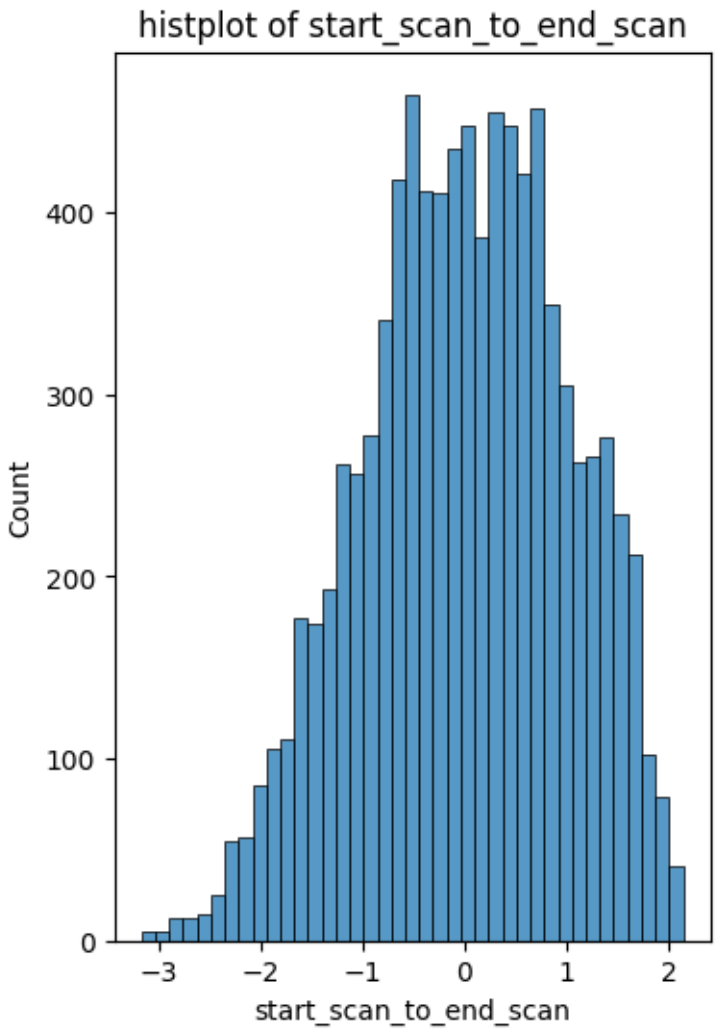
	source_center	destination_center	start_scan_to_end_scan	is_cutoff	\
1	IND561203AAB	IND561203AAB	0.239841	True	
3	IND400072AAB	IND401104AAA	-1.220519	True	
4	IND583101AAA	IND583119AAA	0.782042	True	
5	IND600056AAA	IND600056AAA	-1.009092	True	
6	IND600044AAD	IND600048AAA	-1.906860	True	

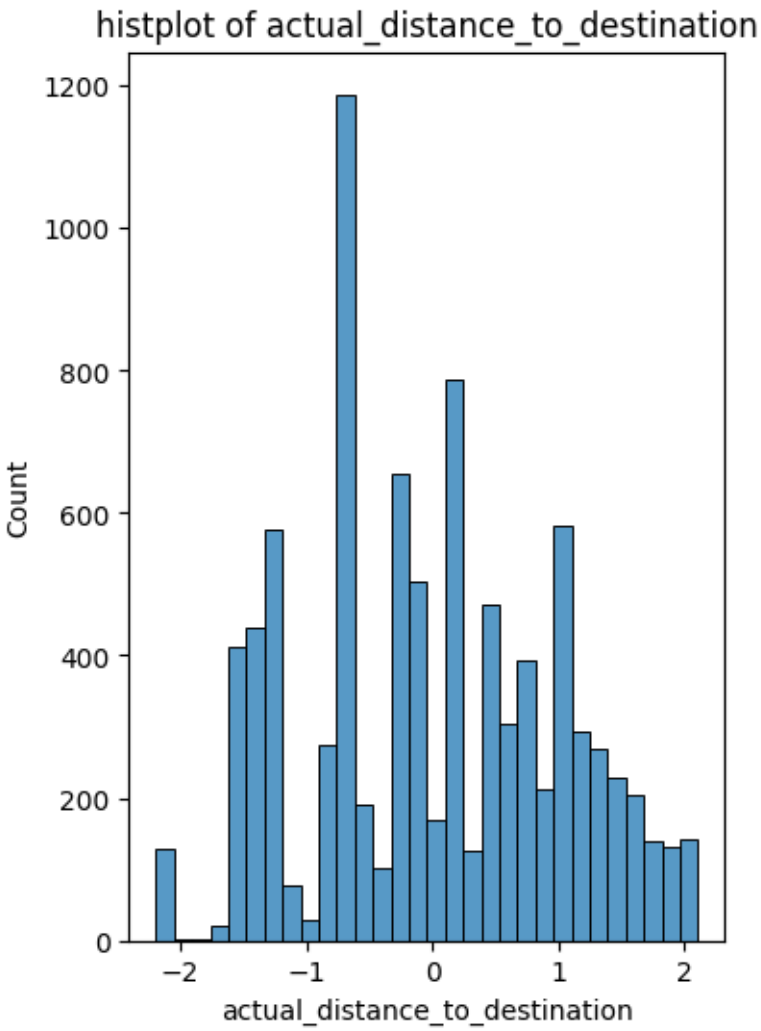
	cutoff_factor	…	source_city	source_place_code	source_code	\
1	18	…	Doddablpur	ChikaDPP	D	

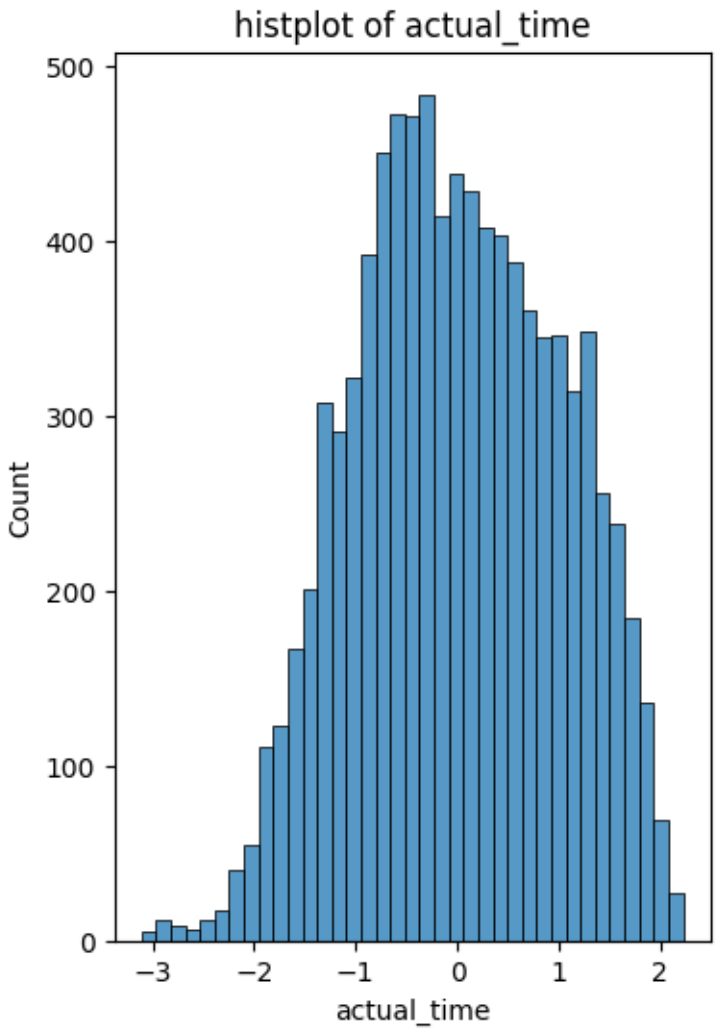
```
3              9  …        None               None          Hub
4             66  …     Bellary            WrdN1DPP           Dc
5             18  …     Chennai               Porur  Poonamallee
6              9  …     Chennai            Chrompet          DPC

   source_state  trip_creation_year  trip_creation_month  trip_creation_day  \
1     Karnataka                2018                    9                 12
3   Maharashtra                2018                    9                 12
4     Karnataka                2018                    9                 12
5    Tamil Nadu                2018                    9                 12
6    Tamil Nadu                2018                    9                 12

   trip_creation_hour  trip_creation_minute  trip_creation_second
1                   0                     0                    22
3                   0                     1                     0
4                   0                     2                     9
5                   0                     2                    34
6                   0                     4                    22

[5 rows x 38 columns]
```

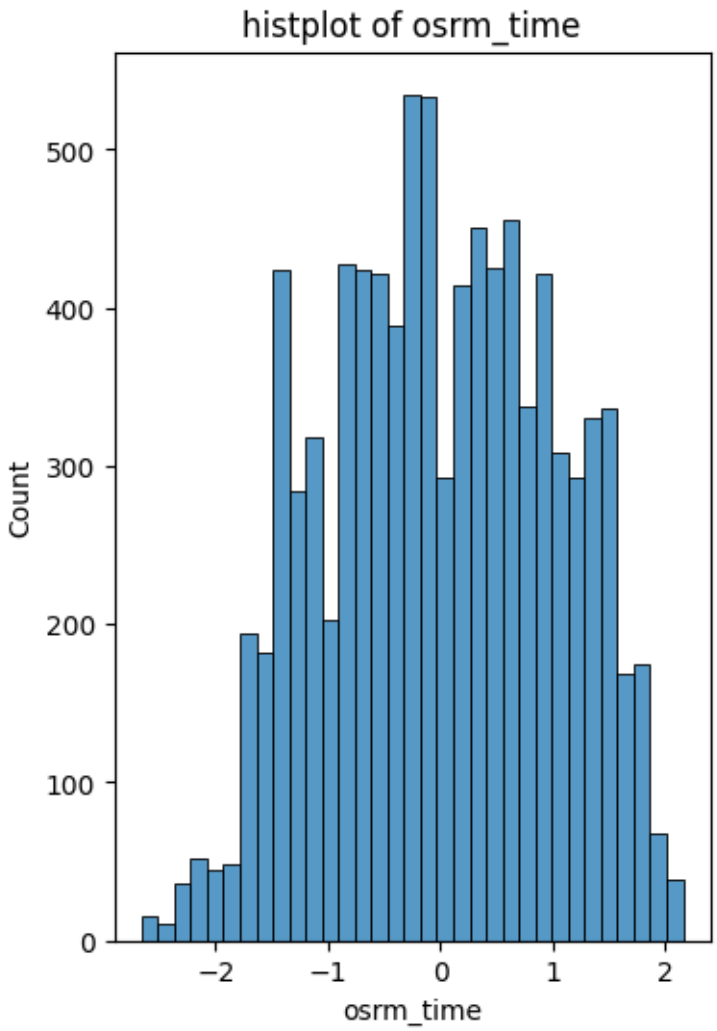
23. Histogram Visualization showing approximation to normality for each numerical column after transformation and scaling.

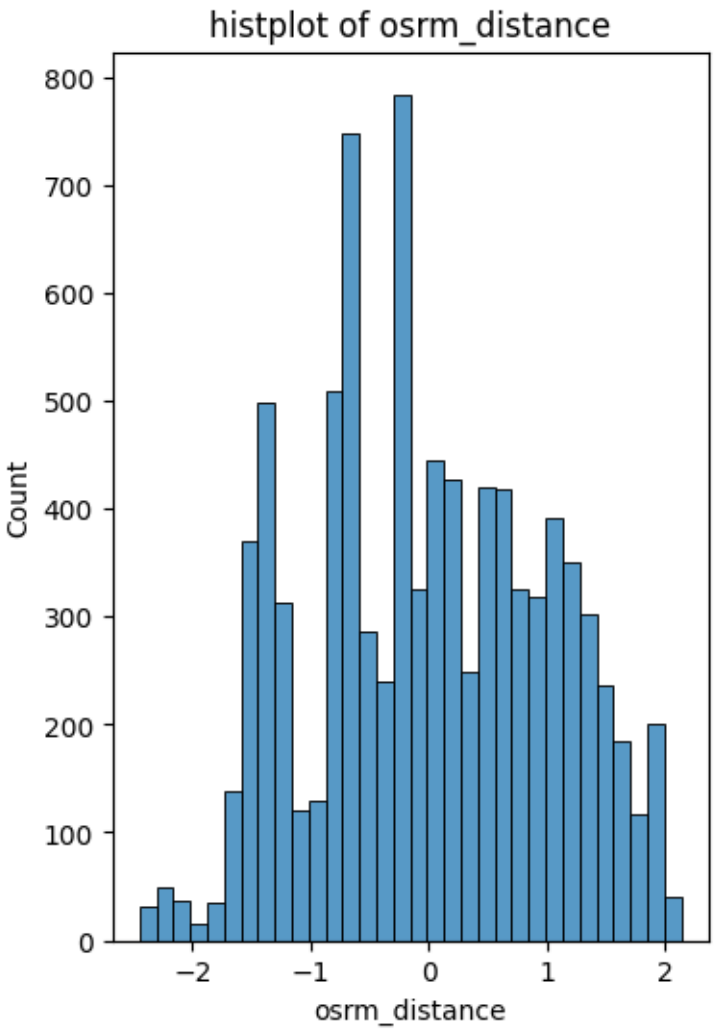
[116]:
```
for col in trip_df[log_transform_columns]:
  plt.figure(figsize = (4, 6))
  sns.histplot(x = trip_df[col])
  plt.title(f"histplot of {col}")
  plt.xlabel(col)
  plt.show()
```

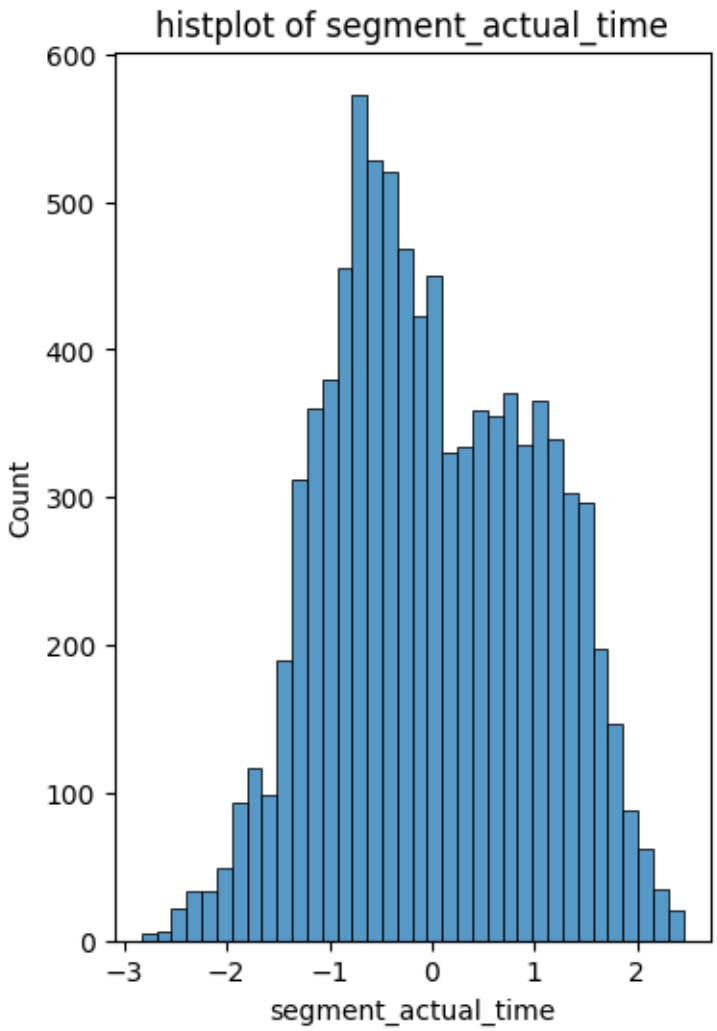

histplot of start_scan_to_end_scan
Count
0
100
200
300
400
−3
−2
−1
0
1
2
start_scan_to_end_scan

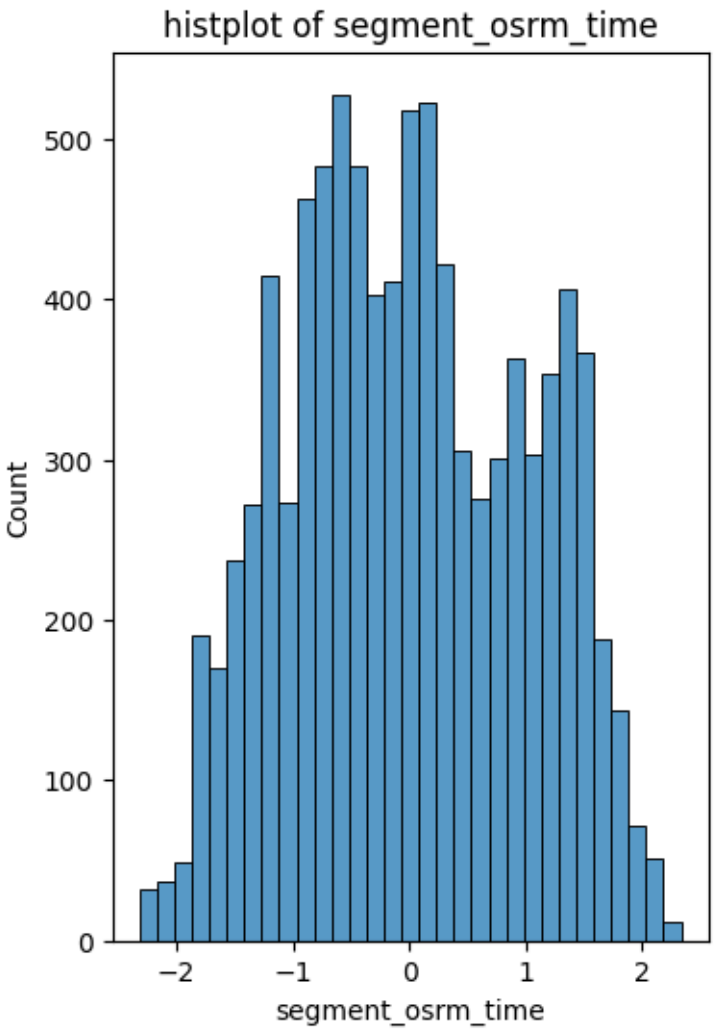

histplot of actual_distance_to_destination
Count
1200
1000
800
600
400
200
0
−2
−1
0
1
2
actual_distance_to_destination

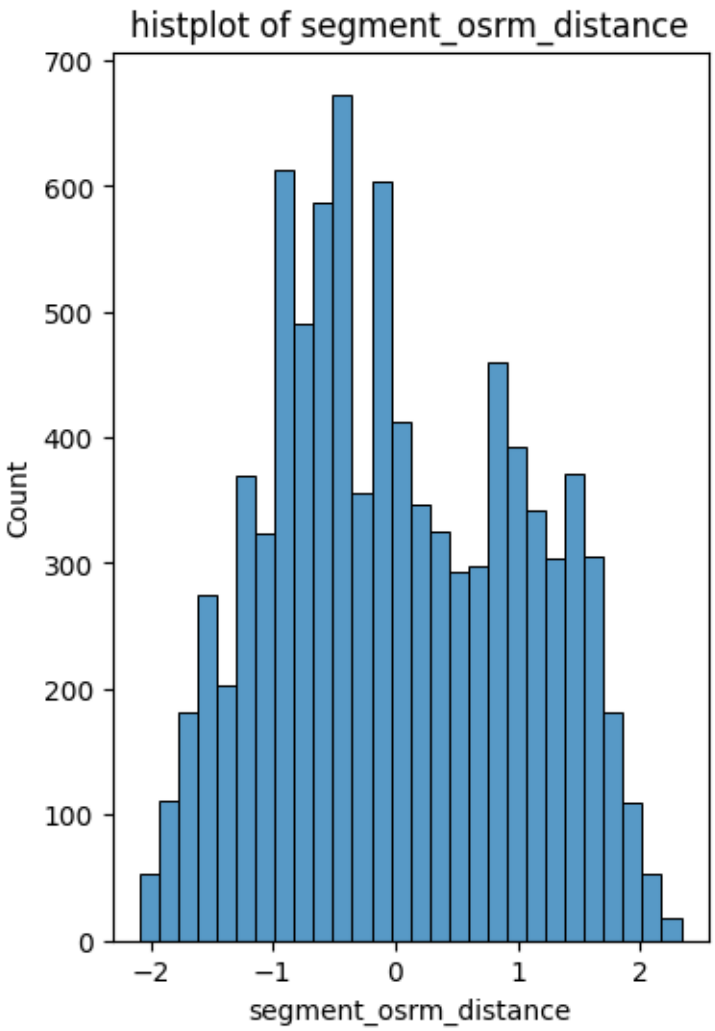

histplot of actual_time
500
400
300
200
100
0
Count
−3
−2
−1
0
1
2
actual_time

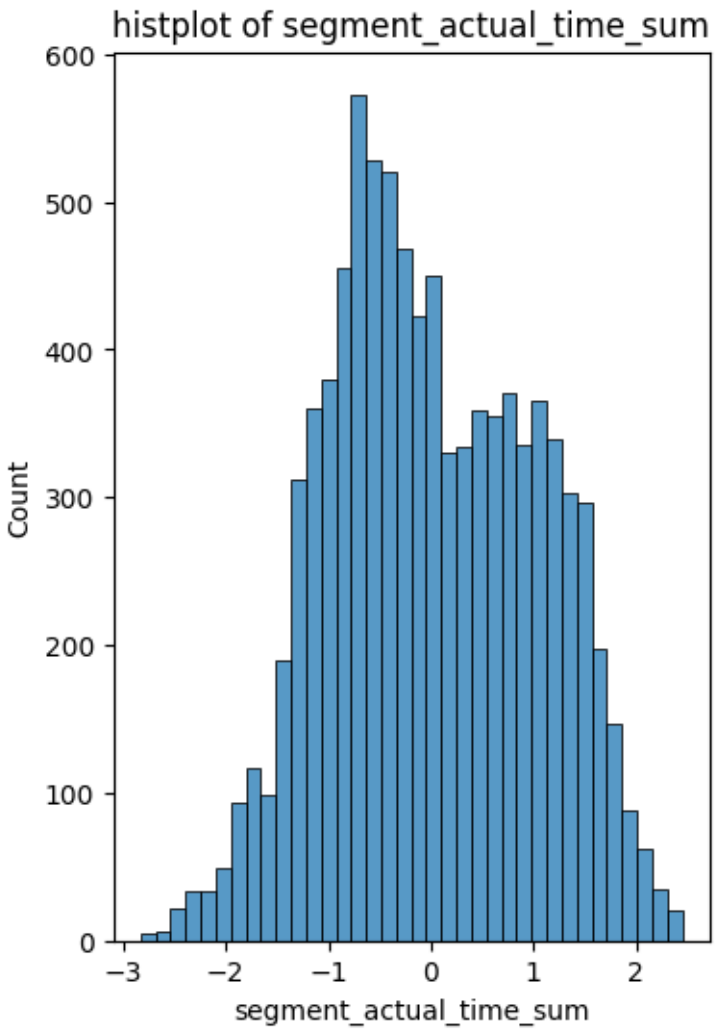

histplot of osrm_time
Count
osrm_time
0
100
200
300
400
500
−2
−1
0
1
2

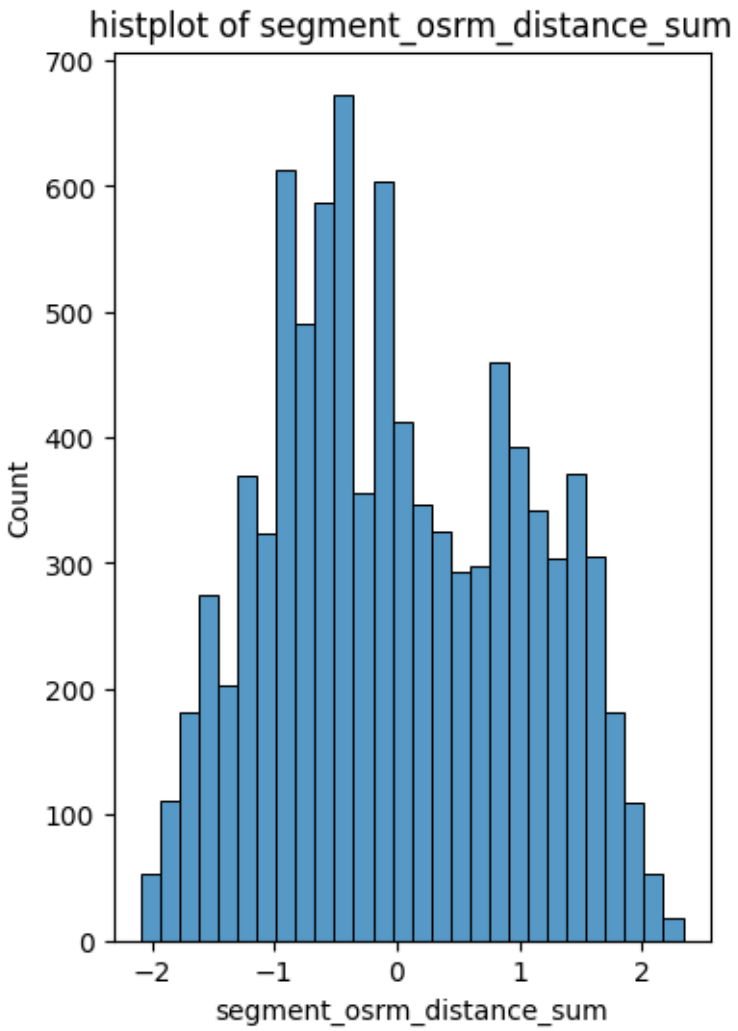

histplot of osrm_distance
Count
osrm_distance
800
700
600
500
400
300
200
100
0
−2
−1
0
1
2

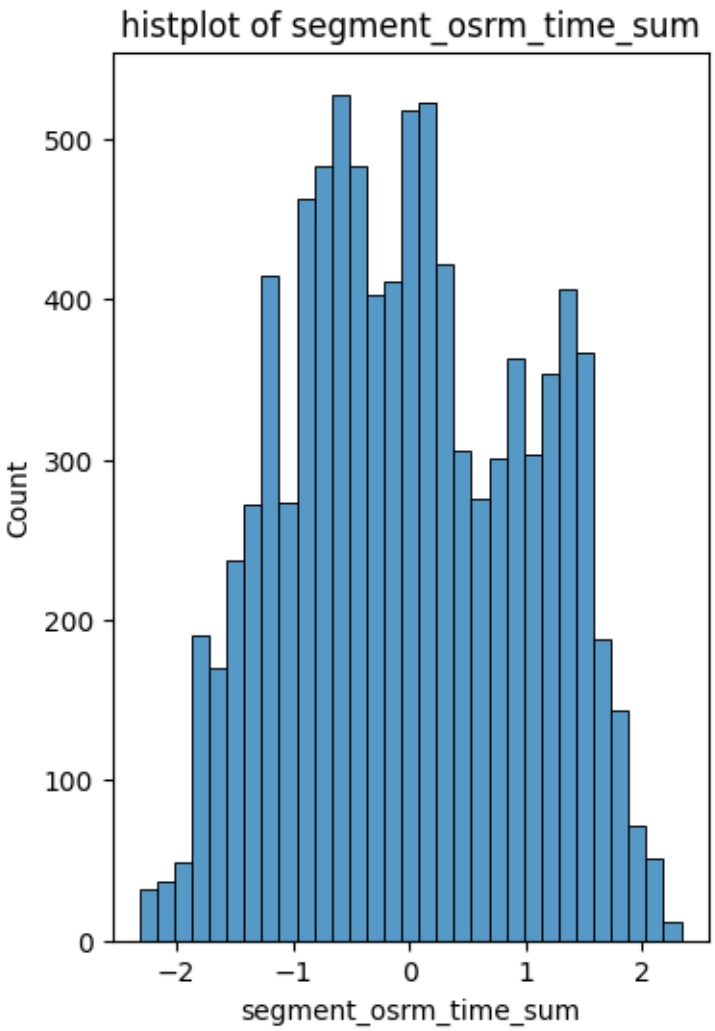

histplot of segment_actual_time
600
500
400
300
200
100
0
Count
−3
−2
−1
0
1
2
segment_actual_time

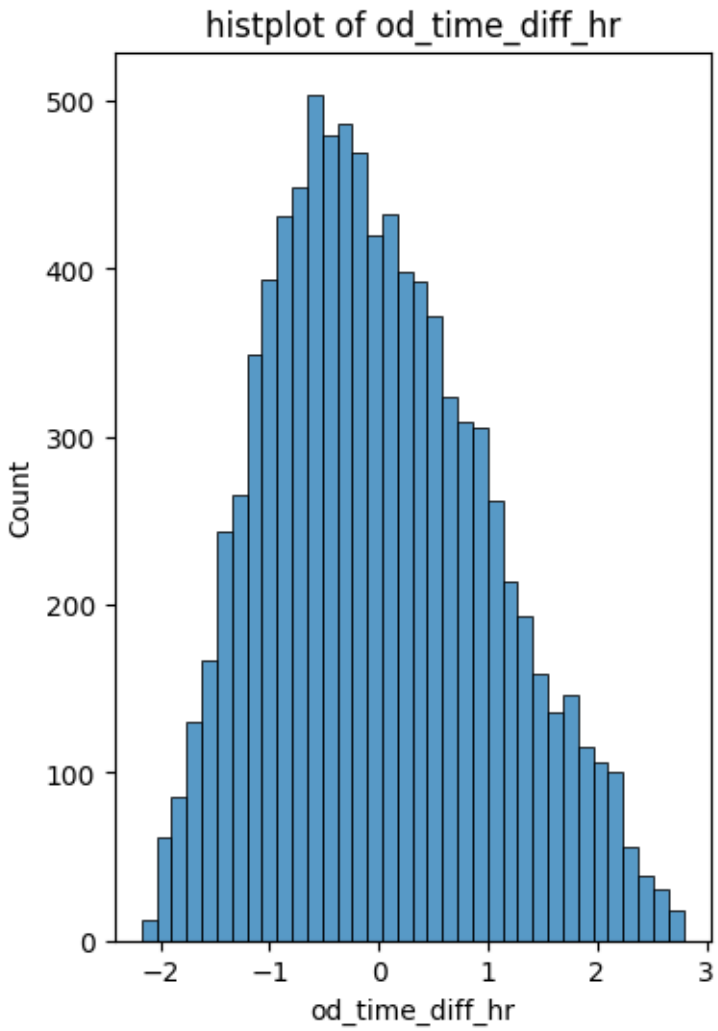

histplot of segment_osrm_time
Count
0
100
200
300
400
500
−2
−1
0
1
2
segment_osrm_time


histplot of segment_osrm_distance
Count
segment_osrm_distance
700
600
500
400
300
200
100
0
−2
−1
0
1
2


histplot of segment_actual_time_sum
600
500
400
300
200
100
0
Count
−3
−2
−1
0
1
2
segment_actual_time_sum


histplot of segment_osrm_distance_sum
Count
segment_osrm_distance_sum
700
600
500
400
300
200
100
0
−2
−1
0
1
2


histplot of segment_osrm_time_sum
Count
500
400
300
200
100
0
−2
−1
0
1
2
segment_osrm_time_sum



24. Reevaluating the number of columns present in the dataset along with their datatypes.

[163]:
```
trip_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 9050 entries, 1 to 13027
Data columns (total 38 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   trip_uuid                    9050 non-null   object
 1   segment_key                  9050 non-null   object
 2   data                         9050 non-null   float64
 3   route_schedule_uuid          9050 non-null   object
 4   route_type                   9050 non-null   object
 5   source_center                9050 non-null   object
 6   destination_center           9050 non-null   object
```

```
 7   start_scan_to_end_scan          9050 non-null   float64
 8   is_cutoff                       9050 non-null   bool
 9   cutoff_factor                   9050 non-null   int64
 10  cutoff_timestamp                9050 non-null   object
 11  actual_distance_to_destination  9050 non-null   float64
 12  actual_time                     9050 non-null   float64
 13  osrm_time                       9050 non-null   float64
 14  osrm_distance                   9050 non-null   float64
 15  factor                          9050 non-null   float64
 16  segment_actual_time             9050 non-null   float64
 17  segment_osrm_time               9050 non-null   float64
 18  segment_osrm_distance           9050 non-null   float64
 19  segment_factor                  9050 non-null   float64
 20  segment_actual_time_sum         9050 non-null   float64
 21  segment_osrm_distance_sum       9050 non-null   float64
 22  segment_osrm_time_sum           9050 non-null   float64
 23  od_time_diff_hr                 9050 non-null   float64
 24  destination_city                8580 non-null   object
 25  destination_place_code          8161 non-null   object
 26  destination_code                9045 non-null   object
 27  destination_state               9045 non-null   object
 28  source_city                     8672 non-null   object
 29  source_place_code               8271 non-null   object
 30  source_code                     9045 non-null   object
 31  source_state                    9045 non-null   object
 32  trip_creation_year              9050 non-null   int32
 33  trip_creation_month             9050 non-null   int32
 34  trip_creation_day               9050 non-null   int32
 35  trip_creation_hour              9050 non-null   int32
 36  trip_creation_minute            9050 non-null   int32
 37  trip_creation_second            9050 non-null   int32
dtypes: bool(1), float64(15), int32(6), int64(1), object(15)
memory usage: 2.4+ MB
```

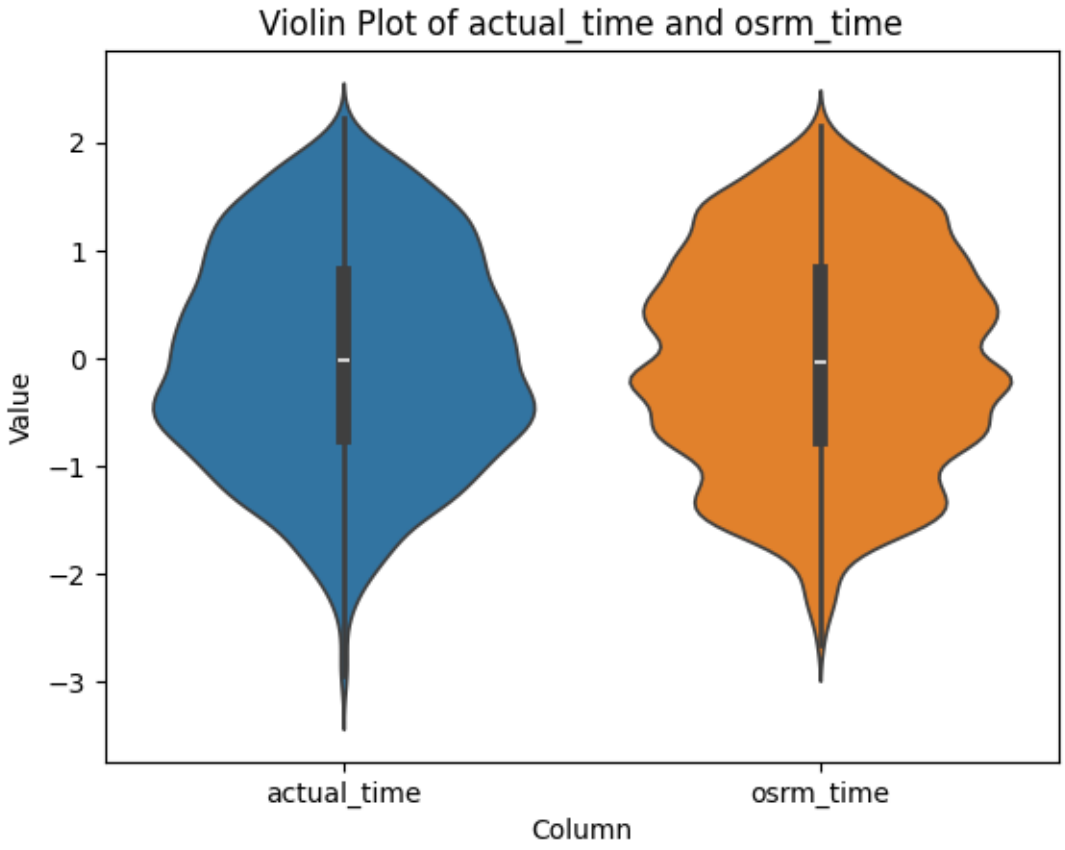
25. Visualization and Hypothesis testing of key numeric columns to find out whether there is a difference between estimation and actuals.

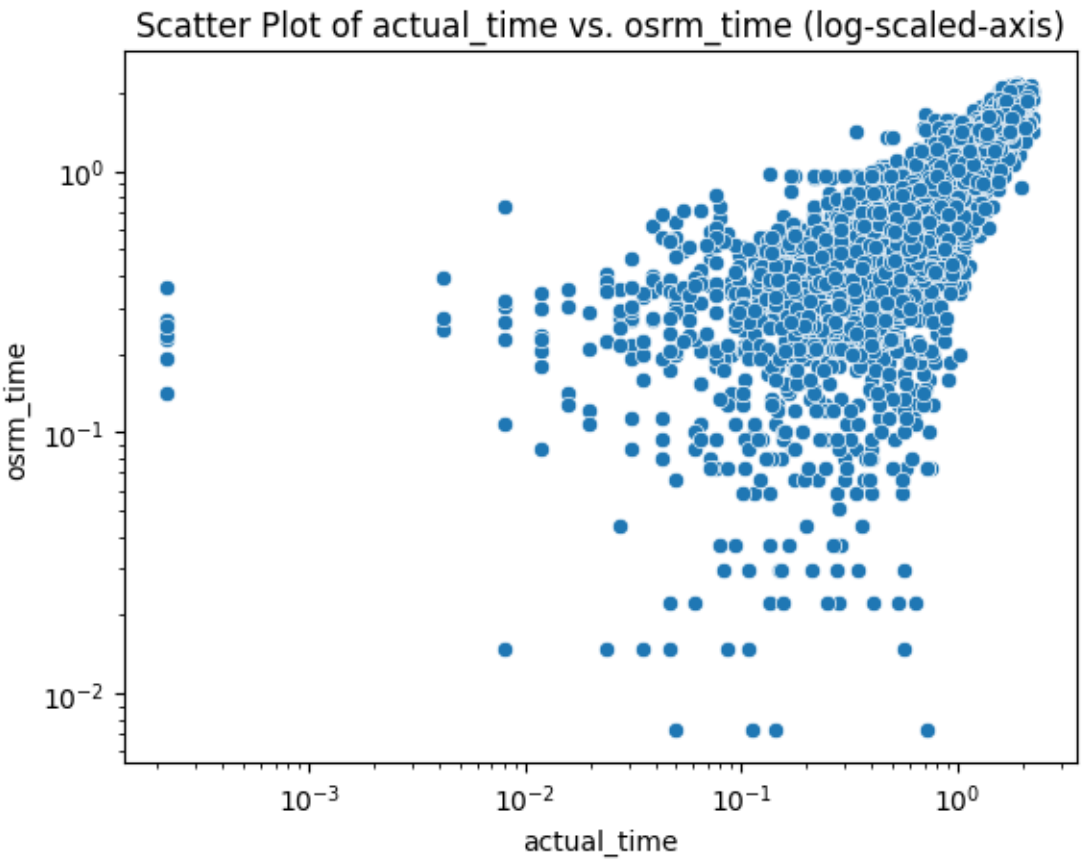
[118]:
```
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import ttest_rel
from scipy.stats import wilcoxon

# H0: There is no differnece between means of the grouped data.
# H1: There is a significant differnece betweeen means of the grouped data.

# Alpha = 0.05

def analyze_columns(trip_df, col1, col2):
```

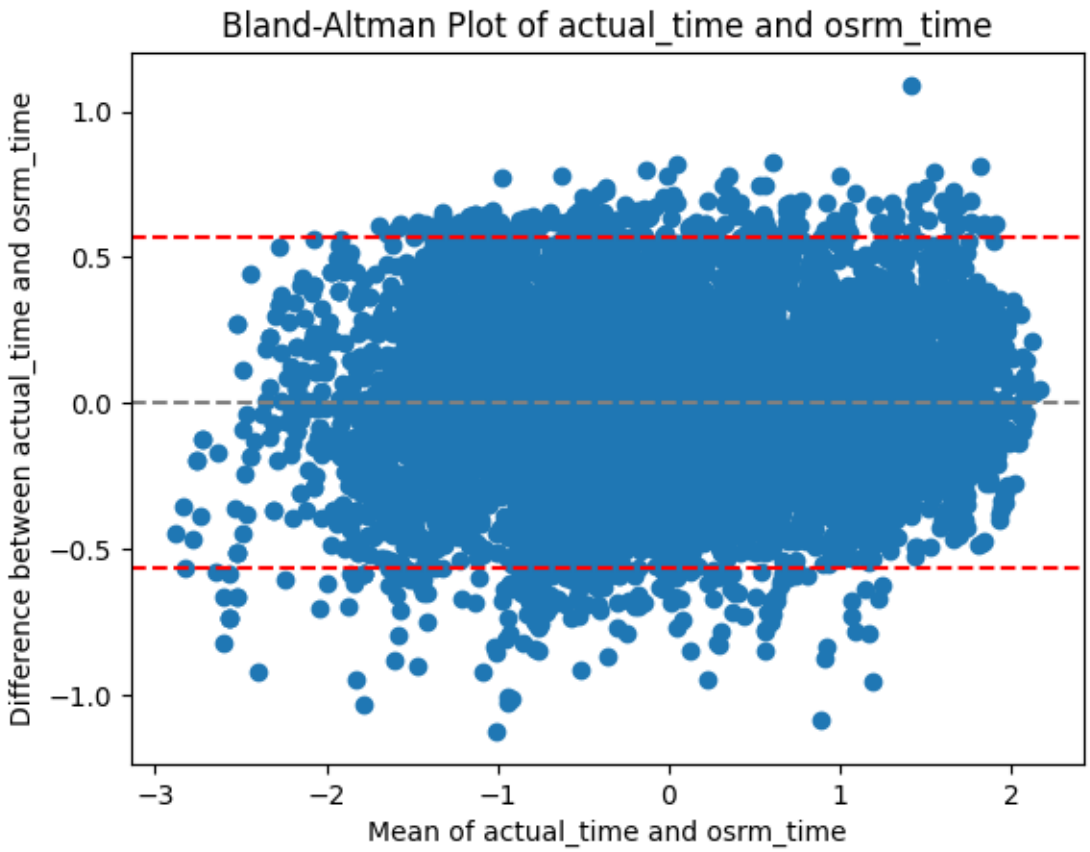
```
    print(f"Analyzing {col1} vs. {col2}")

    # Paired T-Test
    t_stat, p_val = ttest_rel(trip_df[col1], trip_df[col2])
    print(f'Paired T-Test: t-statistic = {t_stat:.3f}, p-value = {p_val:.3f}')

    print("<------------------------------------------------------------>")

    # wilcoxon test
    stat, p_val = wilcoxon(trip_df[col1], trip_df[col2])
    print(f'Wilcoxon Signed-Rank Test: statistic = {stat:.3f}, p-value = {p_val:
↪.3f}')

    # Violin Plot
    sns.violinplot(data= trip_df[[col1, col2]], scale = 'width')
    plt.title(f'Violin Plot of {col1} and {col2}')
    plt.ylabel('Value')
    plt.xlabel('Column')
    plt.show()

    # Scatter Plot
    sns.scatterplot(x=trip_df[col1], y=trip_df[col2])
    plt.xscale('log')  # Log scale for x-axis
    plt.yscale('log')  # Log scale for y-axis
    plt.xlabel(col1)
    plt.ylabel(col2)
    plt.title(f'Scatter Plot of {col1} vs. {col2} (log-scaled-axis)')
    plt.show()

    # Bland-Altman Plot
    mean_diff = trip_df[col1] - trip_df[col2]
    mean_values = (trip_df[col1] + trip_df[col2]) / 2
    plt.scatter(mean_values, mean_diff)
    plt.axhline(np.mean(mean_diff), color='gray', linestyle='--')
    plt.axhline(np.mean(mean_diff) + 1.96 * np.std(mean_diff), color='red',␣
↪linestyle='--')
    plt.axhline(np.mean(mean_diff) - 1.96 * np.std(mean_diff), color='red',␣
↪linestyle='--')
    plt.xlabel(f'Mean of {col1} and {col2}')
    plt.ylabel(f'Difference between {col1} and {col2}')
    plt.title(f'Bland-Altman Plot of {col1} and {col2}')
    plt.show()


column_pairs = [('actual_time', 'osrm_time'),
                ('actual_time', 'segment_actual_time_sum'),
                ('osrm_distance', 'segment_osrm_distance_sum'),
```

```
                ('osrm_time', 'segment_osrm_time_sum')]

for col1, col2 in column_pairs:
    analyze_columns(trip_df, col1, col2)
```

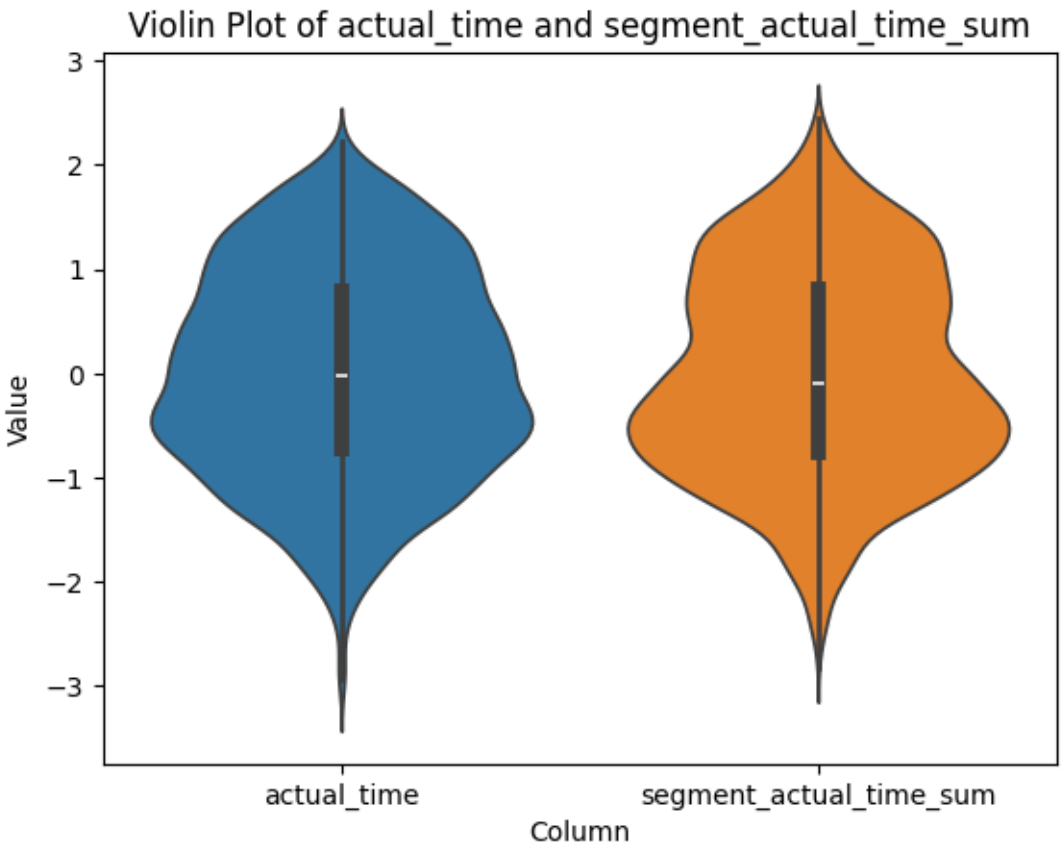
```
Analyzing actual_time vs. osrm_time
Paired T-Test: t-statistic = 0.000, p-value = 1.000
<------------------------------------------------------------>
Wilcoxon Signed-Rank Test: statistic = 20376841.000, p-value = 0.684
```

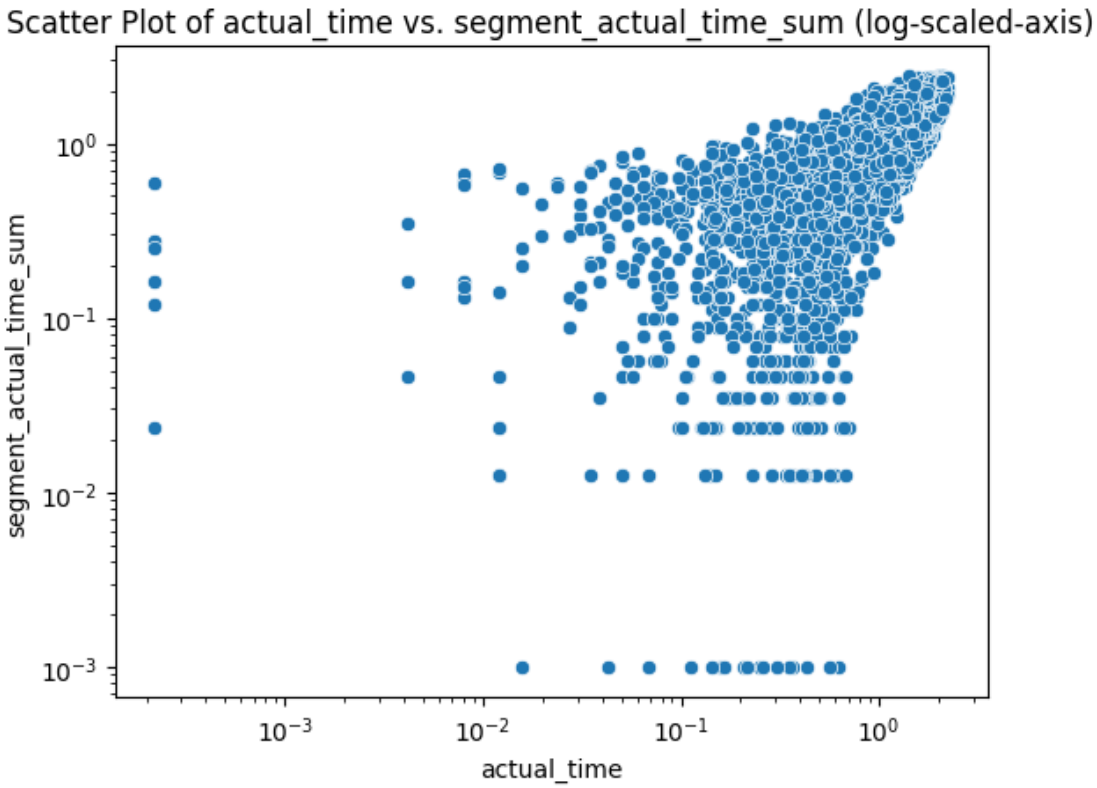


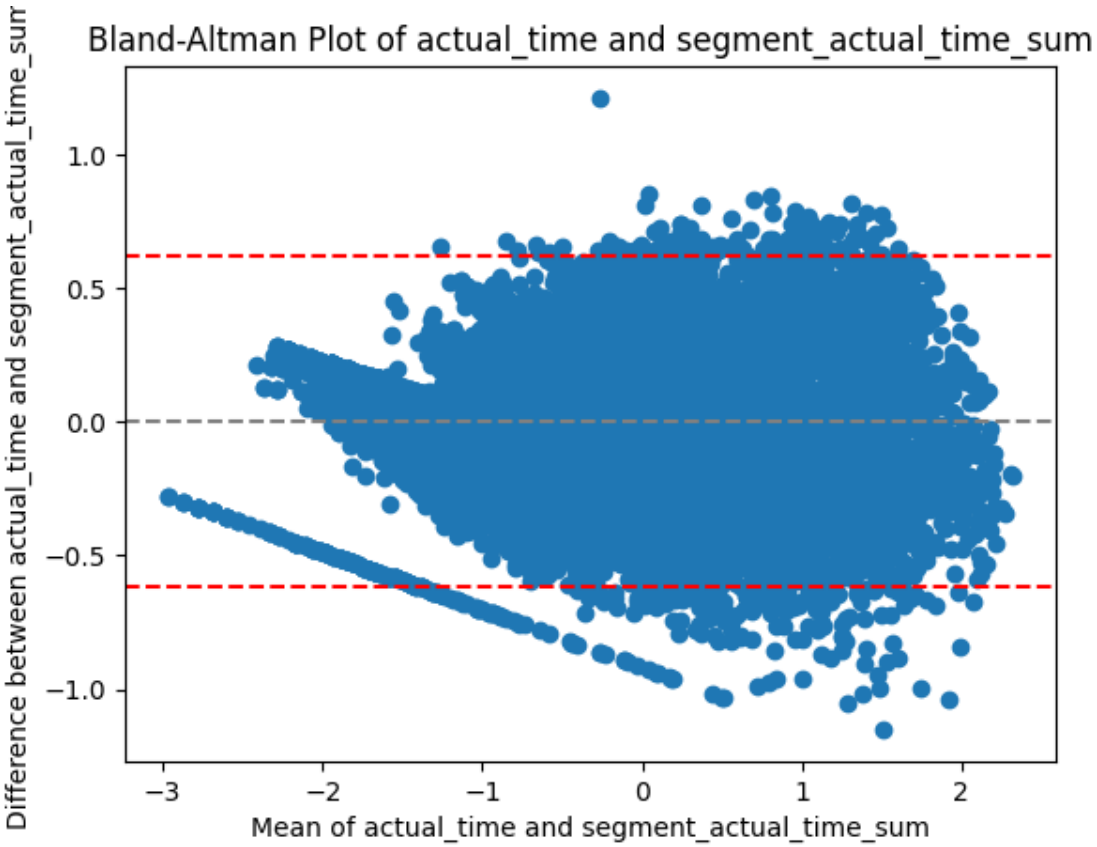

Scatter Plot of actual_time vs. osrm_time (log-scaled-axis)
osrm_time
actual_time
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-2}
10^{-1}
10^{0}



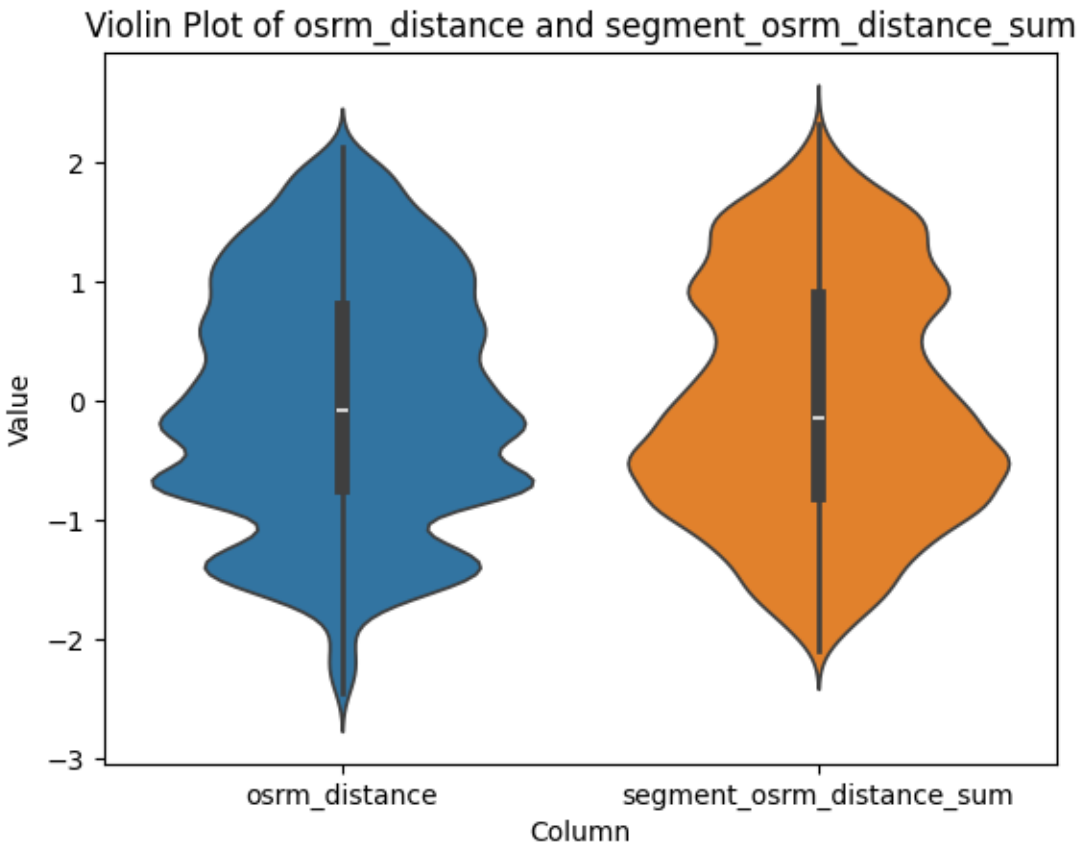

```
Analyzing actual_time vs. segment_actual_time_sum
Paired T-Test: t-statistic = -0.000, p-value = 1.000
<------------------------------------------------------------>
Wilcoxon Signed-Rank Test: statistic = 20103639.000, p-value = 0.132
```

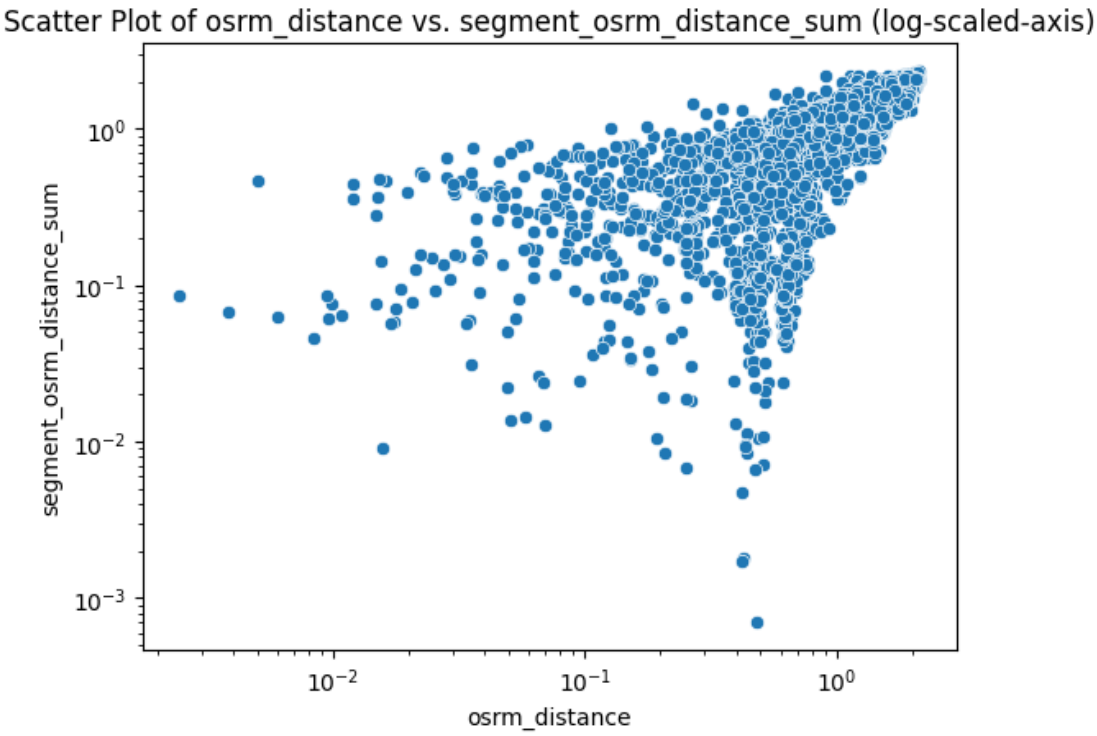

Violin Plot of actual_time and segment_actual_time_sum
Value
3
2
1
0
−1
−2
−3
actual_time
segment_actual_time_sum
Column

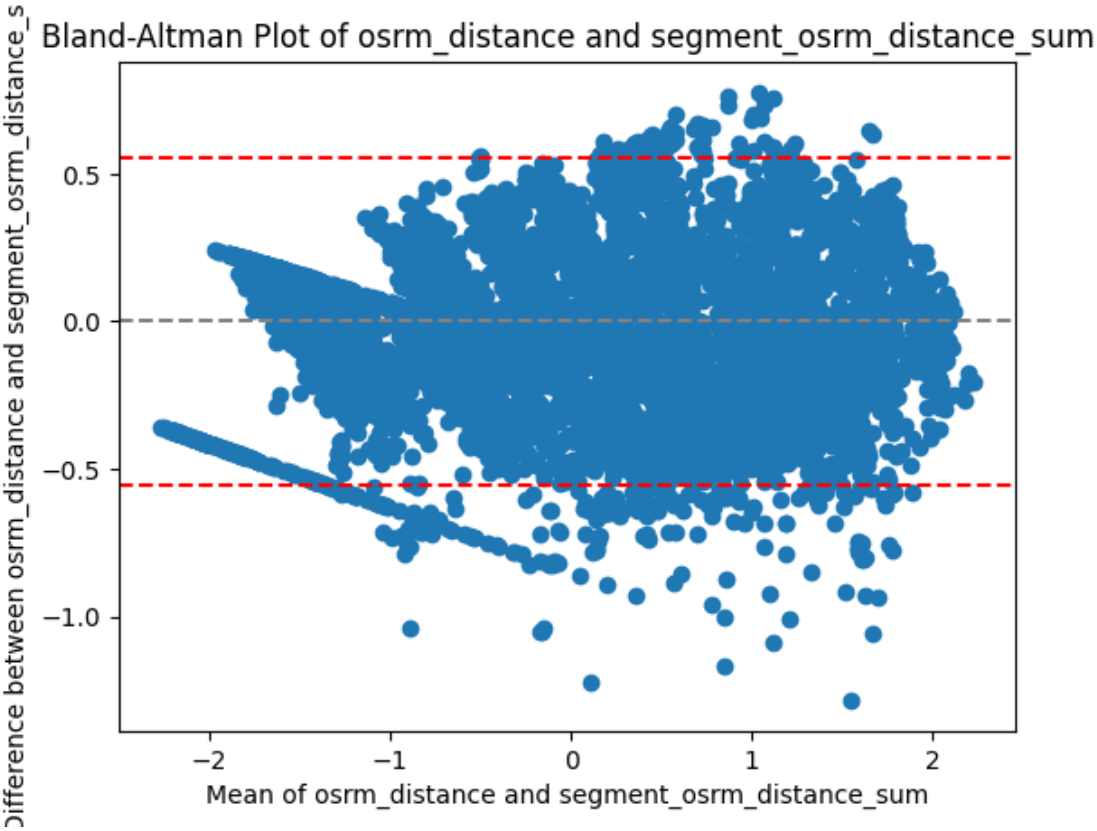

Scatter Plot of actual_time vs. segment_actual_time_sum (log-scaled-axis)
segment_actual_time_sum
actual_time
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-3}
10^{-2}
10^{-1}
10^{0}



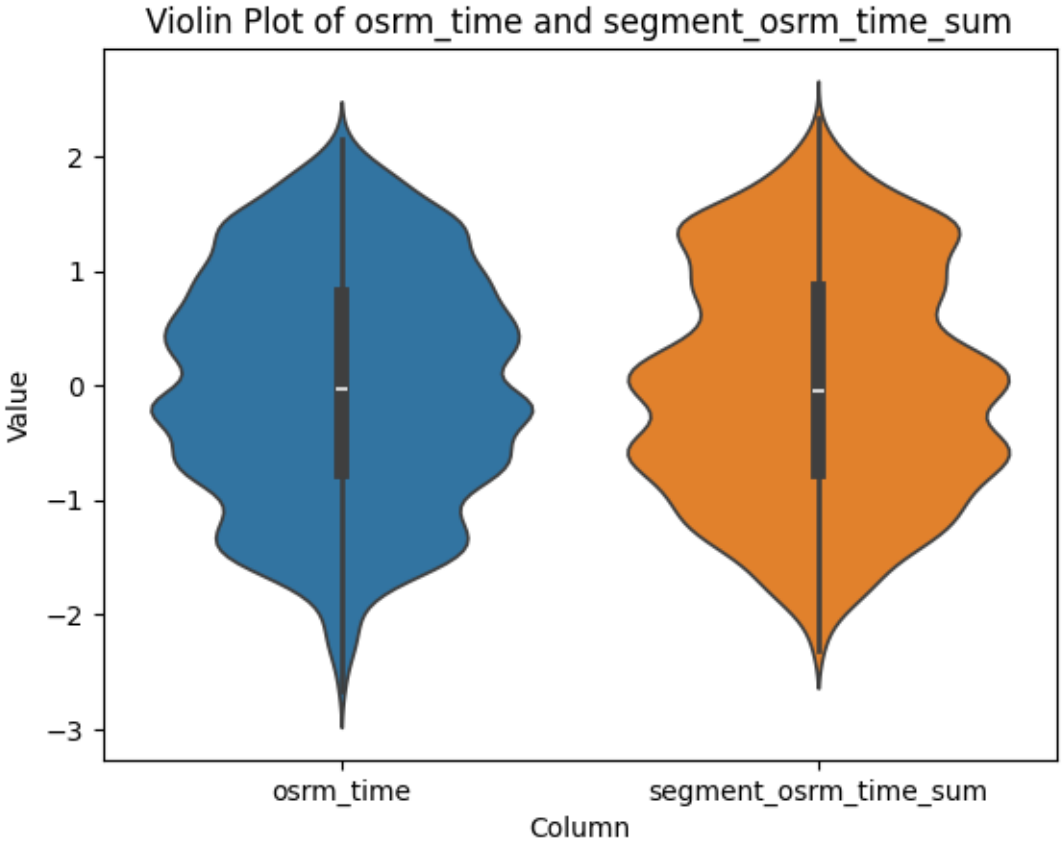

```
Analyzing osrm_distance vs. segment_osrm_distance_sum
Paired T-Test: t-statistic = -0.000, p-value = 1.000
<------------------------------------------------------------>
Wilcoxon Signed-Rank Test: statistic = 19994050.000, p-value = 0.052
```

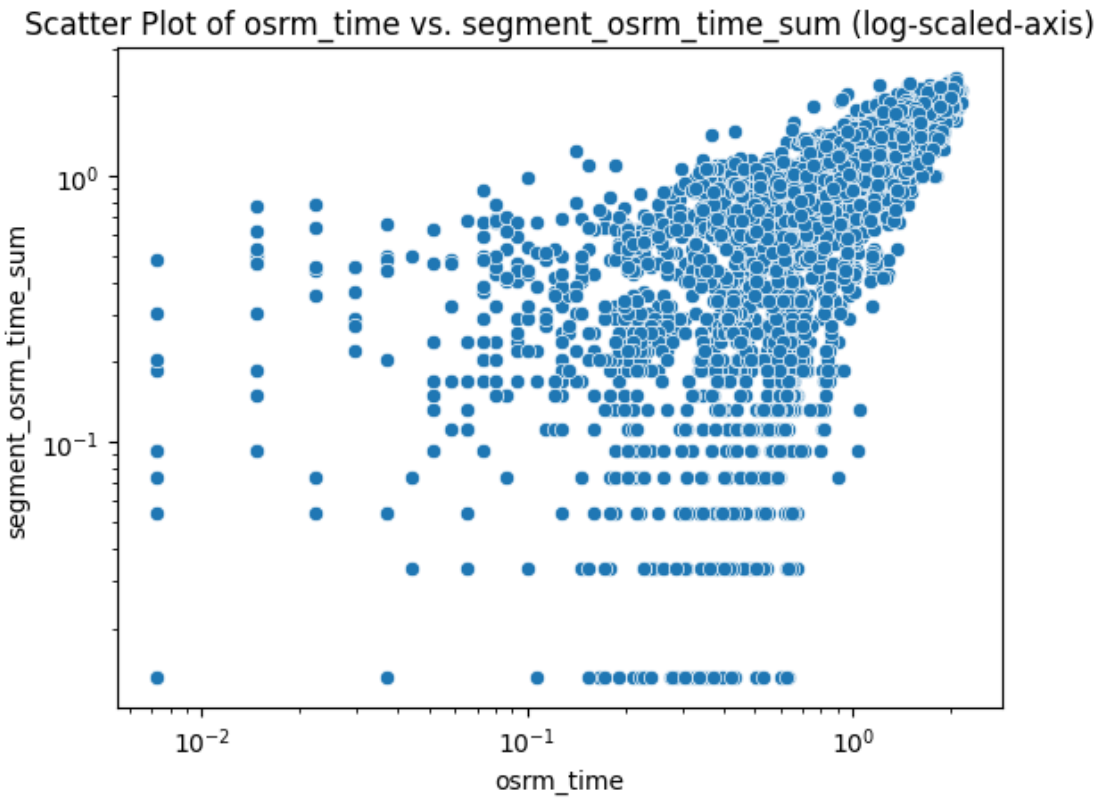

Violin Plot of osrm_distance and segment_osrm_distance_sum
Value
2
1
0
−1
−2
−3
osrm_distance
segment_osrm_distance_sum
Column

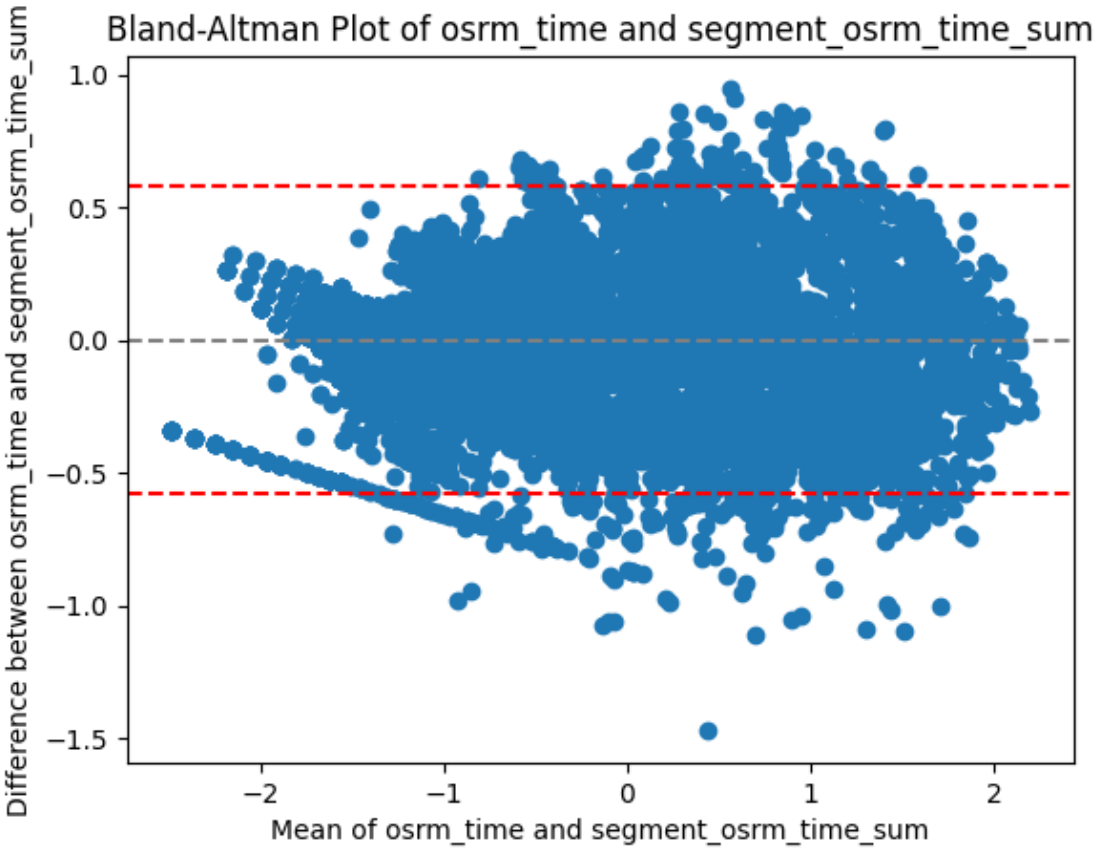

Scatter Plot of osrm_distance vs. segment_osrm_distance_sum (log-scaled-axis)
segment_osrm_distance_sum
osrm_distance
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-2}
10^{-1}
10^{0}


Bland-Altman Plot of osrm_distance and segment_osrm_distance_sum
Difference between osrm_distance and segment_osrm_distance_s
Mean of osrm_distance and segment_osrm_distance_sum
0.5
0.0
−0.5
−1.0
−2
−1
0
1
2

```
Analyzing osrm_time vs. segment_osrm_time_sum
Paired T-Test: t-statistic = -0.000, p-value = 1.000
<------------------------------------------------------------>
Wilcoxon Signed-Rank Test: statistic = 20137666.000, p-value = 0.171
```




Scatter Plot of osrm_time vs. segment_osrm_time_sum (log-scaled-axis)
segment_osrm_time_sum
10^0
10^{-1}
10^{-2}
10^{-1}
10^0
osrm_time



Inference:

('actual_time', 'osrm_time')

Violin Plot: The distributions of actual_time and osrm_time are similar in central tendency and spread, indicating that the predicted times generally match the actual times, though variability and outliers are present.

Scatter Plot: There is a positive correlation between actual_time and osrm_time, with significant variability at lower time values.

Bland-Altman Plot: The differences between actual_time and osrm_time are mostly centered around zero, indicating no systematic bias, but there are some outliers that reflect discrepancies between the two measurements.

('actual_time', 'segment actual time sum')

Violin plot: Both distributions are centered around zero with similar spread, but the shapes indicate slightly different distributions.

Scatter plot: There is a positive correlation between actual_time and actual_segment_time_sum with a significant variability at lower time values.

Bland-Altman Plot: The differences between actual_time and segment_actual_time_sum are

mostly centered around zero, indicating no radical systematic bias.

('osrm_distance', 'segment_osrm_distance_sum')

Violin plot: Both distributions are centered around zero with similar spread, but the shape of osrm_distance indicate a varied peakedness with that of segment distribution, thus indicating a realible difference between distributions.

Scatter plot: There is a positive correlation between osrm_distance and segment_osrm_distance_sum with a significant variability at lower distance values.

Bland-Altman Plot: The differences between osrm_distance and segment_osrm_distance_sum are mostly centered around zero, indicating no radical systematic bias.

('osrm_time', 'segment_osrm_time_sum')

Violin plot: Both distributions are centered around zero with similar spread, but the shapes indicate slightly different distributions.

Scatter plot: There is a positive correlation between osrm_time and segment_osrm_time_sum with very significant variability at lower time values.

Bland-Altman Plot: The differences between osrm_time and segment_osrm_time_sum are mostly centered around zero, indicating no radical systematic bias.

Wilcoxon tests:

If we observe the p-values under Wilcoxon test for each combination, we can pretty much conclude that there is no major difference betweeen means of each column, excepet leaving the combination of osrm_distance and segment_osrm_distance_sum, where there is a reliable difference however still it's not below the alpha level of 0.05

26. Calculation of busiest states with more number of orders

[164]:
```
trip_df.groupby("destination_state")["segment_key"].size().
↪sort_values(ascending = False)
```

[164]:
```
destination_state
Karnataka                1614
Maharashtra              1498
Haryana                  1037
Tamil Nadu                803
Gujarat                   462
Uttar Pradesh             449
Telangana                 439
Delhi                     433
West Bengal               348
Punjab                    339
Rajasthan                 316
Andhra Pradesh            257
Kerala                    213
Bihar                     176
Madhya Pradesh            159
```

```
Assam                        105
Uttarakhand                   74
Jharkhand                     63
Chandigarh                    61
Orissa                        61
Himachal Pradesh              40
Chhattisgarh                  36
Goa                           20
Dadra and Nagar Haveli        14
Meghalaya                     10
Pondicherry                    7
Arunachal Pradesh              6
Jammu & Kashmir                4
Tripura                        1
Name: segment_key, dtype: int64
```

Inference: Karnataka, Maharastra, Haryana stood as top three states with highest number of orders > 1000.

27. Top 20 busiest cities with more number of orders

[165]:
```
trip_df.groupby("destination_city")["segment_key"].size().sort_values(ascending␣
↪= False).head(20)
```

[165]:
```
destination_city
Bengaluru     891
Mumbai        563
Gurgaon       452
Delhi         370
Bangalore     311
Chennai       303
Bhiwandi      274
Hyderabad     248
Sonipat       237
Pune          187
Chandigarh    165
MAA           135
FBD           110
Jaipur        105
Ahmedabad     102
Noida          92
Kolkata        73
CCU            69
BLR            65
Faridabad      60
Name: segment_key, dtype: int64
```

Inference: Cities Like Bengaluru, Mumbai, gurgaon stood as top three in terms of number of

orders > 400.

28. Calculation of Busiest Corridor, avg distance between them, avg time taken, etc.

[169]:
```
trip_df['Corridor'] = df['source_center'] + '-->' + df['destination_center']

corridor_counts = trip_df['Corridor'].value_counts().reset_index()
corridor_counts.columns = ['corridor', 'count']
```

[170]:
```
# Find the busiest corridor (highest frequency)
busiest_corridor = corridor_counts.iloc[0]['corridor']
```

[171]:
```
# Calculate average distance and average time for each corridor
corridor_stats = trip_df.groupby('Corridor').agg({
    'segment_osrm_distance': 'mean',
    'segment_actual_time': 'mean'
}).reset_index()
corridor_stats.columns = ['corridor', 'avg_distance', 'avg_time']
```

[173]:
```
# Merge counts with stats
corridor_summary = pd.merge(corridor_counts, corridor_stats,  on='corridor')
```

[174]:
```
# Get stats for the busiest corridor
busiest_corridor_stats = corridor_summary[corridor_summary['corridor'] ==␣
 ↪busiest_corridor]

# Display the busiest corridor and its stats
print("Busiest Corridor Stats:")
print(busiest_corridor_stats)

# Display overall corridor summary
print("\nOverall Corridor Summary:")
print(corridor_summary)
```

```
Busiest Corridor Stats:
                     corridor  count  avg_distance  avg_time
0  IND000000ACB-->IND562132AAA    278      0.151546  0.087325

Overall Corridor Summary:
                         corridor  count  avg_distance  avg_time
0     IND000000ACB-->IND562132AAA    278      0.151546  0.087325
1     IND562132AAA-->IND000000ACB    193      0.196619  0.162765
2     IND000000ACB-->IND712311AAA    176      0.216941  0.279092
3     IND000000ACB-->IND421302AAG    173     -0.002997 -0.064332
4     IND000000ACB-->IND501359AAE    116     -0.005952 -0.038451
…                             …    …            …         …
1294  IND148101AAA-->IND148102AAA      1     -0.999415 -1.019757
1295  IND263401AAA-->IND244713AAA      1      0.389077  0.373107
```

```
1296  IND263153AAB-->IND263401AAA      1      1.458529  1.528103
1297  IND575004AAB-->IND574211AAA      1      1.448117  1.347231
1298  IND382430AAB-->IND382345AAA      1      0.081928 -0.470388

[1299 rows x 4 columns]
```

Inference: Corridor "IND000000ACB–>IND562132AAA" stood as highest trade activity route with a total order amounting to 278. The avg_distance and avg_time were in logarithmic units.

[14]:
```
!pip install nbconvert
```

```
Requirement already satisfied: nbconvert in /usr/local/lib/python3.10/dist-
packages (6.5.4)
Requirement already satisfied: lxml in /usr/local/lib/python3.10/dist-packages
(from nbconvert) (4.9.4)
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.10/dist-
packages (from nbconvert) (4.12.3)
Requirement already satisfied: bleach in /usr/local/lib/python3.10/dist-packages
(from nbconvert) (6.1.0)
Requirement already satisfied: defusedxml in /usr/local/lib/python3.10/dist-
packages (from nbconvert) (0.7.1)
Requirement already satisfied: entrypoints>=0.2.2 in
/usr/local/lib/python3.10/dist-packages (from nbconvert) (0.4)
Requirement already satisfied: jinja2>=3.0 in /usr/local/lib/python3.10/dist-
packages (from nbconvert) (3.1.4)
Requirement already satisfied: jupyter-core>=4.7 in
/usr/local/lib/python3.10/dist-packages (from nbconvert) (5.7.2)
Requirement already satisfied: jupyterlab-pygments in
/usr/local/lib/python3.10/dist-packages (from nbconvert) (0.3.0)
Requirement already satisfied: MarkupSafe>=2.0 in
/usr/local/lib/python3.10/dist-packages (from nbconvert) (2.1.5)
Requirement already satisfied: mistune<2,>=0.8.1 in
/usr/local/lib/python3.10/dist-packages (from nbconvert) (0.8.4)
Requirement already satisfied: nbclient>=0.5.0 in
/usr/local/lib/python3.10/dist-packages (from nbconvert) (0.10.0)
Requirement already satisfied: nbformat>=5.1 in /usr/local/lib/python3.10/dist-
packages (from nbconvert) (5.10.4)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-
packages (from nbconvert) (24.0)
Requirement already satisfied: pandocfilters>=1.4.1 in
/usr/local/lib/python3.10/dist-packages (from nbconvert) (1.5.1)
Requirement already satisfied: pygments>=2.4.1 in
/usr/local/lib/python3.10/dist-packages (from nbconvert) (2.16.1)
Requirement already satisfied: tinycss2 in /usr/local/lib/python3.10/dist-
packages (from nbconvert) (1.3.0)
Requirement already satisfied: traitlets>=5.0 in /usr/local/lib/python3.10/dist-
packages (from nbconvert) (5.7.1)
Requirement already satisfied: platformdirs>=2.5 in
```