Colab notebook link:

https://colab.research.google.com/drive/1WOv-hZBcClKP7TtbD2BzhaOMdgV5Jk_F?usp=sharing

---

Insights:

Analyzing the dataset:

- There are total 14867 rows and 24 columns.
- Columns are: ['data', 'trip_creation_time', 'route_schedule_uuid', 'route_type', 'trip_uuid', 'source_center', 'source_name', 'destination_center', 'destination_name', 'od_start_time', 'od_end_time', 'start_scan_to_end_scan', 'is_cutoff', 'cutoff_factor', 'cutoff_timestamp', 'actual_distance_to_destination', 'actual_time','osrm_time', 'osrm_distance', 'factor', 'segment_actual_time', 'segment_osrm_time', 'segment_osrm_distance', 'segment_factor']
- Data type of columns: 1 integer type columns, 10 float type columns, 12 object type columns given in the dataset and 1 bool type column given in the dataset.

---

1. Basic data cleaning and exploration:

Removing Unknown field:

- 5 unknown columns are given in data set. So we are removing these columns from datset.
- 5 unknown fields are: ['is_cutoff', 'cutoff_factor', 'cutoff_timestamp', 'factor', 'segment_factor']

Conversion of Categorical Attributes to Category:

- Further analyzing the object type column, we can find out that 2 columns (‘data’ and ‘route type’) are category type columns. So, we are converting these two columns into category type columns.

Missing value & Duplicates:

- There are 2 columns ('source_name' and 'destination_name'), which has missing values.

- Further analyzing these two columns, we have seen that there are 10 unique source place name and 12 unique destination place name in not given in dataset.

- We finally replace these Null values of source name and destination name to source center and destination center respectively.

- There are no duplicates in the dataset.

Merging of rows and aggregation of fields:

- Since delivery details of one package divided into several rows, so we merge these rows according to their trip_uuid, source center and destination centre.

- Further, we carefully aggregate the other fields as per trip_uuid, source center and destination center.

---

2. Build some features to prepare the data for actual analysis:

Split and extract features from Source Name: City-place-code (State):

- Extracting State, city and place from source name and making separate column for source state, source city and source place.

- The source names, which are not given in the dataset, we mentioned it 'unknown' for further uses.

Split and extract features from Destination Name: City-place-code (State):

- Extracting State, city and place from destination name and making separate column for destination state, destination city and destination place.

- The destination names, which are not given in the dataset, we mentioned it 'unknown' for further uses.

Trip_creation_time: Extract features like day, month, year etc:

- Extracting the date, year, month and day from trip creation time column and making the separate column for date, year, month, day and hour.

Calculate the time taken between OD_ start_ time and OD_end_time:

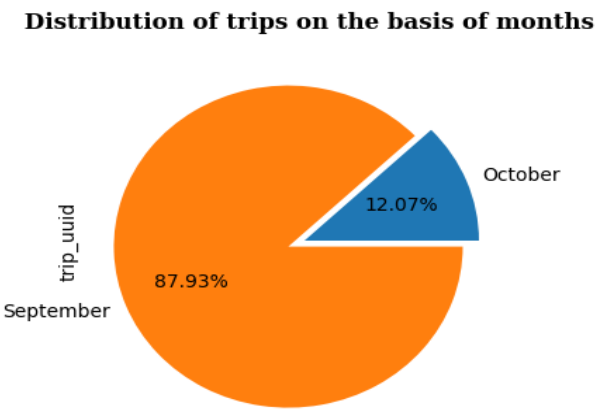
- Making the new column name OD_total_time, this is time difference between the OD_start_time and OD_end_time.

After cleaning, merging and features extraction, the new dataset is ready for analyzing.

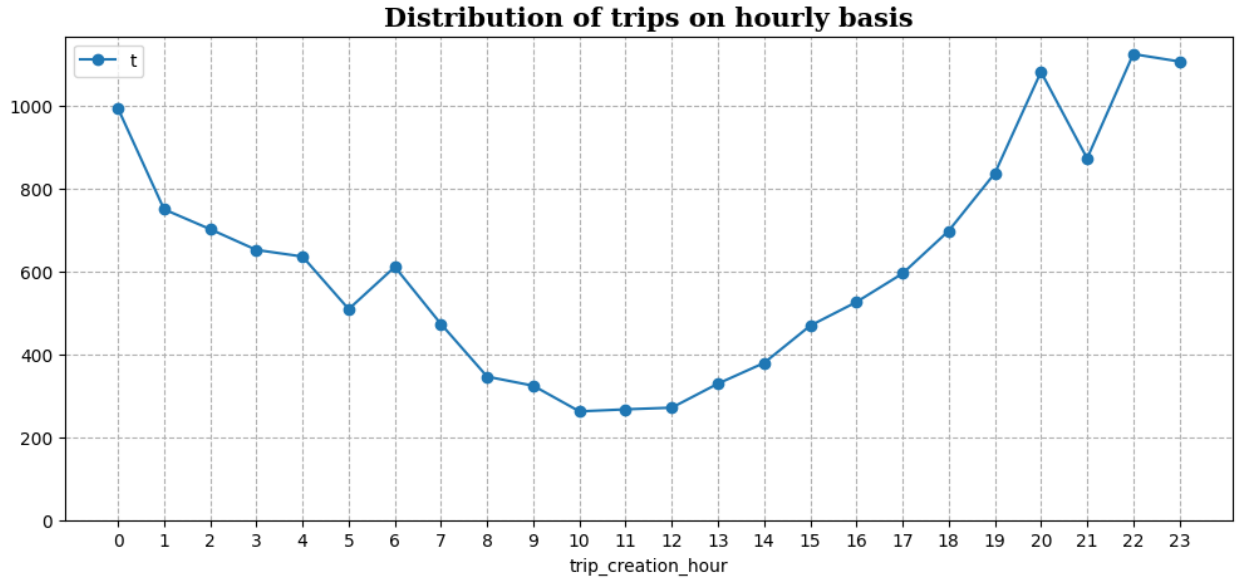
- Shape of new dataset is: 14817 rows and 28 columns
- The data is given from date: 2018-09-12 to date: 2018-10-03 and total 21 days data is given in the dataset.

Distribution of trips on the basis of month:

- There are only two months data is mentioned in dataset: September and October.
- As per dataset, 87.93% of trip was created in the month of September and rest was created in the month of October.

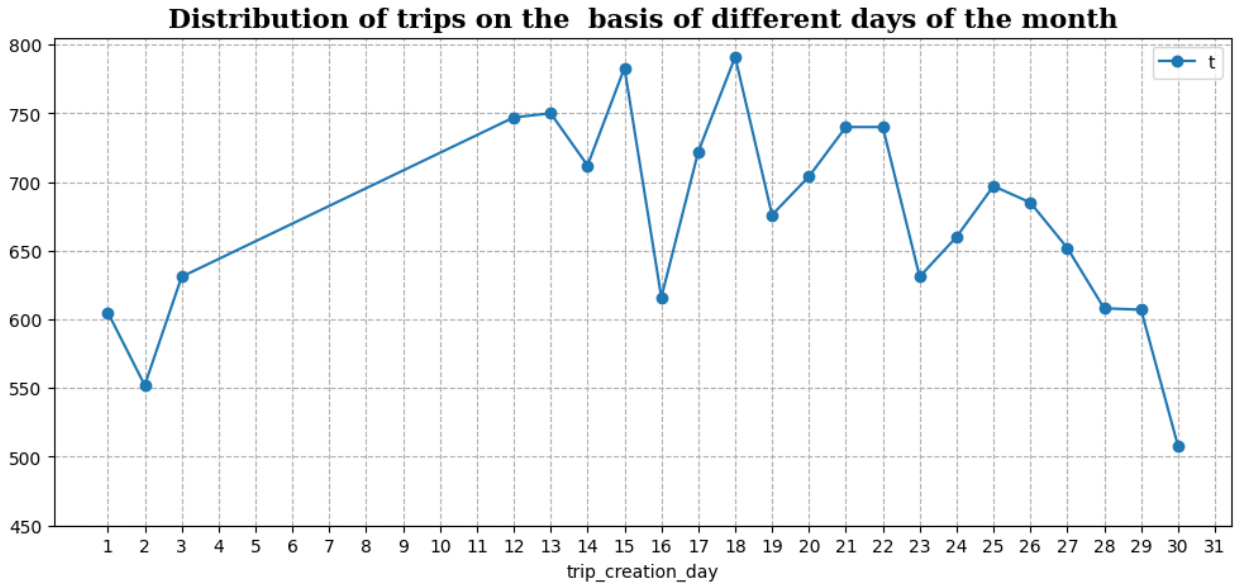


Distribution of trips creation on hourly basis:



- We can observed from the above chart that, the number of trips start increasing after 11 A.M, becomes maximum at 10 P.M and then start decreasing.

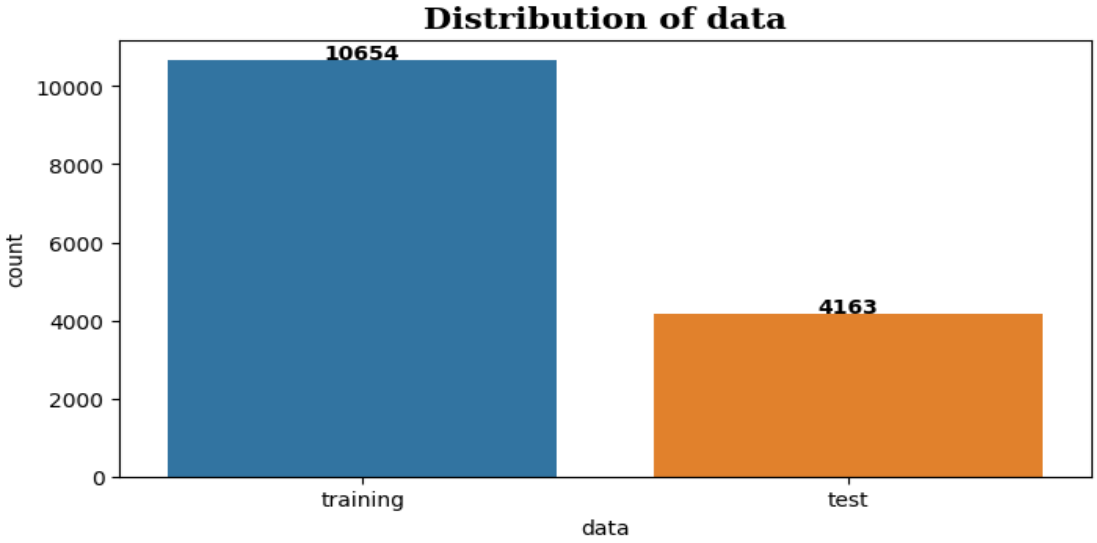
Distribution of trips creation on the basis of different days of the month:



- From the above chart, we can observed that maximum number of trips created in between 12 to 22 of the month, it means in the middle of the month maximum number of trips created.

Categorical Variables:

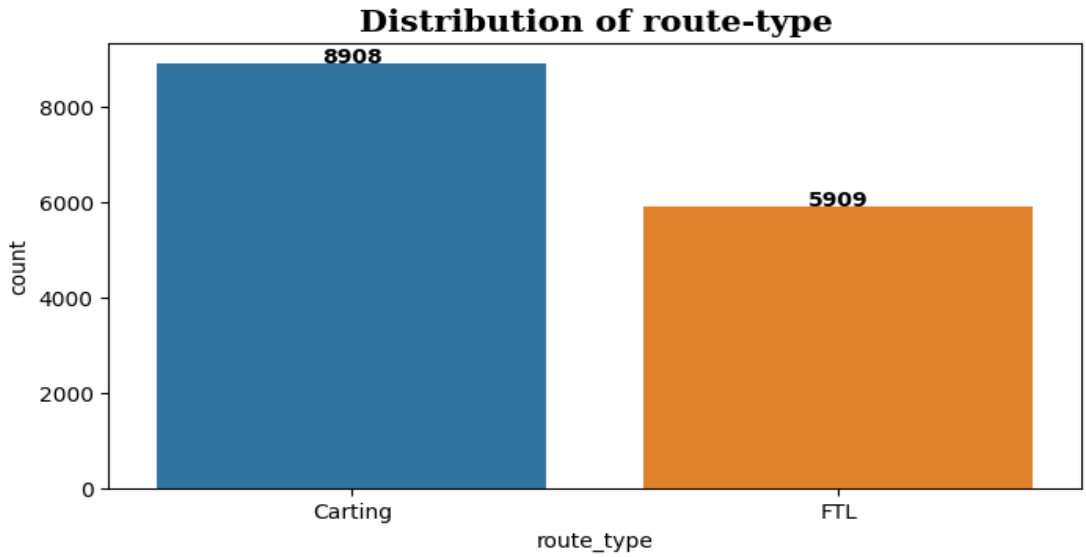
Distribution of data on the basis of trip:



- In the dataset, two types of data is used, one is testing data another is training data.

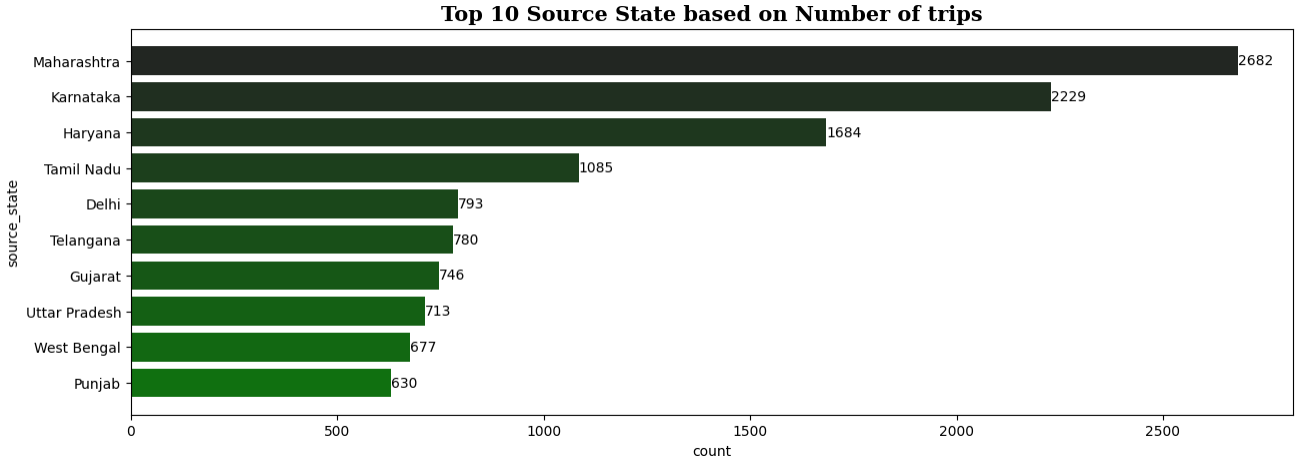
- Training data is nearly 2.5 times more than testing data.

Distribution of route-type on the basis of trip:

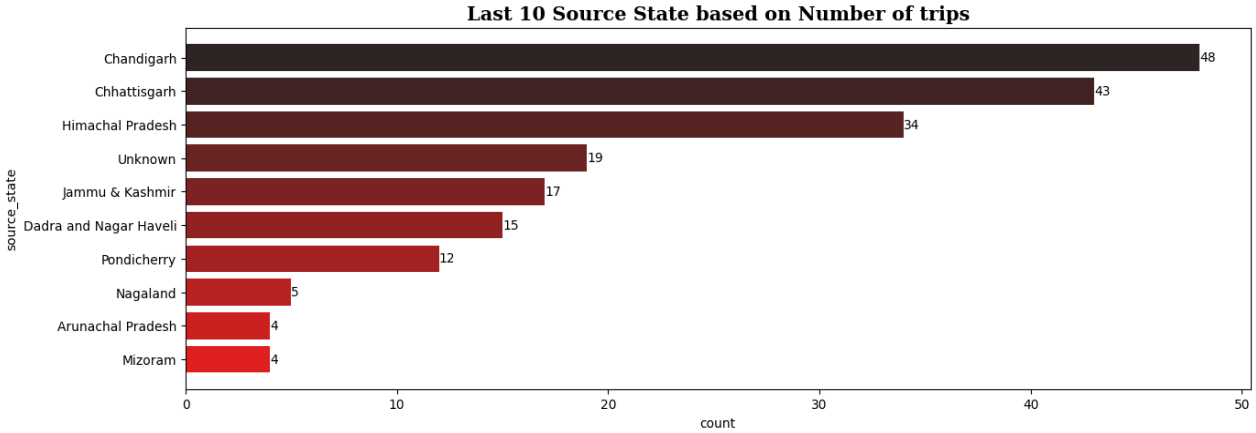


- In the dataset two types of route is used. One is Carting and another is FTL.
- Carting route was used nearly 1.5 times more than FTL route.

Distribution of trips on the basis of Source-state:

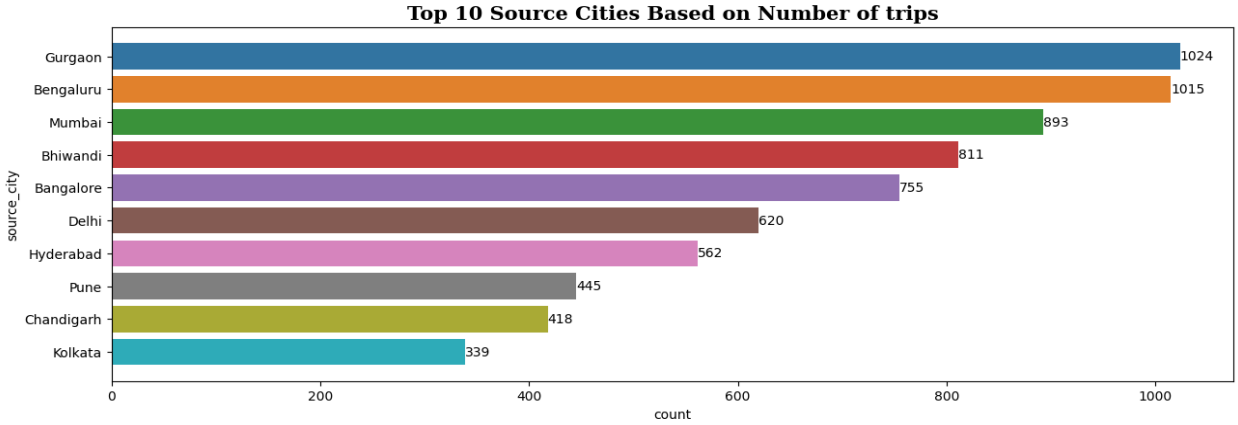


- From the above chart, we can see that the maximum trips originated from Maharashtra state followed by Karnataka and Haryana.



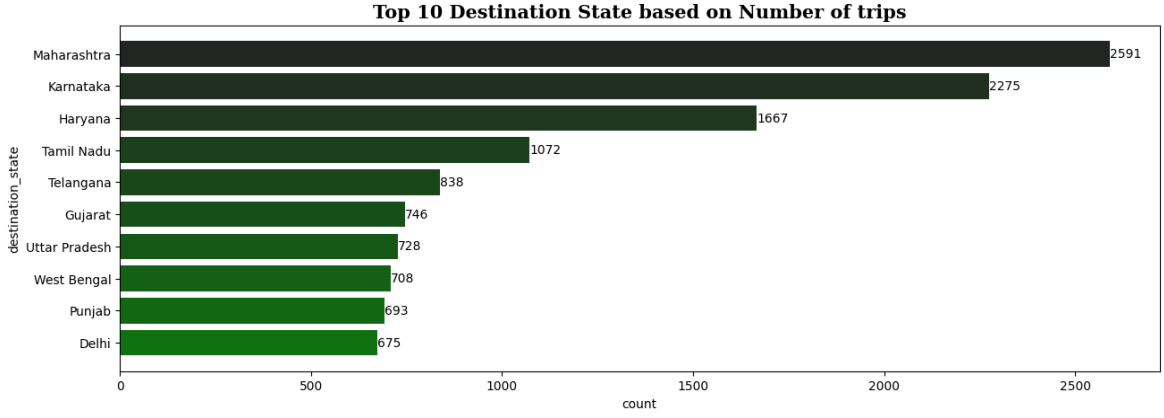
- As per dataset, there is three states i.e, Nagaland, Arunachal Pradesh and Mizoram, where the number of trips originates is less than 10.

Distribution of trips on the basis of Source-cities:



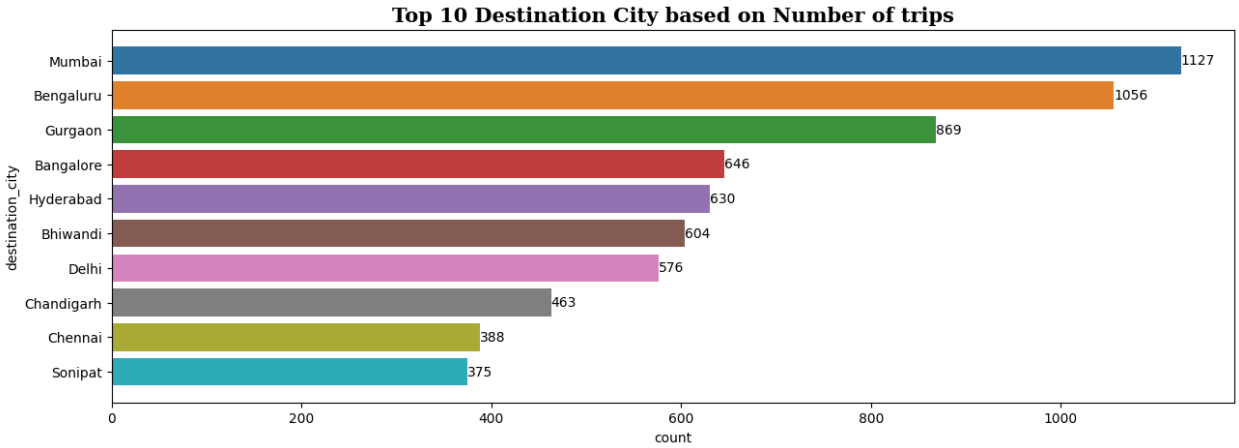
- From the above chart, we can analyze that maximum trips originated from Gurgaon city followed by Bengaluru and Mumbai.

Distribution of trips on the basis of Destination-states:



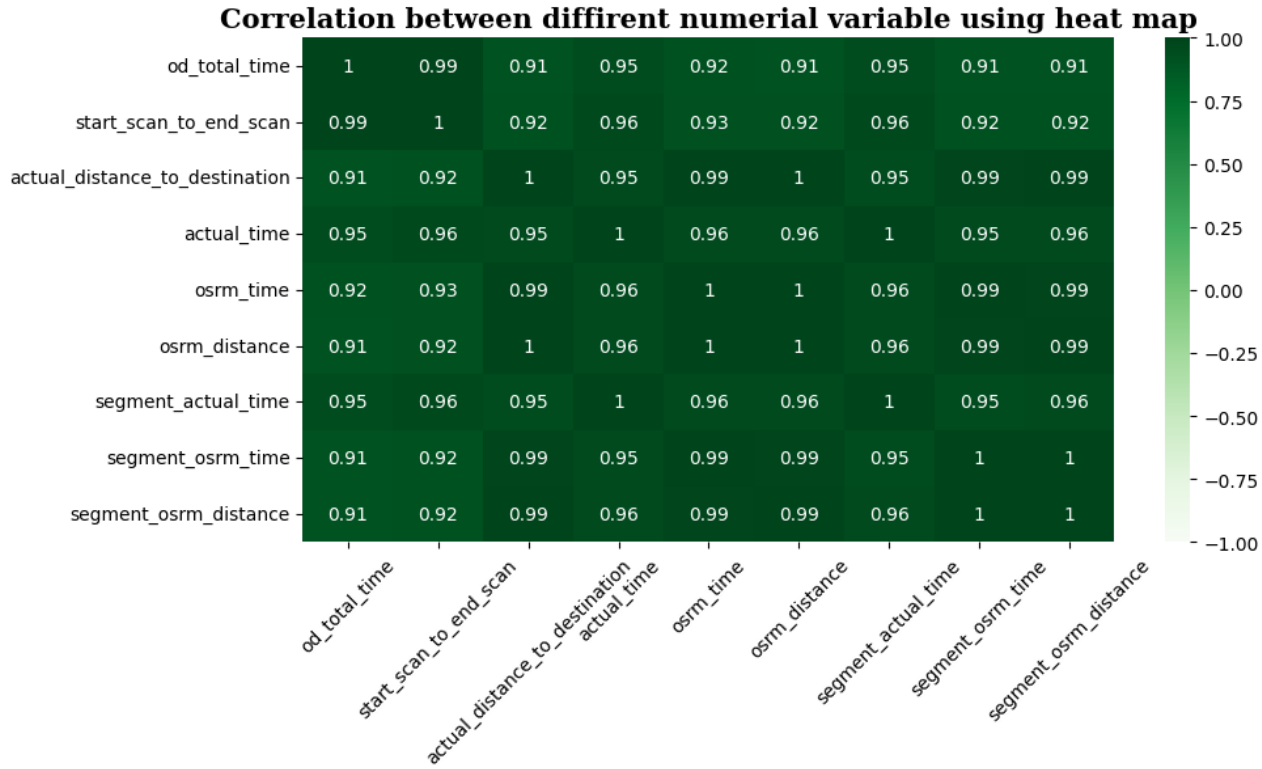
- From the above chart, we can see that the maximum trips originated from Maharashtra state followed by Karnataka and Haryana.

Distribution of trips on the basis of Destination-Cities:



- From the above chart, we can see that the maximum trips originated from Mumbai city followed by Bengaluru and Gurgaon city.

Numerical variables:



- Very strong co-relation (>0.9) exists between all the numerical variables specified above.

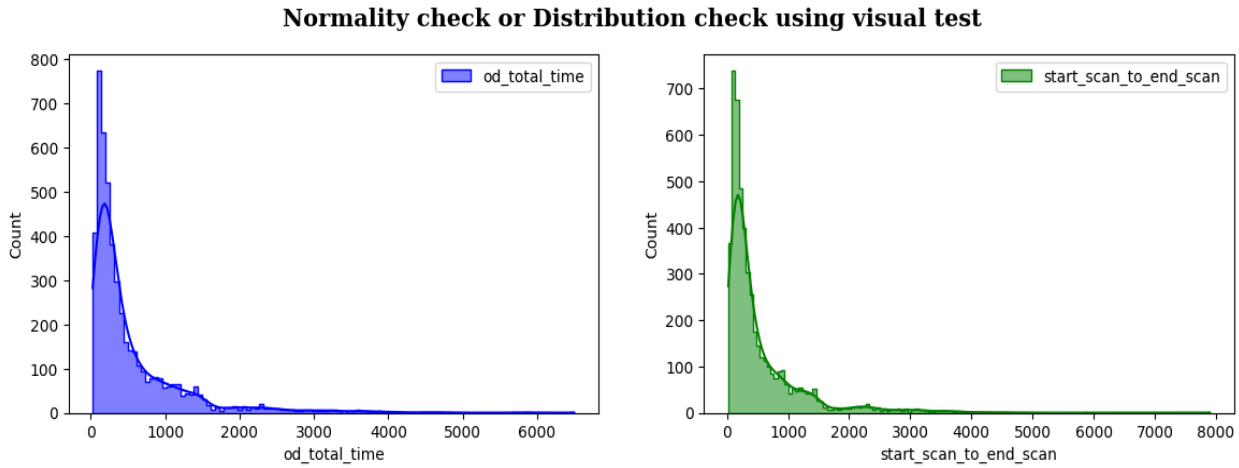
3. In-depth analysis and feature engineering:

Question: Compare the difference between od_total_time and start_scan_to_end_scan.
Hypothesis Testing:

Hypothesis Testing:

- **Step 1: Setup Null Hypothesis**
  - Null Hypothesis (Ho): od_total_time (Total Trip Time) and start_scan_to_end_scan (Expected total trip time) are same.
  - Alternative Hypothesis (Ha): od_total_time (Total Trip Time) and start_scan_to_end_scan (Expected total trip time) are not same.

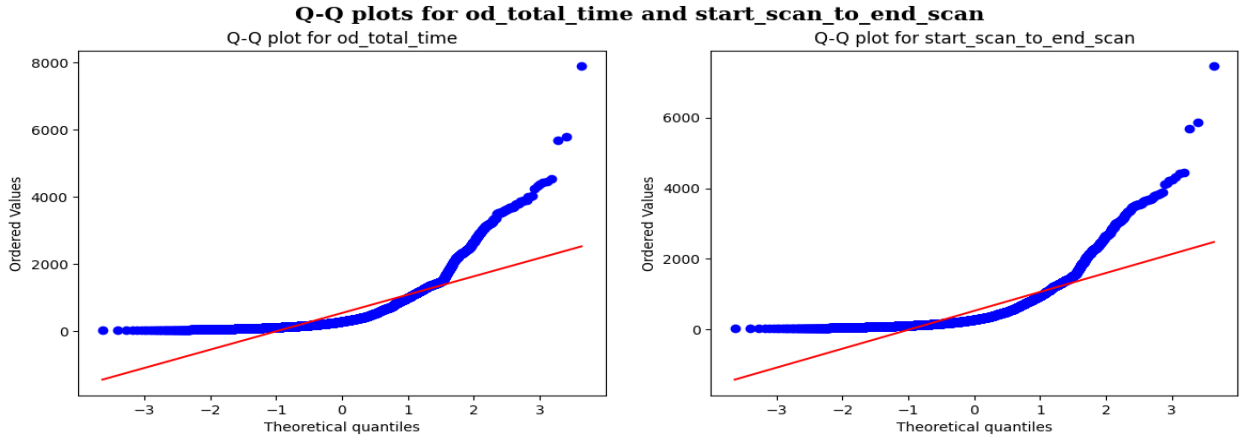
- **Step 2: Checking the assumption for the test.**



- *Normality or Distribution check using histogram or visual test:*
  - We can visualize from the above histogram plot, that the distribution doesn't follow normal distribution.
  - Both the distributions are nearly same and both are right skewed, but the distribution is not normal.

- *Normality or Distribution check using Q-Q plot test:*
  - We can figure it from the above Q-Q plot that the distribution doesn't follow normal distribution.

From the above plots we have seen that the samples do not come from normal distribution. Now we are applying another test Shapiro-wilk test for normality.



- *Normality check using Shapiro-wilk test:*

```
#Normality Check using Shapiro-Wilk test(for od_total_time)

# Ho: The sample follows normal distribution.
# Ha: The sample does not follow normal distribution.

alpha = 0.05
test_stat, p_value = shapiro(df_nw['od_total_time'].sample(5000))
print('p-value', p_value)
if p_value < alpha:
    print('Reject Ho. The sample does not follow normal distribution')
else:
    print('Fail to reject Ho. The sample follows normal distribution')
```

```
p-value 0.0
Reject Ho. The sample does not follow normal distribution
```

  - For od_total_time, the test result is: p-value 0.0, the sample does not follow normal distribution.

```
#Normality Check using Shapiro-Wilk test(for start_scan_to_end_scan)

# Ho: The sample follows normal distribution.
# Ha: The sample does not follow normal distribution.

alpha = 0.05
test_stat, p_value = shapiro(df_nw['start_scan_to_end_scan'].sample(5000))
print('p-value', p_value)
if p_value < alpha:
    print('Reject Ho. The sample does not follow normal distribution')
else:
    print('Fail to reject Ho. The sample follows normal distribution')
```

```
p-value 0.0
Reject Ho. The sample does not follow normal distribution
```

  - For start_scan_to_end_scan, the test result is: p-value 0.0, the sample does not follow normal distribution.

Even after applying Shapiro-wilk test, still we find out that the distribution of the "od_total_time" and "start_scan_to_end_scan" data, the samples do not follow normal distribution.

Now, transforming the data using boxcox transformation to check if the transformed data follows normal distribution.

- *Normality Check or Distribution Check after boxcox transformation:*

```
#Normality Check using Shapiro-Wilk test after Boxcox transformation (for od_total_time)

transformed_od_total_time = spy.boxcox(df_nw['od_total_time'])[0]

# Ho: The sample follows normal distribution.
# Ha: The sample does not follow normal distribution.

alpha = 0.05
test_stat, p_value = shapiro(transformed_od_total_time)
print('p-value', p_value)
if p_value < alpha:
    print('Reject Ho. The sample does not follow normal distribution')
else:
    print('Fail to reject Ho. The sample follows normal distribution')
```

```
p-value 6.041245108788226e-27
Reject Ho. The sample does not follow normal distribution
```

  - For od_total_time, the test result is: p-value 6.041245108788226e-27, the sample does not follow normal distribution.

```
[59] #Normality Check using Shapiro-Wilk test after Boxcox transformation (for start_scan_to_end_scan)

transformed_start_scan_to_end_scan = spy.boxcox(df_nw['start_scan_to_end_scan'])[0]

# Ho: The sample follows normal distribution.
# Ha: The sample does not follow normal distribution.

alpha = 0.05
test_stat, p_value = shapiro(transformed_start_scan_to_end_scan)
print('p-value', p_value)
if p_value < alpha:
    print('Reject Ho. The sample does not follow normal distribution')
else:
    print('Fail to reject Ho. The sample follows normal distribution')
```

```
p-value 1.0471322892609475e-24
Reject Ho. The sample does not follow normal distribution
```

  - For start_scan_to_end_scan, the test result is: p-value 1.0471322892609475e-24, the sample does not follow normal distribution.

Even after applying Boxcox transformation, we can see that the distributions for both “od_total_time” and “start_scan_to_end_scan” columns, data doesn’t follow normal distribution.

- *Variance Check using Levene's test:*

```
# Ho - Varience is Equal. Homogenous Variance
# Ha - Varience is Not Equal. Non Homogenous Variance
od_total_time_sample =df_nw['od_total_time'].sample(5000)
start_scan_to_end_scan_sample = df_nw['start_scan_to_end_scan'].sample(5000)

alpha = 0.05
test_stat, p_value = levene(od_total_time_sample, start_scan_to_end_scan_sample)
print('p-value', p_value)
if p_value < alpha:
    print('reject Ho: The samples do not have  Homogenous Variance')
else:
    print('Fail to Reject Ho: The samples have Homogenous Variance ')
```

```
p-value 0.937848145636102
Fail to Reject Ho: The samples have Homogenous Variance
```

  - The test result is: p-value 0.937848145636102, Fail to Reject Ho: The samples have Homogenous Variance.

➢ **Step 3: Set a significance level (alpha).**

- We set our alpha to be 0.05.

➢ **Step 4: Calculate test statistics.**

- Standard deviation of the population is not known. So, T-test is right choice for checking the statistics.
- But**,** we have seen in previously (using histogram plot, Q-Q plot, Shapiro-wilk test) that the distribution is not normal. And in variance test (using Levene's test), we have seen that the variance is homogeneous.
- Since the samples are not normally distributed. So, T-test is couldn't give us proper statistics result, it probably increase the risk of errors.
- We can perform non-parametric test. i.e; ks- test, ks-test doesn't depend on the distribution.

```
#ks-test
ks_stat,p_value = kstest(df_nw.od_total_time, df_nw.start_scan_to_end_scan)

print('ks test statistic result is:', ks_stat)
print('P value is:', p_value)
```

```
ks test statistic result is: 0.01626510089761768
P value is: 0.039264199380179554
```

- ***ks-Test:***
  - The result of ks-test: ks test statistic result is: 0.01626510089761768, P value is: 0.039264199380179554.

- **Step 5: Decision to accept or reject null hypothesis.**
  - Based on P value, we accept the null hypothesis.
    - If P value < significance level (alpha) then reject null hypothesis.
    - If P value > significance level (alpha) then accept null hypothesis.

```
[75] # Null Hypothesis (Ho): od_total_time (Total Trip Time) and start_scan_to_end_scan (Expected total trip time) are same.
     # Alternative Hypothesis (Ha): od_total_time (Total Trip Time) and start_scan_to_end_scan (Expected total trip time) are not same.



     alpha = 0.05
     if p_value < alpha:
       print('Reject Ho: od_total_time (Total Trip Time) and start_scan_to_end_scan (Expected total trip time) are not same.')
     else:
       print('Accept Ho: od_total_time (Total Trip Time) and start_scan_to_end_scan (Expected total trip time) are same.')

     Reject Ho: od_total_time (Total Trip Time) and start_scan_to_end_scan (Expected total trip time) are not same.
```

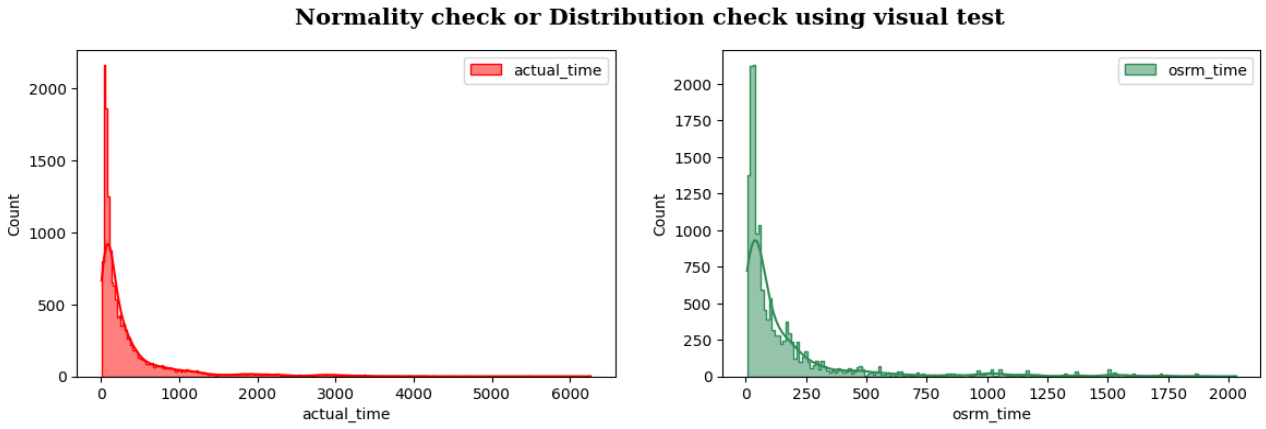
- **Step 6: Inference from the analysis.**
  - Finally, we came to the conclusion from the above ks test that od_total_time and start_scan_to_end_scan are not same.

Question: Hypothesis testing/ visual analysis between actual_time aggregated value and OSRM time aggregated value

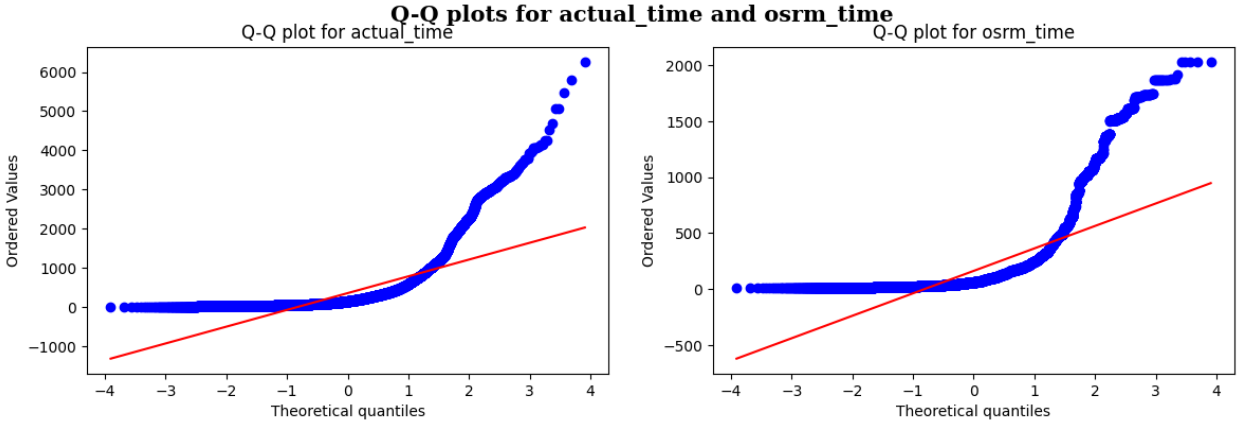
Hypothesis Testing:

- **Step 1: Setup Null Hypothesis**
  - Null Hypothesis (Ho): actual_time aggregated value and osrm_time aggregated value are same.
  - Alternative Hypothesis (Ha): actual_time aggregated value and osrm_time aggregated value are not same.

- **Step 2: Checking the assumption for the test.**



- *Normality or Distribution check using histogram or visual test:*
    - We can visualize from the above histogram plot, that the distribution doesn't follow normal distribution.
    - Both the distributions are nearly same and both are right skewed, but the distribution is not normal.



- *Normality or Distribution check using Q-Q plot test:*
    - We can figure it from the above Q-Q plot that the distribution doesn't follow normal distribution.

From the above plots we have seen that the samples do not come from normal distribution. Now we are applying another test Shapiro-wilk test for normality.

- *Normality check using Shapiro-wilk test:*

```
[94] #Normality Check using Shapiro-Wilk test(for actual_time)

     # Ho: The sample follows normal distribution.
     # Ha: The sample does not follow normal distribution.

     alpha = 0.05
     test_stat, p_value = shapiro(df_nw['actual_time'].sample(5000))
     print('p-value', p_value)
     if p_value < alpha:
         print('Reject Ho. The sample does not follow normal distribution')
     else:
         print('Fail to reject Ho. The sample follows normal distribution')

     p-value 0.0
     Reject Ho. The sample does not follow normal distribution
```

    - For actual_time, the test result is: p-value 0.0, the sample does not follow normal distribution.

```
[95] #Normality Check using Shapiro-Wilk test(for osrm_time)

     # Ho: The sample follows normal distribution.
     # Ha: The sample does not follow normal distribution.

     alpha = 0.05
     test_stat, p_value = shapiro(df_nw['osrm_time'].sample(5000))
     print('p-value', p_value)
     if p_value < alpha:
         print('Reject Ho. The sample does not follow normal distribution')
     else:
         print('Fail to reject Ho. The sample follows normal distribution')

     p-value 0.0
     Reject Ho. The sample does not follow normal distribution
```

- For osrm_time, the test result is: p-value 0.0, the sample does not follow normal distribution.

Even after applying Shapiro-wilk test, still we find out that the distribution of the "actual_time" and "osrm_time" data, the samples do not follow normal distribution.

Now, transforming the data using boxcox transformation to check if the transformed data follows normal distribution.

- *Normality Check or Distribution Check after boxcox transformation:*

```
[68] #Normality Check using Shapiro-Wilk test after Boxcox transformation (for actual_time)

     transformed_actual_time = spy.boxcox(df_nw['actual_time'])[0]

     # Ho: The sample follows normal distribution.
     # Ha: The sample does not follow normal distribution.

     alpha = 0.05
     test_stat, p_value = shapiro(transformed_actual_time)
     print('p-value', p_value)
     if p_value < alpha:
         print('Reject Ho. The sample does not follow normal distribution')
     else:
         print('Fail to reject Ho. The sample follows normal distribution')

     p-value 1.020620453603145e-28
     Reject Ho. The sample does not follow normal distribution
```

- For actual_time, the test result is: p-value 1.020620453603145e-28, the sample does not follow normal distribution.

```
[69] #Normality Check using Shapiro-Wilk test after Boxcox transformation (for osrm_time)

     transformed_osrm_time = spy.boxcox(df_nw['osrm_time'])[0]

     # Ho: The sample follows normal distribution.
     # Ha: The sample does not follow normal distribution.

     alpha = 0.05
     test_stat, p_value = shapiro(transformed_osrm_time)
     print('p-value', p_value)
     if p_value < alpha:
         print('Reject Ho. The sample does not follow normal distribution')
     else:
         print('Fail to reject Ho. The sample follows normal distribution')
```

```
p-value 3.5882550510138333e-35
Reject Ho. The sample does not follow normal distribution
```

- For osrm_time, the test result is: p-value 3.5882550510138333e-35, the sample does not follow normal distribution.

Even after applying Boxcox transformation, we can see that the distributions for both "actual_time" and "osrm_time" columns, data doesn't follow normal distribution.

- *Variance Check using Levene's test:*

```
# Ho - Varience is Equal. Homogenous Variance
# Ha - Varience is Not Equal. Non Homogenous Variance
actual_time_sample =df_nw['actual_time'].sample(5000)
osrm_time_sample = df_nw['osrm_time'].sample(5000)

alpha = 0.05
test_stat, p_value = levene(actual_time_sample, osrm_time_sample)
print('p-value', p_value)
if p_value < alpha:
    print('reject Ho: The samples do not have  Homogenous Variance')
else:
    print('Fail to Reject Ho: The samples have Homogenous Variance ')
```

```
p-value 6.479139573428664e-69
reject Ho: The samples do not have  Homogenous Variance
```

- The test result is: p-value 6.479139573428664e-69, Reject Ho: The samples do not have Homogenous Variance.

➢ **Step 3: Set a significance level (alpha).**
- We set our alpha to be 0.05.

➢ **Step 4: Calculate test statistics.**
- Standard deviation of the population is not known. So, T-test is right choice for checking the statistics.

- But, we have seen in previously (using histogram plot, Q-Q plot, Shapiro-wilk test) that the distribution is not normal. And in variance test (using Levene's test), we have seen that the variance is non homogeneous.
- Since the samples are not normally distributed. So, T-test is couldn't give us proper statistics result, it probably increase the risk of errors.
- We can perform non-parametric test. i.e; ks- test, ks-test doesn't depend on the distribution.

```
#ks-test
ks_stat,p_value = kstest(df_nw['actual_time'], df_nw['osrm_time'])

print('ks test statistic result is:', ks_stat)
print('P value is:', p_value)
```

```
ks test statistic result is: 0.2973611392319633
P value is: 0.0
```

- ***ks-Test:***
  - The result of ks-test: ks test statistic result is: 0.2973611392319633,
    P value is: 0.0

➢ **Step 5: Decision to accept or reject null hypothesis.**

- Based on P value, we accept the null hypothesis.
  - If P value < significance level (alpha) then reject null hypothesis.
  - If P value > significance level (alpha) then accept null hypothesis.

```
[72] # Null Hypothesis (Ho): actual_time aggregated value and osrm_time aggregated value are same.
     # Alternative Hypothesis (Ha): actual_time aggregated value and osrm_time aggregated value are not same.

     alpha = 0.05
     if p_value < alpha:
       print('Reject Ho: actual_time aggregated value and osrm_time aggregated value are not same')
     else:
       print('Accept Ho: actual_time aggregated value and osrm_time aggregated value are same.')
```

```
Reject Ho: actual_time aggregated value and osrm_time aggregated value are not same
```

➢ **Step 6: Inference from the analysis.**

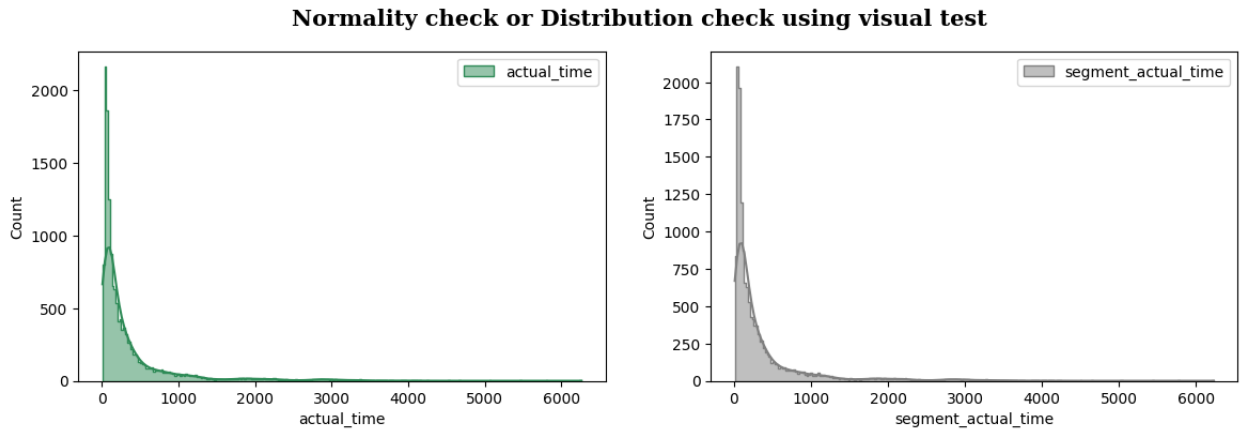
- Finally, we came to the conclusion from the above ks test that actual_time and osrm_time are not same.

Question: Hypothesis testing/ visual analysis between actual_time aggregated value and segment_actual_time aggregated value

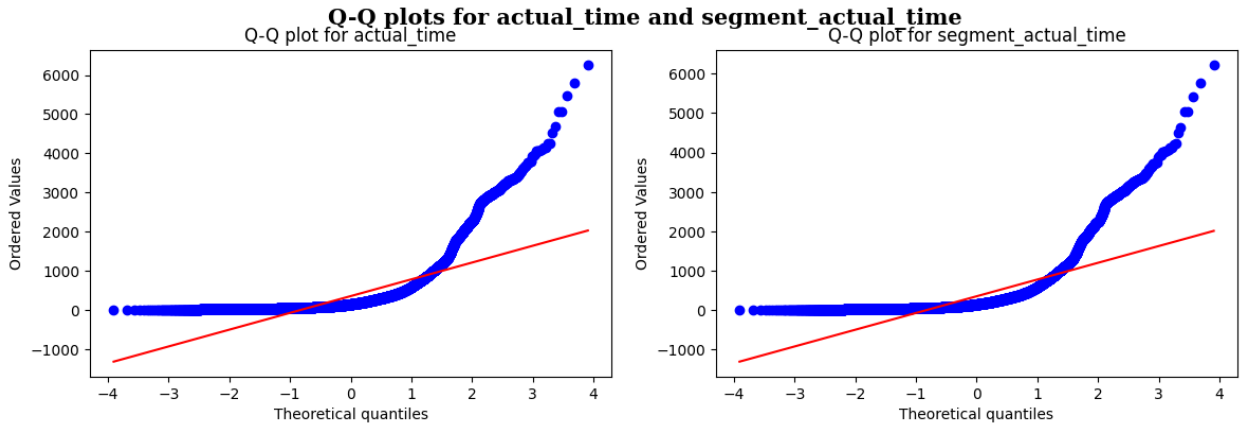
Hypothesis Testing:

- **Step 1: Setup Null Hypothesis**
  - Null Hypothesis (Ho): actual_time aggregated value and segment_actual_time aggregated value are same.
  - Alternative Hypothesis (Ha): actual_time aggregated value and segment_actual_time aggregated value are not same.

- **Step 2: Checking the assumption for the test.**



- *Normality or Distribution check using histogram or visual test:*
  - We can visualize from the above histogram plot, that the distribution doesn't follow normal distribution.
  - Both the distributions are nearly same and both are right skewed, but the distribution is not normal.



- *Normality or Distribution check using Q-Q plot test:*

- We can figure it from the above Q-Q plot that the distribution doesn't follow normal distribution.

From the above plots we have seen that the samples do not come from normal distribution. Now we are applying another test Shapiro-wilk test for normality.

- *Normality check using Shapiro-wilk test:*

```
[89] #Normality Check using Shapiro-Wilk test(for actual_time)

     # Ho: The sample follows normal distribution.
     # Ha: The sample does not follow normal distribution.

     alpha = 0.05
     test_stat, p_value = shapiro(df_nw['actual_time'].sample(5000))
     print('p-value', p_value)
     if p_value < alpha:
         print('Reject Ho. The sample does not follow normal distribution')
     else:
         print('Fail to reject Ho. The sample follows normal distribution')
```

```
p-value 0.0
Reject Ho. The sample does not follow normal distribution
```

- For actual_time, the test result is: p-value 0.0, the sample does not follow normal distribution.

```
[90] #Normality Check using Shapiro-Wilk test(for segment_actual_time)

     # Ho: The sample follows normal distribution.
     # Ha: The sample does not follow normal distribution.

     alpha = 0.05
     test_stat, p_value = shapiro(df_nw['segment_actual_time'].sample(5000))
     print('p-value', p_value)
     if p_value < alpha:
         print('Reject Ho. The sample does not follow normal distribution')
     else:
         print('Fail to reject Ho. The sample follows normal distribution')
```

```
p-value 0.0
Reject Ho. The sample does not follow normal distribution
```

- For segment_actual_time, the test result is: p-value 0.0, the sample does not follow normal distribution.

Even after applying Shapiro-wilk test, still we find out that the distribution of the "actual_time" and "segment_actual_time" data, the samples do not follow normal distribution.

Now, transforming the data using boxcox transformation to check if the transformed data follows normal distribution.

- *Normality Check or Distribution Check after boxcox transformation:*

```
[92] #Normality Check using Shapiro-Wilk test after Boxcox transformation (for actual_time)

     transformed_actual_time = spy.boxcox(df_nw['actual_time'])[0]

     # Ho: The sample follows normal distribution.
     # Ha: The sample does not follow normal distribution.

     alpha = 0.05
     test_stat, p_value = shapiro(transformed_actual_time)
     print('p-value', p_value)
     if p_value < alpha:
         print('Reject Ho. The sample does not follow normal distribution')
     else:
         print('Fail to reject Ho. The sample follows normal distribution')
```

```
p-value 1.020620453603145e-28
Reject Ho. The sample does not follow normal distribution
```

- For actual_time, the test result is: p-value 1.020620453603145e-28, the sample does not follow normal distribution.

```
[93] #Normality Check using Shapiro-Wilk test after Boxcox transformation (for segment_actual_time)

     transformed_segment_actual_time = spy.boxcox(df_nw['segment_actual_time'])[0]

     # Ho: The sample follows normal distribution.
     # Ha: The sample does not follow normal distribution.

     alpha = 0.05
     test_stat, p_value = shapiro(transformed_segment_actual_time)
     print('p-value', p_value)
     if p_value < alpha:
         print('Reject Ho. The sample does not follow normal distribution')
     else:
         print('Fail to reject Ho. The sample follows normal distribution')
```

```
p-value 5.700074948787037e-29
Reject Ho. The sample does not follow normal distribution
```

- For segment_actual_time, the test result is: p-value 5.700074948787037e-29, the sample does not follow normal distribution.

Even after applying Boxcox transformation, we can see that the distributions for both “actual_time” and “segment_actual_time” columns, data doesn’t follow normal distribution.

- *Variance Check using Levene's test:*

```
# Ho - Varience is Equal. Homogenous Variance
# Ha - Varience is Not Equal. Non Homogenous Variance

alpha = 0.05
test_stat, p_value = levene(df_nw['actual_time'], df_nw['segment_actual_time'])
print('p-value', p_value)
if p_value < alpha:
    print('Reject Ho: The samples do not have  Homogenous Variance')
else:
    print('Fail to Reject Ho: The samples have Homogenous Variance ')
```

```
p-value 0.695502241317651
Fail to Reject Ho: The samples have Homogenous Variance
```

- The test result is: p-value 0.695502241317651, Fail to Reject Ho: The samples have Homogenous Variance.

➢ **Step 3: Set a significance level (alpha).**

- We set our alpha to be 0.05.

➢ **Step 4: Calculate test statistics.**

- Standard deviation of the population is not known. So, T-test is right choice for checking the statistics.
- But**,** we have seen in previously (using histogram plot, Q-Q plot, Shapiro-wilk test) that the distribution is not normal. And in variance test (using Levene's test), we have seen that the variance is homogeneous.
- Since the samples are not normally distributed. So, T-test is couldn't give us proper statistics result, it probably increase the risk of errors.
- We can perform non-parametric test. i.e; ks- test, ks-test doesn't depend on the distribution.

```
[82] #ks-test
     ks_stat,p_value = kstest(df_nw['actual_time'], df_nw['segment_actual_time'])

     print('ks test statistic result is:', ks_stat)
     print('P value is:', p_value)
```

```
ks test statistic result is: 0.006344064250523029
P value is: 0.9248583909392553
```

- ***ks-Test:***
  - The result of ks-test: ks test statistic result is: 0.006344064250523029, P value is: 0.9248583909392553

➢ **Step 5: Decision to accept or reject null hypothesis.**

- Based on P value, we accept the null hypothesis.
  - If P value < significance level (alpha) then reject null hypothesis.
  - If P value > significance level (alpha) then accept null hypothesis.

```
[84] # Null Hypothesis (Ho): actual_time aggregated value and segment_actual_time aggregated value are same.
     # Alternative Hypothesis (Ha): actual_time aggregated value and segment_actual_time aggregated value are not same.

     alpha = 0.05
     if p_value < alpha:
       print('Reject Ho: actual_time aggregated value and segment_actual_time aggregated value are not same')
     else:
       print('Accept Ho: actual_time aggregated value and segment_actual_time aggregated value are same.')

     Accept Ho: actual_time aggregated value and segment_actual_time aggregated value are same.
```

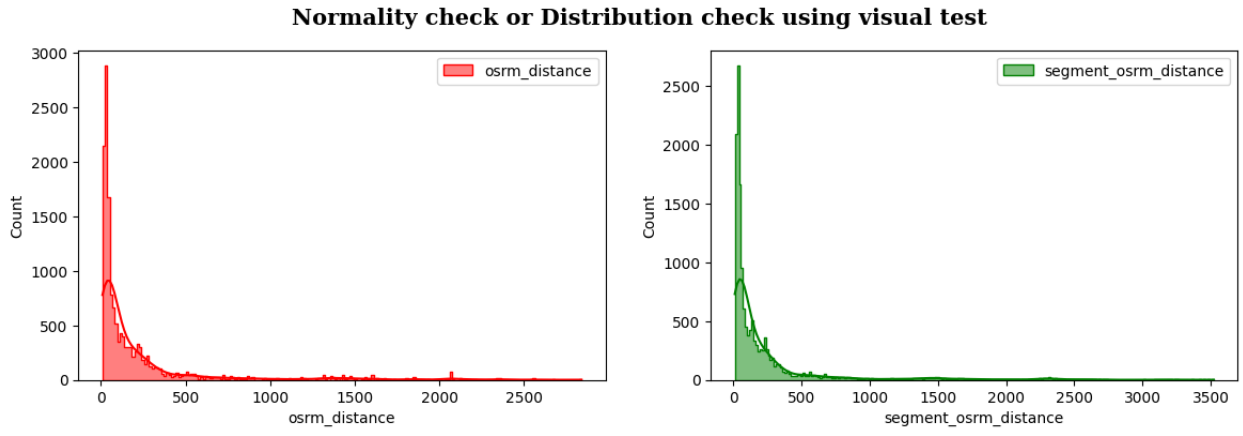
- **Step 6: Inference from the analysis.**
  - Finally, we came to the conclusion from the above ks test that actual_time aggregated value and segment_actual_time aggregated value are same.

Question: Hypothesis testing/ visual analysis between osrm_distance aggregated value and segment_osrm_distance aggregated value

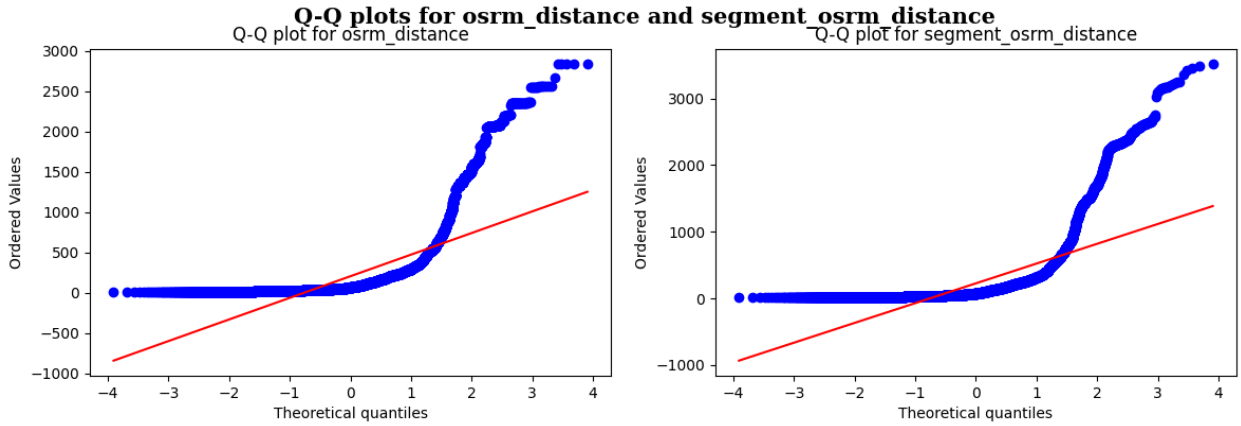
Hypothesis Testing:

- **Step 1: Setup Null Hypothesis**
  - Null Hypothesis (Ho): osrm_distance aggregated value and segment_osrm_distance aggregated value are same.
  - Alternative Hypothesis (Ha): osrm_distance aggregated value and segment_osrm_distance aggregated value are not same.

- **Step 2: Checking the assumption for the test.**



- *Normality or Distribution check using histogram or visual test:*
    - We can visualize from the above histogram plot, that the distribution doesn't follow normal distribution.
    - Both the distributions are nearly same and both are right skewed, but the distribution is not normal.



- *Normality or Distribution check using Q-Q plot test:*
    - We can figure it from the above Q-Q plot that the distribution doesn't follow normal distribution.

From the above plots we have seen that the samples do not come from normal distribution. Now we are applying another test Shapiro-wilk test for normality.

- *Normality check using Shapiro-wilk test:*

```
[90] #Normality Check using Shapiro-Wilk test(for osrm_distance)

     # Ho: The sample follows normal distribution.
     # Ha: The sample does not follow normal distribution.

     alpha = 0.05
     test_stat, p_value = shapiro(df_nw['osrm_distance'].sample(5000))
     print('p-value', p_value)
     if p_value < alpha:
         print('Reject Ho. The sample does not follow normal distribution')
     else:
         print('Fail to reject Ho. The sample follows normal distribution')

     p-value 0.0
     Reject Ho. The sample does not follow normal distribution
```

    - For osrm_distance, the test result is: p-value 0.0, the sample does not follow normal distribution

```
#Normality Check using Shapiro-Wilk test(for segment_osrm_distance)

# Ho: The sample follows normal distribution.
# Ha: The sample does not follow normal distribution.

alpha = 0.05
test_stat, p_value = shapiro(df_nw['segment_osrm_distance'].sample(5000))
print('p-value', p_value)
if p_value < alpha:
    print('Reject Ho. The sample does not follow normal distribution')
else:
    print('Fail to reject Ho. The sample follows normal distribution')
```

```
p-value 0.0
Reject Ho. The sample does not follow normal distribution
```

- For segment_osrm_distance, the test result is: p-value 0.0, the sample does not follow normal distribution.

Even after applying Shapiro-wilk test, still we find out that the distribution of the "osrm_distance" and "segment_osrm_distance" data, the samples do not follow normal distribution.

Now, transforming the data using boxcox transformation to check if the transformed data follows normal distribution.

- *Normality Check or Distribution Check after boxcox transformation:*

```
#Normality Check using Shapiro-Wilk test after Boxcox transformation (for osrm_distance)

transformed_osrm_distance = spy.boxcox(df_nw['osrm_distance'])[0]

# Ho: The sample follows normal distribution.
# Ha: The sample does not follow normal distribution.

alpha = 0.05
test_stat, p_value = shapiro(transformed_osrm_distance)
print('p-value', p_value)
if p_value < alpha:
    print('Reject Ho. The sample does not follow normal distribution')
else:
    print('Fail to reject Ho. The sample follows normal distribution')
```

```
p-value 7.061423221425618e-41
Reject Ho. The sample does not follow normal distribution
```

- For osrm_distance, the test result is: p-value 7.061423221425618e-41, the sample does not follow normal distribution

```
[95] #Normality Check using Shapiro-Wilk test after Boxcox transformation (for osrm_distance)

     transformed_segment_osrm_distance = spy.boxcox(df_nw['segment_osrm_distance'])[0]

     # Ho: The sample follows normal distribution.
     # Ha: The sample does not follow normal distribution.

     alpha = 0.05
     test_stat, p_value = shapiro(transformed_segment_osrm_distance)
     print('p-value', p_value)
     if p_value < alpha:
         print('Reject Ho. The sample does not follow normal distribution')
     else:
         print('Fail to reject Ho. The sample follows normal distribution')
```

```
p-value 3.049169406432229e-38
Reject Ho. The sample does not follow normal distribution
```

- For segment_osrm_distance, the test result is: p-value 3.049169406432229e-38, the sample does not follow normal distribution.

Even after applying Boxcox transformation, we can see that the distributions for both "osrm_distance" and "segment_osrm_distance" columns, data doesn't follow normal distribution.

- *Variance Check using Levene's test:*

```
# Ho - Varience is Equal. Homogenous Variance
# Ha - Varience is Not Equal. Non Homogenous Variance

alpha = 0.05
test_stat, p_value = levene(df_nw['osrm_distance'], df_nw['segment_osrm_distance'])
print('p-value', p_value)
if p_value < alpha:
    print('Reject Ho: The samples do not have  Homogenous Variance')
else:
    print('Fail to Reject Ho: The samples have Homogenous Variance ')
```

```
p-value 0.00020976006524780905
Reject Ho: The samples do not have  Homogenous Variance
```

- The test result is: p-value 0.00020976006524780905, Reject Ho: The samples do not have Homogenous Variance.

➢ **Step 3: Set a significance level (alpha).**
- We set our alpha to be 0.05.

➢ **Step 4: Calculate test statistics.**
- Standard deviation of the population is not known. So, T-test is right choice for checking the statistics.

- But, we have seen in previously (using histogram plot, Q-Q plot, Shapiro-wilk test) that the distribution is not normal. And in variance test (using Levene's test), we have seen that the variance is non homogeneous.
- Since the samples are not normally distributed. So, T-test is couldn't give us proper statistics result, it probably increase the risk of errors.
- We can perform non-parametric test. i.e; ks- test, ks-test doesn't depend on the distribution.

```
[97] #ks-test
     ks_stat,p_value = kstest(df_nw['osrm_distance'], df_nw['segment_osrm_distance'])

     print('ks test statistic result is:', ks_stat)
     print('P value is:', p_value)

     ks test statistic result is: 0.0416413578997098
     P value is: 1.3413627761631081e-11
```

- ***ks-Test:***
  - The result of ks-test: ks test statistic result is: 0.0416413578997098, P value is: 1.3413627761631081e-11

➢ **Step 5: Decision to accept or reject null hypothesis.**

- Based on P value, we accept the null hypothesis.
  - If P value < significance level (alpha) then reject null hypothesis.
  - If P value > significance level (alpha) then accept null hypothesis.

```
[98] # Null Hypothesis (Ho): osrm_distance aggregated value and segment_osrm_distance aggregated value are same.
     # Alternative Hypothesis (Ha): osrm_distance aggregated value and segment_osrm_distance aggregated value are not same.

     alpha = 0.05
     if p_value < alpha:
       print('Reject Ho: osrm_distance aggregated value and segment_osrm_distance aggregated value are not same')
     else:
       print('Accept Ho: osrm_distance aggregated value and segment_osrm_distance aggregated value are same.')

     Reject Ho: osrm_distance aggregated value and segment_osrm_distance aggregated value are not same
```

➢ **Step 6: Inference from the analysis.**

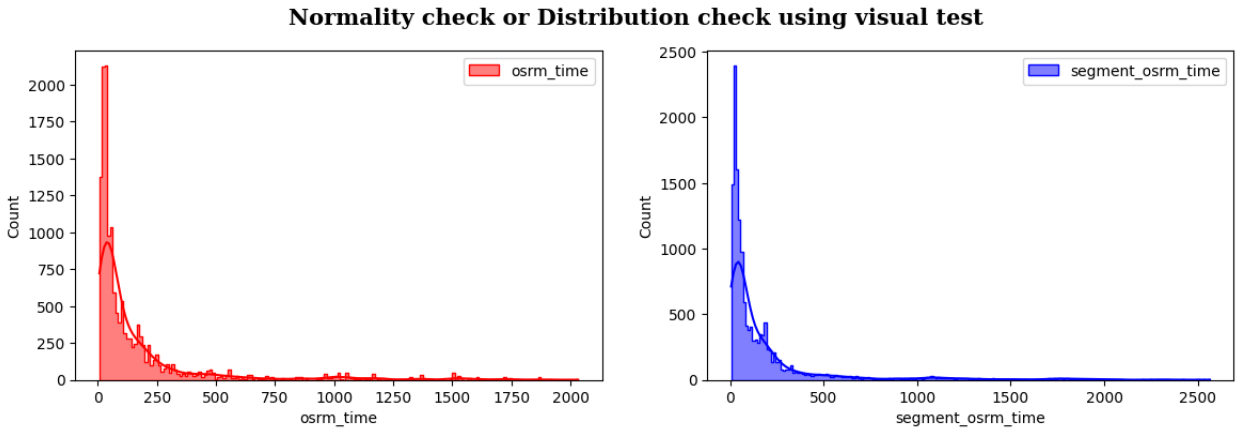
- Finally, we came to the conclusion from the above ks test that osrm_distance aggregated value and segment_osrm_distance aggregated value are not same.

Question: Hypothesis testing/ visual analysis between osrm_time aggregated value and segment_osrm_time aggregated value

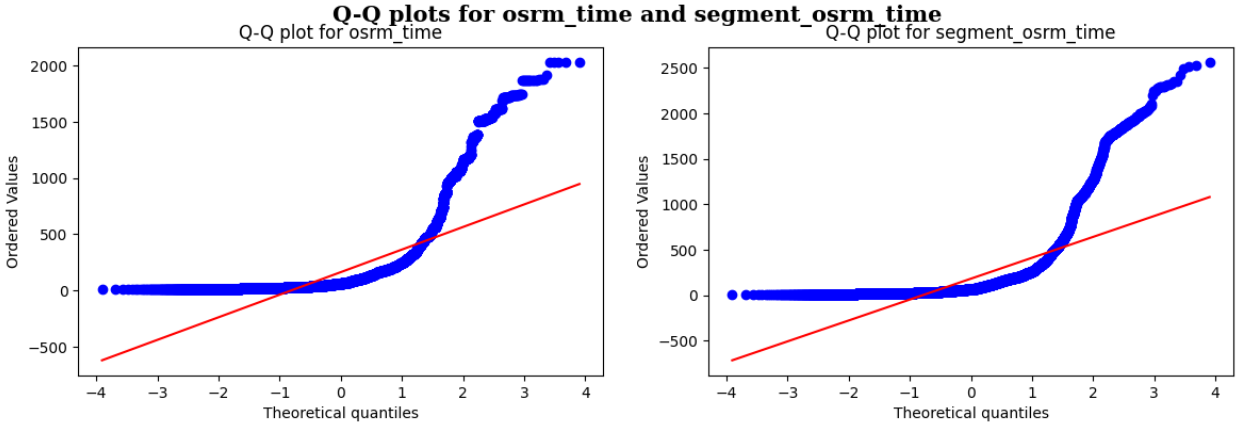
Hypothesis Testing:

- **Step 1: Setup Null Hypothesis**
  - Null Hypothesis (Ho): osrm_time aggregated value and segment_osrm_time aggregated value are same.
  - Alternative Hypothesis (Ha): osrm_time aggregated value and segment_osrm_time aggregated value are not same.

- **Step 2: Checking the assumption for the test.**



- *Normality or Distribution check using histogram or visual test:*
  - We can visualize from the above histogram plot, that the distribution doesn't follow normal distribution.
  - Both the distributions are nearly same and both are right skewed, but the distribution is not normal.



- *Normality or Distribution check using Q-Q plot test:*
  - We can figure it from the above Q-Q plot that the distribution doesn't follow normal distribution.

From the above plots we have seen that the samples do not come from normal distribution. Now we are applying another test Shapiro-wilk test for normality.

- *Normality check using Shapiro-wilk test:*

```
[103] #Normality Check using Shapiro-Wilk test(for osrm_time)

      # Ho: The sample follows normal distribution.
      # Ha: The sample does not follow normal distribution.

      alpha = 0.05
      test_stat, p_value = shapiro(df_nw['osrm_time'].sample(5000))
      print('p-value', p_value)
      if p_value < alpha:
          print('Reject Ho. The sample does not follow normal distribution')
      else:
          print('Fail to reject Ho. The sample follows normal distribution')
```

```
p-value 0.0
Reject Ho. The sample does not follow normal distribution
```

  - For osrm_time, the test result is: p-value 0.0, the sample does not follow. normal distribution

```
[104] #Normality Check using Shapiro-Wilk test(for segment_osrm_time)

      # Ho: The sample follows normal distribution.
      # Ha: The sample does not follow normal distribution.

      alpha = 0.05
      test_stat, p_value = shapiro(df_nw['segment_osrm_time'].sample(5000))
      print('p-value', p_value)
      if p_value < alpha:
          print('Reject Ho. The sample does not follow normal distribution')
      else:
          print('Fail to reject Ho. The sample follows normal distribution')
```

```
p-value 0.0
Reject Ho. The sample does not follow normal distribution
```

  - For segment_osrm_time, the test result is: p-value 0.0, the sample does not follow normal distribution.

Even after applying Shapiro-wilk test, still we find out that the distribution of the "osrm_time" and "segment_osrm_time" data, the samples do not follow normal distribution.

Now, transforming the data using boxcox transformation to check if the transformed data follows normal distribution.

- *Normality Check or Distribution Check after boxcox transformation:*

```
[106] #Normality Check using Shapiro-Wilk test after Boxcox transformation (for osrm_time)

      transformed_osrm_time = spy.boxcox(df_nw['osrm_time'])[0]

      # Ho: The sample follows normal distribution.
      # Ha: The sample does not follow normal distribution.

      alpha = 0.05
      test_stat, p_value = shapiro(transformed_osrm_time)
      print('p-value', p_value)
      if p_value < alpha:
          print('Reject Ho. The sample does not follow normal distribution')
      else:
          print('Fail to reject Ho. The sample follows normal distribution')

      p-value 3.5882550510138333e-35
      Reject Ho. The sample does not follow normal distribution
```

  - For osrm_time, the test result is: p-value 3.5882550510138333e-35, the sample does not follow normal distribution.

```
[107] #Normality Check using Shapiro-Wilk test after Boxcox transformation (for segment_osrm_time)

      transformed_segment_osrm_time = spy.boxcox(df_nw['segment_osrm_time'])[0]

      # Ho: The sample follows normal distribution.
      # Ha: The sample does not follow normal distribution.

      alpha = 0.05
      test_stat, p_value = shapiro(transformed_segment_osrm_time)
      print('p-value', p_value)
      if p_value < alpha:
          print('Reject Ho. The sample does not follow normal distribution')
      else:
          print('Fail to reject Ho. The sample follows normal distribution')

      p-value 4.943039152219146e-34
      Reject Ho. The sample does not follow normal distribution
```

  - For segment_osrm_time, the test result is: p-value 4.943039152219146e-34, the sample does not follow normal distribution.

Even after applying Boxcox transformation, we can see that the distributions for both “osrm_time” and “segment_osrm_time” columns, data doesn’t follow normal distribution.

- *Variance Check using Levene's test:*

```
# Ho - Varience is Equal. Homogenous Variance
# Ha - Varience is Not Equal. Non Homogenous Variance

alpha = 0.05
test_stat, p_value = levene(df_nw['osrm_time'], df_nw['segment_osrm_time'])
print('p-value', p_value)
if p_value < alpha:
    print('reject Ho: The samples do not have  Homogenous Variance')
else:
    print('Fail to Reject Ho: The samples have Homogenous Variance ')
```

```
p-value 8.349506135727595e-08
reject Ho: The samples do not have  Homogenous Variance
```

- The test result is: p-value 8.349506135727595e-08, Reject Ho: The samples do not have Homogenous Variance.

➢ **Step 3: Set a significance level (alpha).**

- We set our alpha to be 0.05.

➢ **Step 4: Calculate test statistics.**

- Standard deviation of the population is not known. So, T-test is right choice for checking the statistics.
- But, we have seen in previously (using histogram plot, Q-Q plot, Shapiro-wilk test) that the distribution is not normal. And in variance test (using Levene's test), we have seen that the variance is non homogeneous.
- Since the samples are not normally distributed. So, T-test is couldn't give us proper statistics result, it probably increase the risk of errors.
- We can perform non-parametric test. i.e; ks- test, ks-test doesn't depend on the distribution.

```
[112] #ks-test

ks_stat,p_value = kstest(df_nw['osrm_time'], df_nw['segment_osrm_time'])

print('ks test statistic result is:', ks_stat)
print('P value is:', p_value)
```

```
ks test statistic result is: 0.0363096443274617
P value is: 6.383943701595088e-09
```

- ***ks-Test:***
  - The result of ks-test: ks test statistic result is: 0.0363096443274617, P value is: 6.383943701595088e-09

➢ **Step 5: Decision to accept or reject null hypothesis.**

- Based on P value, we accept the null hypothesis.

- If P value < significance level (alpha) then reject null hypothesis.
- If P value > significance level (alpha) then accept null hypothesis.

```
# Null Hypothesis (Ho): osrm_time aggregated value and segment_osrm_time aggregated value are same.
# Alternative Hypothesis (Ha): osrm_time aggregated value and segment_osrm_time aggregated value are not same.


alpha = 0.05
if p_value < alpha:
  print('Reject Ho: osrm_time aggregated value and segment_osrm_time aggregated value are not same.')
else:
  print('Accept Ho: osrm_time aggregated value and segment_osrm_time aggregated value are same.')
```

```
Reject Ho: osrm_time aggregated value and segment_osrm_time aggregated value are not same.
```

➢ **Step 6: Inference from the analysis.**

- Finally, we came to the conclusion from the above ks test that osrm_time aggregated value and segment_osrm_time aggregated value are not same.

Outliers in the numerical variables:

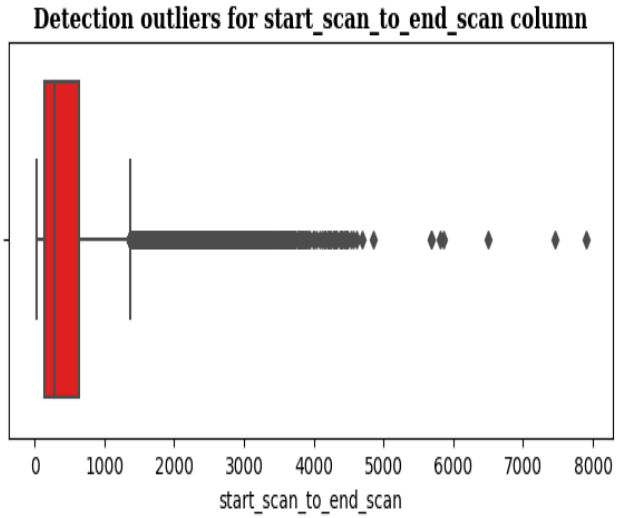
➢ *Analyzing the data summary:*

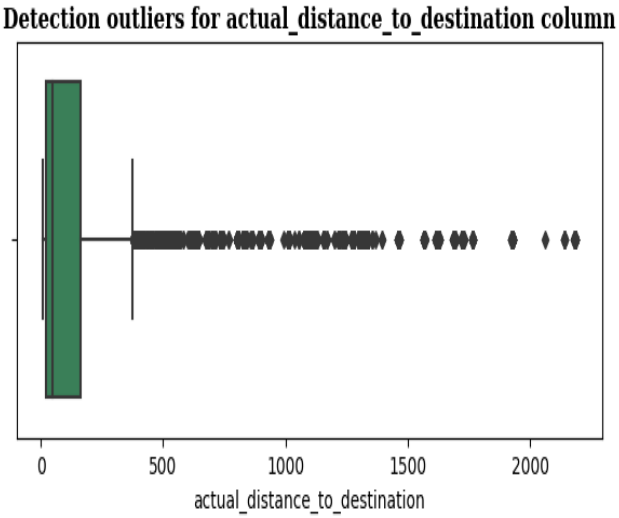
```
numerical_columns = ['start_scan_to_end_scan', 'actual_distance_to_destination',
                     'actual_time', 'osrm_time', 'osrm_distance', 'segment_actual_time',
                     'segment_osrm_time', 'segment_osrm_distance','od_total_time']
df_nw[numerical_columns].describe().T
```

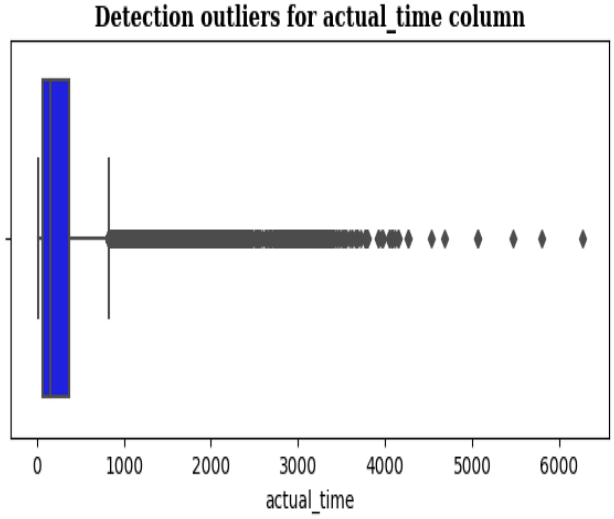
	count	mean	std	min	25%	50%	75%	max
start_scan_to_end_scan	14817.0	530.810016	658.705957	23.000000	149.000000	280.000000	637.000000	7898.000000
actual_distance_to_destination	14817.0	164.477829	305.388153	9.002461	22.837238	48.474072	164.583206	2186.531738
actual_time	14817.0	357.143768	561.396118	9.000000	67.000000	149.000000	370.000000	6265.000000
osrm_time	14817.0	161.384018	271.360992	6.000000	29.000000	60.000000	168.000000	2032.000000
osrm_distance	14817.0	204.344711	370.395569	9.072900	30.819201	65.618805	208.475006	2840.081055
segment_actual_time	14817.0	353.892273	556.247925	9.000000	66.000000	147.000000	367.000000	6230.000000
segment_osrm_time	14817.0	180.949783	314.542053	6.000000	31.000000	65.000000	185.000000	2564.000000
segment_osrm_distance	14817.0	223.201157	416.628387	9.072900	32.654499	70.154404	218.802399	3523.632324
od_total_time	14817.0	547.462995	668.655943	23.460000	151.160000	288.570000	673.300000	7898.550000

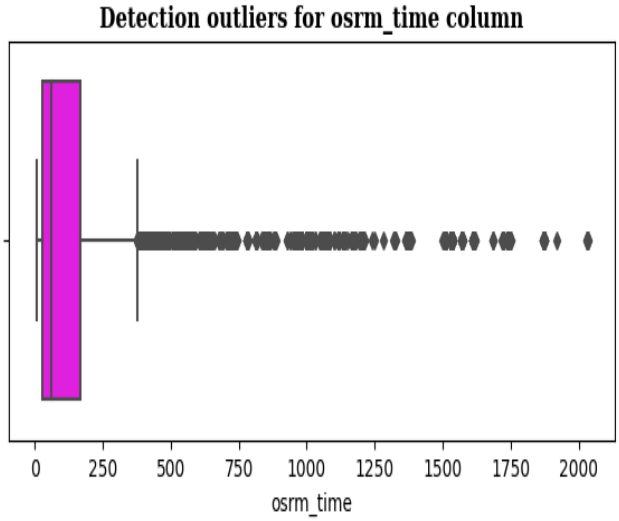
- We can analyze from the above data summary that, in all the numerical columns having a huge difference between their mean and median values, so we can say that all of the numerical columns have outliers.

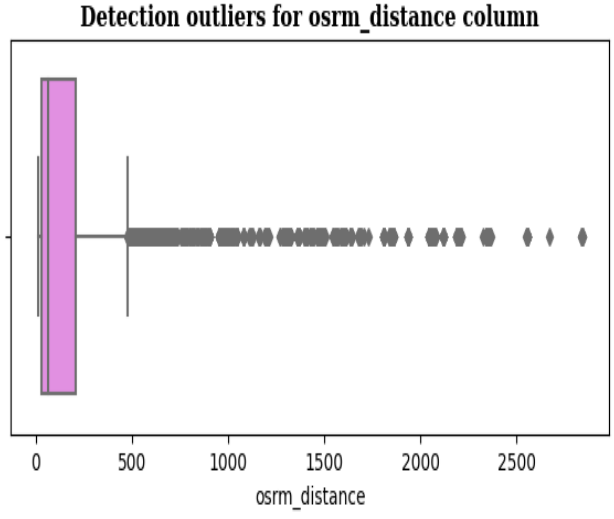
➢ *Outliers Detection by boxplot:*

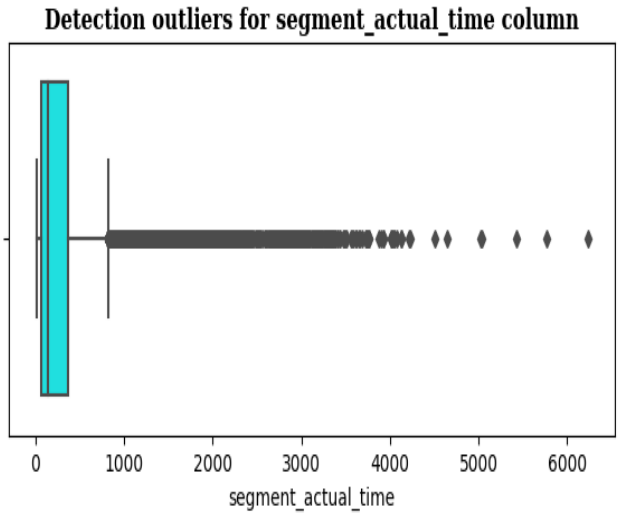

Detection outliers for start_scan_to_end_scan column
0
1000
2000
3000
4000
5000
6000
7000
8000
start_scan_to_end_scan

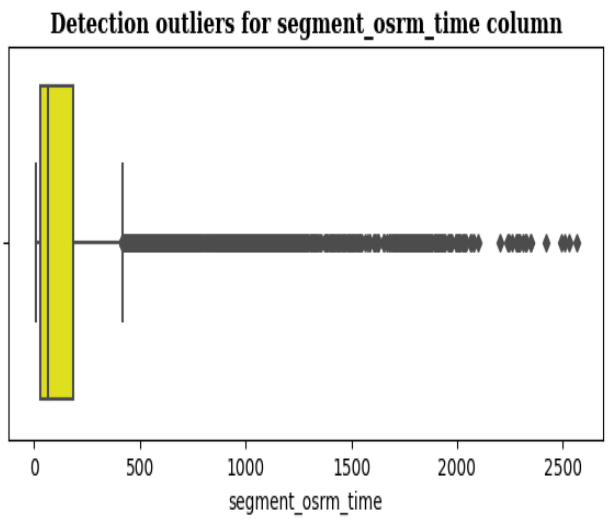

Detection outliers for actual_distance_to_destination column
0
500
1000
1500
2000
actual_distance_to_destination

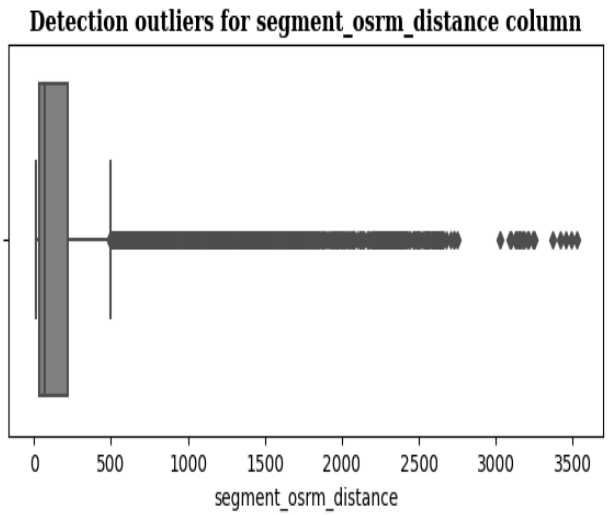

Detection outliers for actual_time column
0
1000
2000
3000
4000
5000
6000
actual_time

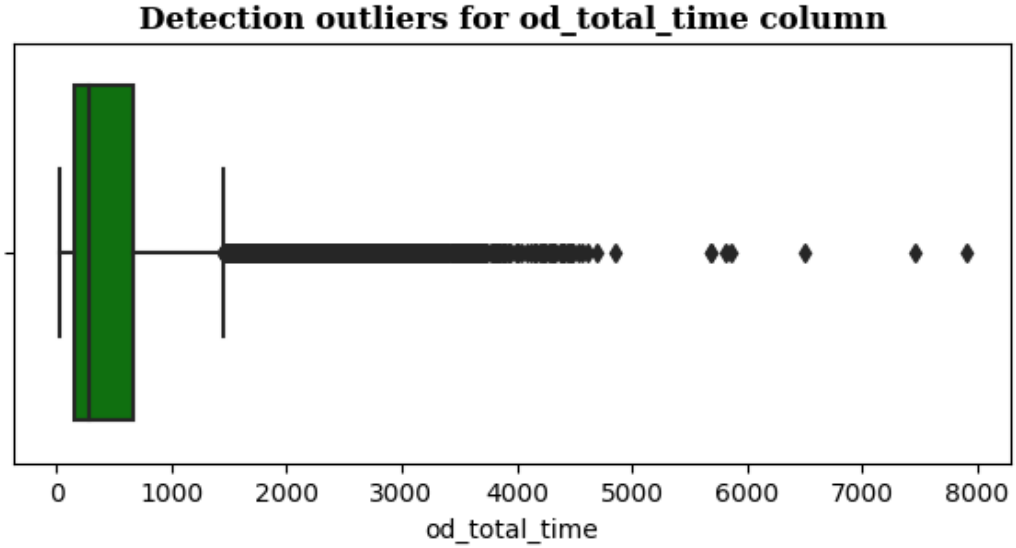

Detection outliers for osrm_time column
0
250
500
750
1000
1250
1500
1750
2000
osrm_time


Detection outliers for osrm_distance column
0
500
1000
1500
2000
2500
osrm_distance


Detection outliers for segment_actual_time column
0
1000
2000
3000
4000
5000
6000
segment_actual_time







- From the above boxplots of numerical variables, we can easily determine that there are outliers in all of these numerical columns.

Handling the outliers using IQR method:

- *Detecting outliers using IQR*

```
Column : start_scan_to_end_scan
Q1 : 149.0
Q3 : 637.0
IQR : 488.0
Lower outlier : -583.0
Upper outlier : 1369.0
Number of outliers : 1267
---------------------------------
Column : actual_distance_to_destination
Q1 : 22.837238311767578
Q3 : 164.5832061767578
IQR : 141.74596786499023
Lower outlier : -189.78171348571777
Upper outlier : 377.20215797424316
Number of outliers : 1449
---------------------------------
Column : actual_time
Q1 : 67.0
Q3 : 370.0
IQR : 303.0
Lower outlier : -387.5
Upper outlier : 824.5
Number of outliers : 1643
---------------------------------
Column : segment_osrm_time
Q1 : 31.0
Q3 : 185.0
IQR : 154.0
Lower outlier : -200.0
Upper outlier : 416.0
Number of outliers : 1492
---------------------------------
Column : segment_osrm_distance
Q1 : 32.65449905395508
Q3 : 218.80239868164062
IQR : 186.14789962768555
Lower outlier : -246.56735038757324
Upper outlier : 498.02424812316895
Number of outliers : 1548
---------------------------------
Column : od_total_time
Q1 : 151.16
Q3 : 673.3
IQR : 522.14
Lower outlier : -632.0500000000001
Upper outlier : 1456.51
Number of outliers : 1115
---------------------------------
Column : osrm_time
Q1 : 29.0
Q3 : 168.0
IQR : 139.0
Lower outlier : -179.5
Upper outlier : 376.5
Number of outliers : 1517
---------------------------------
Column : osrm_distance
Q1 : 30.81920051574707
Q3 : 208.47500610351562
IQR : 177.65580558776855
Lower outlier : -235.66450786590576
Upper outlier : 474.95871448516846
Number of outliers : 1524
---------------------------------
Column : segment_actual_time
Q1 : 66.0
Q3 : 367.0
IQR : 301.0
Lower outlier : -385.5
Upper outlier : 818.5
Number of outliers : 1643
-----------------------------
```

- From the above plots, we can determine that all the numerical columns more than 1200 outliers and these outliers are true outliers.

- Trimming or removing outliers is the best technique to handle the true outliers. So, only when it is required then use trimming process to handle outliers.

➢ *Handling the outliers using IQR by trimming method:*

```
df_without_out = df_nw.copy(deep = True)

for i in numerical_columns:
  Q1 = np.quantile(df_without_out[i], 0.25)
  Q3 = np.quantile(df_without_out[i], 0.75)
  IQR = Q3 - Q1
  Lower_outlier = Q1 - 1.5*IQR
  Higher_outlier =Q3 + 1.5*IQR
    #outliers = df_without_out.loc[(df_without_out[i] < Lower_outlier) | (df_without_out[i] > Higher_outlier)]
  #new dataframe without outliers
  df_without_out = df_without_out.loc[(df_without_out[i] > Lower_outlier) & (df_without_out[i] < Higher_outlier)]
  print(f'Shape of dataframe after removing outliers from {i} column : {df_without_out.shape}')
print('-'*30)
print('Shape of new dataframe after removing all outliers is: ', df_without_out.shape)
```

```
Shape of dataframe after removing outliers from start_scan_to_end_scan column : (13550, 28)
Shape of dataframe after removing outliers from actual_distance_to_destination column : (12737, 28)
Shape of dataframe after removing outliers from actual_time column : (12113, 28)
Shape of dataframe after removing outliers from osrm_time column : (11506, 28)
Shape of dataframe after removing outliers from osrm_distance column : (10782, 28)
Shape of dataframe after removing outliers from segment_actual_time column : (10444, 28)
Shape of dataframe after removing outliers from segment_osrm_time column : (9753, 28)
Shape of dataframe after removing outliers from segment_osrm_distance column : (9153, 28)
Shape of dataframe after removing outliers from od_total_time column : (8701, 28)
------------------------------
Shape of new dataframe after removing all outliers is:  (8701, 28)
```

- After trimming, the shape of new dataframe becomes 8701 rows and 28 columns.

<u>One-Hot Encoding of Categorical Variables:</u>

➢ *One-Hot Encoding for route types & data columns:*

- One hot encoding is a technique that we use to represent categorical variables as numerical values

- In this dataset, two categorical columns are present: data and route_type columns.
- "data" column has two variables: training and test.
  And "route_type" column has two variables: Carting and FTL.

route_type_nw	data_nw	0	1	2	3
1	1	0.0	1.0	0.0	1.0
0	1	1.0	0.0	0.0	1.0
1	1	0.0	1.0	0.0	1.0
0	1	1.0	0.0	0.0	1.0
1	1	0.0	1.0	0.0	1.0

- By one hot encoding technique these variables converted into numerical variables.

Normalize/ Standardize the numerical features using MinMaxScaler or StandardScaler:

- **Normalization or Min-Max Scaling** is used to transform features to be on a similar scale. The new point is calculated as:

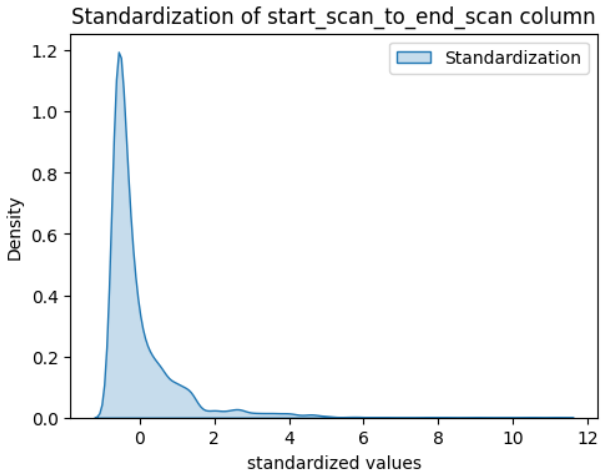
  X_new = (X - X_min)/(X_max - X_min)

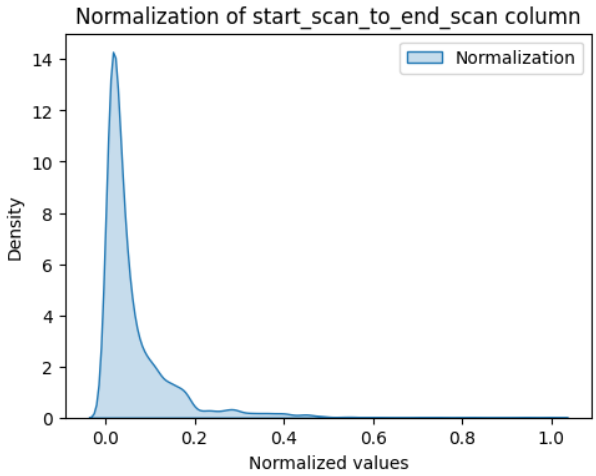
  This scales the range to [0, 1] or sometimes [-1, 1].

- **Standardization or Z-Score Normalization** is the transformation of features by subtracting from mean and dividing by standard deviation. This is often called as Z-score.

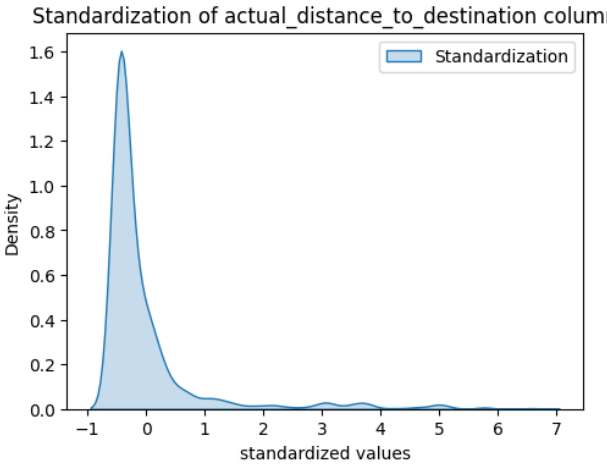
  X_new = (X - mean)/Std

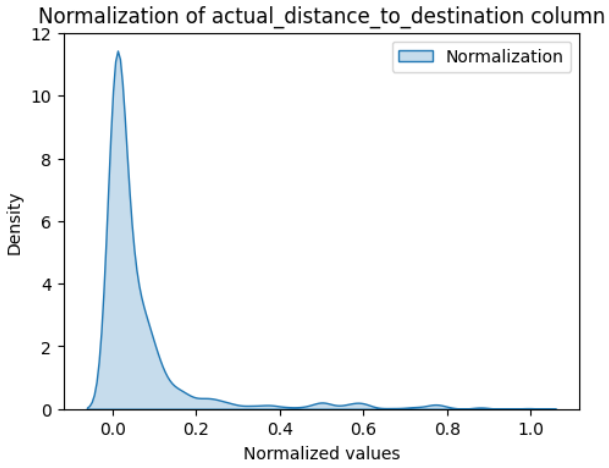
➢ *Normalization & Standardization for start_scan_to_end_scan column:*



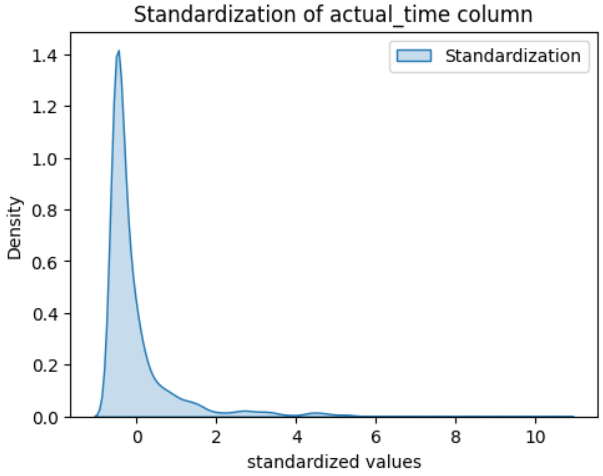


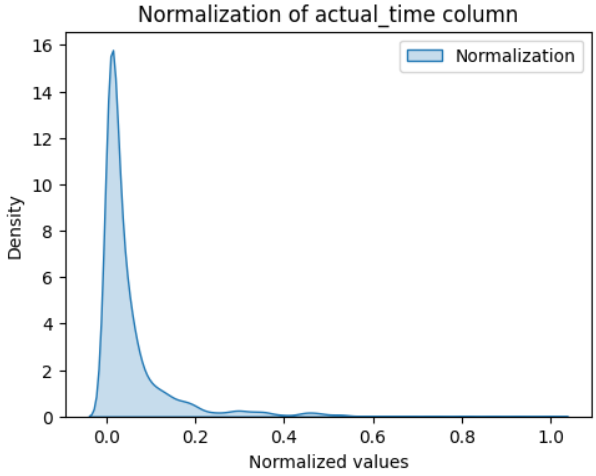
➢ *Normalization & Standardization for actual_distance_to_destination column:*



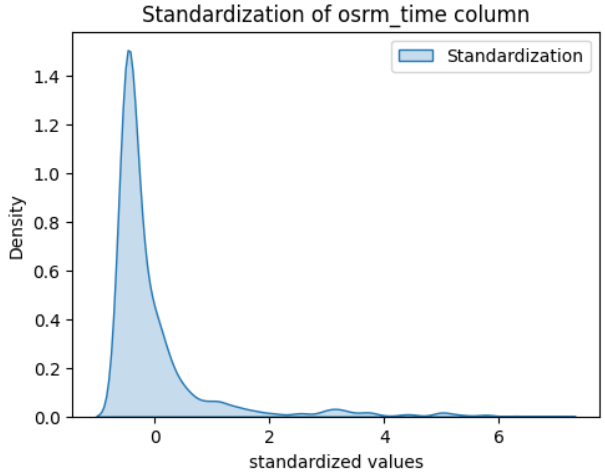


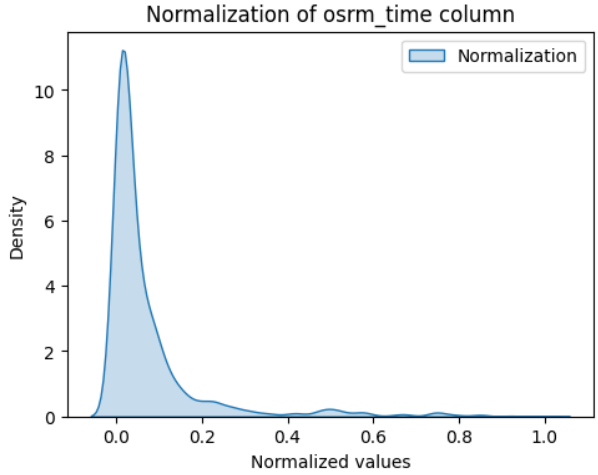
➢ *Normalization & Standardization for actual_time column:*



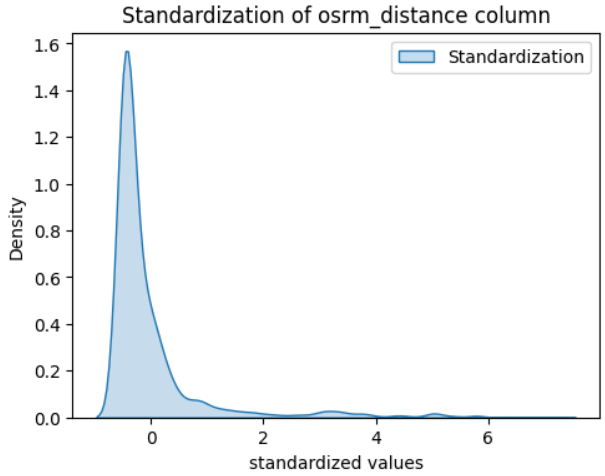


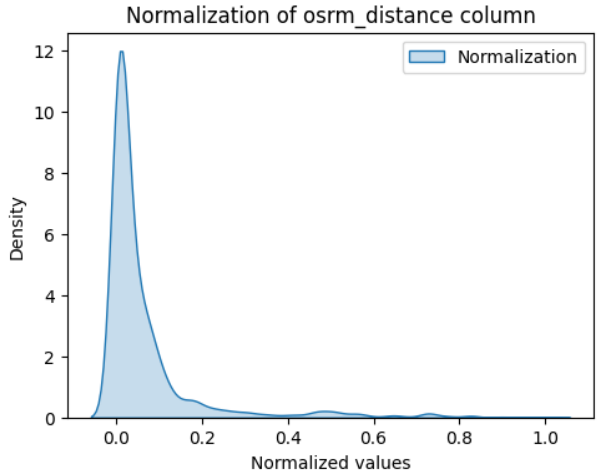
➢ *Normalization & Standardization for osrm_time column:*



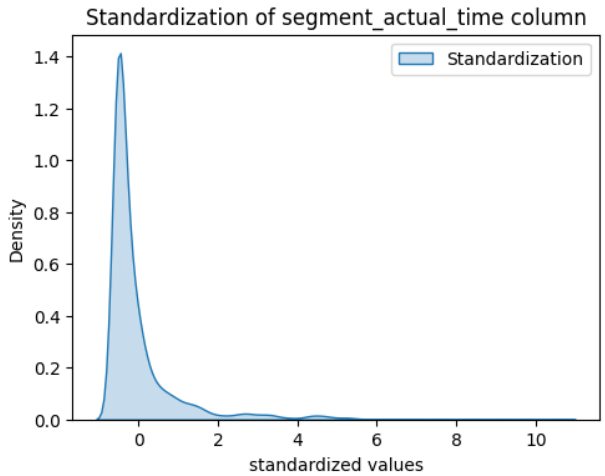


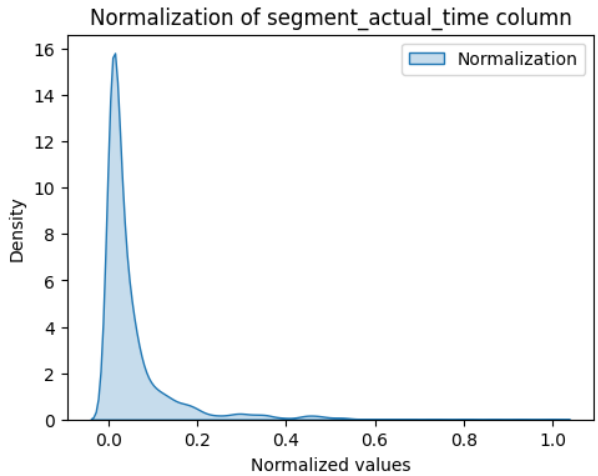
➢ *Normalization & Standardization for osrm_distance column:*



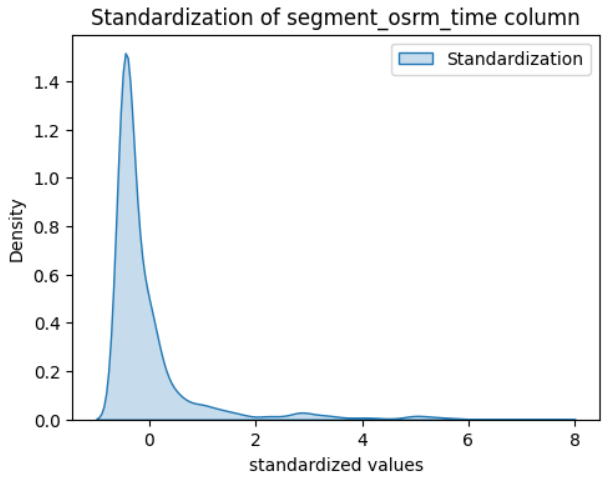


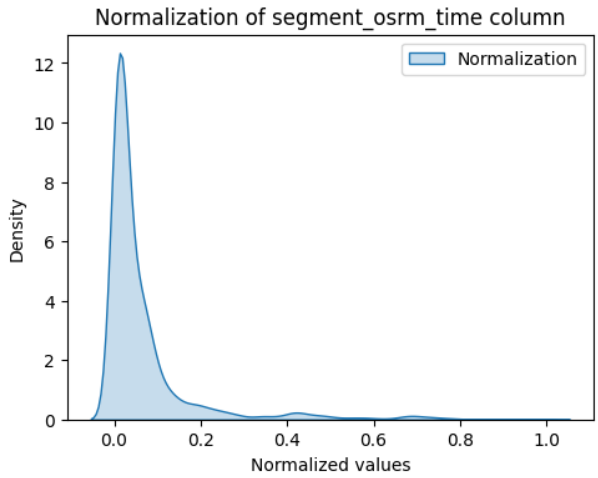
➢ *Normalization & Standardization for segment_actual_time column:*



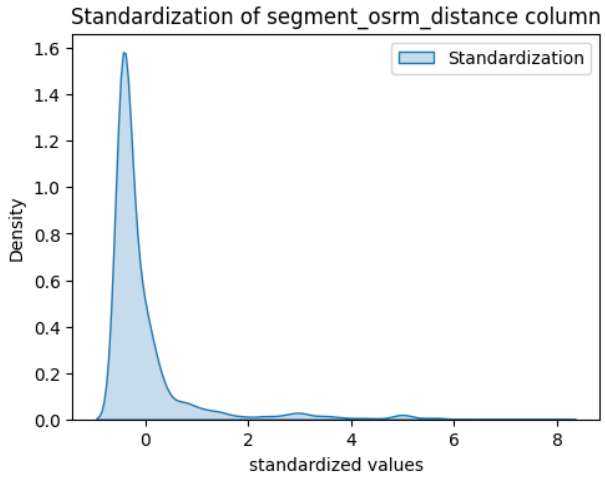


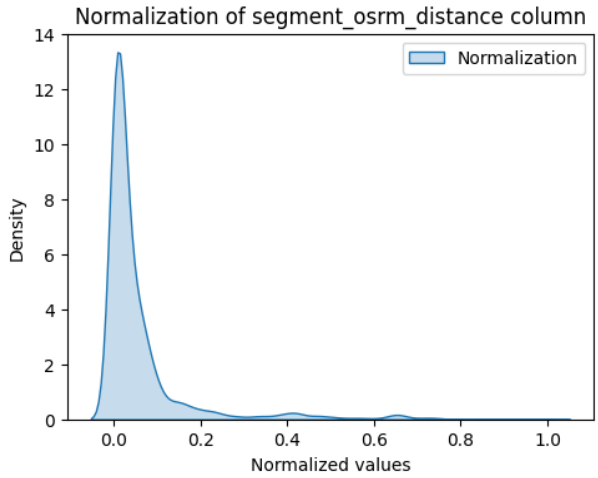
➢ *Normalization & Standardization for segment_osrm_time column:*



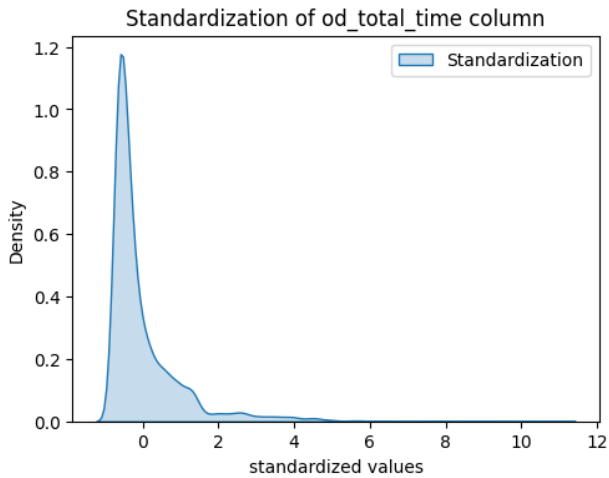


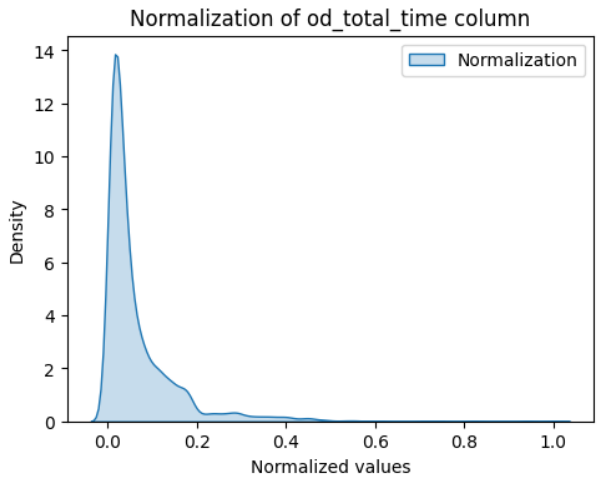
➢ *Normalization & Standardization for segment_osrm_distance column:*





➢ *Normalization & Standardization for od_total_time column:*





Business Insights:

- *Busiest Corridor (City to City) with Average distance and Average time:*

```
#busiest corridor (City to City)

corridor_city = df_nw.groupby(by = ['source_city', 'destination_city']).agg({'trip_uuid' : 'count', 'actual_distance_to_destination': 'mean',
                                                                              'actual_time': 'mean'})
corridor_city.sort_values('trip_uuid', ascending = False)
```

source_city	destination_city	trip_uuid	actual_distance_to_destination	actual_time
Mumbai	Mumbai	600	15.988461	62.814999
Bengaluru	Bengaluru	549	32.252834	88.089256
Bangalore	Bengaluru	455	27.056046	77.720879
Bhiwandi	Mumbai	437	22.610231	74.281464
Hyderabad	Hyderabad	398	85.867233	200.856781
...	...	...	...	...

- *Busiest Corridor (State to State) with Average distance and Average time:*

```
[152] #busiest corridor (state to state)

corridor_state = df_nw.groupby(by = ['source_state', 'destination_state']).agg({'trip_uuid' : 'count', 'actual_distance_to_destination': 'mean',
                                                                                'actual_time': 'mean'})
corridor_state.sort_values('trip_uuid', ascending = False)
```

source_state	destination_state	trip_uuid	actual_distance_to_destination	actual_time
Maharashtra	Maharashtra	2406	60.110615	164.041565
Karnataka	Karnataka	2014	55.177082	137.809341
Tamil Nadu	Tamil Nadu	1016	70.125748	153.220474
Haryana	Haryana	871	122.749580	277.407562
Telangana	Telangana	655	96.038033	217.586258

➢ *Busiest Corridor (Interstate) with Average distance and Average time:*

```
#busiest corridor(interstate)
corridor_interstate = df_nw[(df_nw['destination_state']) != (df_nw['source_state'])].groupby(['source_state','destination_state']
                                                                                             ).agg({'trip_uuid' : 'count',
                                                                                                    'actual_distance_to_destination': 'mean',
                                                                                                    'actual_time': 'mean'})


corridor_interstate.sort_values('trip_uuid', ascending = False)
```

source_state	destination_state	trip_uuid	actual_distance_to_destination	actual_time
Delhi	Haryana	396	44.560055	142.856064
Haryana	Delhi	311	45.686760	126.958199
	Uttar Pradesh	93	143.393845	320.924744
Delhi	Uttar Pradesh	84	236.478729	424.154755
Haryana	Punjab	82	177.346695	365.475616
...	...			

➢ *Busiest Corridor (Intercity) with Average distance and Average time:*

```
#busiest corridor(intercity)
corridor_intercity = df_nw[(df_nw['destination_city']) != (df_nw['source_city'])].groupby(['source_city',
                                                                                           'destination_city']).agg({
                                                                                               'trip_uuid' : 'count',
                                                                                               'actual_distance_to_destination': 'mean',
                                                                                               'actual_time': 'mean'})


corridor_intercity.sort_values('trip_uuid', ascending = False)
```

source_city	destination_city	trip_uuid	actual_distance_to_destination	actual_time
Bangalore	Bengaluru	455	27.056046	77.720879
Bhiwandi	Mumbai	437	22.610231	74.281464
Bengaluru	Bangalore	326	27.675541	92.223923
Mumbai	Bhiwandi	270	21.562836	93.933334
Gurgaon	Delhi	240	48.673473	126.500000

Colab notebook link:

https://colab.research.google.com/drive/1WOv-hZBcClKP7TtbD2BzhaOMdgV5Jk_F?usp=sharing