Business Case: 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 :

- Clean, sanitize and manipulate data to get useful features out of raw fields.
- Make sense out of the raw data and help the data science team to build forecasting models on it.

Column Profiling :

- data - tells whether the data is testing or training data
- trip_creation_time – Timestamp of trip creation
- route_schedule_uuid – Unique Id for a particular route schedule
- route_type – Transportation type
    - FTL – Full Truck Load: FTL shipments get to the destination sooner, as the truck is making no other pickups or drop-offs along the way
    - Carting: Handling system consisting of small vehicles (carts)
- trip_uuid - Unique ID given to a particular trip (A trip may include different source and destination centers)
- source_center - Source ID of trip origin
- source_name - Source Name of trip origin
- destination_cente – Destination ID
- destination_name – Destination Name
- od_start_time – Trip start time

- od_end_time – Trip end time
- start_scan_to_end_scan – Time taken to deliver from source to destination
- is_cutoff – Unknown field
- cutoff_factor – Unknown field
- cutoff_timestamp – Unknown field
- actual_distance_to_destination – Distance in Kms between source and destination warehouse
- actual_time – Actual time taken to complete the delivery (Cumulative)
- osrm_time – An open-source routing engine time calculator which computes the shortest path between points in a given map (Includes usual traffic, distance through major and minor roads) and gives the time (Cumulative)
- osrm_distance – An open-source routing engine which computes the shortest path between points in a given map (Includes usual traffic, distance through major and minor roads) (Cumulative)
- factor – Unknown field
- segment_actual_time – This is a segment time. Time taken by the subset of the package delivery
- segment_osrm_time – This is the OSRM segment time. Time taken by the subset of the package delivery
- segment_osrm_distance – This is the OSRM distance. Distance covered by subset of the package delivery
- segment_factor – Unknown field

Importing Libraries and dataset

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

```
data=pd.read_csv("https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/551/
```

Basic Cleaning and Exploration

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 144867 entries, 0 to 144866
Data columns (total 24 columns):
```

```
 #   Column                          Non-Null Count   Dtype  
---  ------                          --------------   -----  
 0   data                            144867 non-null  object 
 1   trip_creation_time              144867 non-null  object 
 2   route_schedule_uuid             144867 non-null  object 
 3   route_type                      144867 non-null  object 
 4   trip_uuid                       144867 non-null  object 
 5   source_center                   144867 non-null  object 
 6   source_name                     144574 non-null  object 
 7   destination_center              144867 non-null  object 
 8   destination_name                144606 non-null  object 
 9   od_start_time                   144867 non-null  object 
 10  od_end_time                     144867 non-null  object 
 11  start_scan_to_end_scan          144867 non-null  float64
 12  is_cutoff                       144867 non-null  bool   
 13  cutoff_factor                   144867 non-null  int64  
 14  cutoff_timestamp                144867 non-null  object 
 15  actual_distance_to_destination  144867 non-null  float64
 16  actual_time                     144867 non-null  float64
 17  osrm_time                       144867 non-null  float64
 18  osrm_distance                   144867 non-null  float64
 19  factor                          144867 non-null  float64
 20  segment_actual_time             144867 non-null  float64
 21  segment_osrm_time               144867 non-null  float64
 22  segment_osrm_distance           144867 non-null  float64
 23  segment_factor                  144867 non-null  float64
dtypes: bool(1), float64(10), int64(1), object(12)
memory usage: 25.6+ MB
```

```
data.describe()
```

	start_scan_to_end_scan	cutoff_factor	actual_distance_to_destination	actua
count	144867.000000	144867.000000	144867.000000	144867.(
mean	961.262986	232.926567	234.073372	416.!
std	1037.012769	344.755577	344.990009	598.
min	20.000000	9.000000	9.000045	9.(
25%	161.000000	22.000000	23.355874	51.(
50%	449.000000	66.000000	66.126571	132.(
75%	1634.000000	286.000000	286.708875	513.(
max	7898.000000	1927.000000	1927.447705	4532.(

```
data.nunique()
```

	0
data	2
trip_creation_time	14817
route_schedule_uuid	1504
route_type	2
trip_uuid	14817
source_center	1508
source_name	1498
destination_center	1481
destination_name	1468
od_start_time	26369
od_end_time	26369
start_scan_to_end_scan	1915
is_cutoff	2
cutoff_factor	501
cutoff_timestamp	93180
actual_distance_to_destination	144515
actual_time	3182
osrm_time	1531
osrm_distance	138046
factor	45641
segment_actual_time	747
segment_osrm_time	214
segment_osrm_distance	113799
segment_factor	5675

dtype: int64

```
pd.set_option('display.max_columns', None)
# to display all columns
```

```
data.head()
```

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

```
data.shape
```

```
(144867, 24)
```

```
data.ndim
```

```
2
```

```
data.head(5)
```

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

```
data.isna().sum()
```

	0
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

```
# Function to create a data frame with number and percentage of missing data in a data fr

def missing_values(data):
    # Number and percentage of missing data in data set for each column
    total_missing_data = data.isnull().sum().sort_values(ascending =False)
    percent_missing_data = (data.isnull().sum()/data.isnull().count()*100).sort_values(as
    missing_values_data = pd.concat([total_missing_data, percent_missing_data], axis=1, k
```

```
    return missing_values_data
```

```
missing_data=missing_values(data)
missing_data
```

	Total	Percent
source_name	293	0.202254
destination_name	261	0.180165
data	0	0.000000
cutoff_factor	0	0.000000
segment_osrm_distance	0	0.000000
segment_osrm_time	0	0.000000
segment_actual_time	0	0.000000
factor	0	0.000000
osrm_distance	0	0.000000
osrm_time	0	0.000000
actual_time	0	0.000000
actual_distance_to_destination	0	0.000000
cutoff_timestamp	0	0.000000
is_cutoff	0	0.000000
trip_creation_time	0	0.000000
start_scan_to_end_scan	0	0.000000
od_end_time	0	0.000000
od_start_time	0	0.000000
destination_center	0	0.000000
source_center	0	0.000000
trip_uuid	0	0.000000
route_type	0	0.000000
route_schedule_uuid	0	0.000000
segment_factor	0	0.000000

```
data.dropna(inplace=True)
```

```
data.isnull().sum().sum()
```

```
0
```

Understanding the flow

```
data_copy=data.copy()
```

```
data_copy_grouped=data_copy.groupby(['trip_uuid','source_center','destination_center']).c
data_copy_grouped
```

	trip_uuid	source_center	destination_center	data	trip_creation_ti
0	trip-153671041653548748	IND209304AAA	IND000000ACB	18	
1	trip-153671041653548748	IND462022AAA	IND209304AAA	21	
2	trip-153671042288605164	IND561203AAB	IND562101AAA	3	
3	trip-153671042288605164	IND572101AAA	IND561203AAB	6	
4	trip-153671043369099517	IND000000ACB	IND160002AAC	12	
...	...	...	...	...	
26217	trip-153861115439069069	IND628204AAA	IND627657AAA	4	
26218	trip-153861115439069069	IND628613AAA	IND627005AAA	4	
26219	trip-153861115439069069	IND628801AAA	IND628204AAA	2	
26220	trip-153861118270144424	IND583119AAA	IND583101AAA	2	
26221	trip-153861118270144424	IND583201AAA	IND583119AAA	2	

26222 rows × 24 columns

```
data_copy[data_copy['trip_uuid']=='trip-153671041653548748']
```

	data	trip_creation_time	route_schedule_uuid	route_type	tr
124981	training	2018-09-12 00:00:16.535741	thanos::sroute:d7c989ba-a29b-4a0b-b2f4-288cdc6...	FTL	15367104165
124982	training	2018-09-12 00:00:16.535741	thanos::sroute:d7c989ba-a29b-4a0b-b2f4-288cdc6...	FTL	15367104165
124983	training	2018-09-12 00:00:16.535741	thanos::sroute:d7c989ba-a29b-4a0b-b2f4-288cdc6...	FTL	15367104165
124984	training	2018-09-12 00:00:16.535741	thanos::sroute:d7c989ba-a29b-4a0b-b2f4-288cdc6...	FTL	15367104165
124985	training	2018-09-12 00:00:16.535741	thanos::sroute:d7c989ba-a29b-4a0b-b2f4-288cdc6...	FTL	15367104165
124986	training	2018-09-12 00:00:16.535741	thanos::sroute:d7c989ba-a29b-4a0b-b2f4-288cdc6...	FTL	15367104165
124987	training	2018-09-12 00:00:16.535741	thanos::sroute:d7c989ba-a29b-4a0b-b2f4-288cdc6...	FTL	15367104165
124988	training	2018-09-12 00:00:16.535741	thanos::sroute:d7c989ba-a29b-4a0b-b2f4-288cdc6...	FTL	15367104165
124989	training	2018-09-12 00:00:16.535741	thanos::sroute:d7c989ba-a29b-4a0b-b2f4-288cdc6...	FTL	15367104165
124990	training	2018-09-12 00:00:16.535741	thanos::sroute:d7c989ba-a29b-4a0b-b2f4-288cdc6...	FTL	15367104165
124991	training	2018-09-12 00:00:16.535741	thanos::sroute:d7c989ba-a29b-4a0b-b2f4-288cdc6...	FTL	15367104165
124992	training	2018-09-12 00:00:16.535741	thanos::sroute:d7c989ba-a29b-4a0b-b2f4-288cdc6...	FTL	15367104165
124993	training	2018-09-12 00:00:16.535741	thanos::sroute:d7c989ba-a29b-4a0b-b2f4-288cdc6...	FTL	15367104165
124994	training	2018-09-12 00:00:16.535741	thanos::sroute:d7c989ba-a29b-4a0b-b2f4-288cdc6...	FTL	15367104165
124995	training	2018-09-12	thanos::sroute:d7c989ba-a29b-4a0b-b2f4-	FTL	

Converting the datatype to datetime format

```
#changing datatype of date like columns from object to timestamp
data_copy[["od_start_time", "od_end_time",'trip_creation_time']] = data_copy[["od_start_t
data_copy.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 144316 entries, 0 to 144866
Data columns (total 24 columns):
 #   Column                          Non-Null Count   Dtype
---  ------                          --------------   -----
 0   data                            144316 non-null  object
 1   trip_creation_time              144316 non-null  datetime64[ns]
 2   route_schedule_uuid             144316 non-null  object
 3   route_type                      144316 non-null  object
 4   trip_uuid                       144316 non-null  object
 5   source_center                   144316 non-null  object
 6   source_name                     144316 non-null  object
 7   destination_center              144316 non-null  object
 8   destination_name                144316 non-null  object
 9   od_start_time                   144316 non-null  datetime64[ns]
 10  od_end_time                     144316 non-null  datetime64[ns]
 11  start_scan_to_end_scan          144316 non-null  float64
 12  is_cutoff                       144316 non-null  bool
 13  cutoff_factor                   144316 non-null  int64
 14  cutoff_timestamp                144316 non-null  object
 15  actual_distance_to_destination  144316 non-null  float64
 16  actual_time                     144316 non-null  float64
 17  osrm_time                       144316 non-null  float64
 18  osrm_distance                   144316 non-null  float64
 19  factor                          144316 non-null  float64
 20  segment_actual_time             144316 non-null  float64
 21  segment_osrm_time               144316 non-null  float64
 22  segment_osrm_distance           144316 non-null  float64
 23  segment_factor                  144316 non-null  float64
dtypes: bool(1), datetime64[ns](3), float64(10), int64(1), object(9)
memory usage: 26.6+ MB
```

Extracting and Creating New Columns

```
#extracting day, month & year from trip_creation_time
data_copy['trip_creation_month']=data_copy['trip_creation_time'].dt.month
data_copy['trip_creation_year']=data_copy['trip_creation_time'].dt.year
data_copy['trip_creation_day']=data_copy['trip_creation_time'].dt.day
data_copy.head(1)
```

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

thanos::sroute:d7c989ba-

```
#difference between od_start & od_end time in hours
data_copy['Timediff_start_end_H']=round((data_copy['od_end_time']-data_copy['od_start_tim
data_copy.head(1)
```

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

125015 training 00:00:16.535741 a29b-4a0b-b214- FTL 15367104165

Analyzing a single trip and its flow.

288cdc0...

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

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

Creating Features

```
#as below mentioned columns are comprising of segment related details we will do a cum. s
data_copy['agg_segment_actual_time']=data_copy.groupby(['trip_uuid','source_center','dest
data_copy['agg_segment_osrm_time']=data_copy.groupby(['trip_uuid','source_center','destin
data_copy['agg_segment_osrm_distance']=data_copy.groupby(['trip_uuid','source_center','de
```

```
data_copy.head()
```

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

```
#After finding out the cum. sum of above columns we will pick their max
data_copy['agg_segment_actual_time1']=data_copy.groupby(['trip_uuid','source_center','des
data_copy['agg_segment_osrm_time1']=data_copy.groupby(['trip_uuid','source_center','desti
data_copy['agg_segment_osrm_distance1']=data_copy.groupby(['trip_uuid','source_center','d

data_copy.head()
```

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

```
# aggregation of below mentioned based on their Trip_uuid, Source ID and Destination ID
# as they are mentioned as a cumsum in data dictionary we will take max
data_copy['agg_distance_to_destination']=data_copy.groupby(['trip_uuid','source_center','
data_copy['agg_actual_time']=data_copy.groupby(['trip_uuid','source_center','destination_
```

```
data_copy['agg_osrm_time']=data_copy.groupby(['trip_uuid','source_center','destination_ce
data_copy['agg_osrm_distance']=data_copy.groupby(['trip_uuid','source_center','destinatic
data_copy.head()
```

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

```
#creating column with city, place, state from source centre & destination centre
data_copy[['Source_City','Source_Place','Source_Code/State']]=data_copy['source_name'].st
data_copy.head()
```

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

```
data_copy[['destination_City','destination_Place','destination_Code/State']]=data_copy['d
data_copy.head()
```

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

```
#creating Source Code, Source state column, Destination Code, Destination state column fr
data_copy[['Source_Code','Source_State']]=data_copy['Source_Code/State'].str.rsplit('(',n
data_copy[['destination_Code','destination_State']]=data_copy['destination_Code/State'].s
data_copy.head()
```

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

```
data_copy['Source_State'] = data_copy['Source_State'].str.rstrip(')')
data_copy['destination_State'] = data_copy['destination_State'].str.rstrip(')')
data_copy.head()
```

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

```
#dropping the existing columns as we have already got engineered features from them
data_copy.drop(columns=['od_end_time','od_start_time','trip_creation_time','source_name',
```

```
print('Rows:', data_copy.shape[0],'\n' 'Columns: ',data_copy.shape[1])
```

```
Rows: 144316
Columns:  43
```

```
data_copy.drop(columns=['segment_factor','data','factor','is_cutoff', 'cutoff_factor','cu
                 'route_schedule_uuid'],
        axis=1, inplace=True)
```

```
print('Rows:', data_copy.shape[0],'\n' 'Columns: ',data_copy.shape[1])
```

```
Rows: 144316
Columns:  36
```

```
data_copy.duplicated().sum()
```

```
0
```

The data's were not duplicated as the original columns have unique values. Lets create a new dataframe which includes the newly created features column.

```
data_copy.columns
```

```
Index(['route_type', 'trip_uuid', 'source_center', 'destination_center',
       'start_scan_to_end_scan', 'actual_distance_to_destination',
       'actual_time', 'osrm_time', 'osrm_distance', 'segment_actual_time',
       'segment_osrm_time', 'segment_osrm_distance', 'trip_creation_month',
       'trip_creation_year', 'trip_creation_day', 'Timediff_start_end_H',
       'agg_segment_actual_time', 'agg_segment_osrm_time',
       'agg_segment_osrm_distance', 'agg_segment_actual_time1',
       'agg_segment_osrm_time1', 'agg_segment_osrm_distance1',
       'agg_distance_to_destination', 'agg_actual_time', 'agg_osrm_time',
       'agg_osrm_distance', 'Source_City', 'Source_Place', 'Source_Code/State',
       'destination_City', 'destination_Place', 'destination_Code/State',
       'Source_Code', 'Source_State', 'destination_Code', 'destination_State'],
      dtype='object')
```

```
data_merged=data_copy.loc[:,['route_type', 'trip_uuid',
       'start_scan_to_end_scan',  'trip_creation_month',
       'trip_creation_year', 'trip_creation_day', 'Timediff_start_end_H', 'agg_segment_ac
       'agg_segment_osrm_time1', 'agg_segment_osrm_distance1',
       'agg_distance_to_destination', 'agg_actual_time', 'agg_osrm_time',
       'agg_osrm_distance', 'Source_City', 'Source_Place', 'Source_Code/State',
       'destination_City', 'destination_Place', 'destination_Code/State']]
```

```
data_merged.duplicated().sum()
```

```
118093
```

```
data_merged.head()
```

	route_type	trip_uuid	start_scan_to_end_scan	trip_creation_month	trip_
0	Carting	trip-153741093647649320	86.0	9	
1	Carting	trip-153741093647649320	86.0	9	
2	Carting	trip-153741093647649320	86.0	9	
3	Carting	trip-153741093647649320	86.0	9	
4	Carting	trip-153741093647649320	86.0	9	

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

	route_type	trip_uuid	start_scan_to_end_scan	trip_creation_month	trip_
0	Carting	trip-153741093647649320	86.0	9	
1	Carting	trip-153741093647649320	86.0	9	
2	Carting	trip-153741093647649320	86.0	9	
3	Carting	trip-153741093647649320	86.0	9	
4	Carting	trip-153741093647649320	86.0	9	
5	Carting	trip-153741093647649320	109.0	9	
6	Carting	trip-153741093647649320	109.0	9	
7	Carting	trip-153741093647649320	109.0	9	
8	Carting	trip-153741093647649320	109.0	9	
9	Carting	trip-153741093647649320	109.0	9	

```
data_merged.shape
```

```
(144316, 20)
```

```
data_merged.duplicated().sum()
```

```
118093
```

```
data_merged.drop_duplicates(inplace=True)
data_merged.head()
```

	route_type	trip_uuid	start_scan_to_end_scan	trip_creation_month	tri
0	Carting	trip-153741093647649320	86.0	9	
5	Carting	trip-153741093647649320	109.0	9	
10	FTL	trip-153768492602129387	302.0	9	
15	Carting	trip-153693976643699843	108.0	9	
17	FTL	trip-153687145942424248	195.0	9	

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

	route_type	trip_uuid	start_scan_to_end_scan	trip_creation_month	trip_
0	Carting	trip-153741093647649320	86.0	9	
5	Carting	trip-153741093647649320	109.0	9	

```
data_merged.duplicated().sum()
```

```
0
```

```
data_merged.shape
```

```
(26223, 20)
```

```
data_merged.columns
```

```
Index(['route_type', 'trip_uuid', 'start_scan_to_end_scan',
       'trip_creation_month', 'trip_creation_year', 'trip_creation_day',
       'Timediff_start_end_H', 'agg_segment_actual_time1',
       'agg_segment_osrm_time1', 'agg_segment_osrm_distance1',
       'agg_distance_to_destination', 'agg_actual_time', 'agg_osrm_time',
       'agg_osrm_distance', 'Source_City', 'Source_Place', 'Source_Code/State',
       'destination_City', 'destination_Place', 'destination_Code/State'],
      dtype='object')
```

```
data_merged.head()
```

	route_type	trip_uuid	start_scan_to_end_scan	trip_creation_month	tri
0	Carting	trip-153741093647649320	86.0	9	
5	Carting	trip-153741093647649320	109.0	9	
10	FTL	trip-153768492602129387	302.0	9	
15	Carting	trip-153693976643699843	108.0	9	
17	FTL	trip-153687145942424248	195.0	9	

Lets create a dataframe having unique rows for trips by combining, Summing the rows of subset package of the trips

```
data_uuid=data_merged.copy()
```

```
# aggregation of below mentioned based on their Trip_uuid, Source ID and Destination ID
# as they are mentioned as a cum. sum in data dictionary we will take max
data_uuid['start_scan_to_end_scan11']=data_uuid.groupby(['trip_uuid'])['start_scan_to_end
data_uuid['Timediff_start_end_H11']=data_uuid.groupby(['trip_uuid'])['Timediff_start_end_
data_uuid['agg_segment_actual_time11']=data_uuid.groupby(['trip_uuid'])['agg_segment_actu
data_uuid['agg_segment_osrm_time11']=data_uuid.groupby(['trip_uuid'])['agg_segment_osrm_t

data_uuid['agg_segment_osrm_distance11']=data_uuid.groupby(['trip_uuid'])['agg_segment_os
data_uuid['agg_distance_to_destination11']=data_uuid.groupby(['trip_uuid'])['agg_distance
data_uuid['agg_actual_time11']=data_uuid.groupby(['trip_uuid'])['agg_actual_time'].transf
data_uuid['agg_osrm_time11']=data_uuid.groupby(['trip_uuid'])['agg_osrm_time'].transform(
data_uuid['agg_osrm_distance11']=data_uuid.groupby(['trip_uuid'])['agg_osrm_distance'].tr
```

```
data_uuid.head()
```

	route_type	trip_uuid	start_scan_to_end_scan	trip_creation_month	tri
0	Carting	trip-153741093647649320	86.0	9	
5	Carting	trip-153741093647649320	109.0	9	
10	FTL	trip-153768492602129387	302.0	9	
15	Carting	trip-153693976643699843	108.0	9	
17	FTL	trip-153687145942424248	195.0	9	

```
data_uuid['Source_City11']=data_uuid.groupby(['trip_uuid'])['Source_City'].transform('fir
data_uuid['Source_Place11']=data_uuid.groupby(['trip_uuid'])['Source_Place'].transform('f
data_uuid['Source_Code/State11']=data_uuid.groupby(['trip_uuid'])['Source_Code/State'].tr
data_uuid['destination_City11']=data_uuid.groupby(['trip_uuid'])['destination_City'].tran
data_uuid['destination_Place11']=data_uuid.groupby(['trip_uuid'])['destination_Place'].tr
data_uuid['destination_Code/State11']=data_uuid.groupby(['trip_uuid'])['destination_Code/
```

```
data_uuid.head()
```

	route_type	trip_uuid	start_scan_to_end_scan	trip_creation_month	tri
0	Carting	trip-153741093647649320	86.0	9	
5	Carting	trip-153741093647649320	109.0	9	
10	FTL	trip-153768492602129387	302.0	9	
15	Carting	trip-153693976643699843	108.0	9	
17	FTL	trip-153687145942424248	195.0	9	

Creating a new DataFrame for eliminating the duplicates and having only one row detail for one trip which comprises all the details of the trip.

```
data_final=data_uuid.loc[:,['route_type', 'trip_uuid',
       'trip_creation_month', 'trip_creation_year', 'trip_creation_day',
```

```
'start_scan_to_end_scan11', 'Timediff_start_end_H11',
'agg_segment_actual_time11', 'agg_segment_osrm_time11',
'agg_segment_osrm_distance11', 'agg_distance_to_destination11',
'agg_actual_time11', 'agg_osrm_time11', 'agg_osrm_distance11',
'Source_City11', 'Source_Place11', 'Source_Code/State11',
'destination_City11', 'destination_Place11',
'destination_Code/State11']]
```

```
data_final.duplicated().sum()
```

```
11436
```

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

	route_type	trip_uuid	trip_creation_month	trip_creation_year	trip_cre
0	Carting	trip-153741093647649320	9	2018	
5	Carting	trip-153741093647649320	9	2018	

```
data_final.drop_duplicates(inplace=True)
```

```
data_final.duplicated().sum()
```

```
0
```

```
data_final.shape
```

```
(14787, 20)
```

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

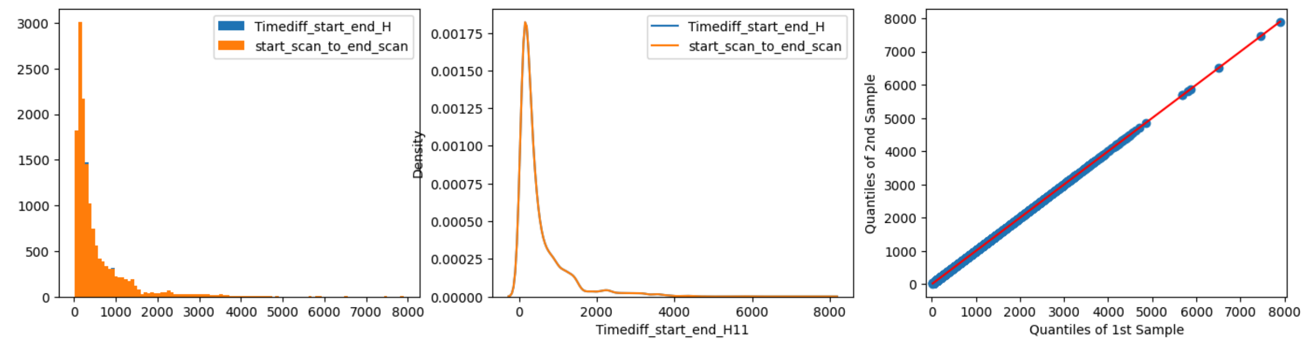
	route_type	trip_uuid	trip_creation_month	trip_creation_year	trip_cre
0	Carting	trip-153741093647649320	9	2018	

Hypothesis/ Visual Analysis

Comparison between Timediff_start_end_H11(od_start_time and od_end_time) and start_scan_to_end_scan

```
import pandas as pd
import numpy as np
from numpy import NaN, nan, NAN
import matplotlib.pyplot as plt
import seaborn as sns
import math, random
from scipy import stats
from statsmodels.stats.weightstats import ztest
from statsmodels.distributions.empirical_distribution import ECDF
from statsmodels.graphics.gofplots import qqplot, qqplot_2samples
import statsmodels.api as sm
import warnings
warnings.filterwarnings("ignore")
```

```
plt.figure(figsize=(17,4))
plt.subplot(131)
plt.hist(data_final['Timediff_start_end_H11'],bins=100,label='Timediff_start_end_H')
plt.hist(data_final['start_scan_to_end_scan11'],bins=100,label='start_scan_to_end_scan')
plt.legend()
plt.subplot(132)
sns.kdeplot(data_final['Timediff_start_end_H11'],label='Timediff_start_end_H')
sns.kdeplot(data_final['start_scan_to_end_scan11'],label='start_scan_to_end_scan')
plt.legend()
# Quantile-Quantile plot for 2samples
qqplot_2samples(data_final['Timediff_start_end_H11'], data_final['start_scan_to_end_scan1
plt.show()
```



Step-1: Defining Null & Alternate Hypothesis

H0 : The mean for Timediff_start_end_H & start_scan_to_end_scan are same

Ha : The mean for start_scan_to_end_scan and start_scan_to_end_scan are difference.

Step-2: Choosing Appropriate test

Here we are using Two Sample T-Test

Step-3: Choosing Significance level

Here we are aiming for 95% confidence, hence alpha=0.05

Step-4: Perform the test and determine the pvalue

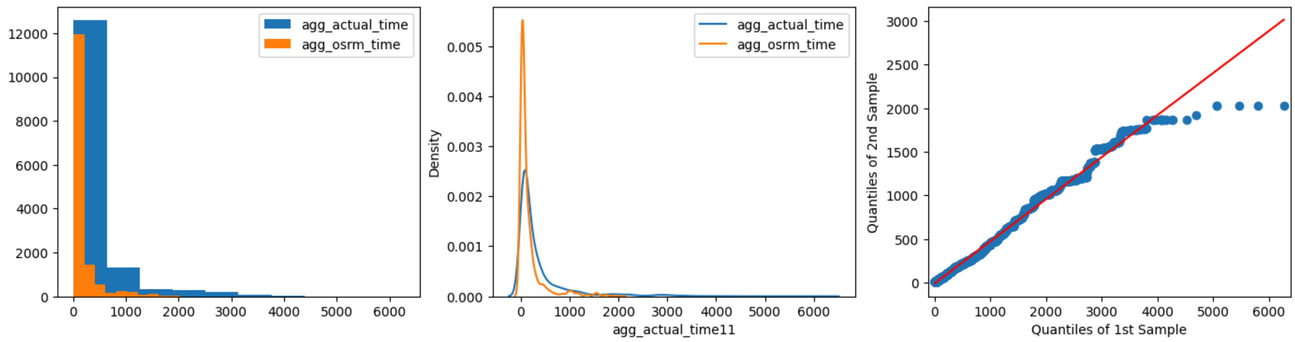
```
import scipy.stats as stats
t_stat,p_value = stats.ttest_ind(data_final['Timediff_start_end_H11'],data_final['start_s
print("t_stat : ",t_stat)
print("p_value : ",p_value)
print('P_value One_side :',(p_value/2))

if p_value < 0.05:
    print('Reject NULL HYPOTHESIS')
else:
    print('Fail to Reject NULL HYPOTHESIS')
```

```
t_stat :  0.11551867533202027
p_value :  0.9080348031420551
P_value One_side : 0.45401740157102755
Fail to Reject NULL HYPOTHESIS
```

The pvalue is not less than alpha, hence the mean between Timediff_start_end_H11 and start_scan_to_end_scan11 are same.

Comparision Between Aggregate Actual time & Aggregate OSRM Time

```
plt.figure(figsize=(17,4))
plt.subplot(131)
plt.hist(data_final['agg_actual_time11'],bins=10,label='agg_actual_time')
plt.hist(data_final['agg_osrm_time11'],bins=10,label='agg_osrm_time')
plt.legend()
plt.subplot(132)
sns.kdeplot(data_final['agg_actual_time11'],label='agg_actual_time')
sns.kdeplot(data_final['agg_osrm_time11'],label='agg_osrm_time')
plt.legend()
# Quantile-Quantile plot for 2samples
qqplot_2samples(data_final['agg_actual_time11'],data_final['agg_osrm_time11'], line="r",
plt.show()
```



Step-1: Defining Null & Alternate Hypothesis

H0 : The mean for agg_actual_time & agg_osrm_time are same

Ha : The mean for agg_actual_time and agg_osrm_time are difference.

Step-2: Choosing Appropriate test

Here we are using Two Sample T-Test

Step-3: Choosing Significance level

Here we are aiming for 95% confidence, hence alpha=0.05

Step-4: Perform the test and determine the pvalue

```
import scipy.stats as stats
t_stat,p_value = stats.ttest_ind(data_final['agg_actual_time11'],data_final['agg_osrm_tim
print('t_stat :', t_stat)
print('P-value :', p_value)
if p_value < 0.05:
    print('Reject NULL HYPOTHESIS')
else:
    print('Fail to Reject NULL HYPOTHESIS')
```

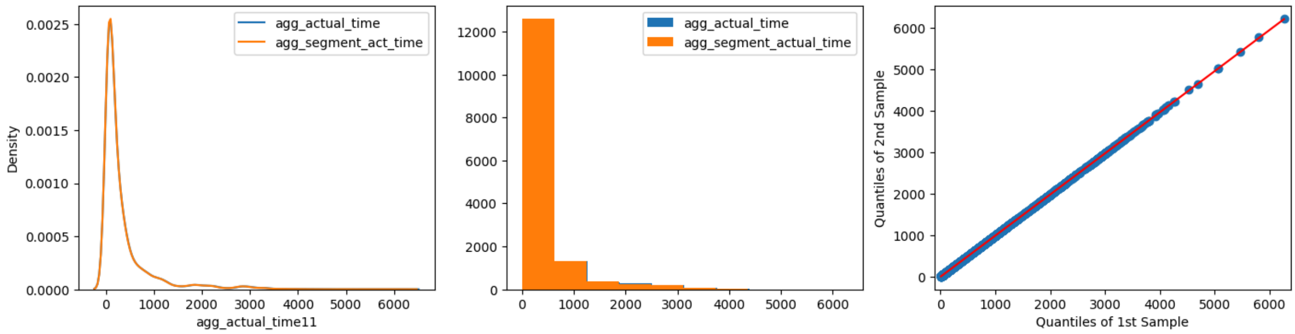
```
t_stat : 37.924611689639825
P-value : 2.358932823082838e-307
Reject NULL HYPOTHESIS
```

The pvalue is less than alpha, hence the mean between agg_actual_time11 and agg_osrm_time11 are not same.

##

Comparision Between Aggregate Actual time & Aggregate segment_actual_time

```
plt.figure(figsize=(17,4))
plt.subplot(131)
sns.kdeplot(data_final['agg_actual_time11'],label='agg_actual_time')
sns.kdeplot(data_final['agg_segment_actual_time11'],label='agg_segment_act_time')
plt.legend()
plt.subplot(132)
plt.hist(data_final['agg_actual_time11'],bins=10,label='agg_actual_time')
plt.hist(data_final['agg_segment_actual_time11'],bins=10,label='agg_segment_actual_time')
plt.legend()
# Quantile-Quantile plot for 2samples
qqplot_2samples(data_final['agg_actual_time11'],data_final['agg_segment_actual_time11'],
plt.show()
```



Step-1: Defining Null & Alternate Hypothesis

H0 : The mean for agg_Actual_time & agg_segment_actual_time are same

Ha : The mean for agg_Actual_time and agg_segment_actual_time are difference.

Step-2: Choosing Appropriate test

Here we are using Two Sample T-Test

Step-3: Choosing Significance level

Here we are aiming for 95% confidence, hence alpha=0.05

Step-4: Perform the test and determine the pvalue

```
t_stat,p_value = stats.ttest_ind(data_final['agg_actual_time11'],data_final['agg_segment_
print('t_stat :', t_stat)
print('P-value :',(p_value))

if p_value < 0.05:
    print('Reject NULL HYPOTHESIS')
else:
    print('Fail to Reject NULL HYPOTHESIS')
```

```
t_stat : 0.4978641813349065
P-value : 0.6185834771383849
Fail to Reject NULL HYPOTHESIS
```

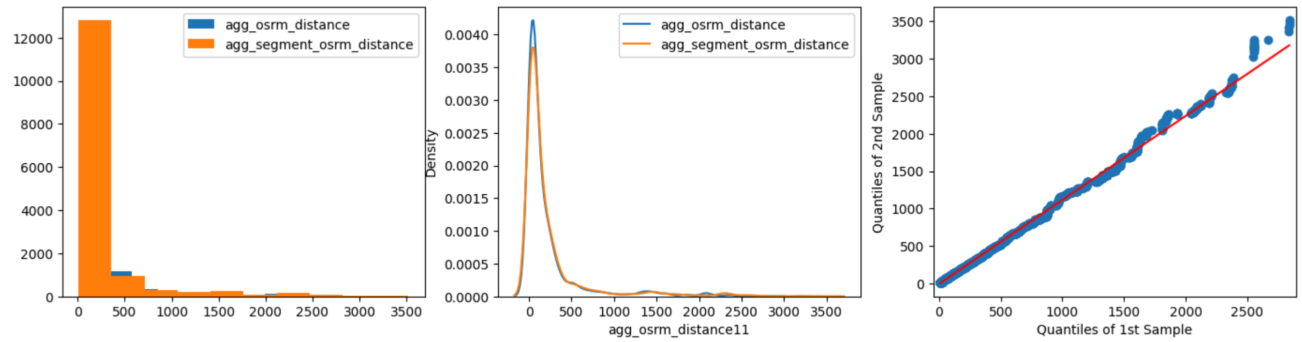
The pvalue is not less than alpha, hence the mean between agg_actual_time11 and agg_segment_actual_time11 are same.

```
##########################################################################################
```

Double-click (or enter) to edit

Comparision Between Aggregate OSRM distance & Aggregate Segment osrm distance

```
plt.figure(figsize=(17,4))
plt.subplot(1,3,1)
plt.hist(data_final['agg_osrm_distance11'],bins=10,label='agg_osrm_distance')
plt.hist(data_final['agg_segment_osrm_distance11'],bins=10,label='agg_segment_osrm_distan
plt.legend()
plt.subplot(1,3,2)
sns.kdeplot(data_final['agg_osrm_distance11'],label='agg_osrm_distance')
sns.kdeplot(data_final['agg_segment_osrm_distance11'],label='agg_segment_osrm_distance')
plt.legend()
# Quantile-Quantile plot for 2samples
qqplot_2samples(data_final['agg_osrm_distance11'],data_final['agg_segment_osrm_distance11
plt.show()
```



Step-1: Defining Null & Alternate Hypothesis

H0 : The mean for Agg_osrm_distance & agg_segment_osrm_distance are same

Ha : The mean for Agg_osrm_distance and agg_segment_osrm_distance are difference.

Step-2: Choosing Appropriate test

Here we are using Two Sample T-Test

Step-3: Choosing Significance level

Here we are aiming for 95% confidence, hence alpha=0.05

Step-4: Perform the test and determine the pvalue

```
import scipy.stats as stats
t_stat,p_value = stats.ttest_ind(data_final['agg_osrm_distance11'],data_final['agg_segmen
print('t_stat :', t_stat)
print('P-value :',(p_value))

if p_value < 0.05:
    print('Reject NULL HYPOTHESIS')
else:
    print('Fail to Reject NULL HYPOTHESIS')
```

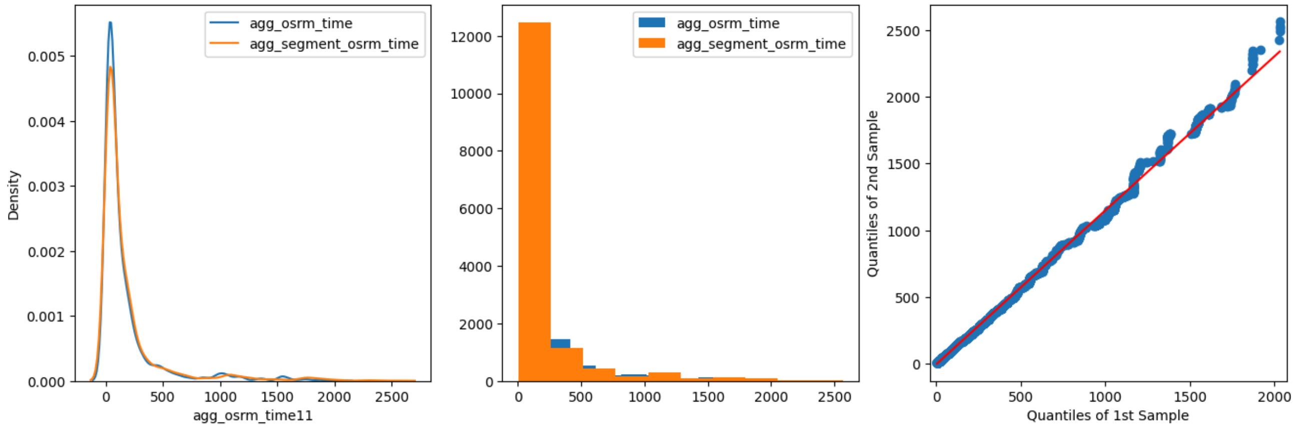
```
t_stat : -3.9379741183399783
P-value : 8.236076174381012e-05
Reject NULL HYPOTHESIS
```

The pvalue is less than alpha, hence the mean between agg_osrm_distance11 and agg_segment_osrm_distance11 are not same.

##

Comparision Between Aggregate OSRM time & Aggregate Segment OSRM Time

```
plt.figure(figsize=(16,5))
plt.subplot(131)
sns.kdeplot(data_final['agg_osrm_time11'],label='agg_osrm_time')
sns.kdeplot(data_final['agg_segment_osrm_time11'],label='agg_segment_osrm_time')
plt.legend()
plt.subplot(132)
plt.hist(data_final['agg_osrm_time11'],bins=10,label='agg_osrm_time')
plt.hist(data_final['agg_segment_osrm_time11'],bins=10,label='agg_segment_osrm_time')
plt.legend()
# Quantile-Quantile plot for 2samples
qqplot_2samples(data_final['agg_osrm_time11'],data_final['agg_segment_osrm_time11'], line
plt.show()
```



Step-1: Defining Null & Alternate Hypothesis

H0 : The mean for Agg_osrm_distance & agg_segment_osrm_distance are same

Ha : The mean for Agg_osrm_distance and agg_segment_osrm_distance are difference.

Step-2: Choosing Appropriate test

Here we are using Two Sample T-Test

Step-3: Choosing Significance level

Here we are aiming for 95% confidence, hence alpha=0.05

Step-4: Perform the test and determine the pvalue

```
import scipy.stats as stats
t_stat,p_value = stats.ttest_ind(data_final['agg_osrm_time11'],data_final['agg_segment_os
print('t_stat :', t_stat)
print('P-value :', p_value)
if p_value < 0.05:
    print('Reject NULL HYPOTHESIS')
else:
    print('Fail to Reject NULL HYPOTHESIS')
```

```
t_stat : -5.505522067054686
P-value : 3.711314386305602e-08
Reject NULL HYPOTHESIS
```

The pvalue is less than alpha, hence the mean between agg_osrm_time11 and agg_segment_osrm_time11 are not same.

```
#######################################################3
```

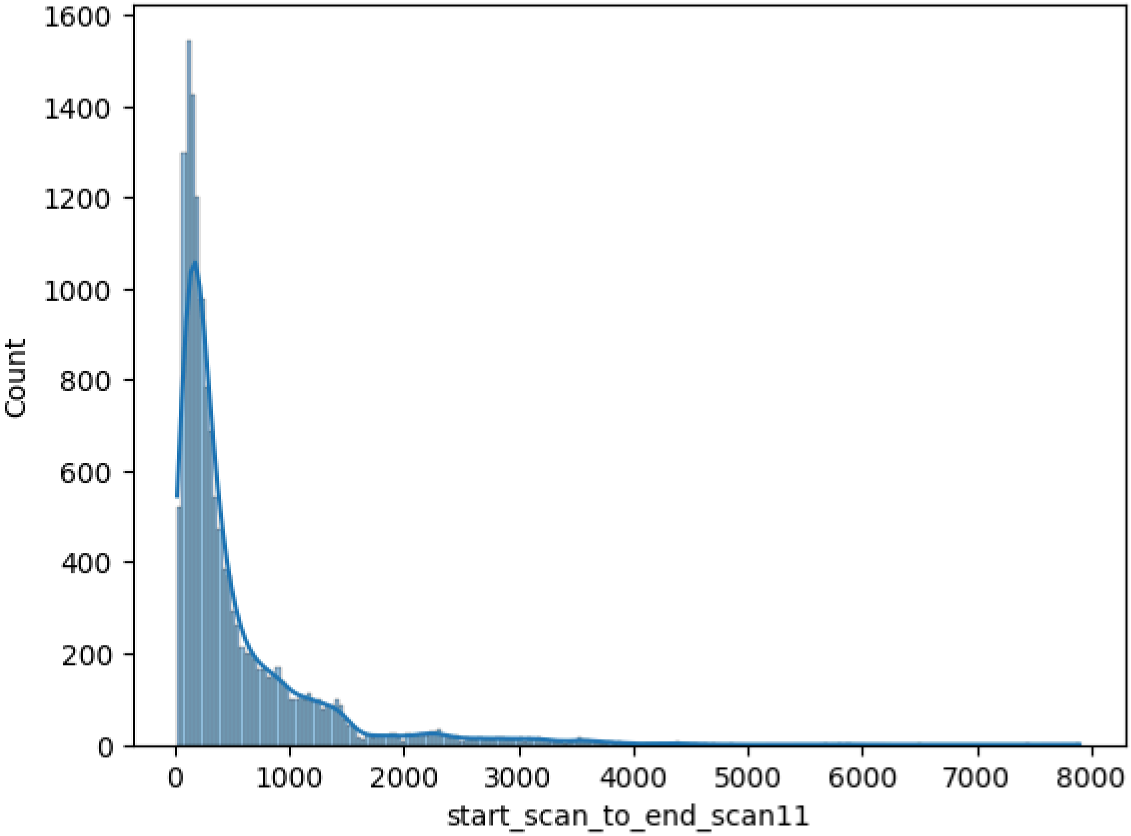
Exploratory Data Analysis

Univariate Data Analysis

```
num_cols = data_final.select_dtypes('float64').columns.values
cat_cols = data_final.select_dtypes('object').columns.values
```

```
for i in num_cols:
    print('##############')
    print(data_final[i].value_counts())
    sns.histplot(data_final[i],kde=True)
    plt.show()
```

```
##############
start_scan_to_end_scan11
148.0      51
115.0      51
87.0       50
113.0      49
128.0      49
           ..
1895.0      1
1634.0      1
1199.0      1
1205.0      1
2429.0      1
Name: count, Length: 2203, dtype: int64
```




```
##############
Timediff_start_end_H11
319.61      4
286.63      4
122.43      4
147.10      4
86.20       4
           ..
227.87      1
924.06      1
658.28      1
3732.37     1
427.69      1
Name: count, Length: 13573, dtype: int64
```

- The data's are heavily right skewed.

```
for i in cat_cols:
    print('##############')
    print(data_final[i].value_counts())
```

```
##############
route_type
Carting    8906
FTL        5881
Name: count, dtype: int64
##############
trip_uuid
trip-153741093647649320    1
trip-153836648611826977    1
trip-153681920064110379    1
trip-153744931166370622    1
trip-153764628243892763    1
                          ..
trip-153741177166786003    1
trip-153801210039247977    1
trip-153737819969505360    1
trip-153739632610417618    1
trip-153746066843555182    1
Name: count, Length: 14787, dtype: int64
##############
Source_City11
Bengaluru               1014
Gurgaon                 1011
Bhiwandi                 811
Bangalore                731
Delhi                    617
                        ...
Parvathipuram_Central      1
Koraput                    1
Jasai                      1
Baripada                   1
Ashta                      1
Name: count, Length: 706, dtype: int64
##############
Source_Place11
Bilaspur           959
Mankoli            811
Nelmngla           732
H                  643
I                  571
                  ...
Ymunpurm             1
Shahdara (Delhi)     1
KalikDPP             1
PuranDPP             1
ShantiNg             1
Name: count, Length: 672, dtype: int64
##############
Source_Code/State11
HB (Haryana)         937
HB (Maharashtra)     811
HB (Karnataka)       757
H (Karnataka)        751
H (Punjab)           370
                    ...
```

```
2 (Andhra Pradesh)      1
1 (Andhra Pradesh)      1
2 (Karnataka)           1
```

Busiest Route

```
1185.0      1
```

```
data_copy_grouped.head()
```

	trip_uuid	source_center	destination_center	data	trip_creation_time
0	trip-153671041653548748	IND209304AAA	IND000000ACB	18	18
1	trip-153671041653548748	IND462022AAA	IND209304AAA	21	21
2	trip-153671042288605164	IND561203AAB	IND562101AAA	3	3
3	trip-153671042288605164	IND572101AAA	IND561203AAB	6	6
4	trip-153671043369099517	IND000000ACB	IND160002AAC	12	12

```
data_copy_grouped.route_type.max()
```

```
81
```

```
# find trip uuid of max count
data_copy_grouped[data_copy_grouped['route_type']==81]
```

	trip_uuid	source_center	destination_center	data	trip_creation_ti
12201	trip-153755502932196495	IND160002AAC	IND562132AAA	81	

```
120.2104    1
```

```
data[data['trip_uuid']=='trip-153755502932196495']
```

	data	trip_creation_time	route_schedule_uuid	route_type	trip_
61008	training	2018-09-21 18:37:09.322207	thanos::sroute:4029a8a2-6c74-4b7e-a6d8-f9e069f...	FTL	15375550293219
61009	training	2018-09-21 18:37:09.322207	thanos::sroute:4029a8a2-6c74-4b7e-a6d8-f9e069f...	FTL	15375550293219
61010	training	2018-09-21 18:37:09.322207	thanos::sroute:4029a8a2-6c74-4b7e-a6d8-f9e069f...	FTL	15375550293219
61011	training	2018-09-21 18:37:09.322207	thanos::sroute:4029a8a2-6c74-4b7e-a6d8-f9e069f...	FTL	15375550293219
61012	training	2018-09-21 18:37:09.322207	thanos::sroute:4029a8a2-6c74-4b7e-a6d8-f9e069f...	FTL	15375550293219
...	...	...	...	...	
61084	training	2018-09-21 18:37:09.322207	thanos::sroute:4029a8a2-6c74-4b7e-a6d8-f9e069f...	FTL	15375550293219
61085	training	2018-09-21 18:37:09.322207	thanos::sroute:4029a8a2-6c74-4b7e-a6d8-f9e069f...	FTL	15375550293219
61086	training	2018-09-21 18:37:09.322207	thanos::sroute:4029a8a2-6c74-4b7e-a6d8-f9e069f...	FTL	15375550293219
61087	training	2018-09-21 18:37:09.322207	thanos::sroute:4029a8a2-6c74-4b7e-a6d8-f9e069f...	FTL	15375550293219
61088	training	2018-09-21 18:37:09.322207	thanos::sroute:4029a8a2-6c74-4b7e-a6d8-f9e069f...	FTL	15375550293219

81 rows × 24 columns

```
data_final[data_final['trip_uuid']=='trip-153755502932196495'][['agg_segment_actual_time1
    'agg_segment_osrm_time11', 'agg_segment_osrm_distance11',
    'agg_distance_to_destination11', 'agg_actual_time11', 'agg_osrm_time11']]
```

	agg_segment_actual_time11	agg_segment_osrm_time11	agg_segment_osrm_distance
61008	3751.0	1864.0	2500.21

60.0 134

Bussiest Route is from source Chandigarh_Mehmdpur_H (Punjab) to Bangalore_Nelmngla_H (Karnataka) Average_distance between them is

1927 kms & average time taken is 3784 mins

```
2331.0      1
```

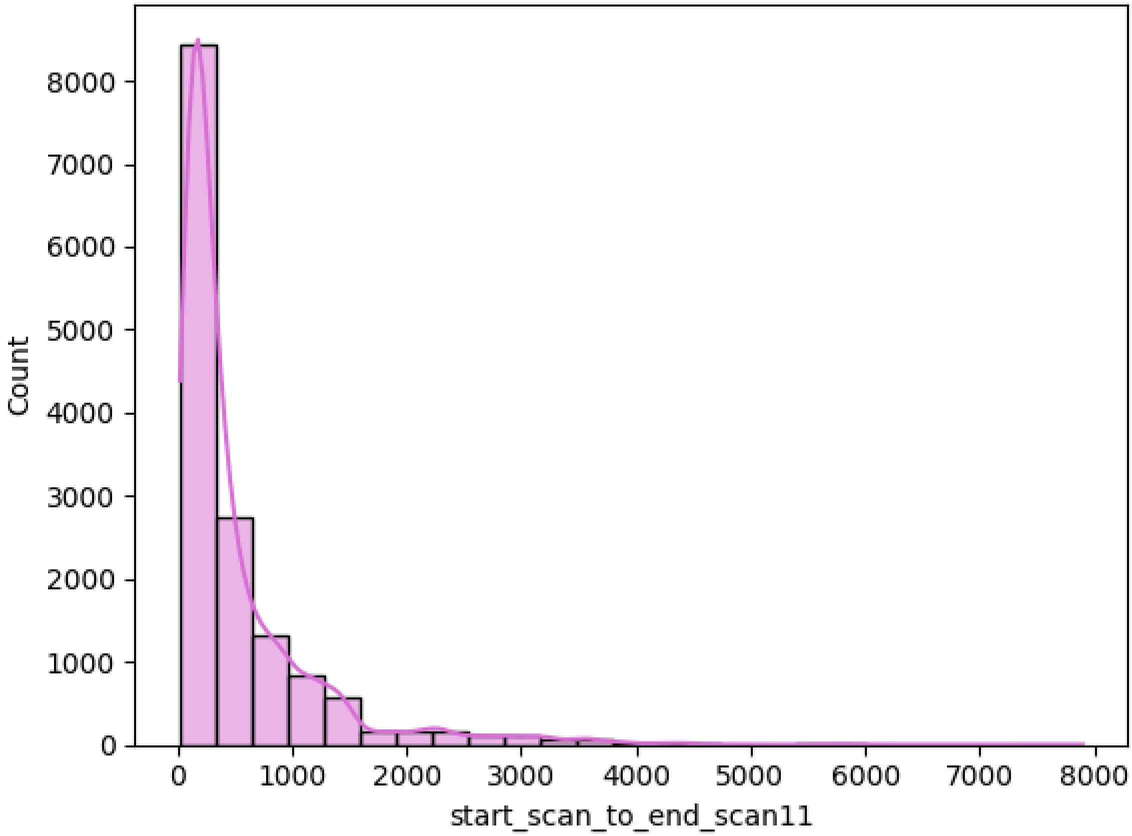
```
temp=['start_scan_to_end_scan11',
       'trip_creation_day', 'Timediff_start_end_H11', 'agg_segment_actual_time11',
       'agg_segment_osrm_time11', 'agg_segment_osrm_distance11',
       'agg_distance_to_destination11', 'agg_actual_time11', 'agg_osrm_time11',
       'agg_osrm_distance11']
```

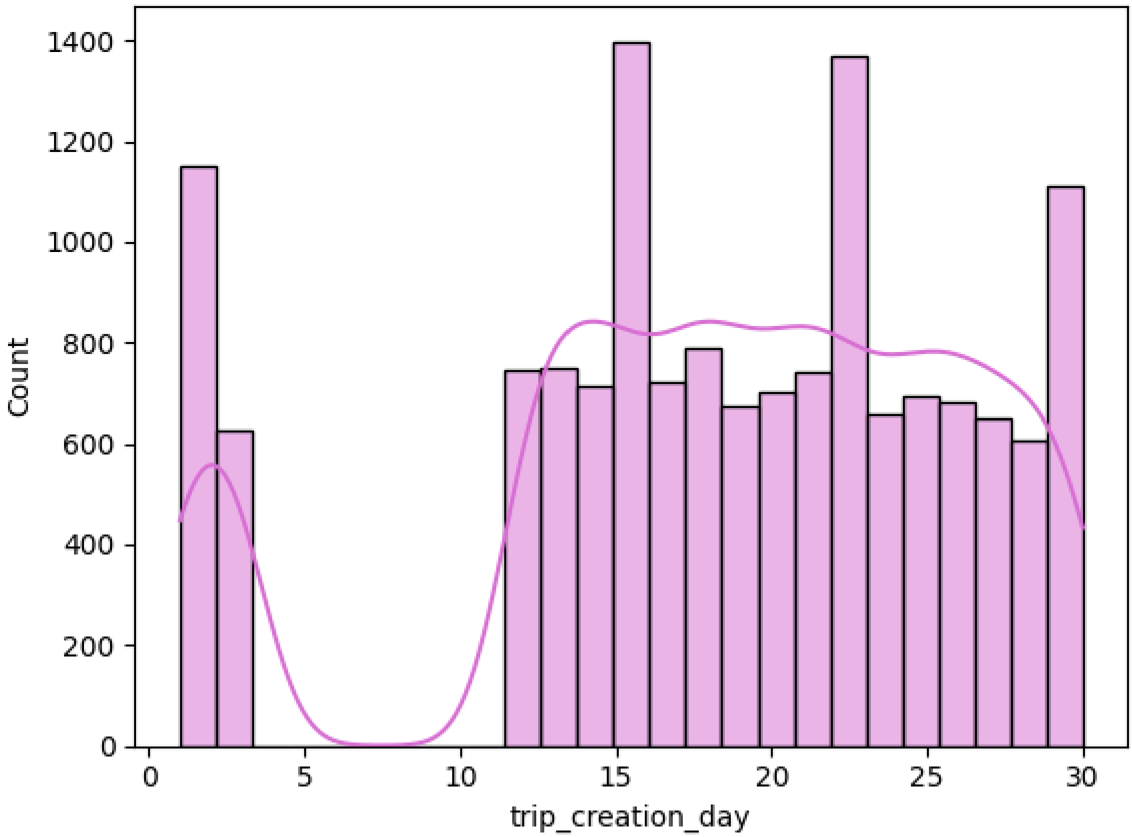
```
temp
```

```
['start_scan_to_end_scan11',
 'trip_creation_day',
 'Timediff_start_end_H11',
 'agg_segment_actual_time11',
 'agg_segment_osrm_time11',
 'agg_segment_osrm_distance11',
 'agg_distance_to_destination11',
 'agg_actual_time11',
 'agg_osrm_time11',
 'agg_osrm_distance11']
```

Data Visualization

```
for i in temp:
  sns.histplot(data_final[i], bins=25, kde=True, color='orchid')
  plt.show()
```


Count
0
1000
2000
3000
4000
5000
6000
7000
8000
start_scan_to_end_scan11


Count
0
200
400
600
800
1000
1200
1400
5
10
15
20
25
30
trip_creation_day

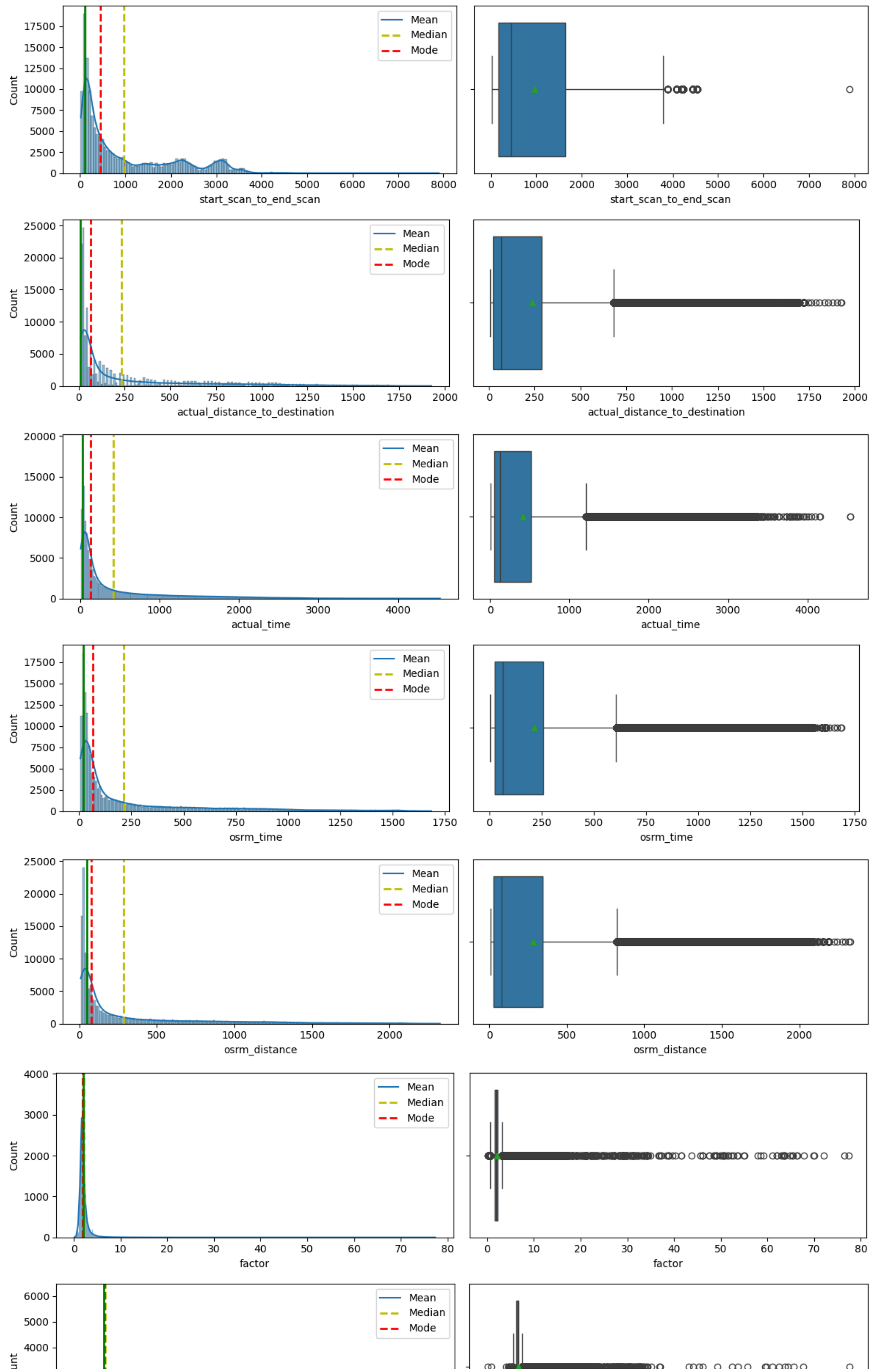

8000
7000

Outlier Detection & their Treatment

```
def uni(d):
    f,ax = plt.subplots(nrows=1,ncols=2,figsize=(12,3))
    sns.histplot(d, kde=True, ax=ax[0])
    ax[0].axvline(d.mean(), color='y', linestyle='--',linewidth=2)
    ax[0].axvline(d.median(), color='r', linestyle='dashed', linewidth=2)
    ax[0].axvline(d.mode()[0],color='g',linestyle='solid',linewidth=2)
    ax[0].legend({'Mean':d.mean(),'Median':d.median(),'Mode':d.mode()})

    sns.boxplot(x=d, showmeans=True, ax=ax[1])
    plt.tight_layout()
```

```
num_cols = data.select_dtypes('float64').columns.values
```

```
for f in num_cols:
    uni(data[f])
plt.show()
```



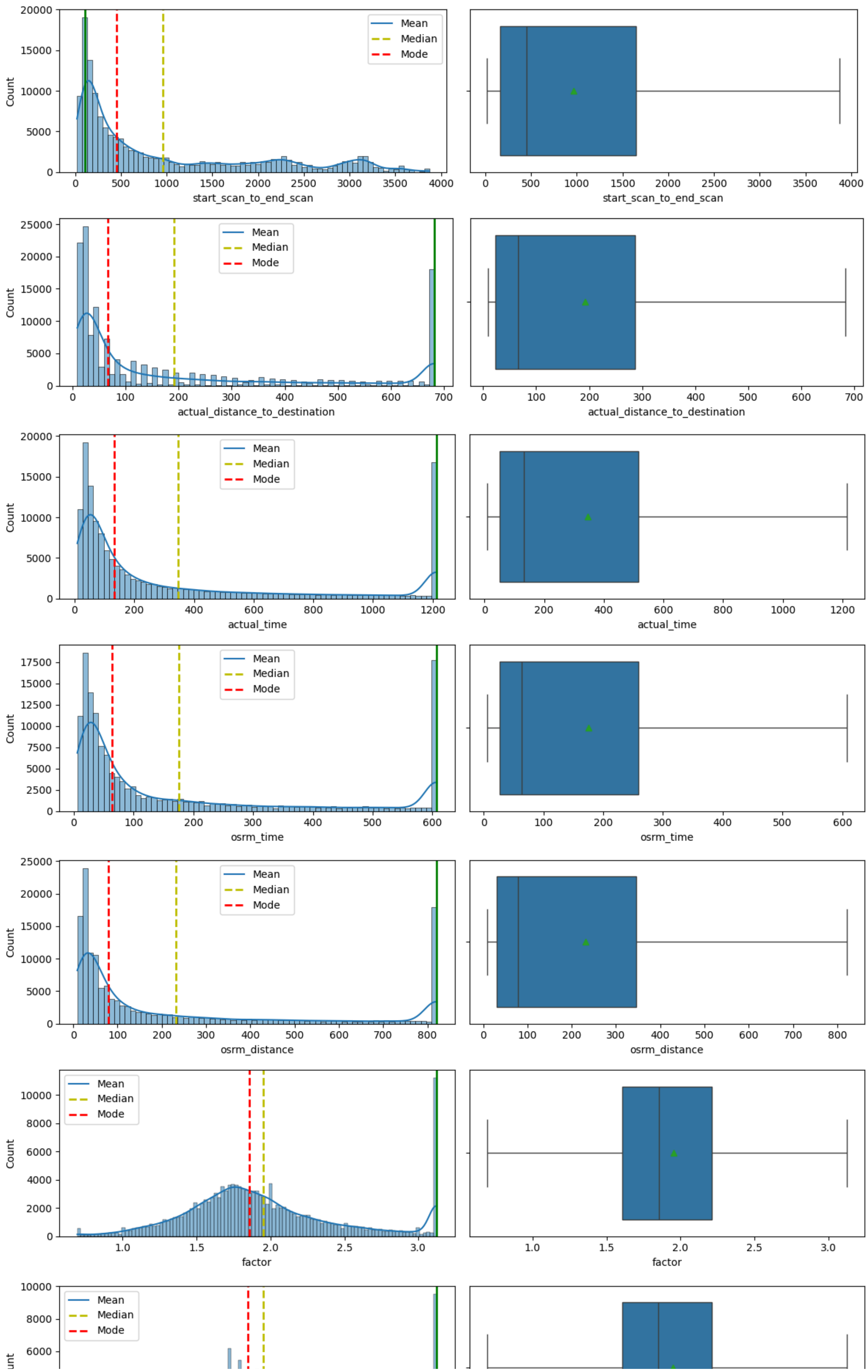

```
#treating outliers:
def treat_outlier(variable):
```

```
    #Takes two parameters: dataframe & variable of interest as string
    q1,q3=np.percentile(variable,[25,75])
    iqr = q3-q1
    lo_range = q1-(1.5*iqr)
    up_range = q3+(1.5*iqr)
    return lo_range,up_range
```

```
for col in num_cols:
    ir,ur=treat_outlier(data[col])
    data[col]=np.where(data[col]>ur,ur,data[col])
    data[col]=np.where(data[col]<ir,ir,data[col])
```

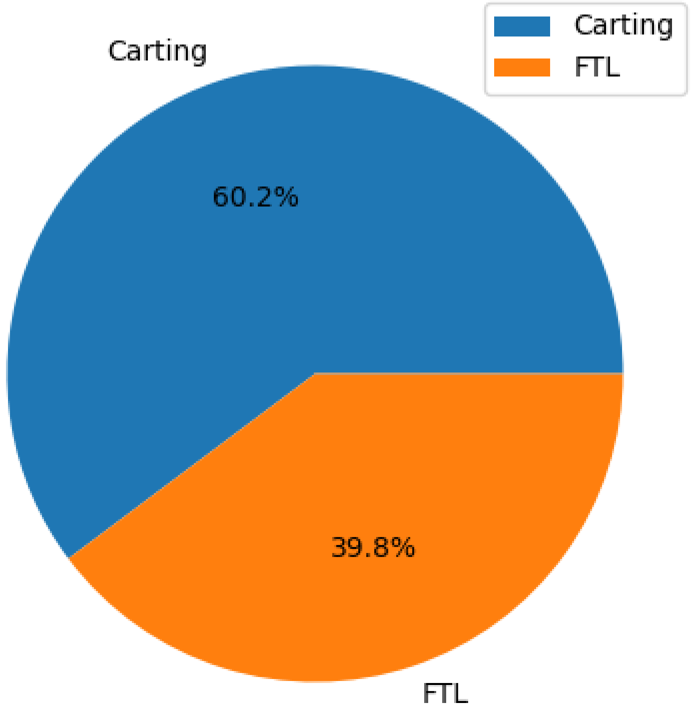
Here I have found the outliers and replaced them with max and min of whisker's value.

```
#Lets check where outliers are removed or not:
for f in num_cols:
    uni(data[f])
plt.show()
```



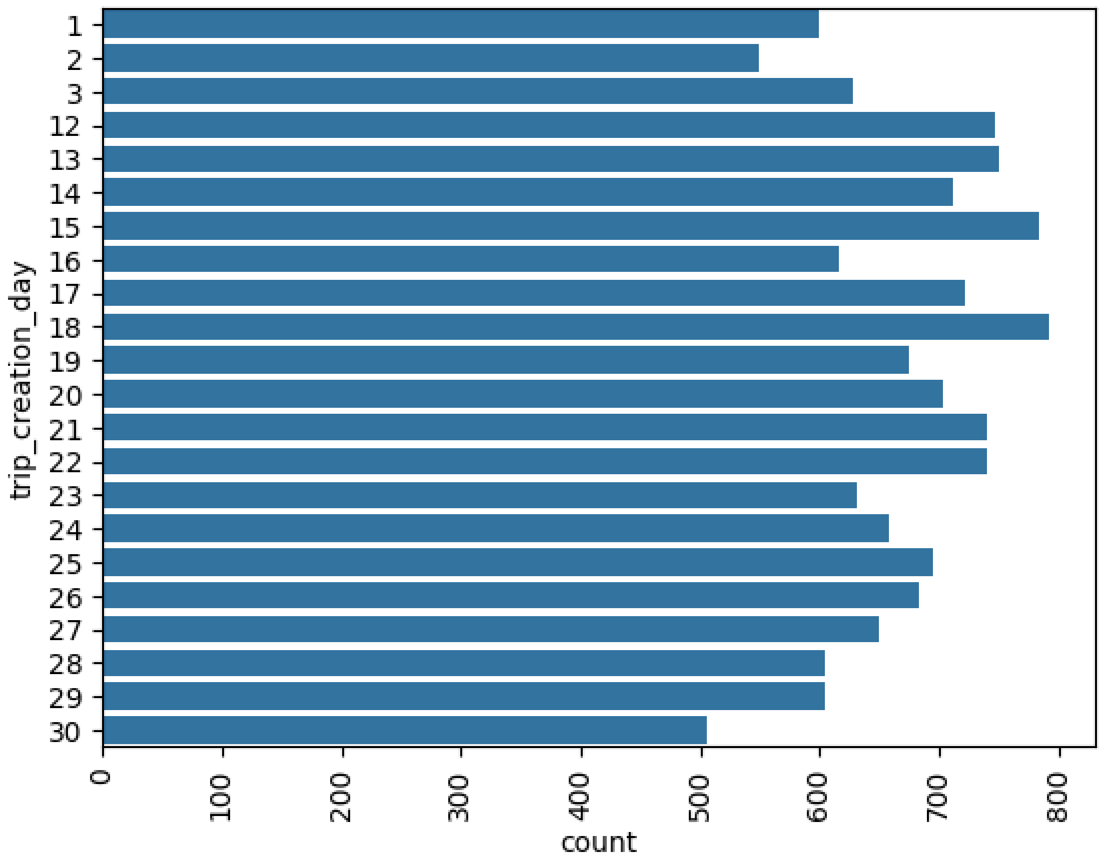
- **The outliers are removed**

```
fig1, ax1 = plt.subplots(figsize=(10,5))
ax1.pie(data_final['route_type'].value_counts(), labels=data_final['route_type'].unique()
plt.legend()
plt.show()
```



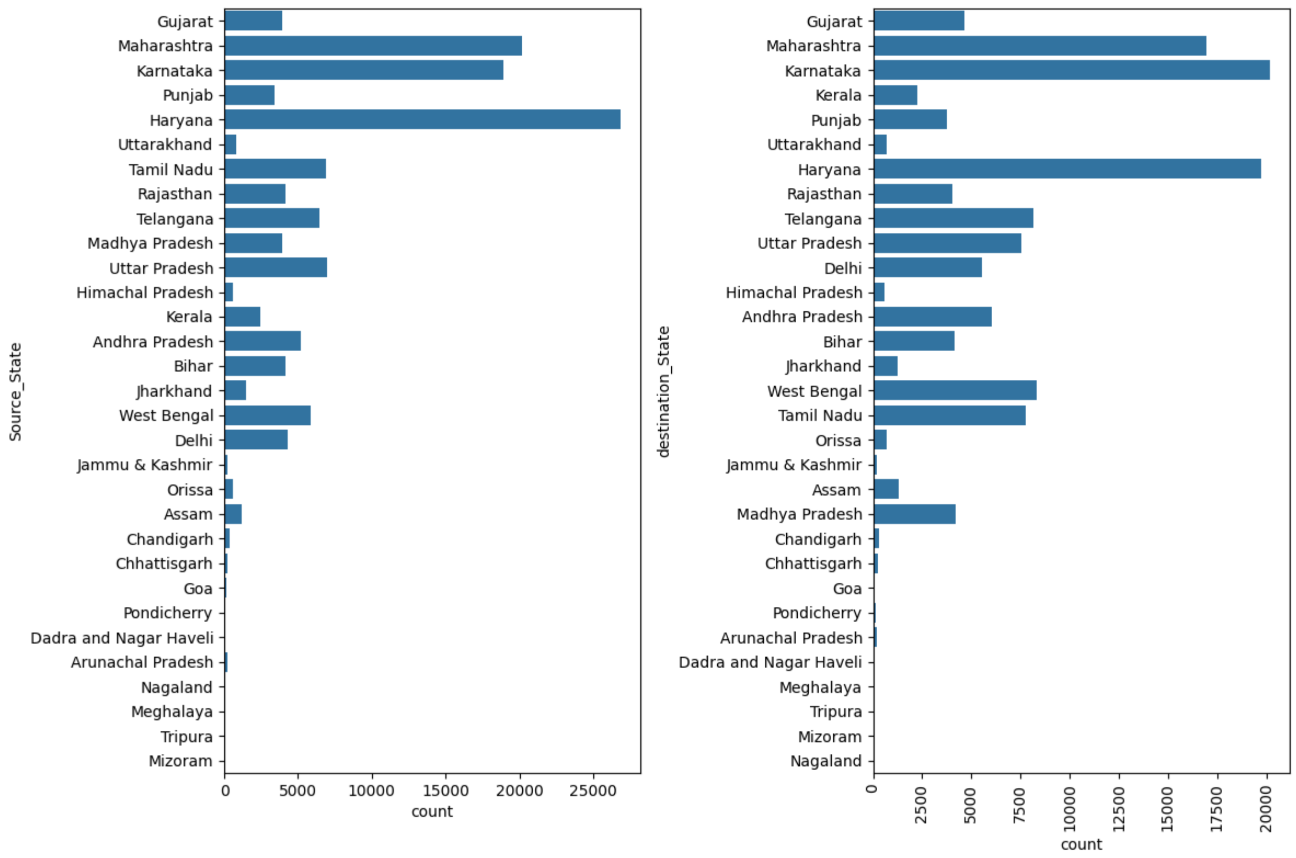
- **Therefore by analyzing the given data, it has been found that 60% of the route type used for delivery were Carting and the remaining were FTL.**

```
sns.countplot(y=data_final['trip_creation_day'])
plt.xticks(rotation=90)
plt.show()
```



- The start and end date of the months the trips were lesser.
- More trips were during mid of the month, but there is not huge diiference. The trips were similar across the month.
- No trips were found from 4th till 11th of the month.

```
f,ax = plt.subplots(nrows=1,ncols=2,figsize=(12,8))
sns.countplot(y=data_copy['Source_State'],ax=ax[0])
sns.countplot(y=data_copy['destination_State'],ax=ax[1])

plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
```



- The top 3 states that contributes to overall trips were Haryana, Maharastra and Karnataka.

Bivariate Analysis

Since most of the features were numerical, instead of using other plots I have used heatmap to know the overall relation between each features

```
rel=data_final.loc[:,['route_type', 'trip_uuid',
       'start_scan_to_end_scan11', 'trip_creation_month',
       'trip_creation_day', 'Timediff_start_end_H11', 'agg_segment_actual_time11',
       'agg_segment_osrm_time11', 'agg_segment_osrm_distance11',
       'agg_distance_to_destination11', 'agg_actual_time11', 'agg_osrm_time11',
       'agg_osrm_distance11', 'Source_City11', 'Source_Place11', 'Source_Code/State11',
       'destination_City11', 'destination_Place11', 'destination_Code/State11']]
```

```
rel.columns
```

```
Index(['route_type', 'trip_uuid', 'start_scan_to_end_scan11',
       'trip_creation_month', 'trip_creation_day', 'Timediff_start_end_H11',
       'agg_segment_actual_time11', 'agg_segment_osrm_time11',
       'agg_segment_osrm_distance11', 'agg_distance_to_destination11',
       'agg_actual_time11', 'agg_osrm_time11', 'agg_osrm_distance11',
       'Source_City11', 'Source_Place11', 'Source_Code/State11',
       'destination_City11', 'destination_Place11',
       'destination_Code/State11'],
      dtype='object')
```