```
# importing libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import LabelEncoder, OneHotEncoder, StandardScaler
!pip install category_encoders
from category_encoders import TargetEncoder
from scipy.stats import norm, ttest_ind, chi2_contingency, f_oneway
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/pub
Requirement already satisfied: category_encoders in /usr/local/lib/python3.9/dist-packag
Requirement already satisfied: scikit-learn>=0.20.0 in /usr/local/lib/python3.9/dist-pac
Requirement already satisfied: statsmodels>=0.9.0 in /usr/local/lib/python3.9/dist-packa
Requirement already satisfied: numpy>=1.14.0 in /usr/local/lib/python3.9/dist-packages (
Requirement already satisfied: pandas>=1.0.5 in /usr/local/lib/python3.9/dist-packages (
Requirement already satisfied: scipy>=1.0.0 in /usr/local/lib/python3.9/dist-packages (f
Requirement already satisfied: patsy>=0.5.1 in /usr/local/lib/python3.9/dist-packages (f
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.9/dist-p
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.9/dist-packages (f
Requirement already satisfied: six in /usr/local/lib/python3.9/dist-packages (from patsy
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.9/dist-pac
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.9/dist-packages (
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.9/dist-packages
```

```
# data download
!gdown https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/551/original/delhiv
```

```
Downloading...
From: https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/551/original/de
To: /content/delhivery_data.csv?1642751181
100% 55.6M/55.6M [00:00<00:00, 117MB/s]
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mour
```

```
# reading the dataframe
df = pd.read_csv('/content/delhivery_data.csv?1642751181')
df.head()
```

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

5 rows × 24 columns

```
# fetching information about the data
df.info()
```

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

```
dtypes: bool(1), float64(10), int64(1), object(12)
memory usage: 25.6+ MB
```

```
# finding the columns with nullvalues
df.isna().sum()
```

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

```
# fetching the stats of numerical data
df.describe()
```

	start_scan_to_end_scan	cutoff_factor	actual_distance_to_destination	actual_t
count	144867.000000	144867.000000	144867.000000	144867.000
mean	961.262986	232.926567	234.073372	416.927
std	1037.012769	344.755577	344.990009	598.103
min	20.000000	9.000000	9.000045	9.000
25%	161.000000	22.000000	23.355874	51.000
50%	449.000000	66.000000	66.126571	132.000
75%	1634.000000	286.000000	286.708875	513.000
max	7898.000000	1927.000000	1927.447705	4532.000

```
# fetching stats of every single column present
df.describe(include = 'all')
```

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uui
count	144867	144867	144867	144867	14486
unique	2	14817	1504	2	148
top	training	2018-09-28 05:23:15.359220	thanos::sroute:4029a8a2-6c74-4b7e-a6d8-f9e069f...	FTL	tri 153811219535896
freq	104858	101	1812	99660	1
mean	NaN	NaN	NaN	NaN	Na
std	NaN	NaN	NaN	NaN	Na
min	NaN	NaN	NaN	NaN	Na
25%	NaN	NaN	NaN	NaN	Na
50%	NaN	NaN	NaN	NaN	Na
75%	NaN	NaN	NaN	NaN	Na
max	NaN	NaN	NaN	NaN	Na

11 rows × 24 columns

```
# total no of missing values
df.shape
```

```
(144867, 24)
```

```
# removing unknown and unwanted columns
df.drop(columns = ['is_cutoff', 'cutoff_factor', 'cutoff_timestamp', 'factor', 'segment_facto
```

```
df.shape
```

```
(144867, 18)
```

```
# dropping of the null values
df.dropna(inplace = True)
```

```
df.duplicated().sum()
```

```
0
```

1. The shape of the dataset is (144867,24) with 0 duplicate values
2. There were few unknown columns and missing values in source_name and destination_name column which were removed

```
df.info()
```

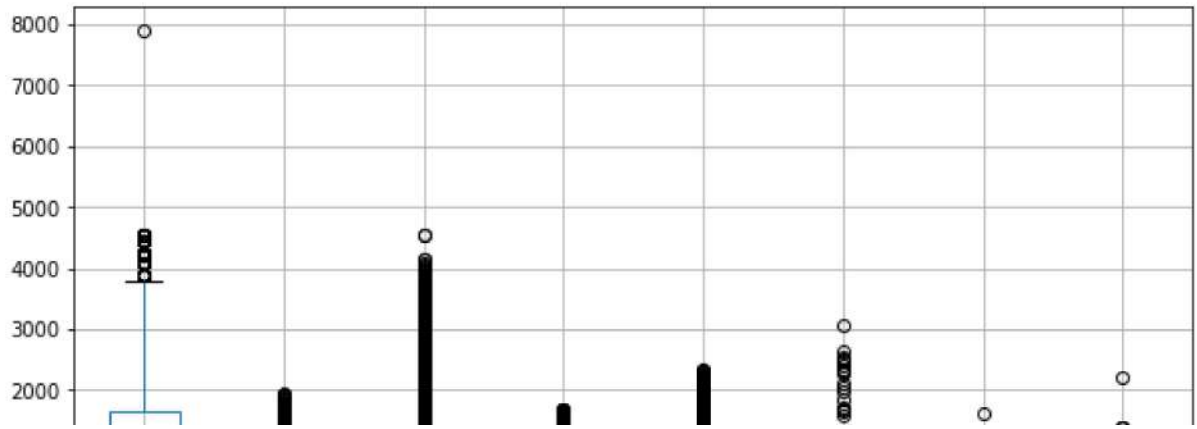
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 144316 entries, 0 to 144866
Data columns (total 18 columns):
 #   Column                          Non-Null Count   Dtype  
---  ------                          --------------   -----  
 0   trip_creation_time              144316 non-null  object 
 1   route_schedule_uuid             144316 non-null  object 
 2   route_type                      144316 non-null  object 
 3   trip_uuid                       144316 non-null  object 
 4   source_center                   144316 non-null  object 
 5   source_name                     144316 non-null  object 
 6   destination_center              144316 non-null  object 
 7   destination_name                144316 non-null  object 
 8   od_start_time                   144316 non-null  object 
 9   od_end_time                     144316 non-null  object 
 10  start_scan_to_end_scan          144316 non-null  float64
 11  actual_distance_to_destination  144316 non-null  float64
 12  actual_time                     144316 non-null  float64
 13  osrm_time                       144316 non-null  float64
 14  osrm_distance                   144316 non-null  float64
 15  segment_actual_time             144316 non-null  float64
 16  segment_osrm_time               144316 non-null  float64
 17  segment_osrm_distance           144316 non-null  float64
dtypes: float64(8), object(10)
memory usage: 20.9+ MB
```

Outlier Analysis in numerical column

```
numcol = ['start_scan_to_end_scan', 'actual_distance_to_destination', 'actual_time', 'osrm_ti
```

```
#outliers in numerical columns
df[numcol].boxplot(rot = 25, figsize = (10,5))
```

```
<Axes: >
```




```
#handling outlier
Q1 = df[numcol].quantile(0.25)
Q3 = df[numcol].quantile(0.75)

IQR = Q3 - Q1

df = df[~((df[numcol] < (Q1 - 1.5 * IQR)) | (df[numcol] > (Q3 + 1.5 * IQR))).any(axis=1)]
df = df.reset_index(drop=True)
```

Feature Engineering

```
df['destination_name'].value_counts()
```

```
Gurgaon_Bilaspur_HB (Haryana)            9601
Bangalore_Nelmngla_H (Karnataka)         6403
Hyderabad_Shamshbd_H (Telangana)         3881
Bhiwandi_Mankoli_HB (Maharashtra)        3747
Pune_Tathawde_H (Maharashtra)            2799
                                         ...
Kangra_Central_D_2 (Himachal Pradesh)       1
Paralakhemundi_Ward14_D (Orissa)            1
Falna_SbhRDDPP_D (Rajasthan)                1
Berhampur_Chatrpr_DC (Orissa)               1
Ranaghat_ArickDPP_D (West Bengal)           1
Name: destination_name, Length: 1450, dtype: int64
```

```
# creating columns from source centre & destination centre
# df = pd.read_csv('/content/delhivery_data.csv?1642751181')
df[['source_city','source_place','source_code/state']]= df['source_name'].str.split('_',2, ex
df[['destination_city','destination_place','destination_code/state']]= df['destination_name']
df[['source_code','source_state']]= df['source_code/state'].str.split('(',2, expand = True)
df['source_state'] = df['source_state'].str.strip(')')
df[['destination_code','destination_state']]= df['destination_code/state'].str.rsplit('(',2,
df['destination_state'] = df['destination_state'].str.strip(')')
```

```
df.head()
```

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

5 rows × 34 columns

```
# changing to datetime format
df[["od_start_time", "od_end_time",'trip_creation_time']] = df[["od_start_time", "od_end_time
```

```
# extracting day, month & year
df['trip_creation_month']=df['trip_creation_time'].dt.month
df['trip_creation_year']=df['trip_creation_time'].dt.year
df['trip_creation_day']=df['trip_creation_time'].dt.day
```

```
# grouping
df2= df.groupby(['trip_uuid','source_center','destination_center']).count().reset_index()
```

```
df2.head()
```

	trip_uuid	source_center	destination_center	data	trip_creation_time	rou
0	trip-153671041653548748	IND209304AAA	IND000000ACB	18	18	
1	trip-153671041653548748	IND462022AAA	IND209304AAA	21	21	
	trip-					

```
# difference between od_start & od_end time
df['timediff']= (df['od_end_time']- df['od_start_time']).dt.total_seconds() / (60)
```

3	153671042288605164	IND572101AAA	IND561203AAB	6	6	

```
df['segment_actual_time_cs']= df.groupby(['trip_uuid','source_center','destination_center'])[
df['segment_osrm_time_cs']= df.groupby(['trip_uuid','source_center','destination_center'])['s
df['segment_osrm_distance_cs']= df.groupby(['trip_uuid','source_center','destination_center']
df['distance_to_destination']= df.groupby(['trip_uuid','source_center','destination_center'])
df['actual_time_1']= df.groupby(['trip_uuid','source_center','destination_center'])['actual_t
df['osrm_time_1']= df.groupby(['trip_uuid','source_center','destination_center'])['osrm_time'
df['osrm_distance_1']= df.groupby(['trip_uuid','source_center','destination_center'])['osrm_d


# dropping unwanted columns
df.drop(columns=['od_end_time','od_start_time','trip_creation_time','source_name','destinatio


df['timediff'].value_counts()

     3802.106157    81
     3259.517945    79
     3159.627845    79
     3280.512927    79
     3466.647781    79
                    ..
     538.728073      1
     52.401932       1
     402.265978      1
     108.016251      1
     767.841807      1
     Name: timediff, Length: 26369, dtype: int64


df.shape

     (144867, 40)


# checking for duplicates
df.duplicated().sum()

     0


# merging
```

```
df_merge= df.loc[:,['route_type', 'trip_uuid',
       'start_scan_to_end_scan',  'trip_creation_month',
       'trip_creation_year', 'trip_creation_day', 'timediff', 'segment_actual_time_cs',
       'segment_osrm_time_cs', 'segment_osrm_distance_cs',
       'source_city', 'source_place', 'source_code', 'source_state', 'osrm_distance_1', 'actu
       'destination_city', 'destination_place', 'destination_code', 'destination_state', 'dis


#checking for duplicate'
df_merge.duplicated().sum()
```

```
0
```

```
# droping duplicates

df_merge.drop_duplicates(inplace = True)


df_merge.duplicated().sum()
```

```
0
```

```
# aggregating based on their trip_uuid

df_merge['start_scan_to_end_scan_1']= df_merge.groupby(['trip_uuid'])['start_scan_to_end_scan
df_merge['timediff_1']= df_merge.groupby(['trip_uuid'])['timediff'].transform('sum')
df_merge['segment_actual_time_1']= df_merge.groupby(['trip_uuid'])['segment_actual_time_cs'].
df_merge['segment_osrm_time_1']= df_merge.groupby(['trip_uuid'])['segment_osrm_time_cs'].tran

df_merge['segment_osrm_distance_1']= df_merge.groupby(['trip_uuid'])['segment_osrm_distance_c
df_merge['distance_to_destination_1']= df_merge.groupby(['trip_uuid'])['distance_to_destinati
df_merge['actual_time11']= df_merge.groupby(['trip_uuid'])['actual_time_1'].transform('sum')
df_merge['osrm_time11']= df_merge.groupby(['trip_uuid'])['osrm_time_1'].transform('sum')
df_merge['osrm_distance11']= df_merge.groupby(['trip_uuid'])['osrm_distance_1'].transform('su

df_merge['source_city11']= df_merge.groupby(['trip_uuid'])['source_city'].transform('first')
df_merge['source_place11']= df_merge.groupby(['trip_uuid'])['source_place'].transform('first'
df_merge['source_code11']= df_merge.groupby(['trip_uuid'])['source_code'].transform('first')
df_merge['source_state11']= df_merge.groupby(['trip_uuid'])['source_state'].transform('first'
df_merge['destination_city11']= df_merge.groupby(['trip_uuid'])['destination_city'].transform
df_merge['destination_place11']= df_merge.groupby(['trip_uuid'])['destination_place'].transfo
df_merge['destination_code11']= df_merge.groupby(['trip_uuid'])['destination_code'].transform
df_merge['destination_state11']= df_merge.groupby(['trip_uuid'])['destination_state'].transfo


df2= df_merge.loc[:,['route_type', 'trip_uuid',
        'trip_creation_month', 'trip_creation_year', 'trip_creation_day',
       'start_scan_to_end_scan_1', 'timediff_1',
       'segment_actual_time_1', 'segment_osrm_time_1',
       'segment_osrm_distance_1', 'distance_to_destination_1',
       'actual_time11', 'osrm_time11', 'osrm_distance11',
```

```
        'source_city11', 'source_place11', 'source_code11', 'source_state11',
        'destination_city11', 'destination_place11',
        'destination_code11', 'destination_state11']]
```

```
# checking and droping duplicate'
df2.duplicated().sum()
df2.drop_duplicates(inplace = True)
```

```
df2.shape
```

```
(14817, 22)
```

```
df2.reset_index(inplace = True)
df2.head()
```

	index	route_type	trip_uuid	trip_creation_month	trip_creation_year	trip
0	0	Carting	trip-153741093647649320	9	2018	
1	10	FTL	trip-153768492602129387	9	2018	
2	15	Carting	trip-153693976643699843	9	2018	
3	17	FTL	trip-153687145942424248	9	2018	
4	35	FTL	trip-153825970514894360	9	2018	

5 rows × 23 columns

EDA

```
plt.pie(df2['route_type'].value_counts(), labels = df['route_type'].unique())
```

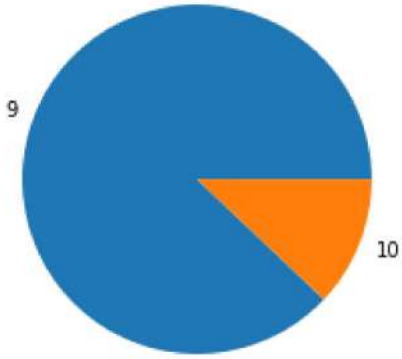
```
([<matplotlib.patches.Wedge at 0x7fab570dc040>,
  <matplotlib.patches.Wedge at 0x7fab570dc550>],
 [Text(-0.34386446114865143, 1.0448718736567406, 'Carting'),
  Text(0.3438643633206845, -1.0448719058516505, 'FTL')])
```


```
df2['route_type'].value_counts(normalize = True)
```

```
Carting    0.601201
FTL        0.398799
Name: route_type, dtype: float64
```

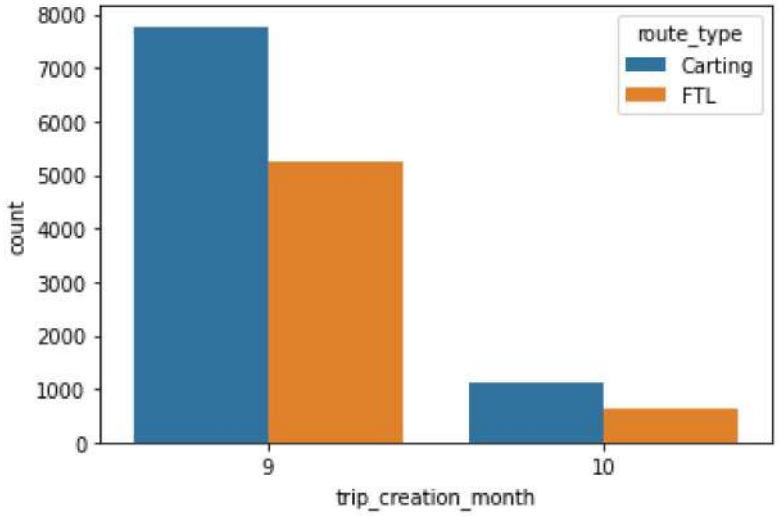
```
plt.pie(df2['trip_creation_month'].value_counts(), labels = df2['trip_creation_month'].unique
```

```
([<matplotlib.patches.Wedge at 0x7fab56b11be0>,
  <matplotlib.patches.Wedge at 0x7fab56b11cd0>],
 [Text(-1.0218966840805375, 0.40709601700974973, '9'),
  Text(1.021896655494225, -0.407096088767403, '10')])
```




```
sns.countplot(x = df2['trip_creation_month'], hue = df2['route_type'])
```

```
<Axes: xlabel='trip_creation_month', ylabel='count'>
```

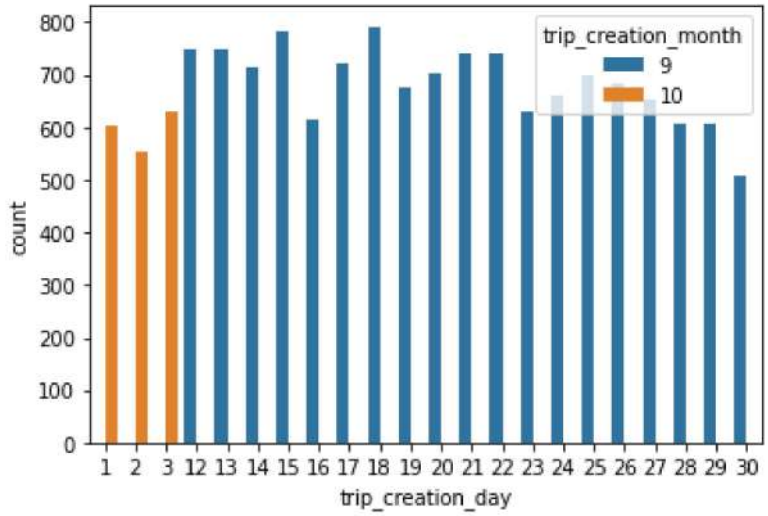


```
df2['trip_creation_month'].value_counts(normalize = True)
```

```
9     0.879328
10    0.120672
Name: trip_creation_month, dtype: float64
```

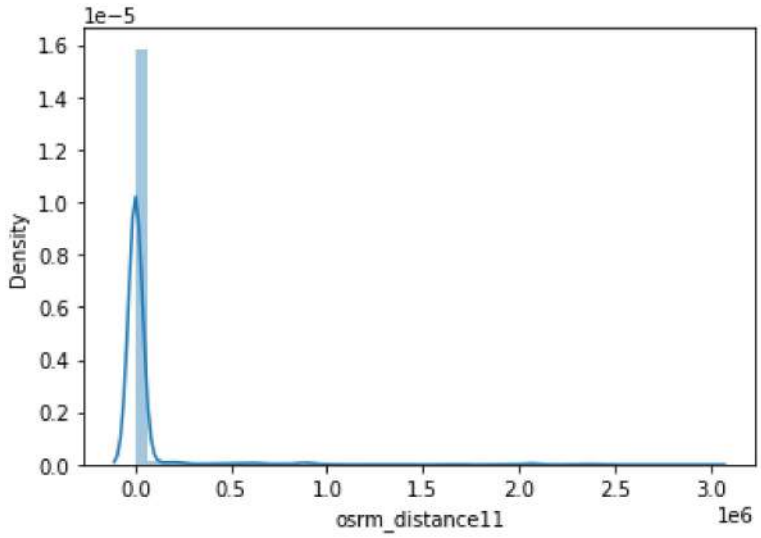
```
sns.countplot(hue = df2['trip_creation_month'], x = df2['trip_creation_day'])
```

```
<Axes: xlabel='trip_creation_day', ylabel='count'>
```



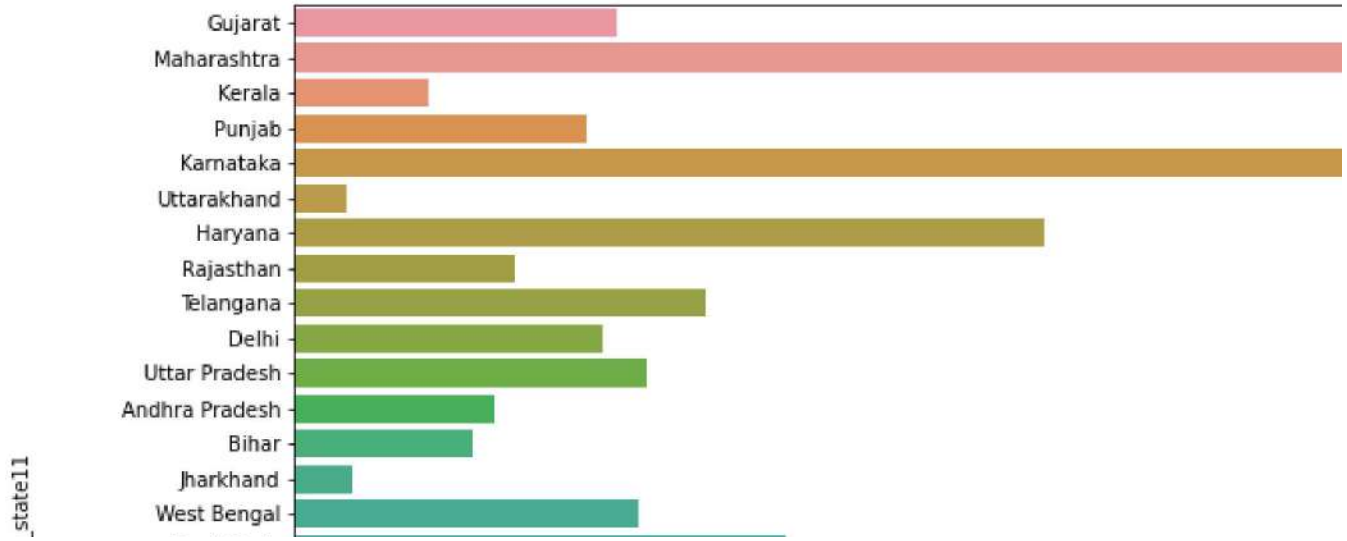

```
sns.distplot(df2['distance_to_destination_1'], kde = True)
```

```
<ipython-input-99-a0e9a8f4fdbf>:1: UserWarning:

sns.distplot(df2['osrm_distance11'], kde = True)

<ipython-input-102-83f293be7c44>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df2['osrm_distance11'], kde = True)
<Axes: xlabel='osrm_distance11', ylabel='Density'>
```



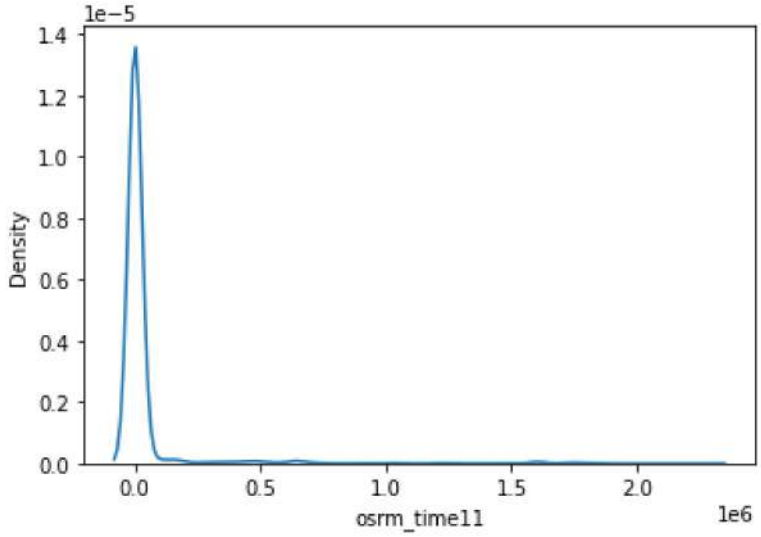

```
plt.figure(figsize = (10,10))
sns.countplot( y = df2['destination_state11'])
```

```
<Axes: xlabel='count', ylabel='destination_state11'>
```




```
sns.kdeplot(df2['osrm_time11'])
```

```
<Axes: xlabel='osrm_time11', ylabel='Density'>
```



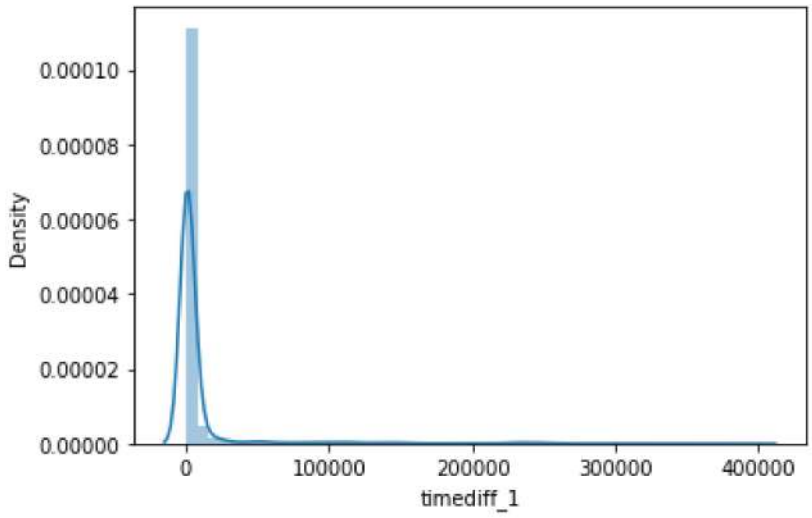

```
sns.kdeplot(df2['actual_time11'])
```

```
<Axes: xlabel='actual_time11', ylabel='Density'>
```

```
1e−6
```

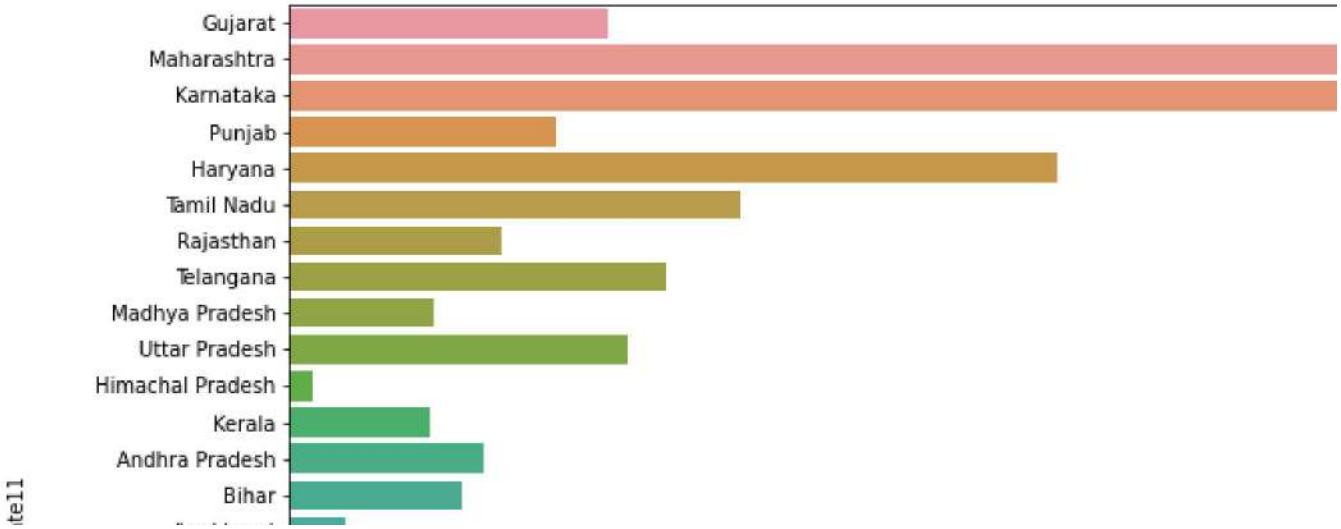
```
sns.distplot(df2['timediff_1'], kde = True)
```

```
<ipython-input-106-73c5d758d082>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df2['timediff_1'], kde = True)
<Axes: xlabel='timediff_1', ylabel='Density'>
```




```
plt.figure(figsize = (10,10))
sns.countplot( y = df2['source_state11'])
```

```
<Axes: xlabel='count', ylabel='source_state11'>
```

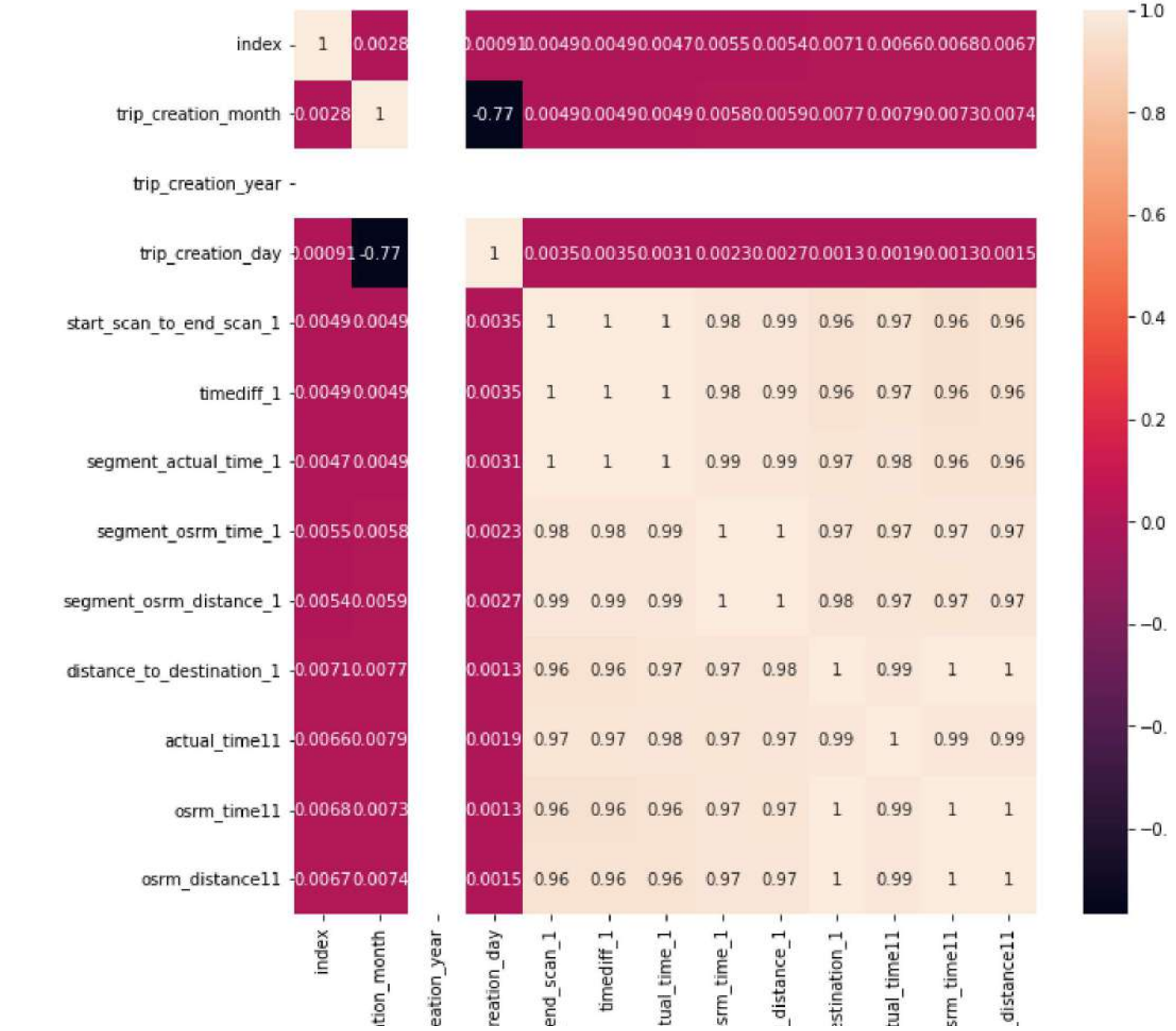



```
df2.corr()
```

	index	trip_creation_month	trip_creation_year	trip_creati
index	1.000000	0.002836	NaN	0.(
trip_creation_month	0.002836	1.000000	NaN	-0.1
trip_creation_year	NaN	NaN	NaN	
trip_creation_day	0.000908	-0.767661	NaN	1.(
start_scan_to_end_scan_1	0.004893	0.004895	NaN	0.(
timediff_1	0.004892	0.004894	NaN	0.(
segment_actual_time_1	0.004700	0.004930	NaN	0.(
segment_osrm_time_1	0.005548	0.005810	NaN	0.(
segment_osrm_distance_1	0.005406	0.005936	NaN	0.(
distance_to_destination_1	0.007051	0.007748	NaN	0.(
actual_time11	0.006562	0.007926	NaN	0.(
osrm_time11	0.006762	0.007303	NaN	0.(
osrm_distance11	0.006734	0.007366	NaN	0.(

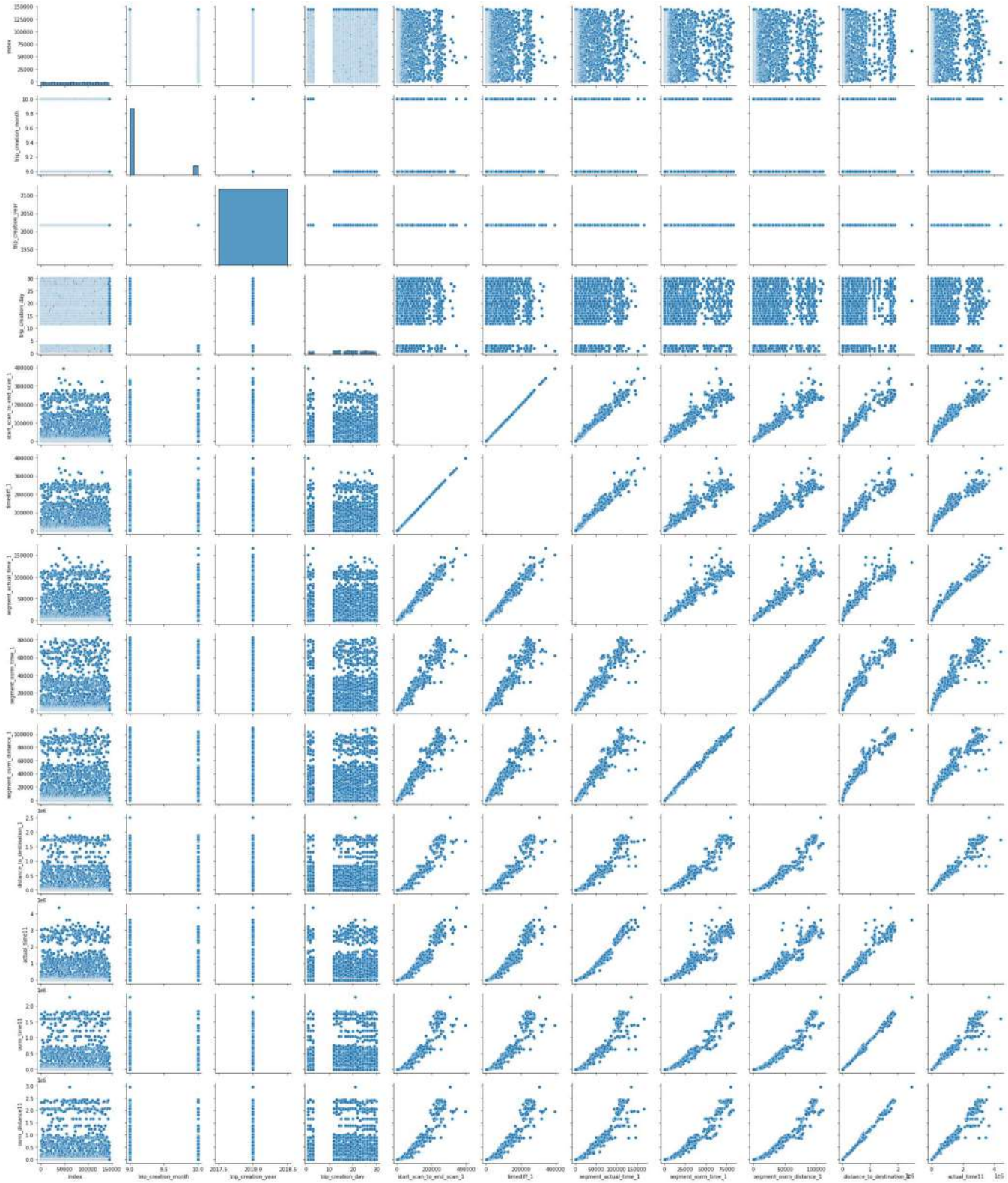
```
plt.figure(figsize = (10,10))
sns.heatmap(df2.corr(), annot = True)
```

```
<Axes: >
```




```
sns.pairplot(df2)
```

```
<seaborn.axisgrid.PairGrid at 0x7fab469162b0>
```



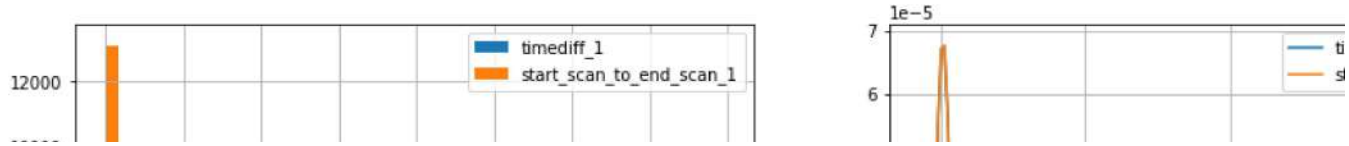
Insights

1. 60% of trandpotation type is carting and 40% is FTL
2. Maharashtra and Karnataka is the most popular source and destination location
3. 88% of trips are created on the month of October
4. 18th followed by 15th has got most of the trips
5. 0 trips between 4th and 11th
6. Most of the features are highly correlated
7. North eastern states like Mizoram, Nagaland, etc are the ones that has got low source and destination

Deep Analysis

Analysis for start_scan_to_end_scan and Timediff

```
plt.figure(figsize=(16,5))
plt.subplot(121)
plt.hist(df2['timediff_1'],bins=50,label='timediff_1')
plt.hist(df2['start_scan_to_end_scan_1'],bins=50,label='start_scan_to_end_scan_1')
plt.legend()
plt.grid()
plt.subplot(122)
sns.kdeplot(df2['timediff_1'],label='timediff_1')
sns.kdeplot(df2['start_scan_to_end_scan_1'],label='start_scan_to_end_scan_1')
plt.legend()
plt.grid()
plt.show()
```



since both the curves are overlapping, it proves that both timediff and start_scan_to_end_scan are same

hypothesis testing

H0: Both the mean for timediff and start_scan_to_end_scan are same

Ha: Both the means are different

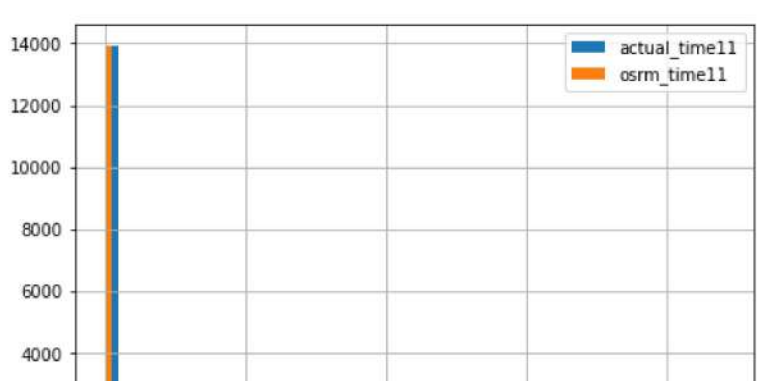
alpha = 0.05 (95% significance)

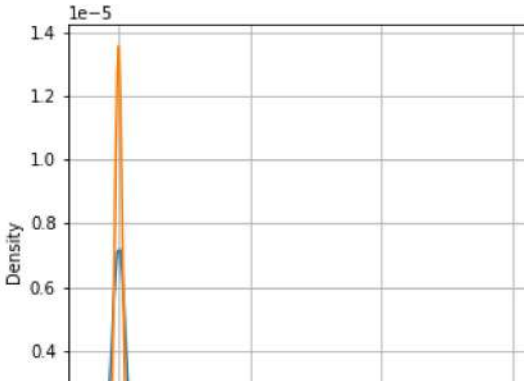
```
#T-test
_, p = ttest_ind(df2['timediff_1'], df2['start_scan_to_end_scan_1'])
if p>0.05:
  print('Fail to rejct null hypothesis')
else:
  print('Reject null hypothesis')
```

```
    Fail to rejct null hypothesis
```

Analysis on Actual time and OSRM time

```
plt.figure(figsize=(16,5))
plt.subplot(121)
plt.hist(df2['actual_time11'],bins=50,label='actual_time11')
plt.hist(df2['osrm_time11'],bins=50,label='osrm_time11')
plt.legend()
plt.grid()
plt.subplot(122)
sns.kdeplot(df2['actual_time11'],label='actual_time11')
sns.kdeplot(df2['osrm_time11'],label='osrm_time11')
plt.legend()
plt.grid()
plt.show()
```





The graph proves that there is a difference

hypothesis testing

H0: Both the mean for Actual time and OSRM time are same

Ha: Both the means are different

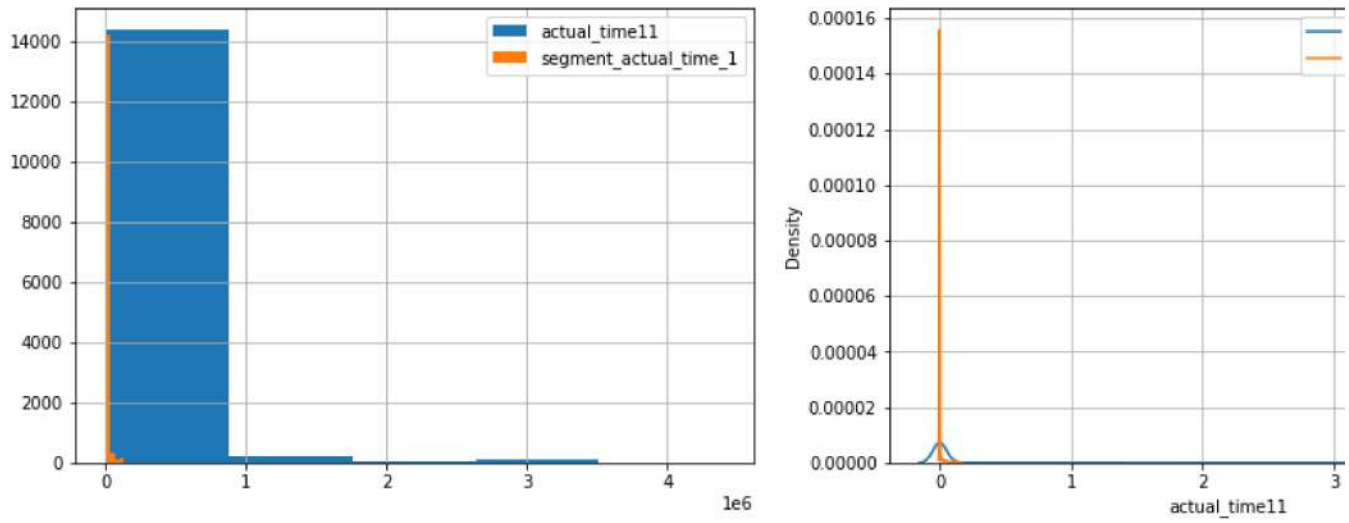
alpha = 0.05 (95% significance)

```
#T-test
_, p = ttest_ind(df2['actual_time11'], df2['osrm_time11'])
if p>0.05:
  print('Fail to rejct null hypothesis')
else:
  print('Reject null hypothesis')
```

```
    Reject null hypothesis
```

Analysis on Actual time and segmented actual time

```
plt.figure(figsize=(16,5))
plt.subplot(121)
plt.hist(df2['actual_time11'],bins=5,label='actual_time11')
plt.hist(df2['segment_actual_time_1'],bins=5,label='segment_actual_time_1')
plt.legend()
plt.grid()
plt.subplot(122)
sns.kdeplot(df2['actual_time11'],label='actual_time11')
sns.kdeplot(df2['segment_actual_time_1'],label='segment_actual_time_1')
plt.legend()
plt.grid()
plt.show()
```



Since there is a slight difference, the graph proves that the means are different

hypothesis testing

H0: Both the mean for Actual time and Segmented actual time are same

Ha: Both the means are different

alpha = 0.05 (95% significance)

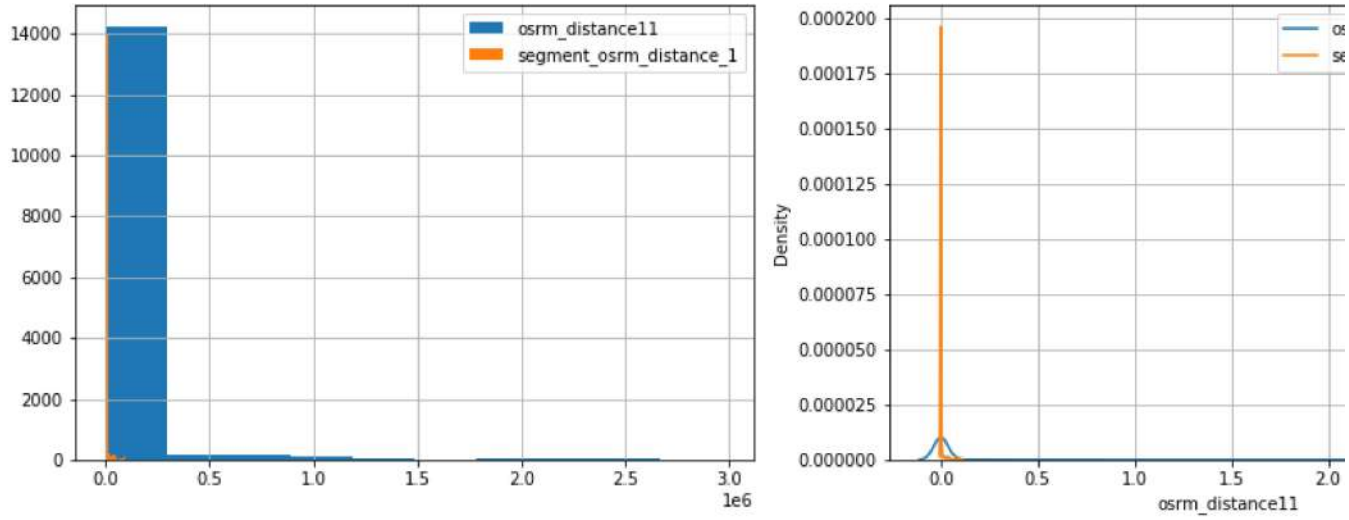
```
#T-test
_, p = ttest_ind(df2['actual_time11'], df2['segment_actual_time_1'])
if p>0.05:
  print('Fail to rejct null hypothesis')
else:
  print('Reject null hypothesis')
```

```
    Reject null hypothesis
```

Analysis on osrm distance aggregated value and segment osrm distance aggregated value

```
plt.figure(figsize=(16,5))
plt.subplot(121)
plt.hist(df2['osrm_distance11'],bins=10,label='osrm_distance11')
plt.hist(df2['segment_osrm_distance_1'],bins=10,label='segment_osrm_distance_1')
plt.legend()
plt.grid()
plt.subplot(122)
sns.kdeplot(df2['osrm_distance11'],label='osrm_distance11')
sns.kdeplot(df2['segment_osrm_distance_1'],label='segment_osrm_distance_1')
```

```
plt.legend()
plt.grid()
plt.show()
```



hypothesis testing

H0: Both the mean for osrm distance aggregated value and segment osrm distance aggregated value are same

Ha: Both the means are different

alpha = 0.05 (95% significance)

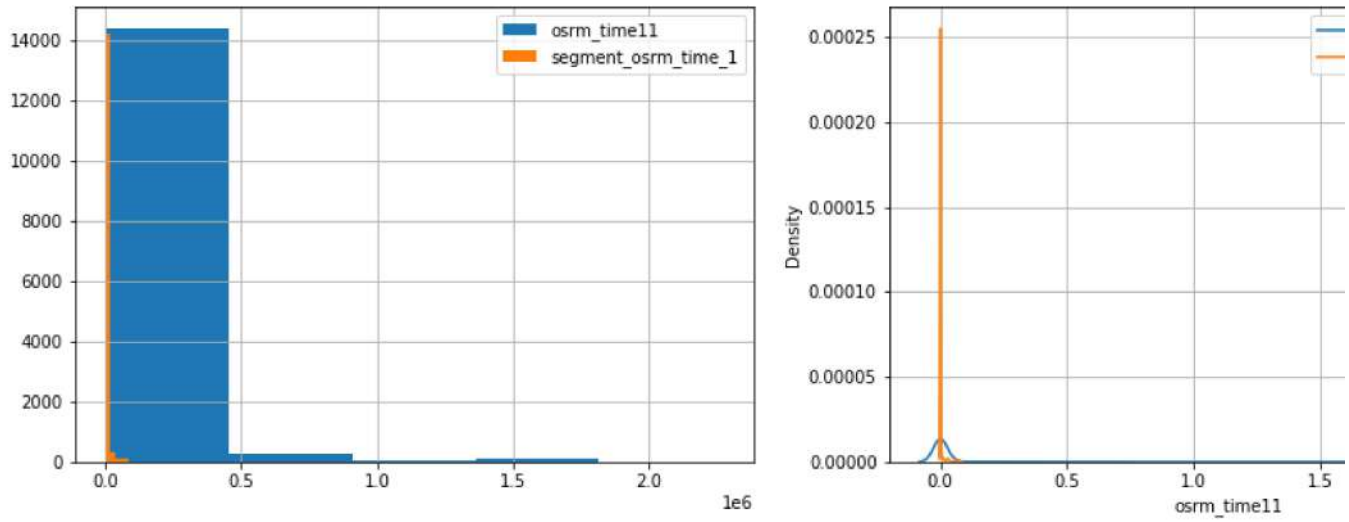
```
#T-test
_, p = ttest_ind(df2['osrm_distance11'], df2['segment_osrm_distance_1'])
if p>0.05:
  print('Fail to rejct null hypothesis')
else:
  print('Reject null hypothesis')
```

```
    Reject null hypothesis
```

Analysis on osrm time aggregated value and segment osrm time aggregated value

```
plt.figure(figsize=(16,5))
plt.subplot(121)
plt.hist(df2['osrm_time11'],bins=5,label='osrm_time11')
plt.hist(df2['segment_osrm_time_1'],bins=5,label='segment_osrm_time_1')
```

```
plt.legend()
plt.grid()
plt.subplot(122)
sns.kdeplot(df2['osrm_time11'],label='osrm_time11')
sns.kdeplot(df2['segment_osrm_time_1'],label='segment_osrm_time_1')
plt.legend()
plt.grid()
plt.show()
```



The graph shows that the means are different

hypothesis testing

H0: Both the mean for osrm time aggregated value and segment osrm time aggregated value are same

Ha: Both the means are different

alpha = 0.05 (95% significance)

```
#T-test
_, p = ttest_ind(df2['osrm_time11'], df2['segment_osrm_time_1'])
if p>0.05:
  print('Fail to rejct null hypothesis')
else:
  print('Reject null hypothesis')
```

```
    Reject null hypothesis
```

Insights

1. There is no difference in means of start_scan_to_end_scan and time difference
2. The means of actual_time aggregated value and OSRM time aggregated value are difference
3. The means of actual_time aggregated value and segment actual time aggregated value are difference
4. The means of osrm distance aggregated value and segment osrm distance aggregated value are difference
5. The means of osrm time aggregated value and segment osrm time aggregated value are difference

One hot encoding of categorical variables

```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14817 entries, 0 to 14816
Data columns (total 23 columns):
 #   Column                     Non-Null Count  Dtype  
---  ------                     --------------  -----  
 0   index                      14817 non-null  int64  
 1   route_type                 14817 non-null  object 
 2   trip_uuid                  14817 non-null  object 
 3   trip_creation_month        14817 non-null  int64  
 4   trip_creation_year         14817 non-null  int64  
 5   trip_creation_day          14817 non-null  int64  
 6   start_scan_to_end_scan_1   14817 non-null  float64
 7   timediff_1                 14817 non-null  float64
 8   segment_actual_time_1      14817 non-null  float64
 9   segment_osrm_time_1        14817 non-null  float64
 10  segment_osrm_distance_1    14817 non-null  float64
 11  distance_to_destination_1  14817 non-null  float64
 12  actual_time11              14817 non-null  float64
 13  osrm_time11                14817 non-null  float64
 14  osrm_distance11            14817 non-null  float64
 15  source_city11              14807 non-null  object 
 16  source_place11             14297 non-null  object 
 17  source_code11              13580 non-null  object 
 18  source_state11             13580 non-null  object 
 19  destination_city11         14809 non-null  object 
 20  destination_place11        14185 non-null  object 
 21  destination_code11         13526 non-null  object 
 22  destination_state11        13526 non-null  object 
dtypes: float64(9), int64(4), object(10)
memory usage: 2.6+ MB
```

```
catcol = ['route_type', 'trip_uuid', 'source_city11', 'source_place11', 'source_code11', 'sou
```

```
df2[catcol]
```

	route_type	trip_uuid	source_city11	source_place11	source_code11	so
0	Carting	trip-153741093647649320	Anand	VUNagar	DC	
1	FTL	trip-153768492602129387	Bhiwandi	Mankoli	HB	
2	Carting	trip-153693976643699843	LowerParel	CP (Maharashtra)	None	
3	FTL	trip-153687145942424248	Bangalore	Nelmngla	H	
4	FTL	trip-153825970514894360	Ludhiana	GillChwk	DC	
...	...	...	...	...	...	
14812	FTL	trip-153799142965708367	Dhar	Trimurti	D	Ma
14813	Carting	trip-153695073416451616	Mumbai	Jogeshwri	L	
14814	FTL	trip-153761584139918815	Bhiwandi	Mankoli	HB	
14815	Carting	trip-153718412883843340	MAA	Poonamallee	HB	
14816	Carting	trip-153746066843555182	Sonipat	Kundli	H	

14817 rows × 10 columns

```
#one hot encoding cat cols
dff = pd.concat([df2, pd.get_dummies(df2['route_type'])], axis = 1)
dff.head()
```

ip_uuid	trip_creation_month	trip_creation_year	trip_creation_day	start_scan_to_end_sc
trip-7649320	9	2018	20	
trip-2129387	9	2018	23	1

```
#droping col route_type
dff.drop(columns = ['route_type', 'index', 'trip_uuid'], axis= 1, inplace = True)
dff.head()
```

	trip_creation_month	trip_creation_year	trip_creation_day	start_scan_to_end_scan_1
0	9	2018	20	975.0
1	9	2018	23	1510.0
2	9	2018	14	216.0
3	9	2018	13	13766.0
4	9	2018	29	433.0

5 rows × 22 columns

Other categorical columns when encoded using one hot encoding will give us a lot of columns. Also trip_uuid can be dropped since all values are unique and is less important. for the rest of the columns, the cardinality can be reduced by aggregating values. As of now i'm moving on with standardization of other numerical values

```
df_f = dff.loc[:, ['trip_creation_month', 'trip_creation_year', 'trip_creation_day', 'start_s

s = StandardScaler()
df_f1 = s.fit_transform(df_f)
```

```
df_f1 = pd.DataFrame(df_f1, columns = df_f.columns)
df_f1.head()
```

	trip_creation_month	trip_creation_year	trip_creation_day	start_scan_to_end_scan_1
0	-0.370449	0.0	0.206412	-0.249947
1	-0.370449	0.0	0.586495	-0.234072
2	-0.370449	0.0	-0.553755	-0.272469
3	-0.370449	0.0	-0.680449	0.129602
4	-0.370449	0.0	1.346661	-0.266029

```
## Recommendations

1. North eastern states like Mizoram, Nagaland
coupon to promote Delhivery's services

2. Also, there are zero sales from 4th to 11th
attract customers and promote their services o

3. Since FTL is being used by 40%, FTL option
stating that shipments and services would be m
attract customers telling that services would

4. More hubs can be setup in Maharashtra, Karn
large numberof shipments are dispatched and ge
can be attracted due to the brand popularity a
faster aswell

5. Month starts and end, discounts could be gi
customers since less number of shipment is bei
```

Recommendations

1. North eastern states like Mizoram, Nagaland can be given more discounts or coupon to promote Delhivery's services
2. Also, there are zero sales from 4th to 11th. Special offers can be given to attract customers and promote their services on those days aswell
3. Since FTL is being used by 40%, FTL option can be promoted and used more stating that shipments and services would be much faster through it. Could attract customers telling that services would be offered much faster
4. More hubs can be setup in Maharashtra, Karnataka, Haryana, Tamilnadu since a large numberof shipments are dispatched and generated there, so more customers can be attracted due to the brand popularity and services could be offered much faster aswell

5. Month starts and end, discounts could be given to attract much more customers since less number of shipment is being appeared during this time

✓ 0s completed at 13:36