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.

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

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

```
data="https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/551/original/delhivery_data.csv?1642751181"
```

```
df=pd.read_csv(data)
```

```
df.head()
```

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_center	source_name
0	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)
1	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)
2	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)
3	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)
4	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)

5 rows × 24 columns

```
df.sample(5)
```

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_center	sou
118328	training	2018-09-18 22:20:12.977102	thanos::sroute:7d7c2e06-d535-48a0-81ab-4ae0189...	FTL	trip-153730921297684409	IND501359AAE	Hyderabad_Sha (T
82334	training	2018-09-17 09:52:21.068600	thanos::sroute:8fa95fb6-1cf5-4fbd-aa64-04418b2...	Carting	trip-153717794106835193	IND560099AAB	Bengaluru_Bom (K
4203	training	2018-09-22 08:39:36.801371	thanos::sroute:ce2ecabf-2dae-4b18-92d1-07b6b69...	FTL	trip-153760557680099399	IND244235AAA	Gajraula_Jav (Uttar
109238	test	2018-09-27 13:12:35.780965	thanos::sroute:7af51efd-ae4d-49bc-9b68-345abe6...	FTL	trip-153805395578070317	IND562132AAA	Bangalore_Ne (K
43618	test	2018-10-03 04:33:42.941135	thanos::sroute:a4bf93af-8105-4dff-818e-cb79ddd...	FTL	trip-153854122294085889	IND000000ACB	Gurgaon_Bil

5 rows × 24 columns

```
df.shape
```

```
(144867, 24)
```

```
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
```

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

data	count
training	104858
test	40009

dtype: int64

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

	count
route_type	
FTL	99660
Carting	45207

dtype: int64

```
df.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

source name and destination name have null values

```
df.duplicated().value_counts()
```

	count
False	144867

dtype: int64

No duplicate values

```
df.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

A quick look at the information of the data reveals that there are 144867 rows and 24 columns implying 144867 trips have been made with each trip having information such as trip_creation_time,trip_uuid, source_center, source_name, destination_center, destination_name to name a few. Most of the datatype are either "object" or "float64" except for is_cutoff and cutoff_factor

We can also infer that there are 293 missing values or null value in source_name and 261 missing values or null value in destination_name in the dataset. As these numbers are small compared to dataset size, 144867, it is safe to drop the rows with the missing values.

There are no duplicate entries.

*** As columns is_cutoff, cutoff_factor, cutoff_timestamp, factor and segment_factor are Unknown fields,there is no harm in dropping these columns.***

***It makes sense to convert columns data and route_type to "category" datatype ***

It makes sense to convert columns trip_creation_time, od_start_time, od_end_time to "datetime" datatype

```
df = df.dropna(how='any')
df = df.drop(columns = ["is_cutoff", "cutoff_factor", "cutoff_timestamp", "factor", "segment_factor"],axis=1)
df["data"] = df["data"].astype("category")
df["route_type"] = df["route_type"].astype("category")
df["trip_creation_time"] = pd.to_datetime(df["trip_creation_time"], format='%Y-%m-%d %H:%M:%S.%f')
df["od_start_time"] = pd.to_datetime(df["od_start_time"], format='%Y-%m-%d %H:%M:%S.%f')
df["od_end_time"] = pd.to_datetime(df["od_end_time"], format='%Y-%m-%d %H:%M:%S.%f')
```

```
df.shape
```

```
(144316, 19)
```

```
df.head()
```

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_center	source_name
0	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)
1	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)
2	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)
3	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)
4	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND388121AAA	Anand_VUNagar_DC (Gujarat)

```
df.describe()
```

	trip_creation_time	od_start_time	od_end_time	start_scan_to_end_scan	actual_distance_to_destination
count	144316	144316	144316	144316.000000	144316.000000
mean	2018-09-22 13:05:09.454117120	2018-09-22 17:32:42.435769344	2018-09-23 09:36:54.057172224	963.697698	234.708498
min	2018-09-12 00:00:16.535741	2018-09-12 00:00:16.535741	2018-09-12 00:50:10.814399	20.000000	9.000045
25%	2018-09-17 02:46:11.004421120	2018-09-17 07:37:35.014584832	2018-09-18 01:29:56.978912	161.000000	23.352027
50%	2018-09-22 03:36:19.186585088	2018-09-22 07:35:23.038482944	2018-09-23 02:49:00.936600064	451.000000	66.135322
75%	2018-09-27 17:53:19.027942912	2018-09-27 22:01:30.861209088	2018-09-28 12:13:41.675546112	1645.000000	286.919294
max	2018-10-03 23:59:42.701692	2018-10-06 04:27:23.392375	2018-10-08 03:00:24.353479	7898.000000	1927.447705
std	NaN	NaN	NaN	1038.082976	345.480571

Insights

. The data is provided from 2018-09-12 00:00:16.535741 to 2018-10-03 23:59:42.701692

. The average time taken to deliver from source to destination is 964 mins with least time being 20mins and maximum time being 7898 mins

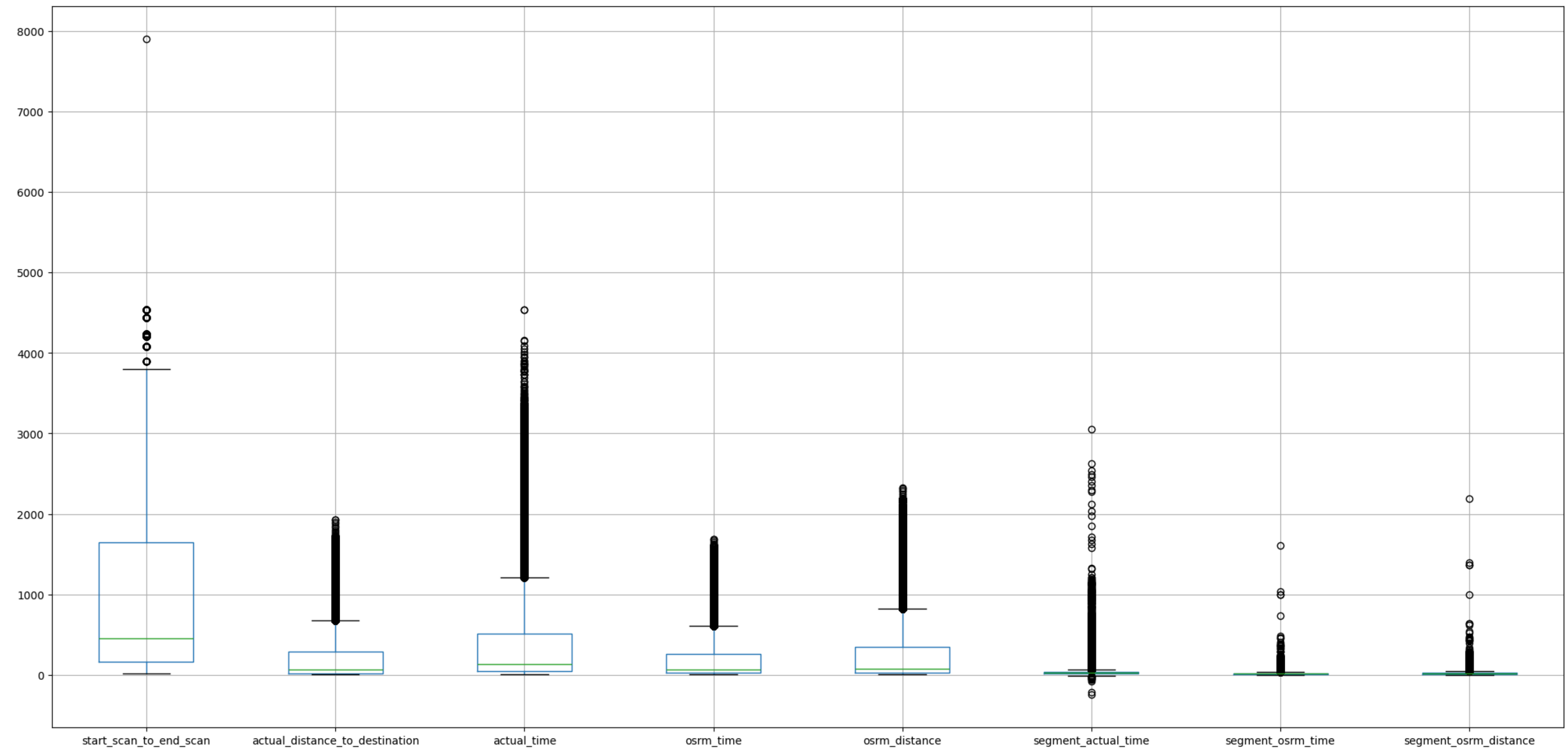
. The average distance between source and destination warehouse is 235 Kms with least distance being 9 Kms and maximum distance being 1927 Kms

Detecting Outliers

```
def detectOutliers(df):
    q1 = df.quantile(0.25)
    q3 = df.quantile(0.75)
    iqr = q3-q1
    lower_outliers = df[df<(q1-1.5*iqr)]
    higher_outliers = df[df>(q3+1.5*iqr)]
    return lower_outliers, higher_outliers


numerical_columns = ['start_scan_to_end_scan', 'actual_distance_to_destination', 'actual_time', 'osrm_time', 'osrm_distance
column_outlier_dictionary = {}
for column in numerical_columns:
    print('*'*50)
    print(f'Outliers of \'{column}\' column are:')
    lower_outliers, higher_outliers = detectOutliers(df[column])
    print("Lower outliers:\n", lower_outliers)
    print("Higher outliers:\n", higher_outliers)
    print('*'*50, end="\n")
    column_outlier_dictionary[column] = [lower_outliers, higher_outliers]
```

```
*************************************************
*************************************************
Outliers of 'segment_osrm_time' column are:
Lower outliers:
 Series([], Name: segment_osrm_time, dtype: float64)
Higher outliers:
 34        70.0
38        45.0
157       81.0
158       81.0
214       44.0
          ...
144802    48.0
144829    74.0
144837    42.0
144843    43.0
144845    54.0
Name: segment_osrm_time, Length: 6348, dtype: float64
*************************************************
*************************************************
Outliers of 'segment_osrm_distance' column are:
Lower outliers:
 Series([], Name: segment_osrm_distance, dtype: float64)
Higher outliers:
 34        72.5561
157       79.6653
158       82.4127
214       52.7136
316       60.0755
           ...
144774    60.6393
144802    61.0445
144829    70.0436
144837    60.4795
144845    55.6993
Name: segment_osrm_distance, Length: 4295, dtype: float64
*************************************************
```

```
df[numerical_columns].boxplot(figsize=(25,12))
plt.show()
```

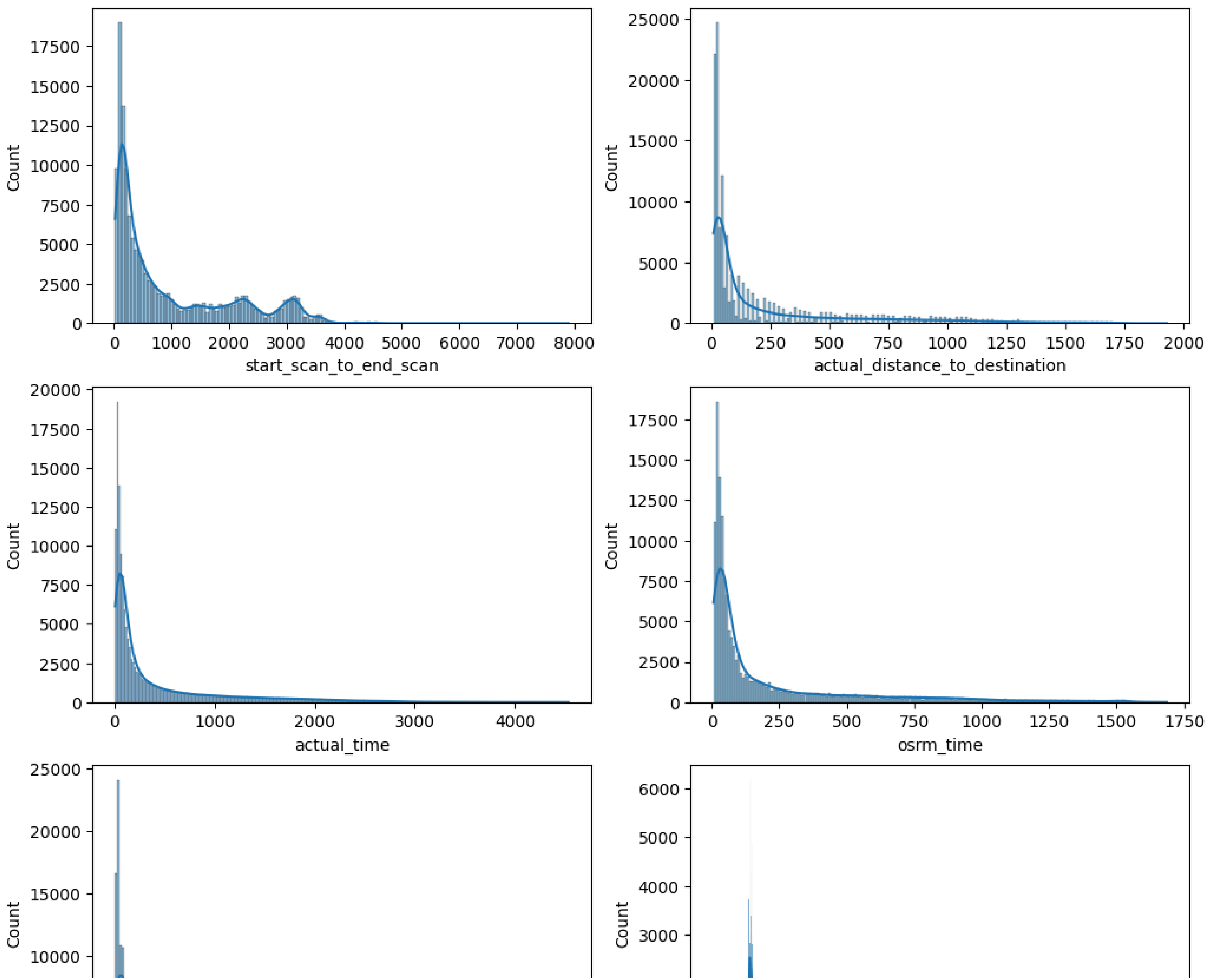

8000
7000
6000
5000
4000
3000
2000
1000
0
start_scan_to_end_scan
actual_distance_to_destination
actual_time
osrm_time
osrm_distance
segment_actual_time
segment_osrm_time
segment_osrm_distance

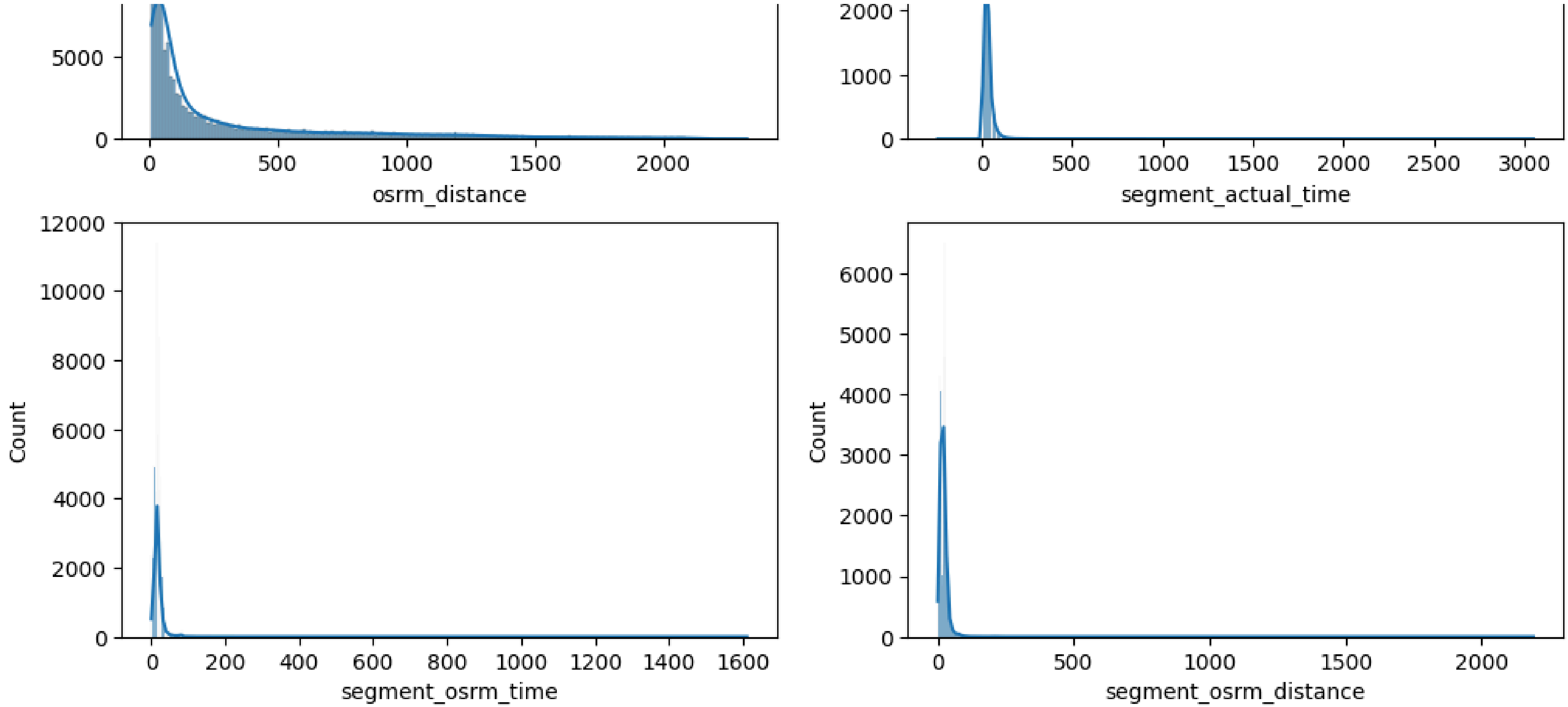
```
for key, value in column_outlier_dictionary.items():
    print(f'The column \'{key}\' has {len(value[0]) + len(value[1])} outliers')
```

```
The column 'start_scan_to_end_scan' has 373 outliers
The column 'actual_distance_to_destination' has 17818 outliers
The column 'actual_time' has 16507 outliers
The column 'osrm_time' has 17406 outliers
The column 'osrm_distance' has 17547 outliers
The column 'segment_actual_time' has 9262 outliers
The column 'segment_osrm_time' has 6348 outliers
The column 'segment_osrm_distance' has 4295 outliers
```

Univariate ANalysis

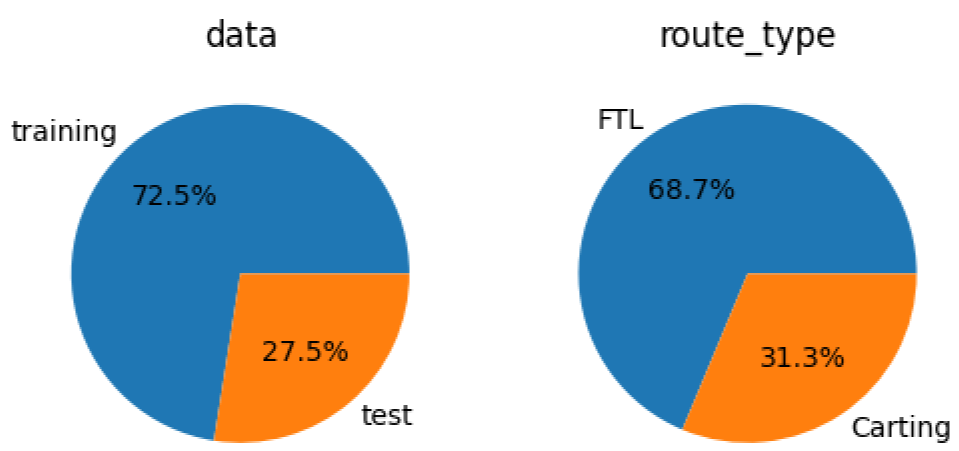
```
fig, ax = plt.subplots(nrows=4, ncols=2, figsize = (12, 16))
sns.histplot(data=df, x = "start_scan_to_end_scan", kde=True, ax=ax[0,0])
sns.histplot(data=df, x = "actual_distance_to_destination", kde=True, ax=ax[0,1])
sns.histplot(data=df, x = "actual_time", kde=True, ax=ax[1,0])
sns.histplot(data=df, x = "osrm_time", kde=True, ax=ax[1,1])
sns.histplot(data=df, x = "osrm_distance", kde=True, ax=ax[2,0])
sns.histplot(data=df, x = "segment_actual_time", kde=True, ax=ax[2,1])
sns.histplot(data=df, x = "segment_osrm_time", kde=True, ax=ax[3,0])
sns.histplot(data=df, x = "segment_osrm_distance", kde=True, ax=ax[3,1])
plt.show()
```


Count
17500
15000
12500
10000
7500
5000
2500
0
0 1000 2000 3000 4000 5000 6000 7000 8000
start_scan_to_end_scan
Count
25000
20000
15000
10000
5000
0
0 250 500 750 1000 1250 1500 1750 2000
actual_distance_to_destination
Count
20000
17500
15000
12500
10000
7500
5000
2500
0
0 1000 2000 3000 4000
actual_time
Count
17500
15000
12500
10000
7500
5000
2500
0
0 250 500 750 1000 1250 1500 1750
osrm_time
Count
25000
20000
15000
10000
Count
6000
5000
4000
3000



osrm_distance
segment_actual_time
Count
segment_osrm_time
Count
segment_osrm_distance

```
categorical_columns = ["data", "route_type"]
plt.figure(figsize=(6,6))
plt.subplot(1,2,1)
data = df["data"].value_counts()
plt.pie(data.values, labels = data.index, autopct='%.1f%%')
plt.title("data")
plt.subplot(1,2,2)
data = df["route_type"].value_counts()
plt.pie(data.values, labels = data.index, autopct='%.1f%%')
plt.title("route_type")
plt.show()
```



Insights

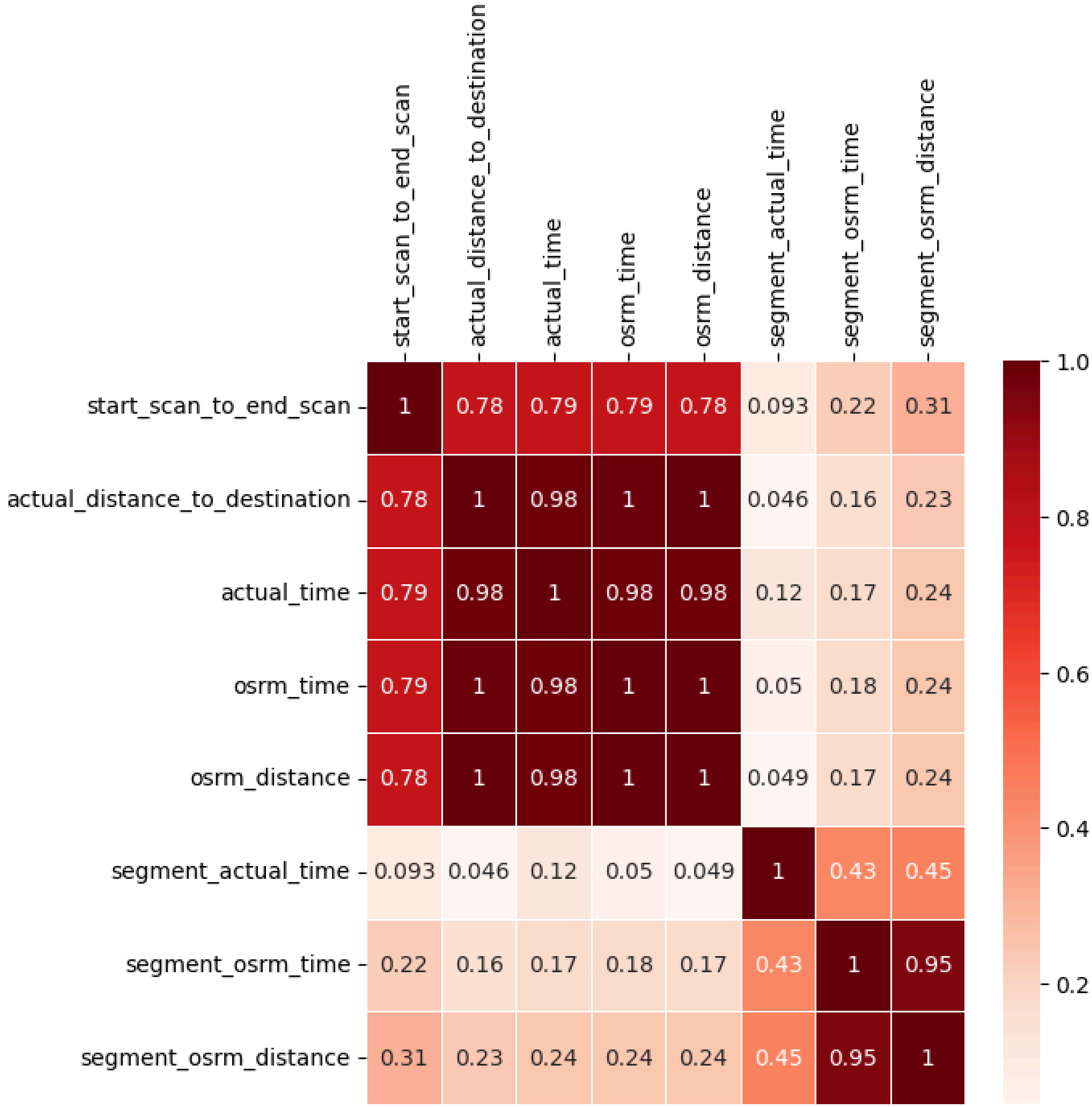
The histogram plot of all the numerical values show that all the data is right skewed

72.5% of the data is training data and remaining 27.5% is testing data

68.7% of the delivery is done via FTL and remaining 31.3% through Carting

```
fig, ax = plt.subplots(figsize=(6,6))
sns.heatmap(df.select_dtypes(include=np.number).corr(), annot=True, linewidth=0.5, cmap = "Reds", ax=ax)
```

```
    ax.xaxis.tick_top()
    plt.xticks(rotation=90)
    plt.show()
```


start_scan_to_end_scan
actual_distance_to_destination
actual_time
osrm_time
osrm_distance
segment_actual_time
segment_osrm_time
segment_osrm_distance
1 0.78 0.79 0.79 0.78 0.093 0.22 0.31
0.78 1 0.98 1 1 0.046 0.16 0.23
0.79 0.98 1 0.98 0.98 0.12 0.17 0.24
0.79 1 0.98 1 1 0.05 0.18 0.24
0.78 1 0.98 1 1 0.049 0.17 0.24
0.093 0.046 0.12 0.05 0.049 1 0.43 0.45
0.22 0.16 0.17 0.18 0.17 0.43 1 0.95
0.31 0.23 0.24 0.24 0.24 0.45 0.95 1
1.0
0.8
0.6
0.4
0.2

Insights

. The heatmap clearly shows high correlation between time and distance. This is expected as the delivery time increases with increase in distance

. Actual_distance_to_destination, actual_time, osrm_time and osrm_distance are highly correlated and segment_osrm_time and segment_osrm_distance are highly correlated

```
df["segment_key"] = df["trip_uuid"] + '_' + df["source_center"] + '_' + df["destination_center"]
df = df.drop(columns=["source_center", "destination_center"])
segment_columns = ["segment_actual_time", "segment_osrm_distance", "segment_osrm_time"]
for col in segment_columns:
    df[col + "_sum"] = df.groupby("segment_key")[col].cumsum()


segment_dict = {
    'data' : 'first',
    'trip_creation_time' : 'first',
    'route_schedule_uuid' : 'first',
    'route_type' : 'first',
    'trip_uuid' : 'first',
    'source_name' : 'first',
    'destination_name' : 'last',
    'od_start_time' : 'first',
    'od_end_time' : 'last',
    'start_scan_to_end_scan' : 'first',
    'actual_distance_to_destination' : 'last',
    'actual_time' : 'last',
    'osrm_time' : 'last',
    'osrm_distance' : 'last',
    'segment_actual_time_sum' : 'last',
    'segment_osrm_distance_sum' : 'last',
    'segment_osrm_time_sum' : 'last',
}

segment_df = df.groupby('segment_key').agg(segment_dict).reset_index()
segment_df = segment_df.sort_values(by=['segment_key', 'od_end_time'], ascending=True).reset_index()
```

```
df.head()
```

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

5 rows × 21 columns

```
segment_df.head()
```

	index	segment_key	data	trip_creation_time	route_schedule_uuid	route_type
0	0	trip-153671041653548748_IND209304AAA_IND000000ACB	training	2018-09-12 00:00:16.535741	thanos::sroute:d7c989ba-a29b-4a0b-b2f4-288cdc6...	FTL
1	1	trip-153671041653548748_IND462022AAA_IND209304AAA	training	2018-09-12 00:00:16.535741	thanos::sroute:d7c989ba-a29b-4a0b-b2f4-288cdc6...	FTL
2	2	trip-153671042288605164_IND561203AAB_IND562101AAA	training	2018-09-12 00:00:22.886430	thanos::sroute:3a1b0ab2-bb0b-4c53-8c59-eb2a2c0...	Carting
3	3	trip-153671042288605164_IND572101AAA_IND561203AAB	training	2018-09-12 00:00:22.886430	thanos::sroute:3a1b0ab2-bb0b-4c53-8c59-eb2a2c0...	Carting
4	4	trip-153671043369099517_IND000000ACB_IND160002AAC	training	2018-09-12 00:00:33.691250	thanos::sroute:de5e208e-7641-45e6-8100-4d9fb1e...	FTL

4.5. Feature Engineering

Extracting features from given data

Extracting time taken between od_start_time and od_end_time

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

```
segment_df.head()
```

	index	segment_key	data	trip_creation_time	route_schedule_uuid	route_type
0	0	trip-153671041653548748_IND209304AAA_IND000000ACB	training	2018-09-12 00:00:16.535741	thanos::sroute:d7c989ba-a29b-4a0b-b2f4-288cdc6...	FTL
1	1	trip-153671041653548748_IND462022AAA_IND209304AAA	training	2018-09-12 00:00:16.535741	thanos::sroute:d7c989ba-a29b-4a0b-b2f4-288cdc6...	FTL
2	2	trip-153671042288605164_IND561203AAB_IND562101AAA	training	2018-09-12 00:00:22.886430	thanos::sroute:3a1b0ab2-bb0b-4c53-8c59-eb2a2c0...	Carting
3	3	trip-153671042288605164_IND572101AAA_IND561203AAB	training	2018-09-12 00:00:22.886430	thanos::sroute:3a1b0ab2-bb0b-4c53-8c59-eb2a2c0...	Carting
4	4	trip-153671043369099517_IND000000ACB_IND160002AAC	training	2018-09-12 00:00:33.691250	thanos::sroute:de5e208e-7641-45e6-8100-4d9fb1e...	FTL

Extracting city, place, code and state from source_name and destination_name

```
segment_df['source_state'] = segment_df['source_name'].str.extract(r'\((.*?)\)')
segment_df['source_data'] = segment_df['source_name'].str.extract(r'^(.*?)\(')
segment_df['source_data'] = segment_df['source_data'].str.strip()

segment_df['destination_state'] = segment_df['destination_name'].str.extract(r'\((.*?)\)')
segment_df['destination_data'] = segment_df['destination_name'].str.extract(r'^(.*?)\(')
segment_df['destination_data'] =segment_df['destination_data'].str.strip()


def extract_city_place_code(name):
    parts = name.split('_')
    num_of_parts = len(parts)
    if(num_of_parts == 3):
        city = parts[0]
```

```
            place = parts[1]
            code = parts[2]
        elif(num_of_parts == 2):
            city = parts[0]
            place = parts[1]
            code = 'none'
        else:
            city = parts[0]
            place = city
            code = 'none'

        if city == 'Bangalore' or city == 'HBR Layout PC' or city == 'BLR':
            city = 'Bengaluru'
        elif city == 'Mumbai Hub' or city == 'BOM':
            city = 'Mumbai'
        elif city == 'Del':
            city = 'Delhi'
        elif city == 'PNQ Pashan DPC' or city == 'PNQ Vadgaon Sheri DPC':
            city = 'Pune'
        elif city == 'MAA':
            city = 'Chennai'
        elif city == 'FBD':
            city = 'Faridabad'
        elif city == 'CCU':
            city = 'Kolkata'
        elif city == 'AMD':
            city = 'Ahmedabad'
        elif city == 'FBD':
            city = 'Faridabad'
        elif city == 'GGN':
            city = 'Gurgaon'
        elif city == 'GZB':
            city = 'Ghaziabad'

        return [city, place, code]
```

```
extracted_df = segment_df['source_data'].apply(lambda x: extract_city_place_code(x))
segment_df[['source_city','source_place','source_code']] = pd.DataFrame(extracted_df.tolist(), index= segment_df.index)
extracted_df = segment_df['destination_data'].apply(lambda x: extract_city_place_code(x))
segment_df[['destination_city','destination_place','destination_code']] = pd.DataFrame(extracted_df.tolist(), index= segmen
segment_df = segment_df.drop(columns=['source_name', 'source_data', 'destination_name', 'destination_data'])
segment_df.head()
```

	index	segment_key	data	trip_creation_time	route_schedule_uuid	route_type
0	0	trip-153671041653548748_IND209304AAA_IND000000ACB	training	2018-09-12 00:00:16.535741	thanos::sroute:d7c989ba-a29b-4a0b-b2f4-288cdc6...	FTL
1	1	trip-153671041653548748_IND462022AAA_IND209304AAA	training	2018-09-12 00:00:16.535741	thanos::sroute:d7c989ba-a29b-4a0b-b2f4-288cdc6...	FTL
2	2	trip-153671042288605164_IND561203AAB_IND562101AAA	training	2018-09-12 00:00:22.886430	thanos::sroute:3a1b0ab2-bb0b-4c53-8c59-eb2a2c0...	Carting
3	3	trip-153671042288605164_IND572101AAA_IND561203AAB	training	2018-09-12 00:00:22.886430	thanos::sroute:3a1b0ab2-bb0b-4c53-8c59-eb2a2c0...	Carting
4	4	trip-153671043369099517_IND000000ACB_IND160002AAC	training	2018-09-12 00:00:33.691250	thanos::sroute:de5e208e-7641-45e6-8100-4d9fb1e...	FTL

5 rows × 24 columns

```
segment_df['trip_creation_year'] = segment_df['trip_creation_time'].dt.year
segment_df['trip_creation_month'] = segment_df['trip_creation_time'].dt.month
segment_df['trip_creation_day'] = segment_df['trip_creation_time'].dt.day
segment_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26222 entries, 0 to 26221
Data columns (total 27 columns):
 #   Column                          Non-Null Count  Dtype
```

```
 ---  ------                          --------------  -----
  0   index                           26222 non-null  int64
  1   segment_key                     26222 non-null  object
  2   data                            26222 non-null  category
  3   trip_creation_time              26222 non-null  datetime64[ns]
  4   route_schedule_uuid             26222 non-null  object
  5   route_type                      26222 non-null  category
  6   trip_uuid                       26222 non-null  object
  7   start_scan_to_end_scan          26222 non-null  float64
  8   actual_distance_to_destination  26222 non-null  float64
  9   actual_time                     26222 non-null  float64
  10  osrm_time                       26222 non-null  float64
  11  osrm_distance                   26222 non-null  float64
  12  segment_actual_time_sum         26222 non-null  float64
  13  segment_osrm_distance_sum       26222 non-null  float64
  14  segment_osrm_time_sum           26222 non-null  float64
  15  od_time_diff_hour               26222 non-null  float64
  16  source_state                    26222 non-null  object
  17  destination_state               26222 non-null  object
  18  source_city                     26222 non-null  object
  19  source_place                    26222 non-null  object
  20  source_code                     26222 non-null  object
  21  destination_city                26222 non-null  object
  22  destination_place               26222 non-null  object
  23  destination_code                26222 non-null  object
  24  trip_creation_year              26222 non-null  int32
  25  trip_creation_month             26222 non-null  int32
  26  trip_creation_day               26222 non-null  int32
dtypes: category(2), datetime64[ns](1), float64(9), int32(3), int64(1), object(11)
memory usage: 4.8+ MB
```

** In-depth analysis**

Grouping and aggregating at trip-level

Group the segment_df by trip_uuid

```
trip_dict = {
    'segment_key' : 'first',
    'data' : 'first',
```

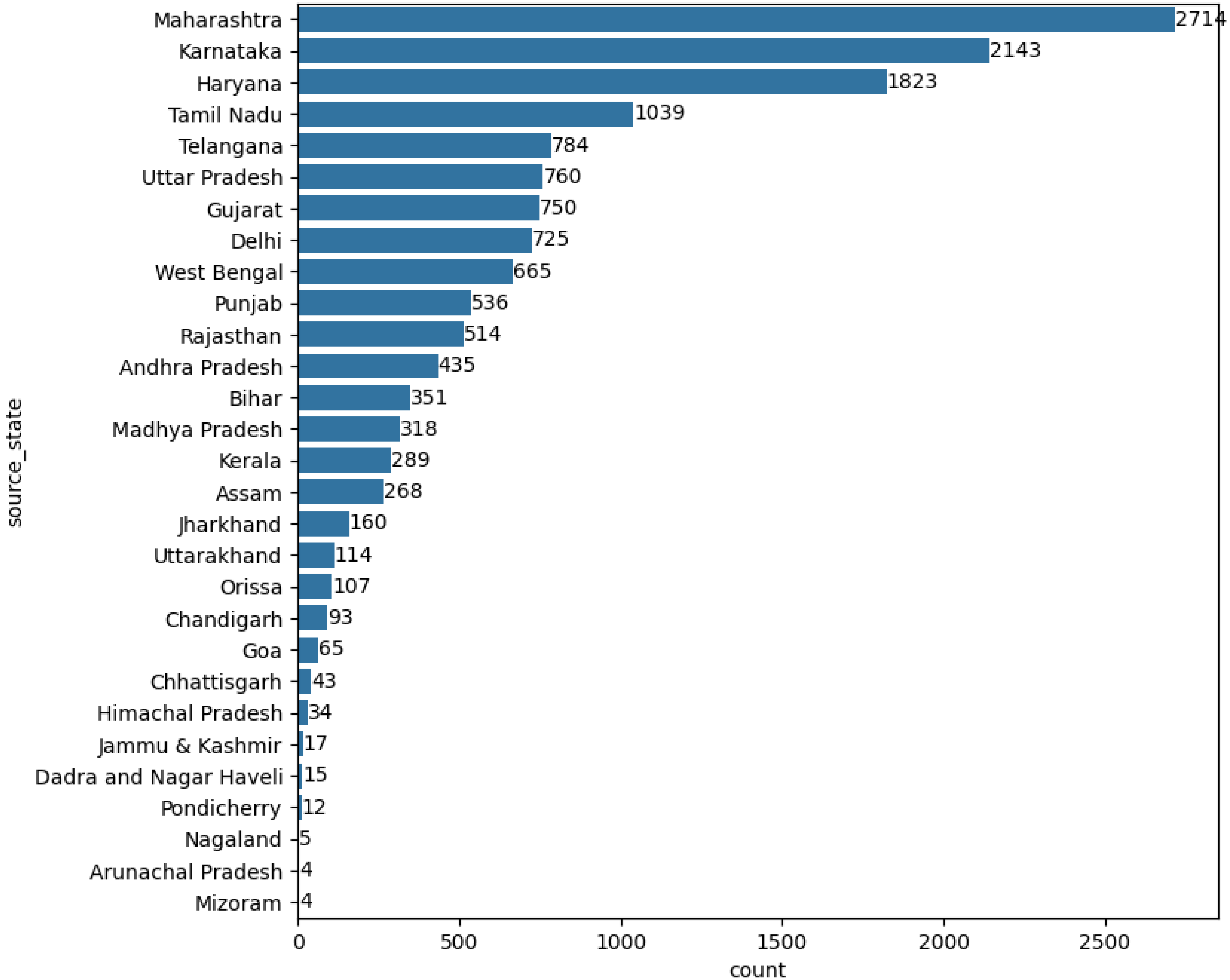
```
    'trip_creation_time' : 'first',
    'route_schedule_uuid' : 'first',
    'route_type' : 'first',
    'start_scan_to_end_scan' : 'sum',
    'actual_distance_to_destination' : 'sum',
    'actual_time' : 'sum',
    'osrm_time' : 'sum',
    'osrm_distance' : 'sum',
    'segment_actual_time_sum' : 'sum',
    'segment_osrm_distance_sum' : 'sum',
    'segment_osrm_time_sum' : 'sum',
    'od_time_diff_hour' : 'sum',
    'source_state' : 'first',
    'destination_state' : 'last',
    'source_city' : 'first',
    'source_place' : 'first',
    'source_code' : 'first',
    'destination_city' : 'last',
    'destination_place' : 'last',
    'destination_code' : 'last',
}
trip_df = segment_df.groupby('trip_uuid').agg(trip_dict).reset_index()


trip_df.head()
```

	trip_uuid	segment_key	data	trip_creation_time	route_schedule_uui
0	trip-153671041653548748	trip-153671041653548748_IND209304AAA_IND000000ACB	training	2018-09-12 00:00:16.535741	thanos::sroute:d7c989b a29b-4a0b-b2f 288cdc6
1	trip-153671042288605164	trip-153671042288605164_IND561203AAB_IND562101AAA	training	2018-09-12 00:00:22.886430	thanos::sroute:3a1b0ab bb0b-4c53-8c5 eb2a2c0
2	trip-153671043369099517	trip-153671043369099517_IND000000ACB_IND160002AAC	training	2018-09-12 00:00:33.691250	thanos::sroute:de5e208 7641-45e6-810 4d9fb1e
3	trip-153671046011330457	trip-153671046011330457_IND400072AAB_IND401104AAA	training	2018-09-12 00:01:00.113710	thanos::sroute:f017649 a679-4597-833 bbd1c7f
4	trip-153671052974046625	trip-153671052974046625_IND583101AAA_IND583201AAA	training	2018-09-12 00:02:09.740725	thanos::sroute:d9f07b1 65e0-4f3b-bec df06134

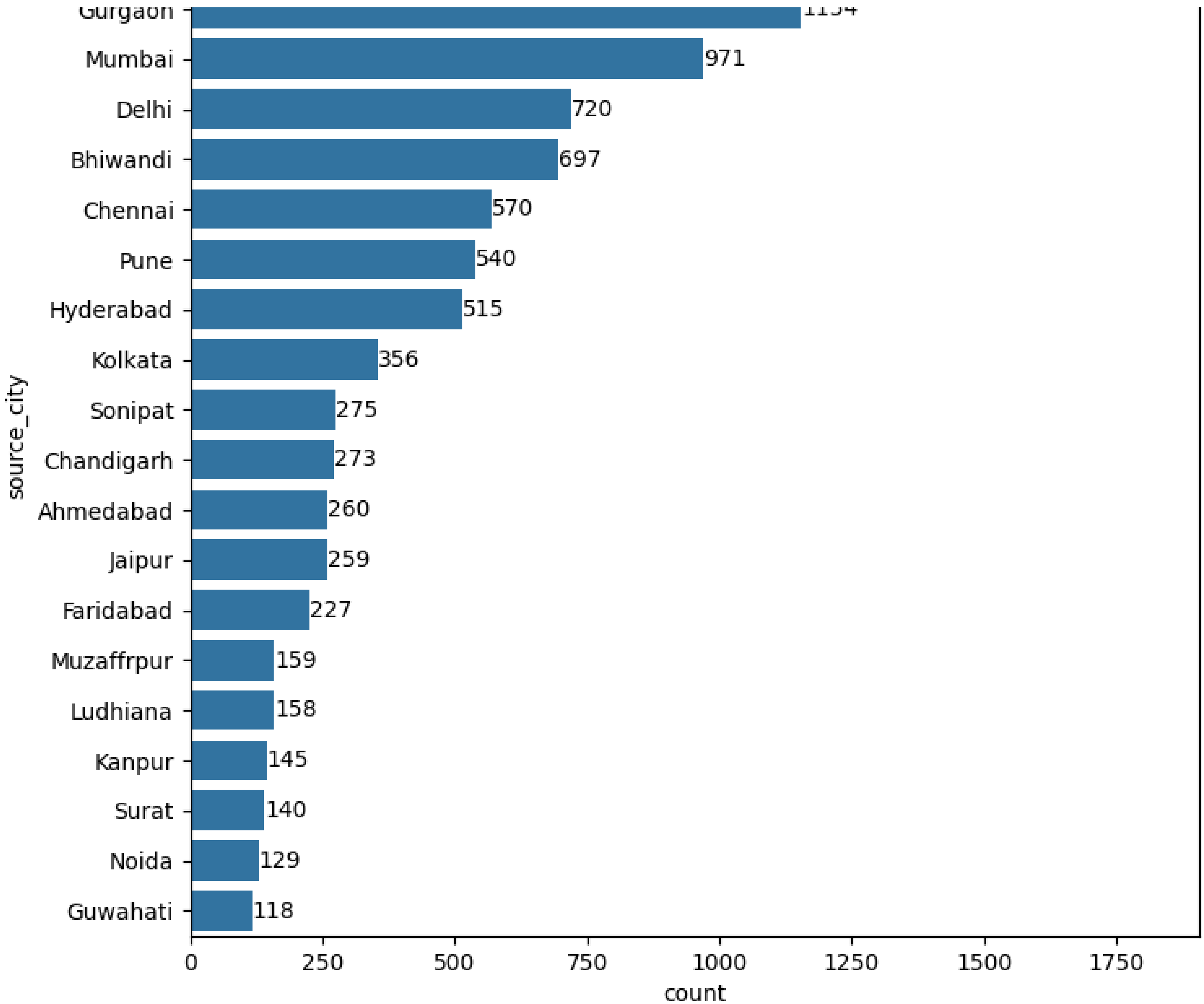
5 rows × 23 columns

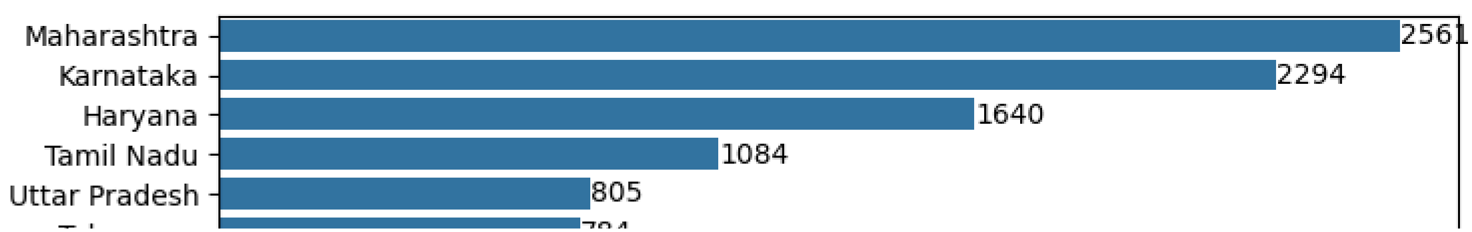
```
plt.figure(figsize=(8,8))
data = trip_df["source_state"]
ax=sns.countplot(y = data, order=data.value_counts().index)
ax.bar_label(ax.containers[0])
plt.show()

plt.figure(figsize=(8,8))
data = trip_df["source_city"]
ax=sns.countplot(y = data, order=data.value_counts()[:20].index)
ax.bar_label(ax.containers[0])
plt.show()

plt.figure(figsize=(8,8))
data = trip_df["destination_state"]
ax=sns.countplot(y = data, order=data.value_counts().index)
```

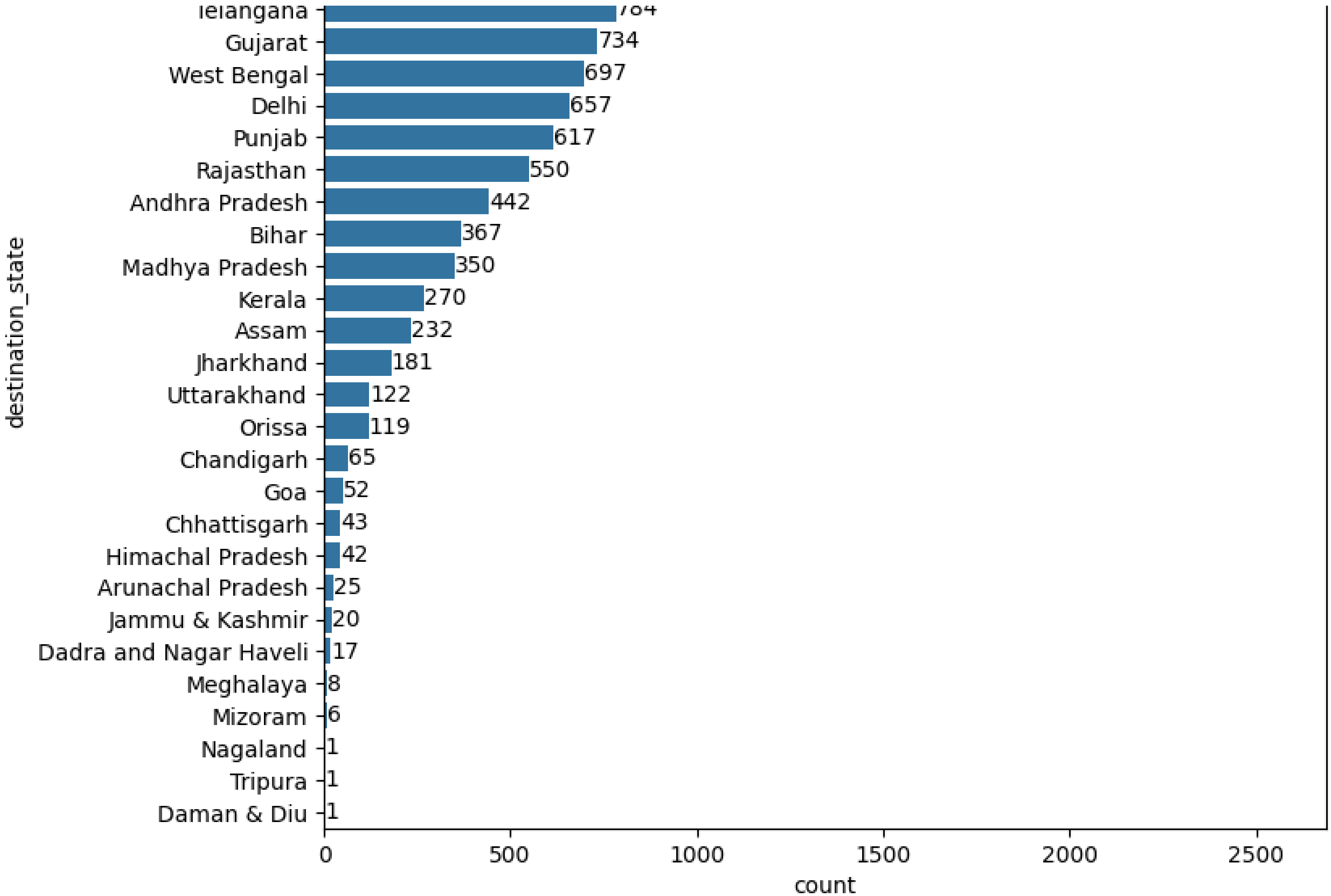
```
ax.bar_label(ax.containers[0])
plt.show()

plt.figure(figsize=(8,8))
data = trip_df["destination_city"]
ax=sns.countplot(y = data, order=data.value_counts()[:20].index)
ax.bar_label(ax.containers[0])
plt.show()
```

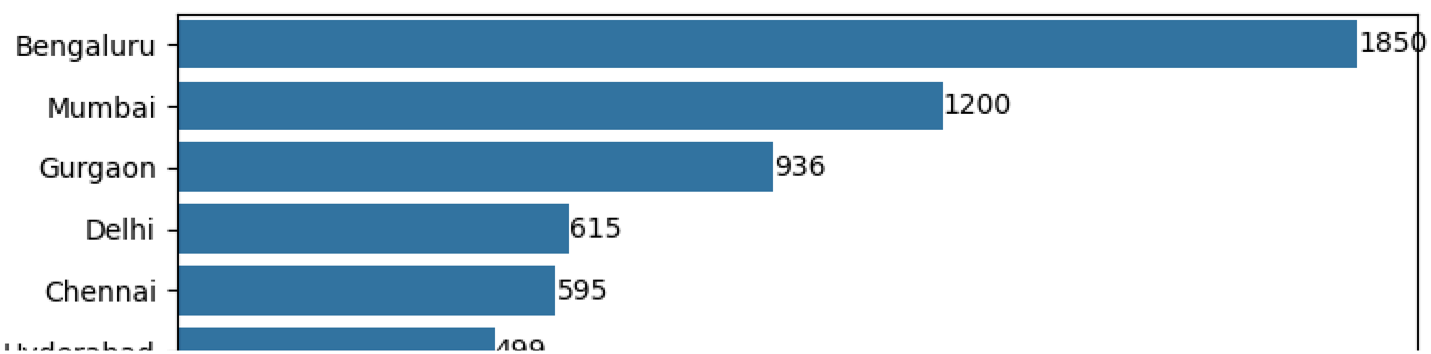

Maharashtra
2714
Karnataka
2143
Haryana
1823
Tamil Nadu
1039
Telangana
784
Uttar Pradesh
760
Gujarat
750
Delhi
725
West Bengal
665
Punjab
536
Rajasthan
514
Andhra Pradesh
435
Bihar
351
Madhya Pradesh
318
Kerala
289
Assam
268
Jharkhand
160
Uttarakhand
114
Orissa
107
Chandigarh
93
Goa
65
Chhattisgarh
43
Himachal Pradesh
34
Jammu & Kashmir
17
Dadra and Nagar Haveli
15
Pondicherry
12
Nagaland
5
Arunachal Pradesh
4
Mizoram
4
source_state
0
500
1000
1500
2000
2500
count

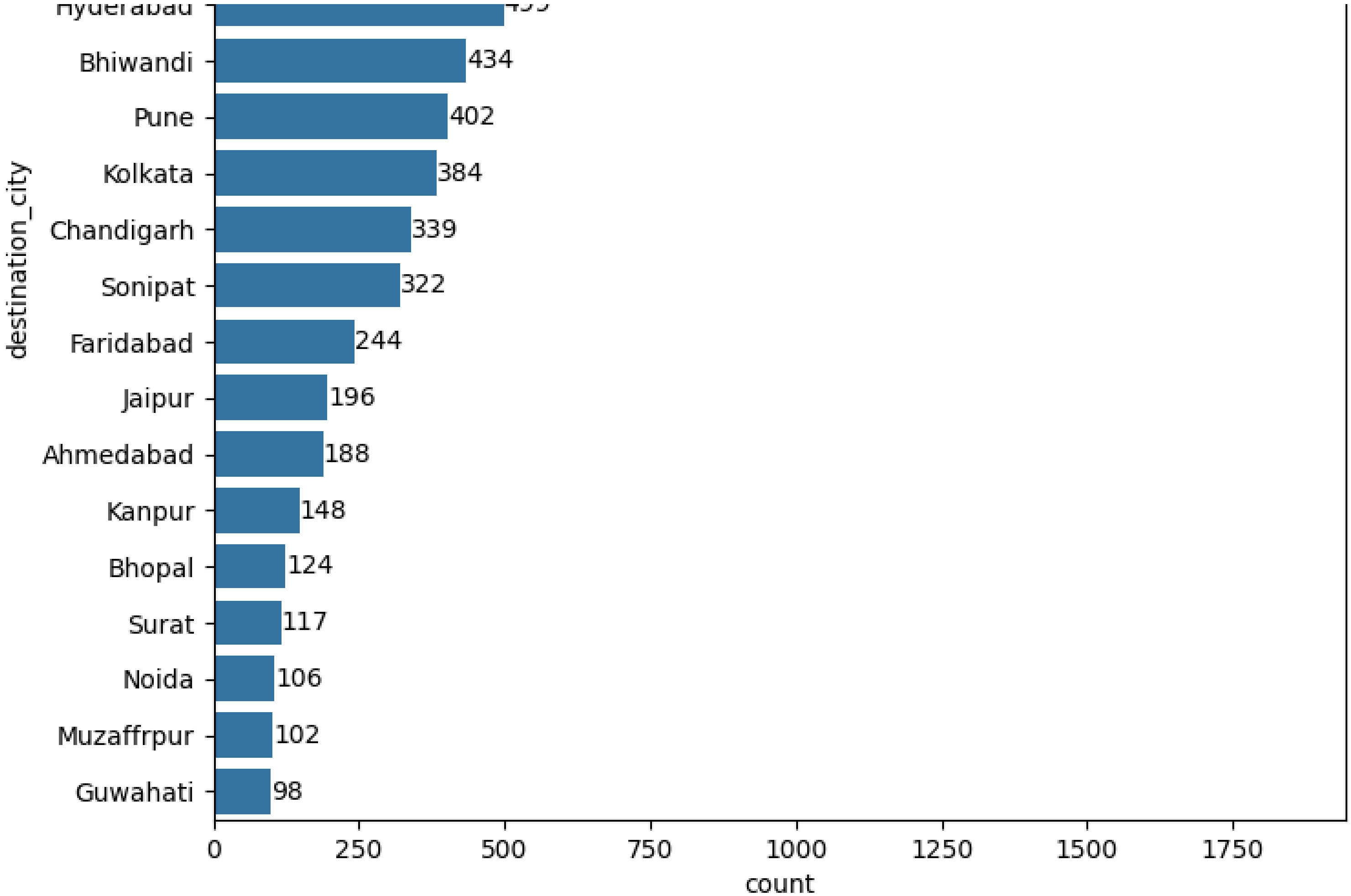

Bengaluru
1815


Mumbai
Delhi
Bhiwandi
Chennai
Pune
Hyderabad
Kolkata
Sonipat
Chandigarh
Ahmedabad
Jaipur
Faridabad
Muzaffrpur
Ludhiana
Kanpur
Surat
Noida
Guwahati
971
720
697
570
540
515
356
275
273
260
259
227
159
158
145
140
129
118
source_city
0
250
500
750
1000
1250
1500
1750
count

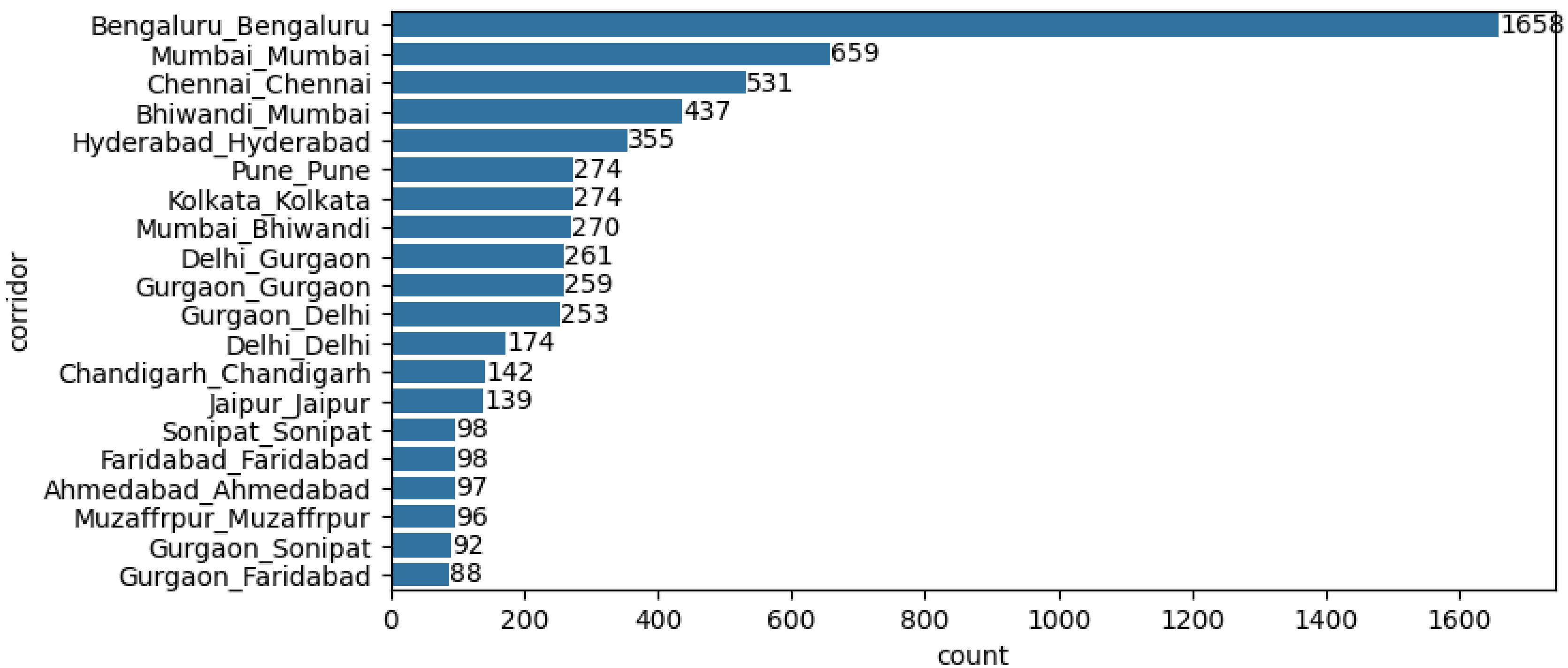

Maharashtra
Karnataka
Haryana
Tamil Nadu
Uttar Pradesh
2561
2294
1640
1084
805


Gujarat 734
West Bengal 697
Delhi 657
Punjab 617
Rajasthan 550
Andhra Pradesh 442
Bihar 367
Madhya Pradesh 350
Kerala 270
Assam 232
Jharkhand 181
Uttarakhand 122
Orissa 119
Chandigarh 65
Goa 52
Chhattisgarh 43
Himachal Pradesh 42
Arunachal Pradesh 25
Jammu & Kashmir 20
Dadra and Nagar Haveli 17
Meghalaya 8
Mizoram 6
Nagaland 1
Tripura 1
Daman & Diu 1
destination_state
0
500
1000
1500
2000
2500
count


Bengaluru 1850
Mumbai 1200
Gurgaon 936
Delhi 615
Chennai 595


Bhiwandi
434
Pune
402
Kolkata
384
Chandigarh
339
Sonipat
322
Faridabad
244
Jaipur
196
Ahmedabad
188
Kanpur
148
Bhopal
124
Surat
117
Noida
106
Muzaffrpur
102
Guwahati
98
destination_city
0
250
500
750
1000
1250
1500
1750
count

```
trip_df["corridor"] = trip_df["source_city"] + '_' + trip_df["destination_city"]
plt.figure(figsize=(8,4))
ax=sns.countplot(y = trip_df["corridor"], order=trip_df["corridor"].value_counts()[:20].index)
ax.bar_label(ax.containers[0])
plt.show()
```

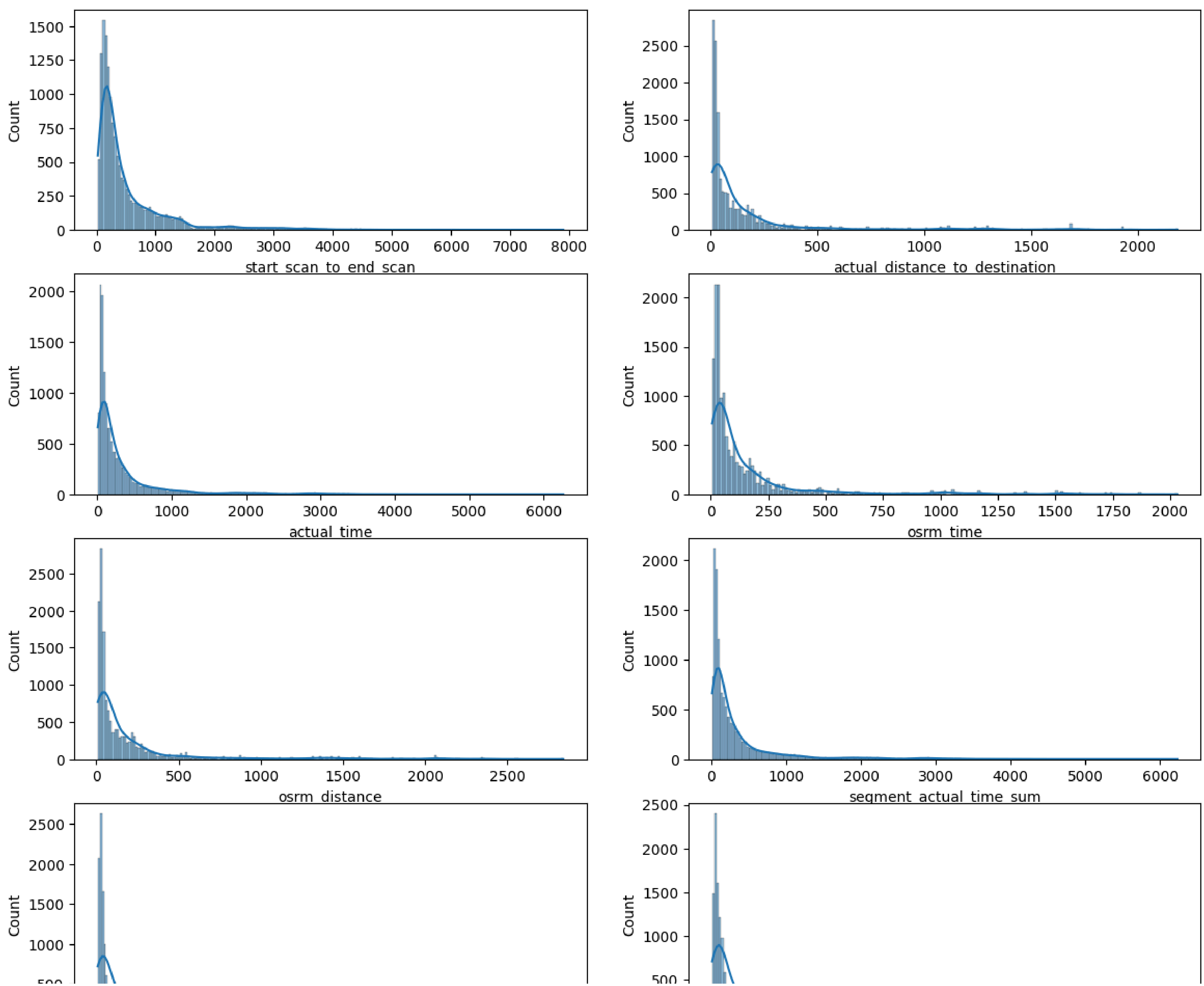


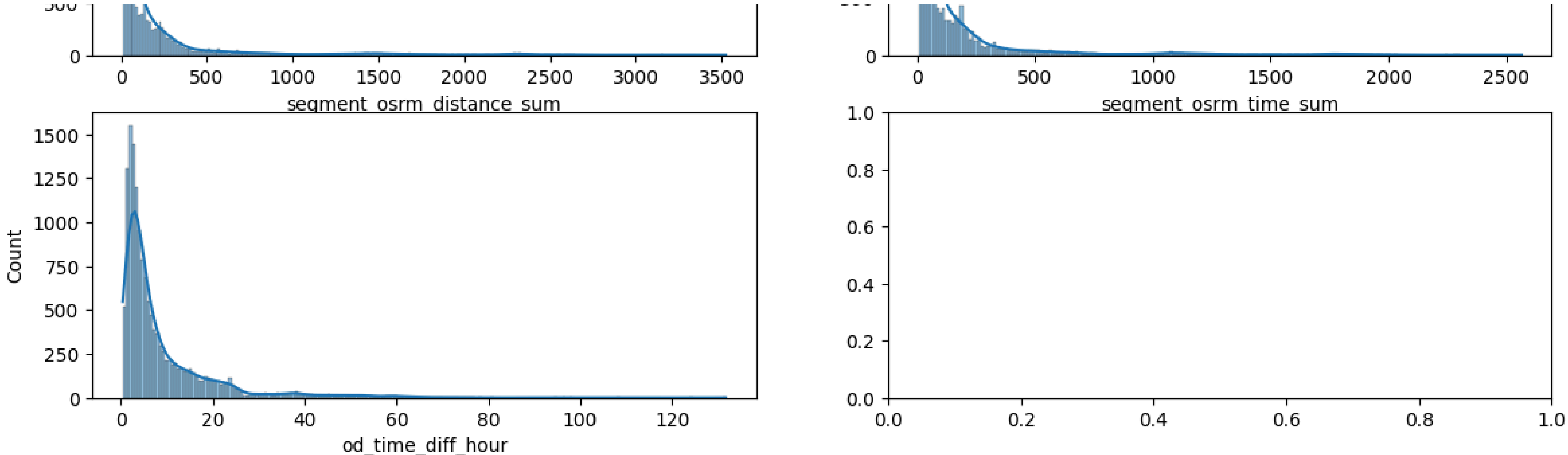

```
Mumbai_Bhiwandi_df = trip_df[((trip_df["corridor"] == "Bhiwandi_Mumbai") | (trip_df["corridor"] == "Mumbai_Bhiwandi"))]
print('Avg time: ', Mumbai_Bhiwandi_df['actual_time'].mean())
print('Avg distance: ', Mumbai_Bhiwandi_df['actual_distance_to_destination'].mean())
```

```
Avg time:  81.78642149929279
Avg distance:  22.210235327215305
```

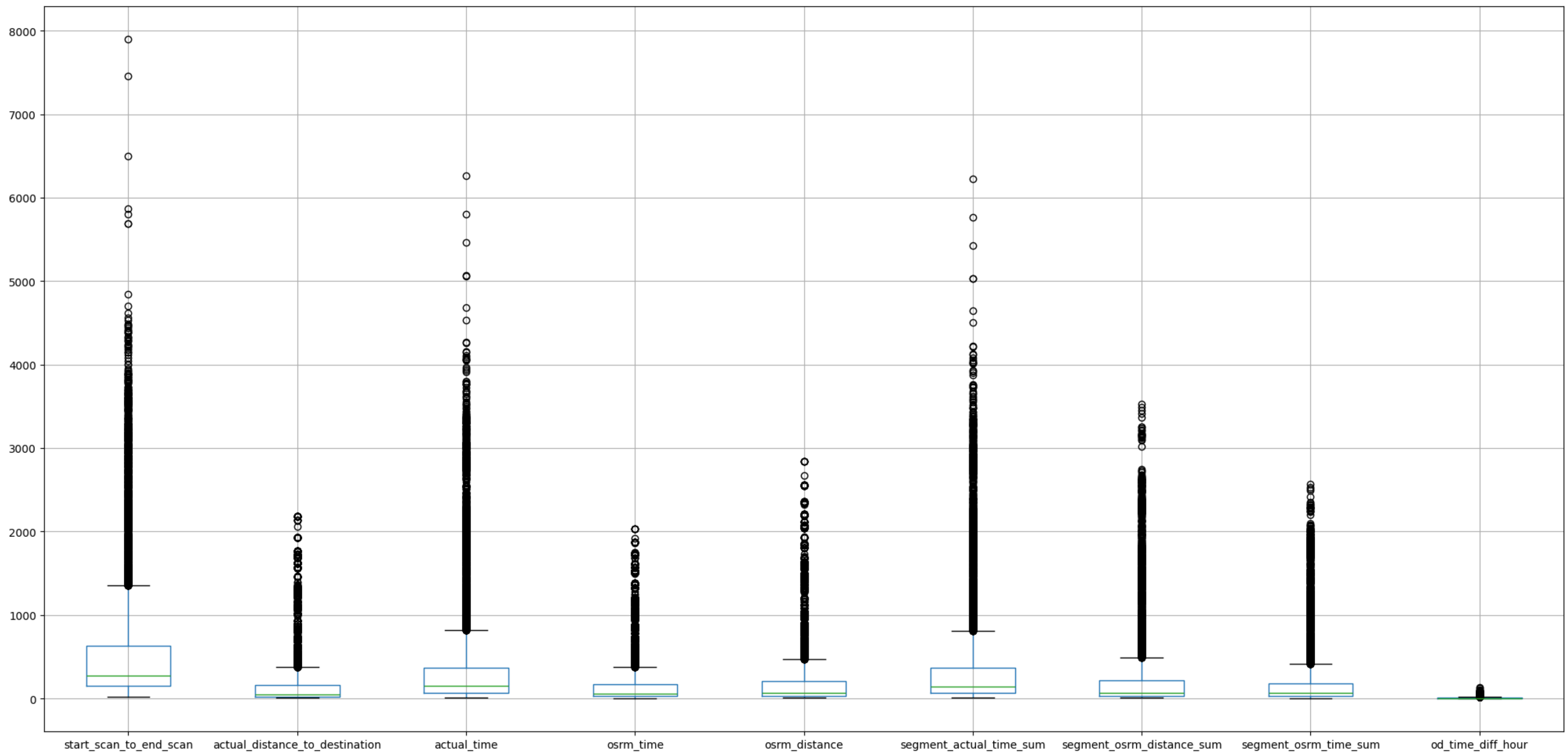
Normalize/ Standardize the numerical features using MinMaxScaler or StandardScaler

```
fig, ax = plt.subplots(nrows=5, ncols=2, figsize = (14, 16))
sns.histplot(data=trip_df, x = "start_scan_to_end_scan", kde=True, ax=ax[0,0])
sns.histplot(data=trip_df, x = "actual_distance_to_destination", kde=True, ax=ax[0,1])
sns.histplot(data=trip_df, x = "actual_time", kde=True, ax=ax[1,0])
sns.histplot(data=trip_df, x = "osrm_time", kde=True, ax=ax[1,1])
sns.histplot(data=trip_df, x = "osrm_distance", kde=True, ax=ax[2,0])
sns.histplot(data=trip_df, x = "segment_actual_time_sum", kde=True, ax=ax[2,1])
sns.histplot(data=trip_df, x = "segment_osrm_distance_sum", kde=True, ax=ax[3,0])
sns.histplot(data=trip_df, x = "segment_osrm_time_sum", kde=True, ax=ax[3,1])
sns.histplot(data=trip_df, x = "od_time_diff_hour", kde=True, ax=ax[4,0])
plt.show()
```


Count
start_scan_to_end_scan
actual_distance_to_destination
actual_time
osrm_time
osrm_distance
segment_actual_time_sum


segment_osrm_distance_sum
segment_osrm_time_sum
Count
od_time_diff_hour

```
trip_numerical_columns = ['start_scan_to_end_scan', 'actual_distance_to_destination',
                          'actual_time', 'osrm_time', 'osrm_distance', 'segment_actual_time_sum',
                          'segment_osrm_distance_sum', 'segment_osrm_time_sum', 'od_time_diff_hour']
trip_df[trip_numerical_columns].boxplot(figsize=(25,12))
plt.show()
```


8000
7000
6000
5000
4000
3000
2000
1000
0
start_scan_to_end_scan
actual_distance_to_destination
actual_time
osrm_time
osrm_distance
segment_actual_time_sum
segment_osrm_distance_sum
segment_osrm_time_sum
od_time_diff_hour

None of the data is gaussian, so we will use MinMaxScaler

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
scaler.fit(trip_df[trip_numerical_columns])
trip_df[trip_numerical_columns] = scaler.transform(trip_df[trip_numerical_columns])
```

```
trip_df.describe()
```

	trip_creation_time	start_scan_to_end_scan	actual_distance_to_destination	actual_time	osrm_time	osrm_dist
count	14787	14787.000000	14787.000000	14787.000000	14787.000000	14787.00
mean	2018-09-22 12:26:28.269885696	0.064308	0.071222	0.055516	0.076501	0.06
min	2018-09-12 00:00:16.535741	0.000000	0.000000	0.000000	0.000000	0.00
25%	2018-09-17 02:38:18.128431872	0.016000	0.006326	0.009271	0.011352	0.00
50%	2018-09-22 03:39:19.609193984	0.032508	0.018041	0.022219	0.026654	0.01