Business Case: Delhivery - Feature Engineering

About Delhivery

In [475]:

```
import numpy as np
import pandas as pd
pd.set_option('display.max_columns', 500)

import matplotlib.pyplot as plt
import seaborn as sns
sns.set(style='whitegrid')

from statsmodels.graphics.gofplots import qqplot

from scipy.stats import norm
from scipy.stats import ttest_ind
from scipy.stats import shapiro, kstest

from sklearn.preprocessing import MinMaxScaler,StandardScaler
```

In [336]:

```
def shapiro_test(series):
    print("Mean : ", round(series.mean(),2),", Standard deviation : ",round(series.

    # calling function for shapiro test
    test_stat, p_value = shapiro(series)

    print("p-value : ", p_value)
    if p_value < 0.05:
        print("Reject H0")
        print("Data is not Gaussian")
    else:
        print("Fail to reject H0")
        print("Data is Gaussian")


def kstest_test(series):
    mu = series.mean()
    sigma = series.std()
    print("Mean : ", round(series.mean(),2),", Standard deviation : ",round(series.

    # calling function for ks-test
    test_stat, p_value = kstest(
        series,
        norm.cdf,
        args=(mu, sigma)
    )

    print("p-value : ", p_value)
    if p_value < 0.05:
        print("Reject H0")
        print("Data is not Gaussian")
    else:
        print("Fail to reject H0")
        print("Data is Gaussian")


def ttest_ind_test(series1,series2,alternative='two-sided'):
    print("Series1 metrics : ")
    print("Mean : ", round(series1.mean(),2),", Standard deviation : ",round(series
    print("Series2 metrics : ")
    print("Mean : ", round(series2.mean(),2),", Standard deviation : ",round(series

    # calling function for t-test for 2 independent samples
    t_stat, p_value = ttest_ind(series1,series2,alternative=alternative)
    print("p-value : ", p_value)
    if p_value < 0.10:
        print("Reject H0")
    else:
        print("Fail to reject H0")
```

In [ ]:

```

```

1. Define Problem Statement and perform Exploratory Data Analysis (10 points)

Business Problem:

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

In [2]:

```
df = pd.read_csv("delhivery_data.csv")
```

In [14]:

```
df.head()
```

Out[14]:

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_
0	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND3881:
1	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND3881:
2	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND3881:
3	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND3881:
4	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND3881:

In [3]:

```
df.shape
```

Out[3]:

```
(144867, 24)
```

In [4]:

```
df.dtypes
```

Out[4]:

```
data                                object
trip_creation_time                  object
route_schedule_uuid                 object
route_type                          object
trip_uuid                           object
source_center                       object
source_name                         object
destination_center                  object
destination_name                    object
od_start_time                       object
od_end_time                         object
start_scan_to_end_scan             float64
is_cutoff                             bool
cutoff_factor                        int64
cutoff_timestamp                    object
actual_distance_to_destination     float64
actual_time                        float64
osrm_time                          float64
```

In [35]:

```
# converting all the dateime fields to datetime form object
df['trip_creation_time'] = pd.to_datetime(df['trip_creation_time'])
df['od_start_time'] = pd.to_datetime(df['od_start_time'])
df['od_end_time'] = pd.to_datetime(df['od_end_time'])
df['cutoff_timestamp'] = pd.to_datetime(df['cutoff_timestamp'])
```

In [10]:

```
df.isna().sum()
```

Out[10]:

```
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
```

In [9]:

```
(100*df.isna().sum()/df.shape[0]).round(2)
```

Out[9]:

```
data                              0.00
trip_creation_time                0.00
route_schedule_uuid               0.00
route_type                        0.00
trip_uuid                         0.00
source_center                     0.00
source_name                       0.20
destination_center                0.00
destination_name                  0.18
od_start_time                     0.00
od_end_time                       0.00
start_scan_to_end_scan            0.00
is_cutoff                         0.00
cutoff_factor                     0.00
cutoff_timestamp                  0.00
actual_distance_to_destination    0.00
actual_time                       0.00
osrm_time                         0.00
```

In [ ]:

3. Merging of rows and aggregation of fields (10 Points)

In [15]:

```
df.head()
```

Out[15]:

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_
0	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND38812
1	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND38812
2	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND38812
3	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND38812
4	training	2018-09-20 02:35:36.476840	thanos::sroute:eb7bfc78-b351-4c0e-a951-fa3d5c3...	Carting	trip-153741093647649320	IND38812

In [494]:

```
## grouping the data summing up all the subtrips to complete trips
```

In [28]:

```
groupby_dict = {
    'data' : 'first',

    'trip_creation_time' : 'first',
    'route_schedule_uuid' : 'first',
    'route_type' : 'first',
    'trip_uuid' : 'first',

    'source_center' : 'first',
    'source_name' : 'first',

    'destination_center' : 'last',
    'destination_name' : 'last',

    'od_start_time' : 'first',
    'od_end_time' : 'first',
    'start_scan_to_end_scan' : 'first',

    'actual_distance_to_destination' : 'last',
    'actual_time' : 'last',

    'osrm_time' : 'last',
    'osrm_distance' : 'last',

    'segment_actual_time' : 'sum',
    'segment_osrm_distance' : 'sum',
    'segment_osrm_time' : 'sum'
}
```

In [29]:

```
df['tsd'] = df['trip_uuid']+df['source_center']+df['destination_center']
```

In [37]:

```
df_tsd = df.groupby('tsd').agg(groupby_dict).reset_index()
```

In [ ]:

In [38]:

```
df_tsd.head()
```

Out[38]:

	tsd	data	trip_creation_time	route_schedul
0	trip-153671041653548748IND209304AAAIND000000ACB	training	2018-09-12 00:00:16.535741	thanos::sroute:d7c a29b-4a0 288
1	trip-153671041653548748IND462022AAAIND209304AAA	training	2018-09-12 00:00:16.535741	thanos::sroute:d7c a29b-4a0 288
2	trip-153671042288605164IND561203AABIND562101AAA	training	2018-09-12 00:00:22.886430	thanos::sroute:3a1 bb0b-4c5 eb2
3	trip-153671042288605164IND572101AAAIND561203AAB	training	2018-09-12 00:00:22.886430	thanos::sroute:3a1 bb0b-4c5 eb2
4	trip-153671043369099517IND000000ACBIND160002AAC	training	2018-09-12 00:00:33.691250	thanos::sroute:de5 7641-45e 4d

In [48]:

```
df_tsd['od_time_diff'] = ((df_tsd['od_end_time'] - df_tsd['od_start_time']).dt.tota
```

In [49]:

```
df_tsd.head()
```

Out[49]:

osrm_distance	segment_actual_time	segment_osrm_distance	segment_osrm_time	od_time_diff
446.5496	728.0	670.6205	534.0	1260.604421
544.8027	820.0	649.8528	474.0	999.505379
28.1994	46.0	28.1995	26.0	58.832388
56.9116	95.0	55.9899	39.0	122.779486
281.2109	608.0	317.7408	231.0	834.638929

In [ ]:

In [55]:

```
groupby_trip_dict = {
    'data' : 'first',

    'trip_creation_time' : 'first',
    'route_schedule_uuid' : 'first',
    'route_type' : 'first',

    'source_center' : 'first',
    'source_name' : 'first',

    'destination_center' : 'last',
    'destination_name' : 'last',

    'start_scan_to_end_scan' : 'sum',
    'od_time_diff' : 'sum',

    'actual_distance_to_destination' : 'sum',
    'actual_time' : 'sum',
    'osrm_time' : 'sum',
    'osrm_distance' : 'sum',

    'segment_actual_time' : 'sum',
    'segment_osrm_distance' : 'sum',
    'segment_osrm_time' : 'sum',
}
```

In [59]:

```
df_trip = df_tsd.groupby('trip_uuid').agg(groupby_trip_dict).reset_index()
```

In [61]:

```
df_trip.head()
```

Out[61]:

	trip_uuid	data	trip_creation_time	route_schedule_uuid	route_type	source_
0	trip-153671041653548748	training	2018-09-12 00:00:16.535741	thanos::sroute:d7c989ba-a29b-4a0b-b2f4-288cdc6...	FTL	IND2093
1	trip-153671042288605164	training	2018-09-12 00:00:22.886430	thanos::sroute:3a1b0ab2-bb0b-4c53-8c59-eb2a2c0...	Carting	IND5612
2	trip-153671043369099517	training	2018-09-12 00:00:33.691250	thanos::sroute:de5e208e-7641-45e6-8100-4d9fb1e...	FTL	IND0000
3	trip-153671046011330457	training	2018-09-12 00:01:00.113710	thanos::sroute:f0176492-a679-4597-8332-bbd1c7f...	Carting	IND4000
4	trip-153671052974046625	training	2018-09-12 00:02:09.740725	thanos::sroute:d9f07b12-65e0-4f3b-bec8-df06134...	FTL	IND5831

In [62]:

```
df_trip.shape
```

Out[62]:

```
(14817, 18)
```

In [63]:

```
df_trip.dtypes
```

Out[63]:

```
trip_uuid                                   object
data                                        object
trip_creation_time                  datetime64[ns]
route_schedule_uuid                         object
route_type                                  object
source_center                               object
source_name                                 object
destination_center                          object
destination_name                            object
start_scan_to_end_scan                     float64
od_time_diff                               float64
actual_distance_to_destination             float64
actual_time                                float64
osrm_time                                  float64
osrm_distance                              float64
segment_actual_time                        float64
segment_osrm_distance                      float64
segment_osrm_time                          float64
dtype: object
```

In [65]:

```
df_trip.isna().sum()
```

Out[65]:

```
trip_uuid                          0
data                               0
trip_creation_time                 0
route_schedule_uuid                0
route_type                         0
source_center                      0
source_name                       10
destination_center                 0
destination_name                   8
start_scan_to_end_scan             0
od_time_diff                       0
actual_distance_to_destination     0
actual_time                        0
osrm_time                          0
osrm_distance                      0
segment_actual_time                0
segment_osrm_distance              0
segment_osrm_time                  0
dtype: int64
```

In [81]:

```
(df_trip.describe().T).round(0)
```

Out[81]:

	count	mean	std	min	25%	50%	75%	max
start_scan_to_end_scan	14817.0	531.0	659.0	23.0	149.0	280.0	637.0	7898.0
od_time_diff	14817.0	532.0	659.0	23.0	150.0	281.0	638.0	7899.0
actual_distance_to_destination	14817.0	164.0	305.0	9.0	23.0	48.0	165.0	2187.0
actual_time	14817.0	357.0	561.0	9.0	67.0	149.0	370.0	6265.0
osrm_time	14817.0	161.0	271.0	6.0	29.0	60.0	168.0	2032.0
osrm_distance	14817.0	204.0	370.0	9.0	31.0	66.0	208.0	2840.0
segment_actual_time	14817.0	354.0	556.0	9.0	66.0	147.0	367.0	6230.0
segment_osrm_distance	14817.0	223.0	417.0	9.0	33.0	70.0	219.0	3524.0
segment_osrm_time	14817.0	181.0	315.0	6.0	31.0	65.0	185.0	2564.0

In [83]:

```
(df_trip.describe(include='object').T)
```

Out[83]:

	count	unique	top	freq
trip_uuid	14817	14817	trip-153671041653548748	1
data	14817	2	training	10654
route_schedule_uuid	14817	1504	thanos::sroute:a16bfa03-3462-4bce-9c82-5784c7d...	53
route_type	14817	2	Carting	8908
source_center	14817	938	IND000000ACB	1063
source_name	14807	933	Gurgaon_Bilaspur_HB (Haryana)	1063
destination_center	14817	1042	IND000000ACB	821
destination_name	14809	1034	Gurgaon_Bilaspur_HB (Haryana)	821

In [ ]:

```

```

In [ ]:

```

```

Continuing Question 1.

5. Missing values Treatment & Outlier treatment (10 Points)

In [232]:

```
# As we say in the vizualization the data is heavily skewed towards right, we do lo
# looks normalized ,so we can do the hypothesis testings perfectly with all the ter

time_distance = ['start_scan_to_end_scan', 'od_time_diff',
       'actual_distance_to_destination', 'actual_time', 'osrm_time',
       'osrm_distance', 'segment_actual_time', 'segment_osrm_distance',
       'segment_osrm_time']

for l in time_distance:
    df_trip[l+'_log'] = df_trip[l].apply(np.log)
```

In [ ]:

```

```

In [ ]:

```

```

filling nulls in source and destination names by checking the already existing source and destination centers

In [236]:

```
df_s = df_trip.loc[~df_trip['source_name'].isna(),['source_center','source_name']]
df_d = df_trip.loc[~df_trip['destination_name'].isna(),['destination_center','desti
```

In [237]:

```
df_d.columns = ['source_center','source_name']
```

In [238]:

```
id_loc_map = dict(pd.concat([df_s,df_d],axis=0).drop_duplicates().values)
```

In [247]:

```
def fill_loc_na(row):
    if pd.isna(row['source_name']) and row['source_center'] in id_loc_map:
            return id_loc_map[row['source_center']]
    else:
        return row['source_name']
def fill_loc_nad(row):
    if pd.isna(row['destination_name']) and row['destination_center'] in id_loc_map
            return id_loc_map[row['destination_center']]
    else:
        return row['destination_name']
```

In [252]:

```
# ([i for i in list(df_trip.loc[df_trip['source_name'].isna(),'source_center'].valu
```

In [250]:

```
df_trip['source_name'] = df_trip.apply(fill_loc_na,axis=1)
df_trip['destination_name'] = df_trip.apply(fill_loc_nad,axis=1)
```

In [251]:

```
df_trip.isna().sum()  # 18 nulls reduced to 8 nulls
```

Out[251]:

```
trip_uuid                         0
data                              0
trip_creation_time                0
route_schedule_uuid               0
route_type                        0
source_center                     0
source_name                       1
destination_center                0
destination_name                  7
start_scan_to_end_scan            0
od_time_diff                      0
actual_distance_to_destination    0
actual_time                       0
osrm_time                         0
osrm_distance                     0
segment_actual_time               0
segment_osrm_distance             0
segment_osrm_time                 0
```

In [254]:

```
# dropping the rest of the missing value data as its very less
df_trip.dropna(subset=['source_name','destination_name'],inplace=True)
```

In [ ]:

In [ ]:

In [ ]:

In [162]:

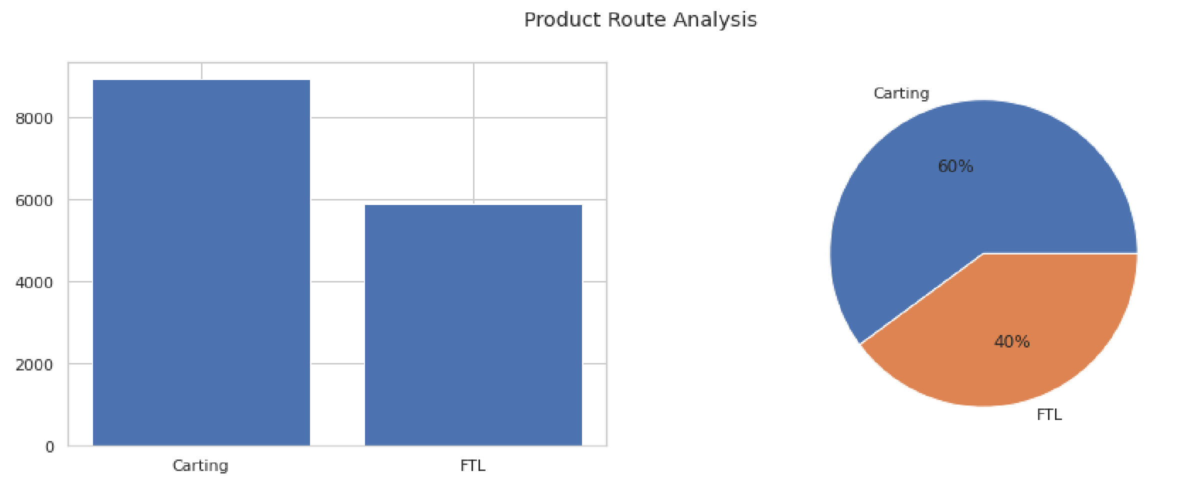
```
df_trip.head()
```

Out[162]:

ource_code	source_state	destination_city	destination_place	destination_code	destination_state
H 6	Uttar Pradesh	Kanpur	Central	H 6	Uttar Pradesh
D	Karnataka	Doddablpur	ChikaDPP	D	Karnataka
HB	Haryana	Gurgaon	Bilaspur	HB	Haryana
	Maharashtra	Mumbai	MiraRd	IP	Maharashtra
	Karnataka	Sandur	WrdN1DPP	D	Karnataka

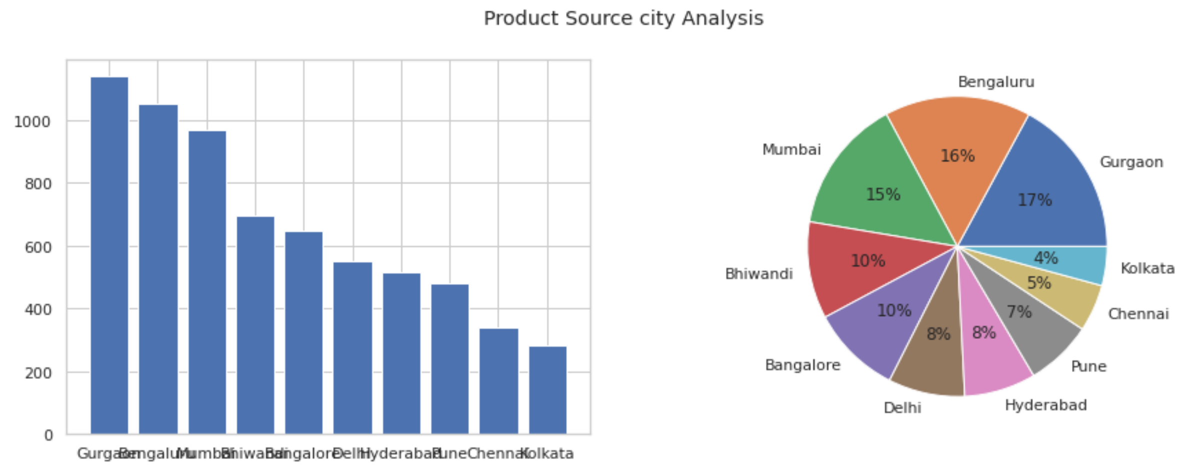
In [171]:

```
s_vc = df_trip['route_type'].value_counts()
fig, axs = plt.subplots(1, 2 , figsize =(15, 5))
fig.suptitle('Product Route Analysis')
axs[0].bar(s_vc.index,s_vc.values)
axs[1].pie(s_vc.values,labels=s_vc.index, autopct='%1.0f%%')
plt.show()
```



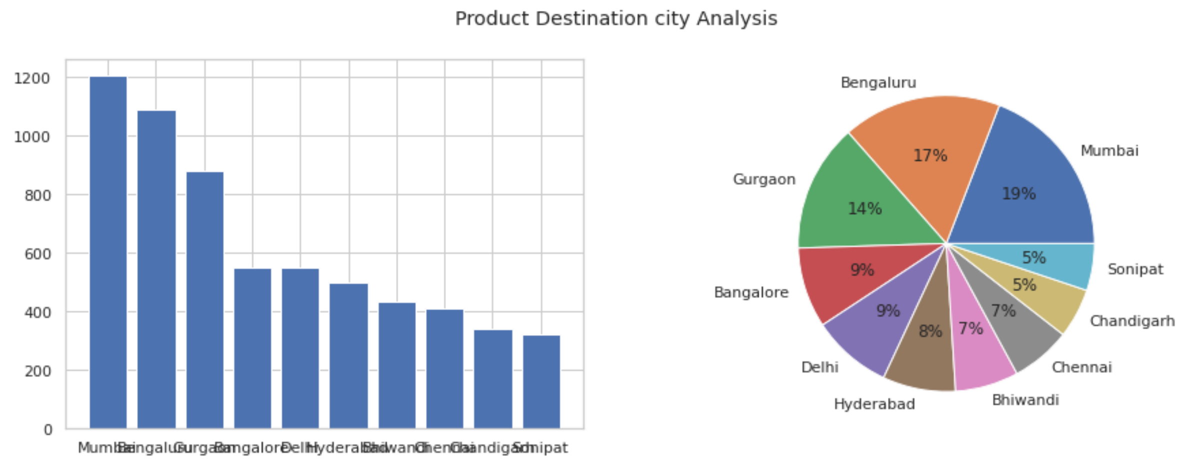
In [172]:

```
s_vc = df_trip['source_city'].value_counts()[:10]
fig, axs = plt.subplots(1, 2 , figsize =(15, 5))
fig.suptitle('Product Source city Analysis')
axs[0].bar(s_vc.index,s_vc.values)
axs[1].pie(s_vc.values,labels=s_vc.index, autopct='%1.0f%%')
plt.show()
```



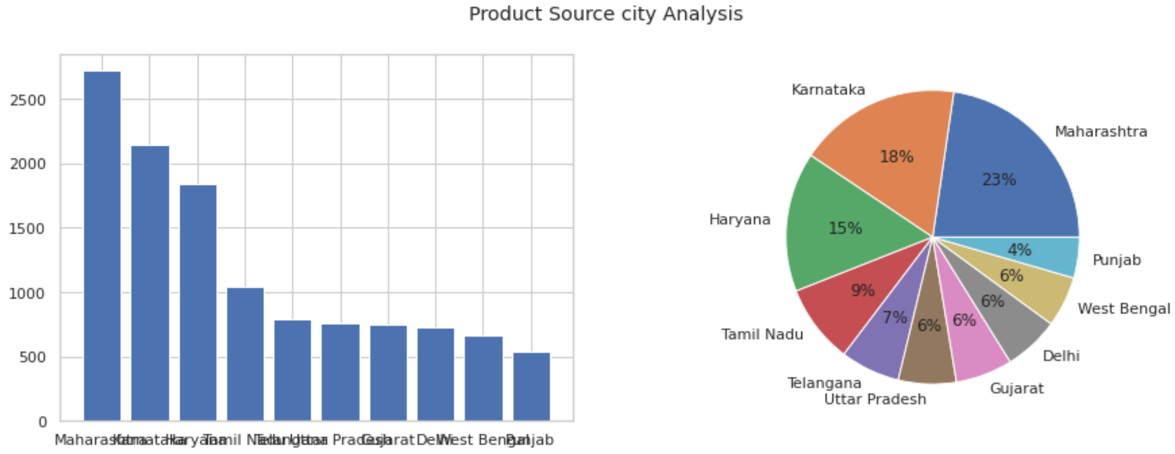
In [173]:

```
s_vc = df_trip['destination_city'].value_counts()[:10]
fig, axs = plt.subplots(1, 2 , figsize =(15, 5))
fig.suptitle('Product Destination city Analysis')
axs[0].bar(s_vc.index,s_vc.values)
axs[1].pie(s_vc.values,labels=s_vc.index, autopct='%1.0f%%')
plt.show()
```



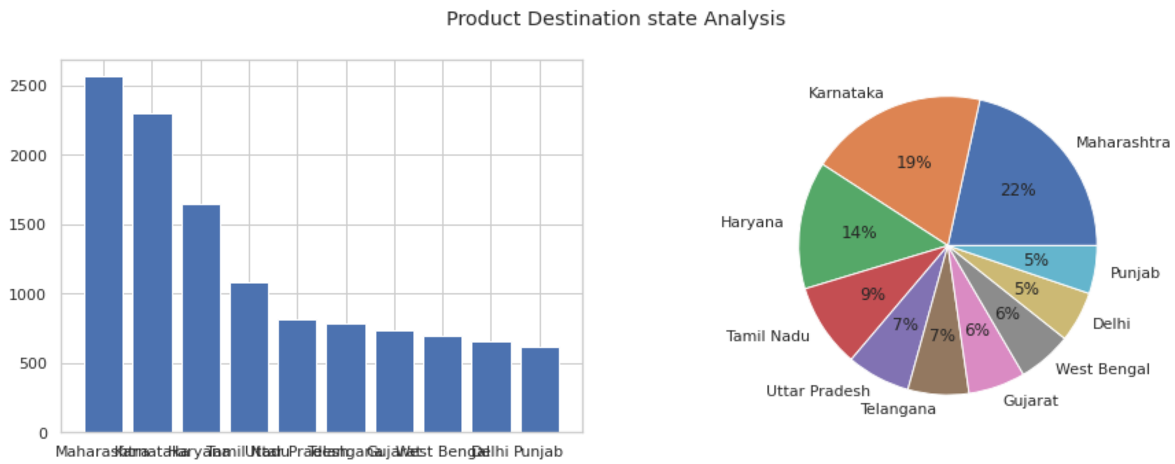
In [174]:

```
s_vc = df_trip['source_state'].value_counts()[:10]
fig, axs = plt.subplots(1, 2 , figsize =(15, 5))
fig.suptitle('Product Source city Analysis')
axs[0].bar(s_vc.index,s_vc.values)
axs[1].pie(s_vc.values,labels=s_vc.index, autopct='%1.0f%%')
plt.show()
```



In [175]:

```
s_vc = df_trip['destination_state'].value_counts()[:10]
fig, axs = plt.subplots(1, 2 , figsize =(15, 5))
fig.suptitle('Product Destination state Analysis')
axs[0].bar(s_vc.index,s_vc.values)
axs[1].pie(s_vc.values,labels=s_vc.index, autopct='%1.0f%%')
plt.show()
```



In [177]:

```
df_trip.head()
```

Out[177]:

	trip_uuid	data	trip_creation_time	route_schedule_uuid	route_type	source_
0	trip-153671041653548748	training	2018-09-12 00:00:16.535741	thanos::sroute:d7c989ba-a29b-4a0b-b2f4-288cdc6...	FTL	IND2093
1	trip-153671042288605164	training	2018-09-12 00:00:22.886430	thanos::sroute:3a1b0ab2-bb0b-4c53-8c59-eb2a2c0...	Carting	IND5612
2	trip-153671043369099517	training	2018-09-12 00:00:33.691250	thanos::sroute:de5e208e-7641-45e6-8100-4d9fb1e...	FTL	IND0000
3	trip-153671046011330457	training	2018-09-12 00:01:00.113710	thanos::sroute:f0176492-a679-4597-8332-bbd1c7f...	Carting	IND4000
4	trip-153671052974046625	training	2018-09-12 00:02:09.740725	thanos::sroute:d9f07b12-65e0-4f3b-bec8-df06134...	FTL	IND5831

In [203]:

```
df_trip.columns
```

Out[203]:

```
Index(['trip_uuid', 'data', 'trip_creation_time', 'route_schedule_uui
d',
       'route_type', 'source_center', 'source_name', 'destination_cent
er',
       'destination_name', 'start_scan_to_end_scan', 'od_time_diff',
       'actual_distance_to_destination', 'actual_time', 'osrm_time',
       'osrm_distance', 'segment_actual_time', 'segment_osrm_distanc
e',
       'segment_osrm_time', 'trip_creation_hour', 'source_city',
       'source_place', 'source_code', 'source_state', 'destination_cit
y',
       'destination_place', 'destination_code', 'destination_state'],
      dtype='object')
```

In [206]:

```

```

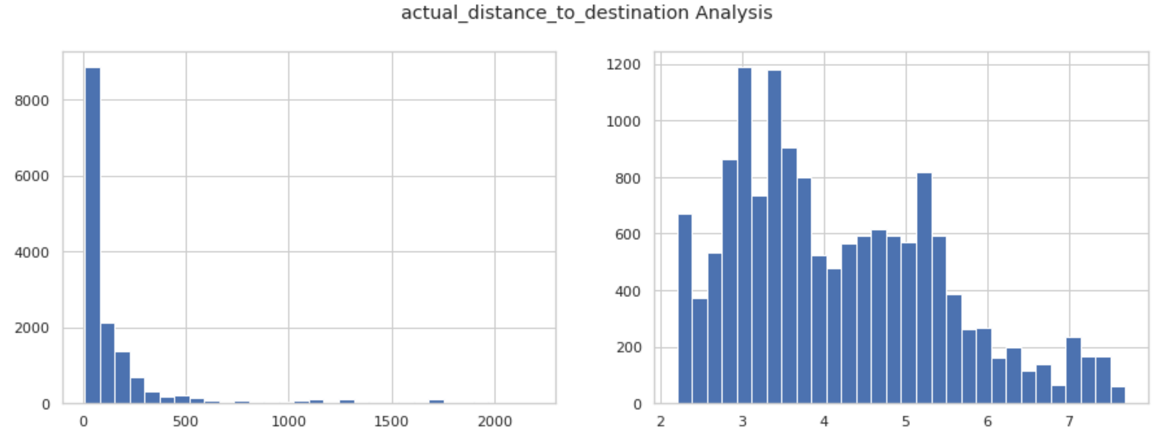
In [ ]:

```

```

In [207]:

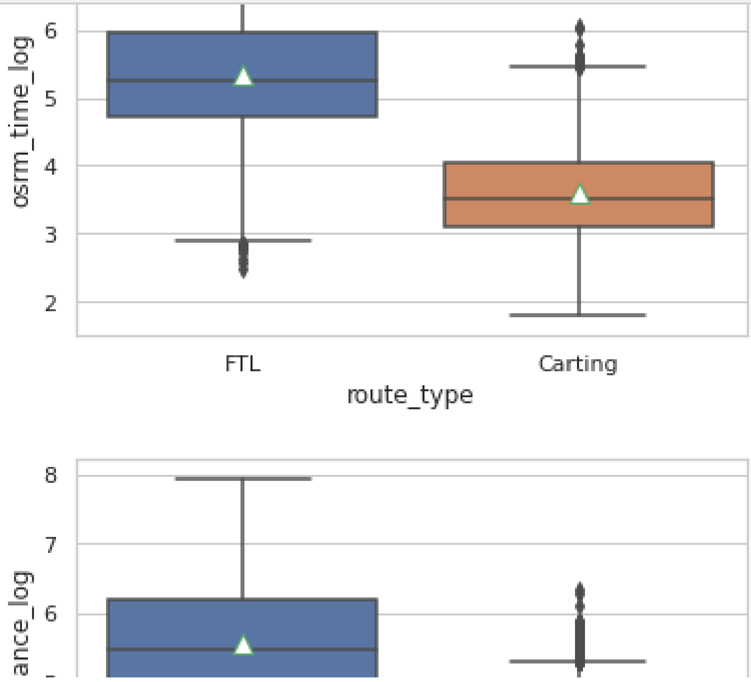
```
for key in time_distance:
    fig, axs = plt.subplots(1, 2 , figsize =(15, 5))
    fig.suptitle(key +' Analysis')
    axs[0].hist(df_trip[key],bins=30)
    axs[1].hist(df_trip[key+'_log'],bins=30)
    plt.show()
```



In [ ]:

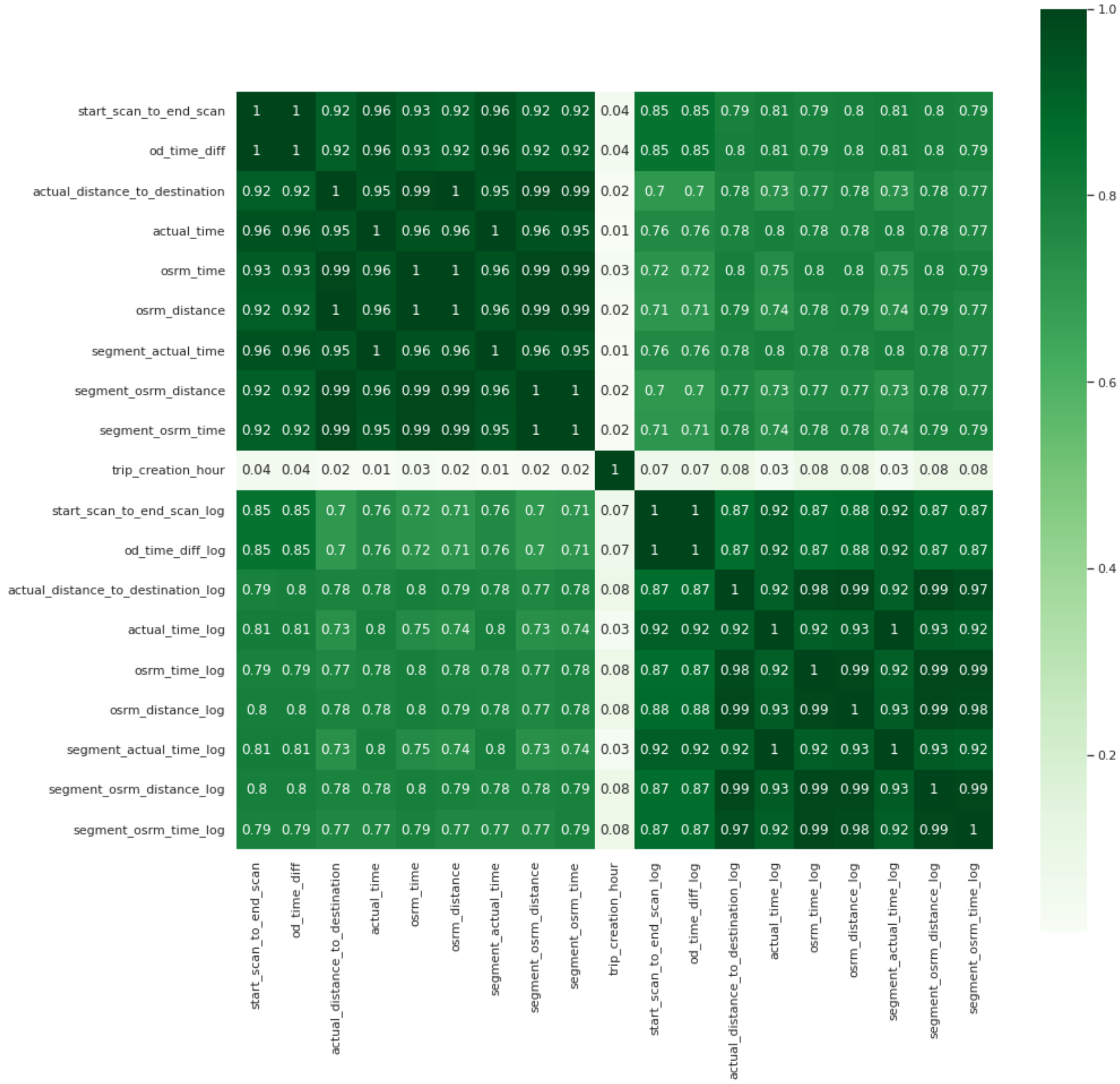
In [214]:

```
key in time_distance:
sns.boxplot(x='route_type',y=key+'_log',data=df_trip,showmeans=True,meanprops={"mark
plt.show()
```



In [218]:

```
plt.figure(figsize = (15, 15))
ax = sns.heatmap(df_trip.corr().round(2),
            annot=True,cmap='Greens',square=True)
```



In [ ]:

In [ ]:

In [ ]:

In [ ]:

2. Feature Creation (10 Points)

In [84]:

```
df_trip.head()
```

Out[84]:

	trip_uuid	data	trip_creation_time	route_schedule_uuid	route_type	source_
0	trip-153671041653548748	training	2018-09-12 00:00:16.535741	thanos::sroute:d7c989ba-a29b-4a0b-b2f4-288cdc6...	FTL	IND2093
1	trip-153671042288605164	training	2018-09-12 00:00:22.886430	thanos::sroute:3a1b0ab2-bb0b-4c53-8c59-eb2a2c0...	Carting	IND5612
2	trip-153671043369099517	training	2018-09-12 00:00:33.691250	thanos::sroute:de5e208e-7641-45e6-8100-4d9fb1e...	FTL	IND000C
3	trip-153671046011330457	training	2018-09-12 00:01:00.113710	thanos::sroute:f0176492-a679-4597-8332-bbd1c7f...	Carting	IND400C
4	trip-153671052974046625	training	2018-09-12 00:02:09.740725	thanos::sroute:d9f07b12-65e0-4f3b-bec8-df06134...	FTL	IND5831

In [255]:

```
df_trip['trip_creation_hour'] = df_trip['trip_creation_time'].dt.hour
```

In [256]:

```
def split_name(name):
    if not pd.isna(name):
        city = name.split(" (",1)
        state = city[1].strip()[:-1]
        cit = city[0].strip().replace("_"," ")
        city_list = cit.split(" ",2)
        if len(city_list)==2:
            city_list.append('')
        elif len(city_list)==1:
            city_list.extend(['',''])
        city_list.append(state)
        return city_list
    else:
        return [None,None,None,None]
```

In [257]:

```
split_name('Kanpur_Central_H_6 (Uttar Pradesh)')
```

Out[257]:

```
['Kanpur', 'Central', 'H 6', 'Uttar Pradesh']
```

In [258]:

```
[(i,l) for i,l in zip(df_trip['destination_name'],df_trip['destination_name'].apply
```

Out[258]:

```
[]
```

In [259]:

```
l_l = list(df_trip['source_name'].apply(split_name))
```

In [260]:

```
df_trip['source_city'] = [l[0] for l in l_l]
df_trip['source_place'] = [l[1] for l in l_l]
df_trip['source_code'] = [l[2] for l in l_l]
df_trip['source_state'] = [l[3] for l in l_l]
```

In [261]:

```
l_l = list(df_trip['destination_name'].apply(split_name))
```

In [262]:

```
df_trip['destination_city'] = [l[0] for l in l_l]
df_trip['destination_place'] = [l[1] for l in l_l]
df_trip['destination_code'] = [l[2] for l in l_l]
df_trip['destination_state'] = [l[3] for l in l_l]
```

In [263]:

```
df_trip.head()
```

Out[263]:

	trip_uuid	data	trip_creation_time	route_schedule_uuid	route_type	source_
0	trip-153671041653548748	training	2018-09-12 00:00:16.535741	thanos::sroute:d7c989ba-a29b-4a0b-b2f4-288cdc6...	FTL	IND2093
1	trip-153671042288605164	training	2018-09-12 00:00:22.886430	thanos::sroute:3a1b0ab2-bb0b-4c53-8c59-eb2a2c0...	Carting	IND5612
2	trip-153671043369099517	training	2018-09-12 00:00:33.691250	thanos::sroute:de5e208e-7641-45e6-8100-4d9fb1e...	FTL	IND000C
3	trip-153671046011330457	training	2018-09-12 00:01:00.113710	thanos::sroute:f0176492-a679-4597-8332-bbd1c7f...	Carting	IND400C
4	trip-153671052974046625	training	2018-09-12 00:02:09.740725	thanos::sroute:d9f07b12-65e0-4f3b-bec8-df06134...	FTL	IND5831

In [ ]:

4. Comparison & Visualization of time and distance fields (10 Points)

In [219]:

```
df_trip.head()
```

Out[219]:

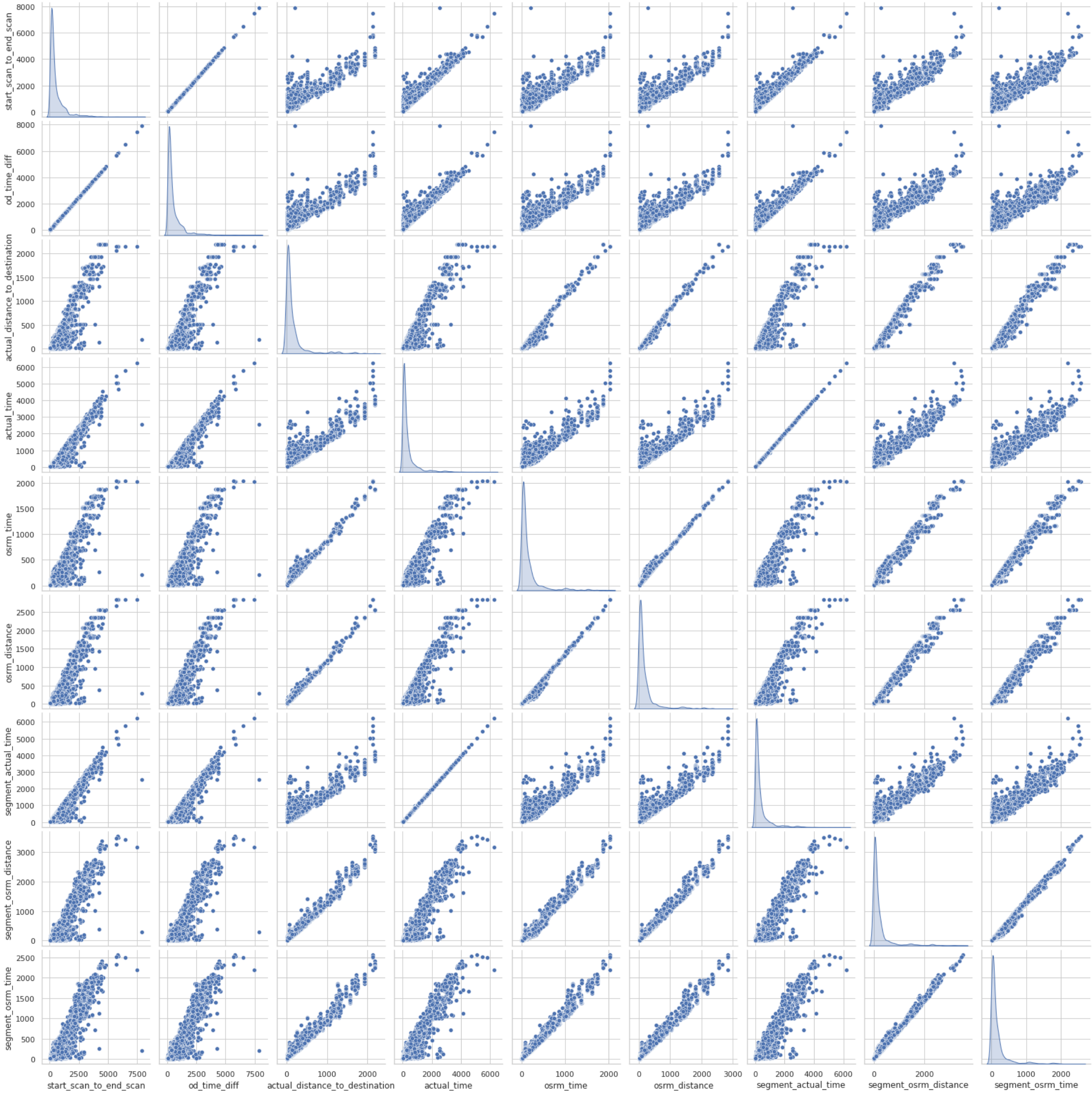
_distance	segment_actual_time	segment_osrm_distance	segment_osrm_time	trip_creation_hour
991.3523	1548.0	1320.4733	1008.0	0
85.1110	141.0	84.1894	65.0	0
ɩ354.0665	3308.0	2545.2678	1941.0	0
19.6800	59.0	19.8766	16.0	0
146.7918	340.0	146.7919	115.0	0

In [229]:

```
sns.pairplot(data=df_trip[time_distance],diag_kind='kde')
```

Out[229]:

```
<seaborn.axisgrid.PairGrid at 0x7f19a85519d0>
```

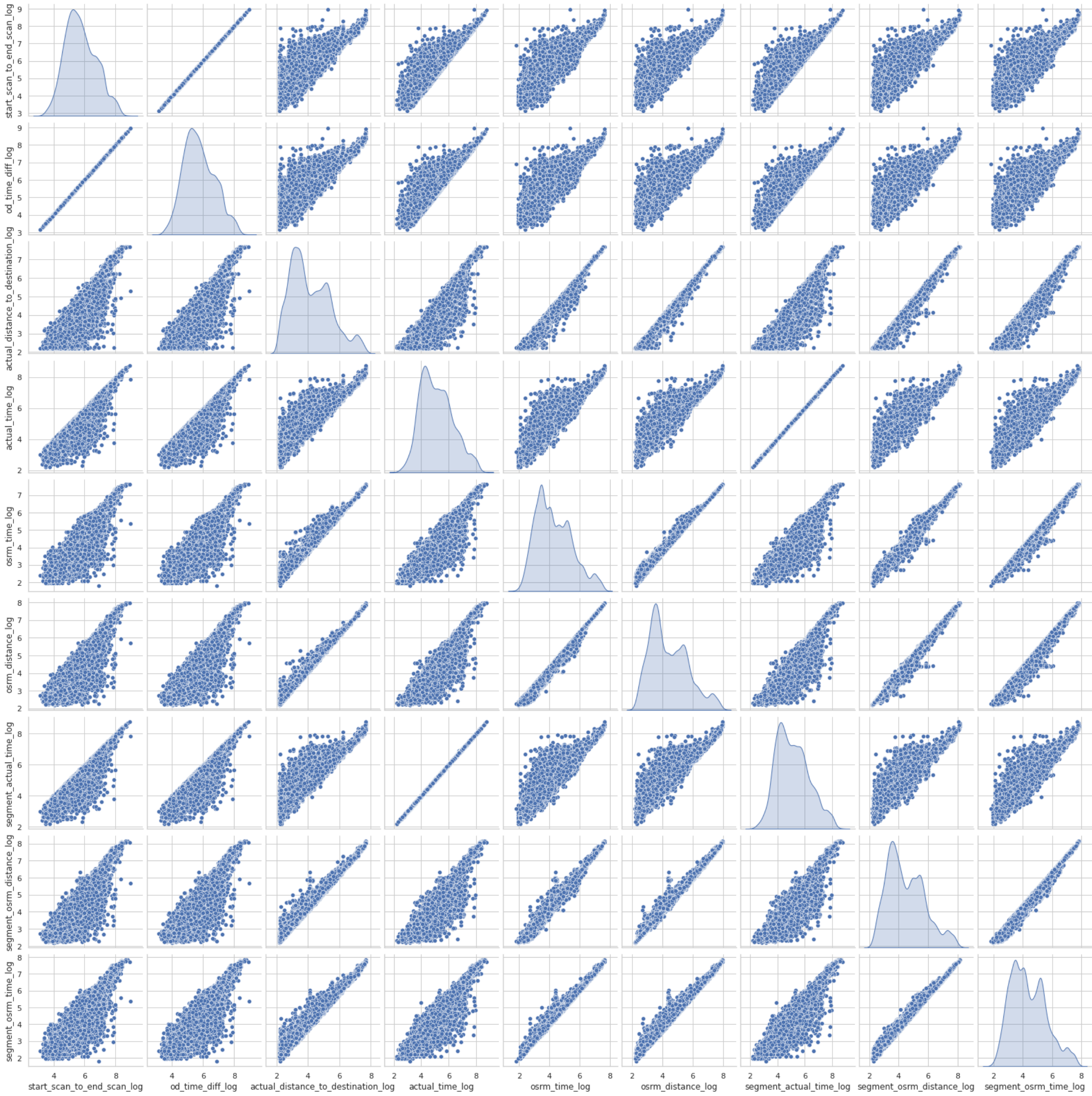


In [230]:

```
sns.pairplot(data=df_trip[[t+'_log' for t in time_distance]],diag_kind='kde')
```

Out[230]:

```
<seaborn.axisgrid.PairGrid at 0x7f199bbeebe0>
```



In [ ]:

```

```

6. Checking relationship between aggregated fields (10 Points)

In [ ]:

H0: Data is Gaussian

Ha: Data is not Gaussian

test : kstest

p-value : 0.05

In [280]:

```
for key in time_distance:
    kstest_test(df_trip[key+'_log'].sample(300))
    print()
```

```
Data is not Gaussian

Mean :  5.13 , Standard deviation :  1.18
p-value :  0.04933229086420676
Reject H0
Data is not Gaussian

Mean :  4.21 , Standard deviation :  1.11
p-value :  0.010791600145108604
Reject H0
Data is not Gaussian

Mean :  4.58 , Standard deviation :  1.3
p-value :  0.0022994138636897553
Reject H0
Data is not Gaussian

Mean :  5.11 , Standard deviation :  1.24
p-value :  0.0035612912421839143
Reject H0
Data is not Gaussian
```

In [281]:

```
time_distance
```

Out[281]:

```
['start_scan_to_end_scan',
 'od_time_diff',
 'actual_distance_to_destination',
 'actual_time',
 'osrm_time',
 'osrm_distance',
 'segment_actual_time',
 'segment_osrm_distance',
 'segment_osrm_time']
```

In [ ]:

In [ ]:

```

```

```
H0: mean actual time taken to deliver is equal to mean osrm time taken \
Ha: mean actual time taken to deliver is not-equal to mean osrm time taken

hypothesis test : T-test for independent samples \
p-value : 0.05 (two-tailed)
```

In [377]:

```
smp1 = df_trip['actual_time_log'].sample(3000)
smp2 = df_trip['osrm_time_log'].sample(3000)
smp1.mean(),smp2.mean()
```

Out[377]:

```
(5.113023069166416, 4.318621085245073)
```

In [378]:

```
ttest_ind_test(smp1,smp2 ,alternative='less')
```

```
Series1 metrics :
Mean :  5.11 , Standard deviation :  1.18
Series2 metrics :
Mean :  4.32 , Standard deviation :  1.19
p-value :  1.0
Fail to reject H0
```

In [ ]:

```

```

H0: mean actual time taken to deliver is equal to mean segment actual time taken
Ha: mean actual time taken to deliver is not-equal to mean segment actual time taken

hypothesis test : T-test for independent samples
p-value : 0.05 (two-tailed)

In [430]:

```
smp1 = df_trip['actual_time_log'].sample(3000)
smp2 = df_trip['segment_actual_time_log'].sample(3000)
smp1.mean(),smp2.mean()
```

Out[430]:

```
(5.149034402451168, 5.108592426731569)
```

In [431]:

```
ttest_ind_test(smp1,smp2 ,alternative='greater')
```

```
Series1 metrics :
Mean :  5.15 , Standard deviation :  1.16
Series2 metrics :
Mean :  5.11 , Standard deviation :  1.17
p-value :  0.08964437416840079
Reject H0
```

In [ ]:

```

```

H0: mean actual time taken to deliver is equal to mean osrm time taken

Ha: mean actual time taken to deliver is not-equal to mean osrm time taken

hypothesis test : T-test for independent samples

p-value : 0.05 (two-tailed)

In [435]:

```
smp1 = df_trip['actual_time_log'].sample(2000)
smp2 = df_trip['osrm_time_log'].sample(2000)
smp1.mean(),smp2.mean()
```

Out[435]:

```
(5.093486622670581, 4.287385001601136)
```

In [436]:

```
ttest_ind_test(smp1,smp2 ,alternative='less')
```

```
Series1 metrics :
Mean :  5.09 , Standard deviation :  1.18
Series2 metrics :
Mean :  4.29 , Standard deviation :  1.2
p-value :  1.0
Fail to reject H0
```

In [ ]:

```

```

H0: mean segment osrm distance taken to deliver is equal to mean osrm distance taken

Ha: mean segment osrm distance taken to deliver is not-equal to mean osrm distance taken

hypothesis test : T-test for independent samples

p-value : 0.05 (two-tailed)

In [455]:

```
smp1 = df_trip['segment_osrm_distance_log'].sample(3000)
smp2 = df_trip['osrm_distance_log'].sample(3000)
smp1.mean(),smp2.mean()
```

Out[455]:

```
(4.428466685593908, 4.4108154524547665)
```

In [456]:

```
ttest_ind_test(smp1,smp2 ,alternative='less')
```

```
Series1 metrics :
Mean :  4.43 , Standard deviation :  1.27
Series2 metrics :
Mean :  4.41 , Standard deviation :  1.25
p-value :  0.7063181754113308
Fail to reject H0
```

In [ ]:

H0: mean actual distance taken to deliver is equal to mean osrm distance taken

Ha: mean actual distance taken to deliver is not-equal to mean osrm distance taken

hypothesis test : T-test for independent samples

p-value : 0.05 (two-tailed)

In [328]:

```
smp1 = df_trip['actual_distance_to_destination_log'].sample(1000)
smp2 = df_trip['osrm_distance_log'].sample(1000)
smp1.mean(),smp2.mean()
```

Out[328]:

```
(4.131260892892641, 4.438064129071632)
```

In [329]:

```
ttest_ind_test(smp1,smp2 ,alternative='greater')
```

```
Series1 metrics :
Mean :  4.13 , Standard deviation :  1.27
Series2 metrics :
Mean :  4.44 , Standard deviation :  1.24
p-value :  0.9999999750000707
Fail to reject H0
```

In [ ]:

7. Handling categorical values (10 Points)

In [468]:

```
df_trips = pd.get_dummies(df_trip, columns = ['route_type','source_state','source_c
```

In [465]:

```
df_trip['source_state'].value_counts()[:20].index
```

Out[465]:

```
Index(['Maharashtra', 'Karnataka', 'Haryana', 'Tamil Nadu', 'Telangan
a',
       'Uttar Pradesh', 'Gujarat', 'Delhi', 'West Bengal', 'Punjab',
       'Rajasthan', 'Andhra Pradesh', 'Bihar', 'Madhya Pradesh', 'Kera
la',
       'Assam', 'Jharkhand', 'Uttarakhand', 'Orissa', 'Chandigarh'],
      dtype='object')
```

In [466]:

```
df_trip['destination_state'].value_counts()[:20].index
```

Out[466]:

```
Index(['Maharashtra', 'Karnataka', 'Haryana', 'Tamil Nadu', 'Uttar Pra
desh',
       'Telangana', 'Gujarat', 'West Bengal', 'Delhi', 'Punjab', 'Raja
sthan',
       'Andhra Pradesh', 'Bihar', 'Madhya Pradesh', 'Kerala', 'Assam',
       'Jharkhand', 'Uttarakhand', 'Orissa', 'Chandigarh'],
      dtype='object')
```

In [ ]:

8. Column Normalization /Column Standardization (10 Points)

In [470]:

```
scaler = MinMaxScaler()
scaler.fit(df_trip[time_distance + time_distance_log])
```

Out[470]:

```
MinMaxScaler()
```

In [472]:

```
mx_data = scaler.transform(df_trip[time_distance + time_distance_log])
```

In [474]:

```
mx_df = pd.DataFrame(data=mx_data,columns=time_distance + time_distance_log)
```

In [480]:

```
mx_df.columns = [col+'_mx' for col in mx_df.columns]
```

In [ ]:

```

```

In [482]:

```
scaler = StandardScaler()
scaler.fit(df_trip[time_distance + time_distance_log])
```

Out[482]:

```
StandardScaler()
```

In [483]:

```
ss_data = scaler.transform(df_trip[time_distance + time_distance_log])
```

In [484]:

```
ss_df = pd.DataFrame(data=ss_data,columns=time_distance + time_distance_log)
```

In [485]:

```
ss_df.columns = [col+'_ss' for col in ss_df.columns]
```

In [489]:

```
df_trips.reset_index(drop=True,inplace=True)
```

In [492]:

```
df_ = pd.concat([df_trips,mx_df,ss_df],axis=1)
```

In [493]:

```
df_.head()
```

Out[493]:

	trip_uuid	data	trip_creation_time	route_schedule_uuid	source_center	
0	trip-153671041653548748	training	2018-09-12 00:00:16.535741	thanos::sroute:d7c989ba-a29b-4a0b-b2f4-288cdc6...	IND209304AAA	
1	trip-153671042288605164	training	2018-09-12 00:00:22.886430	thanos::sroute:3a1b0ab2-bb0b-4c53-8c59-eb2a2c0...	IND561203AAB	Dod
2	trip-153671043369099517	training	2018-09-12 00:00:33.691250	thanos::sroute:de5e208e-7641-45e6-8100-4d9fb1e...	IND000000ACB	(
3	trip-153671046011330457	training	2018-09-12 00:01:00.113710	thanos::sroute:f0176492-a679-4597-8332-bbd1c7f...	IND400072AAB	
4	trip-153671052974046625	training	2018-09-12 00:02:09.740725	thanos::sroute:d9f07b12-65e0-4f3b-bec8-df06134...	IND583101AAA	E

5 rows × 1685 columns

In [ ]:

9. Business Insights (10 Points) - Should include patterns observed in the data along with what you can infer from it.

60% of deliveries are through carting and 40% of deliveries FTL
Bangalore , gurgram, mumbai cities contribute to nearly 40 % of the deliveries
Karnataka, Maharastra, Haryana states contribute to nearly 40% of the deliveries
FTL deliveries takes more time than carting deliveries

In [ ]:

10. Recommendations (10 Points) - Actionable items for business. No technical jargon. No complications. Simple action items that everyone can understand.

We need to focus on increasing carting deliveries because the deliver faster and less distance, so we need more hubs

we need to divercify across the cities and states, because most of our deliveries come from only 3-4 states
actual and osrm time is significant, so we need to optimize the performance of the engine.

In [ ]: