Delhivery

April 17, 2023

[452]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from scipy.stats import pearsonr, spearmanr # For correlation testing
```

[348]:
```
df = pd.read_csv("delhivery_data.csv")
```

[349]:
```
df.head(5)
```

[349]:

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_center	source_name	destination_center	destination_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)	IND388620AAB	Khambhat_MotvdDPP_D (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)	IND388620AAB	Khambhat_MotvdDPP_D (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)	IND388620AAB	Khambhat_MotvdDPP_D (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)	IND388620AAB	Khambhat_MotvdDPP_D (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)	IND388620AAB	Khambhat_MotvdDPP_D (Gujarat)

```
                od_start_time  …             cutoff_timestamp  \
0  2018-09-20 03:21:32.418600  …          2018-09-20 04:27:55
1  2018-09-20 03:21:32.418600  …          2018-09-20 04:17:55
2  2018-09-20 03:21:32.418600  …  2018-09-20 04:01:19.505586
3  2018-09-20 03:21:32.418600  …          2018-09-20 03:39:57
4  2018-09-20 03:21:32.418600  …          2018-09-20 03:33:55

   actual_distance_to_destination  actual_time  osrm_time osrm_distance  \
0                       10.435660         14.0       11.0       11.9653
1                       18.936842         24.0       20.0       21.7243
2                       27.637279         40.0       28.0       32.5395
3                       36.118028         62.0       40.0       45.5620
4                       39.386040         68.0       44.0       54.2181

     factor  segment_actual_time  segment_osrm_time  segment_osrm_distance  \
0  1.272727                 14.0               11.0                11.9653
1  1.200000                 10.0                9.0                 9.7590
2  1.428571                 16.0                7.0                10.8152
3  1.550000                 21.0               12.0                13.0224
4  1.545455                  6.0                5.0                 3.9153

   segment_factor
0        1.272727
1        1.111111
2        2.285714
3        1.750000
4        1.200000

[5 rows x 24 columns]
```

1 1. Basic data cleaning and exploration:

1.0.1 1. Analyze the structure of the data.

[350]:
```
df.shape
```

[350]:
```
(144867, 24)
```

[351]:
```
df.describe()
```

[351]:
```
       start_scan_to_end_scan  cutoff_factor  actual_distance_to_destination  \
count           144867.000000  144867.000000                   144867.000000
mean               961.262986     232.926567                      234.073372
std               1037.012769     344.755577                      344.990009
min                 20.000000       9.000000                        9.000045
25%                161.000000      22.000000                       23.355874
```

```
50%                449.000000      66.000000                      66.126571
75%               1634.000000     286.000000                     286.708875
max               7898.000000    1927.000000                    1927.447705

         actual_time      osrm_time  osrm_distance         factor  \
count  144867.000000  144867.000000  144867.000000  144867.000000
mean      416.927527     213.868272     284.771297       2.120107
std       598.103621     308.011085     421.119294       1.715421
min         9.000000       6.000000       9.008200       0.144000
25%        51.000000      27.000000      29.914700       1.604264
50%       132.000000      64.000000      78.525800       1.857143
75%       513.000000     257.000000     343.193250       2.213483
max      4532.000000    1686.000000    2326.199100      77.387097

       segment_actual_time  segment_osrm_time  segment_osrm_distance  \
count        144867.000000      144867.000000           144867.00000
mean             36.196111          18.507548               22.82902
std              53.571158          14.775960               17.86066
min            -244.000000           0.000000                0.00000
25%              20.000000          11.000000               12.07010
50%              29.000000          17.000000               23.51300
75%              40.000000          22.000000               27.81325
max            3051.000000        1611.000000             2191.40370

       segment_factor
count   144867.000000
mean         2.218368
std          4.847530
min        -23.444444
25%          1.347826
50%          1.684211
75%          2.250000
max        574.250000
```

[352]:
```
df.describe(include="object")
```

[352]:
```
            data          trip_creation_time  \
count     144867                      144867
unique         2                       14817
top     training  2018-09-28 05:23:15.359220
freq      104858                         101

                                   route_schedule_uuid route_type  \
count                                           144867     144867
unique                                            1504          2
top     thanos::sroute:4029a8a2-6c74-4b7e-a6d8-f9e069f…        FTL
freq                                              1812      99660
```

```
                      trip_uuid source_center                    source_name  \
count                    144867        144867                         144574
unique                    14817          1508                           1498
top     trip-153811219535896559  IND000000ACB  Gurgaon_Bilaspur_HB (Haryana)
freq                        101         23347                          23347

       destination_center              destination_name  \
count              144867                        144606
unique               1481                          1468
top           IND000000ACB  Gurgaon_Bilaspur_HB (Haryana)
freq                15192                         15192

                     od_start_time                 od_end_time  \
count                       144867                      144867
unique                       26369                       26369
top     2018-09-21 18:37:09.322207  2018-09-24 09:59:15.691618
freq                            81                          81

           cutoff_timestamp
count                144867
unique                93180
top     2018-09-24 05:19:20
freq                    40
```

[353]:
```
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
```

1.0.2 2. Handle missing values in the data.

[354]:
```
df.isnull().sum()
# Missing values are in source_name and destination_name
# we can check
```

[354]:
```
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
```

[355]:
```
## Condtion we created for this kind of missing values ##
# ******************************************************
```

```
# Splitted the data by missing values and count the number of missing values␣
 ↪for each particular source_center
# Then for each source_center, check if there any matching values in␣
 ↪source_center and source_center column of main data
# Calculate the number of missing count of each center and compare with whole␣
 ↪data count if both are same then that particular data won't be anywhere in␣
 ↪our dataset
# If the above condition is true we can drop the null values
```

Missing values check for source_name feature

[356]:
```
miss_count = df[df['source_name'].apply(pd.isna)]["source_center"].
 ↪value_counts().reset_index()
cant_modify = []
can_modify = []
for i in miss_count["index"]:
    if (df[df["source_center"] == f"{i}"]["source_name"].apply(pd.isna).sum()␣
 ↪== miss_count[miss_count["index"] == f"{i}"]["source_center"].values[0]) and␣
 ↪(df[df["source_center"] == f"{i}"]["source_name"].apply(pd.isna).sum() ==␣
 ↪df[df["source_center"] == f"{i}"].shape[0]):
        cant_modify.append(i)
    else:
        can_modify.append(i)
if can_modify == []:
    print("We can drop the rows of source_name which are having missing values,␣
 ↪there is no way to assume any value there")
```

```
We can drop the rows of source_name which are having missing values, there is no
way to assume any value there
```

Missing values check for source_name feature

[357]:
```
miss_count = df[df['source_name'].apply(pd.isna)]["source_center"].
 ↪value_counts().reset_index()
cant_modify = []
can_modify = []
for i in miss_count["index"]:
    if df[df["source_center"] == f"{i}"]["source_name"].apply(pd.isna).sum() ==␣
 ↪miss_count[miss_count["index"] == f"{i}"]["source_center"].values[0] and␣
 ↪(df[df["source_center"] == f"{i}"]["source_name"].apply(pd.isna).sum() ==␣
 ↪df[df["source_center"] == f"{i}"].shape[0]):
        cant_modify.append(i)
    else:
        can_modify.append(i)
if can_modify == []:
    print("We can drop the rows of source_name which are having missing values,␣
 ↪there is no way to assume any value there")
```

```
We can drop the rows of source_name which are having missing values, there is no
```

way to assume any value there

[358]:
```
# Dropping the missing values
df = df.dropna(axis=0)
```

[359]:
```
df.isnull().sum()
```

[359]:
```
data                                0
trip_creation_time                  0
route_schedule_uuid                 0
route_type                          0
trip_uuid                           0
source_center                       0
source_name                         0
destination_center                  0
destination_name                    0
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
```

[360]:
```
# All missing values are dropped, there is no way to fill up with other values␣
 ↪or with any aggregation values
# Hence we dropped missing values
```

1.0.3 3. Merging the rows

[361]:
```
# Merging the rows with groupby of trip id, source center, destination center␣
 ↪and aggregate sum by segment time and max by actual cumulative time
# So that the we can able to fetch data of source and destination with their␣
 ↪actual time taken and total segment time taken
```

[462]:
```
groupby_trip_source_dest = df.
 ↪groupby(["trip_uuid","source_name","destination_name"]).agg(
    {
```

```
        "segment_actual_time":"sum",
        "segment_osrm_time":"sum",
        "segment_osrm_distance":"sum",
        "actual_time":"max",
        "osrm_time":"max",
        "osrm_distance":"max"
    }).reset_index()
```

[463]:
```
merged_data = groupby_trip_source_dest.groupby("trip_uuid").agg(
    {
        "source_name":"first",
        "destination_name":"last",
        "segment_actual_time":"sum",
        "segment_osrm_time":"sum",
        "segment_osrm_distance":"sum",
        "actual_time":"sum",
        "osrm_time":"sum",
        "osrm_distance":"sum"
    }).reset_index()
```

[463]:

	trip_uuid	source_name	destination_name	segment_actual_time	segment_osrm_time	segment_osrm_distance	actual_time	osrm_time
0	trip-153671041653548748	Bhopal_Trnsport_H (Madhya Pradesh)	Kanpur_Central_H_6 (Uttar Pradesh)	820.0				
1	trip-153671041653548748	Kanpur_Central_H_6 (Uttar Pradesh)	Gurgaon_Bilaspur_HB (Haryana)	728.0				
2	trip-153671042288605164	Doddablpur_ChikaDPP_D (Karnataka)	Chikblapur_ShntiSgr_D (Karnataka)	46.0				
3	trip-153671042288605164	Tumkur_Veersagr_I (Karnataka)	Doddablpur_ChikaDPP_D (Karnataka)	95.0				
4	trip-153671043369099517	Bangalore_Nelmngla_H (Karnataka)	Gurgaon_Bilaspur_HB (Haryana)	2700.0				
…	…	…	…	…				
26217	trip-153861115439069069	Thisayanvilai_UdnkdiRD_D (Tamil Nadu)	Peikulam_SriVnktpm_D (Tamil Nadu)	32.0				
26218	trip-153861115439069069	Tirchchndr_Shnmgprm_D (Tamil Nadu)	Thisayanvilai_UdnkdiRD_D (Tamil Nadu)	49.0				
26219	trip-153861115439069069	Tirunelveli_VdkkuSrt_I (Tamil Nadu)	Eral_Busstand_D (Tamil Nadu)	59.0				
26220	trip-153861118270144424	Hospet (Karnataka)	Sandur_WrdN1DPP_D (Karnataka)	41.0				
26221	trip-153861118270144424	Sandur_WrdN1DPP_D (Karnataka)	Bellary_Dc (Karnataka)	233.0				

```
0                 474.0               649.8528        830.0      394.0
1                 534.0               670.6205        732.0      349.0
2                  26.0                28.1995         47.0       26.0
3                  39.0                55.9899         96.0       42.0
4                1710.0              2227.5270       2736.0     1529.0
…                   …                    …          …          …
26217              30.0                25.7087         33.0       31.0
26218              42.0                42.1431         51.0       41.0
26219              58.0                61.0753         60.0       50.0
26220              25.0                28.0484         42.0       26.0
26221              42.0                52.5303        233.0       42.0

       osrm_distance
0           544.8027
1           446.5496
2            28.1994
3            56.9116
4          2090.8743
…               …
26217        25.7087
26218        42.5213
26219        52.8070
26220        28.0484
26221        52.5303

[26222 rows x 9 columns]
```

[464]:

[465]:
```
merged_data
```

[465]:
```
                     trip_uuid                          source_name  \
0      trip-153671041653548748  Bhopal_Trnsport_H (Madhya Pradesh)
1      trip-153671042288605164   Doddablpur_ChikaDPP_D (Karnataka)
2      trip-153671043369099517    Bangalore_Nelmngla_H (Karnataka)
3      trip-153671046011330457            Mumbai Hub (Maharashtra)
4      trip-153671052974046625              Bellary_Dc (Karnataka)
…                          …                                  …
14782  trip-153861095625827784      Chandigarh_Mehmdpur_H (Punjab)
14783  trip-153861104386292051        FBD_Balabhgarh_DPC (Haryana)
14784  trip-153861106442901555  Kanpur_Central_H_6 (Uttar Pradesh)
14785  trip-153861115439069069        Eral_Busstand_D (Tamil Nadu)
14786  trip-153861118270144424                   Hospet (Karnataka)

                        destination_name  segment_actual_time  \
0           Gurgaon_Bilaspur_HB (Haryana)               1548.0
1       Doddablpur_ChikaDPP_D (Karnataka)                141.0
```

```
2          Chandigarh_Mehmdpur_H (Punjab)               3308.0
3          Mumbai_MiraRd_IP (Maharashtra)                 59.0
4                  Bellary_Dc (Karnataka)                340.0
…                                    …                    …
14782      Chandigarh_Mehmdpur_H (Punjab)                 82.0
14783      Faridabad_Blbgarh_DC (Haryana)                 21.0
14784  Kanpur_Central_H_6 (Uttar Pradesh)                281.0
14785        Eral_Busstand_D (Tamil Nadu)                 258.0
14786                Bellary_Dc (Karnataka)               274.0

       segment_osrm_time  segment_osrm_distance  actual_time  osrm_time  \
0                 1008.0              1320.4733       1562.0      743.0
1                   65.0                84.1894        143.0       68.0
2                 1941.0              2545.2678       3347.0     1741.0
3                   16.0                19.8766         59.0       15.0
4                  115.0               146.7919        341.0      117.0
…                    …                     …          …          …
14782               62.0                64.8551         83.0       62.0
14783               11.0                16.0883         21.0       12.0
14784               88.0               104.8866        282.0       54.0
14785              221.0               223.5324        264.0      184.0
14786               67.0                80.5787        275.0       68.0

       osrm_distance
0           991.3523
1            85.1110
2          2372.0852
3            19.6800
4           146.7918
…              …
14782        73.4630
14783        16.0882
14784        63.2841
14785       177.6635
14786        80.5787

[14787 rows x 9 columns]
```

[ ]:

2 2. Build some features to prepare the data for actual analysis. Extract features from the below fields:

2.0.1 1. Destination Name: Split and extract features out of destination. City-place-code (State)

Seperator function to split

[388]:
```
def seperator(x):
    res = x.split("_")
    if len(res) == 2:
        second_split = res[1].split(" ")
        res.pop()
        for i in second_split:
            res.append(i)
    elif len(res) == 1:
        third_split = res[0].split(" ")
        res.pop()
        for i in third_split:
            res.append(i)
        if len(res) <= 2:
            res.append(third_split[-1])
    return res if len(res) == 3 else res[:3]
```

[366]:
```
df["destination_name"]
```

[366]:
```
0         Khambhat_MotvdDPP_D (Gujarat)
1         Khambhat_MotvdDPP_D (Gujarat)
2         Khambhat_MotvdDPP_D (Gujarat)
3         Khambhat_MotvdDPP_D (Gujarat)
4         Khambhat_MotvdDPP_D (Gujarat)
                      …
144862    Gurgaon_Bilaspur_HB (Haryana)
144863    Gurgaon_Bilaspur_HB (Haryana)
144864    Gurgaon_Bilaspur_HB (Haryana)
144865    Gurgaon_Bilaspur_HB (Haryana)
144866    Gurgaon_Bilaspur_HB (Haryana)
Name: destination_name, Length: 144316, dtype: object
```

[367]:
```
# Split the destination name column with "_" where we observerd this is the␣
 ↪delimiter for destination name.
# Dropped the unwanted columns
# Appending the data to Main Dataframe
destination = pd.DataFrame(df["destination_name"].apply(seperator).tolist(),␣
 ↪index=df.trip_uuid).reset_index()
destination.columns = ["id","City","Place","State"]
df["destination_city"] = destination["City"].to_numpy()
df["destination_place"] = destination["Place"].to_numpy()
df["destination_state"] = destination["State"].to_numpy()
```

2.0.2 2. Source Name: Split and extract features out of destination. City-place-code (State)

[368]:
```
df["source_name"]
```

[368]:
```
0          Anand_VUNagar_DC (Gujarat)
1          Anand_VUNagar_DC (Gujarat)
2          Anand_VUNagar_DC (Gujarat)
3          Anand_VUNagar_DC (Gujarat)
4          Anand_VUNagar_DC (Gujarat)
                      ...
144862     Sonipat_Kundli_H (Haryana)
144863     Sonipat_Kundli_H (Haryana)
144864     Sonipat_Kundli_H (Haryana)
144865     Sonipat_Kundli_H (Haryana)
144866     Sonipat_Kundli_H (Haryana)
Name: source_name, Length: 144316, dtype: object
```

[389]:
```
# Split the source name column with "_" where we observerd this is the
 ↪delimiter for source name.
# Dropped the unwanted columns
# Appending the data to Main Dataframe
source = pd.DataFrame(df["source_name"].apply(seperator).tolist(), index=df.
 ↪trip_uuid).reset_index()
source.columns = ["id","City","Place","Code"]
df["source_city"] = source["City"].to_numpy()
df["source_place"] = source["Place"].to_numpy()
df["source_code"] = source["Code"].to_numpy()
```

[390]:
```
df.isna().sum()
```

[390]:
```
data                            0
trip_creation_time              0
route_schedule_uuid             0
route_type                      0
trip_uuid                       0
source_center                   0
source_name                     0
destination_center              0
destination_name                0
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
destination_city             0
destination_place            0
destination_state            0
source_city                  0
source_place                 0
source_code                  0
dtype: int64
```

2.0.3 3. Trip_creation_time: Extract features like month, year and day etc

[393]:
```
# First will convert the whole column into datetime dtype
# Then will split this into multiple features
df["trip_creation_time"] = pd.to_datetime(df["trip_creation_time"])
df["Trip_Year"] = df["trip_creation_time"].dt.year
df["Trip_Month"] = df["trip_creation_time"].dt.month_name()
df["Trip_day"] = df["trip_creation_time"].dt.day
```

[393]:
```
0         2018-09-20 02:35:36.476840
1         2018-09-20 02:35:36.476840
2         2018-09-20 02:35:36.476840
3         2018-09-20 02:35:36.476840
4         2018-09-20 02:35:36.476840
                     …
144862    2018-09-20 16:24:28.436231
144863    2018-09-20 16:24:28.436231
144864    2018-09-20 16:24:28.436231
144865    2018-09-20 16:24:28.436231
144866    2018-09-20 16:24:28.436231
Name: trip_creation_time, Length: 144316, dtype: datetime64[ns]
```

3 3. In-depth analysis and feature engineering:

1. Calculate the time taken between od_start_time and od_end_time and keep it as a feature. Drop the original columns, if required

[446]:
```
# od_start_time - Trip start time
# od_end_time - Trip end time
# For calculating difference between od_start and od_end we can find out the␣
 ↪original time taken by the order.
df["od_start_time"] = pd.to_datetime(df["od_start_time"])
df["od_end_time"] = pd.to_datetime(df["od_end_time"])
df["time_diff_min"] = (df["od_end_time"] - df["od_start_time"]).dt.
 ↪total_seconds()/60
```

2. Compare the difference between Point a. and start_scan_to_end_scan. Do hypothesis testing/ Visual analysis to check.

[ ]:
```
# start_scan_to_end_scan - Time taken to deliver from source to destination
# time_diff_min - Calculated time diff by their actual timings
```

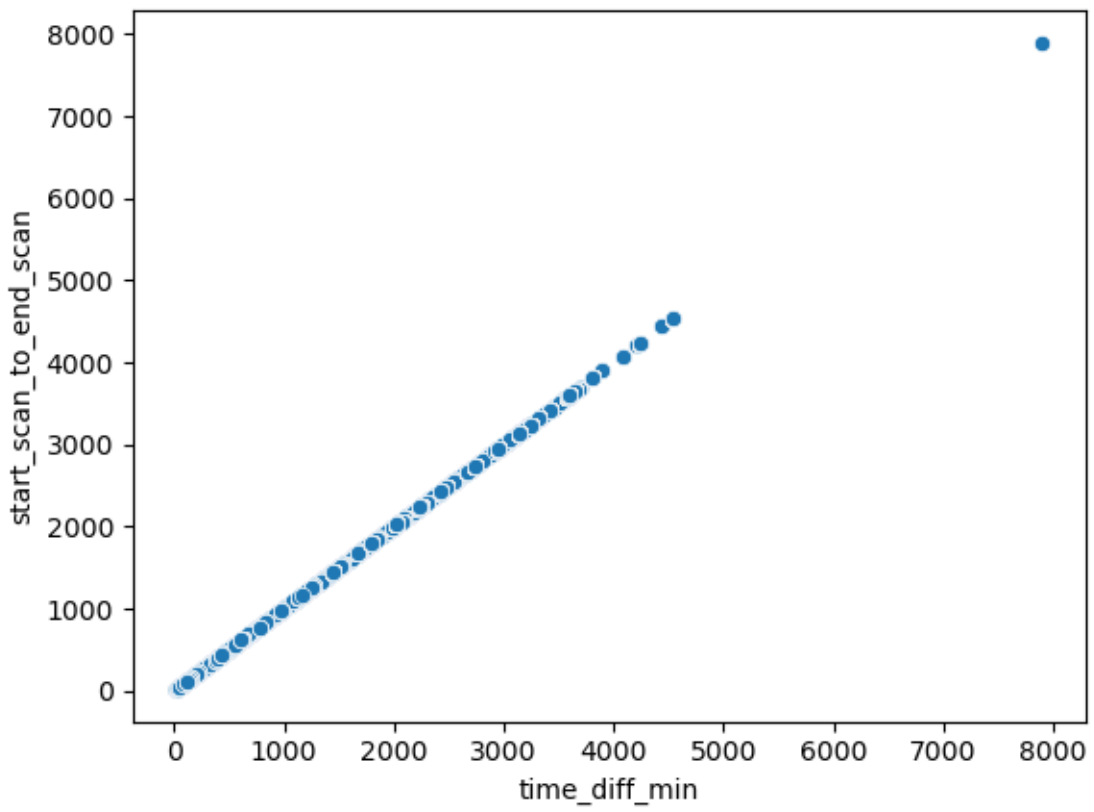
Hypothetical testing for calculated timing and actual timing

[453]:
```
# H0: Both are not correlated
# Ha: Both are correlated

## properties of two variables ##
# 1. Both features are continuous variables
# 2. Data is right skewed in nature

## Correlation Testing ##
alpha = 0.05
corr_stat, p_value = pearsonr(df["start_scan_to_end_scan"], df["time_diff_min"])
if p_value<alpha:
    print("Reject Null Hypothesis")
else:
    print("Fail to reject Null Hypothesis")
print("Test Statistic Value: ",corr_stat)
print("P_value:",p_value)
```

```
Reject Null Hypothesis
Test Statistic Value:  0.9999999609905782
P_value: 0.0
```

[454]:
```
sns.scatterplot(data=df,x="time_diff_min", y="start_scan_to_end_scan")
```

[454]:
```
<AxesSubplot:xlabel='time_diff_min', ylabel='start_scan_to_end_scan'>
```



Inference

[ ]:
```
# Both features are highly correlated
# Even test confirm the same and graph also tells the same
# Our calculated timings and actual timings both are same there is high␣
 ↪colinearity
# we have created a another feature with 95 % confident
```

3. Do hypothesis testing/ visual analysis between actual_time aggregated value and OSRM time aggregated value (aggregated values are the values you'll get after merging the rows on the basis of trip_uuid)

[466]:
```
merged_data
```

[466]:

	trip_uuid	source_name	\
0	trip-153671041653548748	Bhopal_Trnsport_H (Madhya Pradesh)	
1	trip-153671042288605164	Doddablpur_ChikaDPP_D (Karnataka)	
2	trip-153671043369099517	Bangalore_Nelmngla_H (Karnataka)	
3	trip-153671046011330457	Mumbai Hub (Maharashtra)	
4	trip-153671052974046625	Bellary_Dc (Karnataka)	
…	…	…	
14782	trip-153861095625827784	Chandigarh_Mehmdpur_H (Punjab)	

```
14783  trip-153861104386292051        FBD_Balabhgarh_DPC (Haryana)
14784  trip-153861106442901555  Kanpur_Central_H_6 (Uttar Pradesh)
14785  trip-153861115439069069        Eral_Busstand_D (Tamil Nadu)
14786  trip-153861118270144424                  Hospet (Karnataka)

                        destination_name  segment_actual_time  \
0           Gurgaon_Bilaspur_HB (Haryana)               1548.0
1        Doddablpur_ChikaDPP_D (Karnataka)                141.0
2           Chandigarh_Mehmdpur_H (Punjab)               3308.0
3           Mumbai_MiraRd_IP (Maharashtra)                 59.0
4                  Bellary_Dc (Karnataka)                340.0
…                                     …                   …
14782       Chandigarh_Mehmdpur_H (Punjab)                 82.0
14783       Faridabad_Blbgarh_DC (Haryana)                 21.0
14784  Kanpur_Central_H_6 (Uttar Pradesh)                281.0
14785        Eral_Busstand_D (Tamil Nadu)                258.0
14786              Bellary_Dc (Karnataka)                274.0

       segment_osrm_time  segment_osrm_distance  actual_time  osrm_time  \
0                 1008.0              1320.4733       1562.0      743.0
1                   65.0                84.1894        143.0       68.0
2                 1941.0              2545.2678       3347.0     1741.0
3                   16.0                19.8766         59.0       15.0
4                  115.0               146.7919        341.0      117.0
…                    …                     …           …          …
14782               62.0                64.8551         83.0       62.0
14783               11.0                16.0883         21.0       12.0
14784               88.0               104.8866        282.0       54.0
14785              221.0               223.5324        264.0      184.0
14786               67.0                80.5787        275.0       68.0

       osrm_distance
0           991.3523
1            85.1110
2          2372.0852
3            19.6800
4           146.7918
…              …
14782        73.4630
14783        16.0882
14784        63.2841
14785       177.6635
14786        80.5787

[14787 rows x 9 columns]
```

Hypothesis Testing for actual_time aggregated value and OSRM time aggregated value Visual Analysis

[ ]:

[ ]:
```
# H0: Both are not correlated
# Ha: Both are correlated

## properties of two variables ##
# 1. Both features are continuous variables
# 2. Data is right skewed in nature

## Correlation Testing ##
alpha = 0.05
corr_stat, p_value = pearsonr(df["start_scan_to_end_scan"], df["time_diff_min"])
if p_value<alpha:
    print("Reject Null Hypothesis")
else:
    print("Fail to reject Null Hypothesis")
print("Test Statistic Value: ",corr_stat)
print("P_value:",p_value)
```

[ ]: