Downloading the data

In [1]:
```
import pandas as pd
import numpy as np

import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
import matplotlib
import warnings
warnings.filterwarnings("ignore")

matplotlib.rcParams['figure.figsize'] = 10,10
```

In [2]:
```
nyc_df = pd.read_csv('nyc_taxi_trip_duration.csv')
nyc_df
```

Out[2]:

	id	vendor_id	pickup_datetime	dropoff_datetime	passenger_count	pickup_longitude
0	id1080784	2	2016-02-29 16:40:21	2016-02-29 16:47:01	1	-73.953918
1	id0889885	1	2016-03-11 23:35:37	2016-03-11 23:53:57	2	-73.988312
2	id0857912	2	2016-02-21 17:59:33	2016-02-21 18:26:48	2	-73.997314
3	id3744273	2	2016-01-05 09:44:31	2016-01-05 10:03:32	6	-73.961670
4	id0232939	1	2016-02-17 06:42:23	2016-02-17 06:56:31	1	-74.017120
...	...	...	...	...	...	...
729317	id3905982	2	2016-05-21 13:29:38	2016-05-21 13:34:34	2	-73.965919
729318	id0102861	1	2016-02-22 00:43:11	2016-02-22 00:48:26	1	-73.996666
729319	id0439699	1	2016-04-15 18:56:48	2016-04-15 19:08:01	1	-73.997849
729320	id2078912	1	2016-06-19 09:50:47	2016-06-19 09:58:14	1	-74.006706
729321	id1053441	2	2016-01-01 17:24:16	2016-01-01 17:44:40	4	-74.003342

729322 rows × 11 columns

In [3]:
```
nyc_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 729322 entries, 0 to 729321
Data columns (total 11 columns):
 #   Column              Non-Null Count   Dtype  
---  ------              --------------   -----  
 0   id                  729322 non-null  object 
 1   vendor_id           729322 non-null  int64  
 2   pickup_datetime     729322 non-null  object 
 3   dropoff_datetime    729322 non-null  object 
 4   passenger_count     729322 non-null  int64  
 5   pickup_longitude    729322 non-null  float64
 6   pickup_latitude     729322 non-null  float64
 7   dropoff_longitude   729322 non-null  float64
 8   dropoff_latitude    729322 non-null  float64
 9   store_and_fwd_flag  729322 non-null  object 
 10  trip_duration       729322 non-null  int64  
dtypes: float64(4), int64(3), object(4)
memory usage: 61.2+ MB
```

In [4]:
```
nyc_df.describe()
```

Out[4]:

	vendor_id	passenger_count	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff
count	729322.000000	729322.000000	729322.000000	729322.000000	729322.000000	72932
mean	1.535403	1.662055	-73.973513	40.750919	-73.973422	4
std	0.498745	1.312446	0.069754	0.033594	0.069588	
min	1.000000	0.000000	-121.933342	34.712234	-121.933304	3
25%	1.000000	1.000000	-73.991859	40.737335	-73.991318	4
50%	2.000000	1.000000	-73.981758	40.754070	-73.979759	4
75%	2.000000	2.000000	-73.967361	40.768314	-73.963036	4
max	2.000000	9.000000	-65.897385	51.881084	-65.897385	4

Data Preprocessing

In [5]:
```
m = np.mean(nyc_df['trip_duration'])
std = np.std(nyc_df['trip_duration'])
nyc_df = nyc_df[nyc_df['trip_duration'] <= m + 2*std]
nyc_df = nyc_df[nyc_df['trip_duration'] >= m- 2*std]
nyc_df.trip_duration
```

Out[5]:
```
0          400
1         1100
2         1635
3         1141
4          848
          ...
729317     296
729318     315
729319     673
729320     447
729321    1224
Name: trip_duration, Length: 728274, dtype: int64
```

In [6]:
```
print(min(nyc_df['dropoff_latitude']))
print(max(nyc_df['dropoff_latitude']))
print(min(nyc_df['dropoff_longitude']))
print(max(nyc_df['dropoff_longitude']))
```

```
32.1811408996582
43.92102813720703
-121.9333038330078
-65.89738464355469
```

Now looking into the longitude the coordinates lies between (-74.53,-72.71) and the latitude coordinates lies between (40.44,41.09). But the pickup_latitude and pickup_longitude and dropoff_latitude and dropoff_laongitude lies outside this range. So let's clean them.

In [7]:
```
nyc_df = nyc_df[nyc_df['dropoff_latitude']<=41.09]
nyc_df = nyc_df[nyc_df['dropoff_latitude']>=40.44]
nyc_df = nyc_df[nyc_df['dropoff_longitude']<=-73.33]
nyc_df = nyc_df[nyc_df['dropoff_longitude']>=-74.53]


nyc_df = nyc_df[nyc_df['pickup_latitude']<=41.09]
nyc_df = nyc_df[nyc_df['pickup_latitude']>=40.44]
nyc_df = nyc_df[nyc_df['pickup_longitude']<=-73.33]
nyc_df = nyc_df[nyc_df['pickup_longitude']>=-74.53]
```

In [8]:
```
nyc_df.describe()
```

Out[8]:

	vendor_id	passenger_count	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff
count	728191.000000	728191.000000	728191.000000	728191.000000	728191.000000	72819
mean	1.534779	1.661410	-73.973463	40.750925	-73.973359	4
std	0.498789	1.311762	0.038266	0.028061	0.035981	
min	1.000000	0.000000	-74.459015	40.446159	-74.511681	4
25%	1.000000	1.000000	-73.991859	40.737347	-73.991310	4
50%	2.000000	1.000000	-73.981758	40.754082	-73.979752	4
75%	2.000000	2.000000	-73.967369	40.768318	-73.963043	4
max	2.000000	9.000000	-73.330917	41.069038	-73.332817	4

Now let's change the data type of pickup_datetime and dropoff_datetime fields as they may be

useful later.

In [9]:
```
nyc_df['pickup_datetime'] = nyc_df['pickup_datetime'].astype('datetime64')
nyc_df['pickup_date'] = nyc_df['pickup_datetime'].dt.date
nyc_df['pickup_date']
```

Out[9]:
```
0         2016-02-29
1         2016-03-11
2         2016-02-21
3         2016-01-05
4         2016-02-17
             ...
729317    2016-05-21
729318    2016-02-22
729319    2016-04-15
729320    2016-06-19
729321    2016-01-01
Name: pickup_date, Length: 728191, dtype: object
```

In [10]:
```
nyc_df['dropoff_datetime'] = nyc_df['dropoff_datetime'].astype('datetime64')
nyc_df['dropoff_date'] = nyc_df['dropoff_datetime'].dt.date
nyc_df['dropoff_date']
```

Out[10]:
```
0         2016-02-29
1         2016-03-11
2         2016-02-21
3         2016-01-05
4         2016-02-17
             ...
729317    2016-05-21
729318    2016-02-22
729319    2016-04-15
729320    2016-06-19
729321    2016-01-01
Name: dropoff_date, Length: 728191, dtype: object
```

In [11]:
```
nyc_df['Month'] = nyc_df['pickup_datetime'].dt.month
nyc_df['Month']
```

Out[11]:
```
0         2
1         3
2         2
3         1
4         2
          ..
729317    5
729318    2
729319    4
729320    6
729321    1
Name: Month, Length: 728191, dtype: int64
```

In [12]:
```
nyc_df['Hour'] = nyc_df['pickup_datetime'].dt.hour
nyc_df['Hour']
```

Out[12]:
```
0         16
1         23
2         17
3          9
4          6
          ..
729317    13
729318     0
729319    18
729320     9
729321    17
Name: Hour, Length: 728191, dtype: int64
```

In [13]:
```
nyc_df['Year'] = nyc_df['pickup_datetime'].dt.year
nyc_df['Year']
```

Out[13]:
```
0         2016
1         2016
2         2016
3         2016
4         2016
          ...
729317    2016
729318    2016
729319    2016
729320    2016
729321    2016
Name: Year, Length: 728191, dtype: int64
```

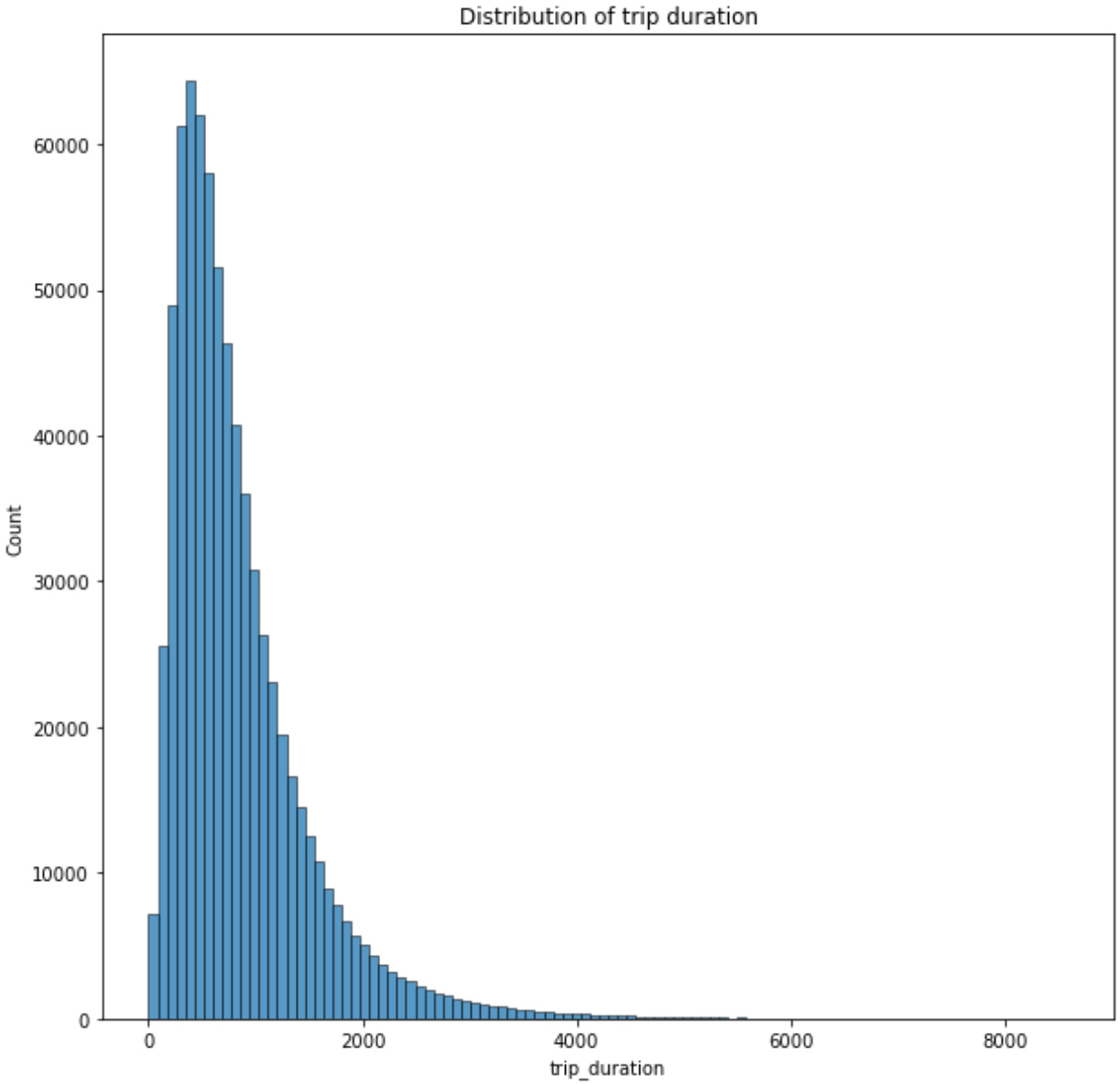
EDA

In [14]:
```
nyc_df.columns
```

Out[14]:
```
Index(['id', 'vendor_id', 'pickup_datetime', 'dropoff_datetime',
       'passenger_count', 'pickup_longitude', 'pickup_latitude',
       'dropoff_longitude', 'dropoff_latitude', 'store_and_fwd_flag',
       'trip_duration', 'pickup_date', 'dropoff_date', 'Month', 'Hour',
       'Year'],
      dtype='object')
```

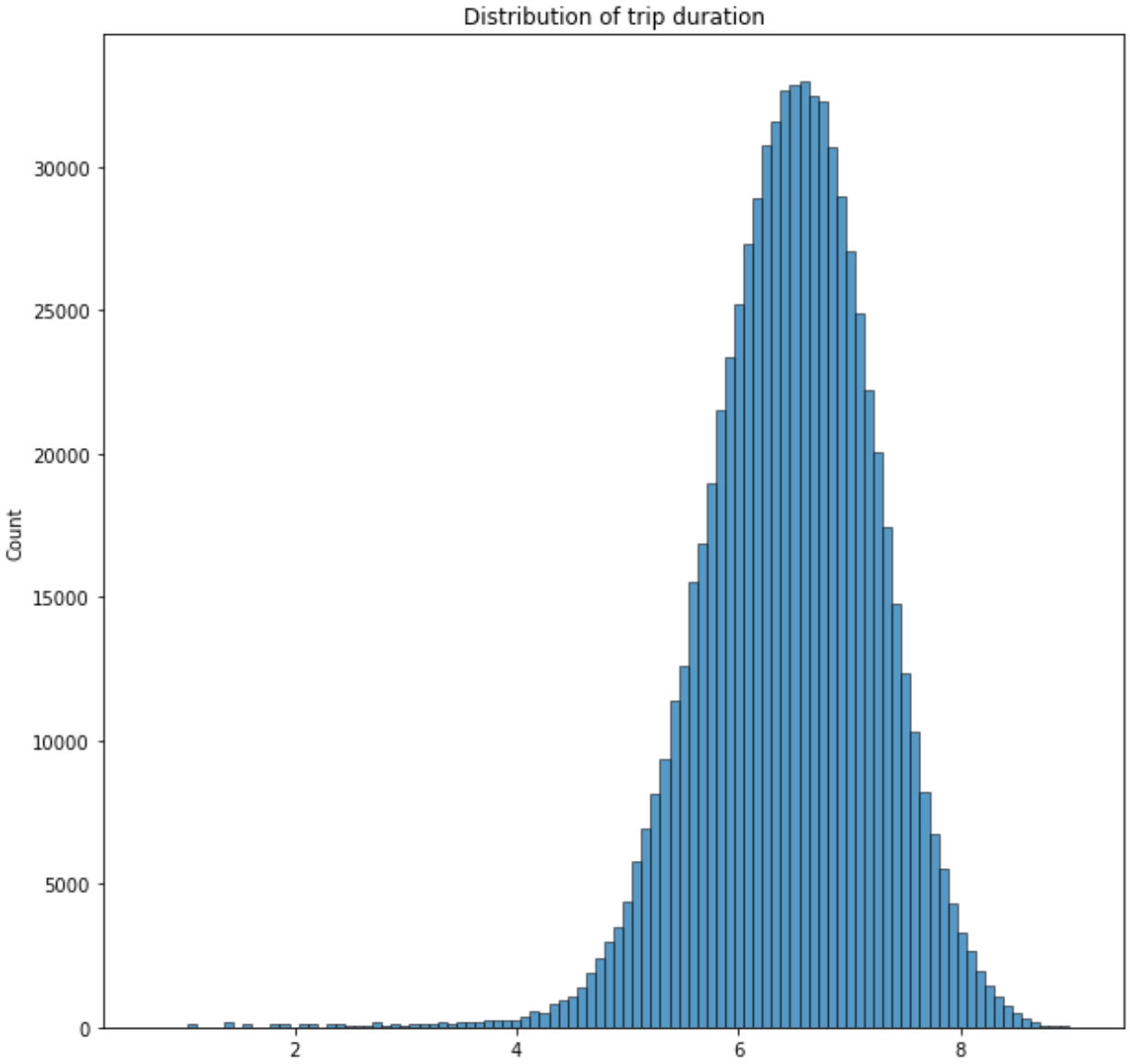
In [15]:
```
plt.title("Distribution of trip duration")
sns.histplot(nyc_df['trip_duration'],bins=100)
```

Out[15]:
```
<AxesSubplot:title={'center':'Distribution of trip duration'}, xlabel='trip_duratio
n', ylabel='Count'>
```



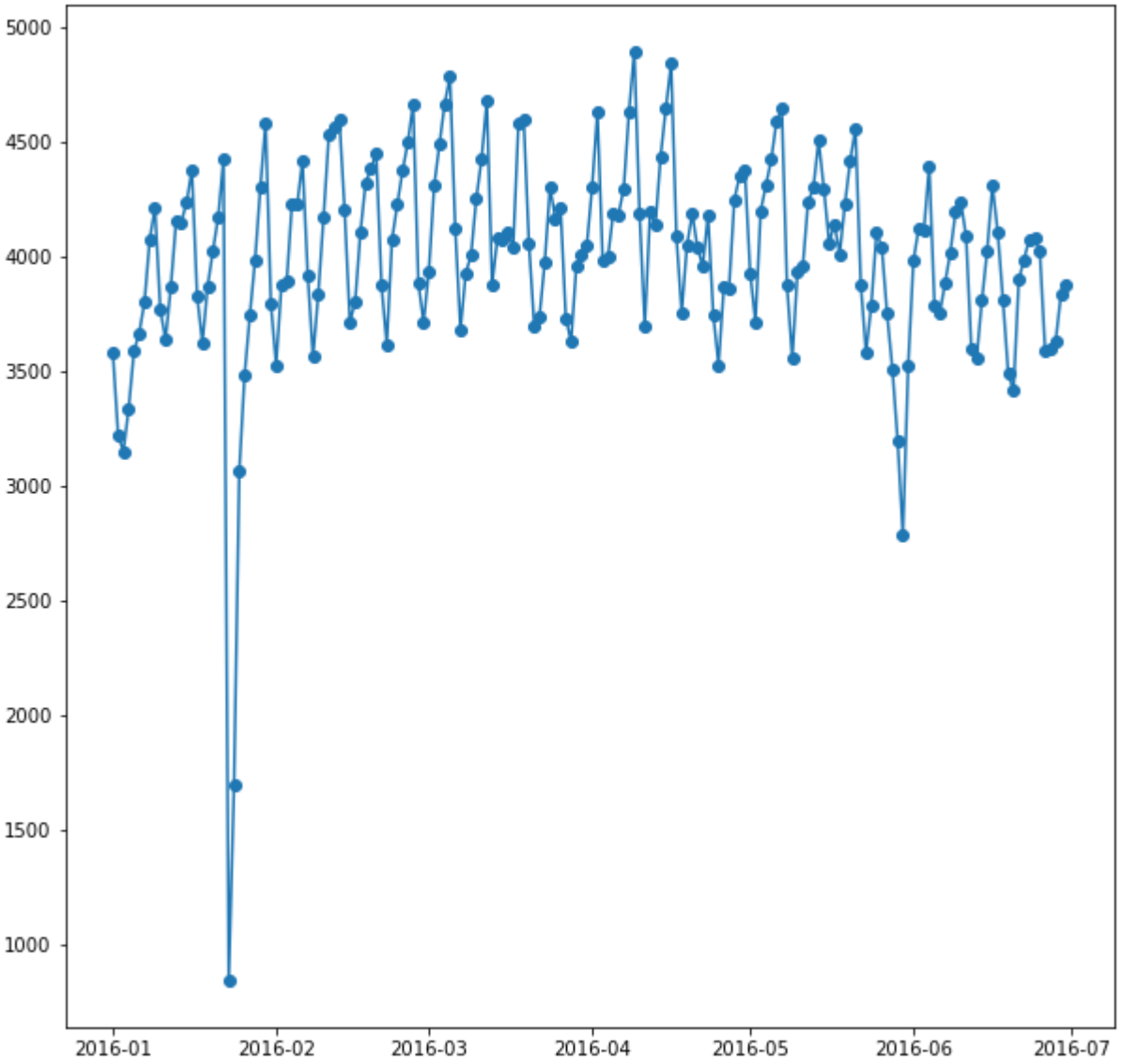
In [16]:
```
nyc_df['log_trip_duration']  = np.log(nyc_df['trip_duration'].values+1)
plt.title('Distribution of trip duration')
sns.histplot(nyc_df['log_trip_duration'].values,bins=100)
```

Out[16]:
```
<AxesSubplot:title={'center':'Distribution of trip duration'}, ylabel='Count'>
```



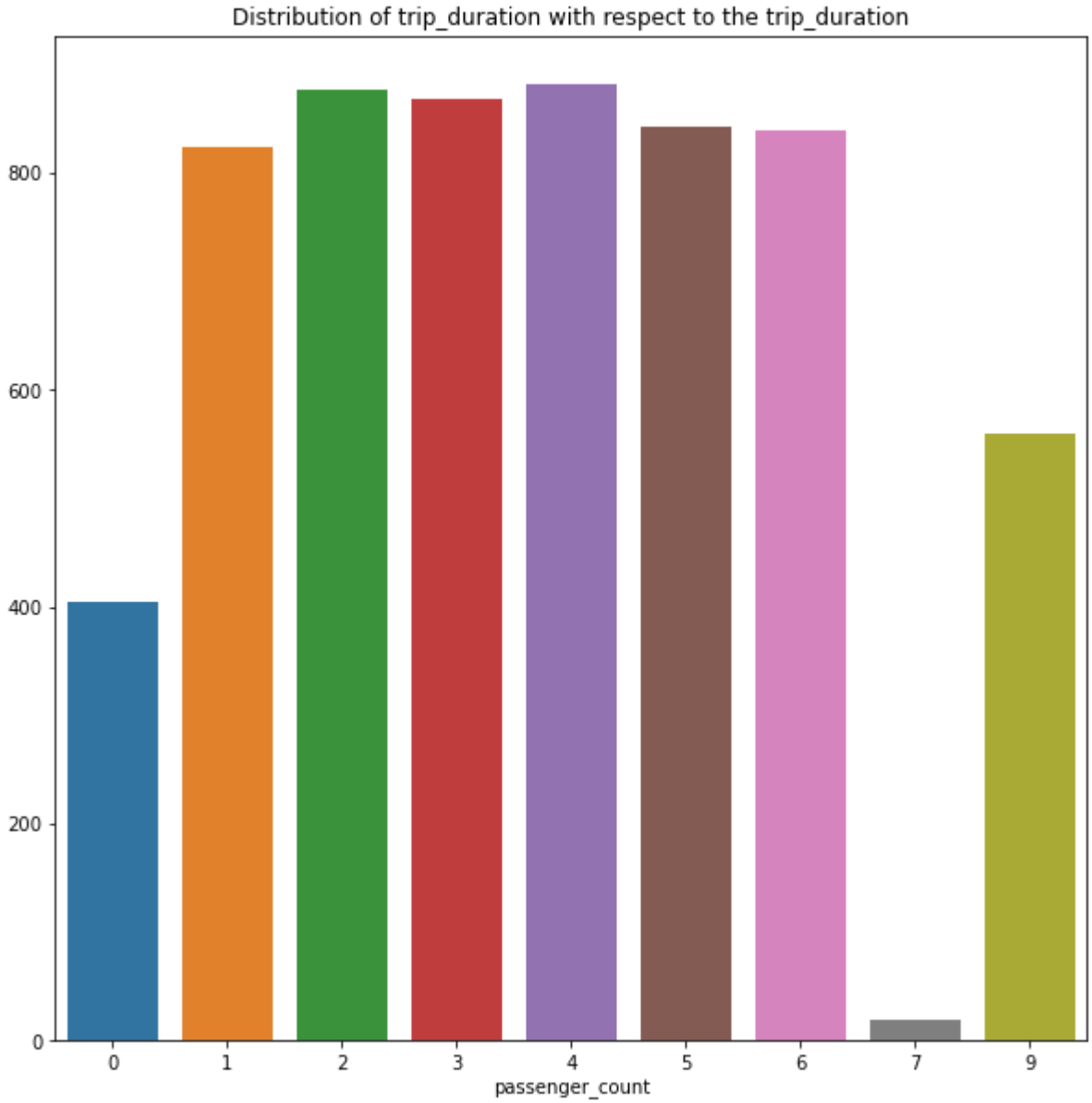
In [17]:
```
nyc_df.groupby('pickup_date').count()['id']
plt.plot(nyc_df.groupby('pickup_date').count()['id'], 'o-', label='train')
```

Out[17]:
```
[<matplotlib.lines.Line2D at 0x1e81c8c5250>]
```



In [18]:
```
df = nyc_df.groupby('passenger_count')['trip_duration'].mean()
plt.title('Distribution of trip_duration with respect to the trip_duration')
sns.barplot(df.index,df.values)
```

Out[18]:
```
<AxesSubplot:title={'center':'Distribution of trip_duration with respect to the trip_
duration'}, xlabel='passenger_count'>
```

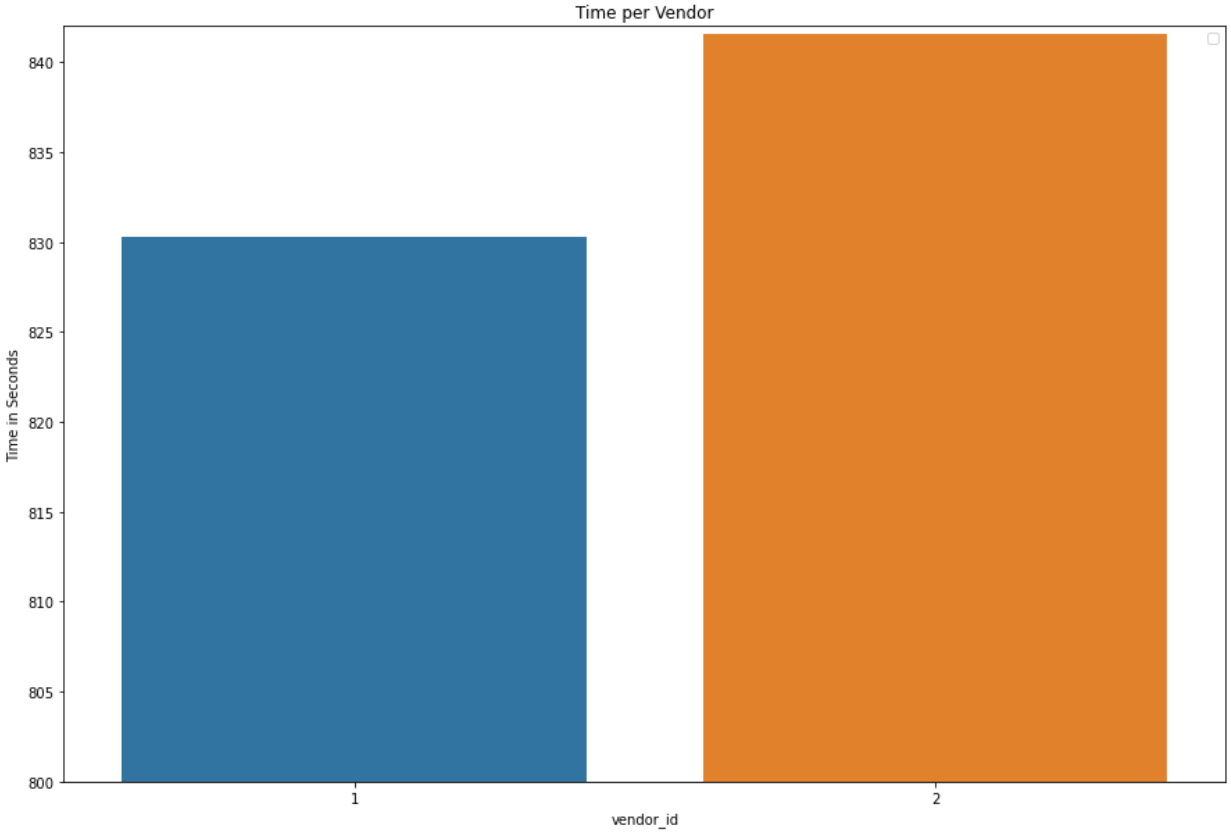


We can see that the number of passengers has nothing to do with the trip_duration

In [19]:
```
df = nyc_df.groupby('vendor_id')['trip_duration'].mean()
print(df)
plt.subplots(1,1,figsize=(15,10))
plt.ylim(ymin=800)
plt.ylim(ymax=842)
sns.barplot(df.index,df.values)
plt.title('Time per Vendor')
plt.legend(loc=0)
plt.ylabel('Time in Seconds')
```

```
No artists with labels found to put in legend.  Note that artists whose label start w
ith an underscore are ignored when legend() is called with no argument.
vendor_id
1    830.268890
2    841.574204
Name: trip_duration, dtype: float64
Text(0, 0.5, 'Time in Seconds')
```

Out[19]:

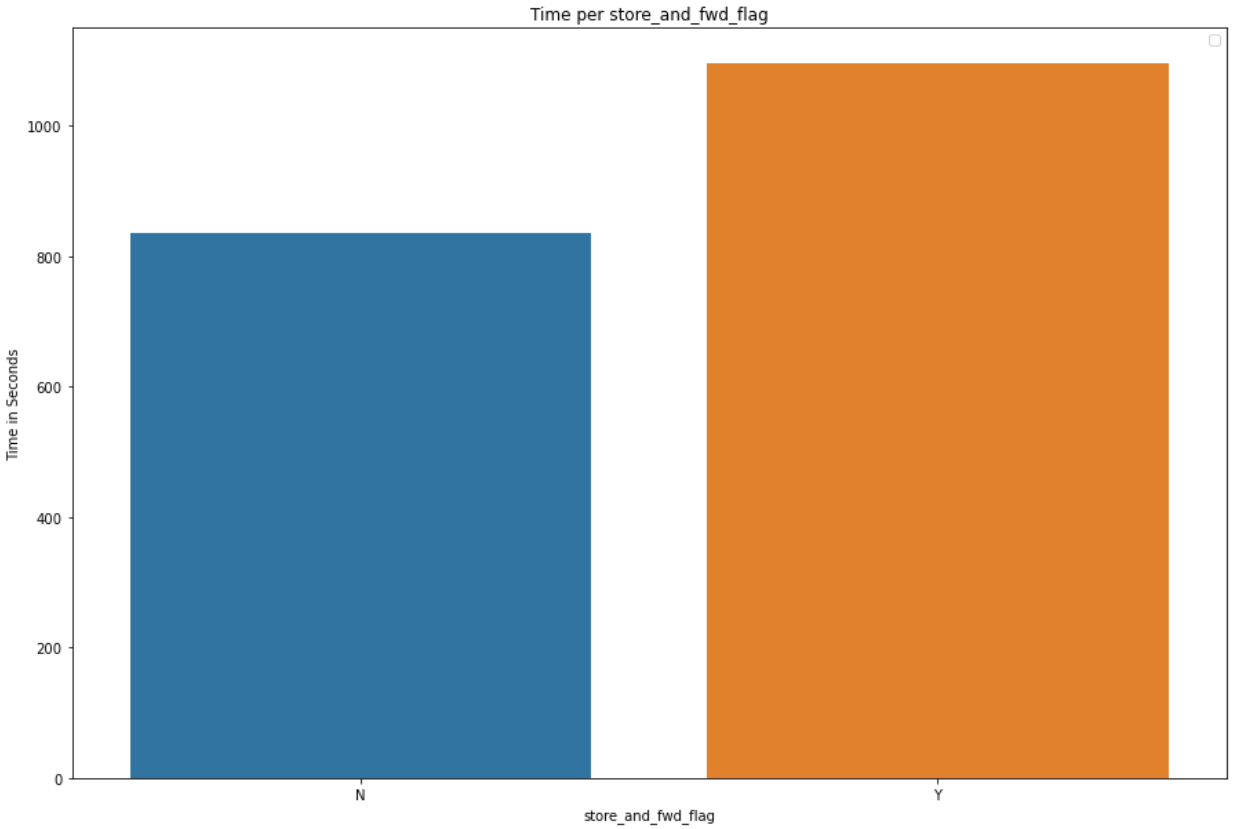


seem to be like the trip_duration differs between the vendors.

In [20]:
```
df = nyc_df.groupby('store_and_fwd_flag')['trip_duration'].mean()
plt.subplots(1,1,figsize=(15,10))
plt.title('Time per store_and_fwd_flag')
plt.legend(loc=0)
plt.ylabel('Time in Seconds')
sns.barplot(df.index,df.values)
```

```
No artists with labels found to put in legend.  Note that artists whose label start w
ith an underscore are ignored when legend() is called with no argument.
```

Out[20]:
```
<AxesSubplot:title={'center':'Time per store_and_fwd_flag'}, xlabel='store_and_fwd_fl
ag', ylabel='Time in Seconds'>
```



So it would seem that the store_and_fwd_flag discriminates well between travel times. Clearly there is a slight skew in the data where some of the vendor employees didn't record their travel times accurately.

Splitting the data

In [21]:
```
from sklearn.model_selection import train_test_split
train_val_df , test_df = train_test_split(nyc_df,test_size=0.2,random_state=42)
train_df , val_df = train_test_split(train_val_df,test_size=0.25,random_state=42)
```

In [22]:
```
train_df
```

Out[22]:

	id	vendor_id	pickup_datetime	dropoff_datetime	passenger_count	pickup_longitude
286013	id0067398	1	2016-03-05 04:07:20	2016-03-05 04:18:30	1	-74.008812
21787	id3410913	1	2016-05-10 10:01:21	2016-05-10 10:10:32	1	-73.953125
95884	id0620121	1	2016-05-22 14:03:24	2016-05-22 14:18:34	1	-73.976868
393606	id2297203	2	2016-02-24 17:11:32	2016-02-24 17:14:00	1	-73.980225
654514	id3648168	2	2016-04-27 20:42:52	2016-04-27 21:29:53	1	-73.782257
...	...	...	...	...	...	...
371761	id2833545	2	2016-02-05 11:11:58	2016-02-05 11:18:12	1	-73.975761
119424	id2060517	2	2016-04-10 01:38:31	2016-04-10 01:56:12	5	-73.991692
390313	id2833290	1	2016-03-21 21:01:32	2016-03-21 21:07:41	1	-73.990349
580262	id3692355	2	2016-02-13 14:58:07	2016-02-13 15:02:56	1	-73.978630
284542	id0235093	2	2016-05-23 06:14:35	2016-05-23 06:16:25	1	-73.978233

436914 rows × 17 columns

In [23]:
```
train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 436914 entries, 286013 to 284542
Data columns (total 17 columns):
 #   Column              Non-Null Count   Dtype
---  ------              --------------   -----
 0   id                  436914 non-null  object
 1   vendor_id           436914 non-null  int64
 2   pickup_datetime     436914 non-null  datetime64[ns]
 3   dropoff_datetime    436914 non-null  datetime64[ns]
 4   passenger_count     436914 non-null  int64
 5   pickup_longitude    436914 non-null  float64
 6   pickup_latitude     436914 non-null  float64
 7   dropoff_longitude   436914 non-null  float64
 8   dropoff_latitude    436914 non-null  float64
 9   store_and_fwd_flag  436914 non-null  object
 10  trip_duration       436914 non-null  int64
 11  pickup_date         436914 non-null  object
 12  dropoff_date        436914 non-null  object
 13  Month               436914 non-null  int64
 14  Hour                436914 non-null  int64
 15  Year                436914 non-null  int64
 16  log_trip_duration   436914 non-null  float64
dtypes: datetime64[ns](2), float64(5), int64(6), object(4)
memory usage: 60.0+ MB
```

Identifying input and output columns

In [24]:
```
train_df.corr()
```

Out[24]:

	vendor_id	passenger_count	pickup_longitude	pickup_latitude	dropoff_longitude
vendor_id	1.000000	0.286284	0.014747	0.004083	0.003772
passenger_count	0.286284	1.000000	0.002214	-0.004782	-0.000170
pickup_longitude	0.014747	0.002214	1.000000	-0.146973	0.281059
pickup_latitude	0.004083	-0.004782	-0.146973	1.000000	0.048042
dropoff_longitude	0.003772	-0.000170	0.281059	0.048042	1.000000
dropoff_latitude	0.006103	-0.003731	0.041242	0.431060	0.123774
trip_duration	0.008507	0.014596	0.359654	-0.236989	0.220217
Month	-0.005568	-0.000834	0.006320	-0.002362	0.005533
Hour	0.011499	0.010299	0.019828	0.012568	-0.041875
Year	NaN	NaN	NaN	NaN	NaN
log_trip_duration	0.012210	0.017886	0.212200	-0.176261	0.144543

In [25]:
```
input_col = ['vendor_id','pickup_datetime','passenger_count','pickup_latitude','dropof
target_col = 'trip_duration'
```

In [26]:
```
train_inputs = train_df[input_col].copy()
train_targets = train_df[target_col].copy()
```

In [27]:
```
val_inputs = val_df[input_col].copy()
val_targets = val_df[target_col].copy()
```

In [28]:
```
test_inputs = test_df[input_col].copy()
test_targets = test_df[target_col].copy()
```

In [29]:
```
numeric_col = train_inputs.select_dtypes(include=np.number).columns.tolist()
cate_col = train_inputs.select_dtypes('object').columns.tolist()
```

In [30]:
```
train_inputs[numeric_col]
```

Out[30]:

	vendor_id	passenger_count	pickup_latitude	dropoff_latitude	dropoff_longitude
286013	1	1	40.713646	40.692276	-73.987366
21787	1	1	40.785812	40.776913	-73.957542
95884	1	1	40.745228	40.774494	-73.872375
393606	2	1	40.783470	40.787838	-73.974861
654514	2	1	40.644661	40.765869	-73.988670
...	...	...	...	...	...
371761	2	1	40.788940	40.803177	-73.967346
119424	2	5	40.726608	40.779530	-73.981941
390313	1	1	40.751183	40.767376	-73.982262
580262	2	1	40.764622	40.771458	-73.982422
284542	2	1	40.752110	40.759312	-73.974487

436914 rows × 5 columns

In [31]:
```
train_inputs[cate_col]
```

Out[31]:

	store_and_fwd_flag
286013	N
21787	N
95884	N
393606	N
654514	N
...	...
371761	N
119424	N
390313	N
580262	N
284542	N

436914 rows × 1 columns

Imputing missing values

In [32]:
```
train_inputs[numeric_col].isna().sum()
```

Out[32]:
```
vendor_id           0
passenger_count     0
pickup_latitude     0
dropoff_latitude    0
dropoff_longitude   0
dtype: int64
```

In [33]:
```
val_inputs[numeric_col].isna().sum()
```

Out[33]:
```
vendor_id           0
passenger_count     0
pickup_latitude     0
dropoff_latitude    0
dropoff_longitude   0
dtype: int64
```

In [34]:
```
test_inputs[numeric_col].isna().sum()
```

Out[34]:
```
vendor_id           0
passenger_count     0
pickup_latitude     0
dropoff_latitude    0
dropoff_longitude   0
dtype: int64
```

It seems like there are no missing values in the train,validation and test datasets.

Scaling numeric values

In [35]:
```
train_inputs[numeric_col].describe()
```

Out[35]:

	vendor_id	passenger_count	pickup_latitude	dropoff_latitude	dropoff_longitude
count	436914.000000	436914.000000	436914.000000	436914.000000	436914.000000
mean	1.534318	1.659494	40.750896	40.751766	-73.973360
std	0.498821	1.309079	0.028091	0.032287	0.036056
min	1.000000	0.000000	40.496761	40.467426	-74.479622
25%	1.000000	1.000000	40.737331	40.735928	-73.991310
50%	2.000000	1.000000	40.754086	40.754517	-73.979774
75%	2.000000	2.000000	40.768318	40.769730	-73.963058
max	2.000000	9.000000	41.069038	41.089260	-73.341927

In [36]:
```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
```

In [37]:
```
scaler.fit(train_inputs[numeric_col])
```

Out[37]:
```
MinMaxScaler()
```

In [38]:
```
train_inputs[numeric_col] = scaler.transform(train_inputs[numeric_col])
val_inputs[numeric_col]   = scaler.transform(val_inputs[numeric_col])
test_inputs[numeric_col] = scaler.transform(test_inputs[numeric_col])
```

In [39]:
```
train_inputs[numeric_col].describe()
```

Out[39]:

	vendor_id	passenger_count	pickup_latitude	dropoff_latitude	dropoff_longitude
count	436914.000000	436914.000000	436914.000000	436914.000000	436914.000000
mean	0.534318	0.184388	0.444077	0.457260	0.444989
std	0.498821	0.145453	0.049087	0.051922	0.031692
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.111111	0.420373	0.431789	0.429211
50%	1.000000	0.111111	0.449650	0.461683	0.439351
75%	1.000000	0.222222	0.474520	0.486148	0.454044
max	1.000000	1.000000	1.000000	1.000000	1.000000

Encoding categorical columns

In [40]:
```
from sklearn.preprocessing import OneHotEncoder
encoder  = OneHotEncoder(sparse=False,handle_unknown='ignore')
```

In [41]:
```
encoder.fit(train_inputs[cate_col])
```

Out[41]:
```
OneHotEncoder(handle_unknown='ignore', sparse=False)
```

In [42]:
```
enc_col = encoder.get_feature_names(cate_col).tolist()
enc_col
```

Out[42]:
```
['store_and_fwd_flag_N', 'store_and_fwd_flag_Y']
```

In [43]:
```
train_inputs[enc_col] = encoder.transform(train_inputs[cate_col])
val_inputs[enc_col] = encoder.transform(val_inputs[cate_col])
test_inputs[enc_col] = encoder.transform(test_inputs[cate_col])
```

In [44]:
```
train_inputs[enc_col]
```

Out[44]:

	store_and_fwd_flag_N	store_and_fwd_flag_Y
286013	1.0	0.0
21787	1.0	0.0
95884	1.0	0.0
393606	1.0	0.0
654514	1.0	0.0
...	...	...
371761	1.0	0.0
119424	1.0	0.0
390313	1.0	0.0
580262	1.0	0.0
284542	1.0	0.0

436914 rows × 2 columns

In [45]:
```
val_inputs
```

Out[45]:

	vendor_id	pickup_datetime	passenger_count	pickup_latitude	dropoff_latitude	dropoff_long
693724	0.0	2016-01-21 10:07:05	0.111111	0.456982	0.490056	0.4
427142	1.0	2016-04-05 10:05:30	0.111111	0.444830	0.448764	0.4
486052	1.0	2016-02-09 21:00:17	0.444444	0.479139	0.390510	0.4
632095	1.0	2016-04-30 18:37:17	0.111111	0.397850	0.472769	0.4
444283	0.0	2016-02-29 18:20:00	0.111111	0.413054	0.341384	0.4
...	...	...	...	...	...	
498341	0.0	2016-02-16 07:49:48	0.111111	0.442304	0.432673	0.4
397765	1.0	2016-05-09 15:37:29	0.111111	0.530966	0.525710	0.4
204431	0.0	2016-06-12 13:20:18	0.111111	0.488345	0.471634	0.4
681512	1.0	2016-05-03 21:12:45	0.444444	0.409555	0.456641	0.4
326236	0.0	2016-01-18 12:17:20	0.111111	0.453103	0.454297	0.4

145638 rows × 9 columns

In [46]:
```
x_train = train_inputs[numeric_col + enc_col]
x_val = val_inputs[numeric_col+enc_col]
x_test = test_inputs[numeric_col + enc_col]
```

Model

we weren't taught decision tree model so the intructor told us that we could skip it

KNN

In [47]:
```
from sklearn.neighbors import KNeighborsClassifier as KNN
from sklearn.metrics import mean_squared_error as mse
```

In [48]:
```
def elbow(k):
    test_mse=[]
    for i in k:
            reg=KNN(n_neighbors=i)
            reg.fit(x_train,train_targets)
            tmp=reg.predict(x_train)
```

```
            tmp=mse(tmp,train_targets)
            test_mse.append(tmp)
    return test_mse
```

In [ ]:
```
elbow(train_targets)
```

the KNN model was taking to long to train so the instructor said that we could skip it

LINEAR MODEL

In [48]:
```
from sklearn.linear_model import LinearRegression
```

In [49]:
```
model = LinearRegression()
```

In [50]:
```
model.fit(x_train,train_targets)
```

Out[50]:
```
LinearRegression()
```

In [51]:
```
lr=model.predict(x_train)
```

In [52]:
```
train_targets
```

Out[52]:
```
286013     670
21787      551
95884      910
393606     148
654514    2821
          ...
371761     374
119424    1061
390313     369
580262     289
284542     110
Name: trip_duration, Length: 436914, dtype: int64
```

In [53]:
```
from sklearn.metrics import mean_squared_error
mean_squared_error(lr,train_targets,squared=False)
```

Out[53]:
```
614.5914915521748
```

In [54]:
```
mean_squared_error(model.predict(x_val),val_targets,squared=False)
```

Out[54]:
```
610.9301753215624
```

In [ ]:
```

```