In [3]:

```
import pandas as pd  #pandas for using dataframe and reading csv
import numpy as np   #numpy for vector operations and basic maths
#import simplejson    #getting JSON in simplified format
import urllib        #for url stuff
#import gmaps       #for using google maps to visulalize places on maps
import re            #for processing regular expressions
import datetime      #for datetime operations
import calendar      #for calendar for datetime operations
import time          #to get the system time
import scipy         #for other dependancies
from sklearn.cluster import KMeans # for doing K-means clustering
from haversine import haversine # for calculating haversine distance
import math          #for basic maths operations
import seaborn as sns #for making plots
import matplotlib.pyplot as plt # for plotting
import os  # for os commands
#from scipy.misc import imread, imresize, imsave  # for plots
import chart_studio.plotly as py
import plotly.graph_objs as go
import plotly
from bokeh.palettes import Spectral4
from bokeh.plotting import figure, output_notebook, show
from IPython.display import HTML
from matplotlib.pyplot import *
from matplotlib import cm
from matplotlib import animation
import io
import base64
output_notebook()
plotly.offline.init_notebook_mode() # run at the start of every ipython notebook
```

(https://bokeh.pydata.org)BokehJS 1.3.4 successfully loaded.

In [4]:

```
s = time.time()
train_fr_1 = pd.read_csv('fastest_routes_train_part_1.csv')
train_fr_2 = pd.read_csv('fastest_routes_train_part_2.csv')
train_fr = pd.concat([train_fr_1, train_fr_2])
train_fr_new = train_fr[['id', 'total_distance', 'total_travel_time', 'number_of_steps']]
train_df = pd.read_csv('train.csv')
train = pd.merge(train_df, train_fr_new, on = 'id', how = 'left')
train_df = train.copy()
end = time.time()
print("Time taken by above cell is {}.".format((end-s)))
train_df.head()
```

```
Time taken by above cell is 23.12317991256714.
```

Out[4]:

	id	vendor_id	pickup_datetime	dropoff_datetime	passenger_count	pickup_longitu
0	id2875421	2	2016-03-14 17:24:55	2016-03-14 17:32:30	1	-73.9821
1	id2377394	1	2016-06-12 00:43:35	2016-06-12 00:54:38	1	-73.9804
2	id3858529	2	2016-01-19 11:35:24	2016-01-19 12:10:48	1	-73.9790
3	id3504673	2	2016-04-06 19:32:31	2016-04-06 19:39:40	1	-74.0100
4	id2181028	2	2016-03-26 13:30:55	2016-03-26 13:38:10	1	-73.9730

In [5]:

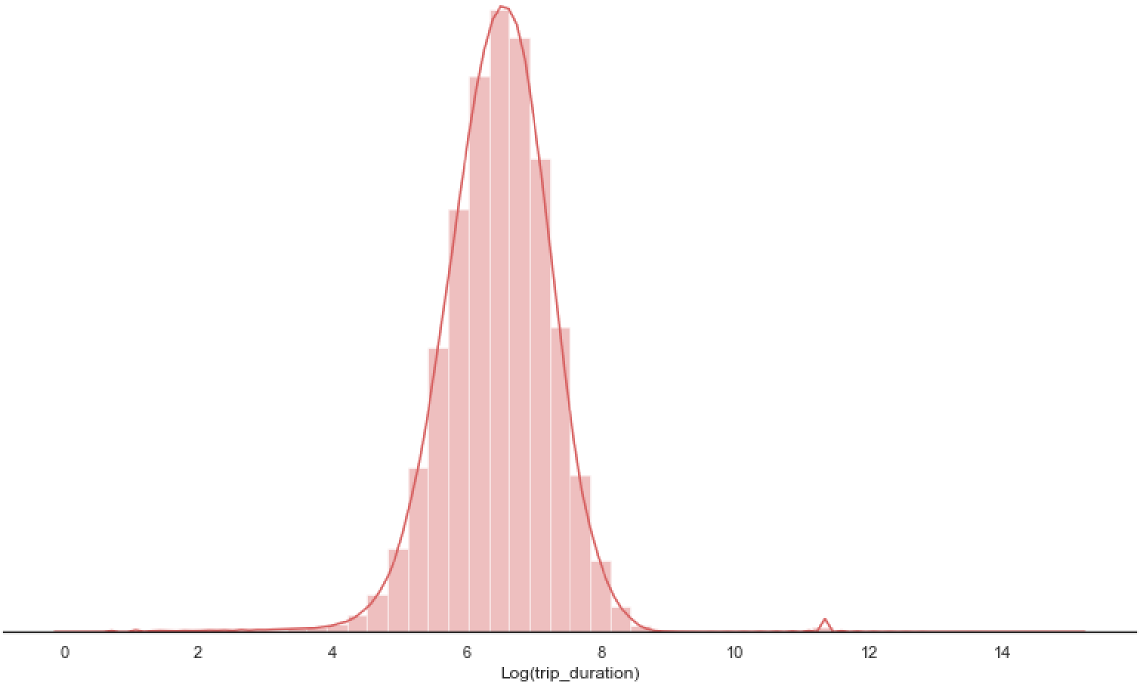
```
# checking if Ids are unique,
start = time.time()
train_data = train_df.copy()
start = time.time()
print("Number of columns and rows and columns are {} and {} respectively.".format(train_data.sh
ape[1], train_data.shape[0]))
if train_data.id.nunique() == train_data.shape[0]:
    print("Train ids are unique")
print("Number of Nulls - {}.".format(train_data.isnull().sum().sum()))
end = time.time()
print("Time taken by above cell is {}.".format(end-start))
```

```
Number of columns and rows and columns are 14 and 1458644 respectively.
Train ids are unique
Number of Nulls - 3.
Time taken by above cell is 0.6502599716186523.
```

In [8]:

```
%matplotlib inline
start = time.time()
sns.set(style="white", palette="muted", color_codes=True)
f, axes = plt.subplots(1, 1, figsize=(11, 7), sharex=True)
sns.despine(left=True)
sns.distplot(np.log(train_df['trip_duration']), axlabel = 'Log(trip_duration)', label = 'log(tri
p_duration)', bins = 50, color="r")
plt.setp(axes, yticks=[])
plt.tight_layout()
end = time.time()
print("Time taken by above cell is {}.".format((end-start)))
plt.show()
```

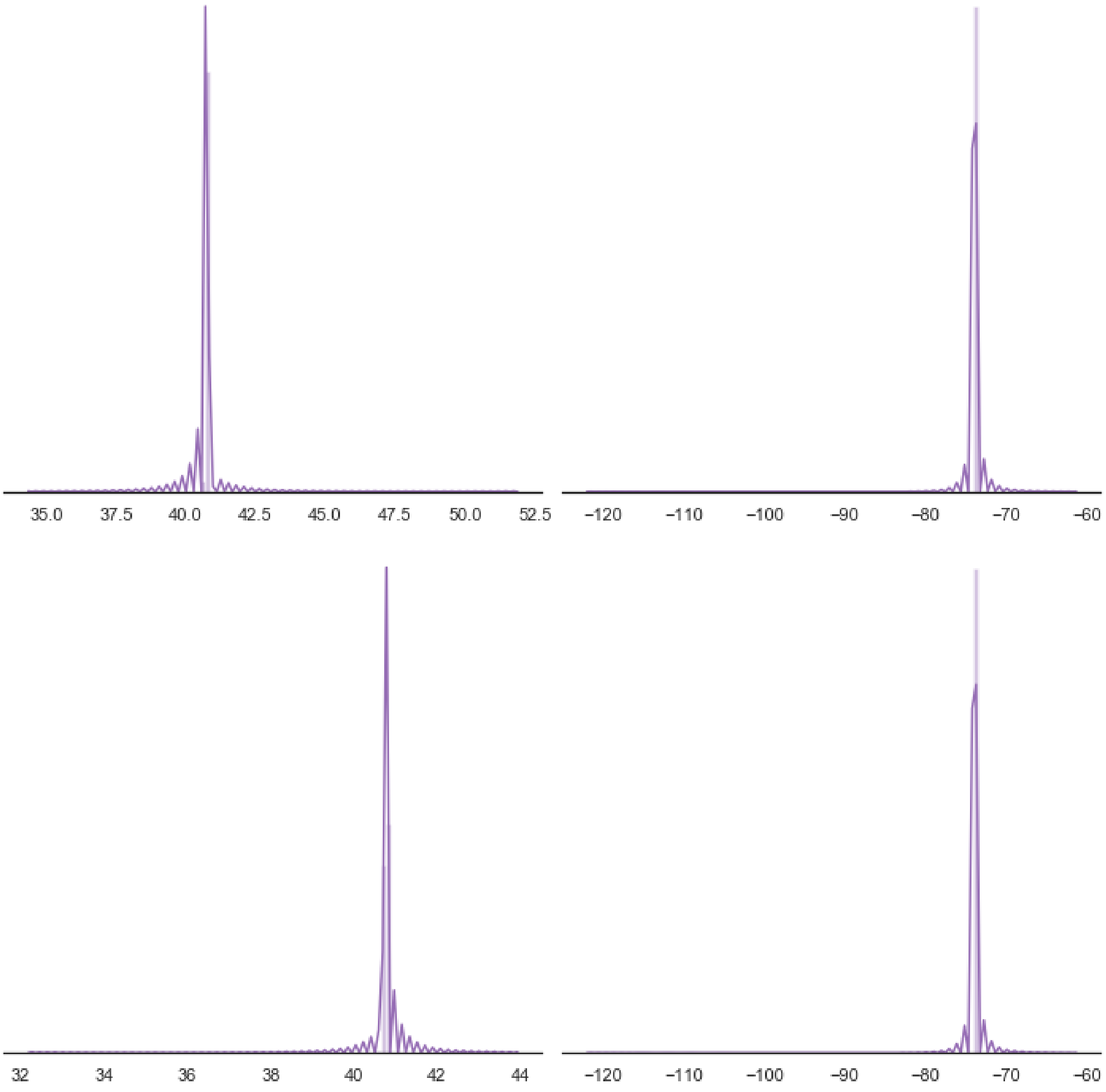
```
Time taken by above cell is 0.343648195266723 63.
```



In [9]:

```
start = time.time()
sns.set(style="white", palette="muted", color_codes=True)
f, axes = plt.subplots(2,2,figsize=(10, 10), sharex=False, sharey = False)
sns.despine(left=True)
sns.distplot(train_df['pickup_latitude'].values, label = 'pickup_latitude',color="m",bins = 100,
ax=axes[0,0])
sns.distplot(train_df['pickup_longitude'].values, label = 'pickup_longitude',color="m",bins =100
, ax=axes[0,1])
sns.distplot(train_df['dropoff_latitude'].values, label = 'dropoff_latitude',color="m",bins =100
, ax=axes[1, 0])
sns.distplot(train_df['dropoff_longitude'].values, label = 'dropoff_longitude',color="m",bins =1
00, ax=axes[1, 1])
plt.setp(axes, yticks=[])
plt.tight_layout()
end = time.time()
print("Time taken by above cell is {}.".format((end-start)))
plt.show()
```

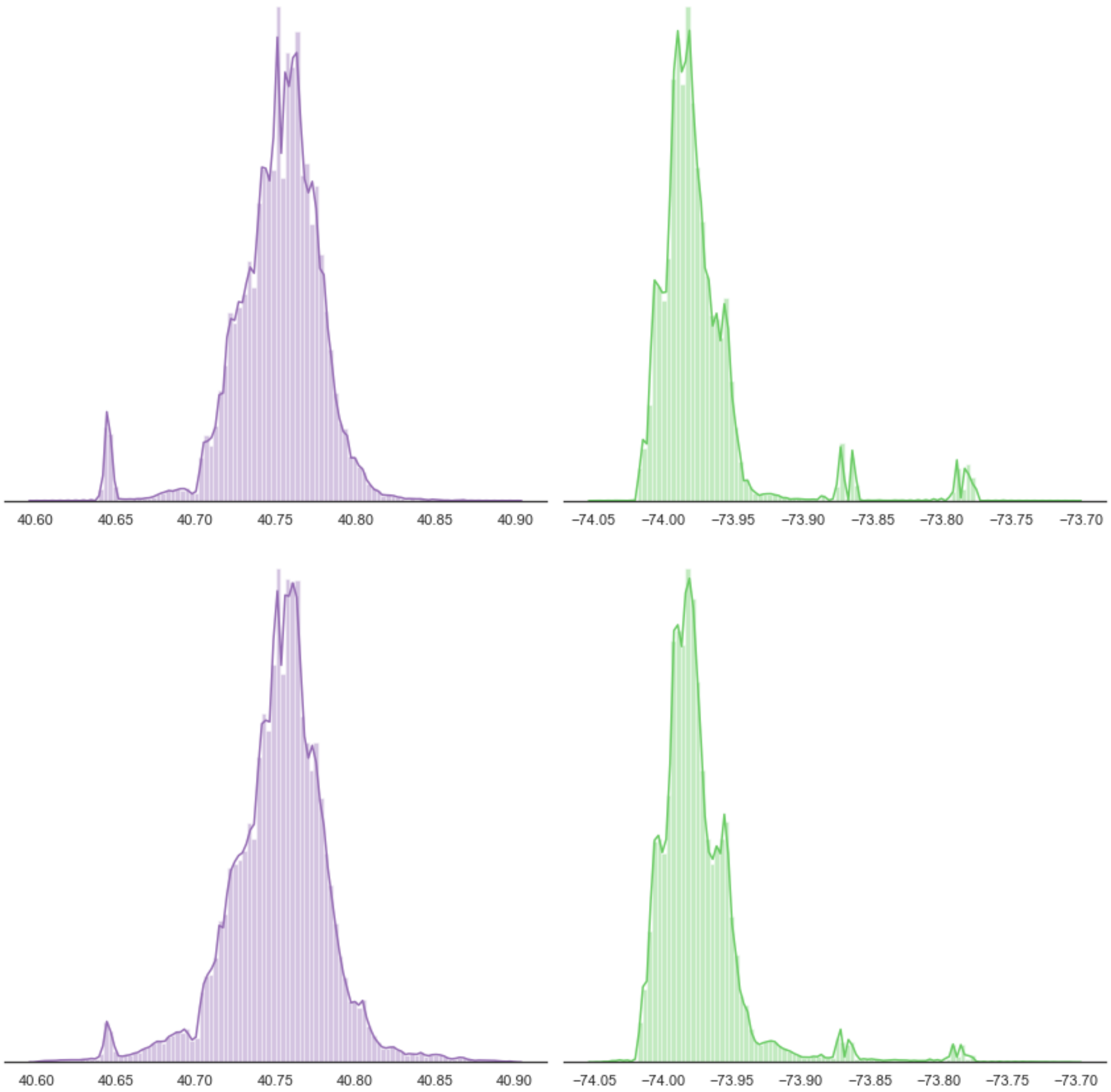
```
Time taken by above cell is 1.5637364387512207.
```



In [10]:

```
start = time.time()
df = train_df.loc[(train_df.pickup_latitude > 40.6) & (train_df.pickup_latitude < 40.9)]
df = df.loc[(df.dropoff_latitude>40.6) & (df.dropoff_latitude < 40.9)]
df = df.loc[(df.dropoff_longitude > -74.05) & (df.dropoff_longitude < -73.7)]
df = df.loc[(df.pickup_longitude > -74.05) & (df.pickup_longitude < -73.7)]
train_data_new = df.copy()
sns.set(style="white", palette="muted", color_codes=True)
f, axes = plt.subplots(2,2,figsize=(12, 12), sharex=False, sharey = False)#
sns.despine(left=True)
sns.distplot(train_data_new['pickup_latitude'].values, label = 'pickup_latitude',color="m",bins = 100, ax=axes[0,0])
sns.distplot(train_data_new['pickup_longitude'].values, label = 'pickup_longitude',color="g",bins =100, ax=axes[0,1])
sns.distplot(train_data_new['dropoff_latitude'].values, label = 'dropoff_latitude',color="m",bins =100, ax=axes[1, 0])
sns.distplot(train_data_new['dropoff_longitude'].values, label = 'dropoff_longitude',color="g",bins =100, ax=axes[1, 1])
plt.setp(axes, yticks=[])
plt.tight_layout()
end = time.time()
print("Time taken by above cell is {}.".format((end-start)))
print(df.shape[0], train_data.shape[0])
plt.show()
```

```
Time taken by above cell is 2.5546224117279053.
1452385 1458644
```



In [11]:

```
temp = train_data.copy()
train_data[’pickup_datetime’] = pd.to_datetime(train_data.pickup_datetime)
train_data.loc[:, ’pick_date’] = train_data[’pickup_datetime’].dt.date
train_data.head()
```

Out[11]:

	id	vendor_id	pickup_datetime	dropoff_datetime	passenger_count	pickup_longitu
0	id2875421	2	2016-03-14 17:24:55	2016-03-14 17:32:30	1	-73.9821
1	id2377394	1	2016-06-12 00:43:35	2016-06-12 00:54:38	1	-73.9804
2	id3858529	2	2016-01-19 11:35:24	2016-01-19 12:10:48	1	-73.9790
3	id3504673	2	2016-04-06 19:32:31	2016-04-06 19:39:40	1	-74.0100
4	id2181028	2	2016-03-26 13:30:55	2016-03-26 13:38:10	1	-73.9730

In [12]:

```
ts_v1 = pd.DataFrame(train_data.loc[train_data[’vendor_id’]==1].groupby(’pick_date’)[’trip_durat
ion’].mean())
ts_v1.reset_index(inplace = True)
ts_v2 = pd.DataFrame(train_data.loc[train_data.vendor_id==2].groupby(’pick_date’)[’trip_duratio
n’].mean())
ts_v2.reset_index(inplace = True)
```

In [14]:

```
train_data
```

Out[14]:

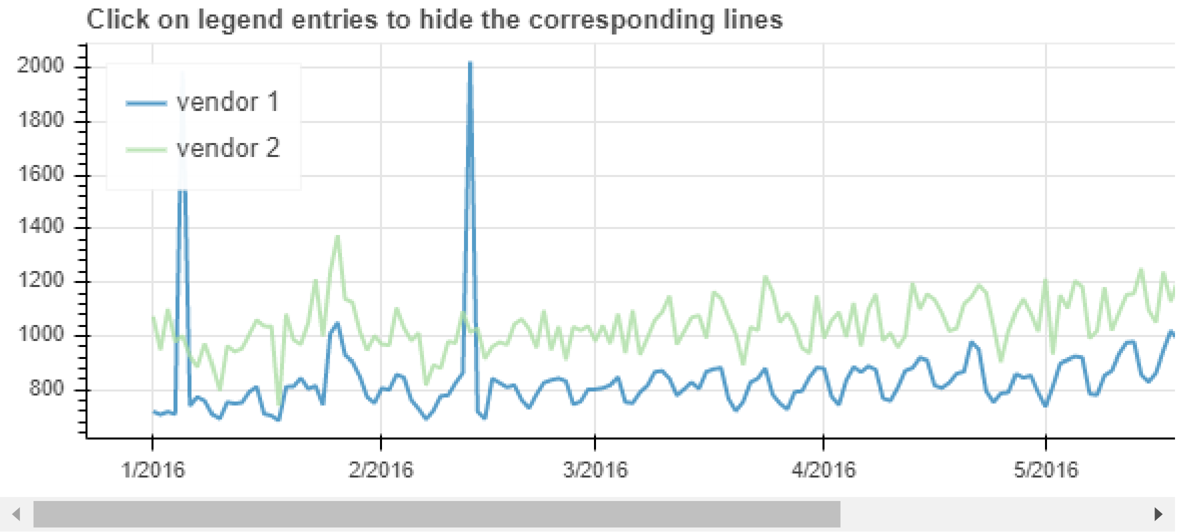
	id	vendor_id	pickup_datetime	dropoff_datetime	passenger_count	pickup_l
0	id2875421	2	2016-03-14 17:24:55	2016-03-14 17:32:30	1	-7
1	id2377394	1	2016-06-12 00:43:35	2016-06-12 00:54:38	1	-7
2	id3858529	2	2016-01-19 11:35:24	2016-01-19 12:10:48	1	-7
3	id3504673	2	2016-04-06 19:32:31	2016-04-06 19:39:40	1	-7
4	id2181028	2	2016-03-26 13:30:55	2016-03-26 13:38:10	1	-7
...	...	...	...	...	...	
1458639	id2376096	2	2016-04-08 13:31:04	2016-04-08 13:44:02	4	-7
1458640	id1049543	1	2016-01-10 07:35:15	2016-01-10 07:46:10	1	-7
1458641	id2304944	2	2016-04-22 06:57:41	2016-04-22 07:10:25	1	-7
1458642	id2714485	1	2016-01-05 15:56:26	2016-01-05 16:02:39	1	-7
1458643	id1209952	1	2016-04-05 14:44:25	2016-04-05 14:47:43	1	-7

1458644 rows × 15 columns

In [15]:

```
from bokeh.palettes import Spectral4
from bokeh.plotting import figure, output_notebook, show
#from bokeh.sampledata.stocks import AAPL, IBM, MSFT, GOOG
output_notebook()
p = figure(plot_width=800, plot_height=250, x_axis_type="datetime")
p.title.text = 'Click on legend entries to hide the corresponding lines'

for data, name, color in zip([ts_v1, ts_v2], ["vendor 1", "vendor 2"], Spectral4):
    df = data
    p.line(df['pick_date'], df['trip_duration'], line_width=2, color=color, alpha=0.8, legend=na
me)

p.legend.location = "top_left"
p.legend.click_policy="hide"
show(p)
train_data = temp
```

(https://bokeh.pydata.org)BokehJS 1.3.4 successfully loaded.



In [16]:

```
rgb = np.zeros((3000, 3500, 3), dtype=np.uint8)
rgb[..., 0] = 0
rgb[..., 1] = 0
rgb[..., 2] = 0
train_data_new['pick_lat_new'] = list(map(int, (train_data_new['pickup_latitude'] - (40.6000))*1
0000))
train_data_new['drop_lat_new'] = list(map(int, (train_data_new['dropoff_latitude'] - (40.6000))*
10000))
train_data_new['pick_lon_new'] = list(map(int, (train_data_new['pickup_longitude'] - (-74.050))*
10000))
train_data_new['drop_lon_new'] = list(map(int,(train_data_new['dropoff_longitude'] - (-74.050))*
10000))

summary_plot = pd.DataFrame(train_data_new.groupby(['pick_lat_new', 'pick_lon_new'])['id'].count
())

summary_plot.reset_index(inplace = True)
summary_plot.head(120)
```

In [17]:

```
summary_plot.head(120)
```

Out[17]:

	pick_lat_new	pick_lon_new	id
0	2	544	1
1	6	840	1
2	8	454	1
3	9	706	1
4	17	1030	1
...	...	...	...
115	220	278	1
116	222	302	1
117	222	853	1
118	222	2380	1
119	223	221	1

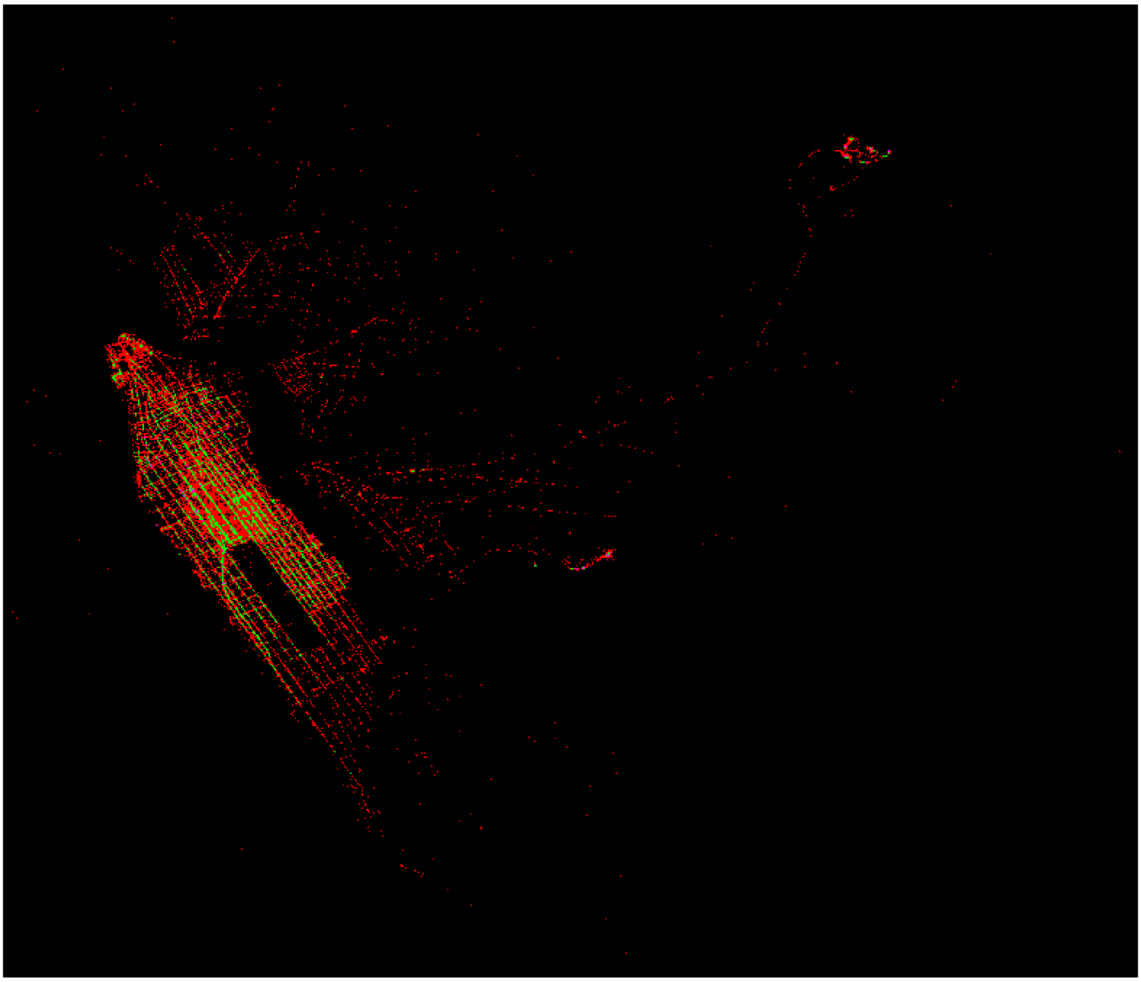
120 rows × 3 columns

In [19]:

```
lat_list = summary_plot['pick_lat_new'].unique()
```

In [20]:

```
for i in lat_list:
    lon_list = summary_plot.loc[summary_plot['pick_lat_new']==i]['pick_lon_new'].tolist()
    unit = summary_plot.loc[summary_plot['pick_lat_new']==i]['id'].tolist()
    for j in lon_list:
        a = unit[lon_list.index(j)]
        if (a//50) >0:
            rgb[i][j][0] = 255
            rgb[i, j, 1] = 0
            rgb[i, j, 2] = 255
        elif (a//10)>0:
            rgb[i, j, 0] = 0
            rgb[i, j, 1] = 255
            rgb[i, j, 2] = 0
        else:
            rgb[i, j, 0] = 255
            rgb[i, j, 1] = 0
            rgb[i, j, 2] = 0
fig, ax = plt.subplots(nrows=1,ncols=1,figsize=(14,20))
ax.imshow(rgb, cmap = 'hot')
ax.set_axis_off()
```



Findings - From the heatmap kind of image above - Red points signifies that 1-10 trips in the given data have that point as pickup point Green points signifies that more than 10-50 trips in the given data have that point as pickup point Yellow points signifies that more than 50+ trips in the given data have that point as pickup point

In [21]:

```python
start = time.time()
def haversine_(lat1, lng1, lat2, lng2):
    """function to calculate haversine distance between two co-ordinates"""
    lat1, lng1, lat2, lng2 = map(np.radians, (lat1, lng1, lat2, lng2))
    AVG_EARTH_RADIUS = 6371  # in km
    lat = lat2 - lat1
    lng = lng2 - lng1
    d = np.sin(lat * 0.5) ** 2 + np.cos(lat1) * np.cos(lat2) * np.sin(lng * 0.5) ** 2
    h = 2 * AVG_EARTH_RADIUS * np.arcsin(np.sqrt(d))
    return(h)

def manhattan_distance_pd(lat1, lng1, lat2, lng2):
    """function to calculate manhatten distance between pick_drop"""
    a = haversine_(lat1, lng1, lat1, lng2)
    b = haversine_(lat1, lng1, lat2, lng1)
    return a + b

import math
def bearing_array(lat1, lng1, lat2, lng2):
    """ function was taken from beluga's notebook as this function works on array
    while my function used to work on individual elements and was noticably slow"""
    AVG_EARTH_RADIUS = 6371  # in km
    lng_delta_rad = np.radians(lng2 - lng1)
    lat1, lng1, lat2, lng2 = map(np.radians, (lat1, lng1, lat2, lng2))
    y = np.sin(lng_delta_rad) * np.cos(lat2)
    x = np.cos(lat1) * np.sin(lat2) - np.sin(lat1) * np.cos(lat2) * np.cos(lng_delta_rad)
    return np.degrees(np.arctan2(y, x))

end = time.time()
print("Time taken by above cell is {}.".format((end-start)))
```

```
Time taken by above cell is 0.0.
```

In [22]:

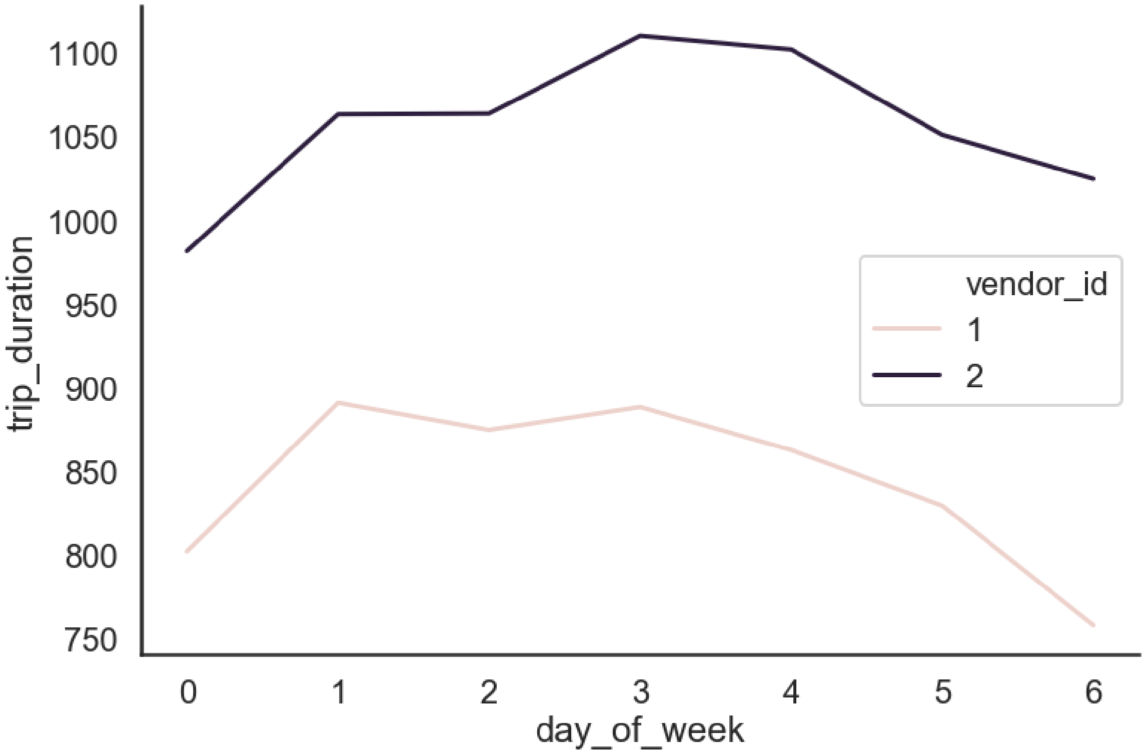
```python
start = time.time()
train_data = temp.copy()
train_data['pickup_datetime'] = pd.to_datetime(train_data.pickup_datetime)
train_data.loc[:, 'pick_month'] = train_data['pickup_datetime'].dt.month
train_data.loc[:, 'hour'] = train_data['pickup_datetime'].dt.hour
train_data.loc[:, 'week_of_year'] = train_data['pickup_datetime'].dt.weekofyear
train_data.loc[:, 'day_of_year'] = train_data['pickup_datetime'].dt.dayofyear
train_data.loc[:, 'day_of_week'] = train_data['pickup_datetime'].dt.dayofweek
train_data.loc[:,'hvsine_pick_drop'] = haversine_(train_data['pickup_latitude'].values, train_data['pickup_longitude'].values, train_data['dropoff_latitude'].values, train_data['dropoff_longitude'].values)
train_data.loc[:,'manhtn_pick_drop'] = manhattan_distance_pd(train_data['pickup_latitude'].values, train_data['pickup_longitude'].values, train_data['dropoff_latitude'].values, train_data['dropoff_longitude'].values)
train_data.loc[:,'bearing'] = bearing_array(train_data['pickup_latitude'].values, train_data['pickup_longitude'].values, train_data['dropoff_latitude'].values, train_data['dropoff_longitude'].values)

end = time.time()
print("Time taken by above cell is {}.".format(end-start))
```

```
Time taken by above cell is 4.230079889297485.
```

In [61]:

```
start = time.time()
summary_wdays_avg_duration = pd.DataFrame(train_data.groupby(['vendor_id','day_of_week'])['trip_
duration'].mean())
summary_wdays_avg_duration.reset_index(inplace = True)
sns.set(style="white", palette="muted", color_codes=True)
sns.set_context("poster")
plt.figure(figsize=(12, 8))
sns.lineplot(x="day_of_week", y="trip_duration", hue="vendor_id",legend="full",data=summary_wday
s_avg_duration)
sns.despine(bottom = False)
end = time.time()
print(end - start)
```

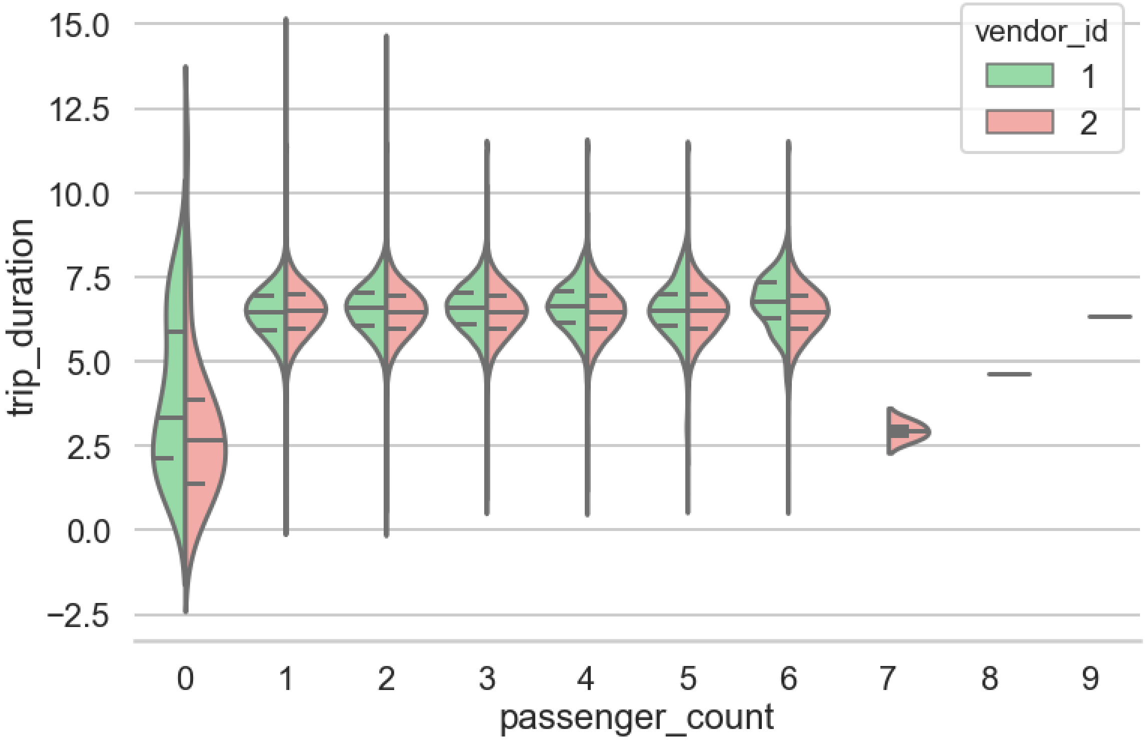
```
0.21294689178466797
```



In [63]:

```
import seaborn as sns
sns.set(style="whitegrid", palette="pastel", color_codes=True)
sns.set_context("poster")
train_data2 = train_data.copy()
train_data2['trip_duration']= np.log(train_data['trip_duration'])
plt.figure(figsize=(12, 8))
sns.violinplot(x="passenger_count", y="trip_duration", hue="vendor_id", data=train_data2, split=
True,
               inner="quart",palette={1: "g", 2: "r"})

sns.despine(left=True)
print(df.shape[0])
```

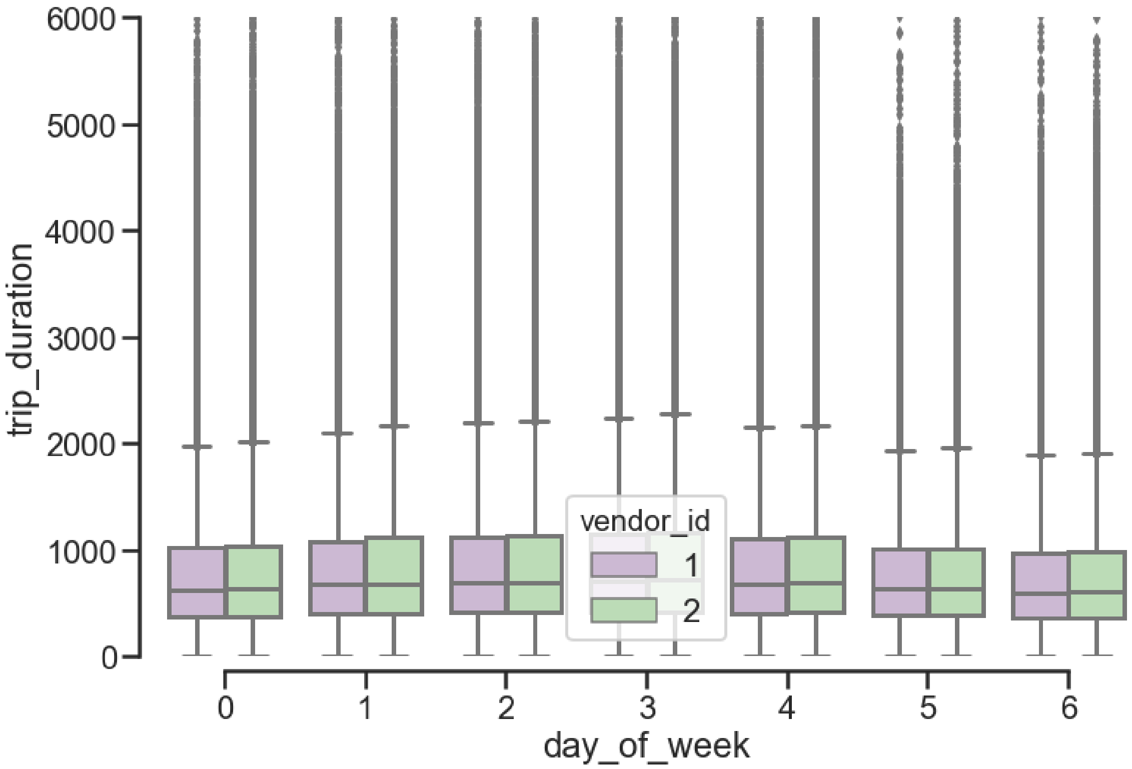
```
182
```



In [71]:

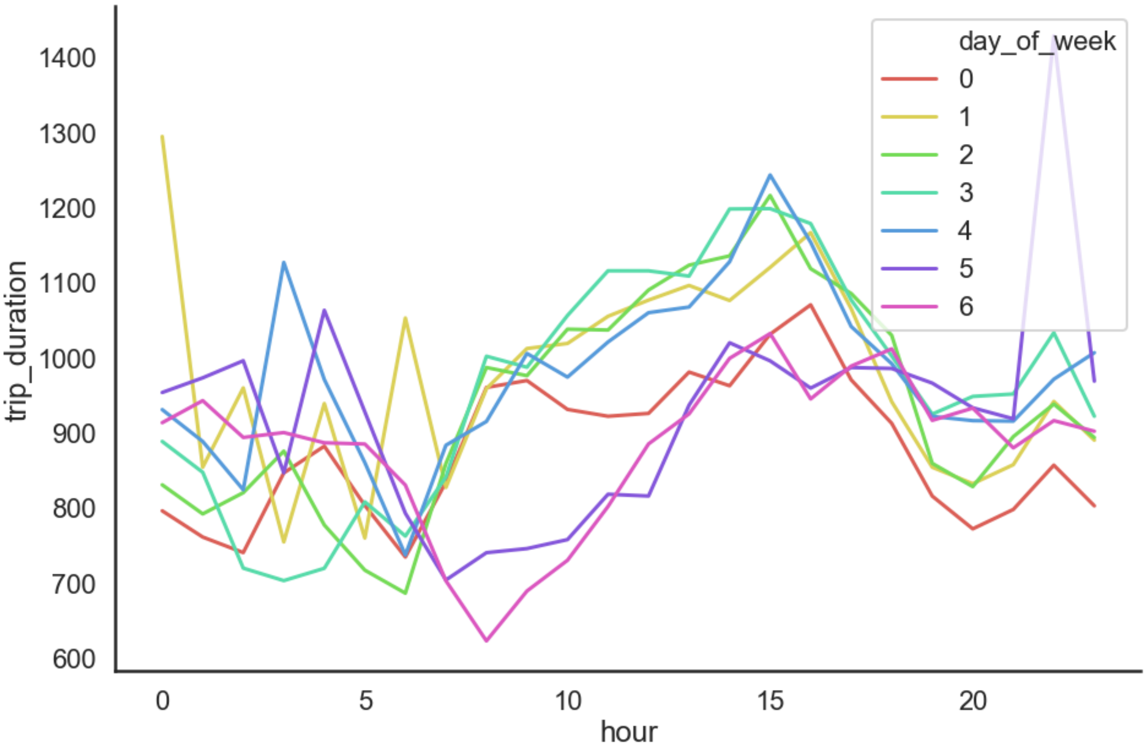
```
start = time.time()
sns.set(style="ticks")
sns.set_context("poster")
plt.figure(figsize=(12, 8))
plt.ylim(0, 6000)
sns.boxplot(x="day_of_week", y="trip_duration", hue="vendor_id", data=train_data, palette="PRGn"
)
sns.despine(offset=10, trim=True)
print(train_data.trip_duration.max())
end = time.time()
print("Time taken by above cell is {}.".format(end-start))
```

```
3526282
Time taken by above cell is 0.452831506729126.
```



In [86]:

```
summary_hour_duration = pd.DataFrame(train_data.groupby(['day_of_week','hour'])['trip_duration']
.mean())
summary_hour_duration.reset_index(inplace = True)
sns.set_context("poster")
palette = sns.color_palette("hls", 7)
plt.figure(figsize=(15, 10))
sns.lineplot(data=summary_hour_duration, x="hour", hue="day_of_week", y="trip_duration",legend =
'full',palette = palette)
sns.despine(bottom = False)
```



In [87]:

```
def assign_cluster(df, k):
    """function to assign clusters """
    df_pick = df[['pickup_longitude','pickup_latitude']]
    df_drop = df[['dropoff_longitude','dropoff_latitude']]
    """I am using initialization as from the output of
    k-means from my local machine to save time in this kernel"""
    init = np.array([[ -73.98737616,   40.72981533],
       [-121.93328857,   37.38933945],
       [ -73.78423222,   40.64711269],
       [ -73.9546417 ,   40.77377538],
       [ -66.84140269,   36.64537175],
       [ -73.87040541,   40.77016484],
       [ -73.97316185,   40.75814346],
       [ -73.98861094,   40.7527791 ],
       [ -72.80966949,   51.88108444],
       [ -76.99779701,   38.47370625],
       [ -73.96975298,   40.69089596],
       [ -74.00816622,   40.71414939],
       [ -66.97216034,   44.37194443],
       [ -61.33552933,   37.85105133],
       [ -73.98001393,   40.7783577 ],
       [ -72.00626526,   43.20296402],
       [ -73.07618713,   35.03469086],
       [ -73.95759366,   40.80316361],
       [ -79.20167796,   41.04752096],
       [ -74.00106031,   40.73867723]])
    k_means_pick = KMeans(n_clusters=k, init=init, n_init=1)
    k_means_pick.fit(df_pick)
    clust_pick = k_means_pick.labels_
    df['label_pick'] = clust_pick.tolist()
    df['label_drop'] = k_means_pick.predict(df_drop)
    return df, k_means_pick
```

In [88]:

```
train_cl, k_means = assign_cluster(train_data, 20)  # make it 100 when extracting features
```

In [95]:

```
k_means.labels_
```

Out[95]:

```
array([14,  0,  6, ...,  3,  7, 14])
```

In [91]:

```
train_cl
```

Out[91]:

	id	vendor_id	pickup_datetime	dropoff_datetime	passenger_count	pickup_l
0	id2875421	2	2016-03-14 17:24:55	2016-03-14 17:32:30	1	-7
1	id2377394	1	2016-06-12 00:43:35	2016-06-12 00:54:38	1	-7
2	id3858529	2	2016-01-19 11:35:24	2016-01-19 12:10:48	1	-7
3	id3504673	2	2016-04-06 19:32:31	2016-04-06 19:39:40	1	-7
4	id2181028	2	2016-03-26 13:30:55	2016-03-26 13:38:10	1	-7
...	...	...	...	...	...	
1458639	id2376096	2	2016-04-08 13:31:04	2016-04-08 13:44:02	4	-7
1458640	id1049543	1	2016-01-10 07:35:15	2016-01-10 07:46:10	1	-7
1458641	id2304944	2	2016-04-22 06:57:41	2016-04-22 07:10:25	1	-7
1458642	id2714485	1	2016-01-05 15:56:26	2016-01-05 16:02:39	1	-7
1458643	id1209952	1	2016-04-05 14:44:25	2016-04-05 14:47:43	1	-7

1458644 rows × 24 columns

In [99]:

```
centroid_pickups = pd.DataFrame(k_means.cluster_centers_, columns = ['centroid_pick_long', 'centroid_pick_lat'])
centroid_dropoff = pd.DataFrame(k_means.cluster_centers_, columns = ['centroid_drop_long', 'centroid_drop_lat'])
centroid_pickups['label_pick'] = centroid_pickups.index
centroid_dropoff['label_drop'] = centroid_dropoff.index
train_cl = pd.merge(train_cl, centroid_pickups, how='left', on=['label_pick'])
train_cl = pd.merge(train_cl, centroid_dropoff, how='left', on=['label_drop'])
train_cl.head(5)
```

Out[99]:

	id	vendor_id	pickup_datetime	dropoff_datetime	passenger_count	pickup_longitu
0	id2875421	2	2016-03-14 17:24:55	2016-03-14 17:32:30	1	-73.9821
1	id2377394	1	2016-06-12 00:43:35	2016-06-12 00:54:38	1	-73.9804
2	id3858529	2	2016-01-19 11:35:24	2016-01-19 12:10:48	1	-73.9790
3	id3504673	2	2016-04-06 19:32:31	2016-04-06 19:39:40	1	-74.0100
4	id2181028	2	2016-03-26 13:30:55	2016-03-26 13:38:10	1	-73.9730

5 rows × 28 columns

In [102]:

```
train_cl.loc[:,'hvsine_pick_cent_p'] = haversine_(train_cl['pickup_latitude'].values, train_cl['pickup_longitude'].values, train_cl['centroid_pick_lat'].values, train_cl['centroid_pick_long'].values)
train_cl.loc[:,'hvsine_drop_cent_d'] = haversine_(train_cl['dropoff_latitude'].values, train_cl['dropoff_longitude'].values, train_cl['centroid_drop_lat'].values, train_cl['centroid_drop_long'].values)
train_cl.loc[:,'hvsine_cent_p_cent_d'] = haversine_(train_cl['centroid_pick_lat'].values, train_cl['centroid_pick_long'].values, train_cl['centroid_drop_lat'].values, train_cl['centroid_drop_long'].values)
train_cl.loc[:,'manhtn_pick_cent_p'] = manhattan_distance_pd(train_cl['pickup_latitude'].values, train_cl['pickup_longitude'].values, train_cl['centroid_pick_lat'].values, train_cl['centroid_pick_long'].values)
train_cl.loc[:,'manhtn_drop_cent_d'] = manhattan_distance_pd(train_cl['dropoff_latitude'].values, train_cl['dropoff_longitude'].values, train_cl['centroid_drop_lat'].values, train_cl['centroid_drop_long'].values)
train_cl.loc[:,'manhtn_cent_p_cent_d'] = manhattan_distance_pd(train_cl['centroid_pick_lat'].values, train_cl['centroid_pick_long'].values, train_cl['centroid_drop_lat'].values, train_cl['centroid_drop_long'].values)

train_cl.loc[:,'bearing_pick_cent_p'] = bearing_array(train_cl['pickup_latitude'].values, train_cl['pickup_longitude'].values, train_cl['centroid_pick_lat'].values, train_cl['centroid_pick_long'].values)
train_cl.loc[:,'bearing_drop_cent_p'] = bearing_array(train_cl['dropoff_latitude'].values, train_cl['dropoff_longitude'].values, train_cl['centroid_drop_lat'].values, train_cl['centroid_drop_long'].values)
train_cl.loc[:,'bearing_cent_p_cent_d'] = bearing_array(train_cl['centroid_pick_lat'].values, train_cl['centroid_pick_long'].values, train_cl['centroid_drop_lat'].values, train_cl['centroid_drop_long'].values)
train_cl['speed_hvsn'] = train_cl.hvsine_pick_drop/train_cl.total_travel_time
train_cl['speed_manhtn'] = train_cl.manhtn_pick_drop/train_cl.total_travel_time
```

In [103]:

```
train_cl.head()
```

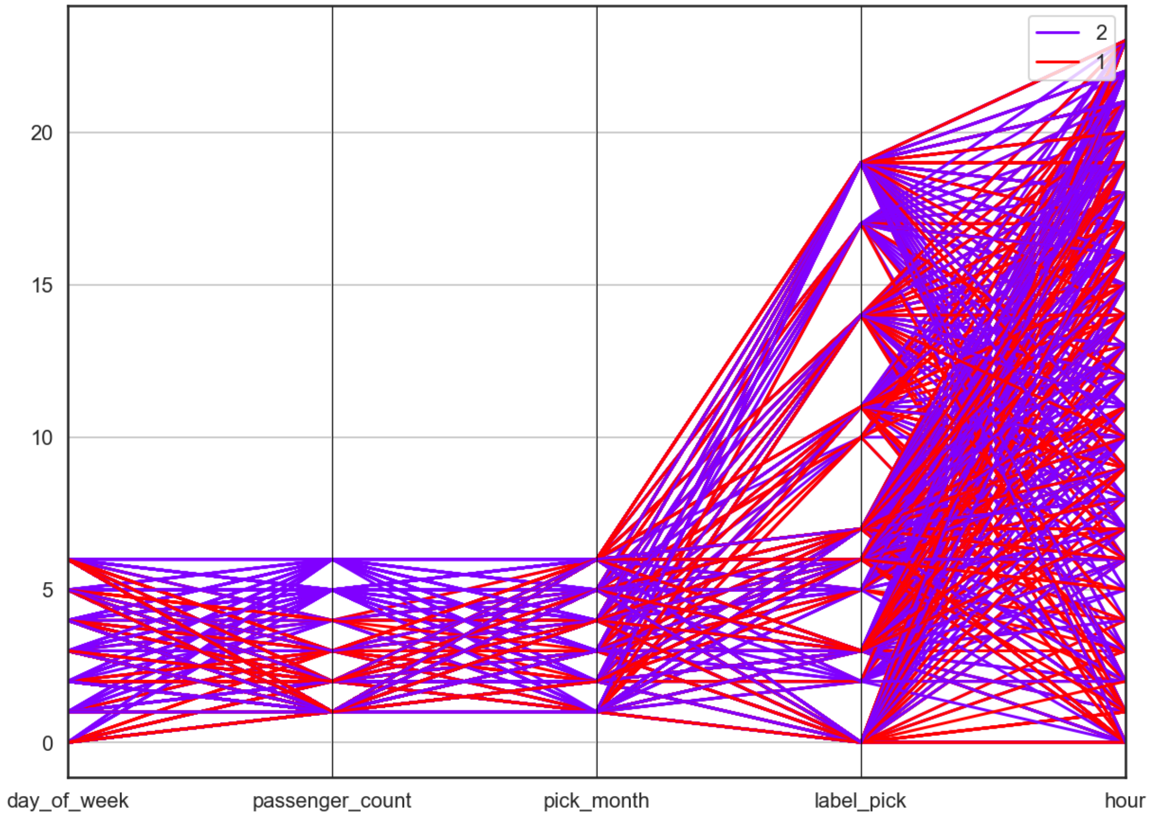
Out[103]:

	id	vendor_id	pickup_datetime	dropoff_datetime	passenger_count	pickup_longitu
0	id2875421	2	2016-03-14 17:24:55	2016-03-14 17:32:30	1	-73.9821
1	id2377394	1	2016-06-12 00:43:35	2016-06-12 00:54:38	1	-73.9804
2	id3858529	2	2016-01-19 11:35:24	2016-01-19 12:10:48	1	-73.9790
3	id3504673	2	2016-04-06 19:32:31	2016-04-06 19:39:40	1	-74.0100
4	id2181028	2	2016-03-26 13:30:55	2016-03-26 13:38:10	1	-73.9730

5 rows × 39 columns

In [106]:

```
from pandas.plotting import parallel_coordinates
plt.figure(figsize=(20, 15))
parallel_coordinates(train_data.sample(1200)[['vendor_id','day_of_week', 'passenger_count', 'pick_month','label_pick', 'hour']], 'vendor_id', colormap='rainbow')
plt.show()
```



In [107]:

```
test_df = pd.read_csv('test.csv')
test_fr = pd.read_csv('fastest_routes_test.csv')
test_fr_new = test_fr[['id', 'total_distance', 'total_travel_time', 'number_of_steps']]
test_df = pd.merge(test_df, test_fr_new, on = 'id', how = 'left')
test_df.head()
```

Out[107]:

	id	vendor_id	pickup_datetime	passenger_count	pickup_longitude	pickup_latitude
0	id3004672	1	2016-06-30 23:59:58	1	-73.988129	40.73202
1	id3505355	1	2016-06-30 23:59:53	1	-73.964203	40.67999
2	id1217141	1	2016-06-30 23:59:47	1	-73.997437	40.73758
3	id2150126	2	2016-06-30 23:59:41	1	-73.956070	40.77190
4	id1598245	1	2016-06-30 23:59:33	1	-73.970215	40.76147

In [108]:

```
start = time.time()
test_data = test_df.copy()
test_data['pickup_datetime'] = pd.to_datetime(test_data.pickup_datetime)
test_data.loc[:, 'pick_month'] = test_data['pickup_datetime'].dt.month
test_data.loc[:, 'hour'] = test_data['pickup_datetime'].dt.hour
test_data.loc[:, 'week_of_year'] = test_data['pickup_datetime'].dt.weekofyear
test_data.loc[:, 'day_of_year'] = test_data['pickup_datetime'].dt.dayofyear
test_data.loc[:, 'day_of_week'] = test_data['pickup_datetime'].dt.dayofweek
end = time.time()
print("Time taken by above cell is {}.".format(end-start))
```

```
Time taken by above cell is 1.261012077331543.
```

In [109]:

```
strat = time.time()
test_data.loc[:,'hvsine_pick_drop'] = haversine_(test_data['pickup_latitude'].values, test_data['pickup_longitude'].values, test_data['dropoff_latitude'].values, test_data['dropoff_longitude'].values)
test_data.loc[:,'manhtn_pick_drop'] = manhattan_distance_pd(test_data['pickup_latitude'].values, test_data['pickup_longitude'].values, test_data['dropoff_latitude'].values, test_data['dropoff_longitude'].values)
test_data.loc[:,'bearing'] = bearing_array(test_data['pickup_latitude'].values, test_data['pickup_longitude'].values, test_data['dropoff_latitude'].values, test_data['dropoff_longitude'].values)
end = time.time()
print("Time taken by above cell is {}.".format(end-strat))
```

```
Time taken by above cell is 0.5911810398101807.
```

In [110]:

```
start = time.time()
test_data['label_pick'] = k_means.predict(test_data[['pickup_longitude','pickup_latitude']])
test_data['label_drop'] = k_means.predict(test_data[['dropoff_longitude','dropoff_latitude']])
test_cl = pd.merge(test_data, centroid_pickups, how='left', on=['label_pick'])
test_cl = pd.merge(test_cl, centroid_dropoff, how='left', on=['label_drop'])
#test_cl.head()
end = time.time()
print("Time Taken by above cell is {}.".format(end-start))
```

```
Time Taken by above cell is 1.2631583213806152.
```

In [111]:

```
start = time.time()
test_cl.loc[:,'hvsine_pick_cent_p'] = haversine_(test_cl['pickup_latitude'].values, test_cl['pickup_longitude'].values, test_cl['centroid_pick_lat'].values, test_cl['centroid_pick_long'].values)
test_cl.loc[:,'hvsine_drop_cent_d'] = haversine_(test_cl['dropoff_latitude'].values, test_cl['dropoff_longitude'].values, test_cl['centroid_drop_lat'].values, test_cl['centroid_drop_long'].values)
test_cl.loc[:,'hvsine_cent_p_cent_d'] = haversine_(test_cl['centroid_pick_lat'].values, test_cl['centroid_pick_long'].values, test_cl['centroid_drop_lat'].values, test_cl['centroid_drop_long'].values)
test_cl.loc[:,'manhtn_pick_cent_p'] = manhattan_distance_pd(test_cl['pickup_latitude'].values, test_cl['pickup_longitude'].values, test_cl['centroid_pick_lat'].values, test_cl['centroid_pick_long'].values)
test_cl.loc[:,'manhtn_drop_cent_d'] = manhattan_distance_pd(test_cl['dropoff_latitude'].values, test_cl['dropoff_longitude'].values, test_cl['centroid_drop_lat'].values, test_cl['centroid_drop_long'].values)
test_cl.loc[:,'manhtn_cent_p_cent_d'] = manhattan_distance_pd(test_cl['centroid_pick_lat'].values, test_cl['centroid_pick_long'].values, test_cl['centroid_drop_lat'].values, test_cl['centroid_drop_long'].values)

test_cl.loc[:,'bearing_pick_cent_p'] = bearing_array(test_cl['pickup_latitude'].values, test_cl['pickup_longitude'].values, test_cl['centroid_pick_lat'].values, test_cl['centroid_pick_long'].values)
test_cl.loc[:,'bearing_drop_cent_p'] = bearing_array(test_cl['dropoff_latitude'].values, test_cl['dropoff_longitude'].values, test_cl['centroid_drop_lat'].values, test_cl['centroid_drop_long'].values)
test_cl.loc[:,'bearing_cent_p_cent_d'] = bearing_array(test_cl['centroid_pick_lat'].values, test_cl['centroid_pick_long'].values, test_cl['centroid_drop_lat'].values, test_cl['centroid_drop_long'].values)
test_cl['speed_hvsn'] = test_cl.hvsine_pick_drop/test_cl.total_travel_time
test_cl['speed_manhtn'] = test_cl.manhtn_pick_drop/test_cl.total_travel_time
end = time.time()
print("Time Taken by above cell is {}.".format(end-start))
```

```
Time Taken by above cell is 1.9373815059661865.
```

In [112]:

```
test_cl.head(5)
```

Out[112]:

	id	vendor_id	pickup_datetime	passenger_count	pickup_longitude	pickup_latitude
0	id3004672	1	2016-06-30 23:59:58	1	-73.988129	40.732029
1	id3505355	1	2016-06-30 23:59:53	1	-73.964203	40.679993
2	id1217141	1	2016-06-30 23:59:47	1	-73.997437	40.737583
3	id2150126	2	2016-06-30 23:59:41	1	-73.956070	40.771900
4	id1598245	1	2016-06-30 23:59:33	1	-73.970215	40.761475

5 rows × 37 columns

In [113]:

```
train_cl.head()
```

Out[113]:

	id	vendor_id	pickup_datetime	dropoff_datetime	passenger_count	pickup_longitu
0	id2875421	2	2016-03-14 17:24:55	2016-03-14 17:32:30	1	-73.9821
1	id2377394	1	2016-06-12 00:43:35	2016-06-12 00:54:38	1	-73.9804
2	id3858529	2	2016-01-19 11:35:24	2016-01-19 12:10:48	1	-73.9790
3	id3504673	2	2016-04-06 19:32:31	2016-04-06 19:39:40	1	-74.0100
4	id2181028	2	2016-03-26 13:30:55	2016-03-26 13:38:10	1	-73.9730

5 rows × 39 columns

In [117]:

```
import xgboost as xgb
from sklearn.model_selection import train_test_split
from sklearn.decomposition import PCA
from sklearn.cluster import MiniBatchKMeans
import warnings
```

In [122]:

```
train = train_cl
test = test_cl
start = time.time()
coords = np.vstack((train[['pickup_latitude', 'pickup_longitude']].values,
                    train[['dropoff_latitude', 'dropoff_longitude']].values,
                    test[['pickup_latitude', 'pickup_longitude']].values,
                    test[['dropoff_latitude', 'dropoff_longitude']].values))

pca = PCA().fit(coords)
train['pickup_pca0'] = pca.transform(train[['pickup_latitude', 'pickup_longitude']])[:, 0]
train['pickup_pca1'] = pca.transform(train[['pickup_latitude', 'pickup_longitude']])[:, 1]
train['dropoff_pca0'] = pca.transform(train[['dropoff_latitude', 'dropoff_longitude']])[:, 0]
train['dropoff_pca1'] = pca.transform(train[['dropoff_latitude', 'dropoff_longitude']])[:, 1]
test['pickup_pca0'] = pca.transform(test[['pickup_latitude', 'pickup_longitude']])[:, 0]
test['pickup_pca1'] = pca.transform(test[['pickup_latitude', 'pickup_longitude']])[:, 1]
test['dropoff_pca0'] = pca.transform(test[['dropoff_latitude', 'dropoff_longitude']])[:, 0]
test['dropoff_pca1'] = pca.transform(test[['dropoff_latitude', 'dropoff_longitude']])[:, 1]
end = time.time()
print("Time Taken by above cell is {}.".format(end - start))
```

```
Time Taken by above cell is 1.8993637561798096.
```

In [123]:

```
train['store_and_fwd_flag_int'] = np.where(train['store_and_fwd_flag']=='N', 0, 1)
test['store_and_fwd_flag_int'] = np.where(test['store_and_fwd_flag']=='N', 0, 1)
```

In [125]:

```
feature_names = list(train.columns)
print("Difference of features in train and test are {}".format(np.setdiff1d(train.columns, test.
columns)))
print("")
do_not_use_for_training = ['pick_date','id', 'pickup_datetime', 'dropoff_datetime', 'trip_durati
on', 'store_and_fwd_flag']
feature_names = [f for f in train.columns if f not in do_not_use_for_training]
print("We will be using following features for training {}.".format(feature_names))
print("")
print("Total number of features are {}.".format(len(feature_names)))
```

```
Difference of features in train and test are ['dropoff_datetime' 'trip_duration']

We will be using following features for training ['vendor_id', 'passenger_count',
'pickup_longitude', 'pickup_latitude', 'dropoff_longitude', 'dropoff_latitude', 't
otal_distance', 'total_travel_time', 'number_of_steps', 'pick_month', 'hour', 'wee
k_of_year', 'day_of_year', 'day_of_week', 'hvsine_pick_drop', 'manhtn_pick_drop',
'bearing', 'label_pick', 'label_drop', 'centroid_pick_long', 'centroid_pick_lat',
'centroid_drop_long', 'centroid_drop_lat', 'hvsine_pick_cent_p', 'hvsine_drop_cent
_d', 'hvsine_cent_p_cent_d', 'manhtn_pick_cent_p', 'manhtn_drop_cent_d', 'manhtn_c
ent_p_cent_d', 'bearing_pick_cent_p', 'bearing_drop_cent_p', 'bearing_cent_p_cent_
d', 'speed_hvsn', 'speed_manhtn', 'pickup_pca0', 'pickup_pca1', 'dropoff_pca0', 'd
ropoff_pca1', 'store_and_fwd_flag_int'].

Total number of features are 39.
```

In [126]:

```
y = np.log(train['trip_duration'].values + 1)
```

In [133]:

```
start = time.time()
Xtr, Xv, ytr, yv = train_test_split(train[feature_names].values, y, test_size=0.2, random_state=
1987)
dtrain = xgb.DMatrix(Xtr, label=ytr)
dvalid = xgb.DMatrix(Xv, label=yv)
dtest = xgb.DMatrix(test[feature_names].values)
watchlist = [(dtrain, 'train'), (dvalid, 'valid')]

xgb_pars = {'min_child_weight': 50, 'eta': 0.3, 'colsample_bytree': 0.3, 'max_depth': 10,
            'subsample': 0.8, 'lambda': 1., 'nthread': -1, 'booster' : 'gbtree', 'silent': 1,
            'eval_metric': 'rmse', 'objective': 'reg:linear'}

# You could try to train with more epoch
model = xgb.train(xgb_pars, dtrain, 15, watchlist, early_stopping_rounds=2,
                  maximize=False, verbose_eval=1)
end = time.time()
print("Time taken by above cell is {}.".format(end - start))
print('Modeling RMSLE %.5f' % model.best_score)
```

```
[0]	train-rmse:4.22685	valid-rmse:4.22797
Multiple eval metrics have been passed: 'valid-rmse' will be used for early stoppi
ng.

Will train until valid-rmse hasn't improved in 2 rounds.
[1]	train-rmse:2.97552	valid-rmse:2.97693
[2]	train-rmse:2.10895	valid-rmse:2.11080
[3]	train-rmse:1.50695	valid-rmse:1.50960
[4]	train-rmse:1.09674	valid-rmse:1.10052
[5]	train-rmse:0.82173	valid-rmse:0.82685
[6]	train-rmse:0.64469	valid-rmse:0.65151
[7]	train-rmse:0.53626	valid-rmse:0.54475
[8]	train-rmse:0.47339	valid-rmse:0.48341
[9]	train-rmse:0.43795	valid-rmse:0.44902
[10]	train-rmse:0.41818	valid-rmse:0.43022
[11]	train-rmse:0.40753	valid-rmse:0.42044
[12]	train-rmse:0.40158	valid-rmse:0.41509
[13]	train-rmse:0.39513	valid-rmse:0.40938
[14]	train-rmse:0.39229	valid-rmse:0.40704
Time taken by above cell is 43.50342893600464.
Modeling RMSLE 0.40704
```

In [134]:

```
weather = pd.read_csv(’weather_data_nyc_centralpark_2016.csv’)
weather.head()
```

Out[134]:

	date	maximum temerature	minimum temperature	average temperature	precipitation	snow fall	snow depth
0	1-1-2016	42	34	38.0	0.00	0.0	0
1	2-1-2016	40	32	36.0	0.00	0.0	0
2	3-1-2016	45	35	40.0	0.00	0.0	0
3	4-1-2016	36	14	25.0	0.00	0.0	0
4	5-1-2016	29	11	20.0	0.00	0.0	0

In [139]:

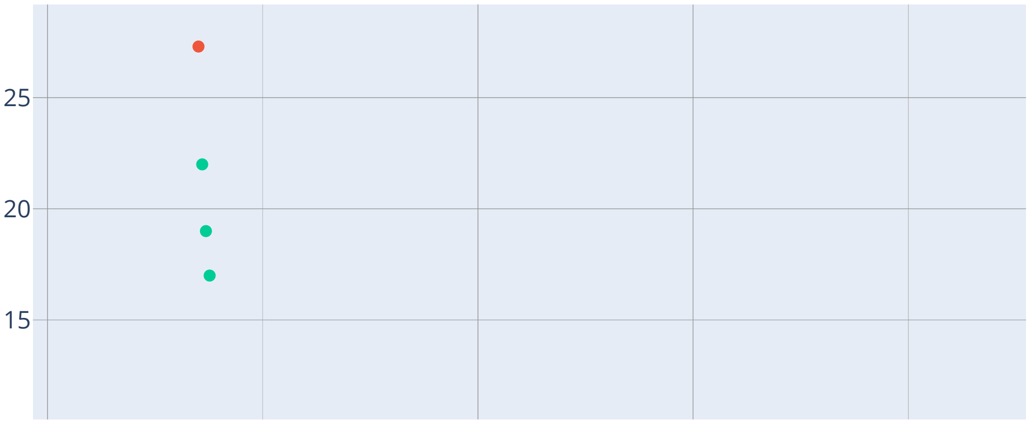
```
weather.date = pd.to_datetime(weather.date)
weather[’day_of_year’]= weather.date.dt.dayofyear
```

In [142]:

```
import matplotlib.pyplot as plt
%matplotlib inline
weather[’precipitation’].unique()
weather[’precipitation’] = np.where(weather[’precipitation’]==’T’, ’0.00’,weather[’precipitatio
n’])
weather[’precipitation’] = list(map(float, weather[’precipitation’]))
weather[’snow fall’] = np.where(weather[’snow fall’]==’T’, ’0.00’,weather[’snow fall’])
weather[’snow fall’] = list(map(float, weather[’snow fall’]))
weather[’snow depth’] = np.where(weather[’snow depth’]==’T’, ’0.00’,weather[’snow depth’])
weather[’snow depth’] = list(map(float, weather[’snow depth’]))
```

In [144]:

```
import chart_studio.plotly as py
import plotly.graph_objs as go
import plotly
random_x = weather['date'].values
random_y0 = weather['precipitation']
random_y1 = weather['snow fall']
random_y2 = weather['snow depth']

# Create traces
trace0 = go.Scatter(
    x = random_x,
    y = random_y0,
    mode = 'markers',
    name = 'precipitation'
)
trace1 = go.Scatter(
    x = random_x,
    y = random_y1,
    mode = 'markers',
    name = 'snow fall'
)
trace2 = go.Scatter(
    x = random_x,
    y = random_y2,
    mode = 'markers',
    name = 'snow depth'
)

data = [trace0, trace1, trace2]
plotly.offline.iplot(data, filename='scatter-mode')
```



In [147]:

```
def freq_turn(step_dir):
    """function to create dummy for turn type"""
    from collections import Counter
    step_dir_new = step_dir.split("|")
    a_list = Counter(step_dir_new).most_common()
    path = {}
    for i in range(len(a_list)):
        path.update({a_list[i]})
    a = 0
    b = 0
    c = 0
    if 'straight' in (path.keys()):
        a = path['straight']
        #print(a)
    if 'left' in (path.keys()):
        b = path['left']
        #print(b)
    if 'right' in (path.keys()):
        c = path['right']
        #print(c)
    return a,b,c
```

In [148]:

```
start = time.time()
train_fr['straight']= 0
train_fr['left'] =0
train_fr['right'] = 0
train_fr['straight'], train_fr['left'], train_fr['right'] = zip(*train_fr['step_direction'].map(
freq_turn))
end = time.time()
print("Time Taken by above cell is {}.".format(end - start))
```

```
Time Taken by above cell is 15.908488512039185.
```

In [149]:

```
train_fr.head()
```

Out[149]:

	id	starting_street	end_street	total_distance	total_travel_time	number_of_steps
0	id2875421	Columbus Circle	East 65th Street	2009.1	164.9	5
1	id2377394	2nd Avenue	Washington Square West	2513.2	332.0	6
2	id3504673	Greenwich Street	Broadway	1779.4	235.8	4
3	id2181028	Broadway	West 81st Street	1614.9	140.1	5
4	id0801584	Lexington Avenue	West 31st Street	1393.5	189.4	5

In [150]:

```
train_fr_new = train_fr[['id','straight','left','right']]
train = pd.merge(train, train_fr_new, on = 'id', how = 'left')
#train = pd.merge(train, weather, on= 'date', how = 'left')
print(len(train.columns))
#train.columns
```

```
47
```

In [151]:

```
train['pickup_datetime'] = pd.to_datetime(train['pickup_datetime'])
train['date'] = train['pickup_datetime'].dt.date
train.head()
```

Out[151]:

	id	vendor_id	pickup_datetime	dropoff_datetime	passenger_count	pickup_longitu
0	id2875421	2	2016-03-14 17:24:55	2016-03-14 17:32:30	1	-73.9821
1	id2377394	1	2016-06-12 00:43:35	2016-06-12 00:54:38	1	-73.9804
2	id3858529	2	2016-01-19 11:35:24	2016-01-19 12:10:48	1	-73.9790
3	id3504673	2	2016-04-06 19:32:31	2016-04-06 19:39:40	1	-74.0100
4	id2181028	2	2016-03-26 13:30:55	2016-03-26 13:38:10	1	-73.9730

5 rows × 48 columns

In [152]:

```
train['date'] = pd.to_datetime(train['date'])
train = pd.merge(train, weather[['date','minimum temperature', 'precipitation', 'snow fall', 'snow depth']], on= 'date', how = 'left')
train.shape[0]
```

Out[152]:

```
1458644
```

In [153]:

```
train.loc[:,'hvsine_pick_cent_d'] = haversine_(train['pickup_latitude'].values, train['pickup_longitude'].values, train['centroid_drop_lat'].values, train['centroid_drop_long'].values)
train.loc[:,'hvsine_drop_cent_p'] = haversine_(train['dropoff_latitude'].values, train['dropoff_longitude'].values, train['centroid_pick_lat'].values, train['centroid_pick_long'].values)

test.loc[:,'hvsine_pick_cent_d'] = haversine_(test['pickup_latitude'].values, test['pickup_longitude'].values, test['centroid_drop_lat'].values, test['centroid_drop_long'].values)
test.loc[:,'hvsine_drop_cent_p'] = haversine_(test['dropoff_latitude'].values, test['dropoff_longitude'].values, test['centroid_pick_lat'].values, test['centroid_pick_long'].values)

print("shape of train_features is {}.".format(len(train.columns)))
```

```
shape of train_features is 54.
```

In [154]:

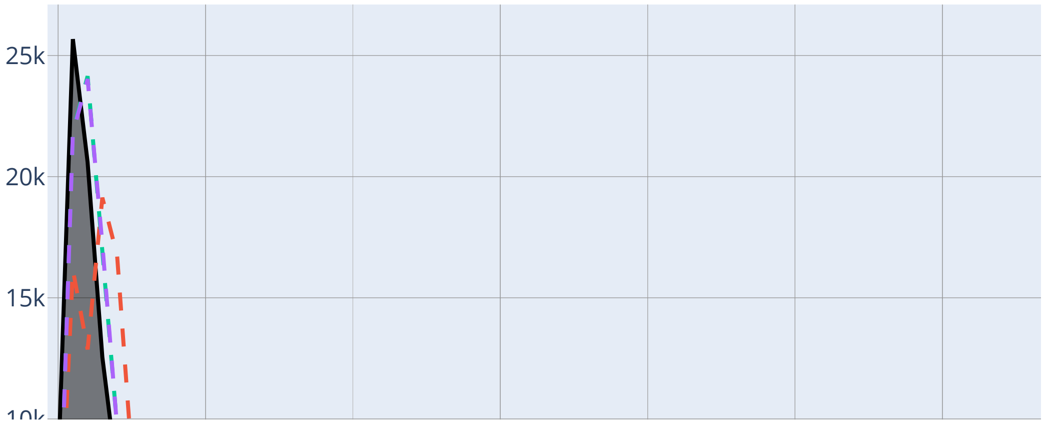
```
start = time.time()
temp = train[['hvsine_drop_cent_p', 'hvsine_pick_cent_d', 'hvsine_drop_cent_d', 'hvsine_pick_cent_p', 'hvsine_pick_drop', 'hvsine_cent_p_cent_d', 'total_distance']]
temp.total_distance.dropna(inplace = True)
print("total number of Nulls {}.".format(temp.total_distance.isnull().sum()))
# Lets take distance of pick---cent_p---cent_d---drop as distance_pick_cp_cd_drop
# Lets take distance of pick---cent_d---drop as distance_pick_cd_drop
# Lets take distance of pick---cent_p---drop as distance_pick_cp_drop
# Lets take distance of pick--drop as total_distance
temp['distance_pick_cp_cd_drop'] = temp['hvsine_pick_cent_p'] + temp['hvsine_cent_p_cent_d'] + temp['hvsine_drop_cent_d']
temp['distance_pick_cd_drop'] = temp['hvsine_pick_cent_d'] + temp['hvsine_drop_cent_d']
temp['distance_pick_cp_drop'] = temp['hvsine_pick_cent_p'] + temp['hvsine_drop_cent_p']
temp['total_distance'] = np.floor(temp['total_distance']/1000)
temp['distance_pick_cp_drop'] = np.floor(temp['distance_pick_cp_drop'])
temp['distance_pick_cd_drop'] = np.floor(temp['distance_pick_cd_drop'])
temp['distance_pick_cp_cd_drop'] = np.floor(temp['distance_pick_cp_cd_drop'])
#temp.head()
temp1 = temp.copy()
temp = temp1.sample(100000)
aggregation = {'distance_pick_cp_cd_drop':'count', 'distance_pick_cd_drop':'count', 'distance_pick_cp_drop':'count', 'total_distance':'count'}
temp2 = pd.DataFrame(temp.groupby('total_distance').agg(aggregation))
X_plot = np.linspace(0, temp.total_distance.max(), temp.shape[0])
temp2.rename(columns={'total_distance':'count'}, inplace = True)
temp2.reset_index(inplace = True)
temp2.total_distance = map(int, temp2.total_distance)
temp = temp.sample(100000)
X_plot = temp.total_distance.unique()
a = np.histogram(temp[['total_distance']].values, range(0,95))
N = temp.shape[0]
data = []
trace1 = go.Scatter(x=np.histogram(temp[['total_distance']].values, range(0,95))[1], y=np.histogram(temp[['total_distance']].values, range(0,95))[0],
                    mode='lines', fill='tozeroy',
                    line=dict(color='black', width=2),
                    name='Total_distance_OSRM')
data.append(trace1)

for kernel in ['distance_pick_cp_cd_drop', 'distance_pick_cd_drop', 'distance_pick_cp_drop']:
    trace2 = go.Scatter(x=np.histogram(temp[['total_distance']].values, range(0,95))[1], y=np.histogram(temp[[kernel]].values, range(0,95))[0],
                        mode='lines',
                        line=dict(width=2, dash='dash'),
                        name=kernel)
    data.append(trace2)

layout=go.Layout(annotations=[dict(x=6, y=0.38, showarrow=False,
                                   text="N={0} points".format(N)),
                                   ],
                 xaxis=dict(zeroline=False), hovermode='closest')
fig = go.Figure(data=data, layout=layout)
```

```
C:\Users\Lin\Anaconda3\lib\site-packages\pandas\core\series.py:4784: SettingWithCo
pyWarning:


A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/
user_guide/indexing.html#returning-a-view-versus-a-copy

C:\Users\Lin\Anaconda3\lib\site-packages\ipykernel_launcher.py:9: SettingWithCopyW
arning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/
user_guide/indexing.html#returning-a-view-versus-a-copy

C:\Users\Lin\Anaconda3\lib\site-packages\ipykernel_launcher.py:10: SettingWithCopy
Warning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/
user_guide/indexing.html#returning-a-view-versus-a-copy

C:\Users\Lin\Anaconda3\lib\site-packages\ipykernel_launcher.py:11: SettingWithCopy
Warning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/
user_guide/indexing.html#returning-a-view-versus-a-copy

C:\Users\Lin\Anaconda3\lib\site-packages\ipykernel_launcher.py:12: SettingWithCopy
Warning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/
user_guide/indexing.html#returning-a-view-versus-a-copy

C:\Users\Lin\Anaconda3\lib\site-packages\ipykernel_launcher.py:13: SettingWithCopy
Warning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/
user_guide/indexing.html#returning-a-view-versus-a-copy
```

```
total number of Nulls 0.

C:\Users\Lin\Anaconda3\lib\site-packages\ipykernel_launcher.py:14: SettingWithCopy
Warning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/
user_guide/indexing.html#returning-a-view-versus-a-copy

C:\Users\Lin\Anaconda3\lib\site-packages\ipykernel_launcher.py:15: SettingWithCopy
Warning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/
user_guide/indexing.html#returning-a-view-versus-a-copy
```

In [155]:

```
plotly.offline.iplot(fig)  # last 150+ lines of code just to make a beautiful histogram ;) haha
```

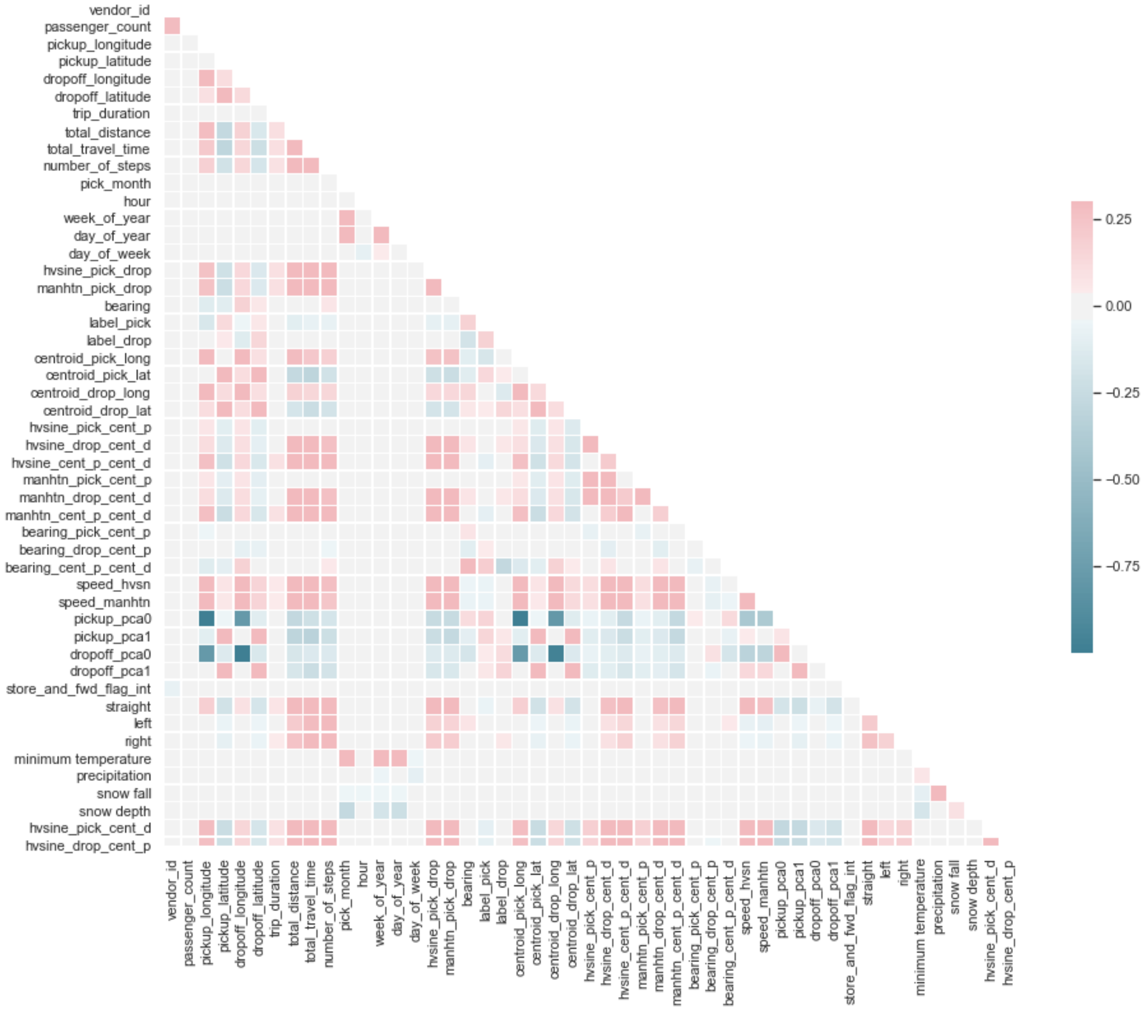


In [158]:

```
sns.set(style="white")

# Generate a large random dataset
temp3 = train.copy()

# Compute the correlation matrix
corr = temp3.corr()

# Generate a mask for the upper triangle
mask = np.zeros_like(corr, dtype=np.bool)
mask[np.triu_indices_from(mask)] = True

# Set up the matplotlib figure
f, ax = plt.subplots(figsize=(15, 13))

# Generate a custom diverging colormap
cmap = sns.diverging_palette(220, 10, as_cmap=True)

# Draw the heatmap with the mask and correct aspect ratio
sns.heatmap(corr, mask=mask, cmap=cmap, vmax=.3, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5})
```

Out[158]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x18e1dca43c8>
```



In [160]:

```
start = time.time()
test_fr['straight']= 0
test_fr['left'] =0
test_fr['right'] = 0
test_fr['straight'], test_fr['left'], test_fr['right'] = zip(*test_fr['step_direction'].map(freq
_turn))
end = time.time()
print("Time Taken by above cell is {}.".format(end - start))
#test_fr.head()
```

```
Time Taken by above cell is 7.569939374923706.
```

In [161]:

```
test_fr_new = test_fr[['id','straight','left','right']]
test = pd.merge(test, test_fr_new, on = 'id', how = 'left')
print(len(test.columns))
#test.columns
```

```
47
```

In [162]:

```
test['pickup_datetime'] = pd.to_datetime(test['pickup_datetime'])
test['date'] = test['pickup_datetime'].dt.date
test['date'] = pd.to_datetime(test['date'])
```

In [163]:

```
test= pd.merge(test, weather[['date','minimum temperature', 'precipitation', 'snow fall', 'snow depth']], on= 'date', how = 'left')
feature_names = list(train.columns)
print("Difference of features in train and test are {}".format(np.setdiff1d(train.columns, test.columns)))
print("")
do_not_use_for_training = ['pick_date','id', 'pickup_datetime', 'dropoff_datetime', 'trip_duration', 'store_and_fwd_flag', 'date']
feature_names = [f for f in train.columns if f not in do_not_use_for_training]
print("We will be using following features for training {}.".format(feature_names))
print("")
print("Total number of features are {}.".format(len(feature_names)))
```

```
Difference of features in train and test are ['dropoff_datetime' 'trip_duration']

We will be using following features for training ['vendor_id', 'passenger_count', 'pickup_longitude', 'pickup_latitude', 'dropoff_longitude', 'dropoff_latitude', 'total_distance', 'total_travel_time', 'number_of_steps', 'pick_month', 'hour', 'week_of_year', 'day_of_year', 'day_of_week', 'hvsine_pick_drop', 'manhtn_pick_drop', 'bearing', 'label_pick', 'label_drop', 'centroid_pick_long', 'centroid_pick_lat', 'centroid_drop_long', 'centroid_drop_lat', 'hvsine_pick_cent_p', 'hvsine_drop_cent_d', 'hvsine_cent_p_cent_d', 'manhtn_pick_cent_p', 'manhtn_drop_cent_d', 'manhtn_cent_p_cent_d', 'bearing_pick_cent_p', 'bearing_drop_cent_p', 'bearing_cent_p_cent_d', 'speed_hvsn', 'speed_manhtn', 'pickup_pca0', 'pickup_pca1', 'dropoff_pca0', 'dropoff_pca1', 'store_and_fwd_flag_int', 'straight', 'left', 'right', 'minimum temperature', 'precipitation', 'snow fall', 'snow depth', 'hvsine_pick_cent_d', 'hvsine_drop_cent_p'].

Total number of features are 48.
```

In [164]:

```
y = np.log(train['trip_duration'].values + 1)
```

In [170]:

```
Xtr, Xv, ytr, yv = train_test_split(train[feature_names].values, y, test_size=0.2, random_state=
1987)
dtrain = xgb.DMatrix(Xtr, label=ytr)
dvalid = xgb.DMatrix(Xv, label=yv)
dtest = xgb.DMatrix(test[feature_names].values)
watchlist = [(dtrain, 'train'), (dvalid, 'valid')]

start = time.time()
xgb_par = {'min_child_weight': 20, 'eta': 0.05, 'colsample_bytree': 0.5, 'max_depth': 30,
            'subsample': 0.9, 'lambda': 2.0, 'nthread': -1, 'booster' : 'gbtree', 'silent': 1,
            'eval_metric': 'rmse', 'objective': 'reg:linear'}

model_1 = xgb.train(xgb_par, dtrain, 50, watchlist, early_stopping_rounds=4, maximize=False, ve
rbose_eval=1)
print('Modeling RMSLE %.5f' % model.best_score)
end = time.time()
print("Time taken in training is {}.".format(end - start))
```

```
[0]	train-rmse:5.72051	valid-rmse:5.72142
Multiple eval metrics have been passed: 'valid-rmse' will be used for early stoppi
ng.

Will train until valid-rmse hasn't improved in 4 rounds.
[1]	train-rmse:5.43629	valid-rmse:5.43725
[2]	train-rmse:5.16653	valid-rmse:5.16753
[3]	train-rmse:4.91031	valid-rmse:4.91138
[4]	train-rmse:4.66679	valid-rmse:4.66790
[5]	train-rmse:4.43572	valid-rmse:4.43689
[6]	train-rmse:4.21611	valid-rmse:4.21734
[7]	train-rmse:4.00761	valid-rmse:4.00888
[8]	train-rmse:3.80990	valid-rmse:3.81124
[9]	train-rmse:3.62190	valid-rmse:3.62330
[10]	train-rmse:3.44353	valid-rmse:3.44502
[11]	train-rmse:3.27419	valid-rmse:3.27579
[12]	train-rmse:3.11336	valid-rmse:3.11506
[13]	train-rmse:2.96064	valid-rmse:2.96244
[14]	train-rmse:2.81557	valid-rmse:2.81747
[15]	train-rmse:2.67811	valid-rmse:2.68013
[16]	train-rmse:2.54770	valid-rmse:2.54980
[17]	train-rmse:2.42387	valid-rmse:2.42611
[18]	train-rmse:2.30640	valid-rmse:2.30879
[19]	train-rmse:2.19499	valid-rmse:2.19754
[20]	train-rmse:2.08916	valid-rmse:2.09188
[21]	train-rmse:1.98890	valid-rmse:1.99181
[22]	train-rmse:1.89385	valid-rmse:1.89702
[23]	train-rmse:1.80373	valid-rmse:1.80710
[24]	train-rmse:1.71812	valid-rmse:1.72172
[25]	train-rmse:1.63719	valid-rmse:1.64106
[26]	train-rmse:1.56045	valid-rmse:1.56460
[27]	train-rmse:1.48777	valid-rmse:1.49226
[28]	train-rmse:1.41867	valid-rmse:1.42353
[29]	train-rmse:1.35341	valid-rmse:1.35862
[30]	train-rmse:1.29143	valid-rmse:1.29707
[31]	train-rmse:1.23283	valid-rmse:1.23895
[32]	train-rmse:1.17727	valid-rmse:1.18393
[33]	train-rmse:1.12460	valid-rmse:1.13181
[34]	train-rmse:1.07497	valid-rmse:1.08270
[35]	train-rmse:1.02809	valid-rmse:1.03646
[36]	train-rmse:0.98392	valid-rmse:0.99300
[37]	train-rmse:0.94198	valid-rmse:0.95180
[38]	train-rmse:0.90248	valid-rmse:0.91312
[39]	train-rmse:0.86497	valid-rmse:0.87656
[40]	train-rmse:0.82967	valid-rmse:0.84223
[41]	train-rmse:0.79620	valid-rmse:0.80986
[42]	train-rmse:0.76471	valid-rmse:0.77951
[43]	train-rmse:0.73469	valid-rmse:0.75074
[44]	train-rmse:0.70642	valid-rmse:0.72383
[45]	train-rmse:0.68019	valid-rmse:0.69900
[46]	train-rmse:0.65527	valid-rmse:0.67556
[47]	train-rmse:0.63172	valid-rmse:0.65357
[48]	train-rmse:0.60953	valid-rmse:0.63306
[49]	train-rmse:0.58884	valid-rmse:0.61412
Modeling RMSLE 0.40704
Time taken in training is 502.9955184459686.
```

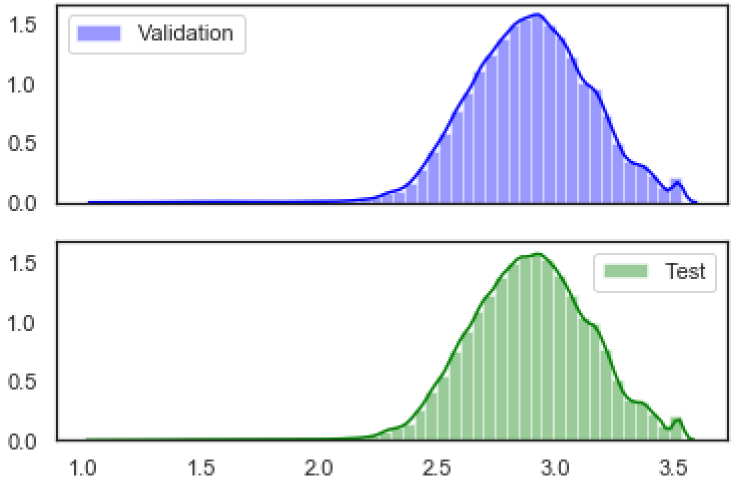
In [166]:

```
print('Modeling RMSLE %.5f' % model_1.best_score)
end = time.time()
print("Time taken in training is {}.".format(end - start))
start = time.time()
yvalid = model_1.predict(dvalid)
ytest = model_1.predict(dtest)
end = time.time()
print("Time taken in prediction is {}.".format(end - start))
```

```
Modeling RMSLE 3.62329
Time taken in training is 141.5664279460907.
Time taken in prediction is 0.22202324867248535.
```

In [167]:

```
# Lets check how the distribution of test and vaidation set looks like ...
start = time.time()
fig, ax = plt.subplots(nrows=2, sharex=True, sharey=True)
sns.distplot(yvalid, ax=ax[0], color='blue', label='Validation')
sns.distplot(ytest, ax=ax[1], color='green', label='Test')
ax[0].legend(loc=0)
ax[1].legend(loc=0)
plt.show()
end = time.time()
print("Time taken by above cell is {}.".format((end-start)))
```



```
Time taken by above cell is 1.3659632205963135.
```

In [168]:

```
start = time.time()
if test.shape[0] == ytest.shape[0]:
    print('Test shape OK.')
test['trip_duration'] = np.exp(ytest) - 1
test[['id', 'trip_duration']].to_csv('mahesh_xgb_submission.csv', index=False)
end = time.time()
print("Time taken in training is {}.".format(end - start))
```

```
Test shape OK.
Time taken in training is 1.6065764427185059.
```

In [173]:

```
train_data.trip_duration
```

Out[173]:

```
0           455
1           663
2          2124
3           429
4           435
           ...
1458639     778
1458640     655
1458641     764
1458642     373
1458643     198
Name: trip_duration, Length: 1458644, dtype: int64
```

In [174]:

```
ytest = model_1.predict(dtest)
```

In [175]:

```
test['trip_duration'] = np.exp(ytest) - 1
```

In [ ]:

```

```