NYC taxi trip duration

October 31, 2021

0.0.1 Problem statement

0.1 Predict the total ride of taxi trips in new York city

Predicting the duration of each taxi trip at the point when the trip starts helps them to plan their fleet in a much better manner

0.1.1 Load libraries

[1]:
```
%matplotlib inline
import numpy as np
import pandas as pd
from datetime import timedelta
import datetime as dt
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

0.1.2 Import dataset

[2]:
```
data=pd.read_csv("nyc_taxi_trip_duration.csv")
```

0.1.3 Data Exploration

[3]:
```
data.head()
```

[3]:

	id	vendor_id	pickup_datetime	dropoff_datetime	\
0	id1080784	2	2016-02-29 16:40:21	2016-02-29 16:47:01	
1	id0889885	1	2016-03-11 23:35:37	2016-03-11 23:53:57	
2	id0857912	2	2016-02-21 17:59:33	2016-02-21 18:26:48	
3	id3744273	2	2016-01-05 09:44:31	2016-01-05 10:03:32	
4	id0232939	1	2016-02-17 06:42:23	2016-02-17 06:56:31	

	passenger_count	pickup_longitude	pickup_latitude	dropoff_longitude	\
0	1	-73.953918	40.778873	-73.963875	
1	2	-73.988312	40.731743	-73.994751	
2	2	-73.997314	40.721458	-73.948029	
3	6	-73.961670	40.759720	-73.956779	

```
4                1        -74.017120        40.708469         -73.988182

   dropoff_latitude store_and_fwd_flag  trip_duration
0         40.771164                  N            400
1         40.694931                  N           1100
2         40.774918                  N           1635
3         40.780628                  N           1141
4         40.740631                  N            848
```

0.1.4 Feature details:

- id - a unique identifier for each trip
- vendor_id - a code indicating the provider associated with the trip record
- pickup_datetime - date and time when the meter was engaged
- dropoff_datetime - date and time when the meter was disengaged
- passenger_count - the number of passengers in the vehicle (driver entered value)
- pickup_longitude - the longitude where the meter was engaged
- pickup_latitude - the latitude where the meter was engaged
- dropoff_longitude - the longitude where the meter was disengaged
- dropoff_latitude - the latitude where the meter was disengaged
- store_and_fwd_flag - This flag indicates whether the trip record was held in vehicle memory before sending to the vendor because the vehicle did not have a connection to the server - Y=store and forward; N=not a store and forward trip.

0.1.5 Label details:

- trip_duration - duration of the trip in seconds

[4]:
```
data.shape
```

[4]:
```
(729322, 11)
```

[5]:
```
data.isnull().sum()
```

[5]:
```
id                  0
vendor_id           0
pickup_datetime     0
dropoff_datetime    0
passenger_count     0
pickup_longitude    0
pickup_latitude     0
dropoff_longitude   0
dropoff_latitude    0
store_and_fwd_flag  0
trip_duration       0
dtype: int64
```

0.1.6 Result:

There are no null values in the dataset

[6]:
```
data.dtypes
```

[6]:
```
id                    object
vendor_id              int64
pickup_datetime       object
dropoff_datetime      object
passenger_count        int64
pickup_longitude     float64
pickup_latitude      float64
dropoff_longitude    float64
dropoff_latitude     float64
store_and_fwd_flag    object
trip_duration          int64
dtype: object
```

[7]:
```
data['pickup_datetime'] = pd.to_datetime(data['pickup_datetime'])
data['dropoff_datetime'] = pd.to_datetime(data['dropoff_datetime'])
```

0.1.7 calculate and assign new columns with week_day, month, pickup hour

[8]:
```
data['weekday'] = data.pickup_datetime.dt.day_name()
data['month'] = data.pickup_datetime.dt.month
data['weekday_num'] = data.pickup_datetime.dt.weekday
data['pick_up_hour'] = data.pickup_datetime.dt.hour
```

[9]:
```
data.head()
```

[9]:

	id	vendor_id	pickup_datetime	dropoff_datetime	passenger_count	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	store_and_fwd_flag	trip_duration	weekday	month
0	id1080784	2	2016-02-29 16:40:21	2016-02-29 16:47:01	1	-73.953918	40.778873	-73.963875	40.771164	N	400	Monday	2
1	id0889885	1	2016-03-11 23:35:37	2016-03-11 23:53:57	2	-73.988312	40.731743	-73.994751	40.694931	N	1100	Friday	3
2	id0857912	2	2016-02-21 17:59:33	2016-02-21 18:26:48	2	-73.997314	40.721458	-73.948029	40.774918	N	1635	Sunday	2
3	id3744273	2	2016-01-05 09:44:31	2016-01-05 10:03:32	6	-73.961670	40.759720	-73.956779					
4	id0232939	1	2016-02-17 06:42:23	2016-02-17 06:56:31	1	-74.017120	40.708469	-73.988182					

```
3         40.780628                  N           1141    Tuesday       1
4         40.740631                  N            848  Wednesday       2

   weekday_num  pick_up_hour
0            0            16
1            4            23
2            6            17
3            1             9
4            2             6
```

[10]:
```
from haversine import haversine
```

0.1.8 calculate distance between pickup and dropoff coordinates using haversine formula

[11]:
```
def calc_distance(data):
    pickup = (data['pickup_latitude'], data['pickup_longitude'])
    drop = (data['dropoff_latitude'], data['dropoff_longitude'])
    return haversine(pickup, drop)
```

0.1.9 calculate distance and assign new column to dataframe

[12]:
```
data['Distance'] = data.apply(lambda x: calc_distance(x), axis = 1)
```

0.1.10 calculate speed in km/hr

[13]:
```
data['speed'] = (data.Distance/(data.trip_duration/3600))
```

[14]:
```
data.dtypes.reset_index()
```

[14]:
```
                index              0
0                  id         object
1           vendor_id          int64
2     pickup_datetime  datetime64[ns]
3    dropoff_datetime  datetime64[ns]
4     passenger_count          int64
5    pickup_longitude        float64
6     pickup_latitude        float64
7   dropoff_longitude        float64
8    dropoff_latitude        float64
9  store_and_fwd_flag         object
10      trip_duration          int64
11            weekday         object
12              month          int64
13        weekday_num          int64
14       pick_up_hour          int64
15           Distance        float64
```

```
16              speed         float64
```

0.1.11 Dummify all the categorical features like "store_and_fwd_flag, vendor_id, month, weekday_num, pickup_hour, passenger_count"

[15]:
```
dummy = pd.get_dummies(data.store_and_fwd_flag, prefix='flag')
dummy.drop(dummy.columns[0], axis=1, inplace=True) #avoid dummy trap
data = pd.concat([data,dummy], axis = 1)

dummy = pd.get_dummies(data.vendor_id, prefix='vendor_id')
dummy.drop(dummy.columns[0], axis=1, inplace=True)
data = pd.concat([data,dummy], axis = 1)

dummy = pd.get_dummies(data.month, prefix='month')
dummy.drop(dummy.columns[0], axis=1, inplace=True)
data = pd.concat([data,dummy], axis = 1)

dummy = pd.get_dummies(data.weekday_num, prefix='weekday_num')
dummy.drop(dummy.columns[0], axis=1, inplace=True)
data = pd.concat([data,dummy], axis = 1)

dummy = pd.get_dummies(data.pick_up_hour, prefix='pickup_hour')
dummy.drop(dummy.columns[0], axis=1, inplace=True)
data = pd.concat([data,dummy], axis = 1)

dummy = pd.get_dummies(data.passenger_count, prefix='passenger_count')
dummy.drop(dummy.columns[0], axis=1, inplace=True)
data = pd.concat([data,dummy], axis = 1)
```

[16]:
```
data.head()
```

[16]:

	id	vendor_id	pickup_datetime	dropoff_datetime	\
0	id1080784	2	2016-02-29 16:40:21	2016-02-29 16:47:01	
1	id0889885	1	2016-03-11 23:35:37	2016-03-11 23:53:57	
2	id0857912	2	2016-02-21 17:59:33	2016-02-21 18:26:48	
3	id3744273	2	2016-01-05 09:44:31	2016-01-05 10:03:32	
4	id0232939	1	2016-02-17 06:42:23	2016-02-17 06:56:31	

	passenger_count	pickup_longitude	pickup_latitude	dropoff_longitude	\
0	1	-73.953918	40.778873	-73.963875	
1	2	-73.988312	40.731743	-73.994751	
2	2	-73.997314	40.721458	-73.948029	
3	6	-73.961670	40.759720	-73.956779	
4	1	-74.017120	40.708469	-73.988182	

	dropoff_latitude	store_and_fwd_flag	…	pickup_hour_22	pickup_hour_23	\
0	40.771164	N	…	0	0	

```
1         40.694931                  N  …                0              1
2         40.774918                  N  …                0              0
3         40.780628                  N  …                0              0
4         40.740631                  N  …                0              0

   passenger_count_1  passenger_count_2  passenger_count_3  passenger_count_4  \
0                  1                  0                  0                  0
1                  0                  1                  0                  0
2                  0                  1                  0                  0
3                  0                  0                  0                  0
4                  1                  0                  0                  0

   passenger_count_5  passenger_count_6  passenger_count_7  passenger_count_9
0                  0                  0                  0                  0
1                  0                  0                  0                  0
2                  0                  0                  0                  0
3                  0                  1                  0                  0
4                  0                  0                  0                  0

[5 rows x 61 columns]
```

[17]:
```
data.columns
```

[17]:
```
Index(['id', 'vendor_id', 'pickup_datetime', 'dropoff_datetime',
       'passenger_count', 'pickup_longitude', 'pickup_latitude',
       'dropoff_longitude', 'dropoff_latitude', 'store_and_fwd_flag',
       'trip_duration', 'weekday', 'month', 'weekday_num', 'pick_up_hour',
       'Distance', 'speed', 'flag_Y', 'vendor_id_2', 'month_2', 'month_3',
       'month_4', 'month_5', 'month_6', 'weekday_num_1', 'weekday_num_2',
       'weekday_num_3', 'weekday_num_4', 'weekday_num_5', 'weekday_num_6',
       'pickup_hour_1', 'pickup_hour_2', 'pickup_hour_3', 'pickup_hour_4',
       'pickup_hour_5', 'pickup_hour_6', 'pickup_hour_7', 'pickup_hour_8',
       'pickup_hour_9', 'pickup_hour_10', 'pickup_hour_11', 'pickup_hour_12',
       'pickup_hour_13', 'pickup_hour_14', 'pickup_hour_15', 'pickup_hour_16',
       'pickup_hour_17', 'pickup_hour_18', 'pickup_hour_19', 'pickup_hour_20',
       'pickup_hour_21', 'pickup_hour_22', 'pickup_hour_23',
       'passenger_count_1', 'passenger_count_2', 'passenger_count_3',
       'passenger_count_4', 'passenger_count_5', 'passenger_count_6',
       'passenger_count_7', 'passenger_count_9'],
      dtype='object')
```

0.1.12 Univariate analysis

0.1.13 Passenger count

0.1.14 New York City Taxi Passenger Limit says:

- A maximum of 4 passengers can ride in traditional cabs, there are also 5 passenger cabs that look more like minivans.

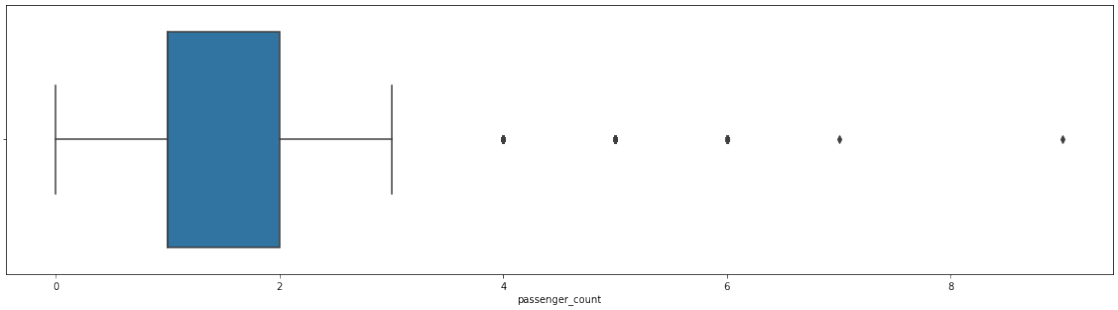
- A child under 7 is allowed to sit on a passenger's lap in the rear seat in addition to the passenger limit.

So, in total we can assume that maximum 6 passenger can board the new york taxi i.e. 5 adult + 1 minor

[18]:
```
data.passenger_count.value_counts()
```

[18]:
```
1    517415
2    105097
5     38926
3     29692
6     24107
4     14050
0        33
7         1
9         1
Name: passenger_count, dtype: int64
```

[19]:
```
plt.figure(figsize = (20,5))
sns.boxplot(data.passenger_count)
plt.show()
```



0.1.15 Observation

- Most of the trips are either 1 or 2
- There are 0 passenger count
- Few trips consisted of even 7, 8 or 9 passengers. Clear outliers and pointers to data inconsistency

0.1.16 Idea:

Passenger count is a driver entered value. Since the trip is not possible without passengers. So the driver forgot to enter the value for the trips with 0 passenger count. Lets analyze the passenger count distribution further to make it consistent for further analysis

[20]:
```
data['passenger_count'].describe()
```

[20]:
```
count    729322.000000
mean          1.662055
std           1.312446
min           0.000000
25%           1.000000
50%           1.000000
75%           2.000000
max           9.000000
Name: passenger_count, dtype: float64
```

As per above details. Mean median and mode are all approx equal to 1. So we would replace the 0 passenger count with 1.

[21]:
```
data['passenger_count'] = data.passenger_count.map(lambda x: 1 if x == 0 else x)
```

Also, we will remove the records with passenger count $>$ 7, 8 or 9 as they are extreme values and looks very odd to be ocupied in a taxi.

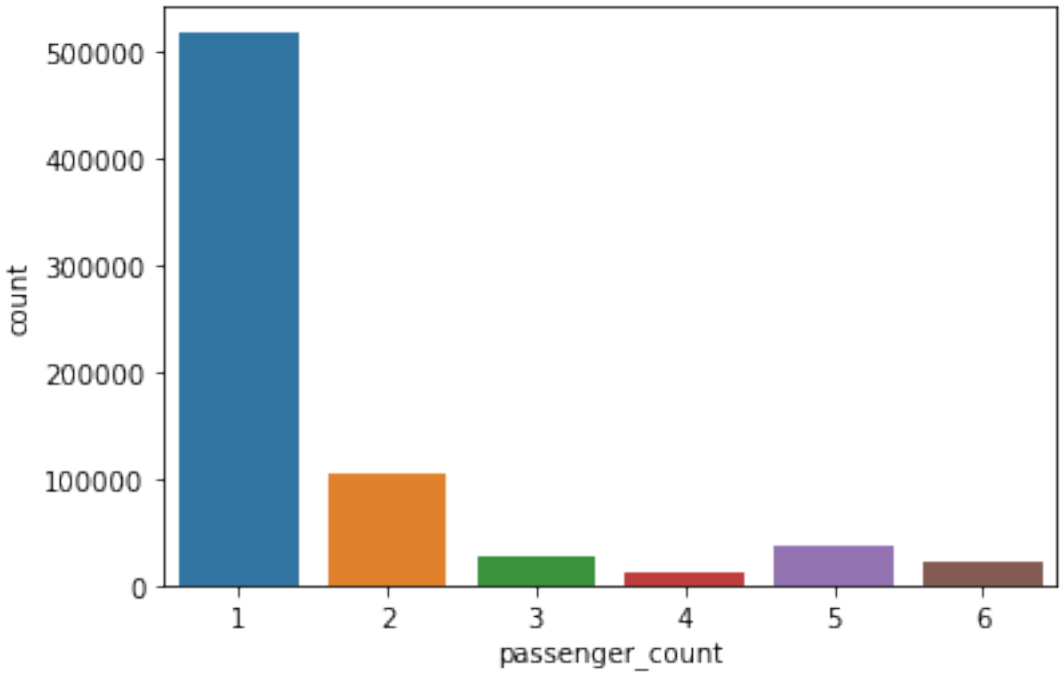
[22]:
```
data = data[data.passenger_count <= 6]
```

[23]:
```
data.passenger_count.value_counts()
```

[23]:
```
1    517448
2    105097
5     38926
3     29692
6     24107
4     14050
Name: passenger_count, dtype: int64
```

0.1.17 Data is consistent.

[24]:
```
sns.countplot(data.passenger_count)
plt.show()
```

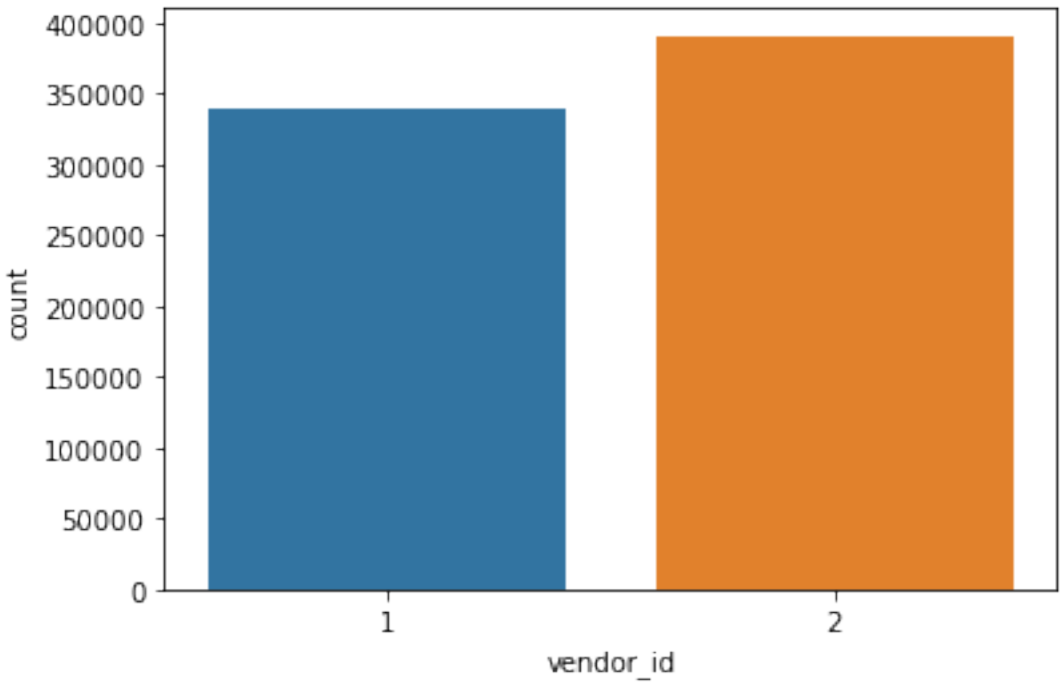


0.1.18 Observation

- Most of the trips was taken by single passenger and that is inline with our day to day observations

0.1.19 Vendor

[25]:
```
sns.countplot(data.vendor_id)
plt.show()
```



0.1.20 Observation

- Vendor 2 is evidently more famous among the population as per the above graph. It has more number of trips as compared to vendor 1
- Vendor 2 has greater market share than vendor 1

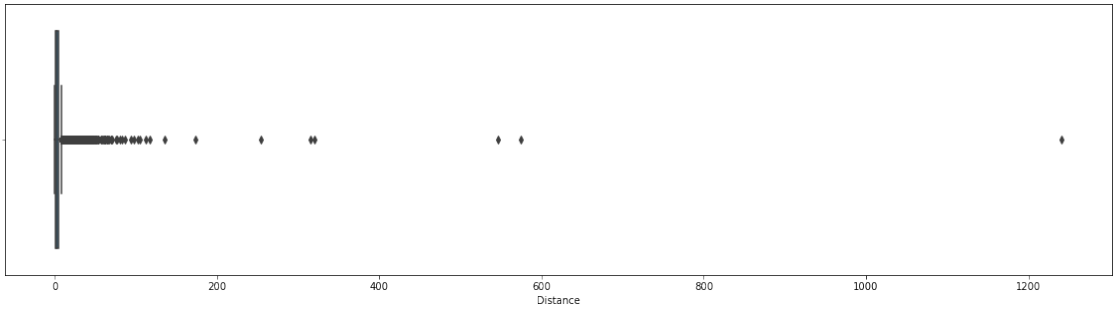
0.1.21 Distance

Distribution of distance at different rides

[26]:
```
data['Distance'].describe()
```

[26]:
```
count    729320.000000
mean          3.441153
std           4.353140
min           0.000000
25%           1.232700
50%           2.095678
75%           3.876491
max        1240.910391
Name: Distance, dtype: float64
```

[27]:
```
plt.figure(figsize = (20,5))
sns.boxplot(data.Distance)
plt.show()
```



0.1.22 Findings

- There are trips where more than 100km distance
- Some of the trips whosee distance value is 0km

0.1.23 Observations

- mean distance travelled is approx 3.5 kms.
- standard deviation of 4.3 which shows that most of the trips are limited to the range of 1-10 kms.

[28]:
```
print("There are {} trip records with 0 km distance".format(data.Distance[data.
↪Distance == 0 ].count()))
```

```
There are 2900 trip records with 0 km distance
```

[29]:
```
data[data.Distance==0].head()
```

[29]:

	id	vendor_id	pickup_datetime	dropoff_datetime	\
263	id3155891	2	2016-06-28 11:21:00	2016-06-28 11:25:00	
327	id0786923	2	2016-03-26 13:34:38	2016-03-26 13:37:17	
795	id2323213	2	2016-06-13 16:49:52	2016-06-13 17:04:49	
1176	id3235868	1	2016-02-29 21:39:52	2016-02-29 21:44:08	
1257	id1865738	2	2016-03-13 11:38:36	2016-03-13 12:00:46	

	passenger_count	pickup_longitude	pickup_latitude	dropoff_longitude	\
263	2	-73.996422	40.298828	-73.996422	
327	1	-73.996323	40.753460	-73.996323	
795	5	-73.967171	40.763500	-73.967171	
1176	1	-73.995232	40.744038	-73.995232	
1257	2	-73.912781	40.804428	-73.912781	

	dropoff_latitude	store_and_fwd_flag	…	pickup_hour_22	pickup_hour_23	\
263	40.298828	N	…	0	0	
327	40.753460	N	…	0	0	
795	40.763500	N	…	0	0	

```
1176         40.744038                  N  …                0              0
1257         40.804428                  N  …                0              0

      passenger_count_1  passenger_count_2  passenger_count_3  \
263                   0                  1                  0
327                   1                  0                  0
795                   0                  0                  0
1176                  1                  0                  0
1257                  0                  1                  0

      passenger_count_4  passenger_count_5  passenger_count_6  \
263                   0                  0                  0
327                   0                  0                  0
795                   0                  1                  0
1176                  0                  0                  0
1257                  0                  0                  0

      passenger_count_7  passenger_count_9
263                   0                  0
327                   0                  0
795                   0                  0
1176                  0                  0
1257                  0                  0

[5 rows x 61 columns]
```

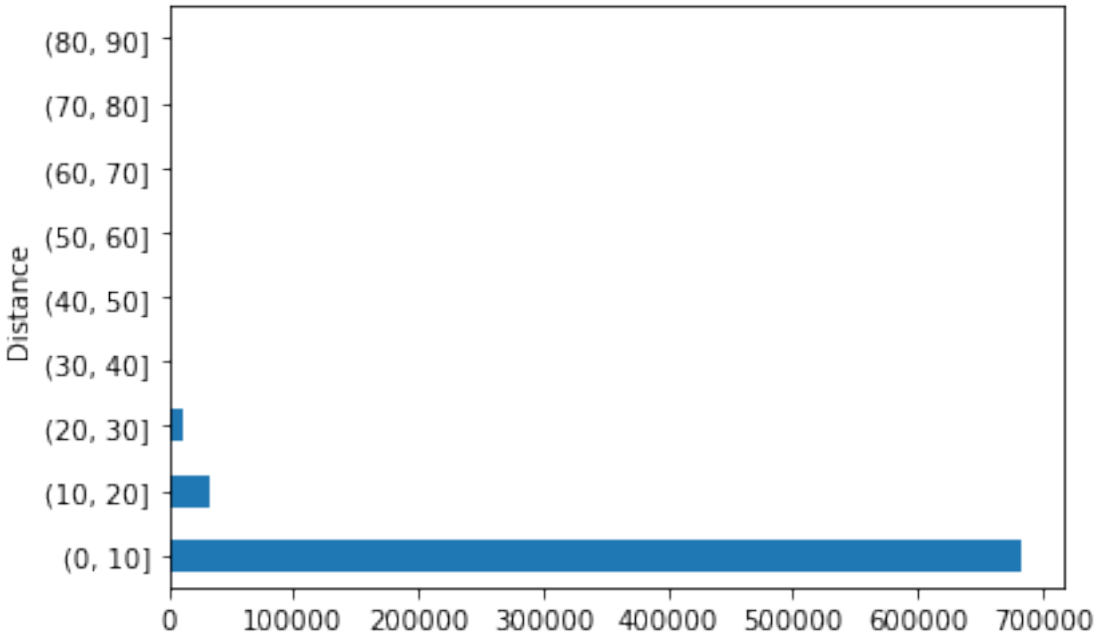
0.1.24 Observation

Around 3K trip record with distance equal to 0. Below are some possible explanation for such records.

1) Customer changed mind and cancelled the journey just after accepting it.
2) Software didn't recorded dropoff location properly due to which dropoff location is the same as the pickup location.
3) Issue with GPS tracker while the journey is being finished.
4) Driver cancelled the trip just after accepting it due to some reason. So the trip couldn't start
5) Or some other issue with the software itself which a technical guy can explain

There is some serious inconsistencies in the data where drop off location is same as the pickup location. Imputing the distance values is not possible by considering a correlation with the duration, then the dropoff_location coordinates would not be inline with the distance otherwise.

[30]:
```
data.Distance.groupby(pd.cut(data.Distance, np.arange(0,100,10))).count().
 ↪plot(kind='barh')
plt.show()
```



0.1.25 Observation

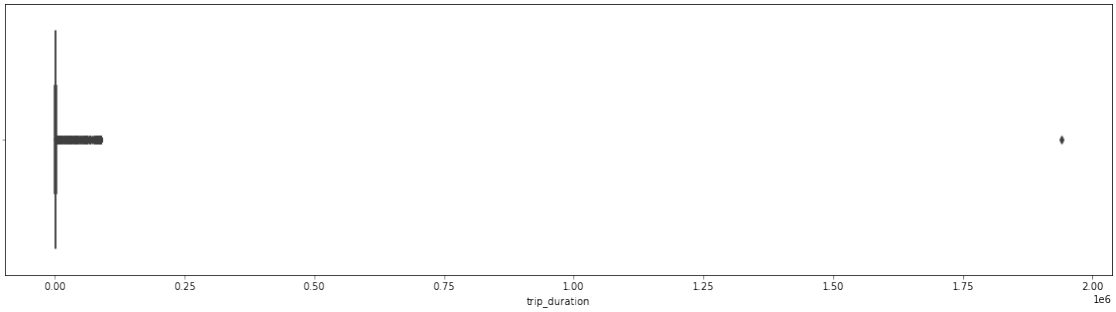
- Most of the rides are completed between 1-10 Kms with some of the rides with distances between 10-30 kms.
- Other slabs bar are not visible because the number of trips are very less as compared to these slabs

0.1.26 Trip Duration

[31]:
```
data.trip_duration.describe()
```

[31]:
```
count    7.293200e+05
mean     9.522310e+02
std      3.864631e+03
min      1.000000e+00
25%      3.970000e+02
50%      6.630000e+02
75%      1.075000e+03
max      1.939736e+06
Name: trip_duration, dtype: float64
```

[32]:
```
plt.figure(figsize = (20,5))
sns.boxplot(data.trip_duration)
plt.show()
```



0.1.27 Observation

- Most of the trips are from 400 seconds to 1075 seconds
- There are outliers which are to be removed.
- Mean and median are not same and mean is greater than median which means the data is right-skewed.
- There are some durations with as low as 1 second. which points towards trips with 0 km distance.

[33]:
```
data.trip_duration.groupby(pd.cut(data.trip_duration, np.arange(1,max(data.
↪trip_duration),3600))).count()
```

[33]:
```
trip_duration
(1, 3601]              723251
(3601, 7201]             4964
(7201, 10801]              61
(10801, 14401]             15
(14401, 18001]              2
                          ...
(1918801, 1922401]          0
(1922401, 1926001]          0
(1926001, 1929601]          0
(1929601, 1933201]          0
(1933201, 1936801]          0
Name: trip_duration, Length: 538, dtype: int64
```

0.1.28 Observation

- Most of the trips occurs within 1 hour with some good numbers of trips duration going above 1 hour.
- There are trips with ore than 24 hours of travel duration i.e. 86400 seconds which might have occured for the outstation travels.

[34]:
```
data[data.trip_duration > 86400]
```

[34]:
```
              id  vendor_id     pickup_datetime    dropoff_datetime  \
21813  id1864733          1 2016-01-05 00:19:42 2016-01-27 11:08:38

       passenger_count  pickup_longitude  pickup_latitude  dropoff_longitude  \
21813                1         -73.78965        40.643559          -73.95681

       dropoff_latitude store_and_fwd_flag  …  pickup_hour_22  \
21813         40.773087                  N  …               0

      pickup_hour_23  passenger_count_1  passenger_count_2  passenger_count_3  \
21813              0                  1                  0                  0

       passenger_count_4  passenger_count_5  passenger_count_6  \
21813                  0                  0                  0

       passenger_count_7  passenger_count_9
21813                  0                  0

[1 rows x 61 columns]
```

0.1.29 Observation

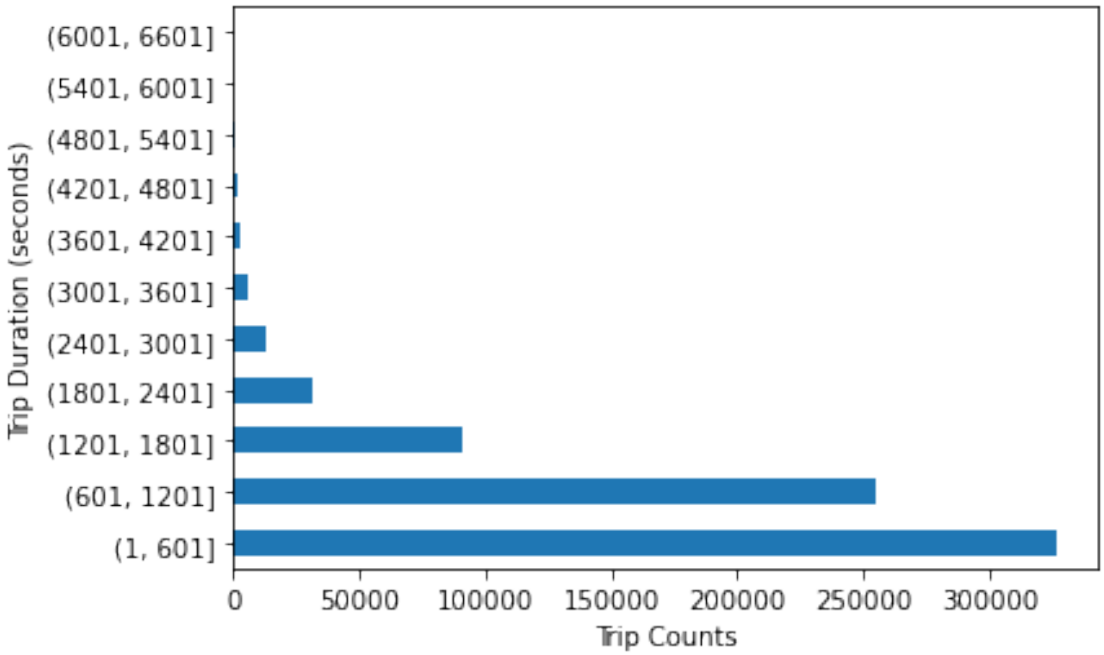
- One trip ran for more than 20 days.
- This trip is taken by vendor 1 which they might allow longer trip
- The trip taken on Tuesday in the first month with the distance of 20kms.

They fail for correct predictions and they bring inconsistency in the algorithm

[35]:
```
data = data[data.trip_duration <= 86400]
```

Visualizing the number of trips taken in slabs of 0-10, 20-30 ... minutes respectively

[36]:
```
data.trip_duration.groupby(pd.cut(data.trip_duration, np.arange(1,7200,600))).
↪count().plot(kind='barh')
plt.xlabel('Trip Counts')
plt.ylabel('Trip Duration (seconds)')
plt.show()
```



0.1.30 Obseravtion

- Most of the trips took 0 - 30 mins to complete

0.1.31 Speed

Speed is a function of distance and time. Let's visualize speed in different trips. Maximum speed limit in NYC is as follows:

25 mph in urban area i.e. 40 km/hr

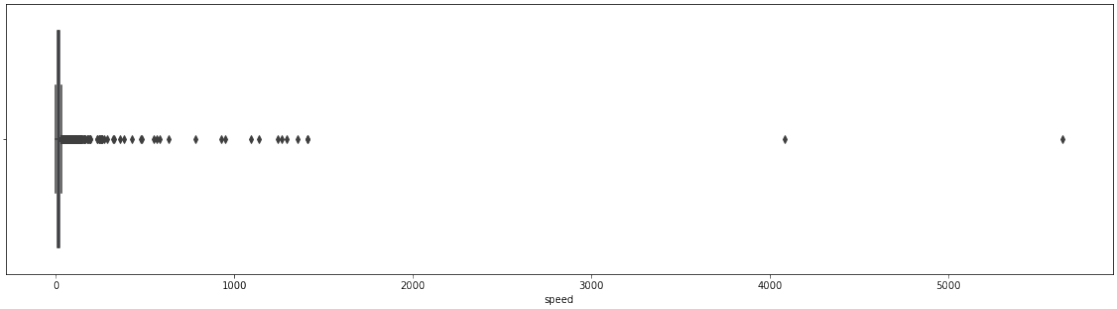
65 mph on controlled state highways i.e. approx 104 km/hr

[37]:
```
data['speed'].describe()
```

[37]:
```
count    729319.000000
mean         14.421527
std          12.341036
min           0.000000
25%           9.124341
50%          12.796887
75%          17.844052
max        5640.501776
Name: speed, dtype: float64
```

[38]:
```
plt.figure(figsize = (20,5))
sns.boxplot(data.speed)
```

```
plt.show()
```

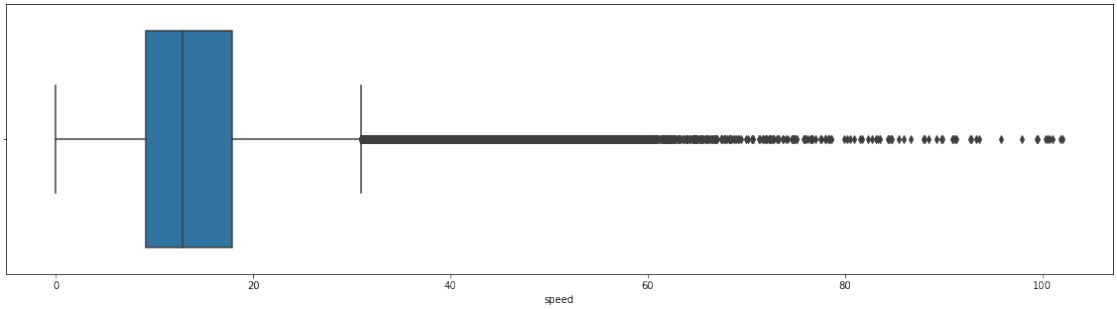


0.1.32 Observations

- Most of the trips are above 104km/hr. These are the outliers
- The maximun speed is 104km/hr on controlles state highways

Removing the speed which are greater than 104km/hr

[39]:
```
data = data[data.speed <= 104]
plt.figure(figsize = (20,5))
sns.boxplot(data.speed)
plt.show()
```



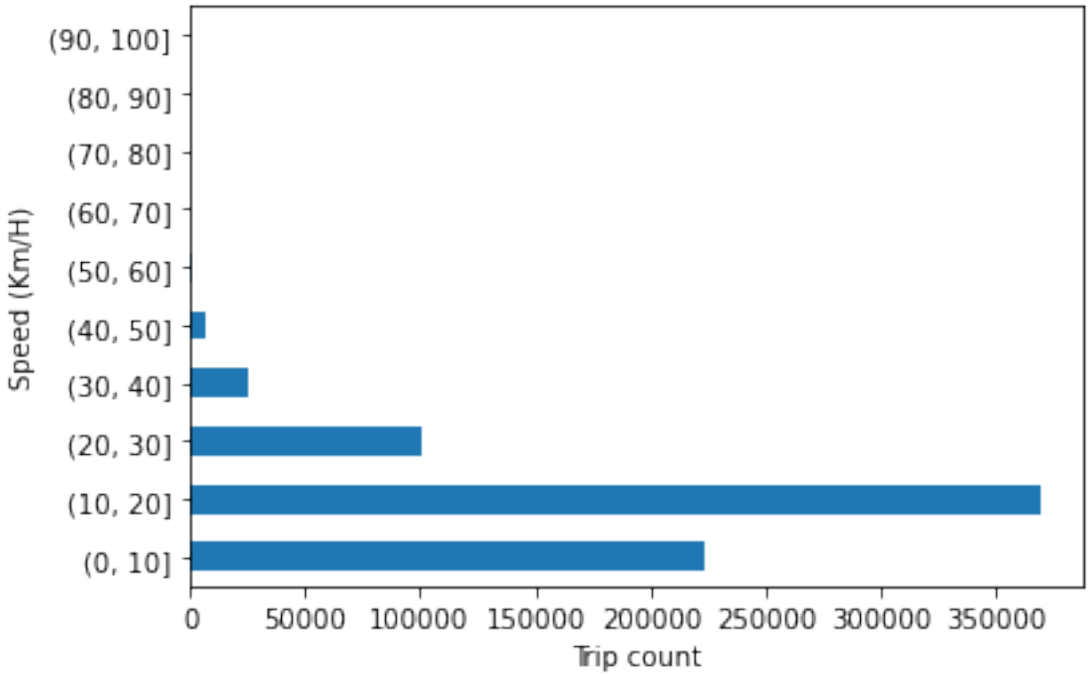
0.1.33 Observations

- Trips over 30 km/hr are being considered as outliers but we cannot ignore them because they are well under the highest speed limit of 104 km/h on state controlled highways.
- Mostly trips are done at a speed range of 10-20 km/hr with an average speed of around 14 km/hr.

speed range ditribution

[40]:
```
data.speed.groupby(pd.cut(data.speed, np.arange(0,104,10))).count().plot(kind =␣
↪'barh')
```

```
plt.xlabel('Trip count')
plt.ylabel('Speed (Km/H)')
plt.show()
```



0.1.34 Observation

It has been proved from this graph that most of the trips were done at a speed range of 10-20 km/hr.

0.1.35 Store_and_fwd_flag

This flag indicates whether the trip record was held in vehicle memory before sending to the vendor because the vehicle did not have a connection to the server - Y=store and forward; N=not a store and forward trip.

[41]:
```
data.flag_Y.value_counts(normalize=True)
```

[41]:
```
0    0.994461
1    0.005539
Name: flag_Y, dtype: float64
```

0.1.36 Observations

- Above result shows that only about 0.5% of the trip details were stored in the vehicle first before sending it to the server.

This might have occured because of the following reasons:

1) Outstation trips didn't had proper connection at the time when trip completes.
2) Temporary loss of signals while the trip was about to finish
3) Inconsistent signal reception over the trip duration.
4) The GPS or mobile device battery was down when the trip finished.

[42]:
```
data.flag_Y.value_counts()
```

[42]:
```
0    725200
1      4039
Name: flag_Y, dtype: int64
```

0.1.37 Observation

- Around 4K trips had to store the flag and then report to the server when the connection was established.

Distribution for the vendors of the offline trip

[43]:
```
data.vendor_id[data.flag_Y == 1].value_counts()
```

[43]:
```
1    4039
Name: vendor_id, dtype: int64
```

0.1.38 Observation

Above result shows that all the offline trips were taken by vendor 1. We already know that vendor 2 has greater market share as compared to vendor 1. So, there can be two reasons for this scenario.

1) Either vendor 1 utilizes advance technology than vendor 2 to store and forward trip details in case of temporary signal loss.
2) Or vendor 1 uses poor infrastructure which often suffers from the server connection instability due to which they have to store the trip info in the vehicle and send it to the server later when the server connection is back.

[44]:
```
data[data.flag_Y == 1]
```

[44]:

	id	vendor_id	pickup_datetime	dropoff_datetime	\
378	id1347533	1	2016-05-27 18:09:01	2016-05-27 18:16:30	
400	id2733049	1	2016-03-02 20:05:12	2016-03-02 20:52:52	
501	id2484490	1	2016-01-21 08:07:13	2016-01-21 08:18:21	
644	id2090829	1	2016-01-11 12:10:13	2016-01-11 12:25:41	
1278	id0512889	1	2016-06-10 21:20:14	2016-06-10 21:26:51	
…	…	…	…	…	
728481	id0008273	1	2016-04-08 17:52:56	2016-04-08 18:35:36	
728607	id3254730	1	2016-06-03 01:21:11	2016-06-03 01:30:16	

```
729074  id1347803          1 2016-03-17 01:24:10 2016-03-17 01:35:25
729119  id2265972          1 2016-01-07 07:51:18 2016-01-07 07:51:41
729217  id2475363          1 2016-06-17 13:40:15 2016-06-17 13:47:04

        passenger_count  pickup_longitude  pickup_latitude  dropoff_longitude  \
378                   1        -73.976051        40.744671         -73.979721
400                   2        -73.978134        40.757484         -73.998955
501                   1        -73.999771        40.739487         -73.983940
644                   2        -74.013611        40.714310         -73.976601
1278                  1        -73.958183        40.766190         -73.956032
…                   …                …                …                 …
728481                1        -73.969627        40.760384         -73.862061
728607                1        -74.004692        40.751846         -74.004860
729074                1        -73.988808        40.723038         -73.997543
729119                3        -73.782356        40.644211         -73.782364
729217                1        -73.985519        40.747314         -73.974854

        dropoff_latitude store_and_fwd_flag  …  pickup_hour_22  \
378           40.722958                   Y  …               0
400           40.614380                   Y  …               0
501           40.761421                   Y  …               0
644           40.751938                   Y  …               0
1278          40.782814                   Y  …               0
…                 …                   …  …               …
728481        40.768559                   Y  …               0
728607        40.735130                   Y  …               0
729074        40.695587                   Y  …               0
729119        40.644211                   Y  …               0
729217        40.755192                   Y  …               0

       pickup_hour_23  passenger_count_1  passenger_count_2  \
378                 0                  1                  0
400                 0                  0                  1
501                 0                  1                  0
644                 0                  0                  1
1278                0                  1                  0
…                 …                  …                  …
728481              0                  1                  0
728607              0                  1                  0
729074              0                  1                  0
729119              0                  0                  0
729217              0                  1                  0

        passenger_count_3  passenger_count_4  passenger_count_5  \
378                     0                  0                  0
400                     0                  0                  0
501                     0                  0                  0
```

```
644                     0                   0                   0
1278                    0                   0                   0
…                     …                   …                   …
728481                  0                   0                   0
728607                  0                   0                   0
729074                  0                   0                   0
729119                  1                   0                   0
729217                  0                   0                   0

        passenger_count_6  passenger_count_7  passenger_count_9
378                     0                   0                   0
400                     0                   0                   0
501                     0                   0                   0
644                     0                   0                   0
1278                    0                   0                   0
…                     …                   …                   …
728481                  0                   0                   0
728607                  0                   0                   0
729074                  0                   0                   0
729119                  0                   0                   0
729217                  0                   0                   0

[4039 rows x 61 columns]
```

0.1.39 Observation

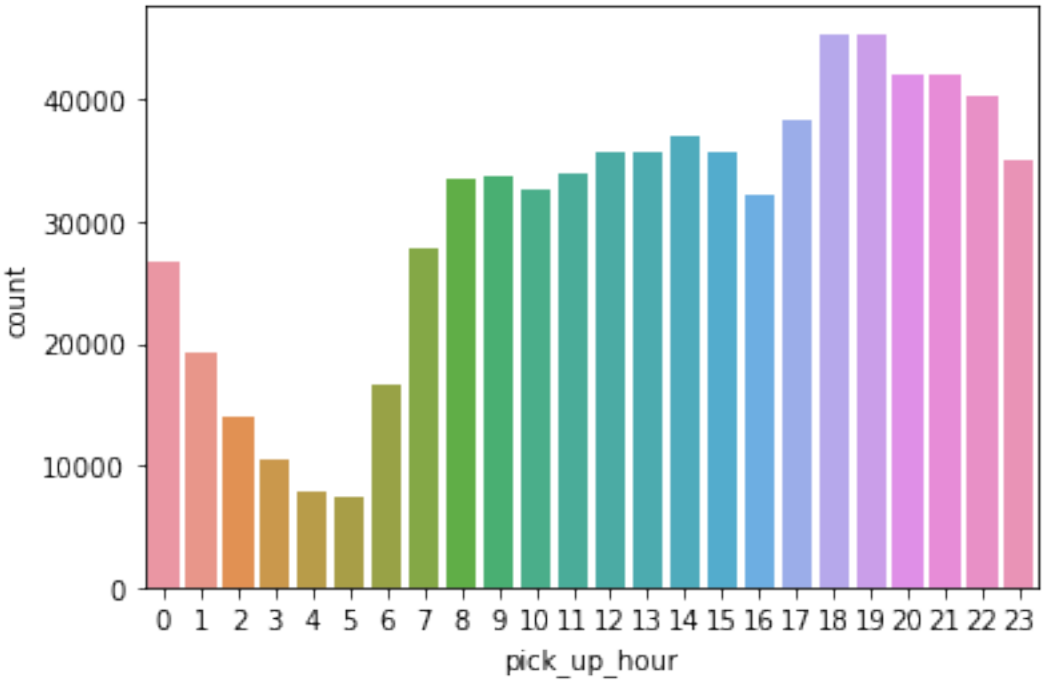
- Some trips are local some cover longer distance
- Almost each day is listed against offline trips.
- Offline trips were taken almost at all hours as per the search result.
- There is no month which appears to be more dominant in the results.
- Even the trip duration covers different scales.

So all in all there doesn't seems to be any relation with either of the metric for the offline

0.1.40 Total Trips per hour

Distribution of the pickups across the 24 hour time scale.

[45]:
```
sns.countplot(data.pick_up_hour)
plt.show()
```



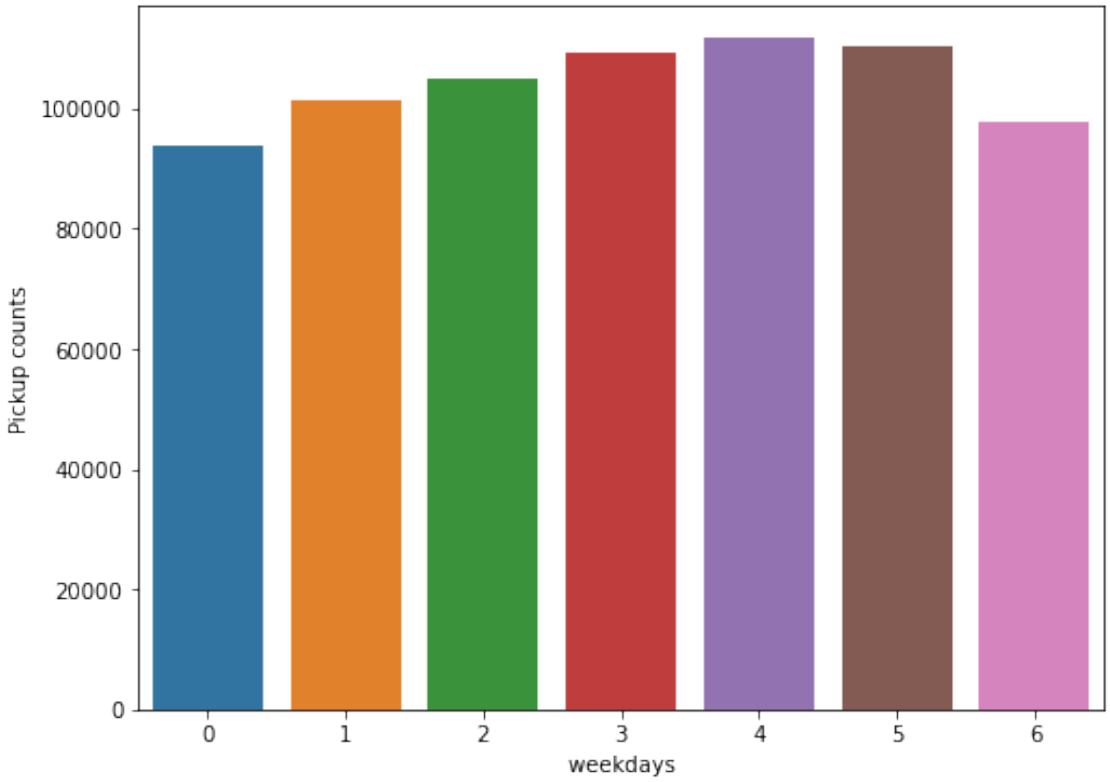
0.1.41 Observation

- Taxi pickups which starts increasing from 6AM in the morning and then declines from late evening i.e. around 7 PM.

0.1.42 Total trips per weekday

Distribution of taxi pickup across the week

[46]:
```
plt.figure(figsize = (8,6))
sns.countplot(data.weekday_num)
plt.xlabel(' weekdays ')
plt.ylabel('Pickup counts')
plt.show()
```

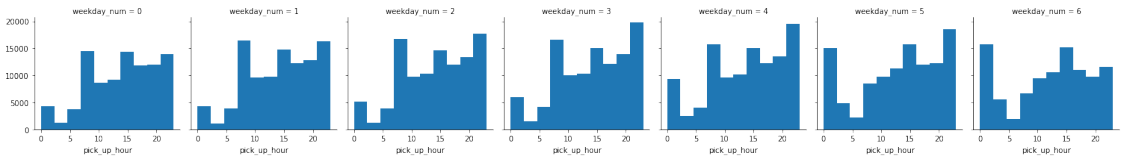


0.1.43 Observation

- Increasing trend of taxi pickups starting from Monday till Friday. The trend starts declining from saturday till monday which is normal where some office going people likes to stay at home for rest on the weekends.

Hourwise pickup pattern across the week

[47]:
```
n = sns.FacetGrid(data, col='weekday_num')
n.map(plt.hist, 'pick_up_hour')
plt.show()
```



0.1.44 Observation

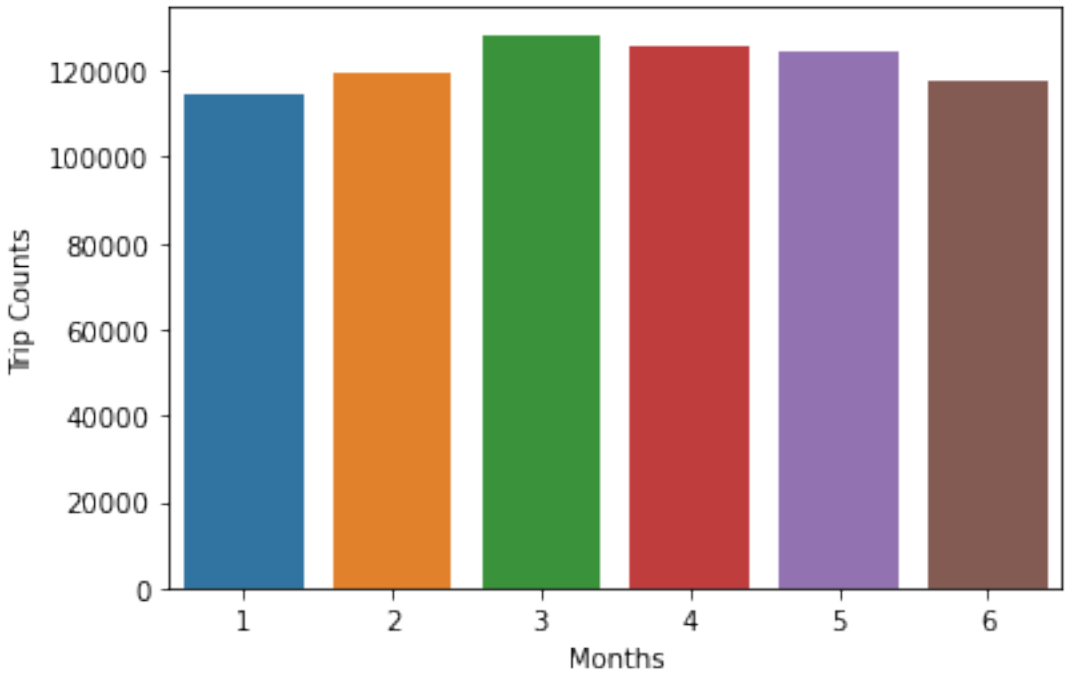
- Taxi pickups increased in the late night hours over the weekend possibly due to more outstation rides or for the late night leisures nearby activities.

- Early morning pickups i.e before 5 AM have increased over the weekend in comparison to the office hours pickups i.e. after 7 AM which have decreased due to obvious reasons.
- Taxi pickups seems to be consistent across the week at 15 Hours i.e. at 3 PM.

0.1.45 Total trips per month

Distribution of trip across the months

[48]:
```
sns.countplot(data.month)
plt.ylabel('Trip Counts')
plt.xlabel('Months')
plt.show()
```



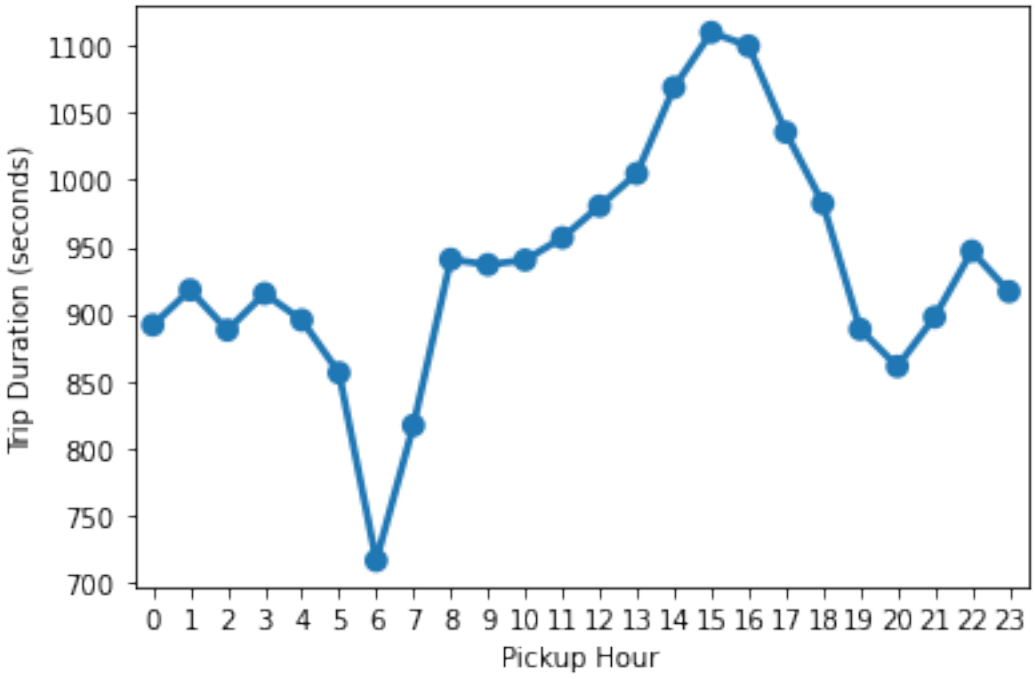
0.1.46 Observation

- There is balance in the trip across months

0.1.47 Bivariate Analysis

0.1.48 Trip duration per hour

[49]:
```
group1 = data.groupby('pick_up_hour').trip_duration.mean()
sns.pointplot(group1.index, group1.values)
plt.ylabel('Trip Duration (seconds)')
plt.xlabel('Pickup Hour')
plt.show()
```

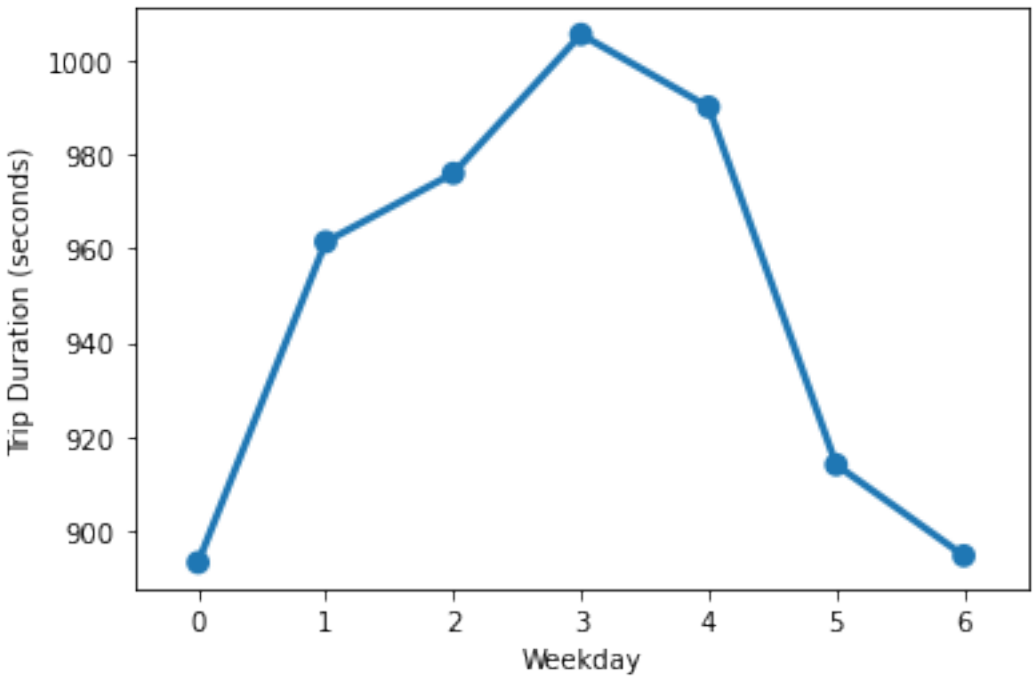


0.1.49 Observation

- Average trip duration is lowest at 6 AM when there is minimal traffic on the roads.
- Average trip duration is generally highest around 3 PM during the busy streets.
- Trip duration on an average is similar during early morning hours i.e. before 6 AM & late evening hours i.e. after 6 PM.

0.1.50 Trip duration per weekday

[50]:
```
group2 = data.groupby('weekday_num').trip_duration.mean()
sns.pointplot(group2.index, group2.values)
plt.ylabel('Trip Duration (seconds)')
plt.xlabel('Weekday')
plt.show()
```

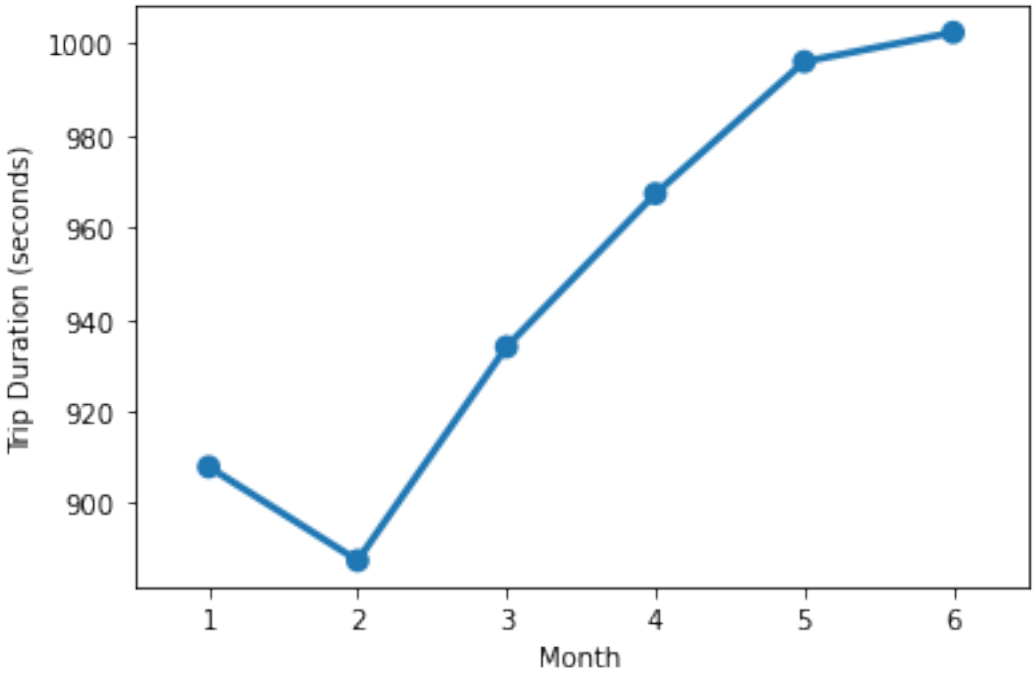


0.1.51 Observation

- Trip duration is almost equally distributed across the week on a scale of 0-1000 minutes with minimal difference in the duration times.
- Trip duration on thursday is longest among all days.

0.1.52 Trip duration per month

[51]:
```
group3 = data.groupby('month').trip_duration.mean()
sns.pointplot(group3.index, group3.values)
plt.ylabel('Trip Duration (seconds)')
plt.xlabel('Month')
plt.show()
```

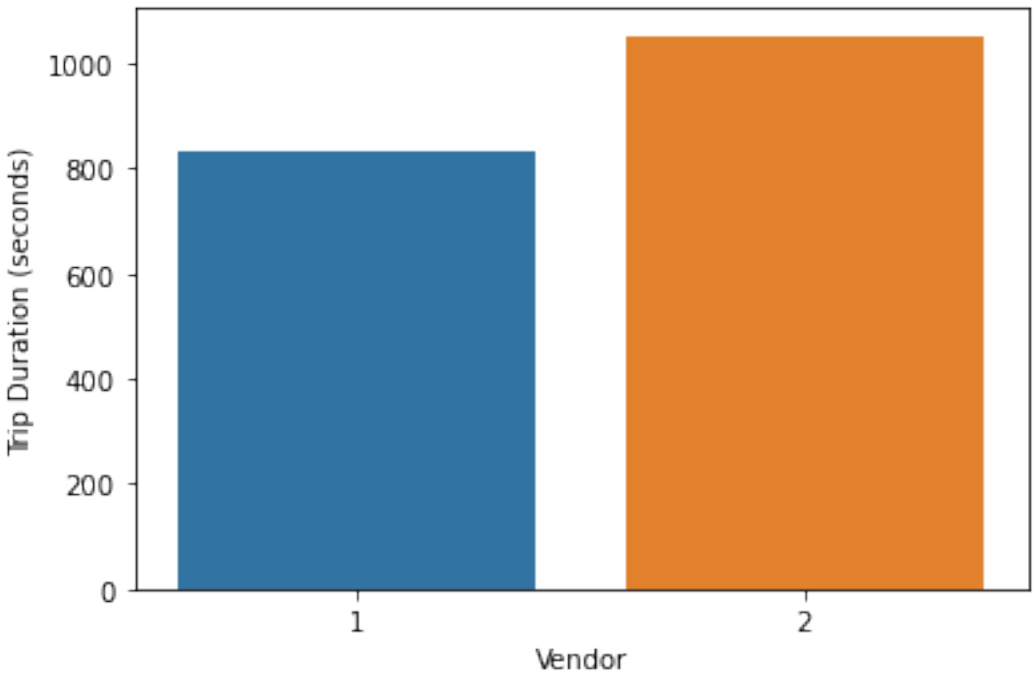


0.1.53 Observation

- Increasing trend in the average trip duration along with each subsequent month.
- The duration difference between each month is not much. It has increased gradually over a period of 6 months.
- It is lowest during february when winters starts declining.
- There might be some seasonal parameters like wind/rain which can be a factor of this gradual increase in trip duration over a period. Like May is generally the rainy season in NYC and which is inline with our visualization. As it generally takes longer on the roads due to traffic jams during rainy season. So natually the trip duration would increase towards April May and June.

0.1.54 Trip Duration per vendor

[52]:
```
group4 = data.groupby('vendor_id').trip_duration.mean()
sns.barplot(group4.index, group4.values)
plt.ylabel('Trip Duration (seconds)')
plt.xlabel('Vendor')
plt.show()
```

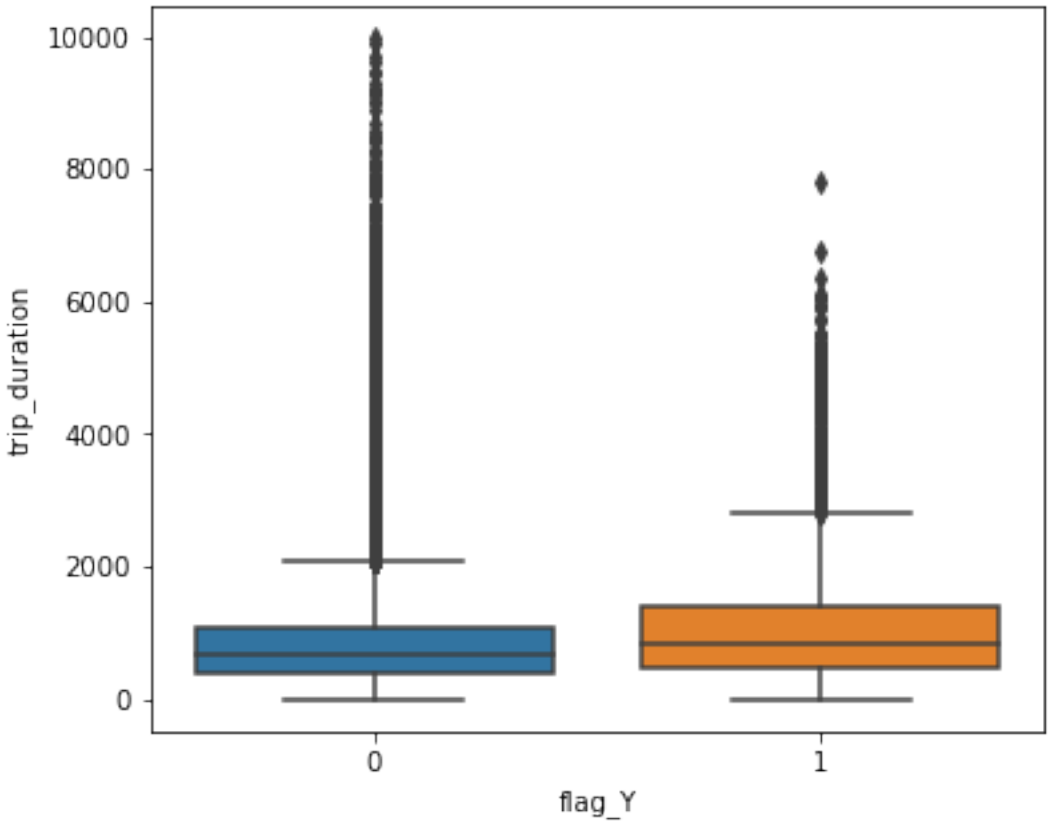


0.1.55 Observation

- Average trip duration for vendor 2 is higher than vendor 1 by approx 200 seconds i.e. atleast 3 minutes per trip.

0.1.56 Trip Duration Vs. flag

[53]:
```
plt.figure(figsize = (6,5))
plot_dur = data.loc[(data.trip_duration < 10000)]
sns.boxplot(x = "flag_Y", y = "trip_duration", data = plot_dur)
plt.show()
```



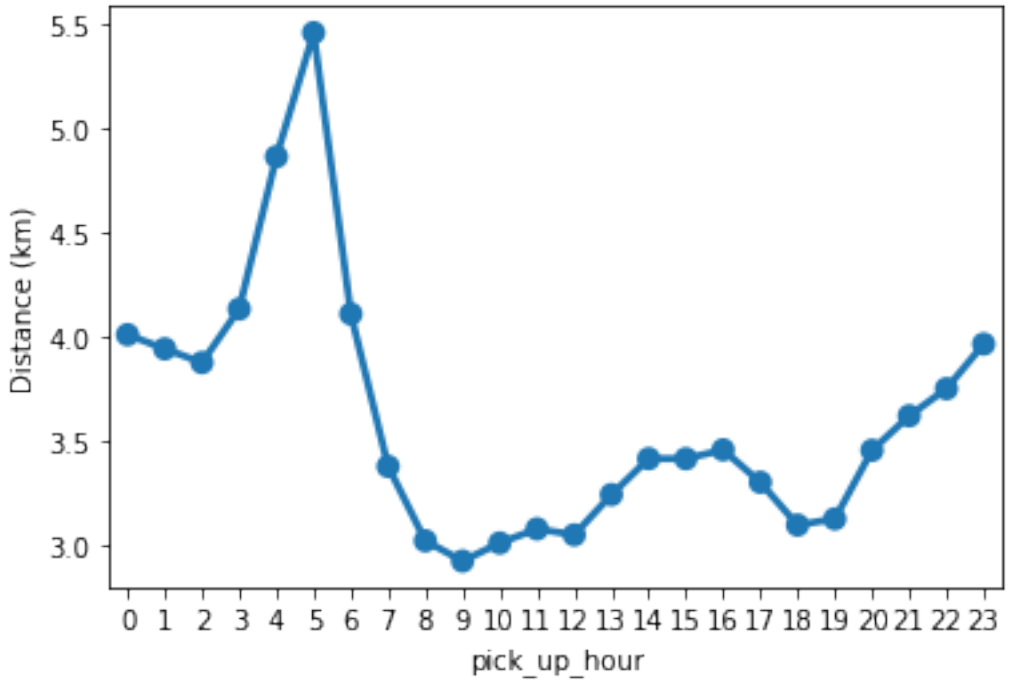
0.1.57 Observation

- Trip durations scale is less for the trips where the flag is set i.e. the trip details are stored before sending it to the server.
- Trip duration outliers are also less for the trips with flag 'Y' as compared the trips with flag 'N'.
- Trip duration is longer for the trips where the flag is not set.
- Inter quartile range of trip duration is more for the trips with the flag 'Y' as compared to the trips with flag 'N' but the median value is almost equal for both.

0.1.58 Distance per hour

Trip distance must be more or less proportional to the trip duration if we ignore general traffic and other stuff on the road.

[54]:
```
group5 = data.groupby('pick_up_hour').Distance.mean()
sns.pointplot(group5.index, group5.values)
plt.ylabel('Distance (km)')
plt.show()
```



0.1.59 Observation

- Trip distance is highest during early morning hours which can account for some things like:

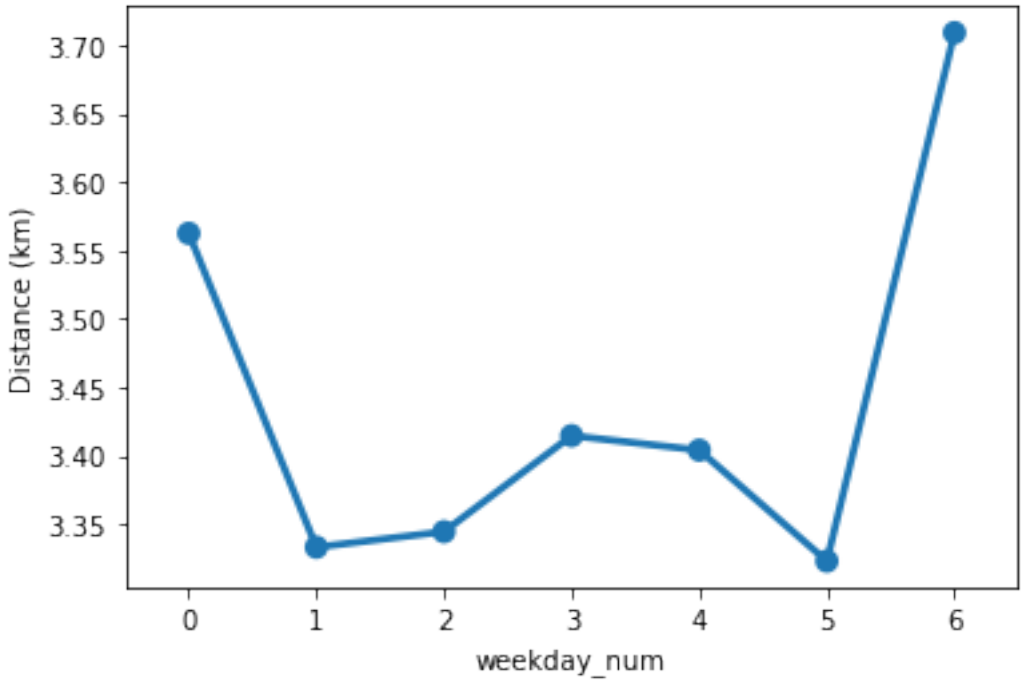
  1) Outstation trips taken during the weekends.

  2) Longer trips towards the city airport which is located in the outskirts of the city.

- Trip distance is fairly equal from morning till the evening varying around 3 - 3.5 kms.
- It starts increasing gradually towards the late night hours starting from evening till 5 AM and decrease steeply towards morning.

0.1.60 Distance per weekday

[55]:
```
group6 = data.groupby('weekday_num').Distance.mean()
sns.pointplot(group6.index, group6.values)
plt.ylabel('Distance (km)')
plt.show()
```

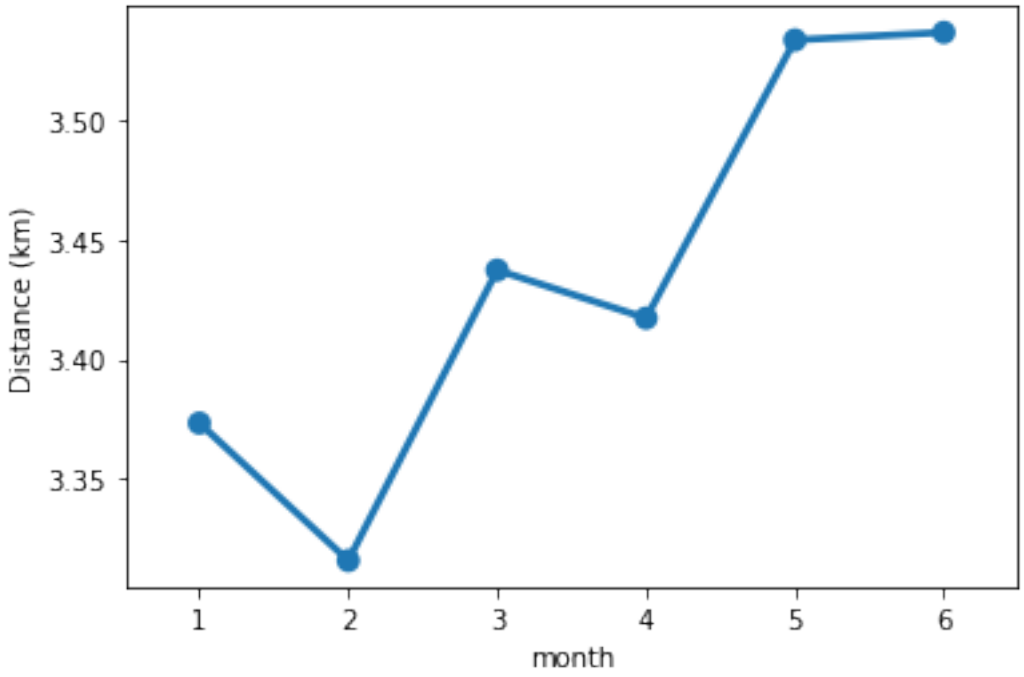


0.1.61 Observation

- Sunday to be in the top may be due to outstation trips or night trips towards the airport.
- Equal distribution with n average distance around 3.5 km/hr

0.1.62 Distance per month

[56]:
```
group7 = data.groupby('month').Distance.mean()
sns.pointplot(group7.index, group7.values)
plt.ylabel('Distance (km)')
plt.show()
```

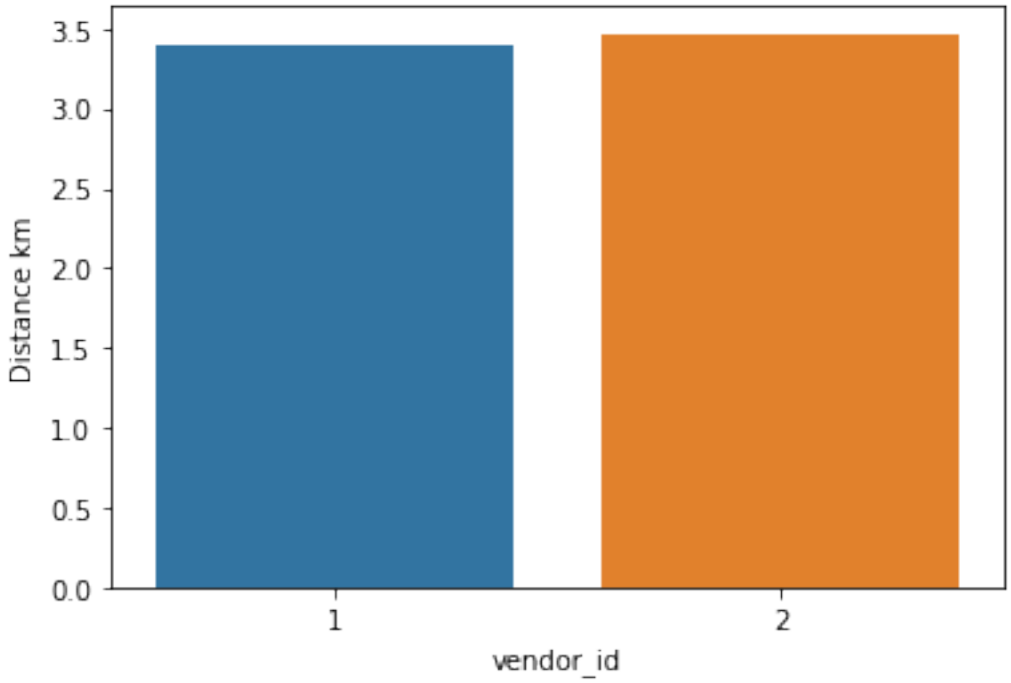


0.1.63 Observation

- The distibution is almost equivalent, varying mostly around 3.5 km/hr with 5th month being the highest in the average distance and 2nd month being the lowest.

0.1.64 Distance per vendor

[57]:
```
group8 = data.groupby('vendor_id').Distance.mean()
sns.barplot(group8.index, group8.values)
plt.ylabel("Distance km")
plt.show()
```

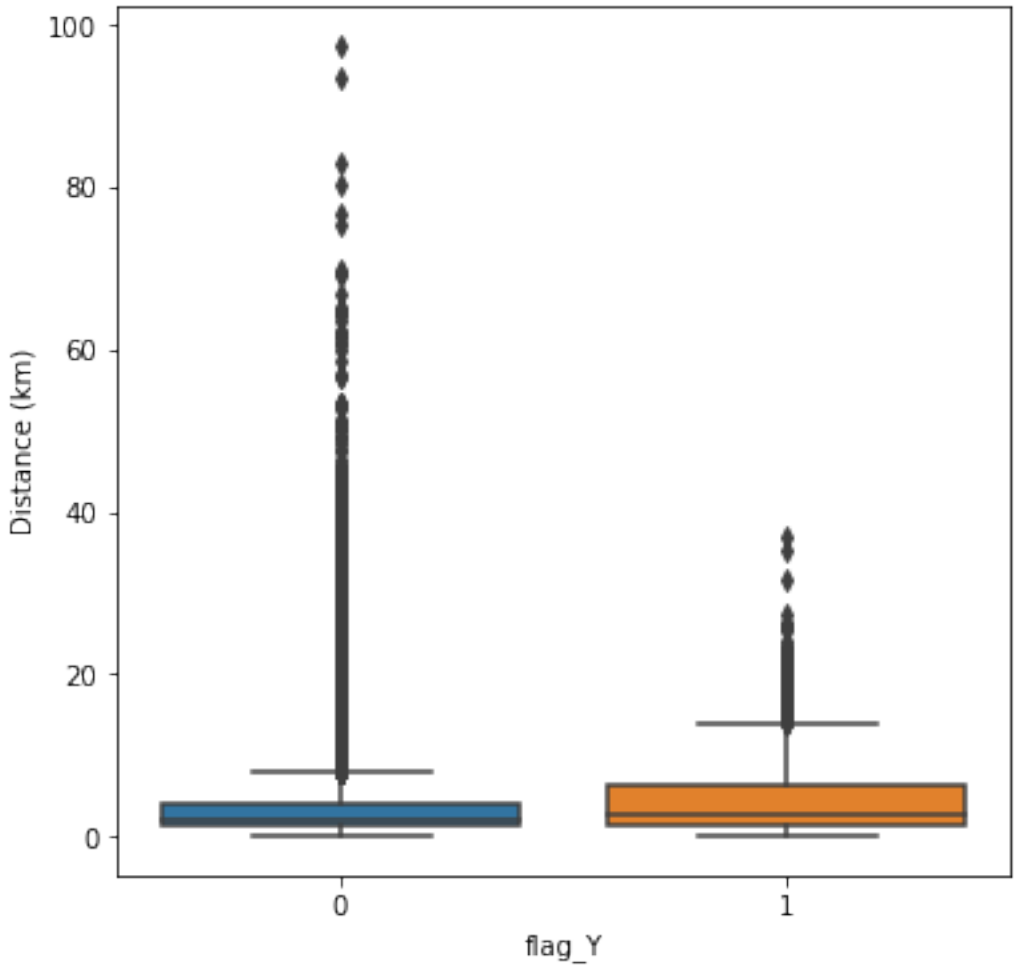


0.1.65 Observation

- Both the vendores are in the same with respect to distance

0.1.66 Distance Vs Flag

[58]:
```
plt.figure(figsize = (6,6))
plot_dist = data.loc[(data.Distance < 100)]
sns.boxplot(x = "flag_Y", y = "Distance", data = plot_dist)
plt.ylabel('Distance (km)')
plt.show()
```

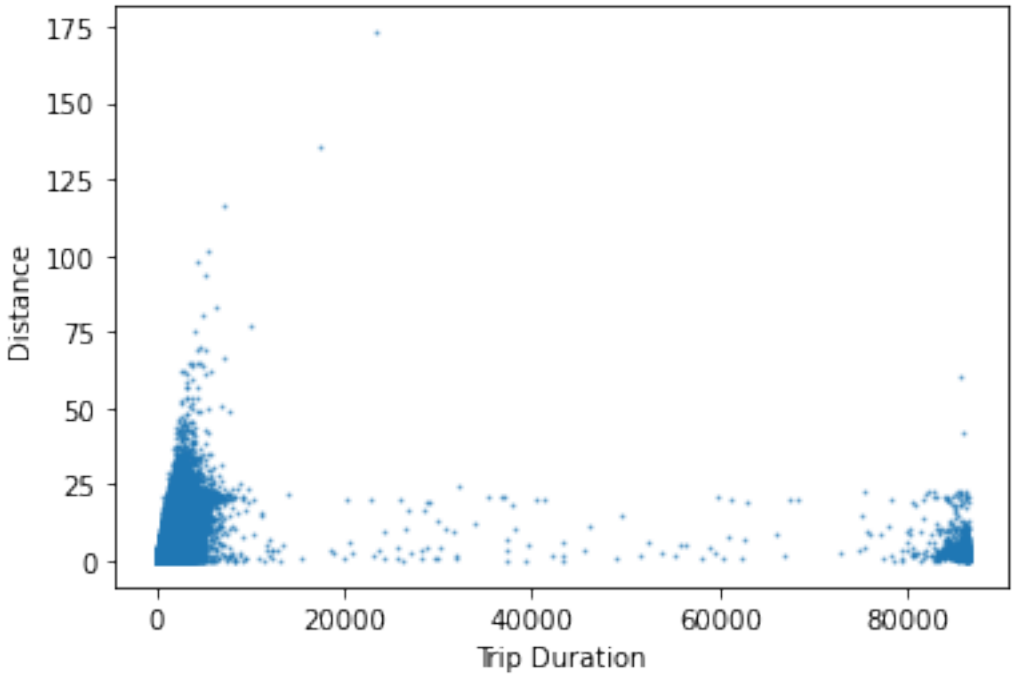


0.1.67 Observation

- Interquartile range of distance is almost twice for Flag 'Y' trips as compared to the Flag 'N' trips
- Median value is much different in both the case as well.
- The range of distance and trip duration for the Flag 'Y' trips is much more limited and confined as compared with the flag 'N' trips and this also resulted in much less number of outliers for Flag 'Y' trips.

0.1.68 Distance Vs. Trip duration

[59]:
```
plt.scatter(data.trip_duration, data.Distance , s=1, alpha=0.5)
plt.ylabel('Distance')
plt.xlabel('Trip Duration')
plt.show()
```

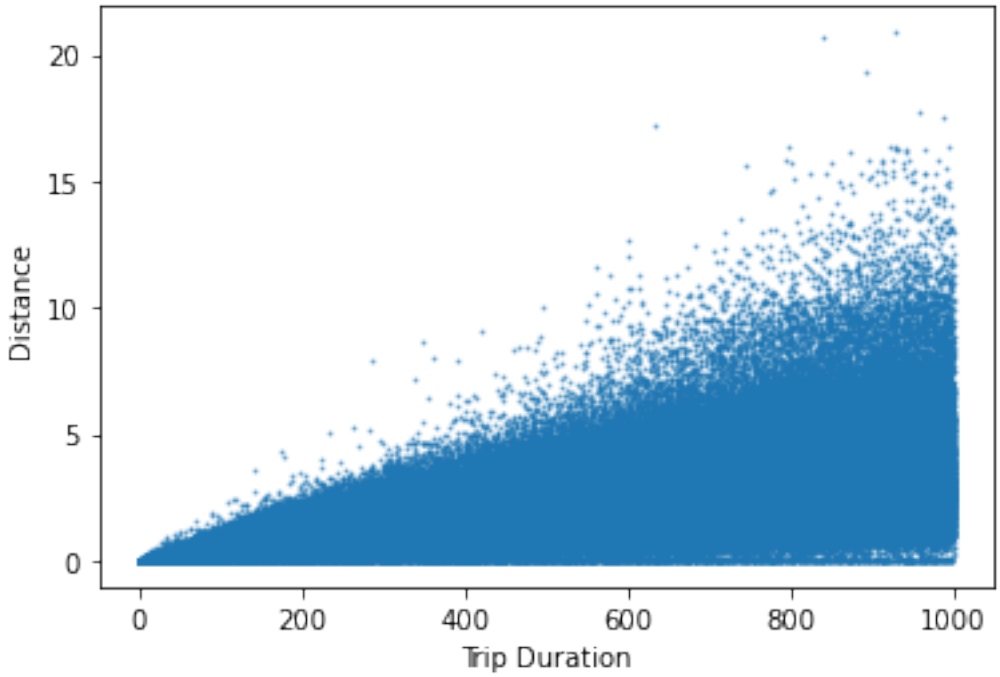


0.1.69 Observation

- There are lots of trips which covered negligible distance but clocked more than 20,000 seconds in terms of the Duration.
- Initially there is some proper correlation between the distance covered and the trip duration in the graph. but later on it all seems uncorrelated.
- There were few trips which covered huge distance of approx 200 kms within very less time frame, which is unlikely and should be treated as outliers.

Graph area where distance is < 50 km and duration is < 1000 seconds

[60]:
```
dur_dist = data.loc[(data.Distance < 50) & (data.trip_duration < 1000),␣
↪['Distance','trip_duration']]
plt.scatter(dur_dist.trip_duration, dur_dist.Distance , s=1, alpha=0.5)
plt.ylabel('Distance')
plt.xlabel('Trip Duration')
plt.show()
```



0.1.70 Observation

- There should have been a linear relationship between the distance covered and trip duration on an average but we can see dense collection of the trips in the lower right corner which showcase many trips with the inconsistent readings.

0.1.71 Idea:

- Remove those trips which covered 0 km distance but clocked more than 1 minute to make our data more consistent for predictive model. Because if the trip was cancelled after booking, than that should not have taken more than a minute time. This is our assumption.

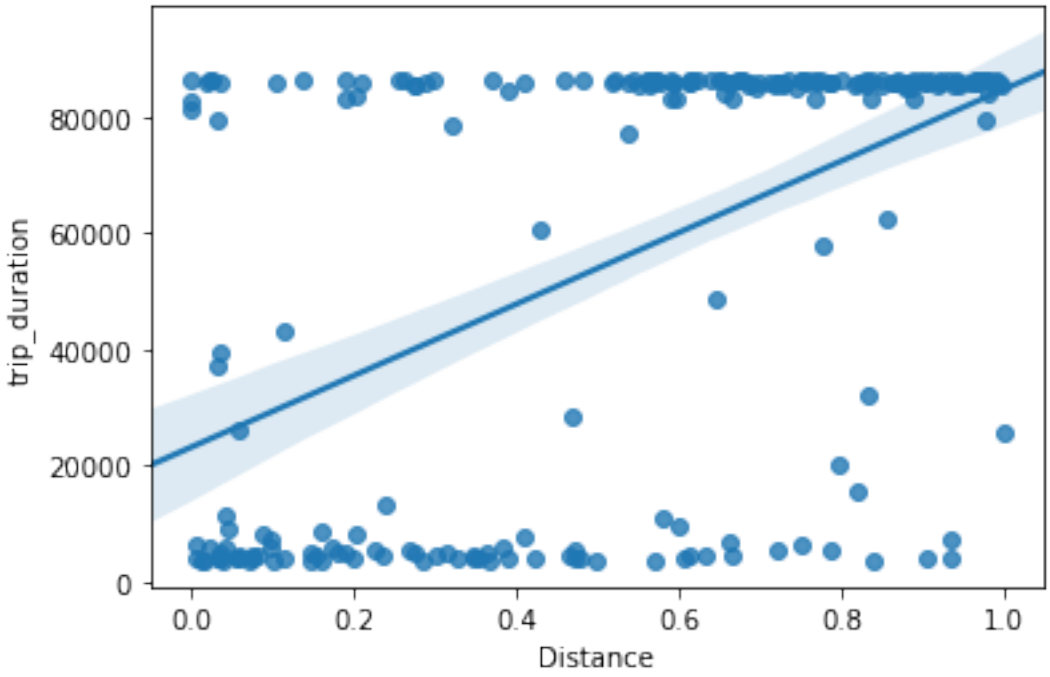
[61]:
```
data = data[~((data.Distance == 0) & (data.trip_duration >= 60))]
```

0.1.72 Observation

- Now, Instead of looking at each and every trip, we should approximate and try to filter those trips which covered less than 1 km distance and but clocked more than an hour.

[62]:
```
duo = data.loc[(data['Distance'] <= 1) & (data['trip_duration'] >=␣
↪3600),['Distance','trip_duration']].reset_index(drop=True)
```

[63]:
```
sns.regplot(duo.Distance, duo.trip_duration)
plt.show()
```



0.1.73 Observations:

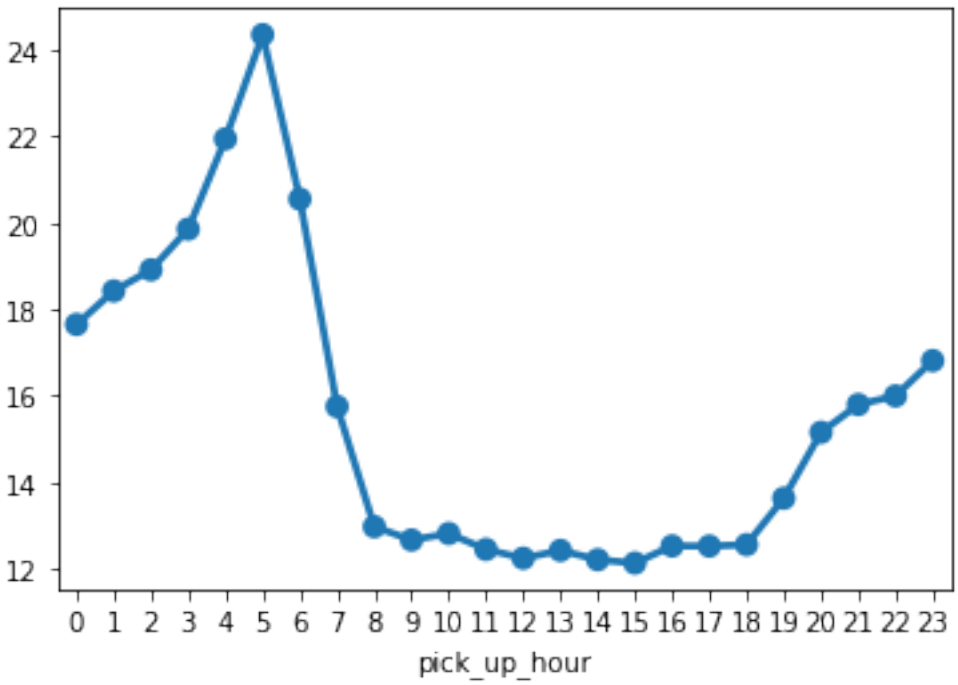
- Though the straight line tries to show some linear relation between the two. But there seems to be negligible correlation between these two metric as seen from the scatter plot where it should have been a linear distribution.
- It is rarely occurs that customer keep sitting in the taxi for more than an hour and it does not travel for even 1 km.

These should be removed to bring in more consistency to our results.

[64]:
```
data = data[~((data['Distance'] <= 1) & (data['trip_duration'] >= 3600))]
```

0.1.74 Average speed per hour

[65]:
```
group9 = data.groupby('pick_up_hour').speed.mean()
sns.pointplot(group9.index, group9.values)
plt.show()
```

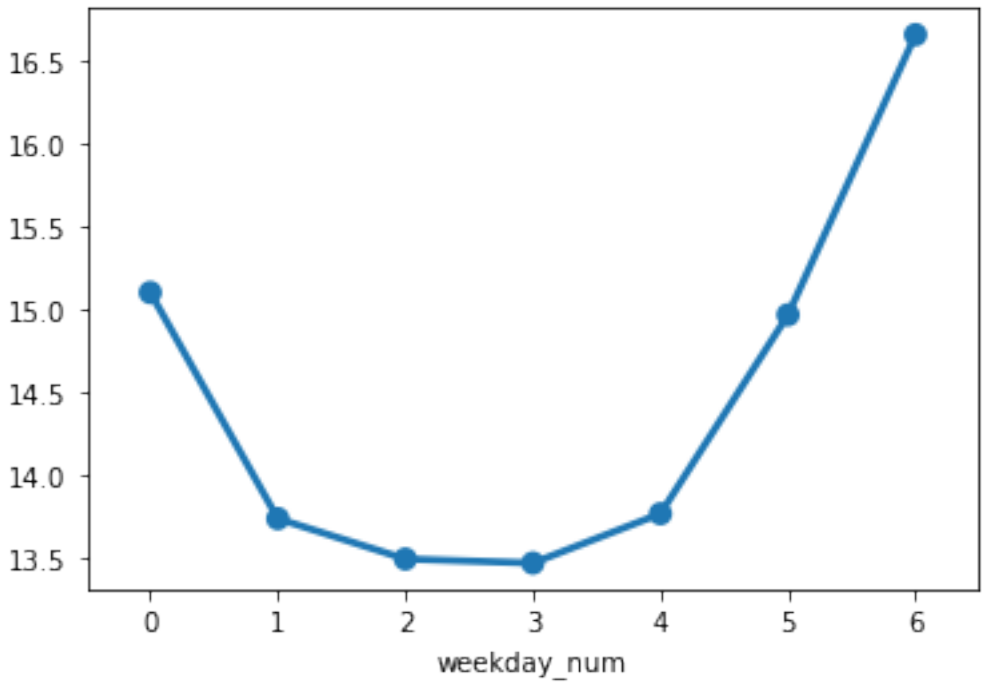


0.1.75 Observation

- The average trend is totally inline with the normal circumstances.
- Average speed tend to increase after late evening and continues to increase gradually till the late early morning hours.
- Average taxi speed is highest at 5 AM in the morning, then it declines steeply as the office hours approaches.
- Average taxi speed is more or less same during the office hours i.e. from 8 AM till 6PM in the evening.

0.1.76 Average speed per weekday

[66]:
```
group10 = data.groupby('weekday_num').speed.mean()
sns.pointplot(group10.index, group10.values)
plt.show()
```

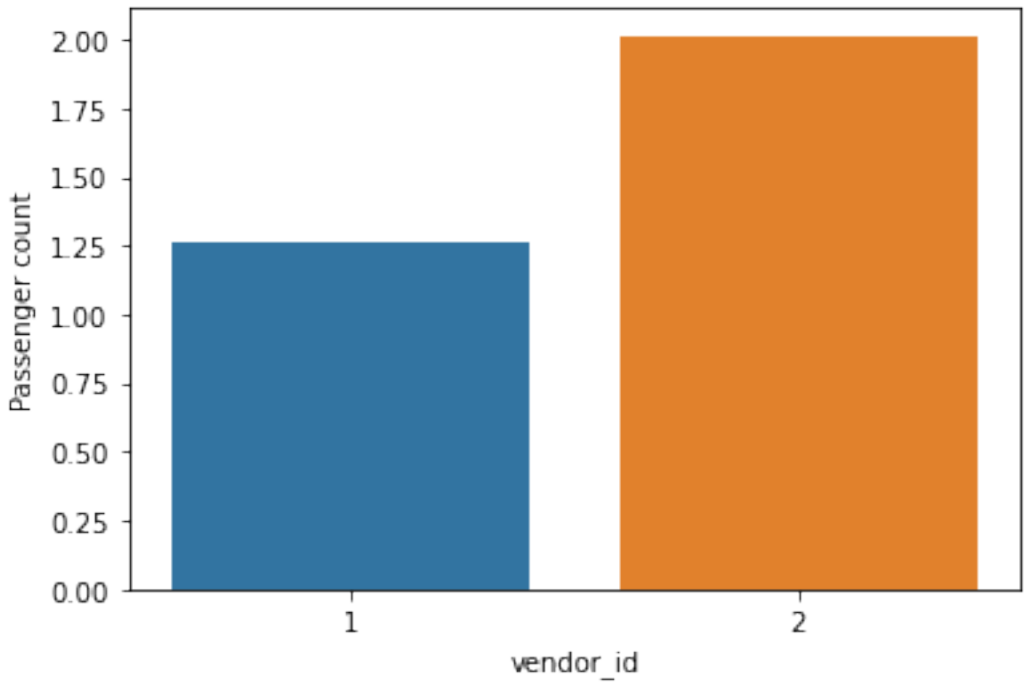


0.1.77 Observations

- Average taxi speed is higher on weekend as compared to the weekdays which is obvious when there is mostly rush of office goers and business owners.
- Even on monday the average taxi speed is shown higher which is quite surprising when it is one of the most busiest day after the weekend. There can be several possibility for such behaviour 1) Lot of customers who come back from outstation in early hours of Monday before 6 AM to attend office on time. 2) Early morning hours customers who come from the airports after vacation to attend office/business on time for the coming week.
- There could be some more reasons as well which only a local must be aware of.

0.1.78 Passenger count per vendor

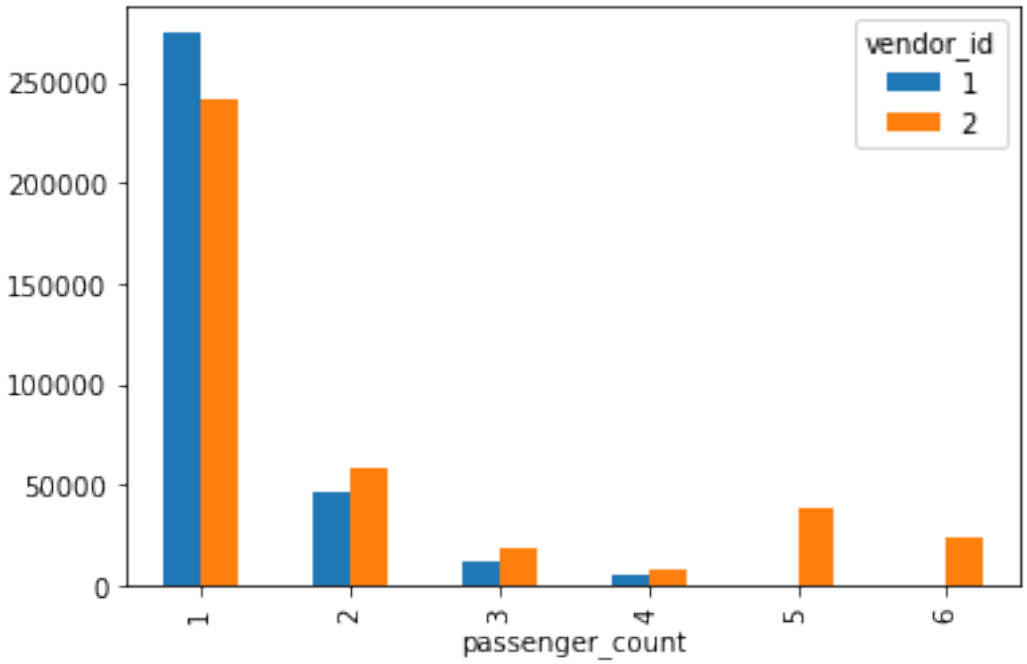
[67]:
```
group9 = data.groupby('vendor_id').passenger_count.mean()
sns.barplot(group9.index, group9.values)
plt.ylabel('Passenger count')
plt.show()
```



0.1.79 Observation

- Vendor 2 trips generally consist of 2 passengers as compared to the vendor 1 with 1 passenger.

[68]:
```
data.groupby('passenger_count').vendor_id.value_counts().
 ↪reset_index(name='count').pivot("passenger_count","vendor_id","count").
 ↪plot(kind='bar')
plt.show()
```



0.1.80 Observation

- Most of the big cars are served by the Vendor 2 including minivans because other than passenger 1, vendor 2 has majority in serving more than 1 passenger count and that explains it greater share of the market.

0.1.81 Map Visualization

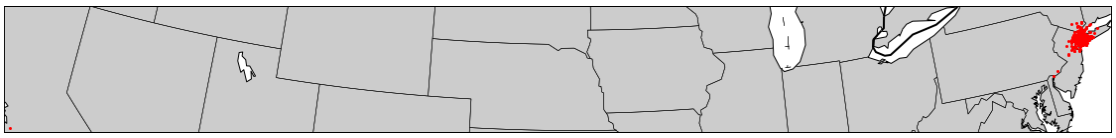
- Visualize the Taxi pickup locations by placing longitude and latitude marker on the MAP of the US. So that we can analyze below questions:
- Are all pickups constrained to NYC and it's surrounding areas?
- Is there any unusual location of the pickup?
- Are the latitude longitude constrained to the land area of the US and nowhere else?

[69]:
```
def map_marker(set):

    from mpl_toolkits.basemap import Basemap
    plt.figure(figsize = (20,20))

    lat_min = data["pickup_latitude"].min() - .2
    lat_max = data["pickup_latitude"].max() + .2
    lon_min = data["pickup_longitude"].min() - .2
    lon_max = data["pickup_longitude"].max() + .2
```

```
cent_lat = (lat_min + lat_max) / 2
cent_lon = (lon_min + lon_max) / 2


map = Basemap(llcrnrlon=lon_min,
              llcrnrlat=lat_min,
              urcrnrlon=lon_max,
              urcrnrlat=lat_max,
              resolution='l',
              projection='tmerc',
              lat_0 = cent_lat,
              lon_0 = cent_lon)


map.drawmapboundary()
map.drawcoastlines()
map.fillcontinents()
map.drawcountries(linewidth=2)
map.drawstates()



long = np.array(data["pickup_longitude"])
lat = np.array(data["pickup_latitude"])

x, y = map(long, lat)
map.plot(x, y,'ro', markersize=2, alpha=1)

plt.show()
```

0.1.82 Taxi pickup location

[70]:
```
map_marker(data)
```



0.1.83 Observation

- One unusual pickup from the CA state.
- There are quite a few pickup from the neighbouring state as well. Some are quite far and some very near to the NYC state

CA state pickup

[71]:
```
data[data.pickup_longitude == data.pickup_longitude.min()]
```

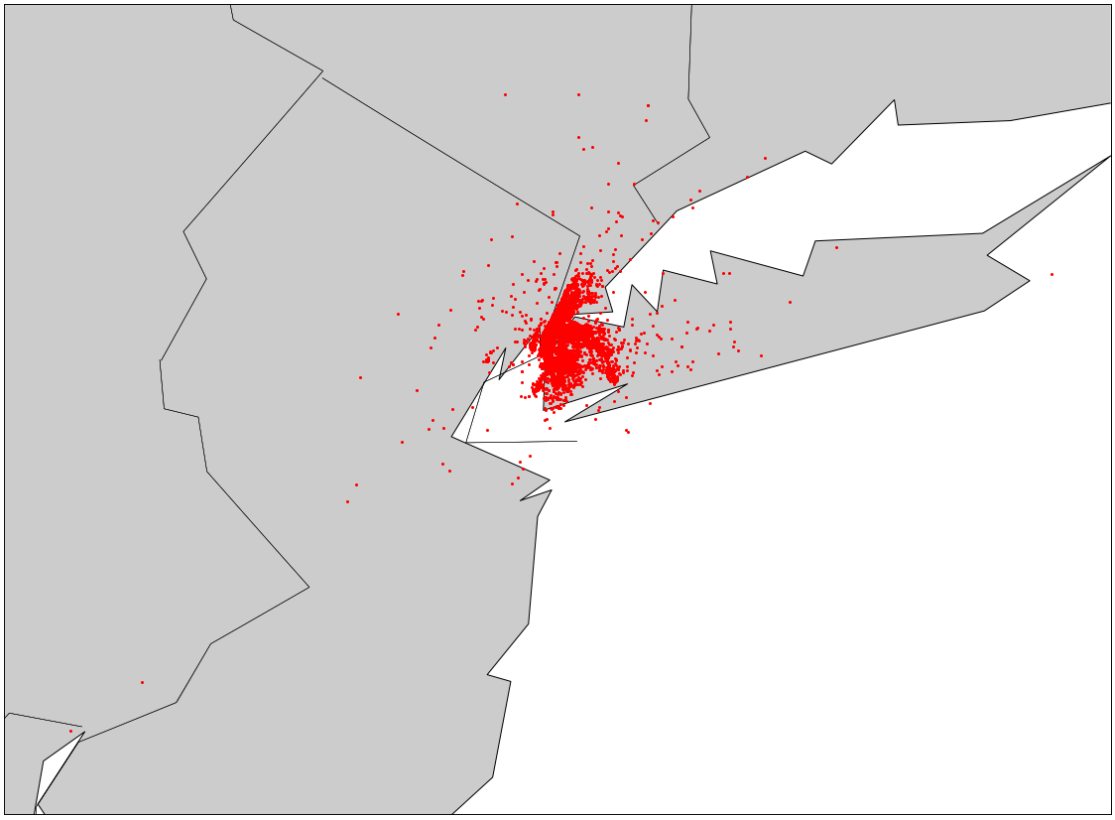
[71]:
```
               id  vendor_id     pickup_datetime    dropoff_datetime  \
421819  id2854272          2 2016-02-26 13:50:19 2016-02-26 13:58:38

        passenger_count  pickup_longitude  pickup_latitude  dropoff_longitude  \
421819                2       -121.933342        37.389381        -121.933304

        dropoff_latitude store_and_fwd_flag  …  pickup_hour_22  \
421819         37.389511                  N  …              0

       pickup_hour_23  passenger_count_1  passenger_count_2  \
421819              0                  0                  1

        passenger_count_3  passenger_count_4  passenger_count_5  \
421819                  0                  0                  0

        passenger_count_6  passenger_count_7  passenger_count_9
421819                  0                  0                  0

[1 rows x 61 columns]
```

0.1.84 Observation

- The trip duration is approx 8 minutes still the distance travelled is just in few meters.
- Moreover the Latitude and Longitude readings are same.
- These are the outliers and should be removed for the consistency of the model

[72]:
```
data = data[data.pickup_longitude != data.pickup_longitude.min()]
```

[73]:
```
map_marker(data)
```



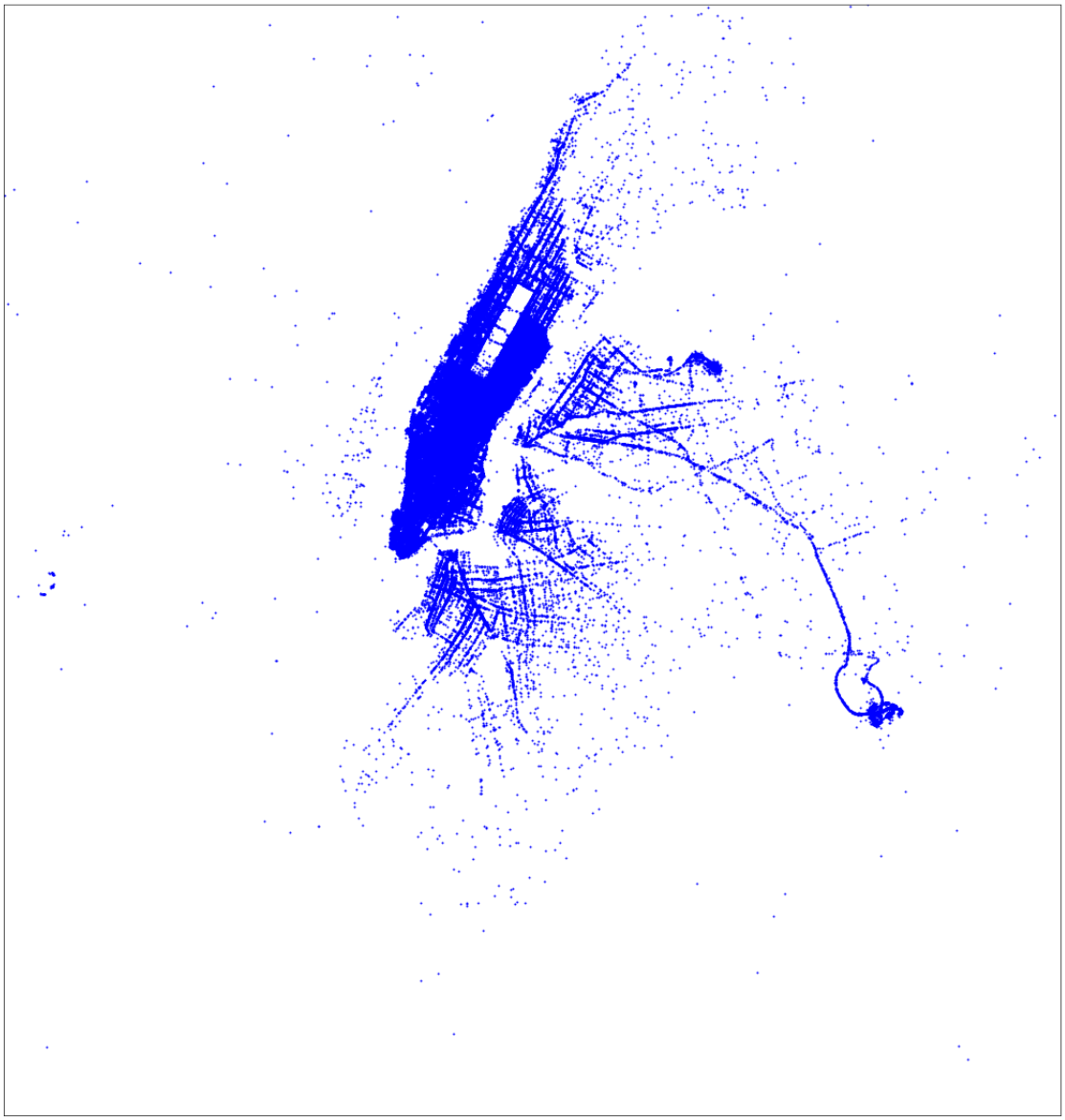
0.1.85 Observation

- There are quite a few pickups being shown off the NYC coast i.e. in the Atlantic ocean.
- Most of the pickups are being shown in and around NYC area.

0.1.86 NYC pickup locations

[74]:
```
plt.figure(figsize=(20,20))
from mpl_toolkits.basemap import Basemap

lat_min = 40.5
lat_max = 40.9
lon_min = -74.2
lon_max = -73.7

cent_lat = (lat_min + lat_max) / 2
cent_lon = (lon_min + lon_max) / 2

map = Basemap(llcrnrlon=lon_min,
              llcrnrlat=lat_min,
              urcrnrlon=lon_max,
```

```
              urcrnrlat=lat_max,
              resolution='l',
              projection='tmerc',
              lat_0 = cent_lat,
              lon_0 = cent_lon)

long = np.array(data["pickup_longitude"])
lat = np.array(data["pickup_latitude"])

x, y = map(long, lat)
map.plot(x, y,'bo', markersize=1, alpha=1)
plt.xticks()
plt.show()
```



0.1.87 Observations

- Most of the taxi pickups were done in the manhattan area as compared to the other areas in NYC.
- A long trail towards the airport shows that the airport is situated quite far from the Manhattan area.
- There must have been some long distance rides towards and from the airport.
- Similarly the average duration for the rides picked-up to or for the airport would have been longer.

0.1.88 NYC drop off location

[75]:
```
plt.figure(figsize=(20,20))

lat_min = 40.5
lat_max = 40.9
lon_min = -74.2
lon_max = -73.7

cent_lat = (lat_min + lat_max) / 2
cent_lon = (lon_min + lon_max) / 2

map = Basemap(llcrnrlon=lon_min,
              llcrnrlat=lat_min,
              urcrnrlon=lon_max,
              urcrnrlat=lat_max,
              resolution='l',
              projection='tmerc',
              lat_0 = cent_lat,
              lon_0 = cent_lon)

long = np.array(data["dropoff_longitude"])
lat = np.array(data["dropoff_latitude"])

x, y = map(long, lat)
map.plot(x, y,'go', markersize=1, alpha=1)
plt.xticks()
plt.show()
```

0.1.89 Observation

- Dropoff's are much more distributed around the NYC area where still most of the dropoff's were done in the Manhattan.

0.1.90 Question:

- Does that mean there were more dropoff's than the pickup's?

0.1.91 Idea:

- Though the dropoffs seems to be larger in number than the pickups. But for each pickup we have a associated dropoffs in the dataset. It's just that the pickups were majorly concentrated in the Manhattan area.

[76]:
```
data
```

[76]:
```
               id  vendor_id     pickup_datetime    dropoff_datetime  \
0       id1080784          2 2016-02-29 16:40:21 2016-02-29 16:47:01
1       id0889885          1 2016-03-11 23:35:37 2016-03-11 23:53:57
2       id0857912          2 2016-02-21 17:59:33 2016-02-21 18:26:48
3       id3744273          2 2016-01-05 09:44:31 2016-01-05 10:03:32
4       id0232939          1 2016-02-17 06:42:23 2016-02-17 06:56:31
…             …        …                   …                   …
729317  id3905982          2 2016-05-21 13:29:38 2016-05-21 13:34:34
729318  id0102861          1 2016-02-22 00:43:11 2016-02-22 00:48:26
729319  id0439699          1 2016-04-15 18:56:48 2016-04-15 19:08:01
729320  id2078912          1 2016-06-19 09:50:47 2016-06-19 09:58:14
729321  id1053441          2 2016-01-01 17:24:16 2016-01-01 17:44:40

        passenger_count  pickup_longitude  pickup_latitude  dropoff_longitude  \
0                     1        -73.953918        40.778873         -73.963875
1                     2        -73.988312        40.731743         -73.994751
2                     2        -73.997314        40.721458         -73.948029
3                     6        -73.961670        40.759720         -73.956779
4                     1        -74.017120        40.708469         -73.988182
…                   …                 …                …                  …
729317                2        -73.965919        40.789780         -73.952637
729318                1        -73.996666        40.737434         -74.001320
729319                1        -73.997849        40.761696         -74.001488
729320                1        -74.006706        40.708244         -74.013550
729321                4        -74.003342        40.743839         -73.945847

        dropoff_latitude store_and_fwd_flag  …  pickup_hour_22  \
0               40.771164                  N  …               0
1               40.694931                  N  …               0
2               40.774918                  N  …               0
3               40.780628                  N  …               0
4               40.740631                  N  …               0
…                     …                …  …  …             …
729317          40.789181                  N  …               0
729318          40.731911                  N  …               0
729319          40.741207                  N  …               0
729320          40.713814                  N  …               0
729321          40.712841                  N  …               0

       pickup_hour_23  passenger_count_1  passenger_count_2  \
```

```
0                   0                   1                   0
1                   1                   0                   1
2                   0                   0                   1
3                   0                   0                   0
4                   0                   1                   0
…                 …                   …                   …
729317              0                   0                   1
729318              0                   1                   0
729319              0                   1                   0
729320              0                   1                   0
729321              0                   0                   0

        passenger_count_3  passenger_count_4  passenger_count_5  \
0                       0                  0                  0
1                       0                  0                  0
2                       0                  0                  0
3                       0                  0                  0
4                       0                  0                  0
…                     …                  …                  …
729317                  0                  0                  0
729318                  0                  0                  0
729319                  0                  0                  0
729320                  0                  0                  0
729321                  0                  1                  0

        passenger_count_6  passenger_count_7  passenger_count_9
0                       0                  0                  0
1                       0                  0                  0
2                       0                  0                  0
3                       1                  0                  0
4                       0                  0                  0
…                     …                  …                  …
729317                  0                  0                  0
729318                  0                  0                  0
729319                  0                  0                  0
729320                  0                  0                  0
729321                  0                  0                  0

[726928 rows x 61 columns]
```

[77]:
```
data['vendor_id_2'].unique()
```

[77]:
```
array([1, 0], dtype=uint8)
```

[78]:
```
list(zip( range(0,len(data.columns)),data.columns))
```

```
[78]: [(0, 'id'),
       (1, 'vendor_id'),
       (2, 'pickup_datetime'),
       (3, 'dropoff_datetime'),
       (4, 'passenger_count'),
       (5, 'pickup_longitude'),
       (6, 'pickup_latitude'),
       (7, 'dropoff_longitude'),
       (8, 'dropoff_latitude'),
       (9, 'store_and_fwd_flag'),
       (10, 'trip_duration'),
       (11, 'weekday'),
       (12, 'month'),
       (13, 'weekday_num'),
       (14, 'pick_up_hour'),
       (15, 'Distance'),
       (16, 'speed'),
       (17, 'flag_Y'),
       (18, 'vendor_id_2'),
       (19, 'month_2'),
       (20, 'month_3'),
       (21, 'month_4'),
       (22, 'month_5'),
       (23, 'month_6'),
       (24, 'weekday_num_1'),
       (25, 'weekday_num_2'),
       (26, 'weekday_num_3'),
       (27, 'weekday_num_4'),
       (28, 'weekday_num_5'),
       (29, 'weekday_num_6'),
       (30, 'pickup_hour_1'),
       (31, 'pickup_hour_2'),
       (32, 'pickup_hour_3'),
       (33, 'pickup_hour_4'),
       (34, 'pickup_hour_5'),
       (35, 'pickup_hour_6'),
       (36, 'pickup_hour_7'),
       (37, 'pickup_hour_8'),
       (38, 'pickup_hour_9'),
       (39, 'pickup_hour_10'),
       (40, 'pickup_hour_11'),
       (41, 'pickup_hour_12'),
       (42, 'pickup_hour_13'),
       (43, 'pickup_hour_14'),
       (44, 'pickup_hour_15'),
       (45, 'pickup_hour_16'),
       (46, 'pickup_hour_17'),
```

```
 (47, 'pickup_hour_18'),
 (48, 'pickup_hour_19'),
 (49, 'pickup_hour_20'),
 (50, 'pickup_hour_21'),
 (51, 'pickup_hour_22'),
 (52, 'pickup_hour_23'),
 (53, 'passenger_count_1'),
 (54, 'passenger_count_2'),
 (55, 'passenger_count_3'),
 (56, 'passenger_count_4'),
 (57, 'passenger_count_5'),
 (58, 'passenger_count_6'),
 (59, 'passenger_count_7'),
 (60, 'passenger_count_9')]
```

0.1.92 Creating Benchmark model

[79]:
```
from sklearn.utils import shuffle

data = shuffle(data, random_state = 42)

div = int(data.shape[0]/4)

train = data.loc[:3*div+1,:]
test = data.loc[3*div+1:]
```

[80]:
```
train.head()
```

[80]:

	id	vendor_id	pickup_datetime	dropoff_datetime	\
128974	id0192028	1	2016-01-25 21:24:25	2016-01-25 21:40:20	
45334	id3265312	2	2016-01-15 06:36:08	2016-01-15 06:45:36	
321158	id0237424	2	2016-01-03 02:05:35	2016-01-03 02:19:08	
202673	id0489855	1	2016-02-19 23:04:44	2016-02-19 23:11:22	
408616	id3975782	1	2016-06-13 06:32:39	2016-06-13 06:35:05	

	passenger_count	pickup_longitude	pickup_latitude	dropoff_longitude	\
128974	1	-74.009605	40.715126	-73.987381	
45334	1	-73.985847	40.763874	-73.952675	
321158	5	-74.002525	40.749935	-73.996269	
202673	1	-74.003922	40.737373	-74.006622	
408616	1	-73.990120	40.757122	-73.979187	

	dropoff_latitude	store_and_fwd_flag	…	pickup_hour_22	\
128974	40.739616	N	…	0	
45334	40.789043	N	…	0	
321158	40.720226	N	…	0	
202673	40.744141	N	…	0	

```
408616         40.756207                  N  …                0

       pickup_hour_23  passenger_count_1  passenger_count_2  \
128974              0                  1                  0
45334               0                  1                  0
321158              0                  0                  0
202673              1                  1                  0
408616              0                  1                  0

        passenger_count_3  passenger_count_4  passenger_count_5  \
128974                  0                  0                  0
45334                   0                  0                  0
321158                  0                  0                  1
202673                  0                  0                  0
408616                  0                  0                  0

        passenger_count_6  passenger_count_7  passenger_count_9
128974                  0                  0                  0
45334                   0                  0                  0
321158                  0                  0                  0
202673                  0                  0                  0
408616                  0                  0                  0

[5 rows x 61 columns]
```

[81]:
```
test.head()
```

[81]:
```
               id  vendor_id     pickup_datetime    dropoff_datetime  \
545197  id0001618          2 2016-04-05 22:58:14 2016-04-05 23:07:08
477367  id1076081          1 2016-02-20 19:28:55 2016-02-20 20:12:09
490112  id2622397          2 2016-04-18 22:56:56 2016-04-18 23:01:19
632259  id0083884          1 2016-03-23 18:29:03 2016-03-23 18:36:27
259210  id3305446          1 2016-06-28 19:41:57 2016-06-28 20:00:53

        passenger_count  pickup_longitude  pickup_latitude  dropoff_longitude  \
545197                1        -73.989052        40.721561         -74.005669
477367                1        -73.993111        40.727898         -73.933746
490112                1        -73.987152        40.756119         -73.978630
632259                2        -73.959091        40.709164         -73.963051
259210                2        -73.995300        40.717667         -73.983711

        dropoff_latitude store_and_fwd_flag  …  pickup_hour_22  \
545197         40.709179                  N  …               1
477367         40.846863                  N  …               0
490112         40.759521                  N  …               1
632259         40.719185                  N  …               0
259210         40.756245                  N  …               0
```

```
        pickup_hour_23  passenger_count_1  passenger_count_2  \
545197               0                  1                  0
477367               0                  1                  0
490112               0                  1                  0
632259               0                  0                  1
259210               0                  0                  1

        passenger_count_3  passenger_count_4  passenger_count_5  \
545197                  0                  0                  0
477367                  0                  0                  0
490112                  0                  0                  0
632259                  0                  0                  0
259210                  0                  0                  0

        passenger_count_6  passenger_count_7  passenger_count_9
545197                  0                  0                  0
477367                  0                  0                  0
490112                  0                  0                  0
632259                  0                  0                  0
259210                  0                  0                  0

[5 rows x 61 columns]
```

0.1.93 Simple mean(Mean of Trip duration)

[82]:
```
test['simple_mean'] = train['trip_duration'].mean()
```

[83]:
```
from sklearn.metrics import mean_absolute_error as mae

simple_mean_error = mae(test['vendor_id'] , test['simple_mean'])
simple_mean_error
```

[83]: 925.5460621739236

[84]:
```
simple_mean_error = mae(test['passenger_count'] , test['simple_mean'])
simple_mean_error
```

[84]: 925.4185414302777

[85]:
```
simple_mean_error = mae(test['Distance'] , test['simple_mean'])
simple_mean_error
```

[85]: 923.6257561766556

[86]:
```
simple_mean_error = mae(test['speed'] , test['simple_mean'])
simple_mean_error
```

[86]: 912.6617823824865

0.1.94 Mean trip duration with respect to passenger_count

[87]:
```
out_type = pd.pivot_table(train, values='trip_duration', index =␣
↪['passenger_count'], aggfunc=np.mean)
out_type
```

[87]:

passenger_count	trip_duration
1	895.529689
2	987.054239
3	1027.874453
4	1007.166781
5	1020.668969
6	1025.422071

[88]:
```
test['pc_mean'] = 0

for i in train['passenger_count'].unique():
    test['pc_mean'][test['passenger_count'] == str(i)] =␣
↪train['trip_duration'][train['passenger_count'] == str(i)].mean()
```

calculating mean absolute error

[89]:
```
out_type_error = mae(test['trip_duration'] , test['pc_mean'] )
out_type_error
```

[89]: 936.5109500236927

0.1.95 Mean trip duration with respect to vendor_id

[90]:
```
out_type = pd.pivot_table(train, values='trip_duration', index = ['vendor_id'],␣
↪aggfunc=np.mean)
out_type
```

[90]:

vendor_id	trip_duration
1	826.228271
2	1014.917320

[91]:
```
test['vid_mean'] = 0

for i in train['vendor_id'].unique():
    test['vid_mean'][test['vendor_id'] == str(i)] =␣
↪train['trip_duration'][train['vendor_id'] == str(i)].mean()
```

calculating mean absolute error

[92]:
```
out_type_error = mae(test['trip_duration'] , test['vid_mean'] )
out_type_error
```

[92]:
```
936.5109500236927
```

0.1.96 Mean trip duration with respect to weekday

[93]:
```
out_type = pd.pivot_table(train, values='trip_duration', index = ['weekday'],␣
↪aggfunc=np.mean)
out_type
```

[93]:

weekday	trip_duration
Friday	964.447934
Monday	858.861128
Saturday	893.315898
Sunday	896.699168
Thursday	975.070615
Tuesday	927.957861
Wednesday	961.103568

[94]:
```
test['weekday_mean'] = 0

for i in train['weekday'].unique():
    test['weekday_mean'][test['weekday'] == str(i)] =␣
↪train['trip_duration'][train['weekday'] == str(i)].mean()
```

calculating mean absolute error

[95]:
```
out_type_error = mae(test['trip_duration'] , test['weekday_mean'] )
out_type_error
```

[95]:
```
591.3940067538442
```

0.1.97 Mean trip duration with respect to store and forward flag

[96]:
```
out_type = pd.pivot_table(train, values='trip_duration', index =␣
↪['store_and_fwd_flag'], aggfunc=np.mean)
out_type
```

[96]:

store_and_fwd_flag	trip_duration
N	926.220358
Y	1079.681853

[97]:
```
test['sf_mean'] = 0

for i in train['store_and_fwd_flag'].unique():
```

```
    test['sf_mean'][test['store_and_fwd_flag'] == str(i)] =␣
↪train['trip_duration'][train['store_and_fwd_flag'] == str(i)].mean()
```

calculating mean absolute error

[98]:
```
out_type_error = mae(test['trip_duration'] , test['sf_mean'] )
out_type_error
```

[98]: 592.456765391404

0.1.98 Mean trip duration with respect to month

[99]:
```
out_type = pd.pivot_table(train, values='trip_duration', index = ['month'],␣
↪aggfunc=np.mean)
out_type
```

[99]:
```
       trip_duration
month
1         892.200560
2         889.693945
3         897.695070
4         945.995096
5         970.045297
6         965.979263
```

[100]:
```
test['month_mean'] = 0

for i in train['month'].unique():
    test['month_mean'][test['month'] == str(i)] =␣
↪train['trip_duration'][train['month'] == str(i)].mean()
```

calculating mean absolute error

[101]:
```
out_type_error = mae(test['trip_duration'] , test['month_mean'] )
out_type_error
```

[101]: 936.5109500236927

0.1.99 Mean trip_duration with respect to both vendor_id and passenger_count

[102]:
```
combo = pd.pivot_table(train, values = 'trip_duration', index =␣
↪['vendor_id','passenger_count'], aggfunc = np.mean)
combo
```

[102]:
```
                          trip_duration
vendor_id passenger_count
1         1                  801.184459
          2                  928.335185
          3                  941.943985
```

```
           4                992.639560
           5                729.444444
           6               1157.869565
2          1               1003.377783
           2               1032.407864
           3               1082.023648
           4               1017.458544
           5               1022.167429
           6               1025.022557
```

[103]:
```
test['Super_mean'] = 0

s2 = 'vendor_id'
s1 = 'passenger_count'

for i in test[s1].unique():
    for j in test[s2].unique():
        test['Super_mean'][(test[s1] == i) & (test[s2]==str(j))] =␣
↪train['trip_duration'][(train[s1] == i) & (train[s2]==str(j))].mean()
```

calculating mean absolute error

[104]:
```
super_mean_error = mae(test['trip_duration'] , test['Super_mean'] )
super_mean_error
```

[104]:
```
936.5109500236927
```

0.1.100 Mean trip_duration with respect to both weekday and store and forward flag

[105]:
```
combo = pd.pivot_table(train, values = 'trip_duration', index =␣
↪['weekday','store_and_fwd_flag'], aggfunc = np.mean)
combo
```

[105]:
```
                              trip_duration
weekday   store_and_fwd_flag
Friday    N                     963.296795
          Y                    1154.807512
Monday    N                     857.405713
          Y                    1111.829412
Saturday  N                     893.502314
          Y                     853.533742
Sunday    N                     896.497868
          Y                     934.870370
Thursday  N                     973.458200
          Y                    1246.794118
Tuesday   N                     927.067631
          Y                    1081.572973
Wednesday N                     960.261480
```

```
Y                    1101.974747
```

[106]:
```
test['Super_mean'] = 0
s2 = 'weekday'
s1 = 'store_and_fwd_flag'

for i in test[s1].unique():
    for j in test[s2].unique():
        test['Super_mean'][((test[s1] == str(i)) & (test[s2]==str(j)))] =␣
↪train['trip_duration'][(train[s1] == str(i)) & (train[s2]==str(j))].mean()
```

calculating mean absolute error

[107]:
```
super_mean_error = mae(test['trip_duration'] , test['Super_mean'] )
super_mean_error
```

[107]:
```
591.240019924098
```

0.1.101 Mean trip_duration with respect to both weekday and month

[108]:
```
combo = pd.pivot_table(train, values = 'trip_duration', index =␣
↪['weekday','month'], aggfunc = np.mean)
combo
```

[108]:

		trip_duration
weekday	month	
Friday	1	920.578811
	2	908.180140
	3	886.245230
	4	1024.302175
	5	1036.493422
	6	1010.064229
Monday	1	842.743100
	2	880.982032
	3	832.726255
	4	823.643323
	5	896.371409
	6	862.428828
Saturday	1	873.565194
	2	840.543251
	3	823.821033
	4	928.823456
	5	892.395547
	6	999.799847
Sunday	1	901.544301
	2	884.134039
	3	921.731936
	4	898.260967

```
             5          877.009423
             6          902.123693
Thursday     1          887.360187
             2          916.590473
             3          962.810317
             4          994.794881
             5         1076.014466
             6         1007.850000
Tuesday      1          955.030612
             2          875.893162
             3          887.527432
             4          927.173452
             5          986.239348
             6          933.111341
Wednesday    1          853.885901
             2          922.316157
             3          945.552758
             4          986.426429
             5         1044.123906
             6         1004.040353
```

[109]:
```
test['Super_mean'] = 0
s2 = 'weekday'
s1 = 'month'

for i in test[s1].unique():
    for j in test[s2].unique():
        test['Super_mean'][(test[s1] == i) & (test[s2]==str(j))] =␣
↪train['trip_duration'][(train[s1] == i) & (train[s2]==str(j))].mean()
```

calculating mean absolute error

[110]:
```
super_mean_error = mae(test['trip_duration'] , test['Super_mean'] )
super_mean_error
```

[110]:
```
590.9150616833672
```

0.1.102 Conclusion

- Chosen Mean Absolute Error as Evaluation metrics since it gives more accurate error than other evaluation metrics
- It can be concluded that mean absolute error for vendor_id, passenger_count, distance, speed are almost same as compared to other evaluation metrics (around 910-925)
- It can be seen that mean absolute error for vendor_id, passenger_count and month with respect to mean trip duration is around 936, also for weekday and store_and_fwd_flag is around 592.

Segregating dependent and independent variable

[111]:
```
y = data.iloc[:,10].values
x = data.iloc[:,range(15,61)].values
x.shape, y.shape
```

[111]:
```
((726928, 46), (726928,))
```

[112]:
```
y = data.iloc[:,10]
x = data.iloc[:,range(15,61)]
x.shape, y.shape
```

[112]:
```
((726928, 46), (726928,))
```

Idea:

- Duration variable assigned to Y because that is the dependent variable.
- Features such as id, timestamp and weekday were not assigned to X array because they are of type object. And we need an array of float data type.

Splitting the data into train set and test set

[113]:
```
from sklearn.model_selection import train_test_split
train_x,test_x,train_y,test_y = train_test_split(x,y, random_state = 56)
```

0.1.103 Implementing Linear Regression

[114]:
```
from sklearn.linear_model import LinearRegression as LR
from sklearn.metrics import mean_absolute_error as mae
```

[115]:
```
lr = LR(normalize = True)

lr.fit(train_x, train_y)
```

[115]:
```
LinearRegression(normalize=True)
```

Predicting over the Train Set and calculating error

[116]:
```
train_predict = lr.predict(train_x)
k = mae(train_predict, train_y)
print('Training Mean Absolute Error', k )
```

```
Training Mean Absolute Error 391.50831757945673
```

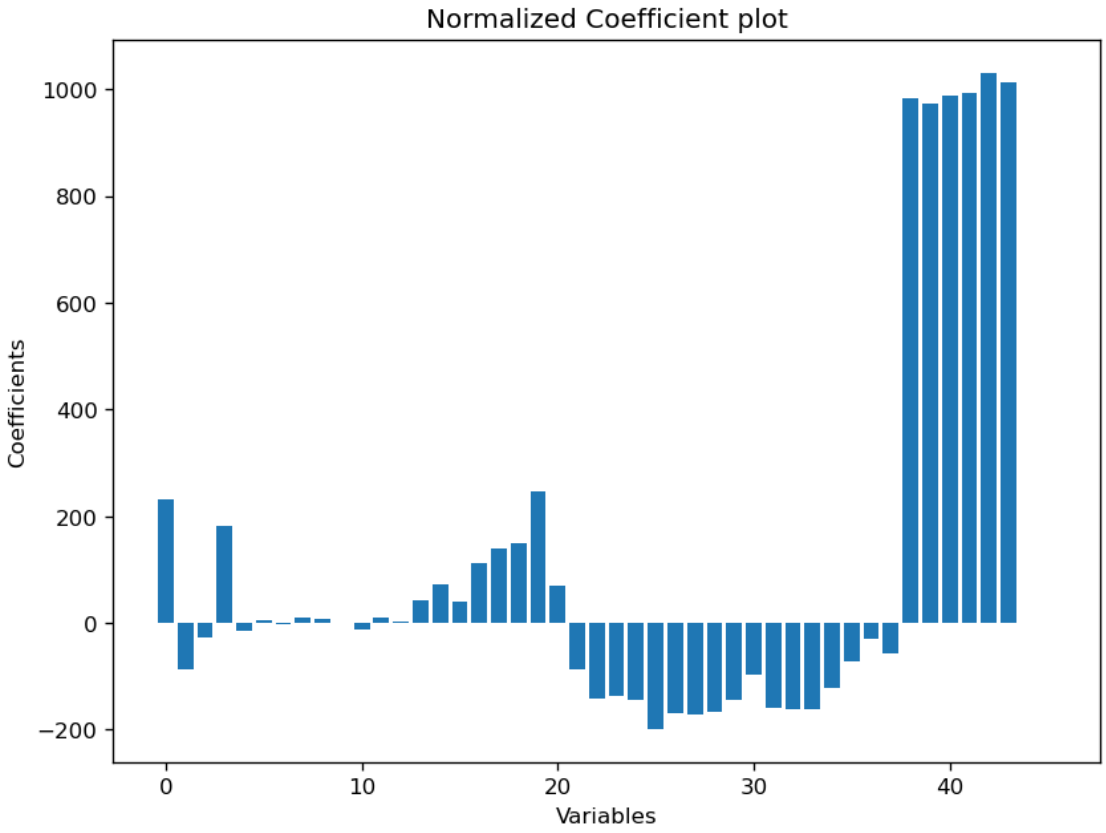
Predicting over the Test Set and calculating error

[117]:
```
test_predict = lr.predict(test_x)
k = mae(test_predict, test_y)
print('Test Mean Absolute Error    ', k )
```

```
Test Mean Absolute Error     379.4247619717158
```

[118]:
```
plt.figure(figsize=(8, 6), dpi=120, facecolor='w', edgecolor='b')
x = range(len(train_x.columns))
y = lr.coef_
plt.bar( x, y )
plt.xlabel( "Variables")
plt.ylabel('Coefficients')
plt.title('Normalized Coefficient plot')
```

[118]:
```
Text(0.5, 1.0, 'Normalized Coefficient plot')
```



Creating new subsets of data

[119]:
```
y = data.iloc[:,10]
x = data.iloc[:,range(15,61)]
x.shape, y.shape
```

[119]:
```
((726928, 46), (726928,))
```

Arranging coefficients with features

[120]:
```
Coefficients = pd.DataFrame({
    'Variable'    : x.columns,
```

```
    'coefficient' : lr.coef_
})
Coefficients.head()
```

[120]:
```
       Variable  coefficient
0     Distance   230.611862
1        speed   -87.888768
2       flag_Y   -26.885824
3  vendor_id_2   180.767959
4      month_2   -13.993392
```

Choosing variables with sigificance greater than 0.5 (Filtering Significant Features)

[121]:
```
sig_var = Coefficients[Coefficients.coefficient > 0.5]
```

Extracting the significant subset do independent Variables

[122]:
```
subset = data[sig_var['Variable'].values]
subset.head()
```

[122]:
```
        Distance  vendor_id_2  month_3  month_5  month_6  weekday_num_3  \
128974  3.305013            0        0        0        0              0
45334   3.954126            1        0        0        0              0
321158  3.345269            1        0        0        0              0
202673  0.786137            0        0        0        0              0
408616  0.926480            0        0        0        1              0

        weekday_num_4  weekday_num_5  weekday_num_6  pickup_hour_1  …  \
128974              0              0              0              0  …
45334               1              0              0              0  …
321158              0              0              1              0  …
202673              1              0              0              0  …
408616              0              0              0              0  …

        pickup_hour_3  pickup_hour_4  pickup_hour_5  pickup_hour_6  \
128974              0              0              0              0
45334               0              0              0              1
321158              0              0              0              0
202673              0              0              0              0
408616              0              0              0              1

        passenger_count_1  passenger_count_2  passenger_count_3  \
128974                  1                  0                  0
45334                   1                  0                  0
321158                  0                  0                  0
202673                  1                  0                  0
408616                  1                  0                  0

        passenger_count_4  passenger_count_5  passenger_count_6
```

128974	0	0	0
45334	0	0	0
321158	0	1	0
202673	0	0	0
408616	0	0	0

[5 rows x 21 columns]

Splitting the data into train set and the test set

[123]:
```
from sklearn.model_selection import train_test_split
train_x,test_x,train_y,test_y = train_test_split(subset, y , random_state = 56)
```

Implementing Linear Regression

[124]:
```
from sklearn.linear_model import LinearRegression as LR
from sklearn.metrics import mean_absolute_error as mae
```

Training Model

[125]:
```
lr = LR(normalize = True)

lr.fit(train_x, train_y)
```

[125]:
```
LinearRegression(normalize=True)
```

Predicting over the train set

[126]:
```
train_predict = lr.predict(train_x)
k = mae(train_predict, train_y)
print('Training Mean Absolute Error', k )
```

```
Training Mean Absolute Error 411.6020129040701
```

Predicting over the test set

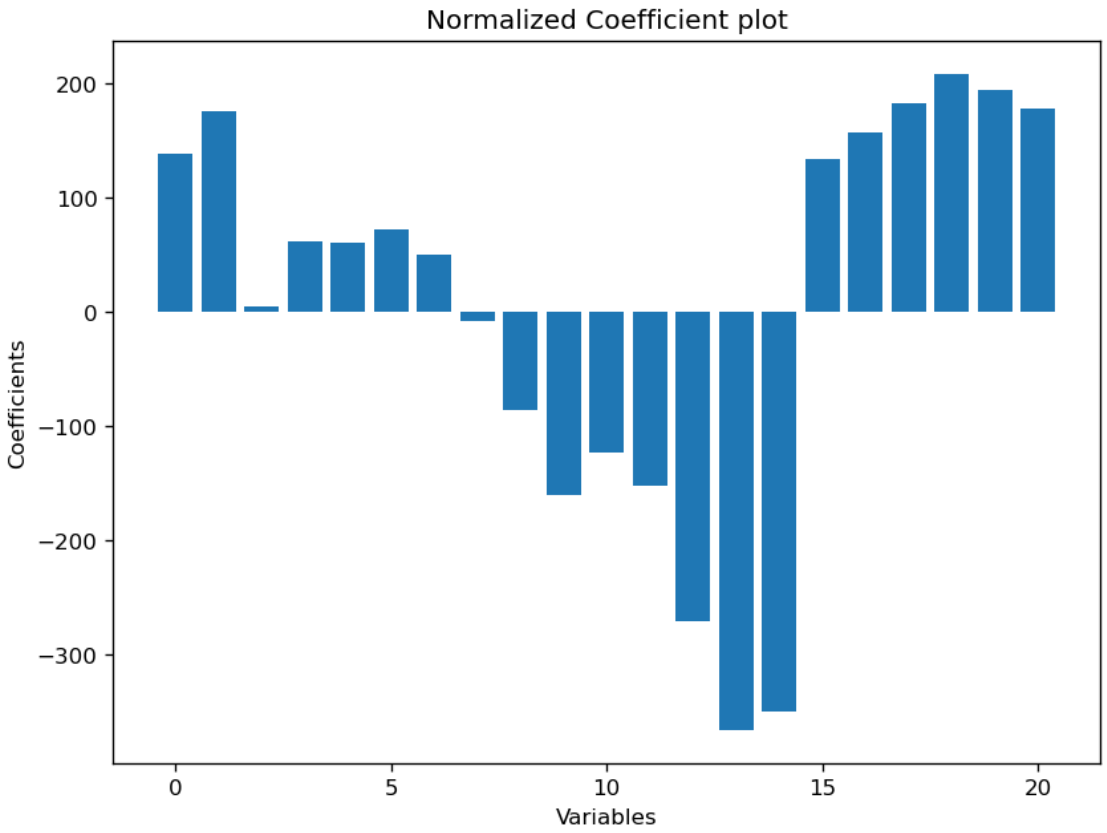
[127]:
```
test_predict = lr.predict(test_x)
k = mae(test_predict, test_y)
print('Test Mean Absolute Error    ', k )
```

```
Test Mean Absolute Error     399.80427755071815
```

Plotting the Coefficient

[128]:
```
plt.figure(figsize=(8, 6), dpi=120, facecolor='w', edgecolor='b')
x = range(len(train_x.columns))
y = lr.coef_
plt.bar( x, y )
plt.xlabel( "Variables")
plt.ylabel('Coefficients')
plt.title('Normalized Coefficient plot')
```

[128]:
```
Text(0.5, 1.0, 'Normalized Coefficient plot')
```



[129]:
```
from sklearn.linear_model import Lasso
lasso = Lasso(alpha=1.5)
```

[130]:
```
lasso.fit(train_x,train_y)
```

[130]:
```
Lasso(alpha=1.5)
```

[131]:
```
y_pred = lasso.predict(test_x)
```

[132]:
```
y_pred
```

[132]:
```
array([885.53576027, 886.7881058 , 621.56898523, …, 484.3922918 ,
       795.37053331, 595.50084388])
```

[133]:
```
print("slope: %.2f" % lasso.coef_[0])
```

```
slope: 137.27
```

[134]:
```
print("Intercept: %.2f" % lasso.intercept_)
```

```
Intercept: 389.68
```

[135]:
```
from sklearn.metrics import mean_absolute_error as mae
```

[136]:
```
mae(test_y,y_pred)
```

[136]:
```
400.6575296326174
```

0.1.104 Generally the line equation is y = mx + c where y = target variable, x = independent variable, m = slope, c = intercept

On any changes in slope the target variable is increasing m times with the independent variable. Here the slope is 137.27 means , there will be increasing in the duration of trip by 137 times on increasing one unit on the other variables. Generally when independent variable is zero then target variable is dependent on intercepts (i.e., when x = 0, y = c (intercept))

0.1.105 Decision Tree Implementation

Segregating Dependent and Independent Variables

[137]:
```
y = data.iloc[:,10]
x = data.iloc[:,range(15,61)]
x.shape, y.shape
```

[137]:
```
((726928, 46), (726928,))
```

[138]:
```
from sklearn.tree import DecisionTreeRegressor
```

[139]:
```
from sklearn.model_selection import train_test_split
```

[140]:
```
from sklearn.metrics import mean_absolute_error as mae
```

[141]:
```
from sklearn.model_selection import train_test_split
train_x,test_x,train_y,test_y = train_test_split(subset, y , random_state = 56)
```

[142]:
```
dt=DecisionTreeRegressor(max_depth = 10, min_samples_leaf = 25, random_state =␣
↪56)
```

[143]:
```
dt.fit(train_x,train_y)
```

[143]:
```
DecisionTreeRegressor(max_depth=10, min_samples_leaf=25, random_state=56)
```

[144]:
```
y_pred = dt.predict(test_x)
```

[145]:
```
mae_dt = mae(test_y, y_pred)
```

[146]:
```
print(mae_dt)
```

```
391.50162871523764
```

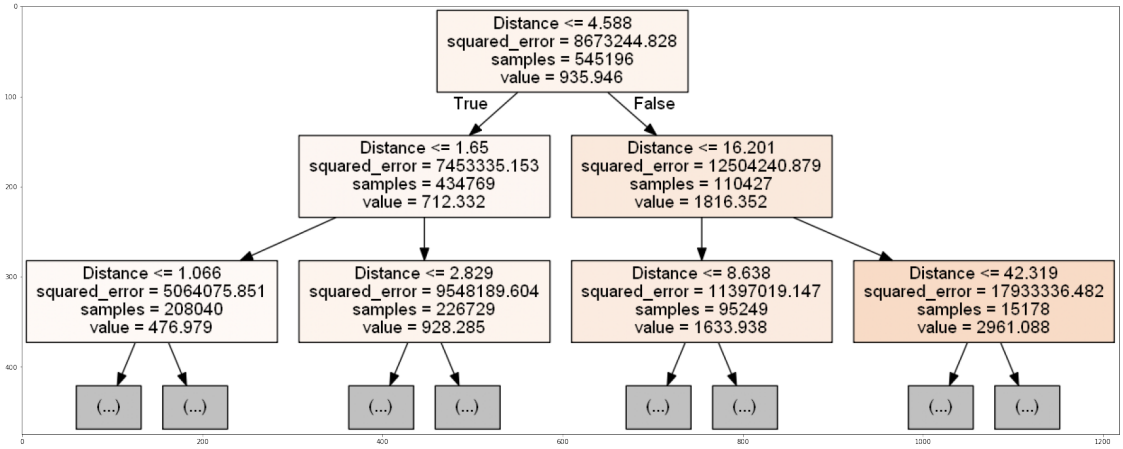
[147]:
```
from sklearn import tree
```

[148]:
```
decision_tree = tree.export_graphviz(dt,out_file='tree.
↪dot',feature_names=train_x.columns,max_depth=2,filled=True)
```

[149]:
```
!dot -Tpng tree.dot -o tree.png
```

[150]:
```
image = plt.imread('tree.png')
plt.figure(figsize=(30,30))
plt.imshow(image)
```

[150]:
```
<matplotlib.image.AxesImage at 0x251a3e3cfa0>
```



0.1.106 Conclusion:

- The splitting was done based on the squared error in decision tree regressoe

0.1.107 KNN Implementation

Segregating variables: Independent and Dependent Variables

[151]:
```
y = data.iloc[:,10]
x = data.iloc[:,range(15,61)]
x.shape, y.shape
```

[151]:
```
((726928, 46), (726928,))
```

Scaling the data (Using MinMax Scaler)

[152]:
```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
x_scaled = scaler.fit_transform(x)
```

[153]:
```
x = pd.DataFrame(x_scaled, columns = x.columns)
```

[154]:
```
x.head()
```

[154]:
```
   Distance     speed  flag_Y  vendor_id_2  month_2  month_3  month_4  \
0  0.019101  0.122096     0.0          0.0      0.0      0.0      0.0
1  0.022852  0.245603     0.0          1.0      0.0      0.0      0.0
2  0.019334  0.145168     0.0          1.0      0.0      0.0      0.0
3  0.004543  0.069686     0.0          0.0      1.0      0.0      0.0
4  0.005354  0.223879     0.0          0.0      0.0      0.0      0.0

   month_5  month_6  weekday_num_1  …  pickup_hour_22  pickup_hour_23  \
0      0.0      0.0            0.0  …             0.0             0.0
1      0.0      0.0            0.0  …             0.0             0.0
2      0.0      0.0            0.0  …             0.0             0.0
3      0.0      0.0            0.0  …             0.0             1.0
4      0.0      1.0            0.0  …             0.0             0.0

   passenger_count_1  passenger_count_2  passenger_count_3  passenger_count_4  \
0                1.0                0.0                0.0                0.0
1                1.0                0.0                0.0                0.0
2                0.0                0.0                0.0                0.0
3                1.0                0.0                0.0                0.0
4                1.0                0.0                0.0                0.0

   passenger_count_5  passenger_count_6  passenger_count_7  passenger_count_9
0                0.0                0.0                0.0                0.0
1                0.0                0.0                0.0                0.0
2                1.0                0.0                0.0                0.0
3                0.0                0.0                0.0                0.0
4                0.0                0.0                0.0                0.0

[5 rows x 46 columns]
```

[155]:
```
from sklearn.model_selection import train_test_split
train_x,test_x,train_y,test_y = train_test_split(x,y, random_state = 56)
```

Implementing KNN Regressor

[156]:
```
from sklearn.neighbors import KNeighborsRegressor as KNN
from sklearn.metrics import mean_absolute_error as mae
```

[157]:
```
reg = KNN(n_neighbors = 5)

reg.fit(train_x, train_y)

test_predict = reg.predict(test_x)
k = mae(test_predict, test_y)
print('Test MAE    ', k )
```

```
Test MAE     301.2533345805912
```

0.1.108 Conclusion:

- The optimum value for KNN Resgressor is 5 and it's corresponding mean absolute error is 301