Trips !!

BY CHINMAY AJNADKAR

Analysis of data collected by New York City Taxi and Limousine commission about Green taxis in September 2015

Chinmay Ajnadkar

Tel - +1 919-564-8255
Email - ajnadkar.chinmay@gmail.com

Contents

Que. 1 – Importing data

I have used python to do all the data analysis. I have used libraries such as Pandas, Numpy, Matplotlib, Skitlearn and datetime etc.

Importing Libraries

```
In [1]: import pandas as pd
        import matplotlib.pyplot as plt
        from datetime import datetime
        import numpy as np
```

Importing Data

I have imported the data in dataframe called 'df' and also checked its first 5 rows to confirm that the data has been imported correctly.

```
In [2]: df = pd.read_csv('green_tripdata_2015-09.csv')
        df.head()
```

Out[2]:

	VendorID	lpep_pickup_datetime	Lpep_dropoff_datetime	Store_and_fwd_flag	RateCodeID	Pickup_longitude	Pickup_latitude	Dropoff_longi
0	2	2015-09-01 00:02:34	2015-09-01 00:02:38	N	5	-73.979485	40.684956	-73.979431
1	2	2015-09-01 00:04:20	2015-09-01 00:04:24	N	5	-74.010796	40.912216	-74.010780
2	2	2015-09-01 00:01:50	2015-09-01 00:04:24	N	1	-73.921410	40.766708	-73.914413
3	2	2015-09-01 00:02:36	2015-09-01 00:06:42	N	1	-73.921387	40.766678	-73.931427
4	2	2015-09-01 00:00:14	2015-09-01 00:04:20	N	1	-73.955482	40.714046	-73.944412

5 rows × 21 columns

Number of rows and columns

This can be checked by using either **df.info()** or **df.shape()** command.

There are **1494926 rows** and **21 columns** in the data

Que. 1 – Importing data

```
In [4]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1494926 entries, 0 to 1494925
Data columns (total 21 columns):
VendorID                 1494926 non-null int64
lpep_pickup_datetime     1494926 non-null object
Lpep_dropoff_datetime    1494926 non-null object
Store_and_fwd_flag       1494926 non-null object
RateCodeID               1494926 non-null int64
Pickup_longitude         1494926 non-null float64
Pickup_latitude          1494926 non-null float64
Dropoff_longitude        1494926 non-null float64
Dropoff_latitude         1494926 non-null float64
Passenger_count          1494926 non-null int64
Trip_distance            1494926 non-null float64
Fare_amount              1494926 non-null float64
Extra                    1494926 non-null float64
MTA_tax                  1494926 non-null float64
Tip_amount               1494926 non-null float64
Tolls_amount             1494926 non-null float64
Ehail_fee                0 non-null float64
improvement_surcharge    1494926 non-null float64
Total_amount             1494926 non-null float64
Payment_type             1494926 non-null int64
Trip_type                1494922 non-null float64
dtypes: float64(14), int64(4), object(3)
memory usage: 239.5+ MB
```

df.shape() gives us following output

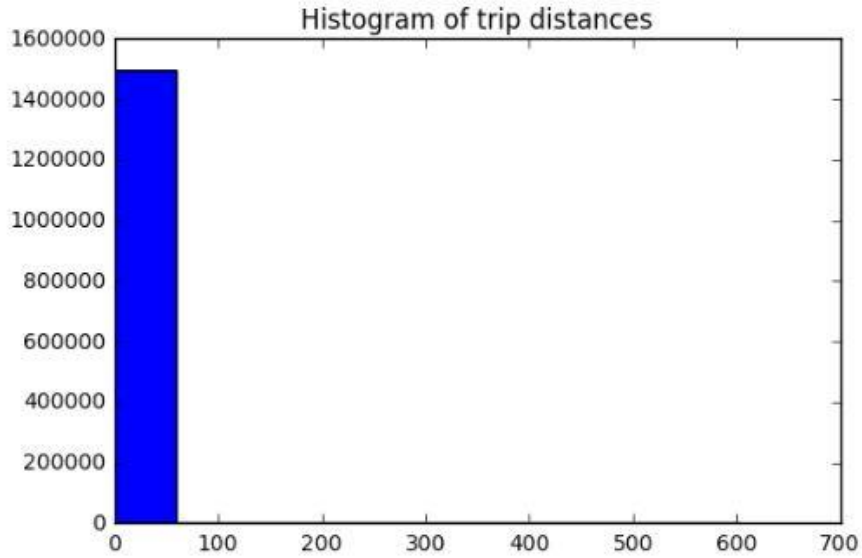
```
In [43]: df.shape

Out[43]: (1494926, 21)
```

Que.2 - Histogram of trip distance

The main issue of plotting the histogram of number of trip distance is that the range of the data is too large. If we plot normal histogram, it looks like this

```
In [7]: plt.hist(df['Trip_distance'])
        plt.title('Histogram of trip distances')
        plt.show()
```



Number of trips from 0 to 50 km are so large, that compared to the number of other trips are not visible in the histogram.

There are **3210** number of unique values for trip distance. And maximum trips are less than 20 km

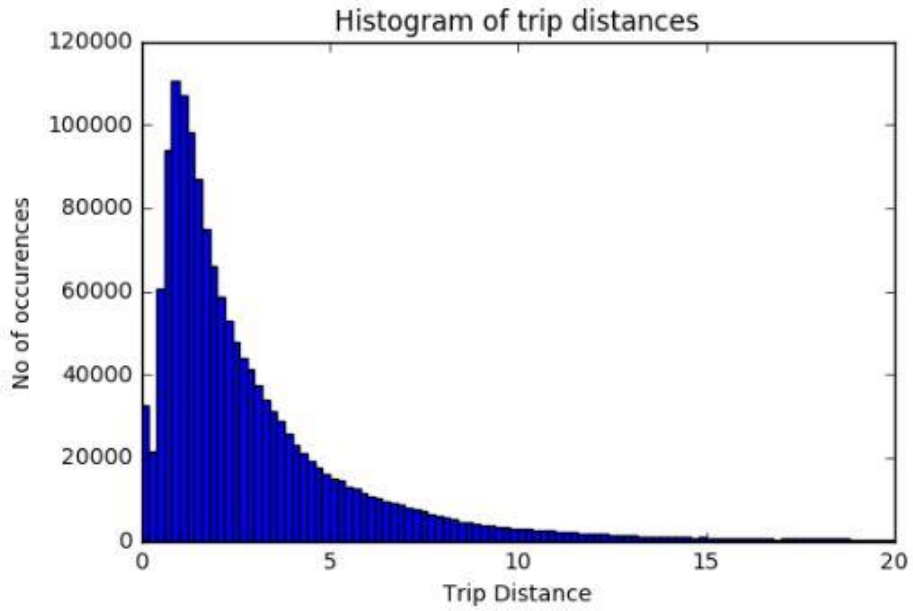
```
In [8]: df['Trip_distance'].nunique()
Out[8]: 3210

In [9]: df['Trip_distance'].value_counts().head()
Out[9]: 0.0    20592
        1.0    16735
        0.9    16699
        0.8    16152
        1.1    16070
        Name: Trip_distance, dtype: int64
```

Que.2 - Histogram of trip distance

So, I have plotted histogram of trips which are less than 20 km.

```
In [10]: plt.hist(df['Trip_distance'][df['Trip_distance']<20],bins = 100)
         plt.title('Histogram of trip distances')
         plt.xlabel('Trip Distance')
         plt.ylabel('No of occurences')
         plt.show()
```



It shows maximum trips are between 0 to 5 km.

Que.3 – Information about trip distance

Converting string into datetime format

First, I converted the ‘lpep_pickup_datetime’ string into datetime format so that the pickup hour can be extracted from it. And then I added new column in the data as ‘pickup_hour’ in the table using functions.

```
In [11]: df['pickup'] = df['lpep_pickup_datetime'].apply(lambda x:
                                                         datetime.strptime(x, '%Y-%m-%d %H:%M:%S'))
```

```
In [12]: df['pickup_hour'] = df['pickup'].apply(lambda x: x.hour)
```

Calculating mean and median of trip distance by pickup hour and plotting it

I used following functions to calculate mean and median and plot them

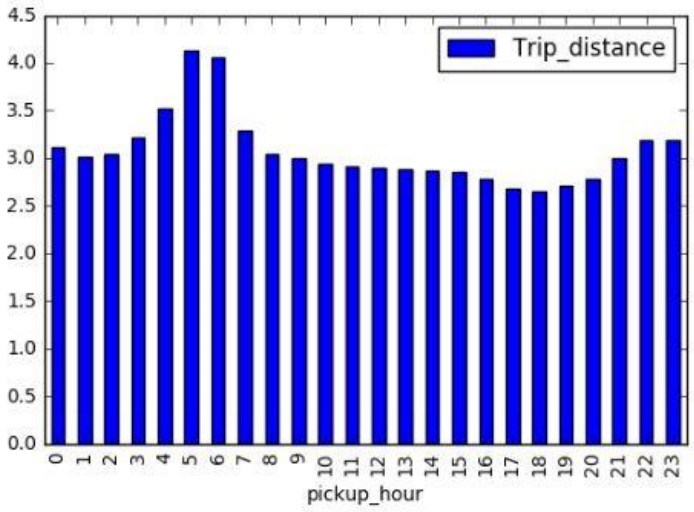
```
In [14]: print (df[['Trip_distance','pickup_hour']].groupby('pickup_hour').mean())

         print (df[['Trip_distance','pickup_hour']].groupby('pickup_hour').median())
```

```
In [13]: df[['Trip_distance','pickup_hour']].groupby('pickup_hour').mean().plot.bar()
         plt.show()
         df[['Trip_distance','pickup_hour']].groupby('pickup_hour').median().plot.bar()
         plt.show()
```

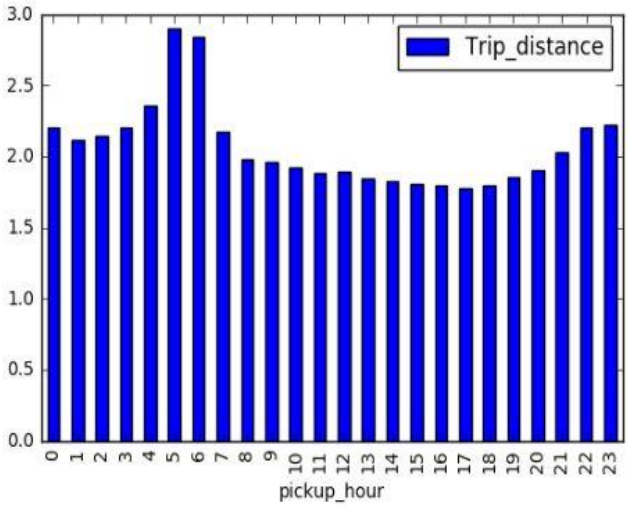
Que.3 – Information about trip distance

Mean



pickup_hour	Trip_distance
0	3.115276
1	3.017347
2	3.046176
3	3.212945
4	3.526555
5	4.133474
6	4.055149
7	3.284394
8	3.048450
9	2.999105
10	2.944482
11	2.912015
12	2.903065
13	2.878294
14	2.864304
15	2.857040
16	2.779852
17	2.679114
18	2.653222
19	2.715597
20	2.777052
21	2.999189
22	3.185394
23	3.191538

Median



pickup_hour	Trip_distance
0	2.20
1	2.12
2	2.14
3	2.20
4	2.36
5	2.90
6	2.84
7	2.17
8	1.98
9	1.96
10	1.92
11	1.88
12	1.89
13	1.84
14	1.83
15	1.81
16	1.80
17	1.78
18	1.80
19	1.85
20	1.90
21	2.03
22	2.20
23	2.22

Que.3 – Information about trip distance

Trips that originates or terminates at one of the NYC airports

To find if trip is originating or terminating at one of the airports or not, I first decided a range of coordinates for airports using google maps. I wrote a function such that, if trip is originating or terminating in that range, a column will store the name of that airport.

```
In [15]: def which_nyairport(row):
             if (
                 ((row['Pickup_longitude'] < -73.750296) & (row['Pickup_longitude'] > -73.821884) &
                (row['Pickup_latitude'] < 40.666467) & (row['Pickup_latitude']>40.646677)) |
                 (
                (row['Dropoff_longitude'] < -73.750296) & (row['Dropoff_longitude'] > -73.821884) &
                (row['Dropoff_latitude'] < 40.666467) & (row['Dropoff_latitude']>40.646677)) ):

                 return 'JFK'
             if (
                 ((row['Pickup_longitude'] < -73.865387) & (row['Pickup_longitude'] > -73.884547) &
                (row['Pickup_latitude'] < 40.773098) & (row['Pickup_latitude']>40.767550)) |
                 (
                (row['Dropoff_longitude'] < -73.865387) & (row['Dropoff_longitude'] > -73.884547) &
                (row['Dropoff_latitude'] < 40.773098) & (row['Dropoff_latitude']>40.767550)) ):

                 return 'LAG'
             else:
                 return None
```

```
In [16]: airport = [None]*df.shape[0]
         for index, row in df.iterrows():
             if index%1000 == 0:
                 print(index)
             airport[index] = which_nyairport(row)
```

If we take a count of number of trips originated or terminated at one of the airports,

JFK – 5216
LaGuardia – 4700

Some other interesting observations

If we calculate average fare for these trips, this is what we get,

Average fare for the trips terminating or originating at one of the airports is 26.62$
While average fare for other trips is 12.44 dollars.

From the data, we can infer that airports are located far from the city as the average fare for trips to/from airports is significantly higher that other trips

Que.4 – Analysis of tip paid

Derived variable for the tip paid

Added a new column as derived variable called ‘tip_percent’.

```
df['tip_percent'] = df['Tip_amount']/df['Fare_amount']
print((df['tip_percent']>1).value_counts())
df = df[df['tip_percent']<1]
```

```
False    1492331
True        2595
Name: tip_percent, dtype: int64
```

Predictive model for the tip percent

First, it is necessary to decide which variables to add to our predictive algorithm. We will do that by finding how many rows have null value in each column using following code.

```
col_list = ['VendorID','Store_and_fwd_flag','RateCodeID','Passenger_count','Payment_type',
            'Trip_type ', 'pickup_hour','airport',]
for col in col_list:
    print('----------------------','\n',col)
    print(df[col].value_counts())
    print(df[col].isnull().value_counts())
```

After observing the output from the above code, following variables are considered for predictive model:

1)'VendorID' (categorical)
2)'Store_and_fwd_flag' (categorical)
3)'Passenger_count' (numeric)
4)'Trip_distance' (numeric)
5)'Fare_amount' (numeric)
6)'Extra' (numeric)
7)'MTA_tax' (numeric)
8)'Tolls_amount' (numeric)
9)'improvement_surcharge'(numeric)

Que.4 – Analysis of tip paid

10)'Payment_type' (categorical)
11)'Trip_type ' (categorical)
12)'pickup_hour' (categorical)
13)'dropoff_hour' (categorical)
14)'airport' (categorical)

Currently our data resides in a DataFrame. But to deal with categorical values we would need to use the one-hot encoding process from the sklearn library which works only on numpy arrays, hence we convert the relevant columns to numpy arrays in further analysis and encode them to one-hot vectors.

After we create variables for categorical data, and run linear regression we get,

Mean Squared error = 0.01
Variance Score = 0.64

And coefficients for all the features as follows (including one-hot vectors coefficients)

```
np.abs(regr.coef_)
```

```
array([ 3.23224575e+06,  3.23224575e+06,  3.15168929e+09,
        3.15168929e+09,  5.83052843e-04,  4.37568036e-04,
        6.30100330e-04,  4.90054557e-03,  1.50352101e-02,
        4.91046906e-03,  1.99712627e-02,  5.49260959e+09,
        5.49260959e+09,  5.49260959e+09,  5.49260959e+09,
        2.72292424e+07,  2.72292424e+07,  8.00659875e+08,
        8.00659875e+08,  8.00659875e+08,  8.00659875e+08,
        8.00659875e+08,  8.00659875e+08,  8.00659875e+08,
        8.00659875e+08,  8.00659875e+08,  8.00659875e+08,
        8.00659875e+08,  8.00659875e+08,  8.00659875e+08,
        8.00659875e+08,  8.00659875e+08,  8.00659875e+08,
        8.00659875e+08,  8.00659875e+08,  8.00659875e+08,
        1.48491763e+09,  1.48491763e+09,  1.48491763e+09,
        1.48491763e+09,  1.48491763e+09,  1.48491763e+09,
        1.48491763e+09,  1.48491763e+09,  1.48491763e+09,
        1.48491763e+09,  1.48491763e+09,  1.48491763e+09,
        1.48491763e+09,  1.48491763e+09,  1.48491763e+09,
        1.48491763e+09,  1.48491763e+09,  1.48491763e+09,
        1.48491763e+09,  1.48491763e+09,  1.48491763e+09,
        4.56884556e+08,  4.56884556e+08,  4.56884556e+08])
```

Que.5 – Distribution of the speed

Trips taking time less than 60 second are slightly hard to be true. So we are omitting all such trips. Total percentage of such trips is 1.588%

```
ans_t = (df['dropoff'] - df['pickup']).apply(lambda x: x.total_seconds())
```

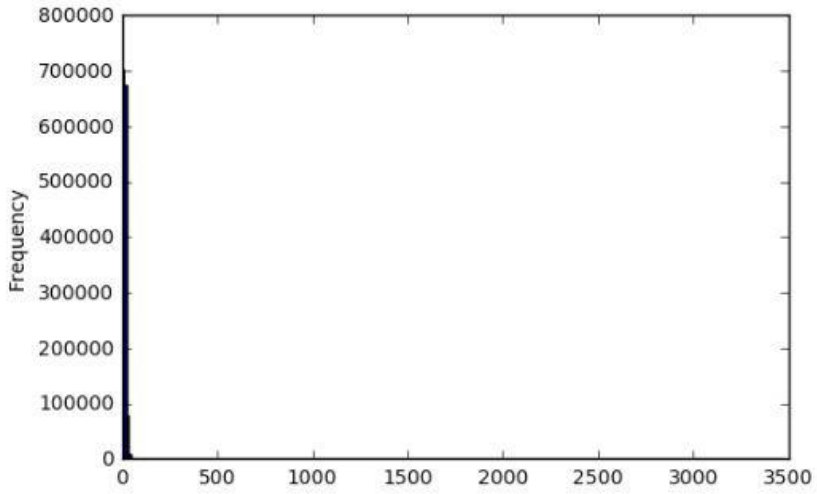
Trips taking less than 60 seconds are slightly hard to be true, hence we omit these entries as they could harm our analysis. There are 23630 such entries i.e. 1.5% of the entire data, hence we can proceed after dropping this part.

```
print('Percentage of entries with travel time less than a minute: ',100 * df[ans_t < 60].shape[0]/df.shape[0],'%')
```

```
Percentage of entries with travel time less than a minute:  1.5888015668235596 %
```

Now, Average speed has been calculated and for each week, histogram is plotted.

```
plt.show()
```



However this is inconclusive as we are getting values till 2500 miles/hour which is not possible. So, we filter away the 93 entries with over 100 miles per hour of average speed as it is unreasonable and must have been a result of some erroneous data collection process or later in the pipeline.

After filtering, number of values in each week are as follows,

37	356715
38	352786
39	332219
36	292139
40	130259

Que.5 – Distribution of the speed

Average speed of those weeks are as follows:

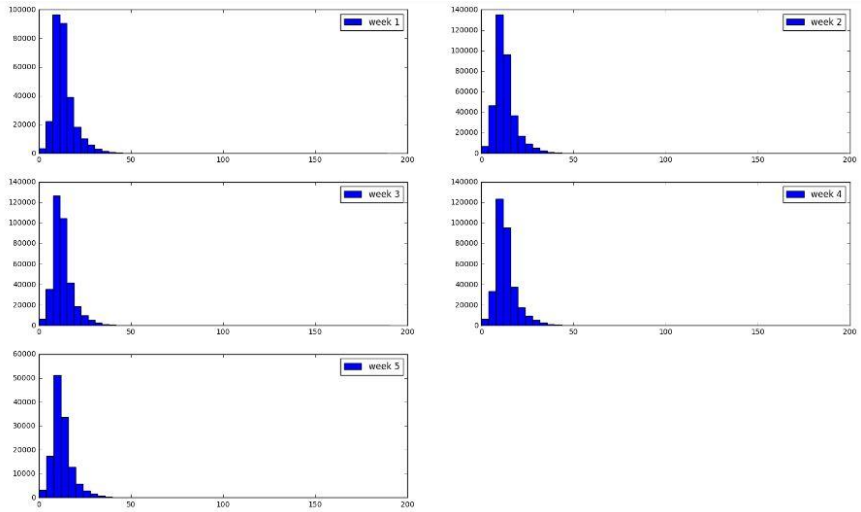
Week 1 : 12.13
Week 2 : 11.62
Week 3 : 12.71
Week 4 : 13.19
Week 5 : 12.47

Now to check, if this variation in speed is just by chance (Null Hypothesis) or it has some statistical significance (Alternative hypothesis) ANOVA analysis has been done.

We get, **F value= 1025.19 and p value= 0.0**

That means we can **reject the null hypothesis and conclude that this variation in average speed is not by chance.**

The ANOVA test indicates a large f-value and a small p-value, therefore we reject the null hypothesis and we conclude that the differences between the groups are statistically significant which implies that the week of the month does seem to be related to the average speed. We further compute the mean, median and histogram for these groups to support our claims



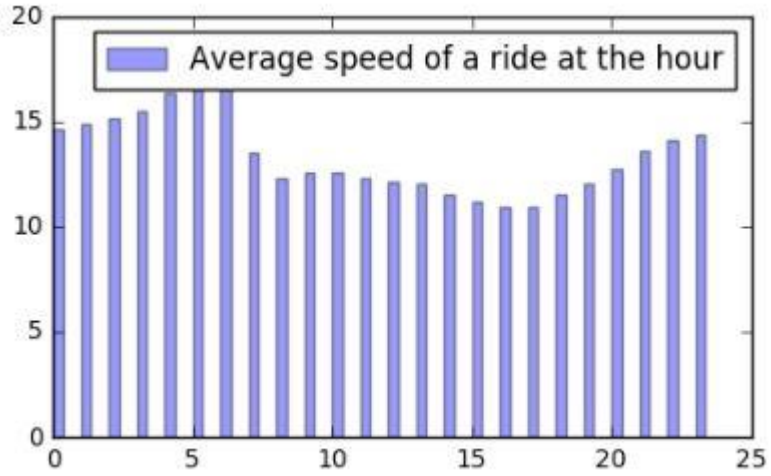
Que.5 – Distribution of the speed

Mean speed grouped by Hour

Similarly, applying ANOVA analysis on the data grouped by hour, we get:

F value = 4941.13 and p value = 0.0

The ANOVA test for sets partitioned as per the hour of the journey also gives a high f-value and p-value of 0, implying that there are statistically significant differences in the data sets considered



Which makes sense as

- There is high traffic in the mornings from 8 to 10 and evenings from 5 to 7 as people are usually going to/coming from work, so average speed is low
- Traffic is usually very low in early mornings so average speed is high