Analysis of NYC Green Taxi Data

Jicheng Lu

1 Introduction

In this report, I will analyze the New York City Green Taxi data of September 2015. This includes the basic data analysis and building of a predictive model for tip percentage. The code is written in Python3 with several additional libraries, such as sklearn, scipy, tabulate. To run the code, please check the code instruction.

2 Question 1

2.1 Download Data and Read Data Size

The first task is to download the original dataset and present its size. This can be done by using "urllib.request.urlretrieve". The code implementation is given as follows:

Box 1. Code – Data download.

```
# fix the error when using urllib.request.urlretrieve
if (not os.environ.get('PYTHONHTTPSVERIFY', '') and
    getattr(ssl, '_create_unverified_context', None)):
      ssl._create_default_https_context = ssl._create_unverified_context

# Download the Trip Record Data
url = 'https://s3.amazonaws.com/nyc-tlc/trip+data/green_tripdata_2015-09.csv'
filename = '/Users/lujicheng/Desktop/capitalone/green_tripdata_2015-09.csv'
urllib.request.urlretrieve(url, filename)

# read the data
green_data = pd.read_csv("green_tripdata_2015-09.csv")
print('Green Taxi data size: ')
print('Number of rows:', green_data.index.size)
print('Number of columns:', green_data.columns.size)
```

The data size is:

```
Number of rows: 1494926
Number of columns: 21
```

3 Question 2

3.1 Trip Distance Histogram

The task is to look into the distribution of trip distance. To do this, I first use the entire data to plot the trip distance distribution. Then I exclude the outliers to get the meaningful distribution. The outliers are defined as the data located further than 3 standard deviations from the mean value. The histogram of trip distance is given in Fig. 1.

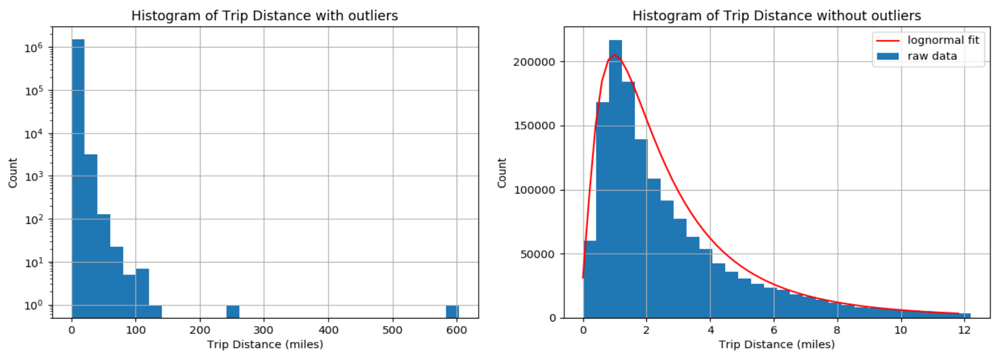


Figure 1: Histogram of trip distance.

It can be seen that the trip distance is not symmetrically distributed. It is skewed to the right and its median is smaller than the mean value. Such structure can be modeled as lognormal distribution, which is indicated by the red line in Fig.1. **Hypothesis:** since it is not a symmetrical normal distribution, we can say the trips are not random. Thus, people in NYC may use green cab for specific reason, such as going to airport or train station.

The code implementation is given as follows:

Box 2. Code – Histogram of trip distance.

```
# define figure
fig,ax = plt.subplots(1,2,figsize = (15,5))

# histogram of trip distance
green_data.Trip_distance.hist(bins=30,ax=ax[0])
ax[0].set_xlabel('Trip Distance (miles)')
ax[0].set_ylabel('Count')
ax[0].set_yscale('log')
ax[0].set_title('Histogram of Trip Distance with outliers')

# get the Trip Distance data
trip_dist = green_data.Trip_distance

# exclude the outliers
trip_dist_filter = trip_dist[ ((trip_dist - trip_dist.mean()).abs() ¿ 3*trip_dist.std())]
trip_dist_filter.hist(bins=30, ax=ax[1])
ax[1].set_xlabel('Trip Distance (miles)')
ax[1].set_ylabel('Count')
ax[1].set_title('Histogram of Trip Distance without outliers')

# apply a lognormal fit – use trip distance mean
scatter, loc, mean = lognorm.fit(green_data.Trip_distance.values,
scale=green_data.Trip_distance.mean(), loc=0)
pdf = lognorm.pdf(np.arange(0,12,0.2), scatter, loc, mean)
ax[1].plot(np.arange(0,12,0.2),700000*pdf,'r')
ax[1].legend(['lognormal fit', 'raw data'])

plt.savefig('Question2_trip_dist_histo.png')
print('Please close the figure to continue ...')
plt.show()
```

4 Question 3

4.1 Mean and Median Trip Distance by Hour of Day

This task is to obtain the mean and median value of the trip distance by hour of day. I use a curve plot to smoothly present the variation of both mean and median of trip distance with the hour in a day. The result is presented in Fig. 2. The exact data is given in Table. 1.

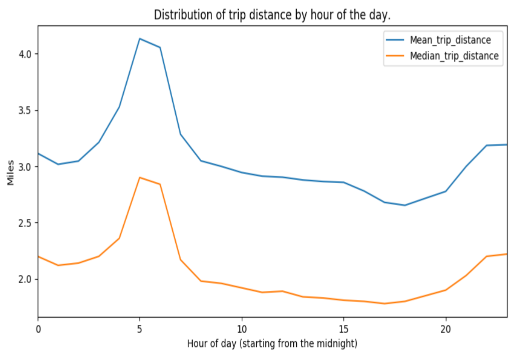


Figure 2: Median and Mean of trip distance versus hour of day.

Table 1: Data of mean and median of trip distance with respect to hour of day.

Hour	Mean trip distance	Median trip distance	**Hour**	Mean trip distance	Median trip distance
0	3.115	2.20	12	2.903	1.89
1	3.017	2.12	13	2.878	1.84
2	3.046	2.14	14	2.864	1.83
3	3.213	2.20	15	2.857	1.81
4	3.527	2.36	16	2.779	1.80
5	4.133	2.90	17	2.679	1.78
6	4.055	2.84	18	2.653	1.80
7	3.284	2.17	19	2.716	1.85
8	3.048	1.98	20	2.777	1.90
9	2.999	1.96	21	2.999	2.03
10	2.944	1.92	22	3.185	2.20
11	2.912	1.88	23	3.192	2.22

We can see from Fig. 2 that there are two peaks in the morning and evening. This may indicate that people in NYC are taking taxi more frequently at these two time intervals. Maybe people are getting up early to avoid being late for work or people are working until midnight and then take taxi home, which is a typical living style in a big city.

The code implement for this task is given as follow:

Box 3. Code – Mean and median of trip distance.

```
# output mean and median of trip distance by pickup hour
fig,ax = plt.subplots(1, 1, figsize=(10, 5))

# use a pivot table to aggregate the trip distance by pickup hour
table = pd.pivot_table(data=green_data, index='Hour',
    values='Trip_distance', aggfunc=np.mean, np.median).reset_index()
```

Box 3. Code – Mean and median of trip distance (continue).

```
# rename columns
table.columns = ['Hour','Mean_trip_distance','Median_trip_distance']
table[['Mean_trip_distance','Median_trip_distance']].plot(ax=ax)

plt.xlabel('Hour of day (starting from the midnight)')
plt.ylabel('Miles')
plt.title('Distribution of trip distance by hour of the day.')
plt.xlim([0, 23])
plt.savefig('Question3_mean_median_trip_dist.png')
print('Please close the figure to continue ...')
plt.show()
```

4.2 Airport Trips

The task is to identify trips that originate or terminate at one of the NYC airports. First, we need to identify the airport area. It is efficient to use the "RateCodeID", where 2 means JFK and 3 means Newark airport. Other methods may use the coordinate of these areas and check the location of pickup or drop-off to see if the data fulfills the requirement.

Next, let's check the number of the trips and the average fare of the trips that originate or terminate at one of the NYC airports.

```
Number of trips from or to one of NYC airports: 5552
Average fare of trips from or to NYC airports: $51.37
```

Then I compare the data of airport trips and non-airport trips. Fig. 3 shows the trip distance count and hourly distribution of these kinds of trips, respectively. It can be seen from the trip distribution that there are two peaks for the airport trips. The first peak occurs for trips within 3 miles, which is similar to the non-airport trips. The second peak for airport trips occurs at around 18 miles. This is because the JFK and Newark airport are located around 17 miles from downtown.

Moreover, we can see from the hourly distribution: people tend to travel to airport more frequently at 3 p.m. and the fewest airport trips occur around 2 a.m. in the morning. For non-airport trips, there are two peaks: one is at 8 a.m. and the other one is at 6 p.m. This makes sense because people are rushing to work or home after work at these two periods.

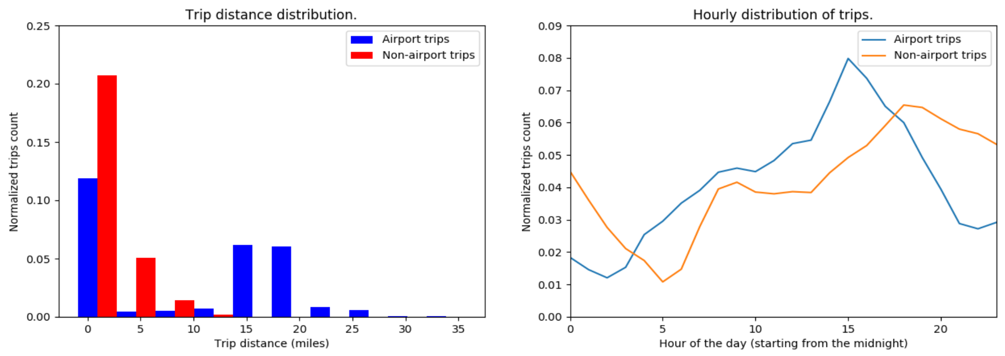


Figure 3: Median and Mean of trip distance versus hour of day.

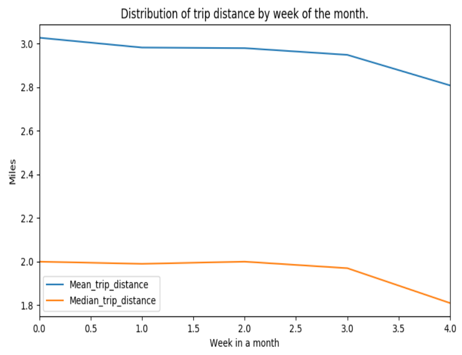
The code implementation is given as follows:

Box 4. Code – airport trips and non-airport trips.

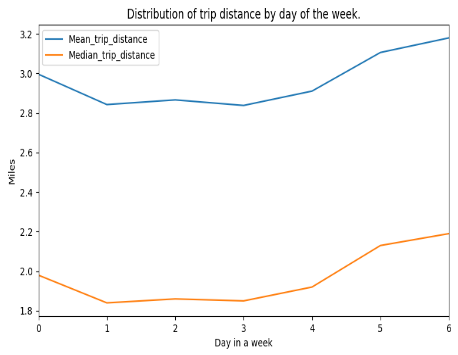
```
airport_trips = green_data[(green_data.RateCodeID==2) — (green_data.RateCodeID==3)]
print("Number of trips from/to one of NYC airports: ", airport_trips.shape[0])
print("Average fare of trips from/to NYC airports: $0:.2f"
  .format(airport_trips.Fare_amount.mean()))

# get the airport trip distance
v1 = airport_trips.Trip_distance
# get the non-airport trip distance
v2 = green_data.loc[ green_data.index.isin(v1.index), 'Trip_distance']

# exculde outliers
v1 = v1[ ((v1 - v1.mean()).abs() >3*v1.std())]
v2 = v2[ ((v2 - v2.mean()).abs() >3*v2.std())]

# define bins boundaries
bins = np.histogram(v1,density=True)[1]
h1 = np.histogram(v1,bins=bins,density=True)
h2 = np.histogram(v2,bins=bins,density=True)

# plot histogram of airport trips and non-airport trips
fig,ax = plt.subplots(1, 2, figsize = (15,5))
width = 0.5*(bins[1]-bins[0])
ax[0].bar(bins[:-1], h1[0], width=width, alpha=1, color='b')
ax[0].bar(bins[:-1] + width, h2[0], width=width, alpha=1, color='r')
ax[0].legend(['Airport trips','Non-airport trips'], loc='best')
ax[0].set_xlabel('Trip distance (miles)')
ax[0].set_ylabel('Normalized trips count')
ax[0].set_ylim([0, 0.25])
ax[0].set_title('Trip distance distribution.')

# plot histribution of airport trips and non-airport trips by hour of day
airport_trips.Hour.value_counts(normalize=True).sort_index().plot(ax=ax[1])
green_data.loc[v2.index, 'Hour'].value_counts(normalize=True).sort_index().plot(ax=ax[1])
ax[1].set_xlabel('Hour of the day (starting from the midnight)')
ax[1].set_ylabel('Normalized trips count')
ax[1].set_ylim([0, 0.09])
ax[1].set_title('Hourly distribution of trips.')
ax[1].legend(['Airport trips','Non-airport trips'], loc='best')
plt.savefig('Question3_airport_trips.png')
plt.show()
```

4.3 Other

In addition, I also explore other aspects of the data. Fig. 4 demonstrates the mean and median of trip distance with week in a month and day in a week, respectively. Fig. 5 presents mean and median of trip distance with the passenger count.



(a) Mean and median of trip distance versus week in the month.



(b) Mean and median of trip distance versus day in a week.

Figure 4: Mean and median of trip distance.

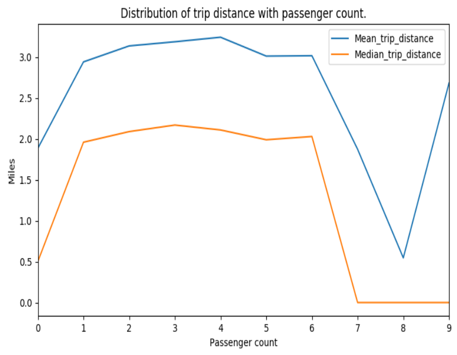


Figure 5: Mean and median of trip distance versus passenger count.

We can see in Fig. 4 that people tend to travel further on weekend than the other weekdays while the trip distance has little difference between different weeks in a month. Moreover, in Fig. 5, 2-4 passengers in a taxi tends to travel further. The code implement is given as follows:

Box 5. Code – Mean and median of trip distance.

```
# plot week_day vs trip_distance
fig,ax = plt.subplots(1, 1, figsize = (9,5))
table2 = pd.pivot_table(data=green_data, index='Week_day',
    values='Trip_distance',aggfunc=np.mean, np.median).reset_index()
# rename columns
table2.columns = ['Week_day','Mean_trip_distance','Median_trip_distance']
table2[['Mean_trip_distance','Median_trip_distance']].plot(ax=ax)

plt.xlabel('Day in a week')
plt.ylabel('Miles')
plt.title('Distribution of trip distance by day of the week.')
plt.savefig('Question3_mean_median_weekday.png')
plt.show()
```

Box 5. Code – Mean and median of trip distance (continue).

```
# plot week vs trip_distance
fig,ax = plt.subplots(1, 1, figsize = (9,5))
table3 = pd.pivot_table(data=green_data, index='Week',
    values='Trip_distance',aggfunc=np.mean, np.median).reset_index()
# rename columns
table3.columns = ['Week','Mean_trip_distance','Median_trip_distance']
table3[['Mean_trip_distance','Median_trip_distance']].plot(ax=ax)

plt.xlabel('Week in a month')
plt.ylabel('Miles')
plt.title('Distribution of trip distance by week of the month.')
plt.savefig('Question3_mean_median_week.png')
plt.show()

# plot passenger_count vs trip_distance
fig,ax = plt.subplots(1, 1, figsize = (9,5))
table4 = pd.pivot_table(data=green_data, index='Passenger_count',
    values='Trip_distance',aggfunc=np.mean, np.median).reset_index()
# rename columns
table4.columns = ['Passenger_count','Mean_trip_distance','Median_trip_distance']
table4[['Mean_trip_distance','Median_trip_distance']].plot(ax=ax)

plt.xlabel('Passenger count')
plt.ylabel('Miles')
plt.title('Distribution of trip distance with passenger count.')
plt.savefig('Question3_mean_median_passenger.png')
plt.show()
```

5 Question 4

5.1 Data Preprocessing

Before building the predictive model, I first check the original data. This includes: finding negative values (e.g., in Total_amount, Fare_amount, Tip_amount) and replace them with absolute values, setting the Total_amount less than $2.5 (since the minimum charge for green taxi is $2.5), and converting the pickup date time and drop-off date time to right format. Part of the code is given as follows:

Box 6. Code – check data.

```
# replace the negative values
data.Total_amount = data.Total_amount.abs()
data.Fare_amount = data.Fare_amount.abs()
data.improvement_surcharge = data.improvement_surcharge.abs()
data.Tip_amount = data.Tip_amount.abs()
data.Tolls_amount = data.Tolls_amount.abs()
data.MTA_tax = data.MTA_tax.abs()

# total amount: the min fare of green taxi is $2.5
tmp_index = data[(data.Total_amount¡2.5)].index
data.loc[tmp_index, 'Total_amount'] = 2.5

# convert time variables to right format
data['Pickup_datetime'] = data.lpep_pickup_datetime.apply(lambda x:dt.datetime.strptime(x,"%Y-%m-%d
%H:%M:%S"))
data['Dropoff_datetime'] = data.Lpep_dropoff_datetime.apply(lambda x:dt.datetime.strptime(x,"%Y-%m-
%d %H:%M:%S"))
```

5.2 Build Tip Percentage

In this part, I build the tip percentage variable by using formula "Tip_amount / Total_amount". Besides, I will add other derived variables, such as Week, Week_day, Month_day, Hour, Trip_duration, and Speed_mph. Some of these derived variables have been used in the analysis in the previous sections. Part of the code is given as follows. Note that some of the "Speed_mph" data maybe either null or larger than 240. I replace these data by values sampled from a random distribution of mean 12.9 and standard deviation 6.8 mph. This idea is obtained from (1).

Box 7. Code – add features.

```
# adding time variables
ref_week = dt.datetime(2015,9,1).isocalendar()[1]
data['Week'] = data.Pickup_datetime.apply(lambda x:x.isocalendar()[1])-ref_week+1
data['Week_day'] = data.Pickup_datetime.apply(lambda x:x.isocalendar()[2])
data['Month_day'] = data.Pickup_datetime.apply(lambda x:x.day)
data['Hour'] = data.Pickup_datetime.apply(lambda x:x.hour)

# add trip duration (in min)
data['Trip_duration'] = ((data.Dropoff_datetime-data.Pickup_datetime)
    .apply(lambda x:x.total_seconds()/60.0))

# add avg speed in mph
data['Speed_mph'] = data.Trip_distance/(data.Trip_duration/60.0)
tmp_index = data[(data.Speed_mph.isnull()) — (data.Speed_mph¿240)].index
data.loc[tmp_index,'Speed_mph'] = np.abs(np.random.normal(loc=12.9,scale=6.8,size=len(tmp_index)))

# add tip percentage
data['Tip_percentage'] = 100 * data.Tip_amount / data.Total_amount
# add 'Has_tip' to indicate whether the passenger gives tip or not
data['Has_tip'] = (data.Tip_percentage¿0)*1
```

5.3 Build Predictive Model for Tip Percentage

Before building the predictive model for tip percentage, we need to understand the data first. Since not all the transactions include tips, we need to build a classifier first in order to identify whether the customers pay the tips or not. Then we can build a regressor to predict the target variable - tip percentage.

First, let's build the classifier. The target parameter is "Has_tip", where 0 means no tip and 1 means paying tip. The input feature parameters I select here are: 'Payment_type', 'Total_amount', 'Trip_duration', and 'Speed_mph'. This is because these variables are highly correlated with the target, based on the analysis from (2). The classifier I have built is using Random Forest Classifier in the sklearn library. After initializing the classifier, I tune the model parameters by using "GridSearchCV" with "roc_auc" scoring method, which performs exhaustive search over input parameter values for an estimator. This idea is obtained from (1). After this, I start to train the classifier with 5 folds cross-validation by using 100,000 samples. The roc-auc socre is used to be model validation metric. Note that only the selected parameters are optimized. The code is too long to present here. For details, please check "question4.py". The training and testing results are presented as follows: (The number of the testing samples is 100,000, which is different from the training samples.)

```
# classifier training result
Training accuracy: 99.97%
Training roc-auc score: 1.00000
Cross-validation score == Mean: 0.9963997 — Std: 0.0002583017 — Max: 0.9966953 — Min: 0.9960083

# classifier testing result
Test accuracy: 97.19%
Test roc-auc score: 0.99660
```

Next, let's build the regressor. The target parameter is "Tip_percentage". The input feature parameters I select here are: 'Payment_type', 'Total_amount', 'Trip_duration', and 'Speed_mph', because of the high correlation with target. The regressor I have built is using Random Forest Regressor in the sklearn library. After initializing the regressor, I again tune the model parameters by using "GridSearchCV" with "negative mean squared error" scoring method. Then, I start to train the regressor with 5 folds cross-validation by using 100,000 samples. The negative-mean-squared-error socre is used to be model validation metric. For code details, please check "question4.py". The training and testing results are presented as follows: (The number of the testing samples is 100,000, which is different from the training samples.)

```
# regressor training result
Training mse: 2.100155780232665
Cross-validation score == Mean: -15.27912 — Std: 0.8115307 — Max: -14.11573 — Min: -16.60088

# regressor testing result
rf model test mse: 14.809298810780854
rf model r2 score: 0.5504112264147301
```

Finally, let's build the predictive model for the tip percentage. The model first identifies whether the customer paid the tip or not by classifier. Then it predicts the specific tip percentage values given inputs. The evaluation results are given as follows with 100,000 random samples, and the residual distribution is given in Fig. 6. We can see that the residual is almost symmetrically distributed and most of the predictions are located around the zero residual, which means the predictive model has a good performance. (For code details, please check "question4.py")

```
# predictive model evaluation
Predictive model test mse: 12.078011638941438 Predictive model test r2 score: 0.84744632739341
```

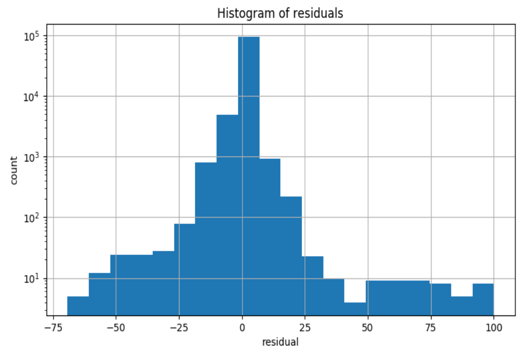


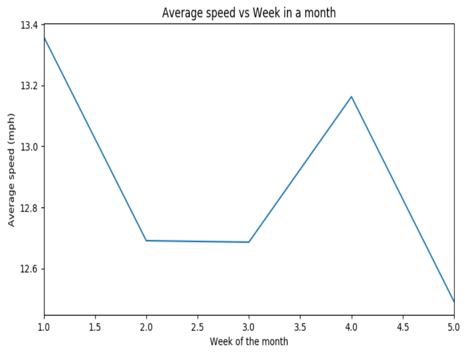
Figure 6: Residual distribution.

6 Question 5

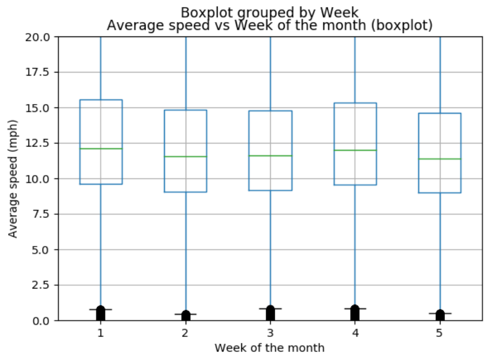
6.1 Speed Distribution

First, I build a variable of average speed in mph. This is implemented in the "feature" function in data preprocessing.

Next, let's analyze whether the average trip speeds are materially the same in all weeks of September. I first present the curve and box plot of the average speed with respect to the weeks (Fig 7).



(a) Curve plot of average speed versus week in the month.



(b) Box plot of average speed versus week in the month.

Figure 7: Average speed versus week in the month.

We can see from Fig 7 that the average speed is not uniformly distributed among the weeks in the month. Thus, it should be week dependent. Then let's perform the student t-test to statically check the result. The null hypothesis for the t-test is that the mean values are the same for the two samples. The p-values are presented in Table. 2.

Table 2: P-values in t-test.

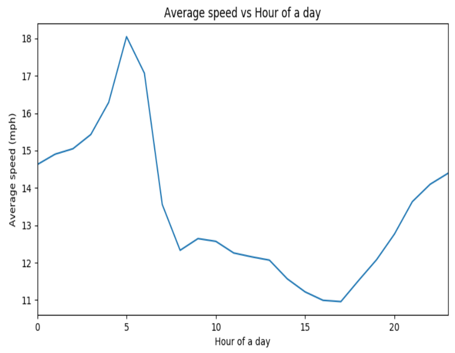
Week	1	2	3	4	5
1	1	0	0	1.65e-28	0
2	0	1	0.746	1.08e-173	1.89e-19
3	0	0.746	1	5.26e-181	1.05e-18
4	1.65e-28	1.08e-173	5.26e-181	1	5.99e-198
5	0	1.89e-19	1.05e-18	5.99e-198	1

It can be seen that the p-value between Week 2 and Week 3 is 0.75 while the rest of p-values are very close to zero. Therefore, we can reject the null hypothesis at 95% level of confidence, which means that the average trip speeds are not the same in all weeks of the month. We can also further check this result by ANOVA test. As we can see, the p-value for ANOVA test is zero, which prove the result. **Hypothesis:** The reason why the average speed differs in different weeks in the month may be that there is some special event in a specific week, which causes more traffics.

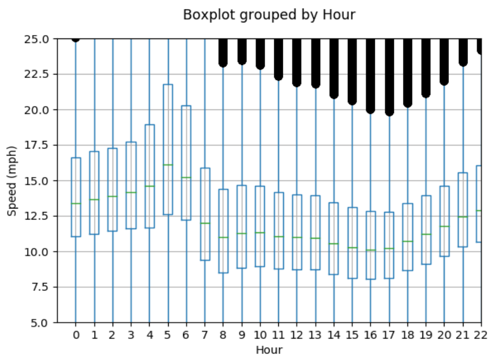
```
Anova test result: F_onewayResult(statistic=719.6066182049119, pvalue=0.0)
```

In the next step, the task is to check the average speed over the hour of day. In this case, I use ANOVA test on the samples from different hours. The result is shown as follows. Since the p-value is zero, we can say the average speed is different in different hours. This validates the results in Fig. 8, where the hourly distribution of speed is apparently not uniform. **Hypothesis:** The reason why the average speed differs in different hours in a day may be the rushing hour for work when there is a traffic jam on the street. So most of the cabs drive very slowly. Meanwhile, the average speed in the early morning and late night is pretty high, since there are not many vehicles on the road at these periods.

```
Anova test result: F_onewayResult(statistic=3386.8024345026515, pvalue=0.0)
```



(a) Curve plot of average speed versus hour in a day.



(b) Box plot of average speed versus hour in a day.

Figure 8: Average speed versus hour in a day.

The code implementation is given as follows:

Box 9. Code – t-test and ANOVA.

```
def t_test(data):
    weeks = [1,2,3,4,5]
    p_vals = []
    # run t-test for each pair
    for i in range(len(weeks)):
       for j in range(len(weeks)):
          p_vals.append((weeks[i], weeks[j],ttest_ind(data[data.Week==weeks[i]].Speed_mph,
              data[data.Week==weeks[j]].Speed_mph, equal_var=False)[1]))
    p_values = pd.DataFrame(p_vals, columns=['week_x', 'week_y', 'p_val'])
    return p_values

def anova_week():
    weeks = [1,2,3,4,5]
    cmd = "f_oneway("
    for w in weeks:
       cmd+="green_data[green_data.Week=="+str(w)+"].Speed_mph,"
    cmd=cmd[:-1]+")"
    return cmd

def anova_hour():
    hours = range(24)
    cmd = "f_oneway("
    for h in hours:
       cmd+="green_data[green_data.Hour=="+str(h)+"].Speed_mph,"
    cmd=cmd[:-1]+")"
    return cmd

p_values = t_test(green_data)
print("p-values:",pd.pivot_table(p_values, index='week_x', columns='week_y', values='p_val').T)
cmd1 = anova_week()
print("Anova test result:", eval(cmd1))

cmd2 = anova_hour()
print("Anova test result:", eval(cmd2))
```

7 Conclusion

This report performs analysis on the NYC Green Taxi data in September 2015. The work includes:

- Programmatically download the data from the website and present the data size.
- Plot a histogram of the number of the trip distance and perform analysis.
- Analyze mean and median trip distance grouped by hour of day and other features, such as week, week day, passenger count.
- Compare the airport trips with non-airport trips.
- Build a derived variable: tip percentage.
- Build a predictive model for tip percentage, based on random forest classifier and regressor.
- Build a derived variable representing the average speed over the course of a trip.
- Perform t-test and ANOVA test to average speed with respect to week of month and hour of day.

8 Reference

(1) https://github.com/kthouz/NYC_Green_Taxi

(2) https://github.com/sabarna/NYC-Green-Cab-Data-Analysis