Analyzing The 2015 September NYC Taxi And Limousine Trip Record Data

Data description:

The NYC (New York City) Taxi and Limousine Commission was created in 1971. It is the agency responsible for licensing and regulating New York City's medallion (yellow) taxicabs, for-hire vehicles (community-based liveries, black cars and luxury limousines), commuter vans, and paratransit vehicles.

Starting from 2009, the TLC trip record data is available on the website. The yellow and green taxi trip records include fields capturing pick-up and drop-off dates/times, pick-up and drop-off locations, trip distances, itemized fares, rate types, payment types, and driver-reported passenger counts.

The data we are going to analyze is the record data of green taxis of 2015 September.

# Question 1.

1. **Programmatically download and load into your favorite analytical tool the trip data for September 2015.**

First, let's import the libraries we are going to use.

1. **import** numpy as np
2. **import** pandas as pd
3. **from** IPython.display **import** display
4. **import** matplotlib.pyplot as plt
5. **import** seaborn as sns
6. **from** sklearn.model\_selection **import** train\_test\_split
7. **import** scipy.stats as stats
8. **import** statistics
9. **import** matplotlib as mpl
10. **from** sklearn.cluster **import** MiniBatchKMeans, KMeans#Clustering

Then let's download the data, and display 10 records (first 5 and last 5) and all 21 columns.

1. # Download the Trip Record Data of 2015-09
2. month = 9
3. urllib.request.urlretrieve("https://s3.amazonaws.com/nyc-tlc/trip+data/"+ \
4. "yellow\_tripdata\_2015-{0:0=2d}.csv".format(month),
5. "nyc.2015-{0:0=2d}.csv".format(month))
7. # Download the Location Data
8. urllib.request.urlretrieve("https://s3.amazonaws.com/nyc-tlc/misc/taxi\_zones.zip", "taxi\_zones.zip")
10. # PATH = "/Users/Kai/Desktop/Project\_capitalOne/green\_tripdata\_2015-09.csv" # local file path where the .csv file is saved.
12. data = pd.read\_csv("/Users/Kai/Dropbox/Project\_capitalOne/green\_tripdata\_2015-09.csv")
14. # Read data into a pandas dataframe and show 10 lines and all 21 columns
15. pd.options.display.max\_rows = 10
16. pd.options.display.max\_columns = 21
17. display(data)

The output data are displayed as below. Only the first few columns are displayed out.

1. VendorID lpep\_pickup\_datetime Lpep\_dropoff\_datetime  \
2. 0               2  2015-09-01 00:02:34   2015-09-01 00:02:38
3. 1               2  2015-09-01 00:04:20   2015-09-01 00:04:24
4. 2               2  2015-09-01 00:01:50   2015-09-01 00:04:24
5. 3               2  2015-09-01 00:02:36   2015-09-01 00:06:42
6. 4               2  2015-09-01 00:00:14   2015-09-01 00:04:20
7. ...                  ...                   ...
8. 1494921         1  2015-09-30 23:00:01   2015-09-30 23:17:21
9. 1494922         1  2015-09-30 23:00:05   2015-09-30 23:08:13
10. 1494923         1  2015-09-30 23:00:30   2015-09-30 23:08:39
11. 1494924         1  2015-09-30 23:00:10   2015-09-30 23:03:49
12. 1494925         1  2015-09-30 23:00:11   2015-09-30 23:05:36
13. **Report how many rows and columns of data you have loaded.**

The following code is to display the number of rows and columns:

1. numRows = len(data.axes[0]) # 0 for row
2. numCols = len(data.axes[1]) # 1 for column
4. **print**("Number of rows is:", numRows)
5. **print**("Number of columns is:", numCols)

and the output is:

1. Number of rows **is**: 1494926
2. Number of columns **is**: 21

There are 1494926 records and 21 fields.

# Question 2

1. **Plot a histogram of the number of the trip distance (Trip Distance).**

The following code is to see the minimum and maximum trip distance.

1. display(data.Trip\_distance)
2. minTripDistance = data['Trip\_distance'].min() # minimum trip distance
3. maxTripDistance = data['Trip\_distance'].max() # maximum trip distance
4. **print**("The minimum trip distance is: ", minTripDistance, " miles")
5. **print**("The maximum trip distance is: ", maxTripDistance, " miles")

and the output is as follows:

1. 0           0.00
2. 1           0.00
3. 2           0.59
4. 3           0.74
5. 4           0.61
6. ...
7. 1494921    10.60
8. 1494922     1.80
9. 1494923     1.40

The minimum trip distance is 0 miles, and maximum be 603 miles, which does not make much sense if the taxi is driving inside the New York City. We will do some data cleansing in following steps.

1. 1494924     0.80
2. 1494925     1.40
3. Name: Trip\_distance, Length: 1494926, dtype: float64
4. The minimum trip distance **is**:  0.0  miles
5. The maximum trip distance **is**:  603.1  miles

The following code is to display the histogram of the trip distances.

1. ax = data['Trip\_distance'].hist(bins=30, figsize=(15,5))
2. ax.set\_yscale('log')
3. ax.set\_xlabel("Trip distance / Miles")
4. ax.set\_ylabel("Number of Trips (Log Scale)")
5. plt.show()

And the histogram is shown as below.

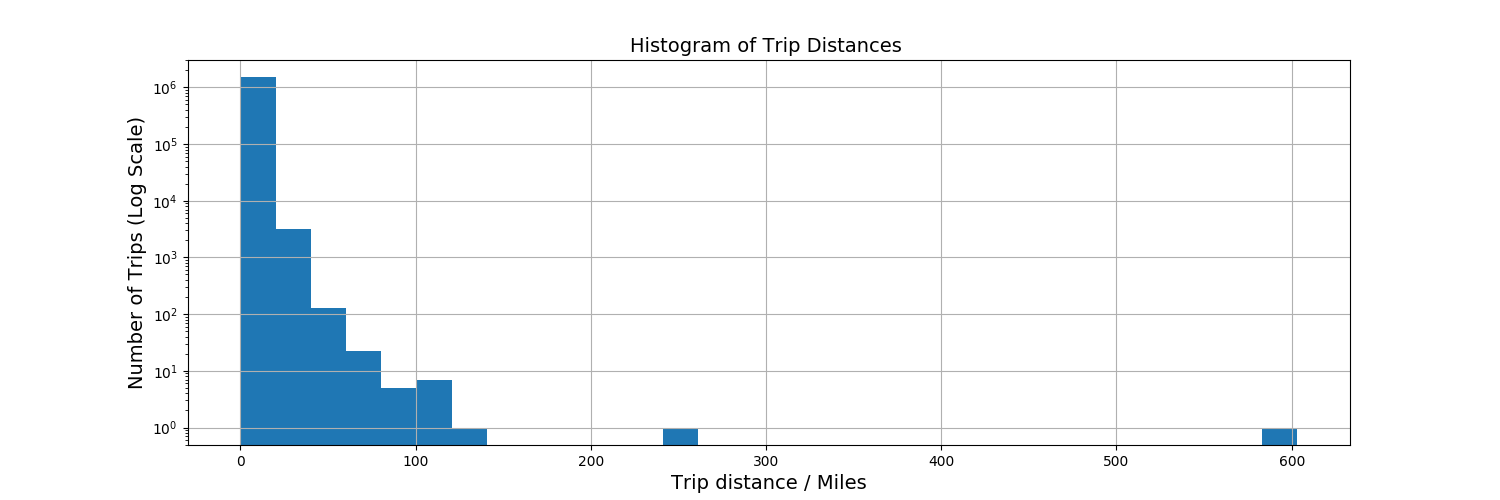


Figure 1. Number of trips with different trip distances

We display the number of trips as in log scale in order to see the small values.

In the figure, as we can see, there are two trips having trip distance of around 250 miles and around 600 miles, which are probably mis-record data. Also, based on the google map, it is hard to imagine a trip could be greater than 100 miles if the driver just drives inside the city.

So, before we move on, let's do the data cleansing to remove the corrupt or inaccurate data, to make our future inferences more reliable.

After a close inspection of the data, it is found that some pickup and dropoff locations are outside the New York city. So the first step is to limit these fields to the right scope. By checking with google map, the scope of New York City are:

1. # remove those not in new york area ===
2. data = data[(data.Pickup\_longitude >= -74.249) & (data.Pickup\_latitude >= 40.526) & (data.Pickup\_longitude <= -73.909) & (data.Pickup\_latitude <= 40.918) \
3. & (data.Dropoff\_longitude >= -74.249) & (data.Dropoff\_latitude >= 40.526) & (data.Dropoff\_longitude <= -73.909) & (data.Dropoff\_latitude <= 40.918)]

We also need to remove those trips with a too long distance, here let's set the maximum travel distance of a trip be 100 miles:

1. data = data2[(data2.Trip\_distance > 0) & (data2.Trip\_distance < 100)]

Let's plot the histogram of the trip distance after cleaning the data:

1. # It seems most of the trip distance are within the range of [0,30]
2. maxDistShow = 100
3. numBins = 100
4. plt.hist(data['Trip\_distance'][data['Trip\_distance']<maxDistShow],bins = numBins)
5. plt.title('Histogram of Trip Distances ( < 100 miles )', fontsize = 14)
6. plt.xlabel('Trip Distance', fontsize = 14)
7. plt.ylabel('Number of Trips', fontsize = 14)
8. plt.show()

The output is shown as below.

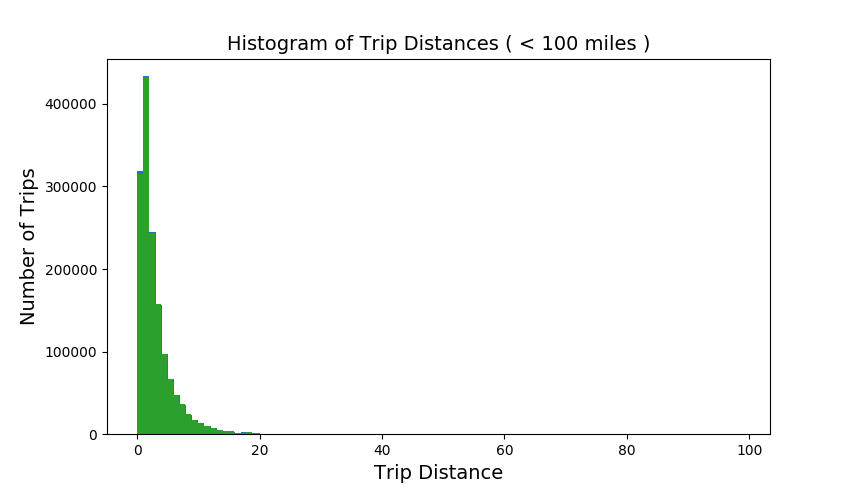


Figure 2. Number of trips with different trip distances

1. **Report any structure you find and any hypotheses you have about that structure.**

From the Figure 2 histogram, we can see it is a bell curve but skewed to the right. Most trips have a distance within 10 miles. To be precise, let's print out the number of trips within a range of distances:

1. # Find the number of trips with distance in [0,5], (6, 10] and (10, 15]
2. data.groupby(pd.cut(data['Trip\_distance'], np.arange(0,20,5))).size()

and the output is

1. Trip\_distance
2. (0, 5]      1231700
3. (5, 10]      188868
4. (10, 15]      39793
5. dtype: int64

It is seen that 85.4% of the trip travel less than 5 miles.

# Question 3

1. **Report mean and median trip distance grouped by hour of day.**
2. # Let’s strip the hour field from the pickup\_datetime field to create a new field named "pickup\_hour"
3. data['pickup'] = pd.to\_datetime(data['lpep\_pickup\_datetime'], format = '%Y-%m-%d %H:%M:%S')# apply(lambda x: datetime.strptime(x, '%Y-%m-%d %H:%M:%S'))
4. data['pickup\_hour'] = data['pickup'].apply(**lambda** x: x.hour) # this is a new field for holding the pickup\_hour.
6. # Do the same thing for dropoff\_datetime field to create a new field named "dropoff\_hour"
7. data['dropoff'] = pd.to\_datetime(data['Lpep\_dropoff\_datetime'], format = '%Y-%m-%d %H:%M:%S')
8. data['dropoff\_hour'] = data['dropoff'].apply(**lambda** x: x.hour) # this is a new field for holding the dropoff\_hour.
10. data[['Trip\_distance', 'pickup\_hour']].groupby('pickup\_hour').mean().plot.bar()
11. plt.title('Mean Trip Distance in One Day', fontsize = 14)
12. plt.xlabel('Pickup Time / 24H', fontsize = 14)
13. plt.ylabel('Trip Distance / Miles', fontsize = 14)
14. plt.show()
16. data[['Trip\_distance','pickup\_hour']].groupby('pickup\_hour').median().plot.bar()
17. plt.title('Median Trip Distance in One Day', fontsize = 14)
18. plt.xlabel('Pickup Time / 24H', fontsize = 14)
19. plt.ylabel('Trip Distance / Miles', fontsize = 14)
20. plt.show()

The mean and median trip distance in one day are plotted as below.

It is seen that the maximum hourly mean and median trip distance happens at 5:00am-6:00am, probably people need to get up early to work.

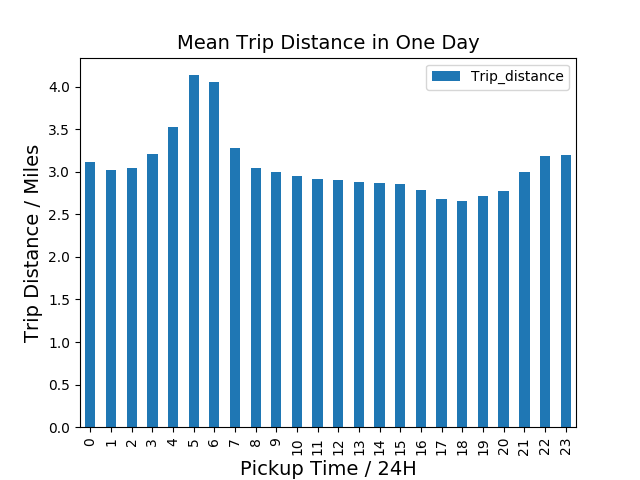


Figure 3. Average Trip distance during each hour of a day

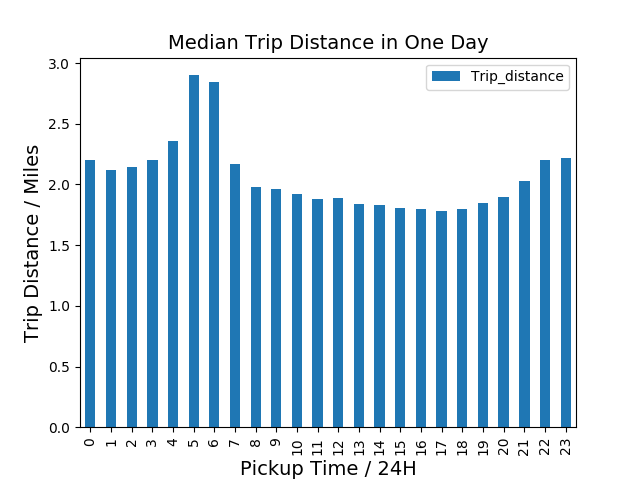


Figure 4. median TRIP DISTANCE DURING EACH HOUR OF A DAY

1. **We'd like to get a rough sense of identifying trips that originate or terminate at one of the NYC area airports. Can you provide a count of how many transactions fit this criteria, the average fare, and any other interesting characteristics of these trips.**

Based on the google map, the there are two airports in the city, namely the LaGuardia Airport (LGA) and the John F. Kennedy International Airport (JFK).

Let's first find the airport region's longitude and latitude, and define a function to tell whether the pickup/dropoff longitude and latitude are within that region,

1. # Define a function to tell whether the pickup/dropoff point is at an airport.
2. airportPos = {'JFK': [[40.646677, 40.666467], [-73.821884, -73.750296]], 'LGA': [[40.767550, 40.773098],[-73.884547, -73.865387]]}
3. **def** findAirport(row):
4. **if** (((row['Pickup\_longitude'] < airportPos['JFK'][1][1]) & (row['Pickup\_longitude'] > airportPos['JFK'][1][0]) &  # we got this directions from google maps
5. (row['Pickup\_latitude'] < airportPos['JFK'][0][1]) & (row['Pickup\_latitude'] > airportPos['JFK'][0][0])) |
6. ((row['Dropoff\_longitude'] < airportPos['JFK'][1][1]) & (row['Dropoff\_longitude'] > airportPos['JFK'][1][0]) &
7. (row['Dropoff\_latitude'] < airportPos['JFK'][0][1]) & (row['Dropoff\_latitude'] > airportPos['JFK'][0][0]))):
8. **return** 'JFK'  # John F. Kennedy International Airport
9. **if** (((row['Pickup\_longitude'] < airportPos['LGA'][1][1]) & (row['Pickup\_longitude'] > airportPos['LGA'][1][0]) &  # long and lat from google maps
10. (row['Pickup\_latitude'] < airportPos['LGA'][0][1]) & (row['Pickup\_latitude'] > airportPos['LGA'][0][0])) |
11. ((row['Dropoff\_longitude'] < airportPos['LGA'][1][1]) & (row['Dropoff\_longitude'] > airportPos['LGA'][1][0]) &
12. (row['Dropoff\_latitude'] < airportPos['LGA'][0][1]) & (row['Dropoff\_latitude'] > airportPos['LGA'][0][0]))):
13. **return** 'LAG'   # LaGuardia Airport
14. **else**:
15. **return** 'NOT' # Not an Airport pickup/dropoff

and then we create a new field named "Airport", by using the above defined function, gives whether a trip begins or ends at an airport.

1. # this is to create a new field in the dataframe based on the helper function written above.
2. data['Airport'] = data.apply(findAirport, axis=1)
4. # what's the distribution of the rides.
5. data['Airport'].value\_counts()
6. **print**('Average Fair for Trips To/From Airport: ', data[data['Airport']!= 'NOT']['Fare\_amount'].mean()) #average fare
7. **print**('Number of Trips To/From Airport: ', data[data['Airport']!='NOT']['Fare\_amount'].shape[0])

The number of trips fit this criteria and the average fare are:

1. Average Fair for Trips To/From Airport:  26.628824122630093
2. Number of Trips To/From Airport:  9916

Another interesting question to ask is, at what time during a day the number of pickups / drop-offs are the most, especially at the airport.

1. # how many customers in each hour in JFK airport ================
2. data[data['Airport']== 'JFK'].groupby(['pickup\_hour']).size().plot.bar() #
3. plt.title('Number of Customers During A Day From/To John Kennedy Airport', fontsize = 14)
4. plt.xlabel('Pickup Time / 24H', fontsize = 14)
5. plt.ylabel('Number of Customers', fontsize = 14)
6. plt.show()
8. # how many customers in each hour in LAG airport
9. data[data['Airport']== 'LAG'].groupby(['pickup\_hour']).size().plot.bar() #
10. plt.title('Number of Customers During A Day From/To LaGuardia Airport', fontsize = 14)
11. plt.xlabel('Pickup Time / 24H', fontsize = 14)
12. plt.ylabel('Number of Customers', fontsize = 14)
13. plt.show()
14. # how many customers in each hour in Non airport airport
15. data[data['Airport']== 'NOT'].groupby(['pickup\_hour']).size().plot.bar() #
16. plt.title('Number of Customers During A Day From & To Non-airport Area', fontsize = 14)
17. plt.xlabel('Pickup Time / 24H', fontsize = 14)
18. plt.ylabel('Number of Customers', fontsize = 14)
19. plt.show()

The histogram is shown as below.

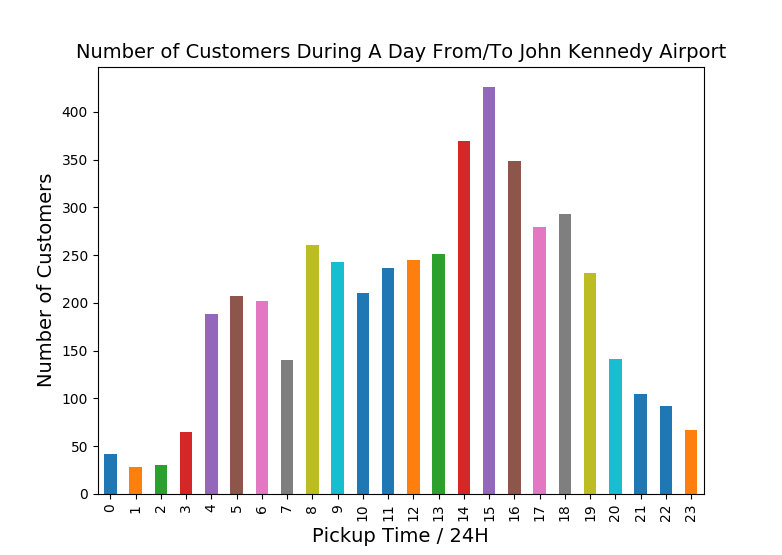


Figure 5. number of customers during a day (from/to JFK)

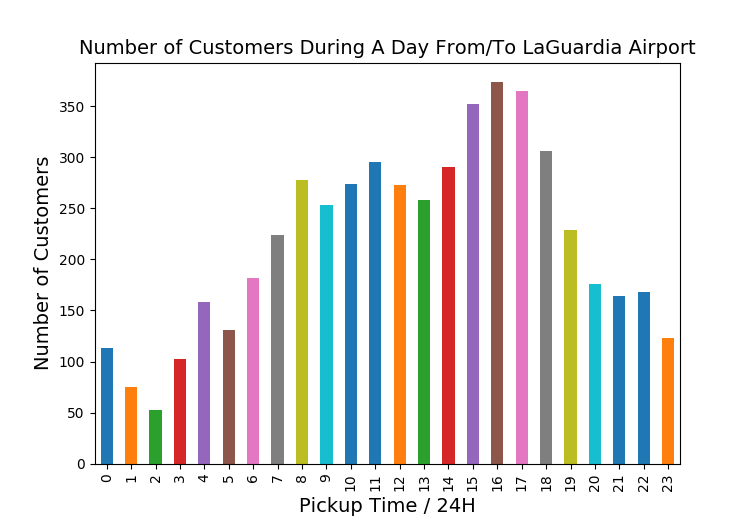


Figure 6. number of customers during a day (from/to LGA)

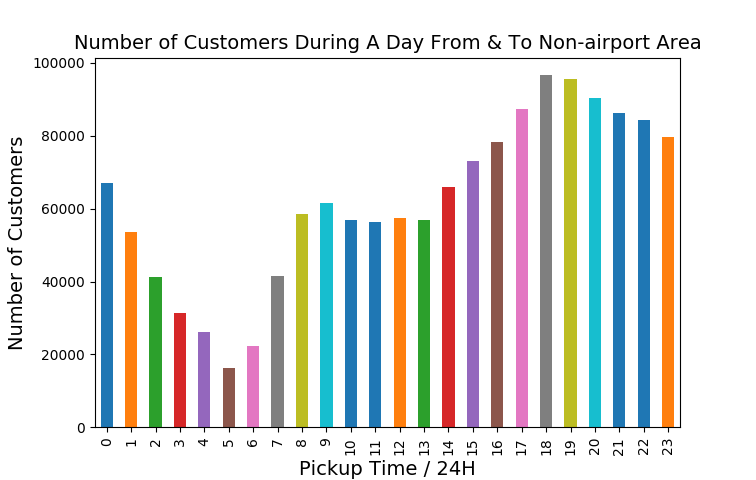


Figure 7. number of customers during a day (Not-airport area)

As we can see, there is a clear pattern from the histogram.

For JFK airport, the number of trips reaches its maximum during 2:00pm-4:00pm.

For LGA airport, the number of trips reaches its maximum during 3:00pm-6:00pm.

For non-airport area, the number of trips reaches its maximum during 6:00pm-9:00pm.

If we have access to the flight schedules in each airport, it is natural to assume this pattern aligns with the schedule: there are almost no flights in the midnight, there are a number of flights take off or arrive during the morning, and the number reaches its maximum in the afternoon.

It also seems that LGA airport should have more flights compared to JFK airport.

For the non-airport area, it is interesting to see that people take taxis during the midnight: the number of trips at 12:00AM is even larger than every hour in the morning and even part of the afternoon (12:00pm-2:00pm). It seems in the New York City a lot of people go out or back home in the midnight. Maybe to relax after a whole day's work under high pressure?

Another thing interests me is the number of trips at different time of a day and across different days during a week:

1. # =========== By each day of a week ==================
2. data['pickup\_day'] = pd.to\_datetime(data['lpep\_pickup\_datetime'], format = '%Y-%m-%d %H:%M:%S').apply(**lambda** x: x.day) # this is a new field for holding the pickup\_hour.
3. data['dropoff\_day'] = pd.to\_datetime(data['Lpep\_dropoff\_datetime'], format = '%Y-%m-%d %H:%M:%S').apply(**lambda** x: x.day) # this is a new field for holding the pickup\_hour.
5. hour\_pickups = []
6. temp = []
7. **for** i **in** range(1,8):
8. **for** j **in** range(0,24):
9. temp.append(data[(data.pickup\_day == i) & (data.pickup\_hour == j) & (data['Airport'] == 'NOT')].shape[0])
10. hour\_pickups.append(temp)
11. temp = []
12. colors = ['xkcd:blue','xkcd:orange','xkcd:brown','xkcd:coral','xkcd:magenta','xkcd:green','xkcd:fuchsia']
13. days = ['Monday','Tuesday','Wednesday','Thursday','Friday','Saturday','Sunday']
15. plt.figure(figsize=(8,4))
16. hours\_lis = [s **for** s **in** range(0,24)]
17. **for** k **in** range(0,7):
18. plt.plot(hours\_lis,hour\_pickups[k],colors[k],label = days[k])
19. plt.plot(hours\_lis,hour\_pickups[k], 'ro',  markersize=2)
21. plt.xticks([s **for** s **in** range(0,24)])
22. plt.xlabel('Hours of a day')
23. plt.ylabel('Number of pickups')
24. plt.title('Pickups for every hour')
25. plt.legend()
26. plt.grid(True)
27. plt.show()

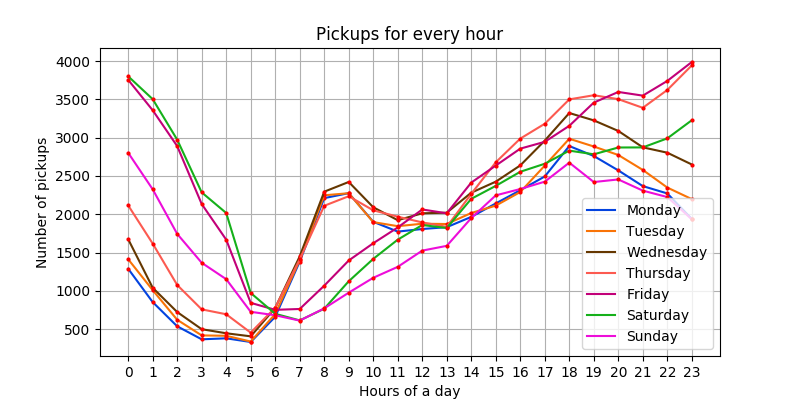


Figure 8. Number of pickups during a day for a week

First, notice that again, a lot of trips happened at midnight. Take a look at the records at 12:00AM, of all the seven days, the highest three days are the Friday, Saturday and Sunday. But people are less active at midnight during Monday through Thursday, maybe because they have to go to bed earlier since they know on the next day they have to work.

Another thing catches eye quickly is the peak happens at 8:00am-9:00am for Monday through Thursday. One explanation might be because people take taxis to work in the morning, since it aligns with starting time of most works (8:00am-9:00am).

The only unexplainable part is Friday, if work is the reason for the peak, then why on Friday people do not take taxi to work in the morning, like they do on Monday-Thursday? Or they do not work on Friday? We might need other data, such as the working hours for normal jobs, to explain this phenomenon.

Since we already have the airport information, it is also interesting to see usually how many passengers on the trips to or from airport.

1. # how many passengers on each trip (To/from JFK airport)  ================
2. data[data['Airport']== 'JFK'].groupby(['Passenger\_count']).size().plot.bar() #
3. plt.title('Number of Passengers on Each Trip From/To John Kennedy Airport', fontsize = 14)
4. plt.xlabel('Trips Number', fontsize = 14)
5. plt.ylabel('Number of Passengers', fontsize = 14)
6. plt.show()
8. data[data['Airport']== 'LAG'].groupby(['Passenger\_count']).size().plot.bar() #
9. plt.title('Number of Passengers on Each Trip From/To LaGuardia Airport', fontsize = 14)
10. plt.xlabel('Trips Number', fontsize = 14)
11. plt.ylabel('Number of Passengers', fontsize = 14)
12. plt.show()
14. data[data['Airport']== 'NOT'].groupby(['Passenger\_count']).size().plot.bar() #
15. plt.title('Number of Passengers on Each Trip From & To Non-airport Area', fontsize = 14)
16. plt.xlabel('Trips Number', fontsize = 14)
17. plt.ylabel('Number of Passengers', fontsize = 14)
18. plt.show()

The histogram is as follows:

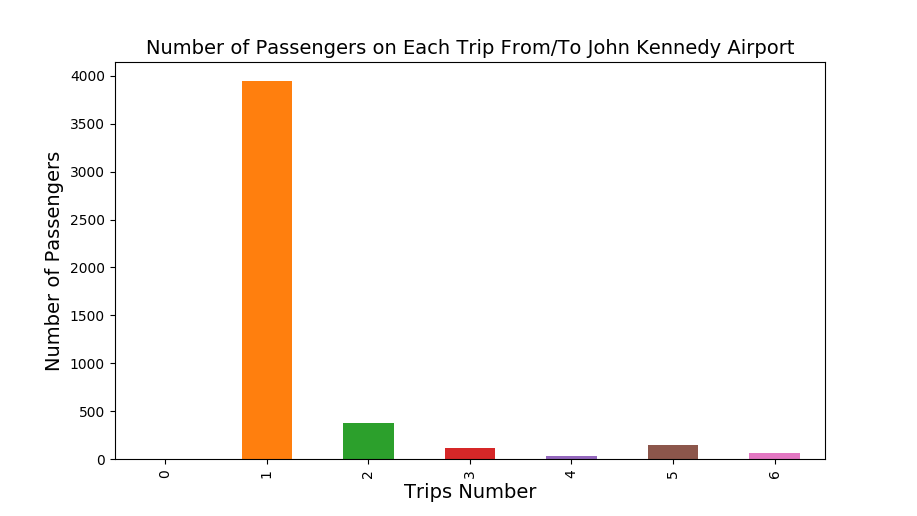


Figure 9 (a)

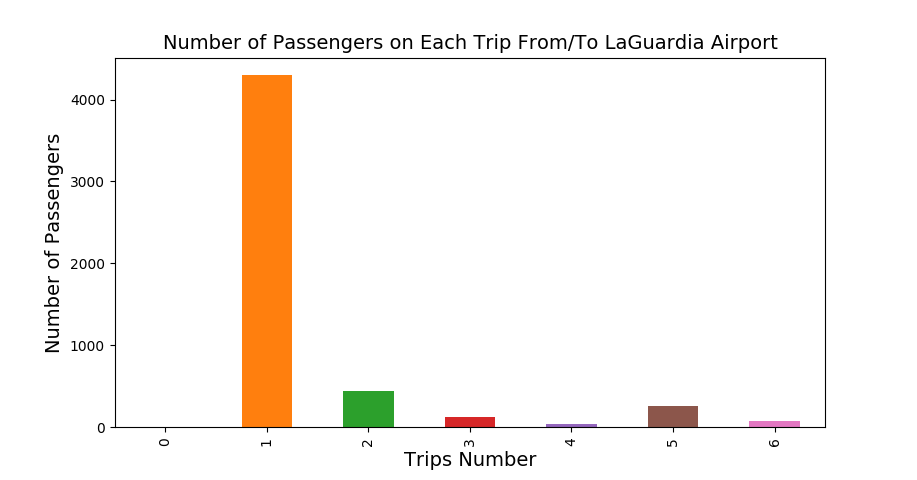


Figure 9 (B)

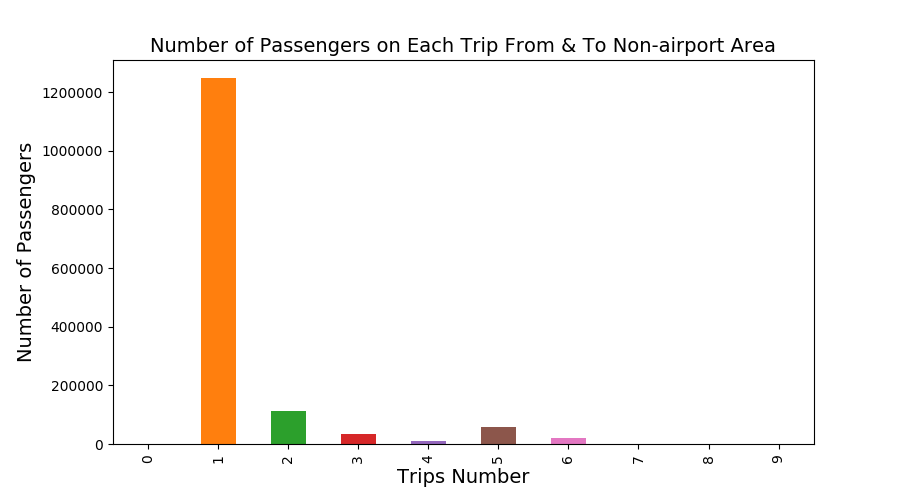


Figure 9 (c)

Figure 9. Number of passengers on each trip

It seems that no matter for the two airports, or not-airport area, most (85.70%) of the trips would take only one passenger.

Based on the histogram, it's difficult to imagine, even a limousine, could take more than 6 passengers, since the room on the third row in a limousine are quite compact actually. On the NYC taxi & Limousine Commission webpage, the green taxi and limousine are of the following size. But still, trips with 6~9 passengers can be found in the data.



This bring to the question that whether the number of passengers is a reliable field?

For all the other data, such as "pickup\_datatime", "dropoff\_datatime", "pickup\_longtide", "dropoff\_latitude", etc. these data are all collected automatically by the GPS. And data such as "Total\_amount", "Payment\_type", "Tip\_amount", etc. are all collected via the POS machine equipped with the taxi.

However, I cannot think of a device in the taxi that has the function of detecting the number of passengers on a ride. The number of passengers might be recorded by human (the driver), maybe by writing it down on paper, or more likely, by just tapping a number on some touch-screen of a electronic device.

This brings the question whether the passenger number information is reliable. If there are more than 1 passenger, does he bother to count the people or remember the number correctly. Or a lazy driver may just input 1 or a random number on the 1~9 number pad. It also explains why there does not exist a trip with 10 or more than 10 passengers, since tapping two digits are too much work to do than tapping one digit?

Another thing to loot at is twhat payment types people like to use? From the data description, the "Payment\_type" has 6 values, interpreted as:

1. """Payment types:
2. 1. Credit card
3. 2. Cash
4. 3. No charge
5. 4. Dispute
6. 5. Unknown
7. 6. Voided trip
8. """

Let's plot the histogram of number of trips using different payment types.

1. payment\_uniq = set(data['Payment\_type'].values)
2. payment\_count = []
3. **for** i **in** payment\_uniq :
4. payment\_count.append(data[data['Payment\_type'] == i].shape[0])
5. **print** (payment\_count)
6. payment\_uniq = list(payment\_uniq)
7. **print** (payment\_uniq)
8. pay\_pickups = [x **for** \_,x **in** sorted(zip(payment\_uniq,payment\_count))]
9. pay = [int(y) **for** y,\_ **in** sorted(zip(payment\_uniq,payment\_count))]
10. pay\_type = ['Credit card','Cash','No charge','Dispute','Unknow']
12. sns.set\_style("whitegrid")
14. fig, ax = plt.subplots()
15. fig.set\_size\_inches(12, 5)
17. ax = sns.barplot(x = np.array(pay\_type) , y = np.array(pay\_pickups))
18. ax.set(ylabel='Pickups',xlabel = 'payment types')
19. sns.plt.title('Number of pickups for each payment types')
20. plt.show()

It's a little bit surprising to see more than half people choose to pay with cash, whereas it's easier to pay by swiping/inserting a card.

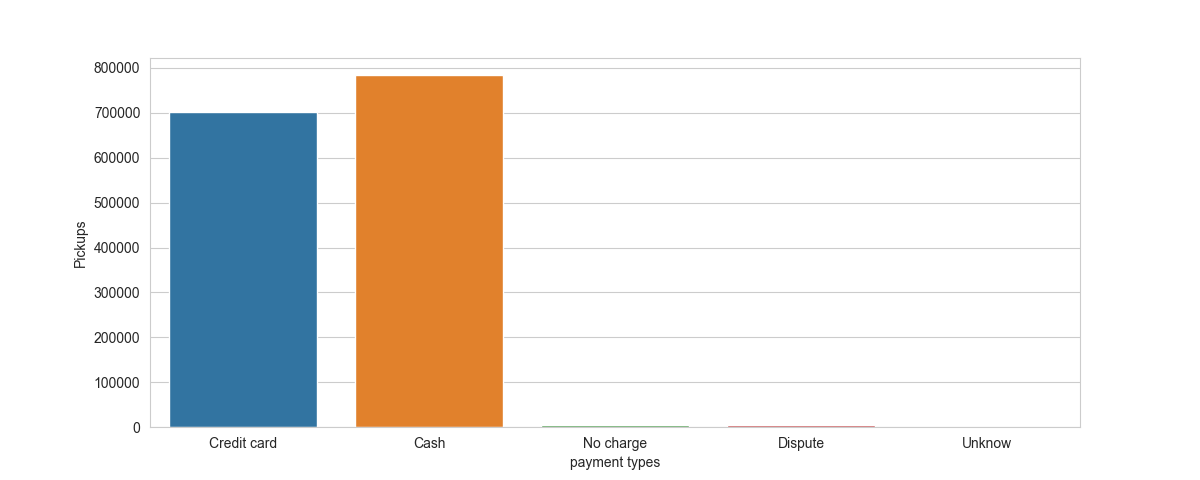


Figure 10. number of trips with different payment method

A further question is that, does people who use credit card pay more tips than those people pay with cash?

The reason why it is interesting is that, I have once read an article about the psychology behind people when they pay tips (<http://www.bbc.com/future/story/20160708-the-mind-tricks-to-get-better-tips>), they draw a conclusion in the article that "The method of payment can also make a difference. People [tend to tip more on credit cards](https://www.uni-muenster.de/imperia/md/content/psyifp/aeechterhoff/wintersemester2011-12/vorlesungkommperskonflikt/feinberg_credcards_jouconres1986.pdf). "

1. # when pay with credit card, the number of trips with different tip amount
2. data2 = data[(data['Payment\_type'] == 1)]
3. data2.groupby(pd.cut(data2['Tip\_amount'], np.arange(-1, 30, 2)))['Tip\_amount'].size().plot.bar()
4. plt.title('Number of Passengers on Each Trip From/To LaGuardia Airport', fontsize = 14)
5. plt.xlabel('Trips Number', fontsize = 14)
6. plt.ylabel('Number of Passengers', fontsize = 14)
7. plt.show()

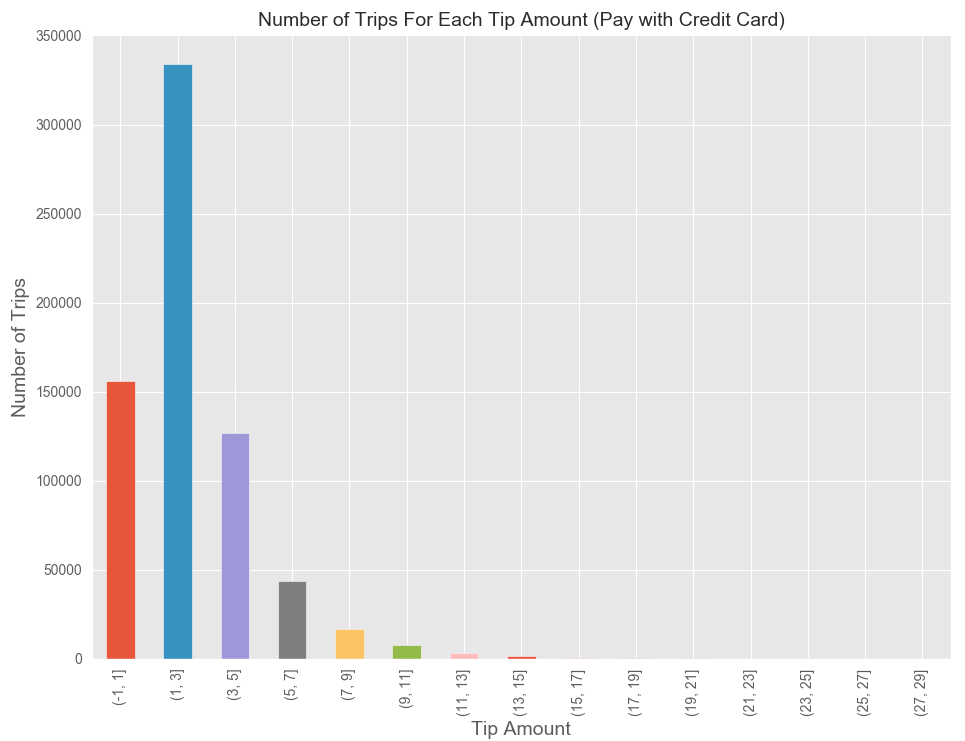


Figure 11. Number of trips with different amount of tips (credit card)

However, when I am trying to plot the histogram for pay with cash, I got the following histogram.

1. data2 = data[(data['Payment\_type'] == 2)] # pay with cache
2. data2.groupby(pd.cut(data2['Tip\_amount'], np.arange(-1, 30, 2)))['Tip\_amount'].size().plot.bar()
3. plt.title('Number of Trips For Each Tip Amount (Pay with Cash)', fontsize = 14)
4. plt.xlabel('Trips Number', fontsize = 14)
5. plt.ylabel('Number of Passengers', fontsize = 14)
6. plt.show()

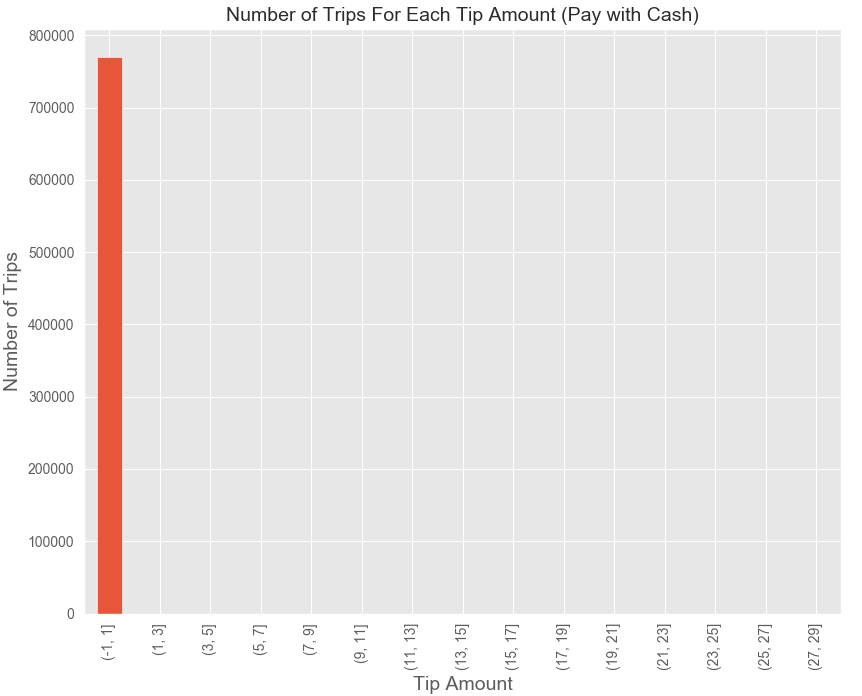


Figure 12. Number of trips with different amount of tips (cash)

It seems that when passenger pay by cash, they all pay zero tips? No.

Again, this brings the question whether the tip data is reliable when people pay by cash. Same inference as above for the number of passengers:

It is more likely for those passengers pay tip by cash, the taxi driver would not bother to input how much tips were given. Whereas for credit card payment, the tip amount is recorded into the system automatically.

Unfortunately, it seems we are unable to verify the statement in that article about tips using this data.

# Question 4

1. **Build a derived variable for tip as a percentage of the total fare.**
2. data['tip\_percent'] = (data['Tip\_amount']/data['Total\_amount']).apply(**lambda** x: x \* 100) # calculate tip percentage
4. grouped\_df = data.groupby(['pickup\_hour', 'Airport'])['tip\_percent'].aggregate(np.mean).reset\_index() #average tip
6. plt.figure(figsize=(12, 8))
7. sns.pointplot(grouped\_df.pickup\_hour.values, grouped\_df.tip\_percent.values, grouped\_df.Airport.values, alpha=0.8)
8. plt.ylabel('Average Tip Percentage of the Fare / USD', fontsize = 14)
9. plt.xlabel('Pick-Up Time / 24H', fontsize = 14)
10. plt.title('Tip Amount During A Day', fontsize = 14)
11. plt.xticks(rotation='vertical', fontsize = 14)
12. plt.show()

The result is:

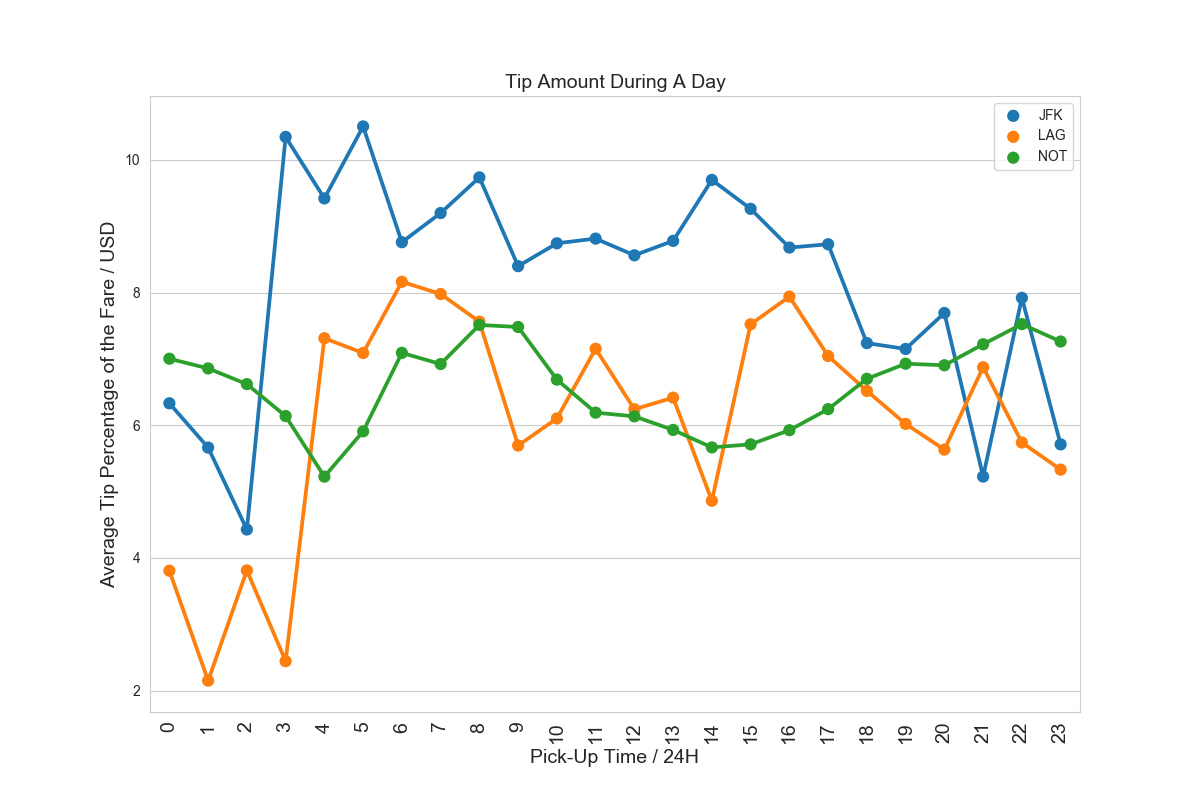


Figure 13. Tip amount during a day for airport/not-airport area

For not-airport trips (green line), the tip amount seems not change much during a day.

For airport trips, it seems that people are more generous during the very early morning (3:00am – 7:00am), but very stingy during midnight (23:00pm – 2:00am).

1. **Build a predictive model for tip as a percentage of the total fare. Use as much of the data as you like (or all of it). Provide an estimate of performance using an appropriate sample, and show your work.**
2. **import** xgboost as xgb
3. **import** lightgbm as lgb
5. df\_model = pd.read\_csv("/Users/Kai/Dropbox/Project\_capitalOne/green\_tripdata\_2015-09.csv", parse\_dates=['lpep\_pickup\_datetime'])
7. df\_model = df\_model.reset\_index() # reset the index of our dataframe.
8. df\_model.rename(columns={'index':'ID'}, inplace=True) # my
10. df\_model.drop(['Ehail\_fee', 'RateCodeID', 'Extra'], axis=1, inplace=True) #drop these features, because they're either all 0's or Nan's
11. display(df\_model) # let's see how our data looks like. The ID field is helpful during the prediction stage)
13. df\_model['tip\_percent'] = df\_model['Tip\_amount']/df\_model['Total\_amount'] # calculate the tip percent
14. df\_model['tip\_percent'] = df\_model['tip\_percent'].apply(**lambda** x: x \* 100) # multiply the value by 100
15. df\_model = df\_model[df\_model['Tip\_amount'] > 0] # make sure the tip is greater than zero
16. df\_model = df\_model[df\_model['Fare\_amount'] > 0] # make sure the fare amount is greater than zero
17. # Train-test split
18. y = df\_model.tip\_percent # tip\_percent is our target variable
19. X = df\_model # predictor varibles
20. X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2) # let’s use 80%-20% split.
21. **print** ( "\nX\_train:\n")
22. **print** (X\_train.shape)
23. **print** ("\nX\_test:\n")
24. **print** (X\_test.shape)

The result is, there are 482052 records in the training set and 120514 records in the testing set.

1. X\_train:
2. (482052, 20)
3. X\_test:
4. (120514, 20)

For the test set, let's drop the tip\_amount field, otherwise the problem becomes trivial: the training model can just use the tip amount to calculate the tip percent. In the model, RMSLE is chosen instead of RMSE, so let's create a new field filled with log(tip\_percent).

Let's take a look at the distribution of the log(tip\_percent).

1. X\_test.shape # test data shape
2. X\_test.head() # test data
3. X\_test.drop(['Tip\_amount'], axis=1, inplace=True)
4. X\_train['log\_tip\_percent'] = np.log1p(X\_train['tip\_percent'].values) # logarithm of the tip\_percent, because we use RMSLE
5. X\_test.shape # test data shape
6. X\_test.head() # test data
8. plt.figure(figsize=(8,6))
9. plt.scatter(range(X\_train.shape[0]), np.sort(X\_train.tip\_percent.values))
10. plt.xlabel('index', fontsize=12)
11. plt.ylabel('log\_tip\_percent', fontsize=12)
12. plt.show()

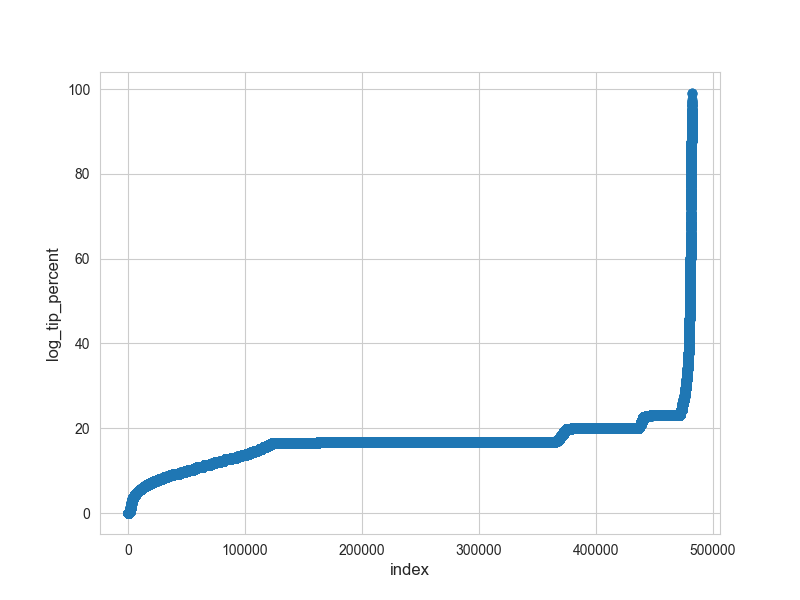


Figure 14. logrithm of tip percent

Let us write a helper function to run the xgboost model and light gbm model. Also, drop the fields that we the model doesn't need. Some codes of this part are attached but not shown in this report. Only the main steps are shown.

Let us use the K-fold cross validation to build the model, and we set the number of rounds to be 5000.

1. kf = model\_selection.KFold(n\_splits=10, shuffle=True, random\_state=2017)
2. cv\_scores = []
3. pred\_test\_full = 0
4. pred\_val\_full = np.zeros(X\_train.shape[0])
5. **for** dev\_index, val\_index **in** kf.split(train\_X):
6. dev\_X, val\_X = train\_X.ix[dev\_index], train\_X.ix[val\_index]
7. dev\_y, val\_y = train\_y[dev\_index], train\_y[val\_index]
8. pred\_val, pred\_test, model = runLGB(dev\_X, dev\_y, val\_X, val\_y, test\_X, num\_rounds=5000, num\_leaves=10, max\_depth=8, eta=0.3)
9. pred\_val\_full[val\_index] = pred\_val
10. pred\_test\_full += pred\_test
11. cv\_scores.append(np.sqrt(metrics.mean\_squared\_error(val\_y, pred\_val)))
12. **print**(cv\_scores)
13. **print**("Mean RMSE score : ",np.mean(cv\_scores))
15. pred\_test\_full = pred\_test\_full / 5.
16. pred\_test\_full = np.expm1(pred\_test\_full)
17. pred\_val\_full = np.expm1(pred\_val\_full)
19. # saving train predictions for ensemble #
20. train\_pred\_df = pd.DataFrame({'ID':train\_id})
21. train\_pred\_df['tip\_percent'] = pred\_val\_full
22. train\_pred\_df.to\_csv("train\_preds\_lgb\_baseline.csv", index=False)
24. # saving test predictions for ensemble #
25. test\_pred\_df = pd.DataFrame({'ID':test\_id})
26. test\_pred\_df['tip\_percent'] = pred\_test\_full
27. test\_pred\_df.to\_csv("test\_preds\_lgb\_baseline.csv", index=False)

And the last few lines of running results are:

1. ...
2. Training until validation scores don't improve **for** 100 rounds.
3. [20]    valid\_0's rmse: 0.360859
4. [40]    valid\_0's rmse: 0.360897
5. [60]    valid\_0's rmse: 0.360904
6. [80]    valid\_0's rmse: 0.36092
7. [100]   valid\_0's rmse: 0.361005
8. Early stopping, best iteration **is**:
9. [1] valid\_0's rmse: 0.360801
10. [0.35964487240921955, 0.35863310008231025, 0.35301895321169874, 0.3571963165220173, 0.34760971475170216, 0.35611596351967423, 0.3541931963927481, 0.35467810156786067, 0.3547784135226888, 0.3608007979054879]
11. Mean RMSE score :  0.35566694298854074

The result is shown as below:

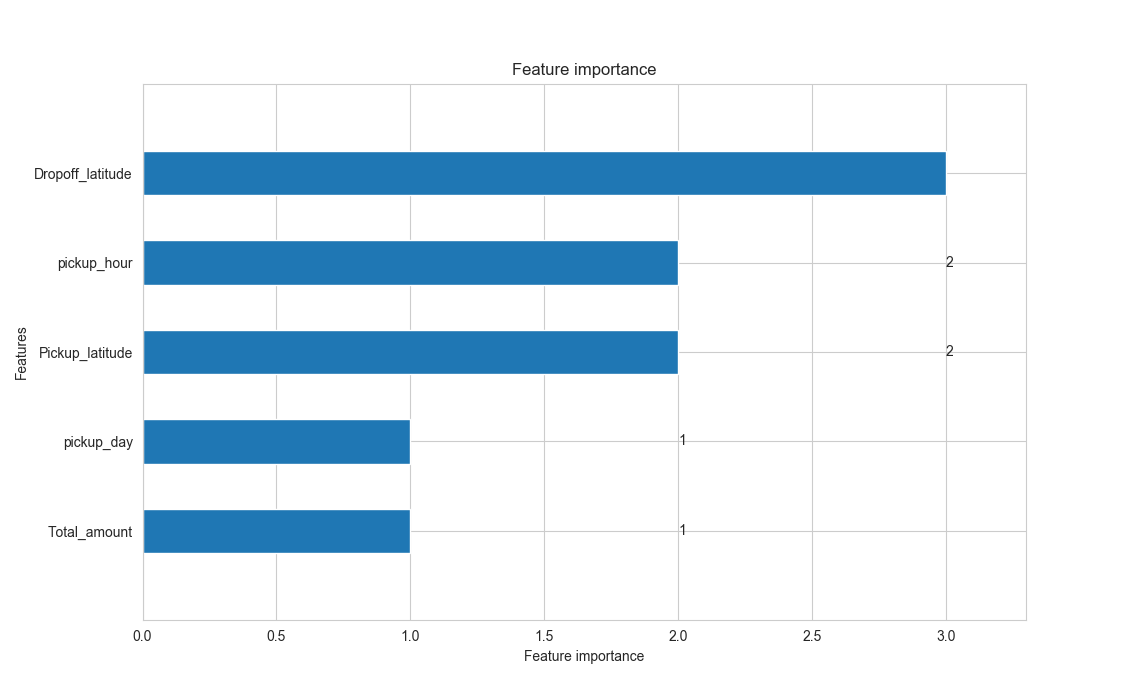


Figure 15. importance rate of different factors to tips

The mean RMSE score is not a perfect score and can be improved by a more careful tuning of our parameters. From the current result, we can see the dropoff latitude has the most influence on the tip amount, then the pickup\_hour and the pickup latitude.

This result aligns with our result of Figure 13, as we can see there, the tip amount is very different for two airports area and non-airport area, which has different latitude. Then the second factor is pickup hour, when people tend to pay less in midnight and the most in the around the morning time.

# Question 5

1. **Option A: Distributions**

**Build a derived variable representing the average speed over the course of a trip.**

**Can you perform a test to determine if the average trip speeds are materially the same in all weeks of September? If you decide they are not the same, can you form a hypothesis regarding why they differ?**

**Can you build up a hypothesis of average trip speed as a function of time of day?**

1. data['travel\_time'] = (data['dropoff'] - data['pickup']).apply(**lambda** x: x.total\_seconds())
2. data['average\_speed'] = 3600\*(data['Trip\_distance']/data['travel\_time'])
3. data['week'] = data['dropoff'].apply(**lambda** x: x.week) # extract week of year
4. data['week'].value\_counts() # week count
5. week\_1 = data['average\_speed'][data['week']==36].as\_matrix() # reassign week=36 to week\_1 df
6. week\_2 = data['average\_speed'][data['week']==37].as\_matrix() # reassign week=37 to week\_2 df
7. week\_3 = data['average\_speed'][data['week']==38].as\_matrix() # reassign week=38 to week\_3 df
8. week\_4 = data['average\_speed'][data['week']==39].as\_matrix() # reassign week=39 to week\_4 df
9. week\_5 = data['average\_speed'][data['week']==40].as\_matrix() # reassign week=40 to week\_5 df
11. stats.f\_oneway(week\_1,week\_2, week\_3,week\_4, week\_5)

The output is

1. F\_onewayResult(statistic=1072.6164959451003, pvalue=0.0)

The large F value and small p value indicates we should reject the null hypothesis. The conclusion is that the differences between the groups are statistically significant. This 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.

1. **print**(week\_1.mean(),week\_2.mean(),week\_3.mean(),week\_4.mean(),week\_5.mean()) **print**(statistics.median(week\_1),statistics.median(week\_2),statistics.median(week\_3),statistics.median(week\_4),
2. statistics.median(week\_5))plt.rcParams["figure.figsize"] = [20,12]
3. plt.subplot(3,2,1)
4. plt.hist(week\_1,bins = 50,label = 'week 1')
5. plt.legend()
6. plt.subplot(3,2,2)
7. plt.hist(week\_2,bins = 50,label = 'week 2')
8. plt.legend()
9. plt.subplot(3,2,3)
10. plt.hist(week\_3,bins = 50,label = 'week 3')
11. plt.legend()
12. plt.subplot(3,2,4)
13. plt.hist(week\_4,bins = 50,label = 'week 4')
14. plt.legend()
15. plt.subplot(3,2,5)
16. plt.hist(week\_5,bins = 50,label = 'week 5')
17. plt.legend()
18. plt.legend()
19. plt.savefig('task5')
20. plt.show()

The mean, median and histograms are as follows:

1. **print**(week\_1.mean(),week\_2.mean(),week\_3.mean(),week\_4.mean(),week\_5.mean())
3. 13.372815533370655 12.710767171035164 12.701577549723268 13.179017685453339 12.45675912192726
5. **print**(statistics.median(week\_1),statistics.median(week\_2),statistics.median(week\_3),statistics.median(week\_4),
6. statistics.median(week\_5))
8. 12.128024390243901 11.616438356164386 11.63855421686747 12.028639618138424 11.421461897356142

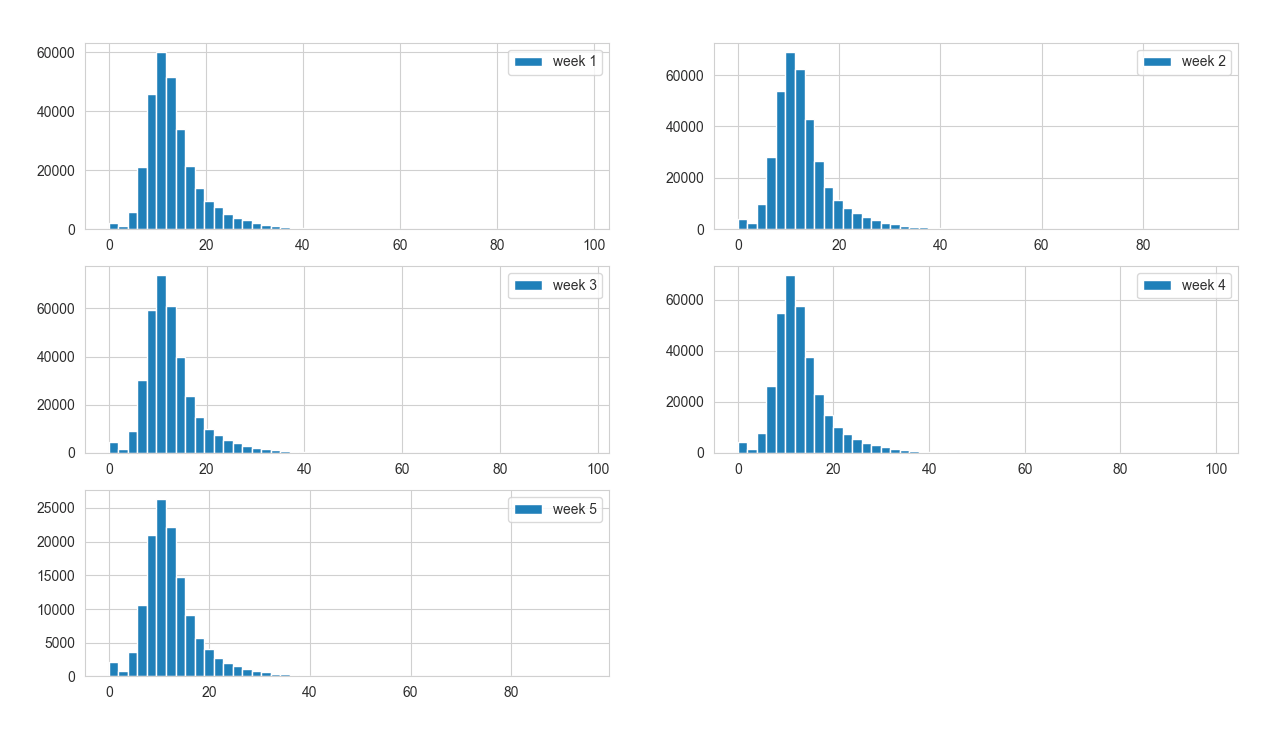


Figure 16. average trip speed for each week in september

Notice that most of the trips have an average speed between 8-15mph, this make sense considering the traffic in the New York City.

At week 1 and 5, the number of trips at each speed level is fewer compared to the week 2,3,4, because week 1 has 5 days and week 5 has only 4 days.

1. grouped = data.groupby('pickup\_hour') # group by the hour
2. samples = []
4. **for** name,group **in** grouped:
5. samples.append(group['average\_speed']) # append the avg speed data

8. sample = samples
9. stats.f\_oneway(sample[0],sample[1],sample[2],sample[3], sample[4],sample[5],sample[6],sample[7],sample[8],sample[9],
10. sample[10],sample[11],sample[12],sample[13],sample[14],sample[15],sample[16],sample[17],sample[18],
11. sample[19],
12. sample[20],sample[21],sample[22],sample[23])

The output is:

1. F\_onewayResult(statistic=5118.296219575005, pvalue=0.0)

The ANOVA test, again, outputs a high f-value and p-value of 0, implying that there are statistifically significant differences for sets partitioned as per the hour of the journey.

1. means = [] # empty list for storing the mean info
2. medians = [] # empty list for storing the median info
3. **for** hour **in** range(24):
4. means.append(statistics.mean(sample[hour]))
5. **print**('Mean:',statistics.mean(sample[hour]))
6. medians.append(statistics.median(sample[hour]))
7. **print**('Median:',statistics.median(sample[hour]))
9. plt.style.use('ggplot')
11. plt.bar(range(1,25), means, color='blue')
12. plt.xlabel("Hour of Day", fontsize=15)
13. plt.ylabel("Average Speed (mi/hr)", fontsize=15)
14. plt.title("Average speed of a ride at the hour")
16. plt.show()

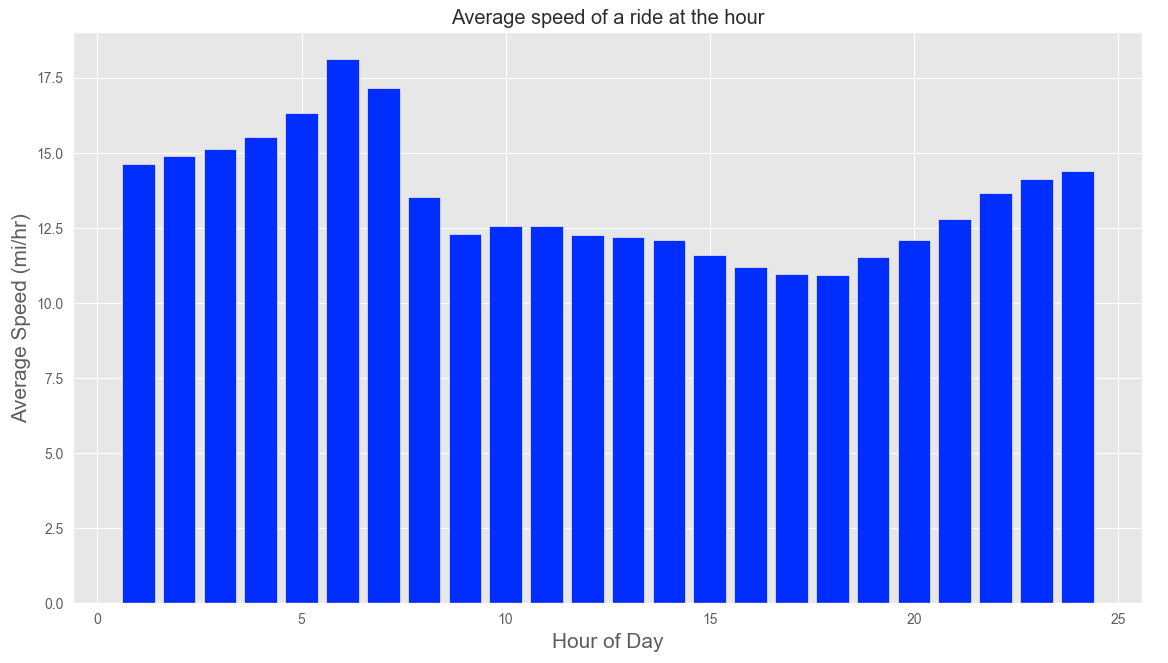


Figure 17.average speed during each hour of a day

References

1. [Analyzing 1.1 Billion NYC Taxi and Uber Trips, with a Vengeance](http://toddwschneider.com/posts/analyzing-1-1-billion-nyc-taxi-and-uber-trips-with-a-vengeance/)
2. [NYC Taxi Trips - Exploratory Data Analysis | Kaggle](https://www.kaggle.com/kartikkannapur/nyc-taxi-trips-exploratory-data-analysis)
3. [nyc-taxi-data/analysis/2017\_update at master · toddwschneider/nyc ...](https://github.com/toddwschneider/nyc-taxi-data/tree/master/analysis/2017_update)
4. [Analyze the NYC Taxi Data | An Explorer of Things](https://chih-ling-hsu.github.io/2018/05/14/NYC)
5. [The Data Science of NYC Taxi Trips: An Analysis & Visualization](https://www.kdnuggets.com/2017/02/data-science-nyc-taxi-trips.html)
6. [Analyzing 1.2 Billion NYC Taxi Rides – Jay Gopalakrishnan – Medium](https://medium.com/@gopalaj61/analyzing-1-2-billion-nyc-taxi-rides-83ea8012827e)
7. [NYC Taxi Duration Exploratory Data Analysis - Amazon AWS](https://rstudio-pubs-static.s3.amazonaws.com/298073_df18c75025884a389091278b481781fe.html)
8. [Analyzing New York City taxi data using big data tools](https://developers.arcgis.com/python/sample-notebooks/analyze-new-york-city-taxi-data/)
9. [Final Project: NYC Taxi Data – Drew Levitt - Open Computing Facility](https://www.ocf.berkeley.edu/~dlevitt/2015/12/13/final-project-nyc-taxi-and-uber-data/)
10. [NYC Taxi Trip and Fare Data Analytics using BigData](https://www.ocf.berkeley.edu/~dlevitt/2015/12/13/final-project-nyc-taxi-and-uber-data/)
11. [Exploratory Analysis Of New York City Yellow Taxi Data](https://nycdatascience.com/blog/student-works/new-york-city/)