CIMS CUSP SPS

DS-GA 1004-001 Big Data Spring 2015

Understanding driver behavior and taxi usage

Author:
Arpit Jain
Jiamin Xuan
Xiaohui Guo

Supervisor:
Prof. Juliana Freire

May 11, 2015

NEW YORK UNIVERSITY

Contents

List of Figures

List of Tables

Abstract

New York City taxicabs are widely recognized icons of the city. It is one of the major mode of transport and a lot of people rely on them for daily commute between there work places and residence. As per 2014 Taxicab fact book there more than 485,000 trips per day and 175 million per year carrying 600,000 passengers per day. All these trips are logged along with other information like trip cost, pickup drop-off locations etc. This increasing amount of data motivates data scientists to analyze this data and come up with strategies and policies to improve the safety and living standards of citizens. In this project we will explore one such prospect of this huge data from taxi trips to understand the behavior of taxi drivers and the usage of taxis.

1 Introduction

New York City taxicab is an import factor in the lives of both the passengers and drivers. With more than 600,000 passengers served each day by nearly 50,000 drivers, NYC taxi not only help passengers reach there destination but also provide employment for drivers and other related people. Passengers of different income groups ranging from under $ 10k to more than $100k and age group ranging from 20-35 to above 70 rely on taxicabs for there daily commute[1]. Taxis are divided into two categories: yellow and green. Yellow are able to pick up passengers from anywhere in five boroughs and green are able to pick up from Upper Manhattan, the Bronx, Brooklyn, Queens (excluding LaGuardia Airport and John F. Kennedy International Airport), and Staten Island[2]

Taxis serve as sensors for collecting useful information about the trips and fares. Analyzing this information can provide great insight in the economic activity to behaviors of both travelers and drivers. Its an interesting endeavor to combine this information with other open datasets available about the city to find new correlations and eventually use these associations to make better decisions.

In this project we will analyze NYC taxi dataset [3] made available by Taxi and Limousine Commission of New York City to understand driver behavior. We will use data to find if there are some drivers who take longer routes than normal and drivers who have similar driving patterns. We will also combine census tract [4] and weather [5] to further enhance our analysis. Along with behavior we will also analyze the distribution of taxis and how this distribution varies for different regions.

In the next section we will briefly describe the dataset and tools used. In Section 3 we will discuss the experiments and techniques used; in Section 4 we will discuss results and Section 5 will conclude this report with conclusion and future possibilities.

2 Data and tools

2.1 Dataset

New York City Taxi Data: Dataset used in this project is provided by Taxi and Limousine Commission of New York City. It contains trips and fair information for 2010-2013. The data is nearly 40 GB and contain more than 500 million trips. It is divided into two table *Trip* and *Fare*. Below are the attributes in each table. Medallion and hack licenses are hashed in order to preserve privacy of actual medallions and drivers. Both the tables have four fields in common

Table 1: Trips Table Attributes

Attribute Name	Type
medallion	string
hack_ license	string
vendor_id	integer
rate_code	integer
store_and_fwd_flag	boolean
pickup_datetime	date
dropoff_datetime	date
passenger_count	integer
trip_time_in_secs	integer
trip_distance	integer
pickup_longitude	float
pickup_latitude	float
dropoff_longitude	float
dropoff_latitude	float

Table 2: Fares Table Attributes

Attribute Name	Type
medallion	string
hack_ license	string
vendor_id	integer
pickup_datetime	date
fare_amount	float
passenger_count	integer
surcharge	float
mta_tax	float
tip_amount	float
tolls_amount	float
total_amount	float

namely *medallion*, *hack_license*, *vendor_id* and *pickup_datetime*. This 4-tuple serve as the key in joining the two tables. There is one to one mapping between tables.

NNDC Climate data: The data used is made available by National Climate data center. It provide various version of climate dataset ranging from hourly to daily for different regions. For our project we have used data from January 1 2010 to December 31 2013 measured at New York John F Kennedy Airport stations of NCDC. It contain daily summarized climate information. The attributes of table are mentioned bellow. We have mentioned only the once that are important

Table 3: Weather Table Attributes

Attribute Name	Type
STN	integer
YEARMODA	date
TEMP	float
FRSHTT	binary

in our context. $FRSHTT$ is the most important feature which classifies weather into following categories or combinations of these using binary representation. For example 011000 represents rain and snow.

- **F**og
- **R**ain
- **S**now
- **H**ail
- **T**hunder

- **T**ornado

Census data: This dataset provides census tract shape files. We used these shape files to partition NYC into small neighborhoods. Neighborhoods information made is possible to group various pickup locations into one cluster and then perform analysis on it. This data is provided by New York City Department of City Planning.

2.2 Tools

Hadoop streaming: Most of the code written for completion of this project uses Hadoop Streaming[6]. It is an open source software framework for storing and processing big data in a distributed fashion on large clusters of commodity hardware. Essentially, it accomplishes two tasks: massive data storage and faster processing. We used hadoop for Map and Reduce tasks.

Hive: Apache Hive is a data warehouse infrastructure built on top of Hadoop for providing data summarization, query, and analysis[7]. We used Hive to clean and pre-process data.

Amazon AWS: Amazon AWS is a collection of remote computing services, also called web services, that make up a cloud computing platform offered by Amazon.com. It provide scalable and re-sizable clusters for computing. We mainly used Elastic Map Reduce (EMR) and Elastic Compute Cloud (EC2) for computations during this project. EMR runs both Hadoop and Hive.

Tableau Desktop: It is a data analysis and visualization tool. Most of the plots are made using Tableu software which is free for students.

CartoDB: It provides GIS, data visualization and web mapping tools. We found it better to use for plotting geo-spatial data.

3 Experiments and Techniques

In this section we will discuss the various experiments conducted on NYC taxi data. We will also discuss some of the techniques used to conduct and optimize these experiments.

3.1 Pre-processing and Cleaning

We first cleaned the data tables to remove inconsistent values and null values. Some of the duplicate records are also removed. This is a important to perform since that duplicate or abnormal value can create noise in the analysis model and might make the results noisy. To clean the dataset we initially used hive and then map-reduce to finalize the dataset.

3.2 Joining datasets

For our analysis we used two additional dataset apart from taxi data namely, weather and census tract data. As mentioned in previous section weather data contains a field $FRSHTT$ which classifies the day weather into various sub-classes and there combinations. We first reduced these combinations to crude classes like fog, snow, rain, sunny and thunder by making reasonable assumptions like "if it is rainy and thunder then we can assume it to be thunder". Once we have these classes we joined the trip and weather data using *pickup_date* and $YEARMODA$ as keys. Thus we added one more column to the trip data namely, *climate*.

Next, the census tract data is added to tables. Census tract data provided shape file for New York City, using which we can divide the city into small neighborhoods. We use the information in

these shape files to classify the *pickup* and $dropoff$ coordinates into neighborhoods. We adopted the Map reduce code by Visualization and Data Analysis lab at New York University to add this information to the trip tables. This was a very crucial step since most of our further analysis are based on this information added.

3.3 Pickup and Drop-off distributions

After joining the census tract information with trip table we have two additional columns giving information about neighborhoods of pickup and dropoff points. To find the distribution of pickup and dropoff neighborhoods we ran two map reduce jobs. The dynamics of the two jobs are given below.

Mapper: $Emit(pickup\_neighborhood, localcount)$

Reduce: Input to reduce will be $pickup\_neighborhood$ as key and a list of $localcount$. At reducer we add the elements of the list and output a single $(pickup\_neighborhood, aggregatecount)$ pair.

The same method is used to $drop\_off$ distribution. In this case, mapper emits $dropoff\_neighborhood$ instead of pickup. We perform local aggregations at each mapper to reduce the number of key-value pairs floating in the system thus reducing the volume of numbers transferred between mappers and reducers.

3.4 Preferred neighborhood of drivers

In order to find preferred neighborhood of drivers or groups of drivers, we created a list of frequency of neighborhood per driver. We further processed this list and found the top k most frequent neighborhoods for each driver. For our experiment we started with k as 10 but realized that it is too small since there are a lot of frequent neighborhoods which are common among drivers. With some more experimentation we changed k to 50. We also placed some thresholds on the minimum number of visits to be even considered for frequent neighborhood. Below are the mapper and reducers.

Mapper: $Emit(hack\_license, pickup\_neighborhood)$ and $Emit(hack\_license, dropoff\_neighborhood)$ for each trip.

Reducer: Input to reducer will a list of neighborhoods. At reducer we compute the frequency of these neighborhoods and threshold them. After thresholding the remaining neighborhoods are sorted and top k are selected. Reducer outputs a list of top k neighborhoods for each driver.

Once we get the list of preferred neighborhoods for each driver we then perform clustering on to find group of drivers with similar preferred neighborhoods. Below are the details of the clustering performed.

K-means: The KMeans algorithm clusters data by trying to separate samples in n groups of equal variance, minimizing a criterion known as the inertia or within-cluster sum-of-squares. This algorithm requires the number of clusters to be specified. It scales well to large number of samples and has been used across a large range of application areas in many different fields. The k-means algorithm divides a set of N samples X into K disjoint clusters C, each described by the mean μ_j of the samples in the cluster. The means are commonly called the cluster "centroids". The K-means algorithm aims to choose centroids that minimize the inertia, or within-cluster sum of

squared criterion:

$$\sum_{i=0}^{n} min(||x_j - \mu_i||)^2 \tag{1}$$

We choose number of clusters to be 5000 based on experimentation. We used scikit-learn [8] package to perform clustering in parallel using multicore optimized code. Algorithm took nearly 90 minutes to train and cluster the drivers.

3.5 Drivers taking longer routes

In this part we will discuss the next very interesting problem about finding drivers who take longer routes Aka "Evil drivers". We divided the problem into two passes; Pass 1: Finding the correct distance ("What is right ?") Pass 2: Find drivers who are taking longer routes ("Who is wrong"). We will now discuss both the passes.

3.5.1 Pass1: Finding the correct distance

In this phase we will find what is right or what is the expected distance between two the points; pickup and dropoff. We explore two approaches for doing this. First, use APIs to get the distance matrix and know the correct distance. This approach required us to make API calls to services like openstreetmap or googlemaps etc with query parameter as the pickup and dropoff coordinates (not neighborhoods); in return these services will return the expected driving distance between these points. However there are limitations to this approach like the maximum number of API calls within a given time-frame, maximum number of overall API calls and most importantly the speed at which we will get back results. All these constraints made this approach very unfavorable and not feasible with such huge data (500 million trips). The second approach is more interesting an does not require any external information.

In this approach we statistically compute what is right. We used map-reduce framework to compute the average trip distance between two neighborhoods. Below are the mapper and reducer for this task.

Mapper: $Emit(pickup_neighbourhood + dropoff_neighbourhood, trip\_distance)$

Reducer: At reducer we receive a $pickup_neighbourhood + dropoff_neighbourhood$ as key and a list of $trip\_distance$. We compute the average trip distance and emit $pickup_neighbourhood + dropoff_neighbourhood$ and $average\_distance$ as output.

We know have the average distance between neighborhoods and hence we know what is right (this is assuming that the proportion of "good" drivers is much more than "evil" driver. Its a nice world !)

3.5.2 Pass2: Detect drivers with average longer routes

In this pass we will find who is wrong or who are the drivers taking longer routes very frequently. We will compute this using another map-reduce task where we will compute the difference between the driver's trip distance and actual trip distance (computed in previous pass). The mapper will produce the difference and value 1 of trips while the reducer will calculate the "evil" score for each driver. Below are the mapper and reducer for this task.

Mapper: Calculate $trip\_distance - lookup\_distance[pickup + dropoff]$. $Emit(hack\_license, difference, 1)$

Reducer: Input to the reducer will be $hack\_license$ as key and list of $difference$ and 1s. Reducer will add the difference and compute the "Evil" score using numtrips (addition of all 1s) as normalization. This normalization will prevent drivers from incorrectly getting high "Evil" score. We do this since it could we a case that there was a road block or the passenger wanted to driver to take a longer route. Since we have a lot of data all these outliers are expected to be averaged out.

3.6 Correlation with weather

As mentioned in previous section we have weather data for each day classified into fog, rain, snow, thunder and sunny. To find correlation between weather and taxi trips we will use this information. We will see what how the number of trips for a day vary with these weather conditions. We used mapreduce to compute the average number of trips for each weather condition. Below are the mapper and reducer.

Mapper: $Emit(weather\_condition, num\_trip, num\_days)$.

Reducer: Reducer receives weather condition as key and list of number of trips and unique days. At reducer we aggregate the number of trips and divide it with the total number of days. Reducer emits $weather\_condition$, $average\_numtrips$ pair. We perform the same experiment using tips and total amount also. Only change will be in mapper where it will now $Emit(weather\_condition, tip, num\_days)$ or $Emit(weather\_condition, total, num\_days)$, rest all will be same.

This will give us insight into how the weather condition effects the taxis.

3.7 Popular Destinations and temporal changes

Next we will explore what are the popular destinations and how popularity changes with time. To calculate it we will use the previous results about dropff distribution. Using the information from the dropoff distribution we selected a few popular (high frequency) destinations and calculated how there distribution changes with time. Below are the map reduce functions.

Mapper: $Emit(date, num_trip)$ for a particular destination i.e where the destination neighborhood is found from the dropoff distribution.

Reducer: Using the list of num_trips aggregate the total number of trips for this destination per day. Reducer emits $(date, total\_trips)$ pair.

Next we plot the $total\_numtrips$ for each date and see how the patterns change.

3.8 Working duration of drivers

Typically a shift is of 9.5 hours however there are few drivers who work overtime and a few who work part time. In this part we will find these outliers. Using map reduce we will find the average working hours for each day.

Pass 1: Compute in car time

This is done by adding the pickup time with $trip\_time$ for each driver. Here the **mapper** emits the $(driver, time)$ pair for each driver.

At **reducer** each time is aggregated and emit total in car time for each driver. This is considering only the trip time.

Pass 2: Compute count of different hours

Here we use map reduce to find unique in car hours for each driver. This is the total duration of work by each driver.

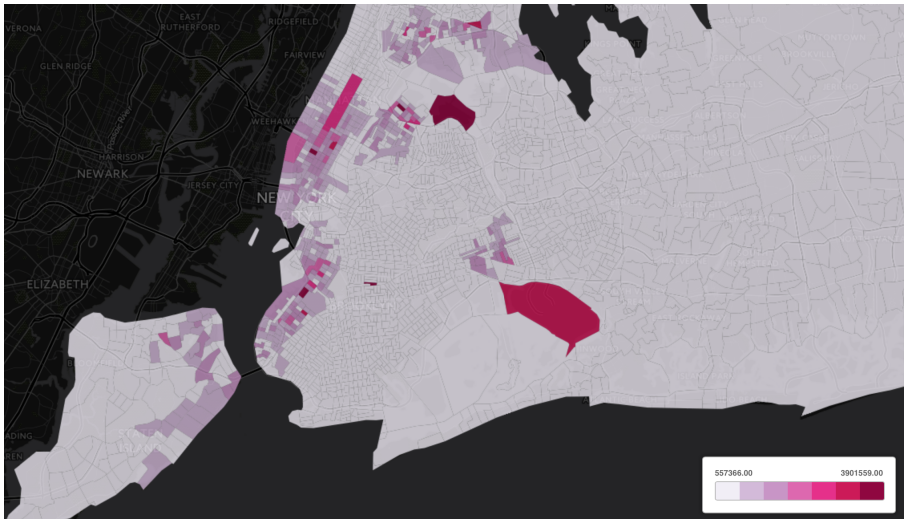

Figure 1: Pickup distribution

3.9 Tips per driver

In this part we will calculate the average tip a driver receives per trip and try to compute find out some outliers.

Each **mapper** emits the tip amount and 1 with *hack* as the key.

At **reducer** it will receive lists of tips and 1s per *hack*. Reducer will first aggregate these number to compute total-tip and total number of trips per driver. Next reducer will average the tips using the total number of tips.

4 Results and Discussions

In this section we will discuss the findings from the experiments conducted in the previous section.

4.1 Pickup and Drop-off distributions

Based on the whole years trip data of 2013, the top 10 pick up neighborhoods are found. The top 1 pick up neighborhood is at **midtown Manhattan Madison Square Garden**, where there were 3901559 pick ups happened during the year 2013. It is no surprise to find this since several popular vacation spots are around this neighborhood, such as the Empire State Building, Macy's and Korean Town. Another two neighborhood also have 3901559 pick ups happened in 2013, which are a neighborhood near **Astoria Park in Queens** and a block near **Green-Wood Cemetery in Brooklyn**. The No.4 top pick up neighborhood is **Laguardia airport in Queens**, at which 3616210 pick ups happened in 2013. Another neighborhood that also has 3616210 pick ups happened is a block near **Grown Heights in Brooklyn**. The No.6 top pick ups neighborhood is the **Grand Central in Midtown Manhattan**, where there were 3103567 pick ups happened. Another neighborhood which also have 3103567 pick ups is a block near **Sunset Park near Brooklyn**. The famous **John**

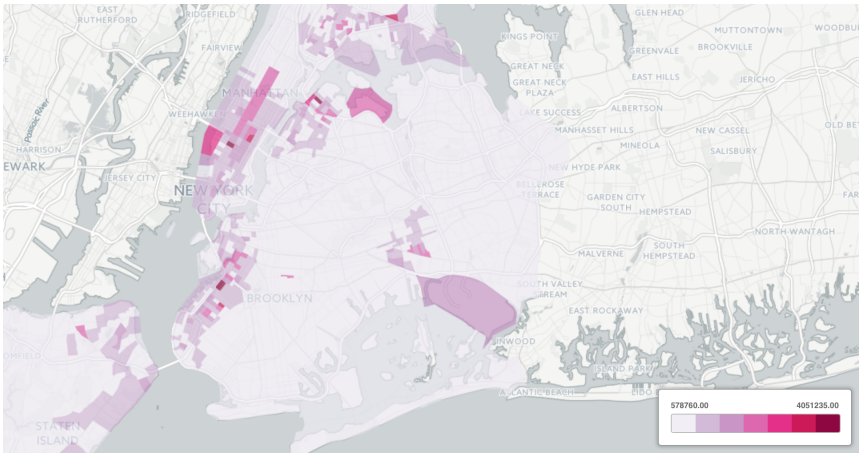

Figure 2: Pickup distribution

F. Kennedy International Airport, where there are 2811411 pick ups happened during 2013, is the No. 8 top pick up neighborhood. A block near **Columbus Circle in Midtown Manhattan** is the No. 9 top pick up neighborhood where there are 2415213 pick ups happened. The No.10 top pick up place is the **Central Park in Manhattan**, at which 2245274 pick up happened. There are many more popular pick up neighborhood in New York City. In the map, the darker the neighborhood is colored, the more pick up happened there. Full version of map available at `https://xuanjm.cartodb.com/viz/419b2ae0-f677-11e4-8503-0e853d047bba/embed_map` Based on the 2013 trip data, a drop off neighborhood map is created. The No.1 hot drop off neighborhood is **Madison Square Garden in Manhattan**, where are 3621804 drop off happened in the year of 2013. The **block near Astoria Park in Queens** and the block near **Green-Wood Cemetery** are also hottest drop off neighborhoods. **Grand Central** is the No.4 hot drop off neighborhood, where there are 2901474 drop off happened. The neighborhood near **Hudson River Chelsea Piers and Chelsea Waterside Park** at Lower west Manhattan is the No.5 hot drop off place. **Astoria Park in Queens** in the No.6 hot drop off place. Similar with the hot pick up neighborhood, the two airports in New York City, which are the **Laguardia Airport** and **JFK Airport** are also in the top 10 drop off neighborhoods. Please check the map for more information, the darker the neighborhoods are colored, the more drop off happened in the neighborhood in 2013. Full version of map available at `https://xuanjm.cartodb.com/viz/ea8ce0d0-f677-11e4-8503-0e853d047bba/public_map` The hot pick up and drop off neighborhood study could give drivers and vendors clue how to deal more trips to increase the revenue.

4.2 Preferred neighborhood of drivers

As discussed in previous section we computed clusters of drivers with common neighborhoods. Figure below shows the cluster These are using 5000 clusters.

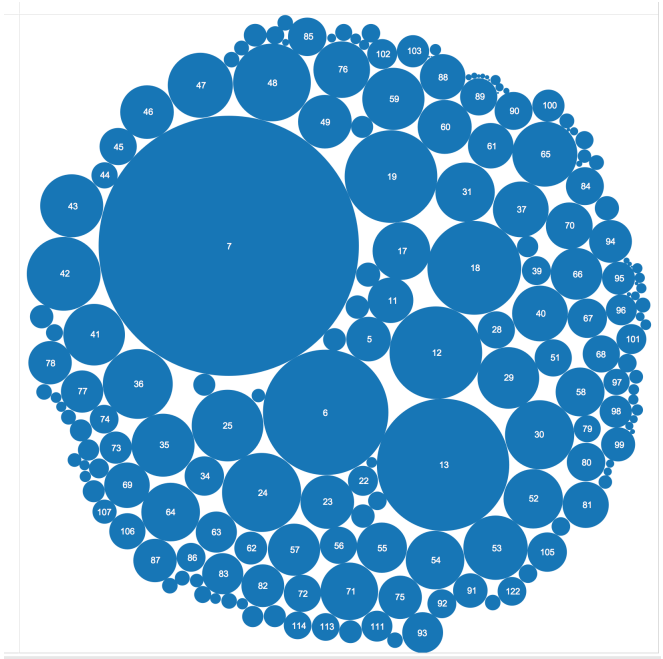

Figure 3: Cluster of different drivers

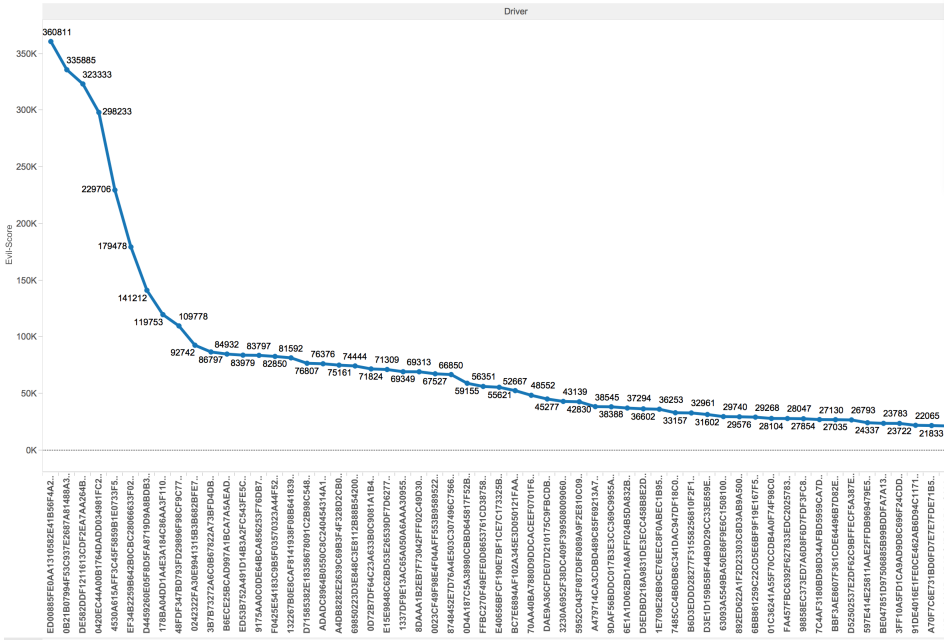

Figure 4: Sorted Evil scores for drivers

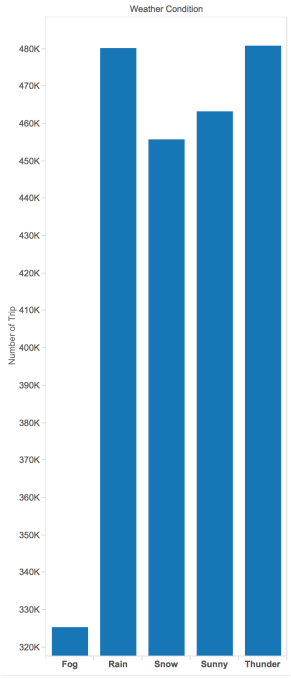

Figure 5: Number of trips vs weather conditions

4.3 Drivers taking longer routes

There are 41457 number of drivers in 2013, and we calculate the evil score by adding all the percentage above average trip distance. As we can see in the plot (only top 50 drivers) it is a long tail model and there is a sudden drop around top 10. Thus we can deeply look into these drivers and investigate their behavior. Further study should be considered to make sure those high score is not caused by data/system problem and set up certain threshold for automatic alert.

4.4 Correlation with weather

We plot the number of trips against the weather conditions. From the plot it is visible that adverse weather effects the number of trips. However it will depend on the adversity. It is clear from the plot that less trips are there on foggy days but on a rainy day the trips increase which might be due to the fact that people are forced to take cab instead of walking. That is evident from the number of trips going down for sunny day.

4.5 Popular Destinations and temporal changes

From the destination distribution we choose three destinations

- Central Park
- JFK

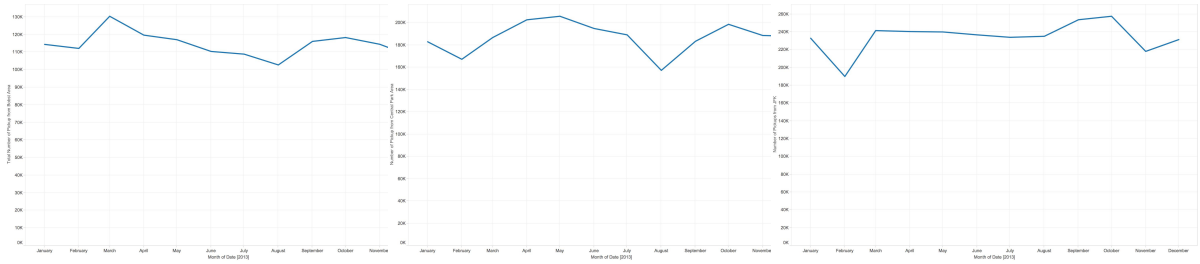

Figure 6: Number of trips vs Place

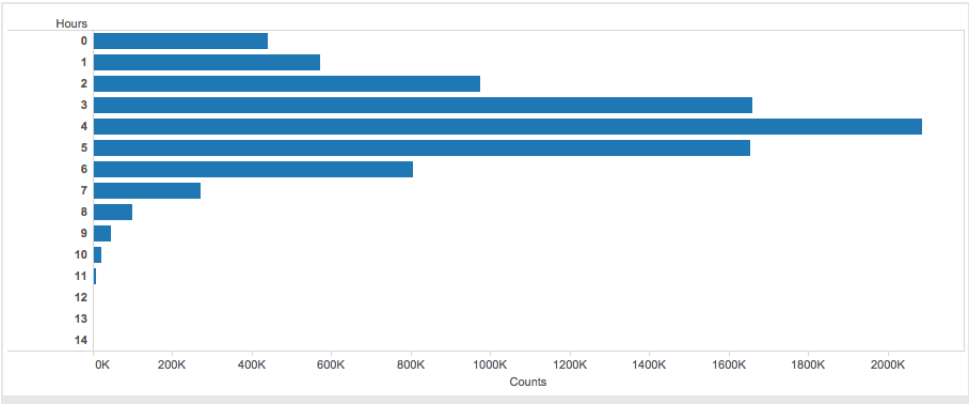

Figure 7: Total trip duration per day

- NYU Bobst area

Following are the plots for number of trips changes for these places.

Observations:

- **Bobst**: It is clear from the graph that taxi activity is grater in month of semester. It goes down in summer and start to come back up in august end, which is the start of semester.
- **Central Park**: This is nearly opposite to Bobst, this goes up in summer months since a lot of people might visit park and again goes down as it start to become chilly again.
- **JFK**: It is mostly constant throughout the year since the traveling is mostly constant except for October-November when its holiday season approaching. Two dips in December and February are indications of really bad weather due to which a lot of flights got canceled.

4.6 Working duration of drivers

4.6.1 Total working hours per day

Plot shows the total trip duration per day. It is visible that most the drivers work for 4-5 hours while there are few who work nearly 11 hours a day also.

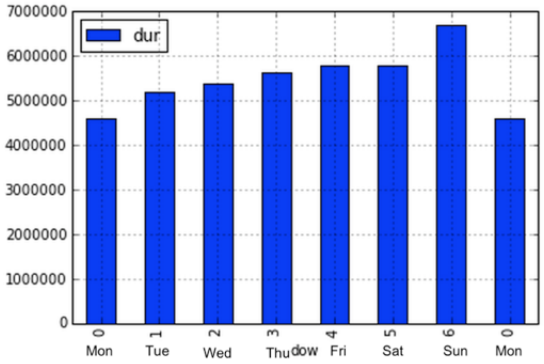

Figure 8: Total trip duration daily

4.6.2 Daily trip duration

This graph shows the accumulation of the trip duration happened in each Sunday, Monday, Tuesday, Wednesday, Thursday, Friday and Saturday in the year of 2013 in New York City. For example, the sum of trip duration of each Monday is around 4600000 hours. According to the graph, Sunday has the highest sum of trip duration and Monday has the lowest sum of trip duration in 2013. This study shows either more trips happened on Sunday, or trip time is longer on Sunday, or both.

4.6.3 Trip duration yearly

This graph shows the comparisons among the trip duration in the year of 2011, 2012 and 2013. Based on the graph, the trends of trip duration change in a similar way. March, May and October have higher trip duration and January and February have lower trip duration in all of the three years.

4.7 Tips per driver

Figure shows the average tip for each driver. Most of the drivers receive near to average tip but there are some outliers get nearly $ 30 tips o average. We explore these outliers further and found some of them are faulty data points where tips are wrongly written. Using this information we calculated the higher income group drivers using a threshold of one standard deviation above the average. We found that most of the high income group drivers work full time and usually pickup passengers from the popular pickup places.

4.8 Tip distribution

The following plots show tip distribution both yearly and quarterly. It is evident that the number of tips are increasing each year.

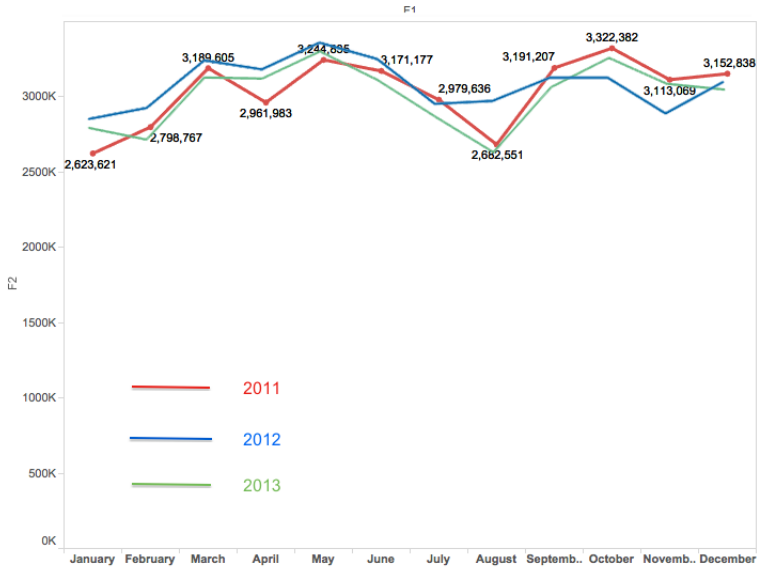

Figure 9: Total trip duration yearly

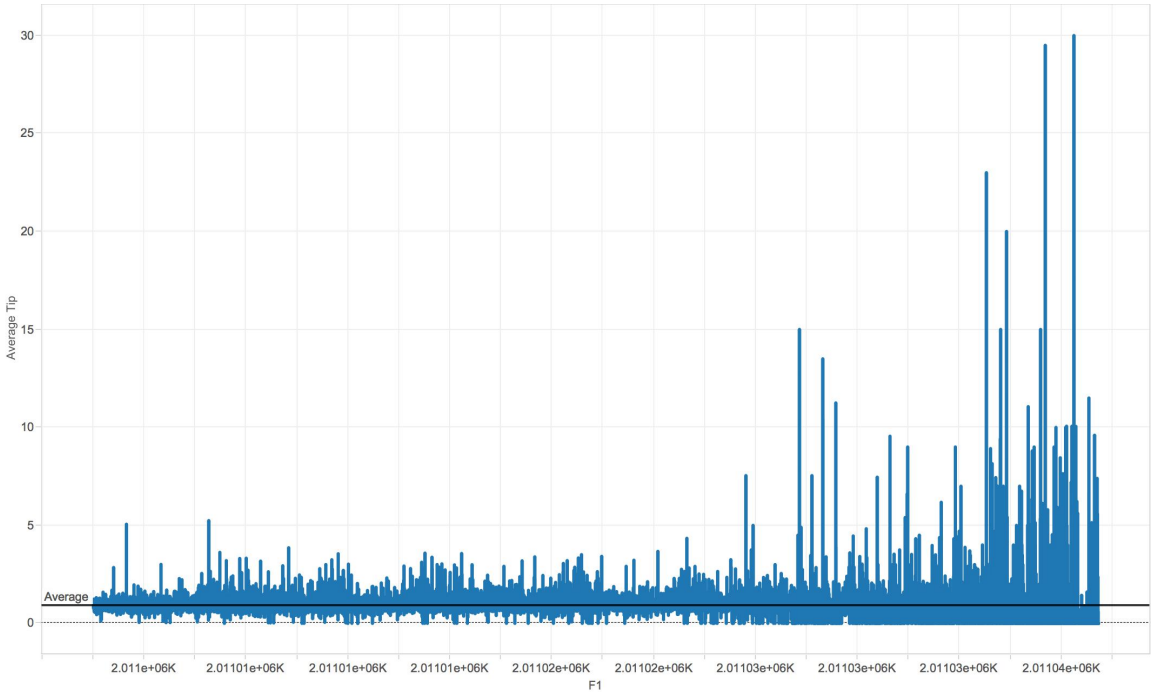

Figure 10: Average Tip per trip per driver

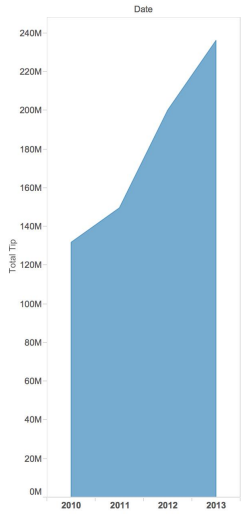

Figure 11: Total Tips per year

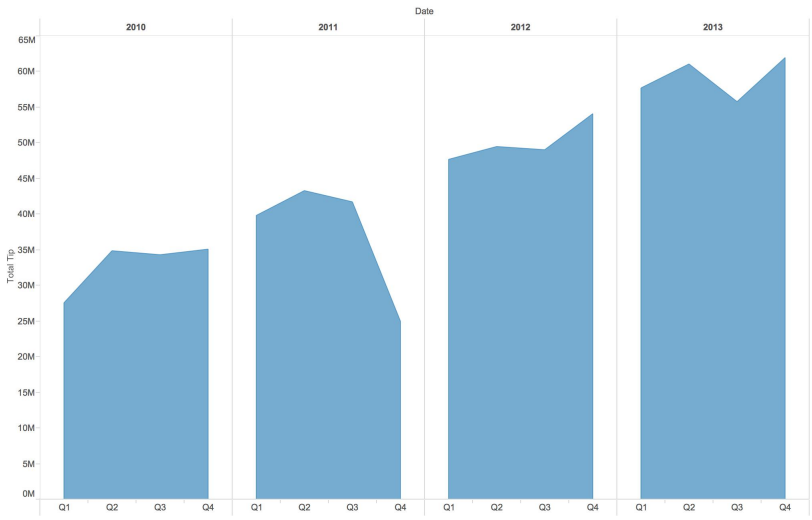

Figure 12: Total Tips per Quarter

4.9 Trip distribution

This graph shows the accumulation of the number of trips happened in each Sunday, Monday, Tuesday, Wednesday, Thursday, Friday and Saturday in the year of 2013 in New York City. We can conclude that less people use taxis on Sundays and there are fewer drivers working on Sunday.

5 Conclusion

In concussion we would regard this project as a very good learning experience where we dealt with not only finding solutions and patterns from data but also dealing with huge amount of data. We found a lot of insightful information about the behavior of taxi drivers and passengers of New York City. The most interesting one being the "Evil" score finding. Hack license "ED00885FE0AA1310582E41B56F4A25E3" and "0B21B07994F53C937E2687A81488A3C5" should be investigated as they consistently take longer routes. For future work we would like to add more external datasets and infer more useful facts about NYC taxis.

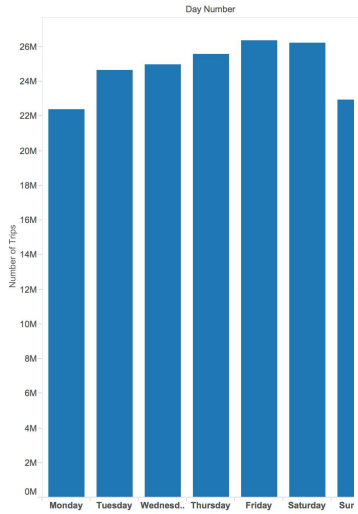

Figure 13: Total Trips per day

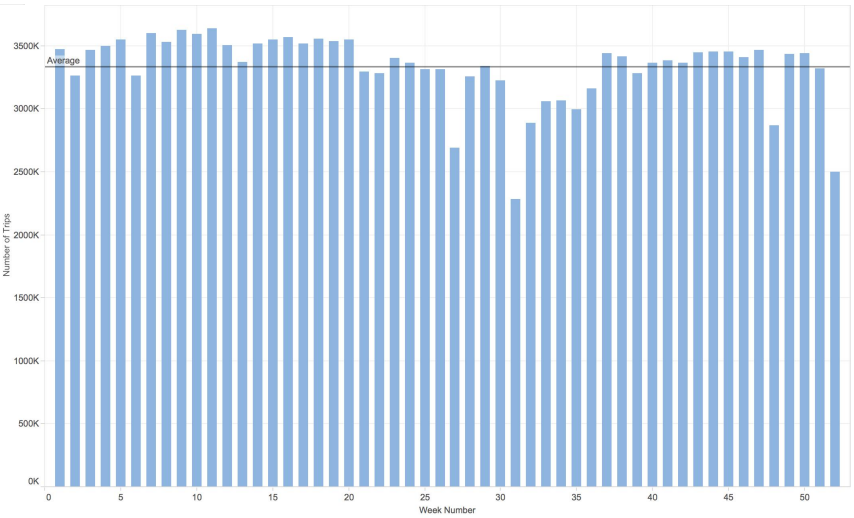

Figure 14: Total Trips per week

Acknowledgements

We would like to thank Professor Juliana Freire for providing us with Amazon AWS credits and HPC access. We would also like to appreciate NYU VIDA for their code. Lastly we would like to acknowledge Tableau Software for providing free student version of their software.

6 References

[1] N. T. L. Commission, "2014 taxicab factbook," 2014. [Online]. Available: http://www.nyc.gov/html/tlc/downloads/pdf/2014_taxicab_fact_book.pdf

[2] ——, "Your guide to boro taxis," 2003–. [Online]. Available: http://www.nyc.gov/html/tlc/html/passenger/shl_passenger.shtml

[3] ——, "Your guide to boro taxis," 2003–. [Online]. Available: https://nycopendata.socrata.com

[4] N. Y. C. D. of City Planning, "Political and administrative districts data." [Online]. Available: http://www.nyc.gov/html/dcp/html/bytes/districts_download_metadata.shtml

[5] N. C. D. Center, "Nndc climate data online." [Online]. Available: http://www7.ncdc.noaa.gov/CDO/dataproduct

[6] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The hadoop distributed file system," in *Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on*, May 2010, pp. 1–10.

[7] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, P. Wyckoff, and R. Murthy, "Hive: A warehousing solution over a map-reduce framework," *Proc. VLDB Endow.*, vol. 2, no. 2, pp. 1626–1629, Aug. 2009. [Online]. Available: http://dx.doi.org/10.14778/1687553.1687609

[8] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

A Data Experience

1. Clean data :

   Some data include tab and newline character, especially after dealing with the data with AWS. Then need to use strip() function and csv package to deal with the data.

2. Data type Inconsistent :

   For some data attribute, the data type in different files is different. For instance, some hack_lisence's data type is integer, the rest hack lisence's data type is hashed data. Some trip time is in seconds and the rest are in minutes. The data type inconsistent leads us into trouble. Such as, when creating a hive table, the data type may works for some data files, but may cause problems with other data files when uploading data into hive tables.

3. Outlier data :

   Additionally, as described in the finding part, we find dirty data in the trip data files. In the data file of 2013 August 25th, there are 950 extremely large data about trip duration. Those dirty data create weird output (shown in the first graph below). In order to make the output reasonable, those dirty data were deleted manually. The first graph below was created using data set, which includes the dirty data. The second graph below was created after the dirty data were deleted.

B Software Experience

1. Hive:

   We planed to use Hive to create a relational database and store all the data from 2010 to 2013 in the hive tables. But the hive tables always have multiple null columns and could not find a method to solve the problem. Then we plan to use Oracle SQL developer to create a relational database to store all the trip data and trip fare data. But it takes too long time to upload data to tables in Oracle SQL Developer. Additionally, the Oracle SQL developer is provided by another course in NYU. Professor gives every student an account with limited space to store data, thus there may not be enough space to store all of the data. Thus we focus on Hadoop Streaming to do the project

2. Hadoop Streaming

   Sometimes we met problems when debugging. Even thought using "cat" to test the python code, which shows the python code doesn't have any problems, those code may not work in hadoop. Then need to check the log in very computer node and revise the python code. Logs are very cryptic and sometimes took a lot of time to debug simple errors.

C Individual Contributions

- Arpit Jain:
  - Finding similar drivers and preferred neighborhoods
  - Temporal and weather analysis of frequent neighborhoods

  - Tableau to plot graph and using LaTeXto create report

- Jiamin Xuan:
  - Find the trip duration and evil drivers
  - Use geopandas spatial join data to find driver behaviors
  - Use cartodb to create graphs

- Xiaohui Guo:
  - Tried to work on hive and oracle SQL developer
  - Use Tableau to create graphs
  - Write Report and create Power Point