TAMIDS

DATA SCIENCE COMPETITION 2018

DATAGEEKS

TAMIDS

Participants: Mustafa Panbiharwala

Contents

Introduction

Importance of the problem

In this project, I have conducted analysis and prediction on the publicly available Chicago Taxi Data from the year 2013-2017(July). This project addresses the challenges faced by the transportation industry on how to maximize their revenue i.e. which community area has the greatest pick up density, what are some of the important parameters that determine the total trip rate for a particular ride.

Taxi business in Chicago city has always been dynamic and with the advent of tech savvy competitors like Uber and Lyft, it has become highly important for Chicago Transit Authority to strategically allocate their resources. Number of public taxis in the city has been on the decline and total revenue shows the same trend.

One of the major difference between Uber and a typical Chicago taxi is that latter does not provide predicted total amount for a particular trip. This is a major problem for both drivers and customers, customers cannot handle uncertainty attached with the ride. This analysis aims to make transportation more efficient for the Chicago Transit Authority.

Data

Data for Chicago taxis from the year 2013-2016 contains over 105 million rows, this makes it difficult and computationally intensive for us to conduct any type of analysis on such a huge dataset. Most of the attributes pertinent to the ride are present. Each trip's record fields: pick-up and drop-off dates/times, pick-up and drop-off coordinates, trip distances, itemized fares (fare, toll, tax and tip amounts), payment types, and trip duration. I have also incorporated the effect of weather parameters like temperature, average wind speed, precipitation and snow on taxi's revenue. Data from the year 2017 was used for testing purposes.

To analyze the effect of special occasions on Taxi business, some predictors were also added into the dataset which weren't present, for instance I wanted to check how much revenue increases on St. Patrick's Day, and decreases on Christmas Day. Hence some of the important holidays were also taken into consideration while performing our analysis. Drawing the analogy from fanaticism that residents of college station have for Aggie football, I have also focused on how Chicago Cub's home game impact the taxi business.

Exploratory Data Analysis and Prediction

In our analysis, I have examined how total revenue has changed over the past four years. How fares and tips change on an hourly basis and how week days affect the number of rides in a particular community. After our exploratory data analysis, it was clear that Chicago's "Loop", which is the city's official downtown area, contributes a major proportion to trip revenue as it has the highest number of pick-ups.

In this project, I have developed a predictive model where trip total is the response variable. I studied the impact of various features and also attempted to find the best possible way to leverage them. I reviewed the performance of various models (Linear Regression, Random Forest and Light GBM) and discussed the effectiveness and shortcomings of these models.

Executive Summary

Project Objective

The objective of this project is to build visualization models which help in the interpretation of how Chicagoan uses their public taxi on an hourly, daily and weekly basis. Exploration and visualization aids in feature extraction for developing predictive models using different statistical learning algorithm. Our Predictive models aim to forecast the trip total amount and number of rides from a particular location on a given hour of the day.

Target Audience

Predictive and visualization models like these are interesting for many stakeholders: -

1. **Taxi firms**: Companies can allocate their resources effectively by diverting the cabs to a specific location during specific times.
2. **Traffic planning**: I all have experienced a traffic delay on any special occasion, be it the end of a movie or on a Sunday when you are at the church, this project will help in traffic management.
3. **Data scientists**: As budding data scientists I always try to learn from other models and think about ways on how to improve the existing model, this helps in eliminating the need of reinventing the wheel.

Challenges & Counter Measures

This project was challenging in terms of the volume of data it contains, hence this created a need to conduct the predictive analysis on a subset of the data. In the competition there are number of ways one could create a subset, I opted for two ways for aggregating 105 million rows: -

1) Calculated the weekly total taxi fares for each Taxi ID for each week in the data and took the median of total taxi fares for each Taxi ID.
2) Calculated the daily total taxi fares for each Taxi ID for each week in the data and took the median of total taxi fares for each Taxi ID.
3) Calculated the hourly total taxi fares for each Taxi ID for each week in the data and took the median of total taxi fares for each Taxi ID.

Extraction of the desired data from a 40 gigabyte CSV was a still a difficult task. To query the data, I used Google Cloud Platform and Google Bigquery. (Code-2 Subset selection on Google Bigquery)
Data Visualization models were built by using the entire data on Tableau, Tableau enabled us to extract the data using Google Bigquery and hence all the plots were developed easily.

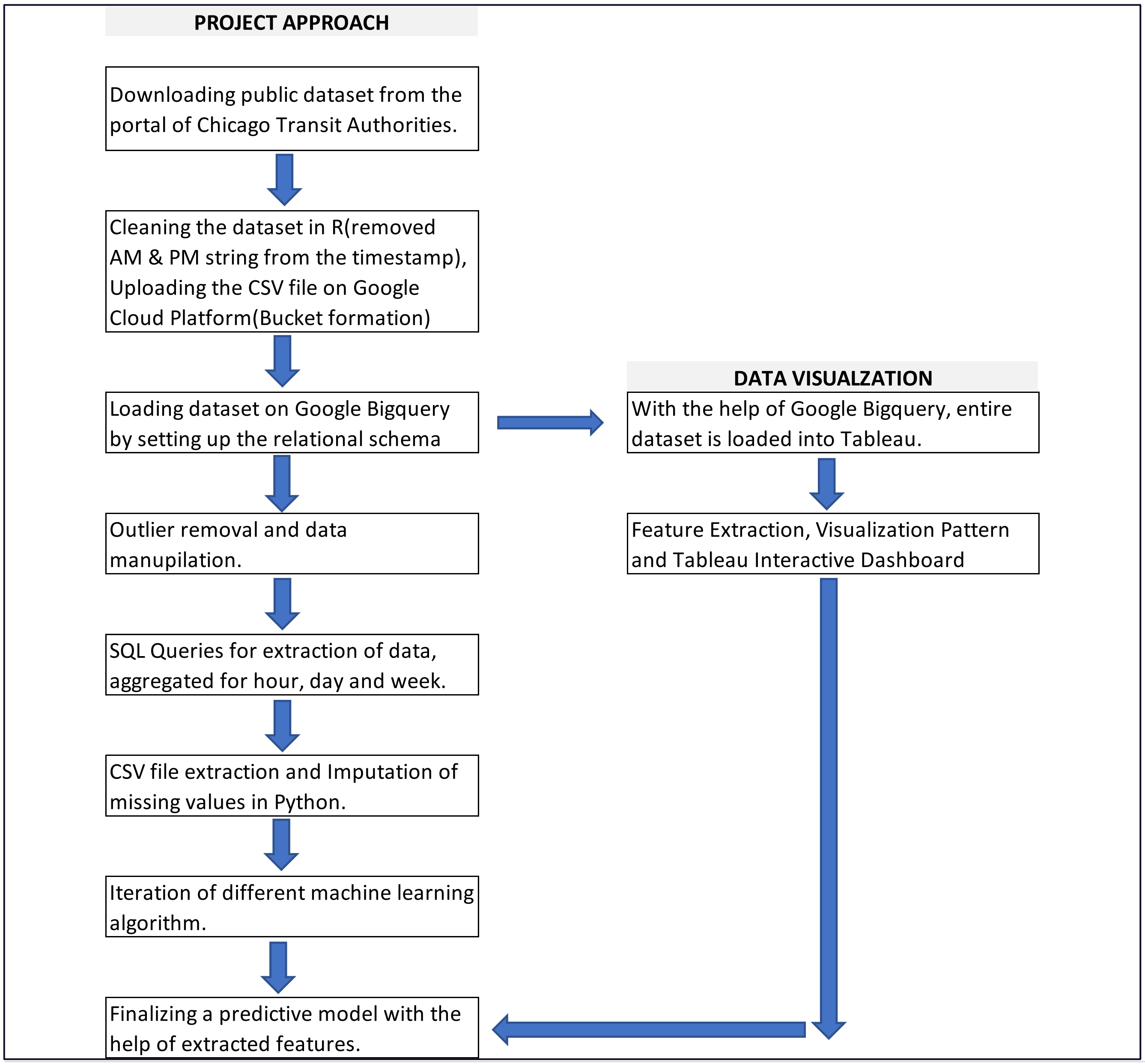
Conclusion

This project entails how weather, holidays, mode of payment and time of the day plays an important role on tips and taxi fares. Trip distance remained the most influential parameter which helped in determining the trip total cost for each trip. Different models were tried including Linear Regression, Random Forest and Gradient Boosting Method. Accuracy of 74% was obtained for the Gradient Boosting for hourly and daily fares, 68.8% for Random Forest while predicting hourly fares, 70% for Random Forest for predicting daily fares and 81% for Random Forest for predicting weekly fares. An accuracy of 52.4 % for predicting hourly fares and 54.9% for predicting daily fares and 81.8% for predicting weekly fares.

Methodology Used: -

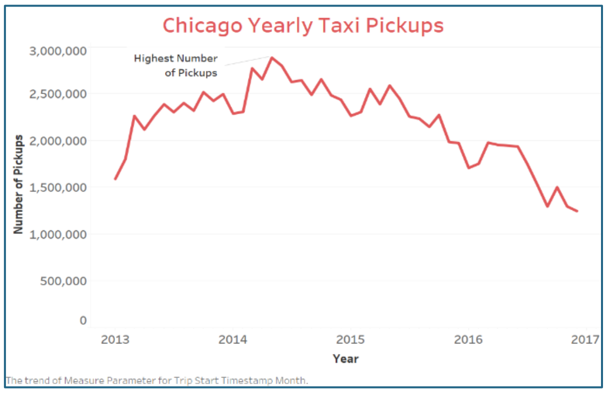
The following procedure was used to build predictive and data visualization models. The flow chart elaborates the entire process. Google Bigquery was extensively used to decrease the amount of time that was required to conduct exploratory data analysis on Tableau. It also helped in the extraction of the required datasets, as simple SQL commands allowed for the creation of subsets of data. The queried data was small in size and hence made it easier to carry out analysis in Python (Code-3 Predictive Modeling on R and Python). One of the main reason for using Google Cloud Platform and Google Bigquery was that it was free, hence no cost was incurred during our project.

Timestamp column contained AM/PM symbols and because of these symbols uploading a CSV file on Google Bigquery was difficult. While defining the relational schema, Bigquery needs either timestamp which is in the format '' but the presence of AM/PM changed the datatype to strings. Without the timestamp value, aggregation on Bigquery was not possible and for this reason data was preprocessed in R. Instead of processing 4 single files, one big CSV file (44gigabytes) was used. (Code-1 Loading dataset into R for cleaning)


PROJECT APPROACH
Downloading public dataset from the portal of Chicago Transit Authorities.
Cleaning the dataset in R(removed AM & PM string from the timestamp), Uploading the CSV file on Google Cloud Platform(Bucket formation)
Loading dataset on Google Bigquery by setting up the relational schema
Outlier removal and data manupilation.
SQL Queries for extraction of data, aggregated for hour, day and week.
CSV file extraction and Imputation of missing values in Python.
Iteration of different machine learning algorithm.
Finalizing a predictive model with the help of extracted features.
DATA VISUALZATION
With the help of Google Bigquery, entire dataset is loaded into Tableau.
Feature Extraction, Visualization Pattern and Tableau Interactive Dashboard

Exploratory Data Analysis

To better understand this huge dataset and the features it contains, I first performed EDA. The main purpose of this is to see how the prediction variable behaves with various features and how I can clean and leverage these features in our models to achieve best results.



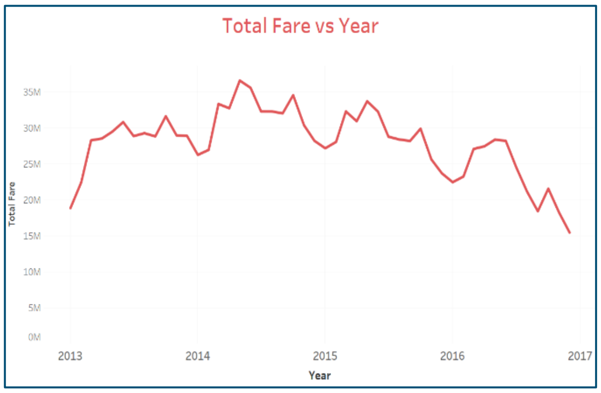


Figure 1 **Figure 2**

First, I wanted to visualize how much business has the Chicago cabs lost due to Uber and Lyft. It can be clearly seen from Figure 1 that there has been a dramatic decrease in the number of pickups, with an annual rate of 35%, making the Chicago cabs to lose around 45% of their business by the May 2015.

With number of rides decreasing one can easily deduce that the total revenue followed the same trend, total revenue for the Chicago taxi was reduced from $35 million to $18 million. This has caused huge economic burden on the cabbies as they aren't generating enough fares to keep up with their loan payments and meet their expenses.

More than 350 foreclosure notices or foreclosure lawsuits have been initiated against medallion owners in the year 2017, compared to 266 in 2016 and 59 in 2015. Since October, lenders have filed lawsuits against at least 107 medallion owners who have fallen behind on loan payments, according to the union's count. The major reason behind this financial distress is that since the emergence of Uber, Cabbies face an uneven playing field with the ride-share companies, who typically don't face the same permitting and fee rules. **[1]**

After digging a little deeper, I realized that the city of Chicago has enforced set of rules for taxi and ride sharing industries and made it impossible for cabbies to compete with Uber and Lyft. Transportation companies compete for customers, and ultimately it is the consumer who makes the choice. Consumers always look for cheaper options and sadly this has been the reason for the decline in number of rides for the Chicago's Taxis.

Figure 1 shows that there is correlation between number of rides and year. Hence every component of time be it year, week, day or hour should have a role to play in estimating number of rides on any specific day. I wanted to understand the pattern exhibited by pick up density over the course of an entire year. Hence, I plotted a graph (Figure 3) between number of rides vs number of week. This graph led us to one insightful conclusion. Almost every week the number of rides remained constant, except for week 10, 48 and 52.

Week 10 contains the most important holiday celebrated widely in Chicago City i.e. St. Patrick's Day. Week 48 and week 52 contains Thanksgiving and Christmas holiday. Impact of holidays are analyzed later in the report.

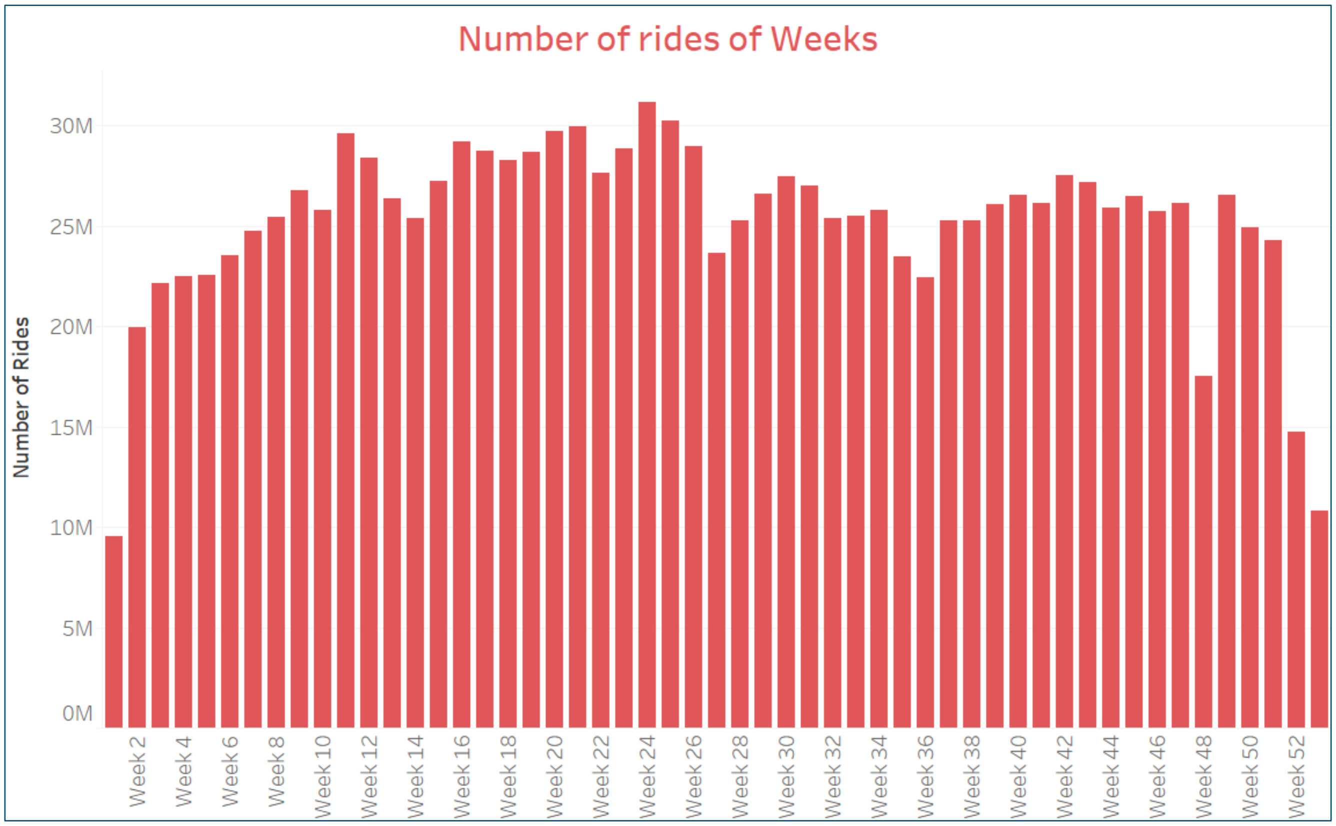


Figure 3

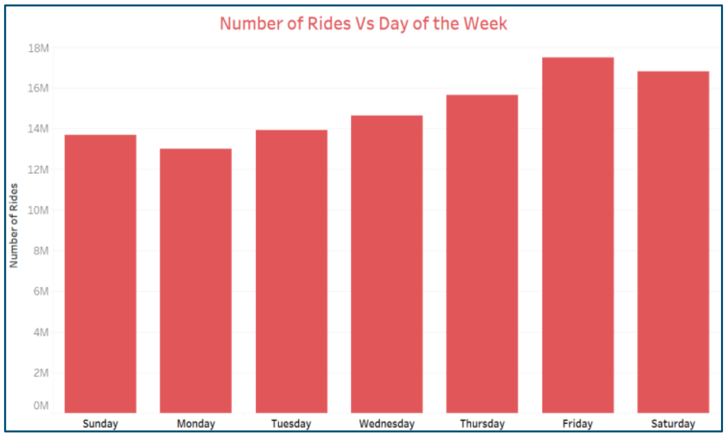


Figure 4

After analyzing the weekly trend, we wanted to see how number of rides varied with day of the week. The trend was pretty obvious, while most of the days number of rides were constant except on weekend i.e. on Friday, which had maximum number of rides. Total trip amount followed the same trend as well.

For more granularity I analyzed the number of pickups occurring by hour over the course of the day.

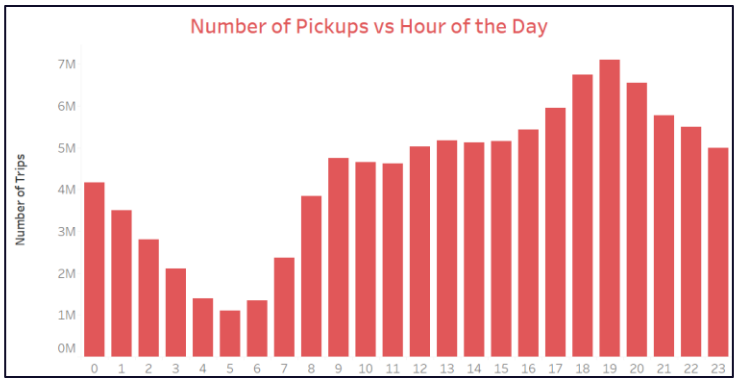


Figure 5

The morning and evening rush hours are clearly visible. Chicago is an all-day and all-night city as evidenced by the number of pickups throughout the hours of the day and night. It can be seen that more taxi trips begin an hour before the midnight than there are in the morning.

Notice the dip in the number of trips at the 5th-6th hour of the day. The number gets reduced from 2000k to 1400k, I analyzed how fare changed in this interval.

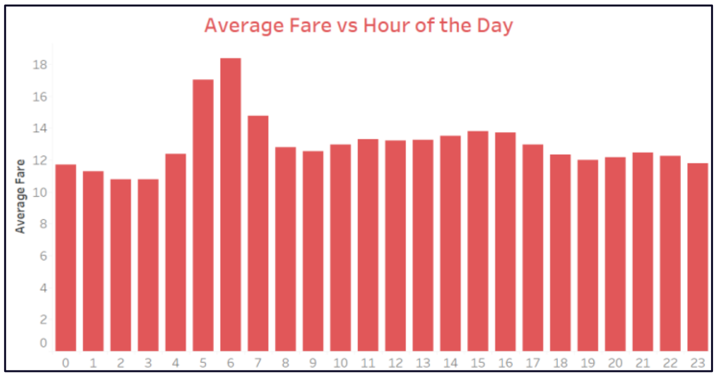


Figure 6

Results were quite interesting as contrary to our hypothesis; average fare i.e. fare/ride has peaked during 5th-6th hour of the day. Although graph for total fare vs hour followed the same trend as number of pickups.

After some iterations I figured out that trip distance has been influencing the average fare, it has the same bar chart as well.

I plotted the distribution of trip miles (figure 7) and deduced why trip miles were skyrocketing during that interval. It turns out that number of drop offs at O'Hare International Airport (Second busiest airport worldwide) were maximum during the 6th hour of the day i.e. many Chicagoans uses taxi to board their early morning flights. Since O'Hare is located on the far Northwest Side of Chicago, Illinois, 14 miles northwest of Chicago's Loop business district (community responsible for highest number of pick-ups), trip miles and total fare are usually higher **[2]**. Although the number of drops is greater in the evening as well, but the average remains low due to large number of rides.

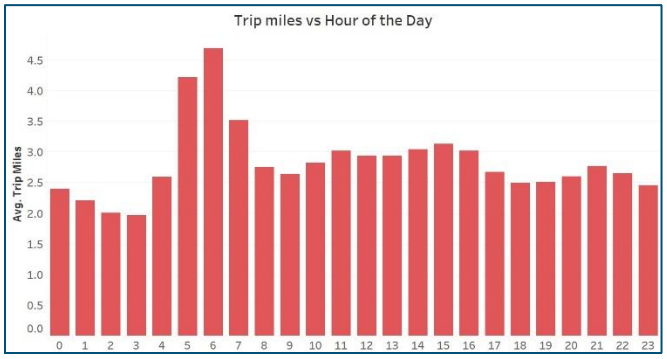


Figure 7

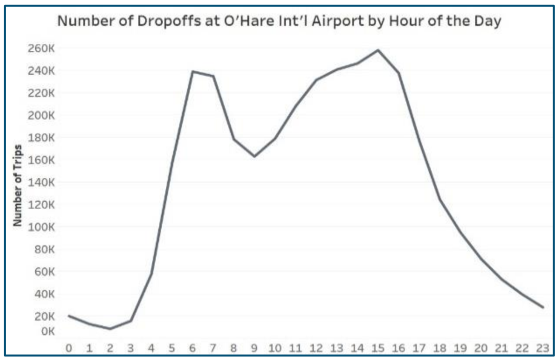


Figure 8

Weekly and hourly analysis made us curious and I wanted to see the combine effect of day of the week and hour of the day on number of rides. It would provide the exact the day and hour at which Chicago Transport Authority (CTA) should have maximum number of active taxis.

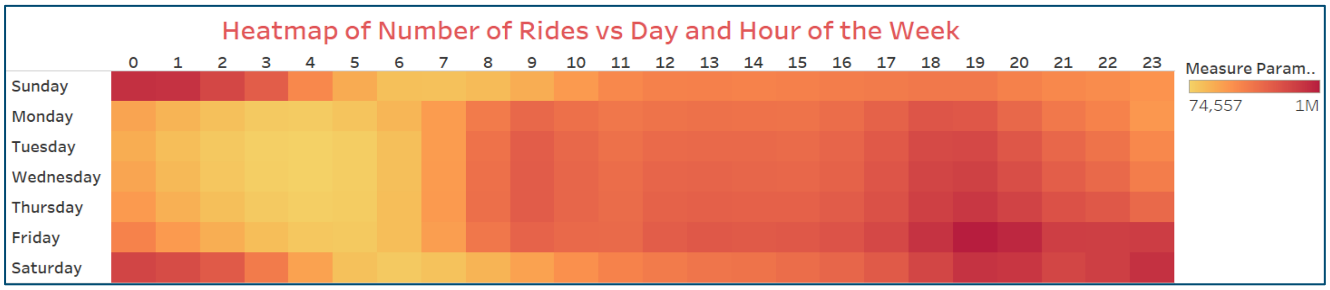


Figure 9

This heatmap (figure 9) gives a clear picture about dependence of both the parameters on number of rides. From 0-6 on Monday through Friday there are less number of rides, hence less number of taxis required to balance the demand. Chicagoans generally like to relax on Friday and Saturday, traveling a lot within the city, drinking which gives rise to more number of rides. Also, talking about allocation of resources CTA should have large number of taxis from 0th hour to 6th hour on Saturday and Sunday. *Heatmap for Total trip is similar to the number of rides.*

As mentioned previously, heatmap is entirely different for average fare. If some of the cabbies are looking to make more money per ride, following heatmap is very useful for them. It paints a profitable picture where there is high return on investment, if a cabby is willing to serve from 4AM-7AM. The traffic density at this time is less which aides higher fuel efficiency.

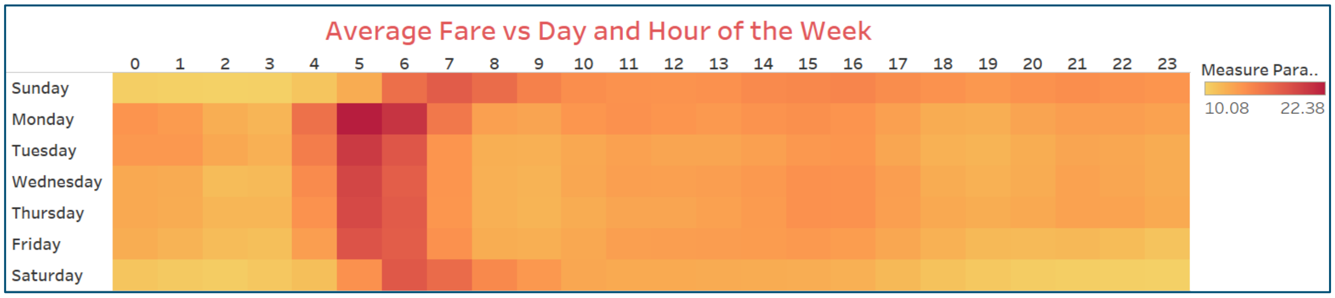


Figure 10

Also keeping in mind that since these rides are lengthier time and distance wise, from our own experiences, I have always tipped more in such situations. Hence, I plotted the similar heatmap for average tips, our intuition was right as many people appreciated the work done by cabbies and that too at 5AM or 6AM. Average tips were fairly high in the night time as well.

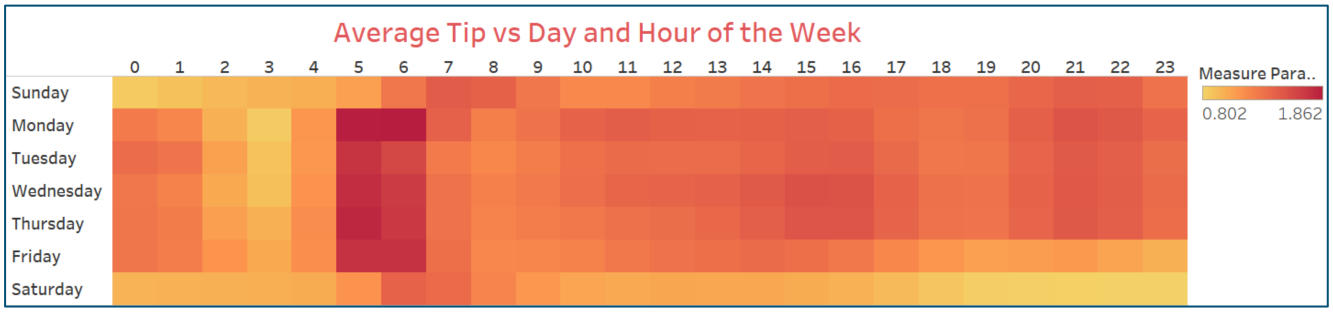


Figure 11

So far, I understood the importance of time features. Next, I wanted to understand the importance of pick up and drop off location.

Chicago's taxi pickup declines are not evenly distributed among the city's 77 community areas. For example, the Loop, Chicago's central business district, shows a 23% annual decline, while Logan Square on the northwest side shows a 50% annual decline. In general, the areas located closest to the central business district show smaller declines in taxi activity.

I defined 5 particular community areas—the Loop, Near North Side, Near West Side, Near South Side, and O'Hare Airport—as the "core", then compared pickups inside and outside of the core. (Figure 12) As of November 2016, pickups inside the core shows a 27% annual decline compared to a 42% annual decline outside of the core. On a cumulative basis, core pickups have declined 39% since June 2014, while non-core pickups have declined a whopping 65%. **[3]**

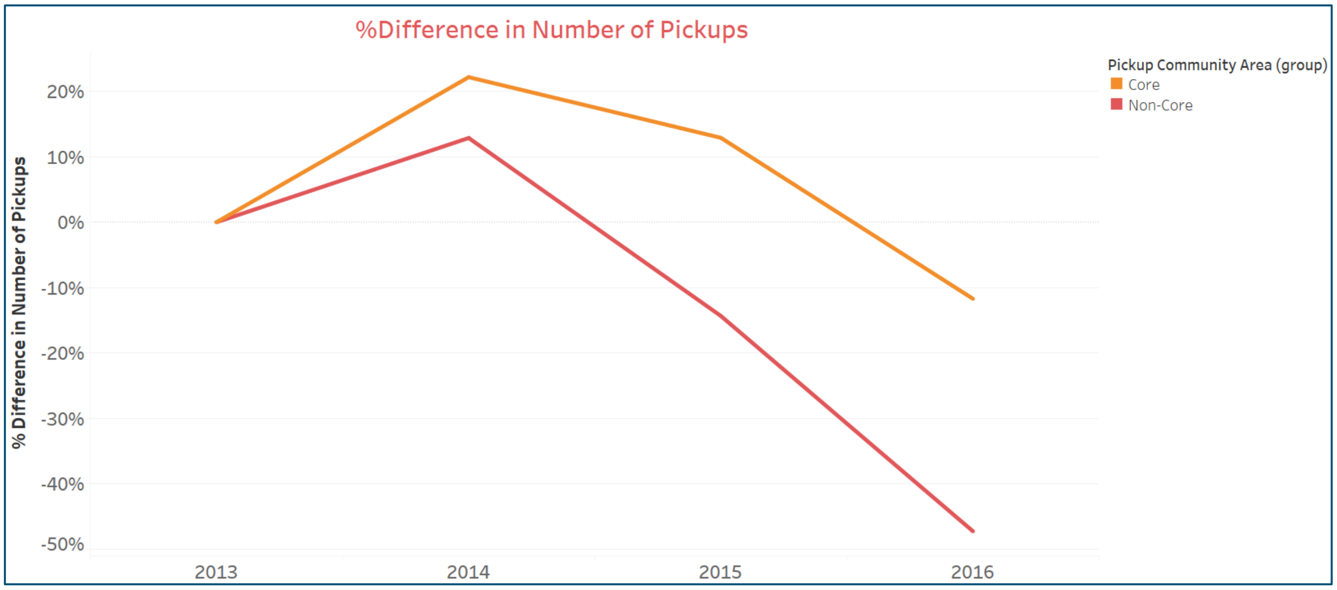


Figure 12

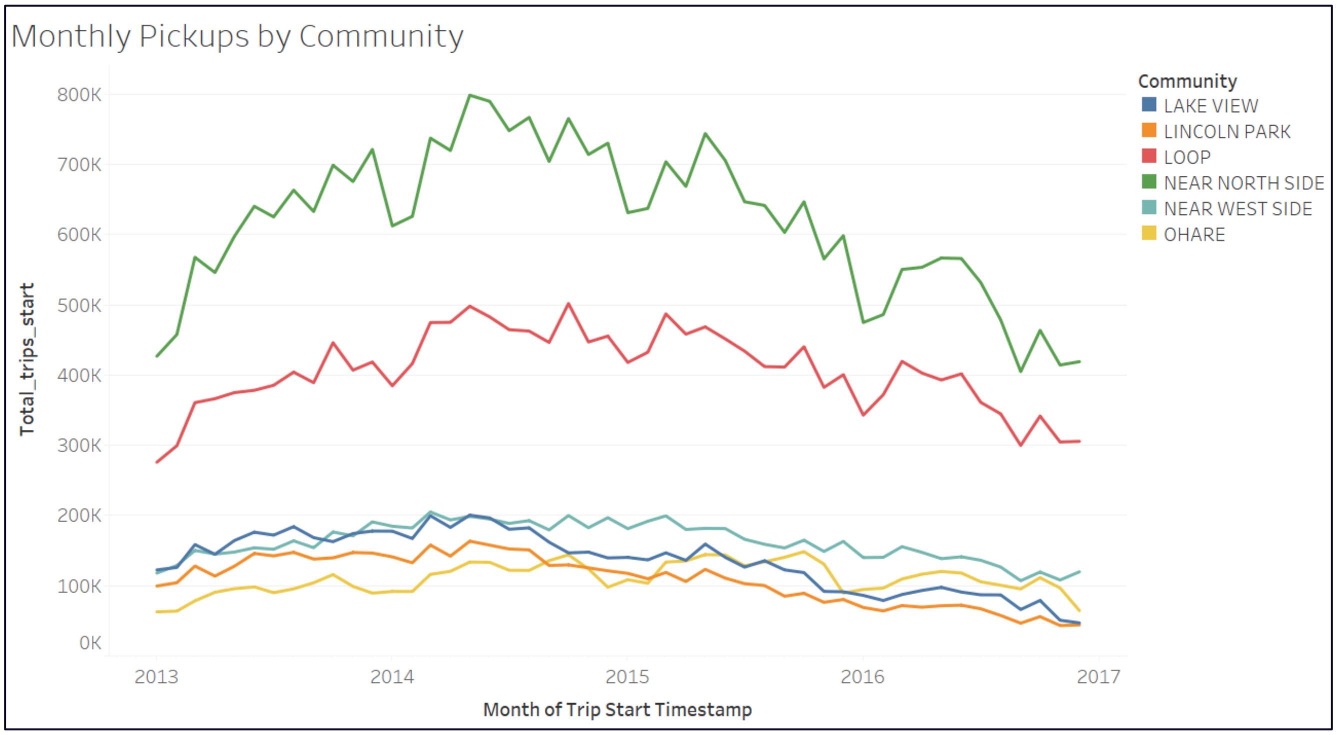


Figure 14

The above figure depicts the distribution of number of rides for six different areas. It again emphasizes the importance of those 'Core' areas I talked about earlier.

One of the parameter that is most important for any taxi company to know is what are some of the most influential pick up areas in a particular city. Generally, these are the areas which have a great revenue potential and by focusing on these areas one can increase a taxi's overall profit. As mentioned earlier there are some of the core areas, the reason I called them as a core area was because of their extreme affluence, typified by the Magnificent Mile, Gold Coast, Navy Pier, and its world-famous skyscrapers. Magnificent mile and Gold Coast are among the top ten richest neighborhoods in America. Navy Pier is the number one tourist destination in Chicago City, drawing nearly nine million visitors annually. **[4]**

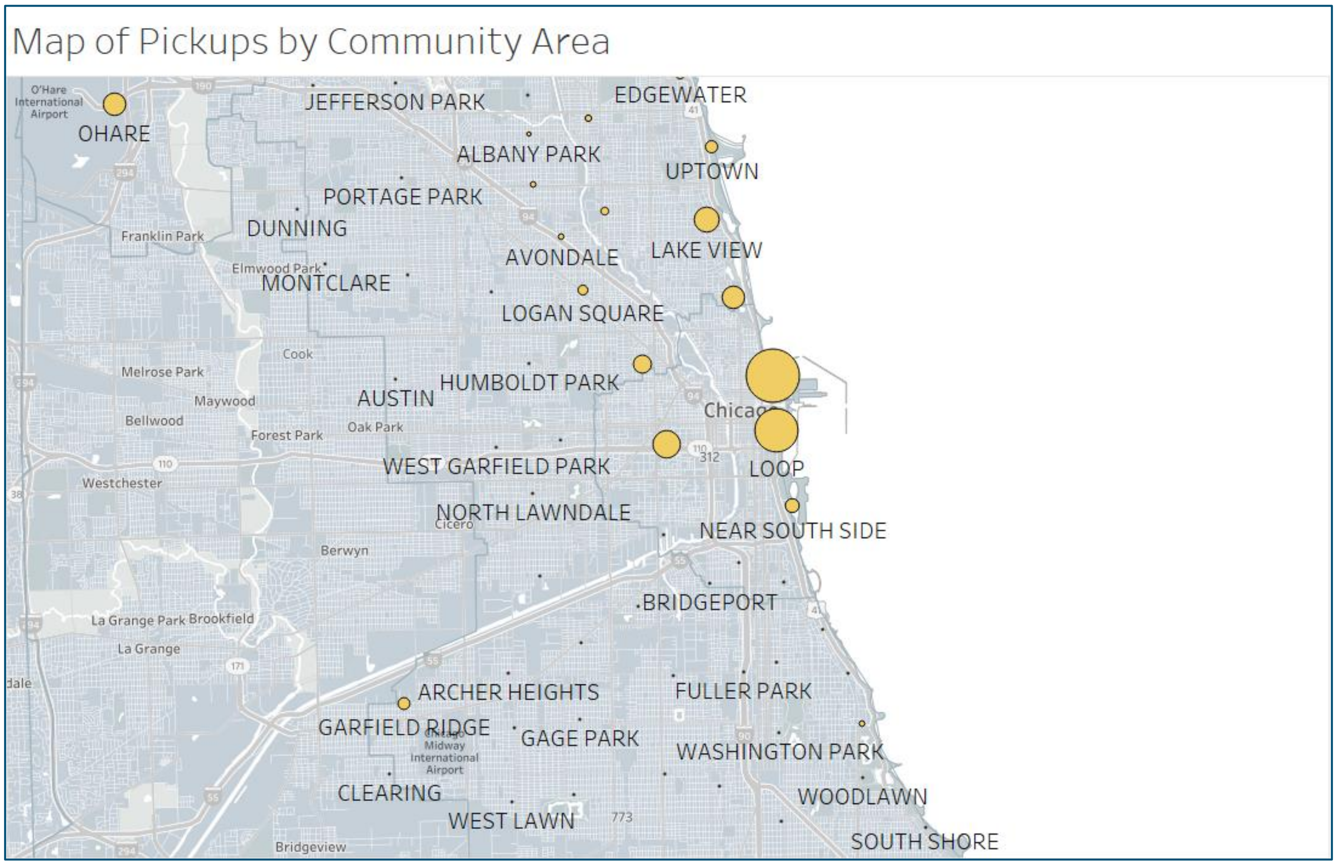


Figure 13

Because of such affluence, business and tourism these areas account for a major portion of revenue for the Chicago Taxis. O'Hare, as mentioned before, is the second-busiest airport in the world by the number of takeoffs and landings and also contributes majorly towards the revenue. **[2]**

So far in this report I have talked about how time have affected taxi revenues, how location plays an important role in determining whether or not return on investment will be higher. Its time to talk about other things which has a huge impact on taxi revenues.

Remember those holidays I talked about, let's discuss about them in detail.

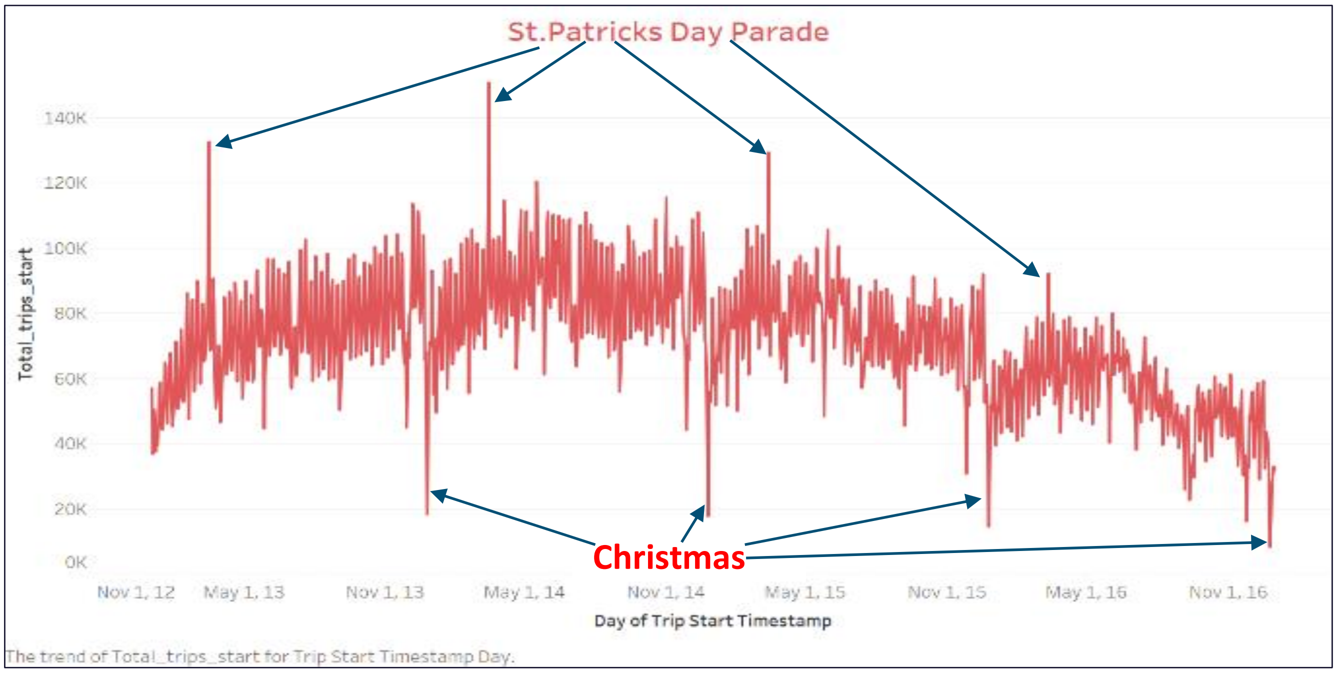


Figure 15

Holidays have always been special to the Chicago City. From raucous pub crawls to lively parades, there is nothing quite like St. Patrick's Day in Chicago. In March, Irish taverns are packed with revelers, jovial crowds jam the city streets and the Chicago River sparkles brilliant shades of emerald green. With so much going on be it the downtown parade or dyeing of the Chicago river, Chicagoans travels a lot and they generally prefer public taxis. **[5]**

It turns out that St. Patrick's Day parade which is held in march of every year accounts for maximum number of taxi trips on a particular day.

One of the important national holiday is Labor Day, created to celebrate the contributions of the American worker. It falls on the first Monday of September, resulting in a dearly coveted three-day weekend. Labor Day is essentially a day off for cabbies and this results in less number of ride on that day.

Memorial Day, Thanksgiving and Christmas of every year has the least number of taxi trips respectively. This is again due to the fact that there are few Taxi's in service on these holidays. This explicit visualization motivated us to incorporate important holidays as parameters in our predictive model.

After all of the Exploratory Data Analysis, it was of utmost importance to look for anomaly in this huge dataset, hence while querying the data most of the outliers were treated by ensuring following things: -

- The ratio of trip total and trip miles should be greater than 2, for instance if trip distance is 2 miles, the charge should be at least $4. Also, to prevent high outliers the same ratio should also be less than 10.
- With the city traffic and speed restrictions, speed i.e. ratio of trip miles and trip hour was kept less than 70. This enforces another restriction that trip seconds should have non-zero values.
- The minimum fare amount for any trip in Chicago is $2.25, hence all the fare values should have non-zero values.

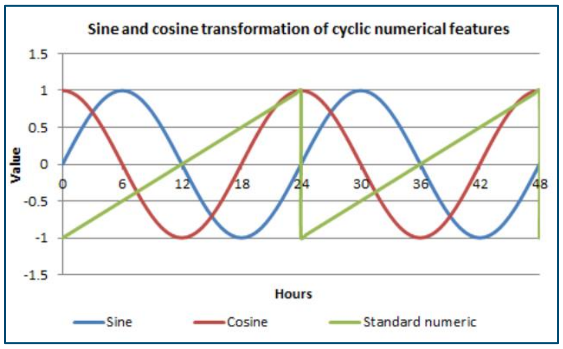
Predictive Model

As depicted by EDA, time plays an important role in the prediction of total fare amount. The major challenge in including time as a predictor is that it is inherently cyclical. Time has components which are minutes, hours, seconds, day of week, week of month, month, season, and so on all follow cycles. Ecological features like tide, astrological features like position in orbit, spatial features like rotation or longitude, visual features like color wheels are all naturally cyclical. [6]

I wanted our statistical learning algorithm to incorporate the cyclic nature of the time. For instance, I want our predictive model to see that 23:55 and 00:05 are 10 minutes apart, but as it stands, those times will appear to be 23 hours and 50 minutes apart! Therefore, I broke the time variable into small pieces.

Independent Cyclic Features: -

- Hourfloat: - Hour of the day as a (binned!) floating point number between 0 and 1. Since there are 24 hours in a day, our bin contains total 24 divisions. If hour and minute component of trip_start_timestamp has a value 14:20, the minute component is converted into hourly component and whole fraction value is divided by 24. So, 14:20 becomes, $\frac{14.33}{24}$ =0.597.
- Day_cat: -Day of the week as a categorical feature: "Monday", "Tuesday", etc.
- Day_num: - Day of the week as a numerical feature going from 0 (Monday morning, start of the week) to 1 (Sunday night), European style. With 24 bins, Tuesday afternoon 14:20:00 would translate to $\frac{1+\text{hourfloat}}{7} = 0.2292$. [7]



Similar procedure was performed to evaluate monthfloat and yearfloat. But to remove cyclicity the binned "hourfloat " variable converted to a sine and cosine version so that time nicely 'loops' rather than going saw-like when it traverses midnight. As depicted in the figure, this transformation doesn't have any magic powers, but it can make it easier for a model to find the right patterns. [reference]

- hourfloat_sin: -Binned "hourfloat" variable converted to a cosine version. "hourfloat_sin" = $\cos(\text{hourfloat} \times 2 \times \pi)$**.** For 24 bins per day, 14:20:00 would translate to, $\cos(0.597 \times 2 \times \pi)$**=** 0.99.
- hourfloat_cos: -Binned "hourfloat" variable converted to a cosine version. "hourfloat_cos" = $\sin(\text{hourfloat} \times 2 \times \pi)$. For 24 bins per day, 14:20:00 would translate to, $\sin(0.597 \times 2 \times \pi)$**=** 0.05.

Similarly predictor space with parameters day_sin, day_cos, year_sin, year_cos,month_sin, month_cos was built. I can feed the sin and cos features into our machine learning model, and the cyclical nature of 24-hour time will carry over.

After visualizing important features affecting trip fare, we are going to build predictive models and compare the accuracy of them to predict the total trip fare for hourly, daily and weekly data. I chose to use three algorithms for the prediction task: Linear Regression, Random Forest and LightGBM. LightGBM is a relatively new algorithm in the data science field and focuses on accuracy with speed on large datasets.

LightGBM is a gradient boosting technique that uses tree-based learning algorithm. LightGBM grows tree vertically as compared to the other techniques which grow trees horizontally. In short, LightGBM grows tree leaf-wise instead of level-wise.

A simple representation of LightGBM is as follows:

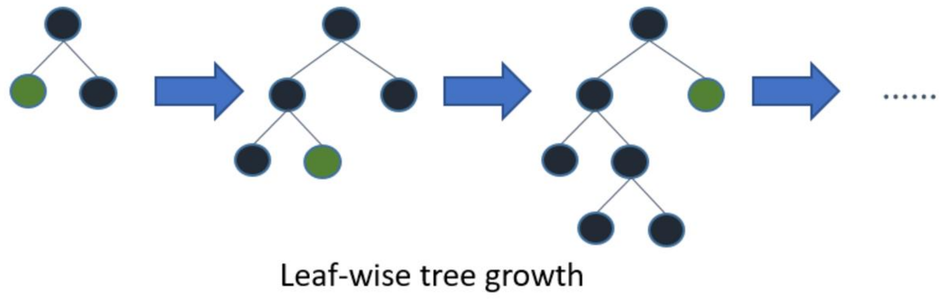


Figure 2 LightGBM

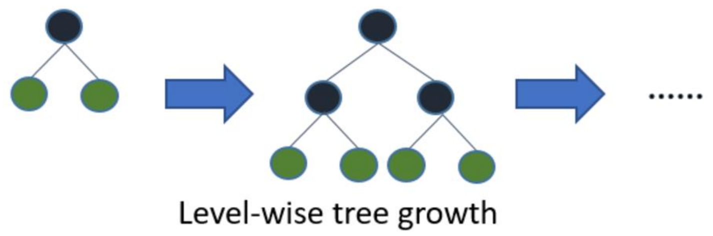


Figure 3 Other Tree Models

To measure the model's accuracy, I decided to use RMSE and R-squared metrics. While calculating RMSE, predicted values far from the true value are penalized more, it seemed apt to use RMSE to check the accuracy of our models.

I have in total four categorical values in the dataset including 'day_cat' and 'Special_day' which were added after feature engineering. In the table below, I can look at the different companies operating in Chicago for the hourly dataset.

It seems that there are only five top companies having highest distribution of the number of taxi rides in the city. The companies having total number of rides less than 1500 in the four-year span from 2013-2016 were clubbed together as the same class 'other'. The same approach was used for weekly and daily data.

Companies	# Rides
Taxi Affiliation Services	17048
Dispatch Taxi Affiliation	10607
Choice Taxi Association	5579
Northwest Management LLC	3022
KOAM Taxi Association	1990
Top Cab Affiliation	1299
Chicago Medallion Leasing INC	560
Chicago Medallion Management	349
Blue Ribbon Taxi Association Inc.	60
3201 - C&D Cab Co Inc	22
6743 - 78771 Luhak Corp	22
585 - 88805 Valley Cab Co	22
3141 - 87803 Zip Cab	22
….	
….	

Payment Type	#Rides
Cash	69811
Credit Card	18179
No Charge	399
Dispute	48
Unknown	21
Pcard	10
Prcard	9

Similarly, for the payment types as shown in the table above, I clubbed all the classes other than Cash and Credit Card because Cash and Credit Card are the dominant payment mode and the total number of payments through other mode seems to be insignificant.

Cross-validation

To reduce the overfitting in machine learning algorithms, I are going to use cross-validation to optimize the hyper-parameters before predicting on the test dataset.

5-fold Cross validation was performed in case of Random Forest and LightGBM. Then best parameters from cross-validation were used to predict the year 2017 total fare.

For Random Forest, cross validation is done using **n_estimators**, the number of trees constructed in the forest, equal to 10, 15 and 20. The **max_features** (the number of features allowed in each tree) parameter is set to auto, sqrt and log2.

For LightGBM, cross validation is done using **num_leaves** (number of leaves in full tree) equal to 20,40,60,80 and **learning rate** being set to 0.01 and 0.1 (the rate at which the model converges).

After optimizing the hyper-parameters, the models were fit on the testing data. I used three different datasets to predict hourly, daily, and weekly median fare. The hourly data have around 33000 rows, daily around 1600 rows and weekly data around 210 rows. Metrics R-squared and RMSE for training and testing data is shown below for each model.

RESULTS

Model		Model Metrics	Training	Testing
Linear Regression	Hourly	R-Squared	0.5258	0.5294
		RMSE	3.0326	3.2577
	Daily	R-Squared	0.7287	0.5484
		RMSE	19.5233	21.4986
	Weekly	R-Squared	0.8772	0.8539
		RMSE	88.4606	81.4277
Random Forest	Hourly	R-Squared	0.9668	0.6887
		RMSE	0.80	2.65
	Daily	R-Squared	0.9726	0.6998
		RMSE	6.20	17.528
	Weekly	R-Squared	0.9806	0.8172
		RMSE	37.83	91.078
LightGBM	Hourly	R-Squared	0.8796	0.7405
		RMSE	1.52	2.419
	Daily	R-Squared	0.9497	0.7327
		RMSE	8.40	16.54
	Weekly	R-Squared	0.9999	0.5091
		RMSE	2.74	149.239

Interpretation

Linear Regression seems to perform worst against Random Forest and LightGBM for the hourly and daily data. It is not able to catch the non-linear relationship in the model. However, Random Forest and LightGBM seems to be overfitting the data even after cross-validation. The possible explanation for overfitting is the less number of data points. Since, I reduced the number of data points from 110 million rows to 33000 rows in case of hourly data, it makes sense that the models are not able to catch the variability in the data.

For the weekly data, Linear Regression performs better than the other two models which can be noticed in the almost similar values both for R-squared and RMSE. Since LightGBM performs better on bigger datasets, it might be the reason that there is a huge difference between R-squared and RMSE value both for training and testing data.

If I consider only hourly and daily data, which has considerably large number of points than the weekly data, LightGBM performs better than the other two Linear Regression and Random Forest. It is able to predict both the hourly and daily median fare with 74% accuracy which seems a good value given variability in the data. I can say that the aggregated data is not a represented sample of the complete dataset. I could have predicted better if I had aggregated by the community area. Then I would have sufficient data points to make accurate predictions.

Feature Importance Plots:

Hourly Data

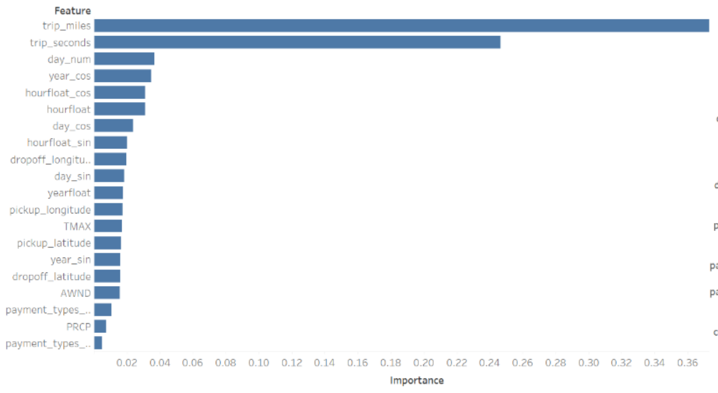


Figure 1 Random Forest

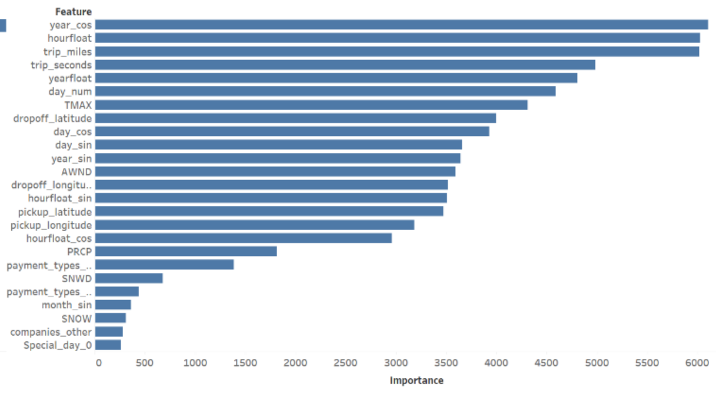


Figure 2 LightGBM

Daily Data

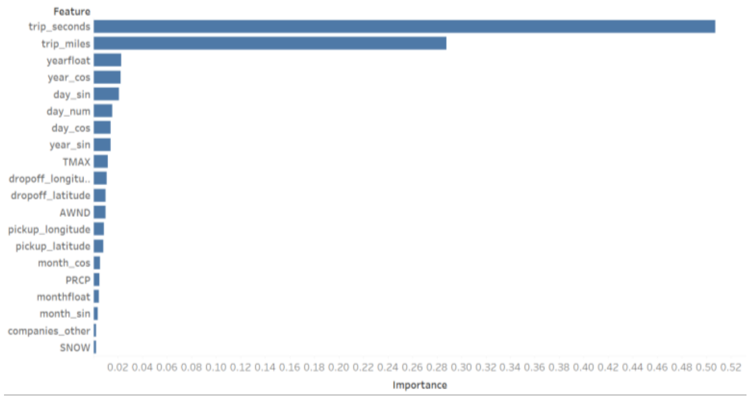


Figure 3 Random Forest

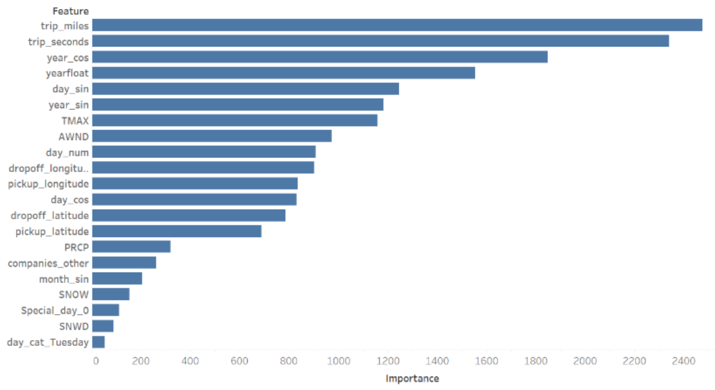


Figure 4 LightGBM

Weekly Data:

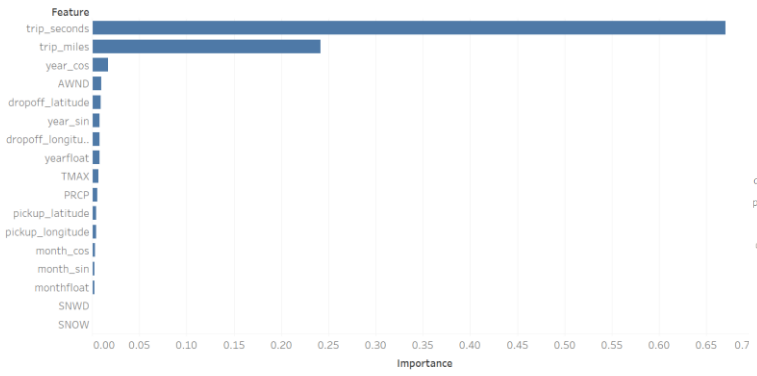


Figure 6 Random Forest

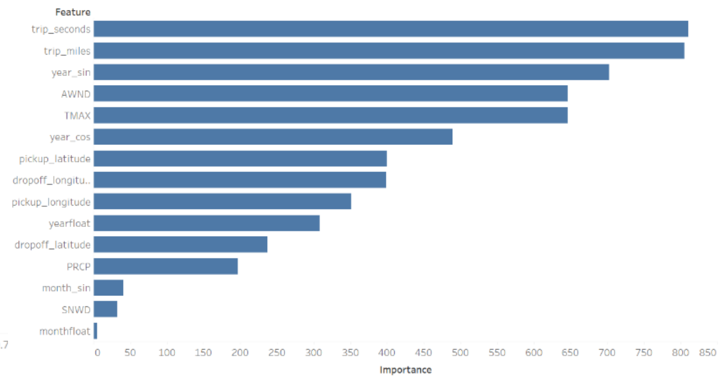


Figure 5 LightGBM

The feature importance plots show that trip miles and trip seconds are the most important factors in predicting the total fare. The other factors which seems to be important are hourfloat, year_cos, year_sin and all the other time related features derived from the timestamp

Conclusion

In this project with the help of explicit of data visualization and predictive modeling I analyzed how Chicago Transportation Authority (CTA) should develop strategies in order to stay ahead of the competition. Tech savvy giants like Uber and Lyft have advanced their businesses by using these common but important data science tools and techniques. I showed in this report how return on investment for the taxi business can be increased by focusing on some key points.

In the report I mentioned how business areas of Chicago are the key areas for revenue growth. Even with the emergence of Uber, number of rides have not gone down in these areas as compared to other parts of the city. Holidays and weather data plays important role in determination of number of rides and total fare; hence CTA should consider all these parameters while allocating their resources. I also addressed some anomalies like why at some point in the day total fare is less but average fare is very high.

I was able to build fairly accurate predictive models for weekly, daily and hourly aggregated data. This will provide an edge to Chicago Taxis as they will be able to eliminate some uncertainties associated with the ride. Our light GBM yields out the most important factors that a taxi firm should use to determine the total trip amount.

For future work I want to make use of the entire dataset for predictive modeling by making use of Amazon Web Services.

References

[1] USA TODAY, "Chicago cabbies say industry is teetering toward collapse," 5 June 2017. [Online]. Available: https://www.usatoday.com/story/news/2017/06/05/chicago-cabbies-say-industry-teetering-toward-collapse/102524634/.

[2] Wikipedia, "O'Hare International Airport," [Online]. Available: https://en.wikipedia.org/wiki/O%27Hare_International_Airport.

[3] T. W. Schneider, "Chicago's Public Taxi Data," 17 January 2017. [Online]. Available: http://toddwschneider.com/posts/chicago-taxi-data/.

[4] Wikipedia, "Near North Side, Chicago," 10 March 2018. [Online]. Available: https://en.wikipedia.org/wiki/Near_North_Side,_Chicago.

[5] C. v. o. WGN9, "Chicago best city to celebrate St. Patrick's Day in America, website says," 9 March 2018. [Online]. Available: http://wgntv.com/2018/03/09/chicago-best-place-to-celebrate-st-patricks-day-in-america-website-says/.

[6] I. London, "Encoding cyclical continuous features - 24-hour time," 31 July 2016. [Online]. Available: https://ianlondon.github.io/blog/encoding-cyclical-features-24hour-time/.

[7] H. C. 109, "NYC Taxi Dataset Project - Data Prep AWS Spark," 10 December 2015. [Online]. Available: https://github.com/sdaulton/TaxiPrediction/blob/master/2.%20DataPrepAWSSpark.ipynb.

Code

Code-1 Loading dataset into R for cleaning

#Loading libraries

```
library(dplyr)
library(data.table)
```

#Reading dataset and defining metadata

```
df= fread("H:/Competition/Chicago_taxi_trips.csv",
        colClasses = c("Trip ID"="character","Taxi ID"= "character",
       "Trip Start Timestamp"= "Date", "Trip End Timestamp"="Date",
      "Trip Seconds"= "integer","Trip Miles"= "integer",
      "Pickup Census Tract"= "character", "Dropoff Census Tract"="character",
      "Pickup Community Area"= "integer","Pickup O'Hare Community Area"="integer", "Dropoff Community Area"=
"integer",
      "Fare"="character","Tips"="character", "Tolls"= "character","Extras"=  "character","Trip Total"= "character",
     "Payment Type"=  "factor","Company"= "factor",
   "Pickup Centroid Latitude"= "integer","Pickup Centroid Longitude"= "integer",
   "Pickup Centroid Location"= "character","Dropoff Centroid Latitude"= "integer",
     "Dropoff Centroid Longitude"= "integer","Dropoff Centroid  Location" ="character"))
```

#Renaming all the variables

```
df= df %>%
 rename(unique_key=`Trip ID`, taxi_id= `Taxi ID`,
       trip_start_timestamp=`Trip Start Timestamp`,
       trip_end_timestamp= `Trip End Timestamp`,
       trip_seconds=`Trip Seconds`, trip_miles=`Trip Miles`,
       pickup_census_tract= `Pickup Census Tract`,
       dropoff_census_tract= `D Census Tract`,
       pickup_community_area= `Pickup Community Area`,
       dropoff_community_area= `Dropoff Community Area`,
       fare=Fare, tips= Tips, tolls= Tolls, extras= Extras,
       trip_total= `Trip Total`, payment_type= `Payment Type`,
       company= Company, pickup_latitude= `Pickup Centroid Latitude`,
       pickup_longitude= `Pickup Centroid Longitude`,
       pickup_location= `Pickup Centroid Location`,
       dropoff_latitude= `Dropoff Centroid Latitude`,
       dropoff_longitude= `Dropoff Centroid Longitude`,
       dropoff_location= `Dropoff Centroid  Location`)
```

Removing AM/PM string from the timestamp values

```
df[,3]=  lapply(df[,3],as.POSIXct ,format='%m/%d/%Y %I:%M:%S %p')
df[,4]=  lapply(df[,4],as.POSIXct ,format='%m/%d/%Y %I:%M:%S %p')
```

Removing $ string from the monetary columns

```
df=df %>%
  mutate(fare= substring(fare,2, nchar(fare)),
        tips= substring(tips,2,nchar(tips)),
tolls= substring(tolls,2,nchar(tolls)),
extras= substring(extras,2,nchar(extras)),
trip_total= substring(trip_total,2,nchar(trip_total)))

df=df %>%
  mutate(fare= as.numeric(fare),
        tips= as.numeric(tips),
        tolls= as.numeric(tolls),
        extras= as.numeric(extras),
        trip_total=as.numeric(trip_total))

df= df %>%
  rename(unique_key=`Trip ID`, taxi_id= `Taxi ID`,
        trip_start_timestamp=`Trip Start Timestamp`,
        trip_end_timestamp= `Trip End Timestamp`,
        trip_seconds=`Trip Seconds`, trip_miles=`Trip Miles`,
        pickup_census_tract= `Pickup Census Tract`,
        dropoff_census_tract= `D Census Tract`,
        pickup_community_area= `Pickup Community Area`,
        dropoff_community_area= `Dropoff Community Area`,
        fare=Fare, tips= Tips, tolls= Tolls, extras= Extras,
        trip_total= `Trip Total`, payment_type= `Payment Type`,
        company= Company, pickup_latitude= `Pickup Centroid Latitude`,
        pickup_longitude= `Pickup Centroid Longitude`,
        pickup_location= `Pickup Centroid Location`,
        dropoff_latitude= `Dropoff Centroid Latitude`,
        dropoff_longitude= `Dropoff Centroid Longitude`,
        dropoff_location= `Dropoff Centroid  Location`)
```

After uploading the dataset on Google Cloud Platform after creating a bucket, this dataset was linked to a project and finally Google Bigquery with the help of a relational schema. Different subsets were formed to execute hourly, daily and weekly analysis.
Weekly data contained 212 rows for the training set. (2013-2016)
Daily data contained 1461 rows for the training set. (2013-2016)
Hourly data contained 1461 rows for the training set. (2013-2016)

Test dataset contained values from 2017.

Code-2 Subset selection on Google Bigquery

MySQL Queries – Weekly Data

In the following query I calculated the weekly total (SUM) taxi fares for each Taxi ID for each week in the data. The whole dataset was grouped by Taxi_ID and week of the date. Apart from trip seconds, trip miles and trip total median values were selected for all other parameters. I also took care of the **outliers** in the query. I selected the following constraints as mentioned above: -

- The ratio of trip total and trip miles should be greater than two, for instance if trip distance is 2 miles, the charge should be at least $4. Also, to prevent high outliers the same ratio should be less than 10.
- With the city traffic and speed restrictions, speed i.e. ratio of trip miles and trip hour was kept less than 70. This enforces another restriction that trip seconds should have non-zero values.
- The minimum fare amount for any trip in Chicago is $2.25, hence all the fare values should have non-zero values.

TABLE-1

```
SELECT
  WEEK(trip_start_timestamp) AS time_identifier_week,
  YEAR(trip_start_timestamp) AS time_identifier_year,
  taxi_id,
  COUNT( unique_key ) AS Number_rides,
  NTH(501, QUANTILES( trip_start_timestamp , 1001)) AS trip_start_timestamp ,
  NTH(501, QUANTILES( trip_end_timestamp , 1001)) AS trip_end_timestamp ,
  SUM(trip_seconds) AS trip_seconds,
  NTH(501, QUANTILES( payment_type , 1001)) AS payment_type ,
  NTH(501, QUANTILES( company , 1001)) AS company ,
  SUM(trip_miles) AS trip_miles,
  SUM(trip_total) AS trip_total ,
  NTH(501, QUANTILES( payment_type , 1001)) AS payment_types ,
  NTH(501, QUANTILES( company , 1001)) AS companies ,
  NTH(501, QUANTILES( pickup_latitude , 1001)) AS pickup_latitude ,
  NTH(501, QUANTILES( pickup_longitude , 1001)) AS pickup_longitude ,
  NTH(501, QUANTILES( pickup_location , 1001)) AS pickup_location ,
  NTH(501, QUANTILES( dropoff_latitude , 1001)) AS dropoff_latitude ,
  NTH(501, QUANTILES( dropoff_longitude , 1001)) AS dropoff_longitude ,
  NTH(501, QUANTILES( dropoff_location , 1001)) AS dropoff_location

  FROM [authentic-block-200023:Taxis.Final_merged_data_2013_to_2016]
  WHERE ((trip_total)/(trip_miles))<10 AND ((trip_total)/(trip_miles))>2
  AND trip_total>0 AND trip_miles>0
  AND trip_seconds>0 AND (( trip_miles*3600 )/( trip_seconds ))<70
  GROUP BY  taxi_id,time_identifier_week, time_identifier_year
  ORDER BY time_identifier_year, time_identifier_week
```

This query yielded out 602327 rows. I wanted those rows which contained median of the total taxi fares for each Taxi ID. To do that I created a TABLE-2 which contained only median values for the trip_total field.

TABLE-2

```
SELECT
NTH(501, QUANTILES( trip_total , 1001)) AS trip_total, time_identifier_week, time_identifier_year
FROM [authentic-block-200023:1234.week_table_updated]
GROUP BY time_identifier_week, time_identifier_year
ORDER BY  time_identifier_year,time_identifier_week
```

After creation of these two tables I used a CROSS JOIN to extract those rows from TABLE-1 which contained median of the total taxi fares for each Taxi ID. I CROSS JOINED the two tables on week, year and taxi_id.

FINAL DATA (WEEKLY): -

```
SELECT * FROM [authentic-block-200023:1234.week_table_updated] AS week_table
CROSS JOIN [authentic-block-200023:1234.week_table_median] AS median_table
WHERE week_table.trip_total = median_table.trip_total AND
week_table.time_identifier_week = median_table.time_identifier_week
AND week_table.time_identifier_year = median_table.time_identifier_year
ORDER BY week_table.time_identifier_week, week_table.time_identifier_year
```

This dataset was used for weekly predictive analysis

MySQL Queries – Daily Data

Similar strategy was used to obtain data for daily analysis with an additional group by **day.**

TABLE-1

```
SELECT
  DAY(trip_start_timestamp) AS time_identifier_day,
  YEAR(trip_start_timestamp) AS time_identifier_year,
  MONTH(trip_start_timestamp) AS time_identifier_month,
  taxi_id,
  COUNT( unique_key ) AS Number_rides,
  NTH(501, QUANTILES( trip_start_timestamp , 1001)) AS trip_start_timestamp ,
  NTH(501, QUANTILES( trip_end_timestamp , 1001)) AS trip_end_timestamp ,
  SUM(trip_seconds) AS trip_seconds,
  NTH(501, QUANTILES( payment_type , 1001)) AS payment_type ,
  NTH(501, QUANTILES( company , 1001)) AS company ,
  SUM(trip_miles) AS trip_miles,
  SUM(trip_total) AS trip_total ,
  NTH(501, QUANTILES( payment_type , 1001)) AS payment_types ,
  NTH(501, QUANTILES( company , 1001)) AS companies ,
  NTH(501, QUANTILES( pickup_latitude , 1001)) AS pickup_latitude ,
  NTH(501, QUANTILES( pickup_longitude , 1001)) AS pickup_longitude ,
  NTH(501, QUANTILES( pickup_location , 1001)) AS pickup_location ,
  NTH(501, QUANTILES( dropoff_latitude , 1001)) AS dropoff_latitude ,
  NTH(501, QUANTILES( dropoff_longitude , 1001)) AS dropoff_longitude ,
  NTH(501, QUANTILES( dropoff_location , 1001)) AS dropoff_location

  FROM [authentic-block-200023:Taxis.Final_merged_data_2013_to_2016]
  WHERE ((trip_total)/(trip_miles))<10 AND ((trip_total)/(trip_miles))>2 AND trip_total>0 AND trip_miles>0
  AND trip_seconds>0 AND (( trip_miles*3600 )/( trip_seconds ))<70 AND
  GROUP BY  taxi_id,time_identifier_day, time_identifier_month,time_identifier_year
  ORDER BY time_identifier_year,time_identifier_month, time_identifier_day
```

TABLE-2

```
SELECT
NTH(501, QUANTILES( trip_total , 1001)) AS trip_total, time_identifier_day, time_identifier_month,
time_identifier_year
FROM [authentic-block-200023:1234.daily_table_updated]
GROUP BY time_identifier_day, time_identifier_month , time_identifier_year
ORDER BY  time_identifier_year, time_identifier_month, time_identifier_day
```

FINAL DATA (DAILY): -

```
SELECT *
FROM [authentic-block-200023:1234.daily_table_updated] AS daily_table
CROSS JOIN [authentic-block-200023:1234.daily_table_median] AS median_table
WHERE daily_table.trip_total = median_table.trip_total
AND daily_table.time_identifier_day = median_table.time_identifier_day
AND daily_table.time_identifier_year = median_table.time_identifier_year
AND daily_table.time_identifier_month = median_table.time_identifier_month
ORDER BY daily_table.time_identifier_year,daily_table.time_identifier_month,daily_table.time_identifier_day
```

This dataset was used for daily predictive analysis.

MySQL Queries – Hourly Data

Similar strategy was used to obtain data for daily analysis with an additional group by **day** and grouped by **hour.**

TABLE-1

```
SELECT
  HOUR(trip_start_timestamp) AS time_identifier_hour,
  DAY(trip_start_timestamp) AS time_identifier_day,
  YEAR(trip_start_timestamp) AS time_identifier_year,
  MONTH(trip_start_timestamp) AS time_identifier_month,
  taxi_id,
  COUNT( unique_key ) AS Number_rides,
  NTH(501, QUANTILES( trip_start_timestamp , 1001)) AS trip_start_timestamp ,
  NTH(501, QUANTILES( trip_end_timestamp , 1001)) AS trip_end_timestamp ,
  SUM(trip_seconds) AS trip_seconds,
  NTH(501, QUANTILES( payment_type , 1001)) AS payment_type ,
  NTH(501, QUANTILES( company , 1001)) AS company ,
  SUM(trip_miles) AS trip_miles,
  SUM(trip_total) AS trip_total ,
  NTH(501, QUANTILES( payment_type , 1001)) AS payment_types ,
  NTH(501, QUANTILES( company , 1001)) AS companies ,
  NTH(501, QUANTILES( pickup_latitude , 1001)) AS pickup_latitude ,
  NTH(501, QUANTILES( pickup_longitude , 1001)) AS pickup_longitude ,
  NTH(501, QUANTILES( pickup_location , 1001)) AS pickup_location ,
  NTH(501, QUANTILES( dropoff_latitude , 1001)) AS dropoff_latitude ,
  NTH(501, QUANTILES( dropoff_longitude , 1001)) AS dropoff_longitude ,
  NTH(501, QUANTILES( dropoff_location , 1001)) AS dropoff_location

  FROM [authentic-block-200023:Taxis.Final_merged_data_2013_to_2016]
  WHERE ((trip_total)/(trip_miles))<10 AND ((trip_total)/(trip_miles))>2 AND trip_total>0 AND trip_miles>0
  AND trip_seconds>0 AND (( trip_miles*3600 )/( trip_seconds ))<70
  GROUP BY  taxi_id,time_identifier_hour, time_identifier_day, time_identifier_month,time_identifier_year
  ORDER BY time_identifier_year,time_identifier_month, time_identifier_day
```

TABLE-2

```
SELECT
NTH(501, QUANTILES( trip_total , 1001)) AS trip_total, time_identifier_hour, time_identifier_day,
time_identifier_month, time_identifier_year
FROM [authentic-block-200023:1234.hourly_table_updated]
GROUP BY time_identifier_hour ,time_identifier_day, time_identifier_month , time_identifier_year
ORDER BY  time_identifier_hour ,time_identifier_year, time_identifier_month, time_identifier_day
```

FINAL DATA (HOURLY): -

```
SELECT *
FROM [authentic-block-200023:1234. hourly_table_updated] AS hourly_table
CROSS JOIN [authentic-block-200023:1234. hourly_table_median] AS median_table
WHERE hourly_table.trip_total = median_table.trip_total
AND hourly_table.time_identifier_hour = median_table.time_identifier_hour
AND hourly_table.time_identifier_day = median_table.time_identifier_day
AND hourly_table.time_identifier_year = median_table.time_identifier_year
AND hourly_table.time_identifier_month = median_table.time_identifier_month
ORDER BY daily_table.time_identifier_year,daily_table.time_identifier_month,daily_table.time_identifier_day,
time_identifier_hour
```

This dataset was used for hourly predictive analysis

Code-3 Predictive Modeling on R and Python

```
# Loading Libraries
library(dplyr)
library(data.table)
library(anytime)
library(lubridate)
library(lightgbm)
library(caret)


# Loading Aggregated Data
df= fread("hourly_data_2013-2016.csv")
df=df %>%
  mutate(trip_start_timestamp = substring(trip_start_timestamp,1, nchar(trip_start_timestamp)-4),
       trip_end_timestamp= substring(trip_end_timestamp, 1,nchar(trip_end_timestamp)-4))

df$trip_start_timestamp=  as.POSIXct(df$trip_end_timestamp,format="%Y-%m-%d %H:%M:%S")
df$trip_end_timestamp=  as.POSIXct(df$trip_end_timestamp,format="%Y-%m-%d %H:%M:%S")
df$time_identifier_day= date(df$trip_start_timestamp)

df1= df %>%
  select( -taxi_id,
       -pickup_location,
       -dropoff_location)


## Grouping less frequent payment mode as ‘others’
df1=df1 %>%
  mutate(payment_types= ifelse(payment_types=="Cash", "Cash",
                    ifelse(payment_types=="Credit Card","Card","Other")))%>%
  mutate(payment_types= as.factor(payment_types))
df1$companies = gsub("[[:punct:]]","",df1$companies)
df1$companies=  gsub('[0-9]+', '', df$companies)


# Grouping less frequent companies as ‘others’
df1= df1 %>%
  mutate(companies= ifelse(companies=='Taxi Affiliation Services',"Taxi Affiliation Services",
                   ifelse(companies=="Northwest Management LLC","Northwest Management LLC",
                      ifelse(companies=="KOAM Taxi Association","KOAM Taxi Association",
                         ifelse(companies== "Dispatch Taxi Affiliation","Dispatch Taxi Affiliation",
                            ifelse(companies== "Choice Taxi Association", "Choice Taxi Association","other"))))))
```

```
df1$companies[is.na(df1$companies)]= "other"
df1$companies= as.factor(df1$companies)

df1=df1 %>%
  arrange(time_identifier_day)
```

```
# Feature engineering on timestamp
df1$year= year(df1$trip_start_timestamp)
df1$month= month(df1$trip_start_timestamp)
df1$day=  day(df1$trip_start_timestamp)
df1$hourfloat= (hour(df1$trip_start_timestamp) + minute(df1$trip_start_timestamp)/60)/24
df1$hourfloat_sin= sin(2*pi*df1$hourfloat)
df1$hourfloat_cos= cos(2*pi*df1$hourfloat)
df1$day_cat= weekdays(df1$trip_start_timestamp)
df1$day_num= (as.POSIXlt(df1$trip_start_timestamp)$wday + df1$hourfloat)/7
df1$day_sin= sin(2*pi*df1$day_num)
df1$day_cos= cos(2*pi*df1$day_num)
df1$year= as.factor(df1$year)
df1$month= as.factor(df1$month)
df1$day= as.factor(df1$day)
df1$day_cat= as.factor(df1$day_cat)
df1$yearfloat = (yday(df1$time_identifier_day)-1)/365
df1$year_cos= cos(2*pi*df1$yearfloat)
df1$year_sin= sin(2*pi*df1$yearfloat)
df1$month= as.numeric(df1$month)
df1$monthfloat= (df1$month-1)/30
df1$month_cos= cos(2*pi*df1$monthfloat)
df1$month_sin= sin(2*pi*df1$monthfloat)

# Loading weather data
weather= fread("chicago_weather_data.csv")
weather= weather %>%
  select(-STATION, -STATION_NAME)
colnames(weather)[1]='time_identifier_day'
weather$time_identifier_day= anydate(weather$time_identifier_day)

df1$time_identifier_day= anydate(df1$time_identifier_day)

# Joining trip and weather data
df2= left_join(df1, weather, by= "time_identifier_day")
df2$unique= format(df2$trip_start_timestamp, "%Y-%m-%d %H")
df2= df2[!duplicated(df2$unique),]

# Adding columns for special days which might affect trip fare
df2$Special_day=""
```

```
df2$Special_day= ifelse(df2$time_identifier_day=='2013-03-17'|
                df2$time_identifier_day=='2014-03-17'|
                df2$time_identifier_day=='2015-03-17'|
                df2$time_identifier_day=='2016-03-17'|
                df2$time_identifier_day=='2013-12-25'|
                df2$time_identifier_day=='2014-12-25'|
                df2$time_identifier_day=='2015-12-25'|
                df2$time_identifier_day=='2016-12-25'|
                df2$time_identifier_day=='2013-09-02'|
                df2$time_identifier_day=='2014-09-01'|
                df2$time_identifier_day=='2015-09-07'|
                df2$time_identifier_day=='2016-09-06'|
                df2$time_identifier_day=='2013-05-27'|
                df2$time_identifier_day=='2014-05-26'|
                df2$time_identifier_day=='2015-05-25'|
                df2$time_identifier_day=='2016-05-30'|
                df2$time_identifier_day=='2013-11-28'|
                df2$time_identifier_day=='2014-11-27'|
                df2$time_identifier_day=='2015-11-26'|
                df2$time_identifier_day=='2016-11-24',1,0)
```

# Writing csv for Prediction modeling

```
write.csv(df2, "x_train_hourly.csv", row.name=FALSE)
```

# Test data

```
# Same above operations were performed on the test data except the weather data and special
# occasion days are different.
weather= fread("weather_2017_data.csv")
df$Special_day= ifelse(df$time_identifier_day=='2017-05-29'|
                   df$time_identifier_day=='2017-03-17',
                  1,0)
```

# Python Code for Machine Learning

```
import pandas as pd
import numpy as np
import os
import time
import math
from datetime import date
import math
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
from sklearn import preprocessing
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import cross_validate
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import cross_val_score
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.metrics import median_absolute_error
from sklearn.metrics import accuracy_score, precision_score, recall_score
from sklearn.metrics import f1_score, confusion_matrix, classification_report
from sklearn.metrics import roc_curve, auc
from sklearn.grid_search import GridSearchCV
from sklearn.ensemble import RandomForestRegressor
import lightgbm as lgb
from Class_tree import DecisionTree
# Loading Data
df = pd.read_csv("x_train_daily.csv")
df_test= pd.read_csv("x_test_daily.csv")
df= df[['trip_seconds', 'trip_miles', 'trip_total', 'companies', 'pickup_latitude', 'pickup_longitude',
'dropoff_latitude', 'dropoff_longitude', 'hourfloat', 'hourfloat_sin', 'hourfloat_cos','payment_types', 'day_num',
'day_sin','day_cos','yearfloat', 'year_cos','year_sin', 'monthfloat',  'month_cos', 'month_sin', 'PRCP', 'SNWD',
'SNOW', 'TMAX', 'AWND', 'day_cat', 'Special_day']]
df_test= df_test[['trip_seconds', 'trip_miles', 'trip_total', 'companies','pickup_latitude', 'pickup_longitude',
'dropoff_latitude', 'payment_types', 'dropoff_longitude', 'hourfloat','hourfloat_sin', 'hourfloat_cos','day_num',
'day_sin','day_cos', 'yearfloat', 'year_cos','year_sin', 'monthfloat', 'month_cos', 'month_sin', 'PRCP', 'SNWD',
'SNOW', 'TMAX', 'AWND', 'day_cat', 'Special_day']]
# One Hot Encoding of nominal variables
df_nominal= df[['companies', 'day_cat', 'Special_day', 'payment_types']]
df_encoded=pd.get_dummies(df_nominal,columns=['companies', 'day_cat', 'Special_day',
'payment_types'],drop_first=False)
df_nominal_test= df_test[['companies','day_cat' , 'Special_day', 'payment_types']]
df_encoded_test=pd.get_dummies(df_nominal_test,columns=['companies', 'day_cat', 'Special_day',
'payment_types'],drop_first=False)
# Inmputing missing values with mean
df_interval=df.drop(['companies', 'day_cat', 'Special_day', 'payment_types'] ,axis=1).copy()
df_interval= df_interval.apply(lambda x: x.fillna(x.mean()),axis=0)
df_interval_test=df_test.drop(['companies', 'day_cat', 'Special_day', 'payment_types'] ,axis=1).copy()
df_interval_test= df_interval_test.apply(lambda x: x.fillna(x.mean()),axis=0)
df_final=pd.concat([df_interval,df_encoded],ignore_index=False,axis=1)
df_final_test=pd.concat([df_interval_test,df_encoded_test],ignore_index=False,axis=1)
# Linear Regression
predictor_space=df_final.drop('trip_total',axis=1)
lr=LinearRegression()
X=np.asarray(df_final.drop('trip_total',axis=1))
y= np.asarray(df_final['trip_total'])
Xt=np.asarray(df_final_test.drop('trip_total',axis=1))
yt= np.asarray(df_final_test['trip_total'])
lr.fit(X,y)
def display_metrics(lr, X, y):
    predictions = lr.predict(X)
    print("\nModel Metrics")
    print("{:.<23s}{:15d}".format('Observations', X.shape[0]))
    print("{:.<23s}{:15d}".format('Coefficients', X.shape[1]+1))
    print("{:.<23s}{:15d}".format('DF Error', X.shape[0]-X.shape[1]-1))
    R2 = r2_score(y, predictions)
    print("{:.<23s}{:15.4f}".format('R-Squared', R2))
    print("{:.<23s}{:15.4f}".format('Avg Squared Error', \
              mean_squared_error(y,predictions)))
    print("{:.<23s}{:15.4f}".format('Square Root ASE', \
              math.sqrt(mean_squared_error(y,predictions))))

def display_split_metrics(lr, Xt, yt, Xv, yv):
    predict_t = lr.predict(Xt)
    predict_v = lr.predict(Xv)
    print("\n")
    print("{:.<23s}{:>15s}{:>15s}".format('Model Metrics', \
                      'Training', 'Validation'))
    print("{:.<23s}{:15d}{:15d}".format('Observations', \
                        Xt.shape[0], Xv.shape[0]))
    print("{:.<23s}{:15d}{:15d}".format('Coefficients', \
                        Xt.shape[1]+1, Xv.shape[1]+1))
    print("{:.<23s}{:15d}{:15d}".format('DF Error', \
              Xt.shape[0]-Xt.shape[1]-1, Xv.shape[0]-Xv.shape[1]-1))
    R2t = r2_score(yt, predict_t)
    R2v = r2_score(yv, predict_v)
    print("{:.<23s}{:15.4f}{:15.4f}".format('R-Squared', R2t, R2v))
    print("{:.<23s}{:15.4f}{:15.4f}".format('RMSE', \
              math.sqrt(mean_squared_error(yt,predict_t)), \
              math.sqrt(mean_squared_error(yv,predict_v))))

display_split_metrics(lr, X, y, Xt, yt)
```

# Random Forest with Cross-Validation

```
estimator = RandomForestRegressor(n_estimators=20, n_jobs=-1)
# Funtion for cross-validation over a grid of parameters
def cv_optimize(clf, parameters, X, y, n_jobs=1, n_folds=5, score_func=None, \
            verbose=0):
  if score_func:
    gs = GridSearchCV(clf, param_grid=parameters, cv=n_folds, \
                n_jobs=n_jobs, scoring=score_func, verbose=verbose)
  else:
    gs = GridSearchCV(clf, param_grid=parameters, n_jobs=n_jobs,\
                cv=n_folds, verbose=verbose)
  gs.fit(X, y)
  print ("BEST", gs.best_params_, gs.best_score_, gs.grid_scores_,\
      gs.scorer_)
  print ("Best score: ", gs.best_score_)
  best = gs.best_estimator_
  return best
# Define a grid of parameters over which to optimize the random forest
# We will figure out which number of trees is optimal
parameters = {"n_estimators": [10,20,30],
          "max_features":  ["auto","sqrt","log2"]
          "max_depth": [50]}
best = cv_optimize(estimator, parameters, X, y, n_folds=5,\
              score_func='mean_squared_error', verbose=3)
```

```
# Fit the best Random Forest and calculate R^2 values for training and test sets
reg=best.fit(X, y)
training_accuracy = reg.score(X, y)
test_accuracy = reg.score(Xt, yt)
print ("############# based on standard predict ################")
print ("R^2 on training data: %0.4f" % (training_accuracy))
print ("R^2 on test data:     %0.4f" % (test_accuracy))
# Calculate the Root Mean Squared Error
rmse = np.sqrt(mean_squared_error(reg.predict(Xt),yt))
print ("RMSE = %0.3f (this is in log-space!)" % rmse)
print ("So two thirds of the records would be a factor of less than %0.2f \
             away from the real value." % np.power(10,rmse))
import operator
dict_feat_imp = dict(zip(list(predictor_space.columns.values),reg.feature_importances_))
sorted_features = sorted(dict_feat_imp.items(), key=operator.itemgetter(1), reverse=True)
sorted_features
```

# LightGBM with Cross-Validation

```
estimator = LGBMRegressor(num_boost_round=1000, n_jobs=-1)
# Funtion for cross-validation over a grid of parameters
def cv_optimize(clf, parameters, X, y, n_jobs=1, n_folds=10, score_func=None, \
            verbose=0):
   if score_func:
      gs = GridSearchCV(clf, param_grid=parameters, cv=n_folds, \
                n_jobs=n_jobs, scoring=score_func, verbose=verbose)
   else:
      gs = GridSearchCV(clf, param_grid=parameters, n_jobs=n_jobs,\
                cv=n_folds, verbose=verbose)
   gs.fit(X, y)
   print ("BEST", gs.best_params_, gs.best_score_, gs.grid_scores_,\
         gs.scorer_)
   print ("Best score: ", gs.best_score_)
   best = gs.best_estimator_
   return best

# Define a grid of parameters over which to optimize the random forest
# We will figure out which number of trees is optimal
parameters = {'boosting_type': ['gbdt'],
         'objective': ['regression'],
         'metric': ['auc'],
         'num_leaves': [20,40,60,80],
         'learning_rate': [0.01,0.1],
         'bagging_fraction': [0.85],
         'bagging_freq': [20],
         'verbose': [1],
         'max_bin':[200]}
best = cv_optimize(estimator, parameters, X, y, n_folds=5,\
               score_func='mean_squared_error', verbose=3)
# Fit the best Random Forest and calculate R^2 values for training and test sets
reg=best.fit(X, y)
training_accuracy = reg.score(X, y)
```

```
test_accuracy = reg.score(Xt, yt)
print ("############# based on standard predict ################")
print ("R^2 on training data: %0.4f" % (training_accuracy))
print ("R^2 on test data:     %0.4f" % (test_accuracy))
# Calculate the Root Mean Squared Error
rmse = np.sqrt(mean_squared_error(reg.predict(Xt),yt))
print ("RMSE = %0.3f (this is in log-space!)" % rmse)
print ("So two thirds of the records would be a factor of less than %0.2f \
            away from the real value." % np.power(10,rmse))
import operator
dict_feat_imp = dict(zip(list(predictor_space.columns.values),reg.feature_importances_))
sorted_features = sorted(dict_feat_imp.items(), key=operator.itemgetter(1), reverse=True)
sorted_features
```