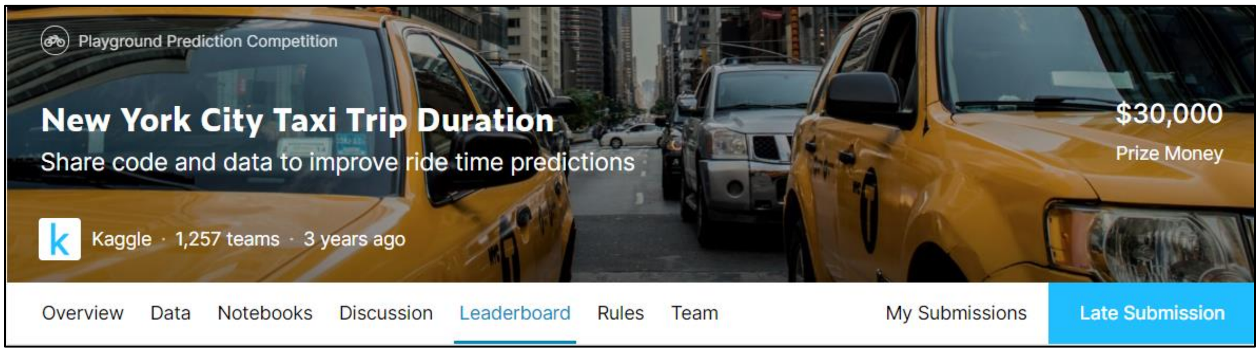
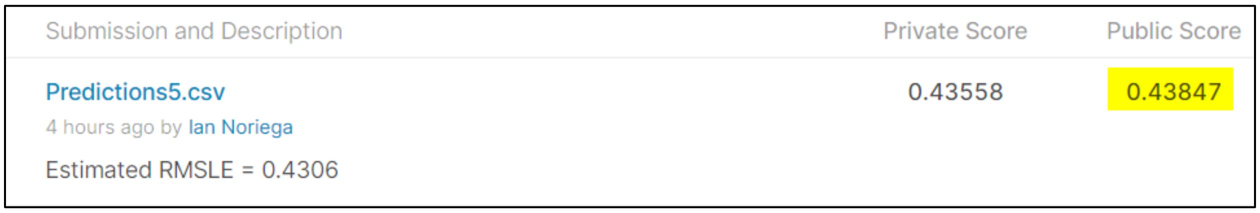
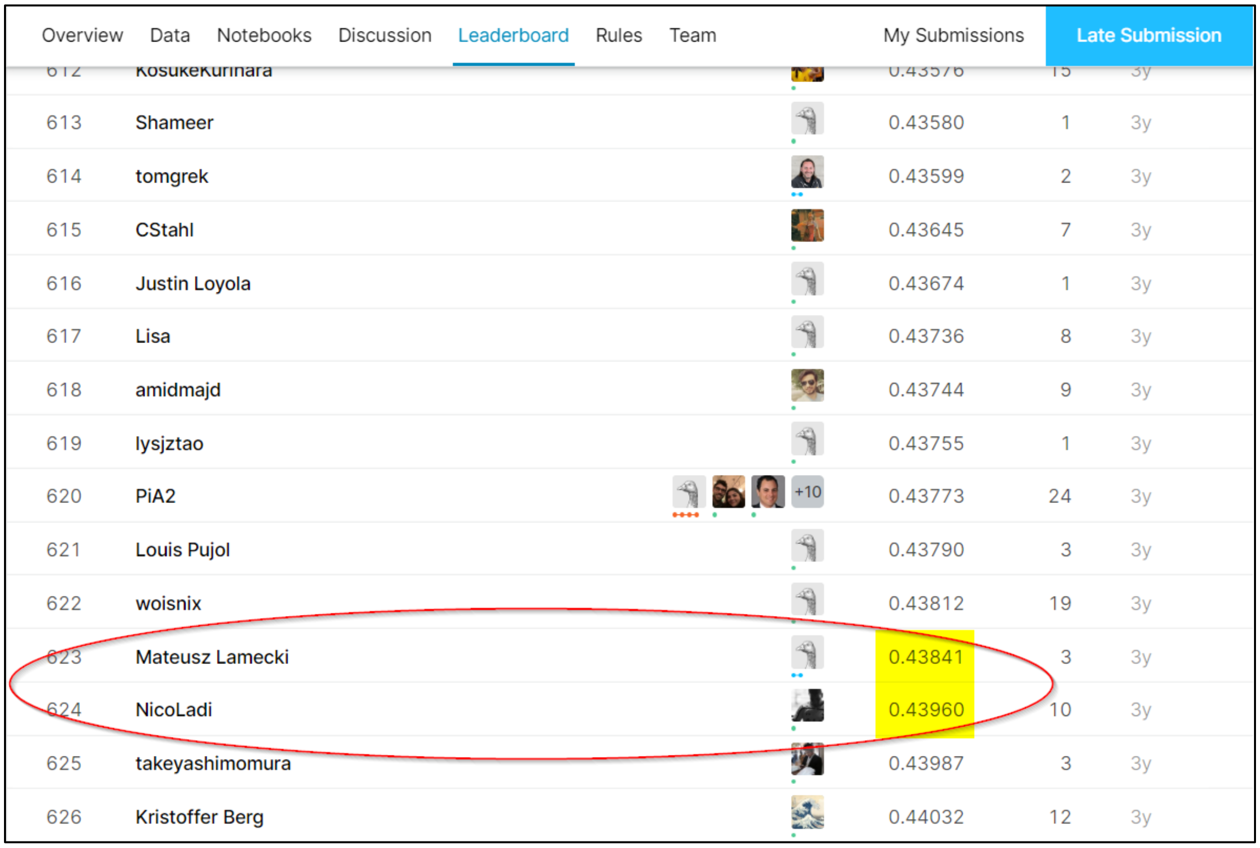
**MMA 867 – Predictive Modeling** **Ian Noriega**
**Assignment 1** **2020-05-03**

Kaggle Name: Ian Noriega
Competition: New York City Taxi Trip Duration
(https://www.kaggle.com/c/nyc-taxi-trip-duration/overview)
Total Number of Teams on Leaderboard: 1257
Position on Leaderboard: 623


Playground Prediction Competition
New York City Taxi Trip Duration
Share code and data to improve ride time predictions
$30,000
Prize Money
Kaggle · 1,257 teams · 3 years ago
Overview
Data
Notebooks
Discussion
Leaderboard
Rules
Team
My Submissions
Late Submission



| Submission and Description | Private Score | Public Score |
|---|---|---|
| Predictions5.csv<br>4 hours ago by Ian Noriega<br>Estimated RMSLE = 0.4306 | 0.43558 | 0.43847 |



| | | | | | |
|---|---|---|---|---|---|
| 612 | KosukeKurihara | 0.43576 | 15 | 3y |
| 613 | Shameer | 0.43580 | 1 | 3y |
| 614 | tomgrek | 0.43599 | 2 | 3y |
| 615 | CStahl | 0.43645 | 7 | 3y |
| 616 | Justin Loyola | 0.43674 | 1 | 3y |
| 617 | Lisa | 0.43736 | 8 | 3y |
| 618 | amidmajd | 0.43744 | 9 | 3y |
| 619 | lysjztao | 0.43755 | 1 | 3y |
| 620 | PiA2 | 0.43773 | 24 | 3y |
| 621 | Louis Pujol | 0.43790 | 3 | 3y |
| 622 | woisnix | 0.43812 | 19 | 3y |
| 623 | Mateusz Lamecki | 0.43841 | 3 | 3y |
| 624 | NicoLadi | 0.43960 | 10 | 3y |
| 625 | takeyashimomura | 0.43987 | 3 | 3y |
| 626 | Kristoffer Berg | 0.44032 | 12 | 3y |

## Contents

## Introduction

Three Kaggle competitions were determined as a potential choice:

1) Competitive Data Science: Predict Future Sales
2) New York City Taxi Trip Duration
3) House Prices: Advanced Regression Techniques

Ultimately, I decided to attempt the 'New York City Taxi Trip Duration' competition. From a brief review of this competition's overview and data, apart from being an interesting topic, the competition looked like it would require a significant focus on feature engineering, mainly revolving around geospatial data. With the goal being to predict taxi ride duration in one of the busiest cities in the world, this case ultimately appeared to be a fun and challenging (classified as "Playground" level in Kaggle) opportunity to test my regression model-building skills, and to build upon some of the techniques touched on in the first lecture of MMA 867.

## Data

For this competition, Kaggle has provided us with NYC Taxi data for the first 6 months of 2016. This dataset includes the following variables:

| Variable | Description |
|---|---|
| id | A unique identifier for each trip |
| vendor_id | A code indicating the provider associated with the trip record |
| pickup_datetime | Date and time when the meter was engaged |
| dropoff_datetime | Date and time when the meter was disengaged |
| passenger_count | The number of passengers in the vehicle |
| pickup_longitude | The longitude where the meter was engaged |
| pickup_latitude | The latitude where the meter was engaged |
| dropoff_longitude | The longitude where the meter was disengaged |
| dropoff_latitude | The latitude where the meter was disengaged |
| store_and_fwd_flag | Indicates if the trip record was held in vehicle memory before sending to the vendor due to vehicle not having a connection to the server |
| trip_duration | Duration of the trip in seconds |

Additionally, there were a few external data sources that were incorporated throughout the model building process. The significance and usage of the variables in these external datasets will be further explained in the *Feature Engineering* portion of the report.

## Exploration

To begin, from doing a quick look at some summary statistics for our data, the first thing that we notice is that we have no missing data! This is fortunate as we will not have to worry about imputing any values.

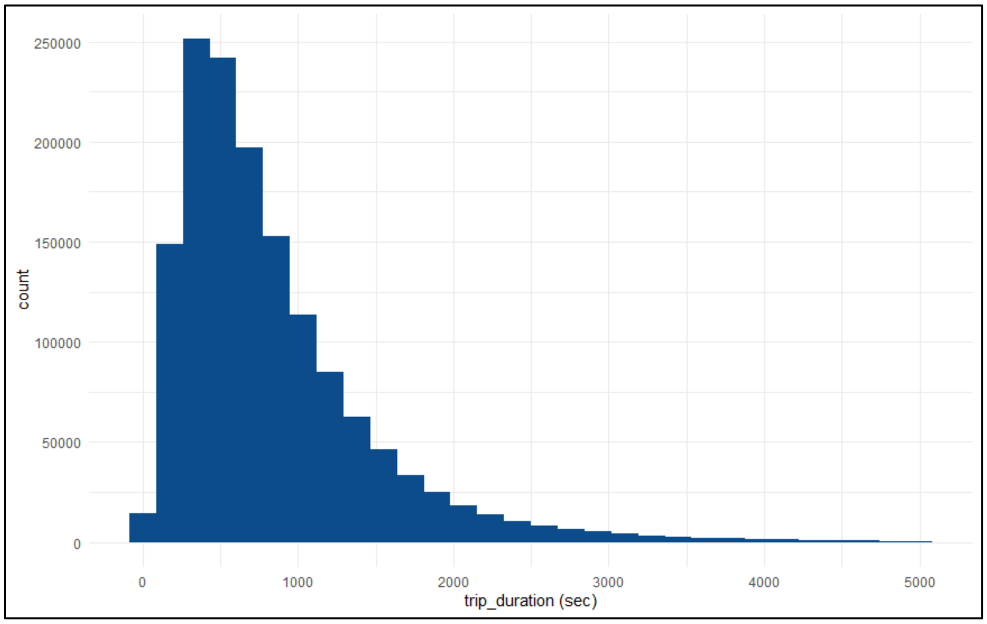


*Figure 1: Histogram of Trip Duration*

In Figure 1 above, we observe that the distribution of trip duration is significantly right skewed. We notice that most trips last in the range of approximately 400-1000 seconds (6.5-16.5 minutes), with a median value of 662 seconds or approximately 11 minutes. We do observe that the distribution is heavily skewed with some exceedingly large duration times (some of which were deemed improbable and removed from the data). To ensure our model captures this behaviour, we'll be sure to apply a log transform to our *trip_duration* variable.

Next, let's examine how trip duration behaves against some of the other variables provided to us in the dataset, namely *passenger_count*, *vendor_id*, and *store_and_fwd_flag*.

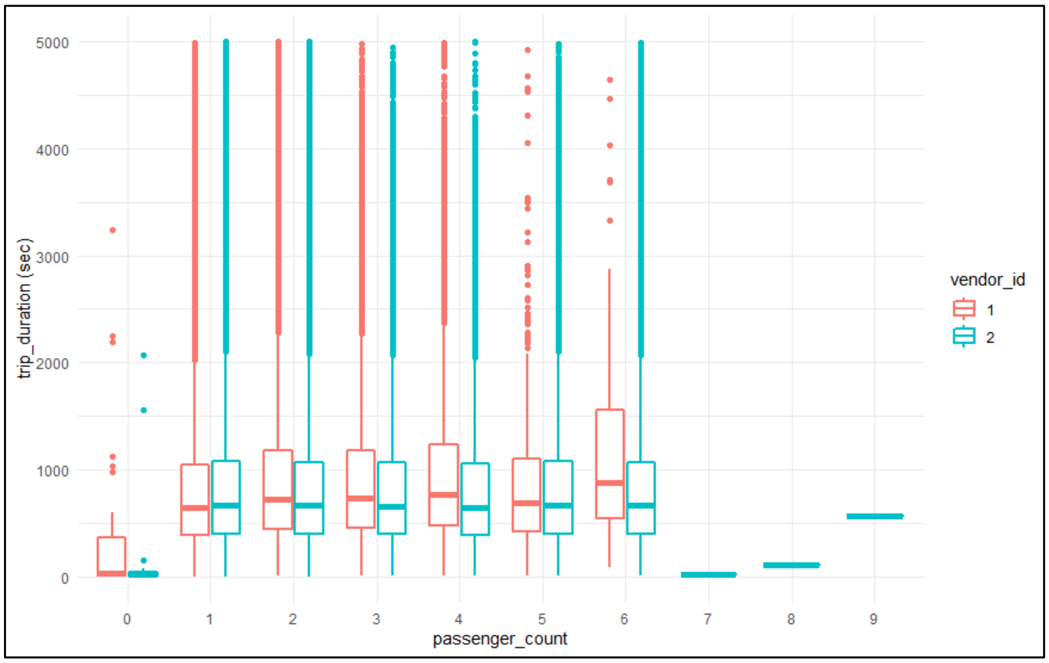


*Figure 2: Trip Duration by Passenger Count x Vendor ID*

Figure 2 displays trip duration for different numbers of passengers, along with an indicator for vendor within each of the passenger buckets. It is worth noting that 99.99% of observations are encompassed within 1-6 passengers. Apart from the few odd observations of 0, 7, 8, and 9 passengers, the number of riders in the taxi does not appear to associate with a noteworthy difference in trip duration. As for *vendor_id*, it appears as though vendor 1 could be associated with a marginally higher trip duration than vendor 2. Interestingly enough (or perhaps just by fluke), the 5 odd observations of 7-9 passengers all came from vendor 2.

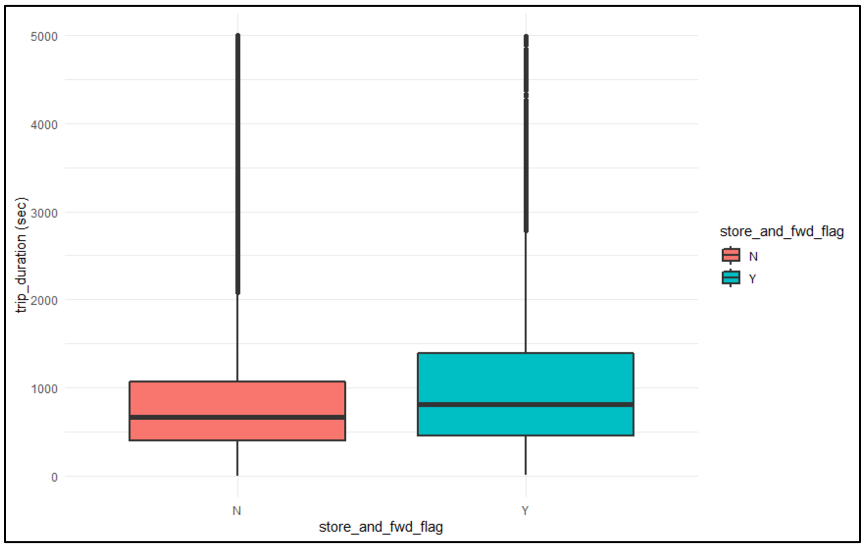


*Figure 3: Trip Duration by Store & Forward Flag*

Figure 3 displays trip duration by *store_and_fwd_flag*. From this visual it appears as though the 'Y' case (record was held and sent later due to loss of server connectivity) is associated with longer trip durations. A potential hypothesis is that these cases (only 8,045 out of nearly 1.5M records) involved travel over more rural areas where instances of network loss become more of a possibility.

Finally, before incorporating any feature engineering and modelling attempts, we split our initial dataset into training and testing sets (based off of a 70:30 split).

## Feature Engineering

### Temporal Data

Using the *pickup_datetime* variable from the primary dataset, we extract the month, day of the week, and hour for each of our recorded trips.

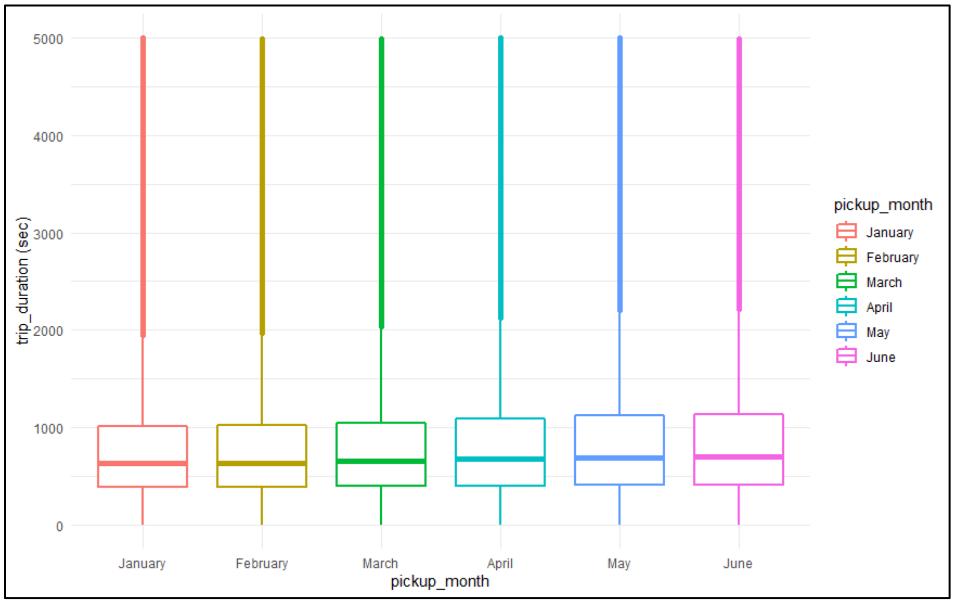


*Figure 4: Trip Duration by Month*

In the plot displayed above in Figure 4, we observe that there looks to be no reportable difference in terms of the recorded trip duration by month.

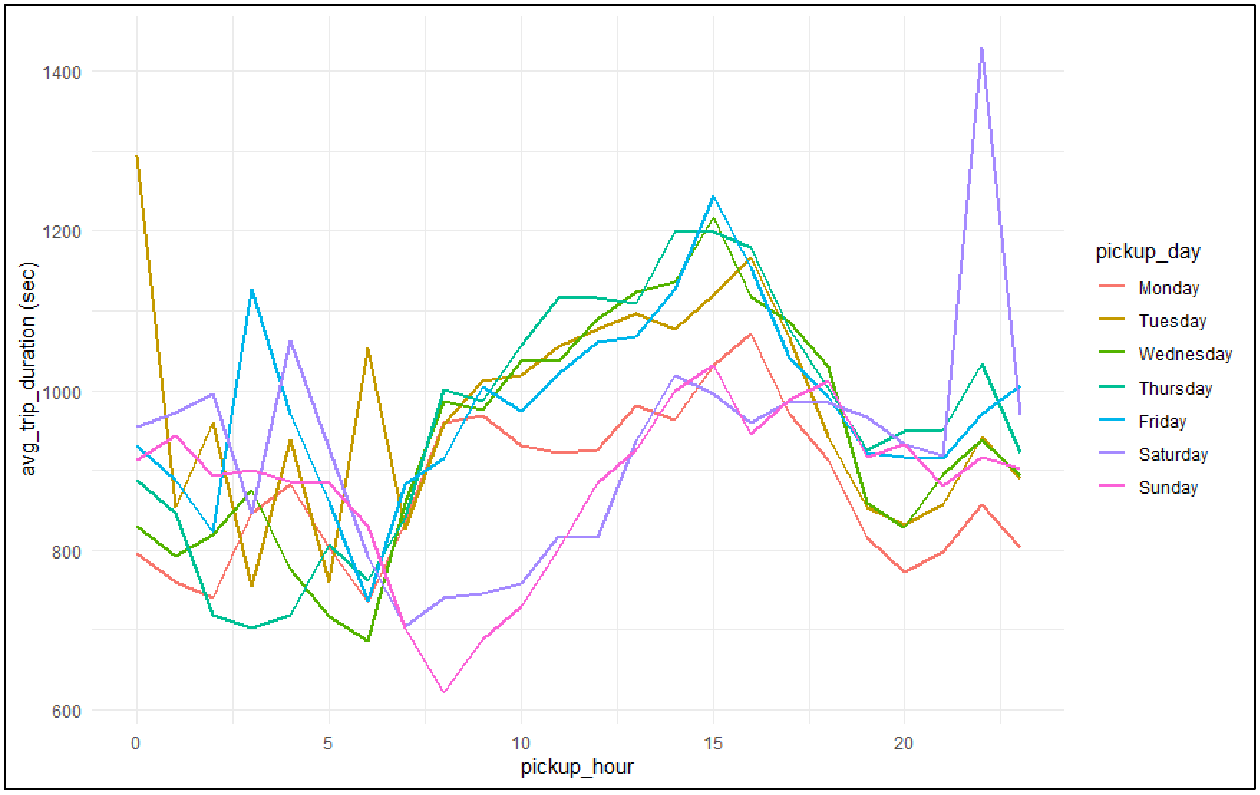


*Figure 5: Trip Duration by Hour x Day of Week*

Next, we examine day and hour in Figure 5, and make a few interesting observations. First, we observe that for most days the most steadily-high trip durations are occurring between the hours of about 7AM – 6PM. This is especially apparent for weekdays (coinciding with typical working hours), and we can even see weekdays spike around 3PM – 5PM, roughly coinciding with "rush hour". We also observe some relatively abnormal spikes in the early hours of the weekdays, likely due to workers who may have a lengthier commute to account for. Alternatively, the weekends look to be tied to shorter trip durations in comparison, especially Sundays. We do, however, notice the highest peak of our graph occurs on Saturday night, along with a rather unexplainable (and suspect) spike at 12AM on Tuesdays. Given the variability observed, day and hour should certainly be factored into our predictive model.

## Distance

Using the *geosphere* package in R, we engineer a new variable calculating the direct geographical distance between two points, by leveraging the pick-up and drop-off coordinates provided in the dataset. In doing so, we also identify a few extreme trip distance values (some of which were unreasonably large and removed from the data).

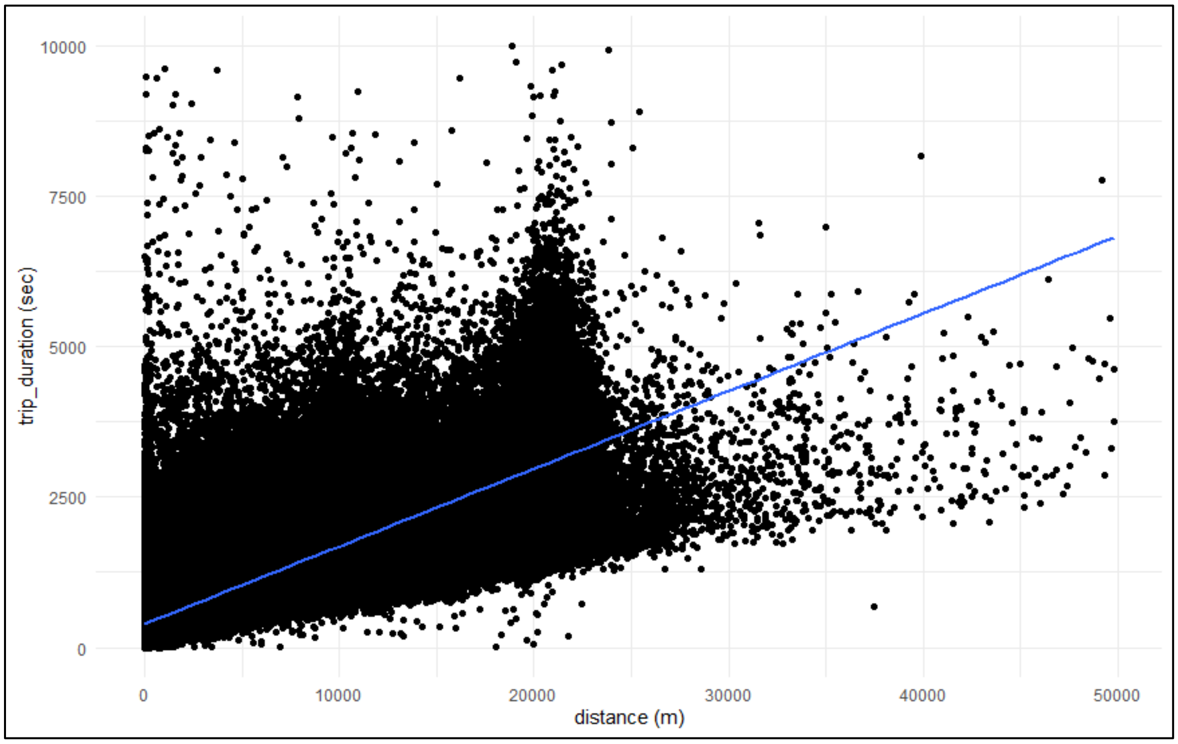


Figure 6: Trip Duration and Direct Geographical Distance

As seen in Figure 6 above, there is an evident positive relationship between direct distance and trip duration. As distance tends to increase, so too does the trip duration. We'll definitely want to include this feature in our model and may want to make use of another log transform for some of the extreme values observed.

***Model Testing***: With the addition of our temporal and distance features, we take a first stab at building our predictive model. A solid improvement is observed when moving from the simple linear model in *mod1* to the log-log model in *mod7*, however, unfortunately this is only good enough to be in the 33rd percentile of submissions. These models can be seen in more detail in Table 1 of the Appendix.

## Popular Destinations

Let's attempt to identify some popular, high-traffic destinations specific to New York City, where taxis are often used. The first that comes to mind are airports. We obtain the coordinates for JFK and LaGuardia and create a binary variable for each, indicating whether the drop-off location of the taxi ride was in close vicinity of the respective airport (< 2KM).

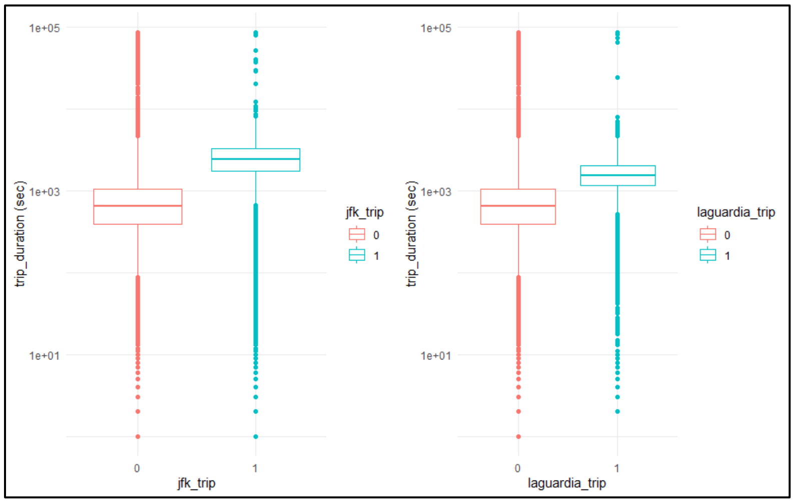


*Figure 7: Trip Duration by Airport Trip (JFK & LaGuardia)*

From the boxplots above, it is evident that trips to these major NYC airports are associated with increased taxi trip durations.

Using the same logic, we engineer a few other binary variables for 4 other high-traffic destinations in New York City – Times Square, Madison Square Garden, Yankee Stadium, and the Empire State Building.

## Busy Hours

Following our exploration of temporal data, we engineer a pair of categorical variables to try and capture a couple key hourly intervals – rush hour and working hours. Rush hour is defined as 4PM – 6PM on weekdays while working hours are defined as 8AM – 6PM on weekdays. In Figure 8 below, we observe perhaps a slight difference in these time ranges, although they do not look overly significant.

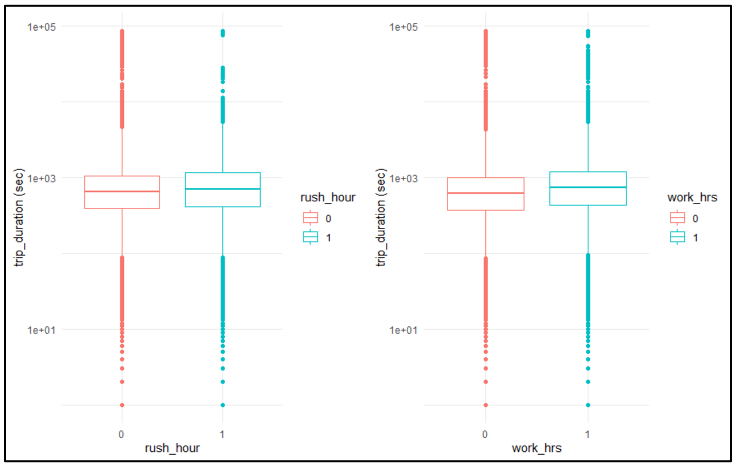


*Figure 8: Trip Duration by Rush Hour & Work Hours*

***Model Testing***: With the addition of our popular destination and busy hour features, we further build upon our log-log model and evaluate any progress made. In *mod9* we see that these additional features have provided a slight boost to prediction accuracy, but there is still work to be done.

## External OSRM Data

To further enhance predictive power, an external dataset provided on Kaggle is referenced. This dataset contains Open Source Routing Machine (OSRM) data for the taxi trips under inspection – more detail into this data source is available in the *External Data Sources* portion of the Appendix. After joining this dataset to our base dataset, we examine the relationship of some of its variables with our trip duration variable.

### a. OSRM Variables

A few variables from the OSRM dataset stand out in particular – *total_distance*, *total_travel_time*, and *number_of_steps*. It is important to note here how the distance variable from this dataset differs from our originally feature engineered *distance* variable. The *total_distance* variable here measures the distance of the trip based on an actual street route, whereas our initial *distance* variable measures the direct distance between two points geographically.

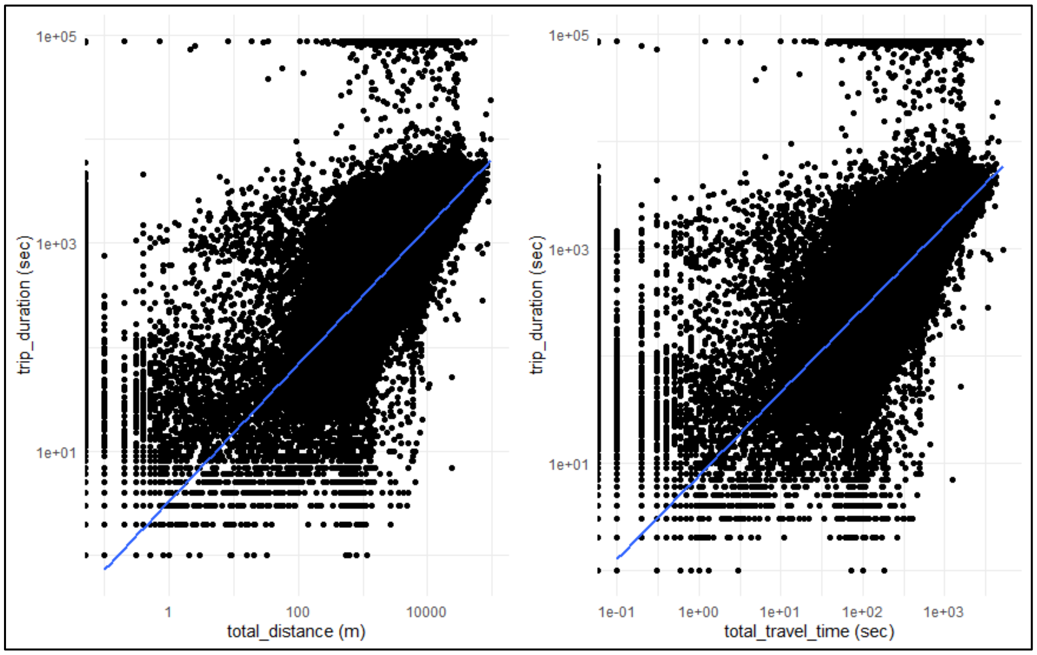


*Figure 9: Trip Duration by OSRM Distance & Travel Time*

Similar to what we observed with geographical distance, it appears that both *total_distance* and *total_travel_time* hold a strong positive relationship with trip duration, as seen above in Figure 9.

### b. Turns and Left Turns

We also use the OSRM dataset to engineer a couple new features, namely number of turns and number of left turns. Along with the total number of steps, these two features also appear to hold a positive relationship with trip duration. Logically, as we observe an increase in the number of steps/turns/lefts on a given trip, the trip duration also increases. Figure 10 below displays one of these cases, examining trip duration by the number of suggested left turns.

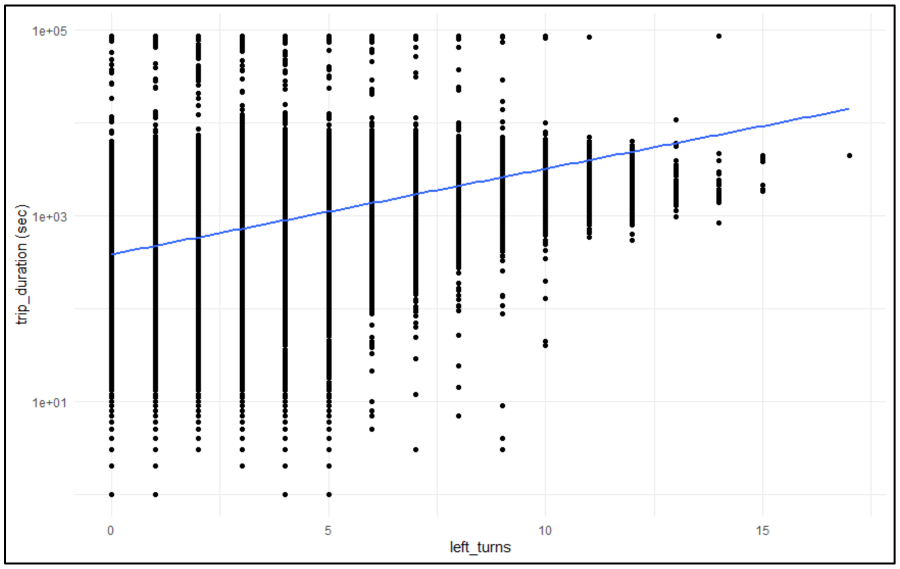


*Figure 10: Trip Duration by Number of Suggested Left Turns*

***Model Testing***: Going back to our model, we now add in our *turns* and *left_turns* features as well as *total_distance*, *total_travel_time*, and *number_of_steps* that are native to the OSRM dataset. The resulting *mod11* sees another nice boost in accuracy (refer to Table 1 in the Appendix). Note that this still has us in the 40th percentile of the competition.

## Direction of Travel

Returning to how we calculated geographical direct distance, we will use a similar function from the *geosphere* package to calculate the direction/bearing of travel.

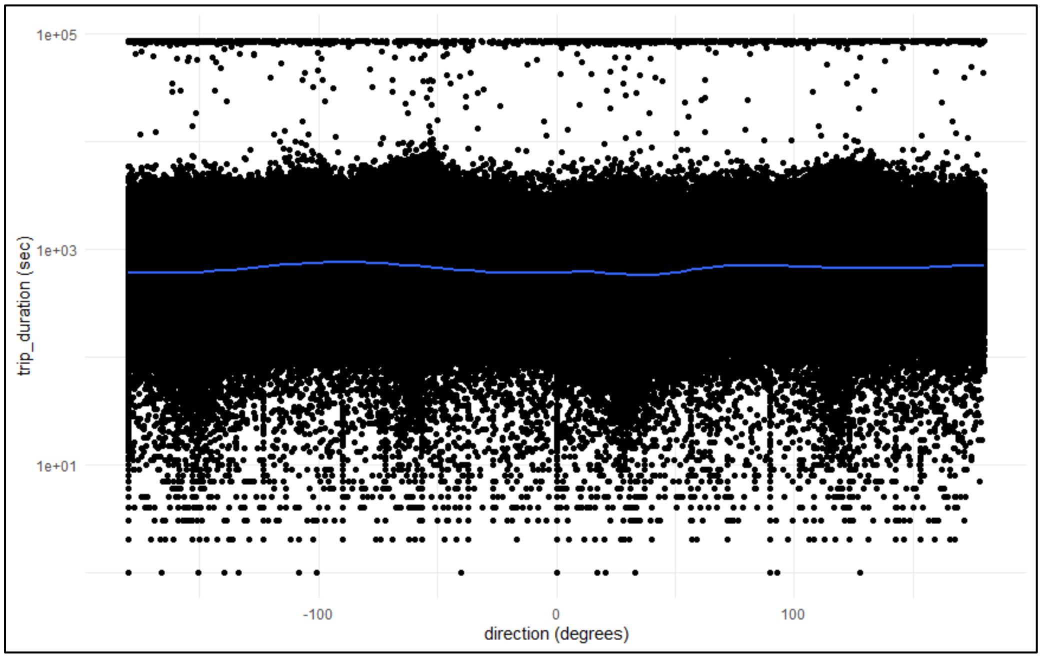


*Figure 11: Trip Duration by Directional Bearing*

Referring to Figure 11, here we observe that the overall direction of the taxi trip does not appear to hold any noteworthy pattern with trip duration.

***Model Testing***: At this point we return to our model. While we add in our direction, we also decide to experiment with some additional transformations and interactions between the wide range of variables we have added. Eventually we reach *mod16*, which features a square root transformation on the OSRM *total_distance* and *total_travel_time* variables, as well as a logarithmic transformation on *direction*. The model can be seen in more detail in Table 1 of the Appendix, and results in a significant accuracy boost, pushing us to the 43rd percentile in the competition leaderboard.

## Busiest Streets

After some quick google researching on busy streets in New York City, 4 streets stood out in particular – East 34th Street, Fifth Avenue, East 42nd Street, and Eighth Avenue. Leveraging the recently joined OSRM data, a new binary variable is engineered. By combing through the *street_for_each_step* variable for these famously busy streets, the *busy_street* variable is created and indicates if a taxi ride involved a maneuver on any one of these streets (see *Key Code Snippets* portion of the Appendix for Rcode on how this was done).

While the difference is not strikingly apparent, as seen below in Figure 12, it does appear as though these busy streets are associated with a slight uptick in trip duration.

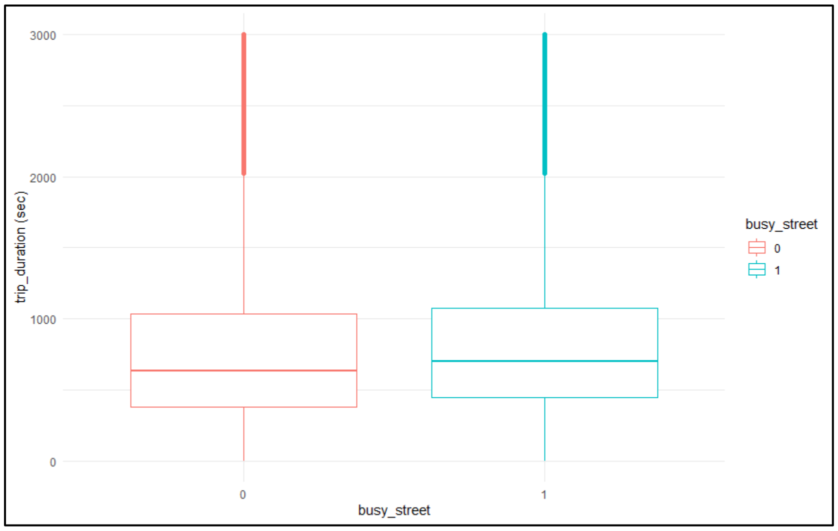


*Figure 12: Trip Duration by Busy Street*

***Model Testing***: In *mod18* (refer to Table 1 in Appendix) we see that the addition of our feature looking at busy streets shaves a little bit more of our estimated RMSLE score.

## External Weather Data

We now look at incorporating some weather data – perhaps lower temperatures or precipitation and snow fall have an impact on trip duration. More information on this weather dataset is found in the *External Data Sources* section of the Appendix.

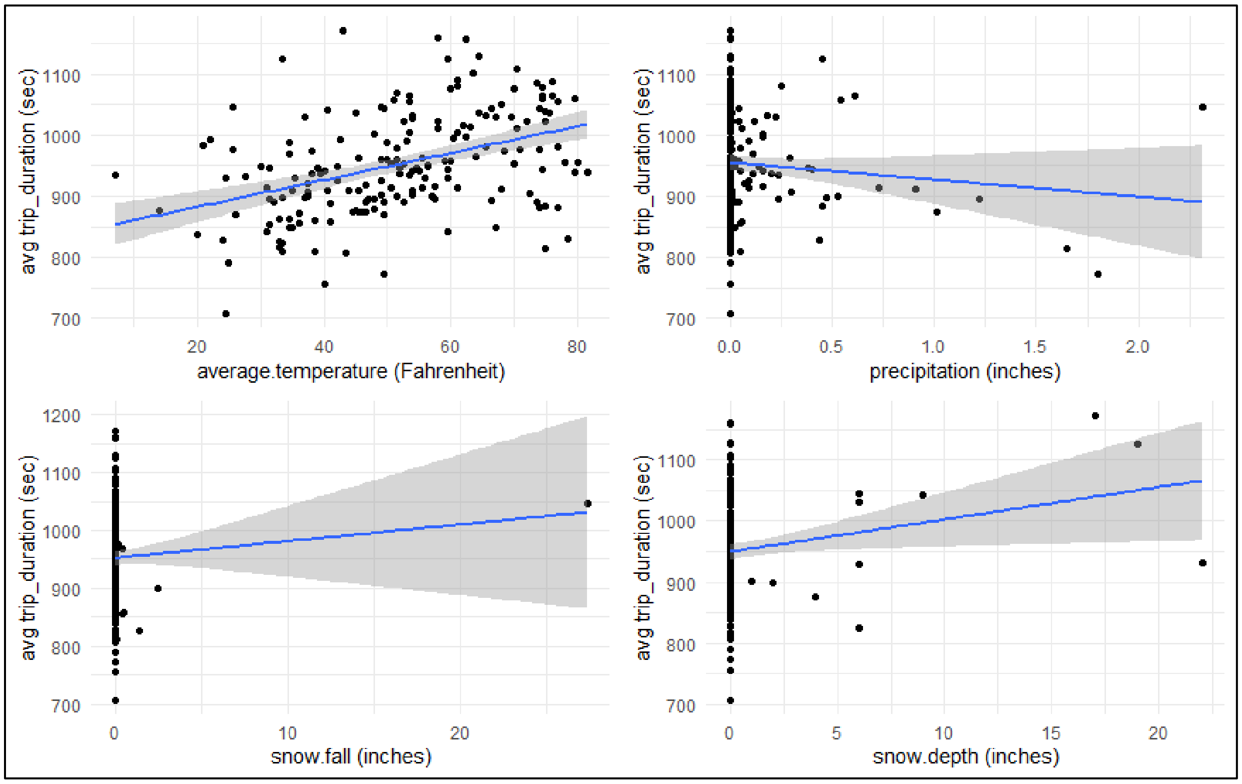


*Figure 13: Scatter plots examining Avg Daily Trip Duration by 1) Avg Temperature, 2) Precipitation, 3) Snow Fall, 4) Snow Depth*

Referring to Figure 13 above, the first observation we make is that it appears as though trip duration tends to increase as the temperature increases – this could be due to people being more inclined to be up and about on more pleasant days temperature-wise. Second, we see that precipitation does not appear to lead to increased trip duration. In fact, we observe a slight decline in trip duration as precipitation increases. When considering snow, we observe an increase in trip durations as snow fall and snow depth both increase (although these are based on a rather minimal sample). We do take notice of a few 2016 dates that experienced extreme levels of snow (especially the case for snow fall). Interestingly enough, this corresponds to a huge blizzard that New York City experienced back at the end of Jan 2016 (see NYC Blizzard Biggest Ever January 23rd 2016).

***Model Testing***: With our external weather data added, we revisit our predictive model. After adding in *average.temperature*, *precipitation*, *snow.fall*, and *snow.depth*, along with a few more variable transformations and interactions, we reach *mod20* (refer to Table 1 in Appendix). The new model observes another nice drop in RMSLE, inching us closer to beating the average.

## Final Model and Results

The final model is a multiple regression model involving all variables that I have outlined over the course of this report. The resulting model includes a few variable transformations, as well as numerous interactions implemented primarily through trial and error (using iterative evaluation against estimated RMSLE). Composition of the model (*mod22*) can be seen in more detail in Table 1 of the Appendix. Additionally, more specific details on the model in R (i.e., model summary and RMSLE estimation) can be found in *Final Model in R* section of the Appendix.

Predictive strength of the model was evaluated using Root Mean Squared Logarithmic Error (RMSLE). Our final model nets an official RMSLE score of 0.43847 from Kaggle (estimated RMSLE in R was 0.431), when evaluated against the true test set. This is good enough to push us into the top 50% of the competition.

Considering the amount of feature engineering that was done, this result seems respectable when considering that only regression analysis techniques were used, as opposed to more advanced techniques such as random forests and decision tree algorithms like XGBoost.

## Appendix

### Summary of Noteworthy Models

Note that models were evaluated by Root Mean Squared Logarithmic Error (RMSLE).

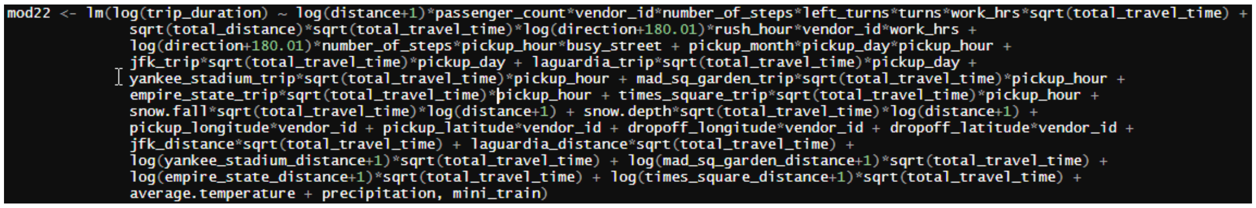
| Model | Model Composition | RMSLE* |
|---|---|---|
| *mod1* | trip_duration~distance+passenger_count+store_and_fwd_flag+pickup_month+pickup_day+pickup_hour | 0.670 |
| *mod7* | log(trip_duration)~log(distance+1)+passenger_count+pickup_month+pickup_day+pickup_hour+ pickup_longitude+pickup_latitude+vendor_id | 0.540 |
| *mod9* | log(trip_duration)~log(distance+1)+passenger_count+pickup_month+pickup_day+pickup_hour+ pickup_longitude+pickup_latitude+dropoff_longitude+dropoff_latitude+vendor_id+jfk_trip+laguardia_trip+ rush_hour+work_hrs+times_square_trip+mad_sq_garden_trip+yankee_stadium_trip+empire_state_trip | 0.519 |
| *mod11* | log(trip_duration)~log(distance+1)+passenger_count+pickup_month+pickup_day+pickup_hour+ pickup_longitude+pickup_latitude+dropoff_longitude+dropoff_latitude+vendor_id+jfk_trip+laguardia_trip+rush_hour+ work_hrs+times_square_trip+mad_sq_garden_trip+yankee_stadium_trip+empire_state_trip+ total_distance+total_travel_time+number_of_steps+left_turns+turns | 0.477 |
| *mod16* | log(trip_duration)~log(distance+1)*passenger_count*vendor_id*number_of_steps*left_turns*turns+pickup_month +pickup_day*pickup_hour+pickup_longitude+pickup_latitude+dropoff_longitude+dropoff_latitude+jfk_trip* sqrt(total_travel_time)+laguardia_trip*sqrt(total_travel_time)+sqrt(total_distance)*sqrt(total_travel_time)* log(direction+180.01)*rush_hour*vendor_id+jfk_distance+laguardia_distance+store_and_fwd_flag | 0.464 |
| *mod18* | log(trip_duration)~log(distance+1)*passenger_count*vendor_id*number_of_steps*left_turns*turns+pickup_month +pickup_day*pickup_hour+pickup_longitude+pickup_latitude+dropoff_longitude+dropoff_latitude+jfk_trip* sqrt(total_travel_time)+laguardia_trip*sqrt(total_travel_time)+sqrt(total_distance)*sqrt(total_travel_time)* log(direction+180.01)*rush_hour*vendor_id+jfk_distance+laguardia_distance+store_and_fwd_flag+ yankee_stadium_trip+mad_sq_garden_trip+empire_state_trip+times_square_trip+log(yankee_stadium_distance+1) +log(mad_sq_garden_distance+1)+log(empire_state_distance+1)+log(times_square_distance+1)+busy_street | 0.455 |
| *mod20* | log(trip_duration)~log(distance+1)*passenger_count*vendor_id*number_of_steps*left_turns*turns*work_hrs+ pickup_month*pickup_day*pickup_hour+pickup_longitude+pickup_latitude+dropoff_longitude+dropoff_latitude+ jfk_trip*sqrt(total_travel_time)*pickup_day+laguardia_trip*sqrt(total_travel_time)*pickup_day+sqrt(total_distance)* sqrt(total_travel_time)*log(direction+180.01)*rush_hour*vendor_id*work_hrs+jfk_distance+laguardia_distance+ yankee_stadium_trip*sqrt(total_travel_time)+mad_sq_garden_trip*sqrt(total_travel_time)+empire_state_trip* sqrt(total_travel_time)+times_square_trip*sqrt(total_travel_time)+log(yankee_stadium_distance+1)+ log(mad_sq_garden_distance+1)+log(empire_state_distance+1)+log(times_square_distance+1)+pickup_hour* busy_street+average.temperature+precipitation+snow.fall*sqrt(total_travel_time)*log(distance+1)+ snow.depth*sqrt(total_travel_time)*log(distance+1) | 0.434 |
| *mod22* | log(trip_duration)~log(distance+1)*passenger_count*vendor_id*number_of_steps*left_turns*turns*work_hrs* sqrt(total_travel_time)+sqrt(total_distance)*sqrt(total_travel_time)*log(direction+180.01)*rush_hour*vendor_id* work_hrs+log(direction+180.01)*number_of_steps*pickup_hour*busy_street+pickup_month*pickup_day*pickup_hour+ jfk_trip*sqrt(total_travel_time)*pickup_day+laguardia_trip*sqrt(total_travel_time)*pickup_day+yankee_stadium_trip* sqrt(total_travel_time)*pickup_hour+mad_sq_garden_trip*sqrt(total_travel_time)*pickup_hour+empire_state_trip* sqrt(total_travel_time)*pickup_hour+times_square_trip*sqrt(total_travel_time)*pickup_hour+snow.fall* sqrt(total_travel_time)*log(distance+1)+snow.depth*sqrt(total_travel_time)*log(distance+1)+pickup_longitude* vendor_id+pickup_latitude*vendor_id+dropoff_longitude*vendor_id+dropoff_latitude*vendor_id+jfk_distance* sqrt(total_travel_time)+laguardia_distance*sqrt(total_travel_time)+log(yankee_stadium_distance+1)* sqrt(total_travel_time)+log(mad_sq_garden_distance+1)*sqrt(total_travel_time)+log(empire_state_distance+1)* sqrt(total_travel_time)+log(times_square_distance+1)*sqrt(total_travel_time)+average.temperature+precipitation | 0.431 |

*Table 1: Summary of Notable Models*

**Note that RMSLE here refers to* ***estimated*** *RMSLE scores based off of train/test sets*

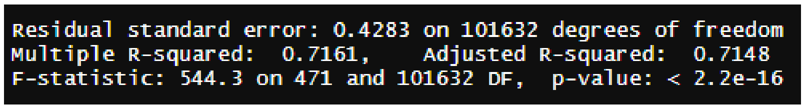
## Final Model in R

- Model Composition:



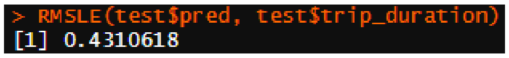
```
mod22 <- lm(log(trip_duration) ~ log(distance+1)*passenger_count*vendor_id*number_of_steps*left_turns*turns*work_hrs*sqrt(total_travel_time) +
            sqrt(total_distance)*sqrt(total_travel_time)*log(direction+180.01)*rush_hour*vendor_id*work_hrs +
            log(direction+180.01)*number_of_steps*pickup_hour*busy_street + pickup_month*pickup_day*pickup_hour +
            jfk_trip*sqrt(total_travel_time)*pickup_day + laguardia_trip*sqrt(total_travel_time)*pickup_day +
            yankee_stadium_trip*sqrt(total_travel_time)*pickup_hour + mad_sq_garden_trip*sqrt(total_travel_time)*pickup_hour +
            empire_state_trip*sqrt(total_travel_time)*pickup_hour + times_square_trip*sqrt(total_travel_time)*pickup_hour +
            snow.fall*sqrt(total_travel_time)*log(distance+1) + snow.depth*sqrt(total_travel_time)*log(distance+1) +
            pickup_longitude*vendor_id + pickup_latitude*vendor_id + dropoff_longitude*vendor_id + dropoff_latitude*vendor_id +
            jfk_distance*sqrt(total_travel_time) + laguardia_distance*sqrt(total_travel_time) +
            log(yankee_stadium_distance+1)*sqrt(total_travel_time) + log(mad_sq_garden_distance+1)*sqrt(total_travel_time) +
            log(empire_state_distance+1)*sqrt(total_travel_time) + log(times_square_distance+1)*sqrt(total_travel_time) +
            average.temperature + precipitation, mini_train)
```

- Model Summary Statistics:



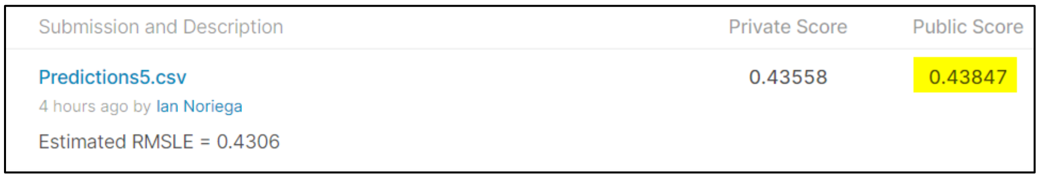
```
Residual standard error: 0.4283 on 101632 degrees of freedom
Multiple R-squared:  0.7161,	Adjusted R-squared:  0.7148 
F-statistic: 544.3 on 471 and 101632 DF,  p-value: < 2.2e-16
```

- Estimating Model's RMSLE:

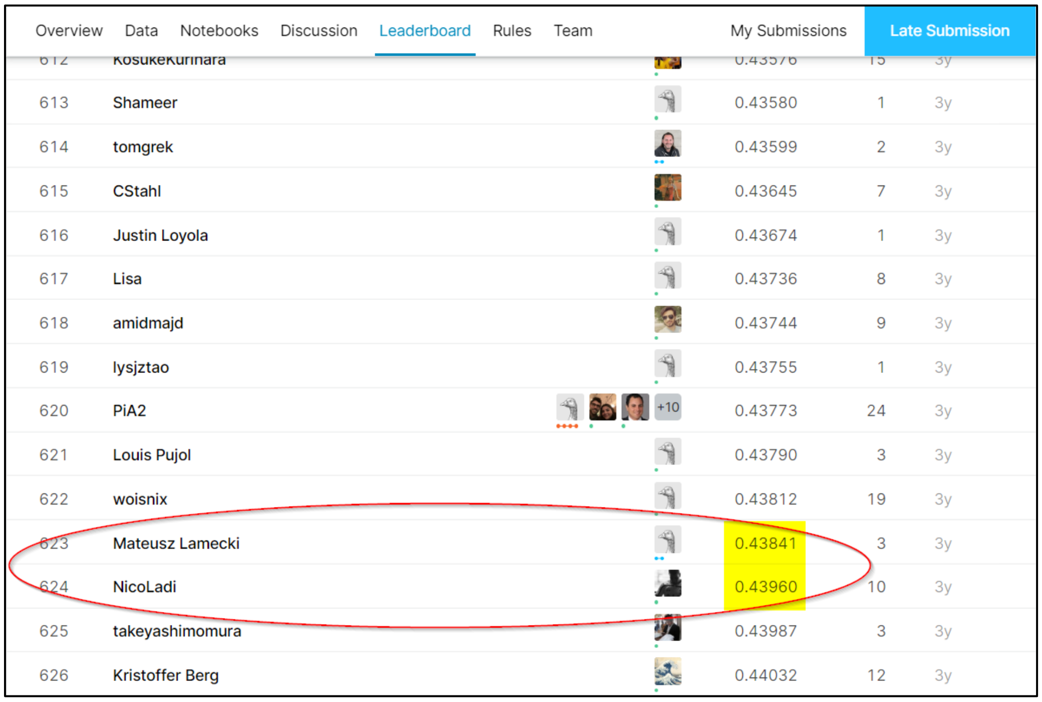


```
> RMSLE(test$pred, test$trip_duration)
[1] 0.4310618
```

As mentioned in the *Final Model and Results* section of the report, the official RMSLE score for the final model was **0.438**.



| Submission and Description | Private Score | Public Score |
|---|---|---|
| Predictions5.csv<br>4 hours ago by Ian Noriega<br>Estimated RMSLE = 0.4306 | 0.43558 | 0.43847 |



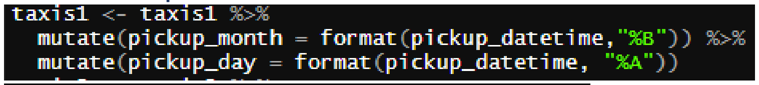
Overview Data Notebooks Discussion Leaderboard Rules Team My Submissions Late Submission

| | | | | |
|---|---|---|---|---|
| 612 | KosukeKurihara | 0.43576 | 15 | 3y |
| 613 | Shameer | 0.43580 | 1 | 3y |
| 614 | tomgrek | 0.43599 | 2 | 3y |
| 615 | CStahl | 0.43645 | 7 | 3y |
| 616 | Justin Loyola | 0.43674 | 1 | 3y |
| 617 | Lisa | 0.43736 | 8 | 3y |
| 618 | amidmajd | 0.43744 | 9 | 3y |
| 619 | lysjztao | 0.43755 | 1 | 3y |
| 620 | PiA2 | 0.43773 | 24 | 3y |
| 621 | Louis Pujol | 0.43790 | 3 | 3y |
| 622 | woisnix | 0.43812 | 19 | 3y |
| 623 | Mateusz Lamecki | 0.43841 | 3 | 3y |
| 624 | NicoLadi | 0.43960 | 10 | 3y |
| 625 | takeyashimomura | 0.43987 | 3 | 3y |
| 626 | Kristoffer Berg | 0.44032 | 12 | 3y |

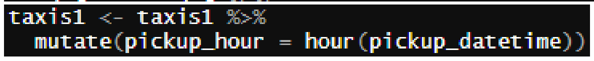
## Key Code Snippets

Full code for this assignment is available on GitHub here.

### a. Temporal Features

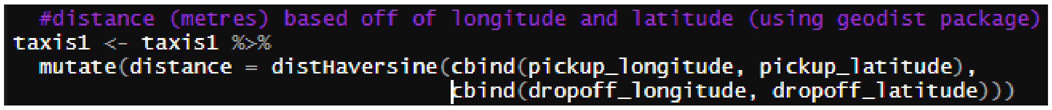


```
taxis1 <- taxis1 %>%
  mutate(pickup_month = format(pickup_datetime,"%B")) %>%
  mutate(pickup_day = format(pickup_datetime, "%A"))
```



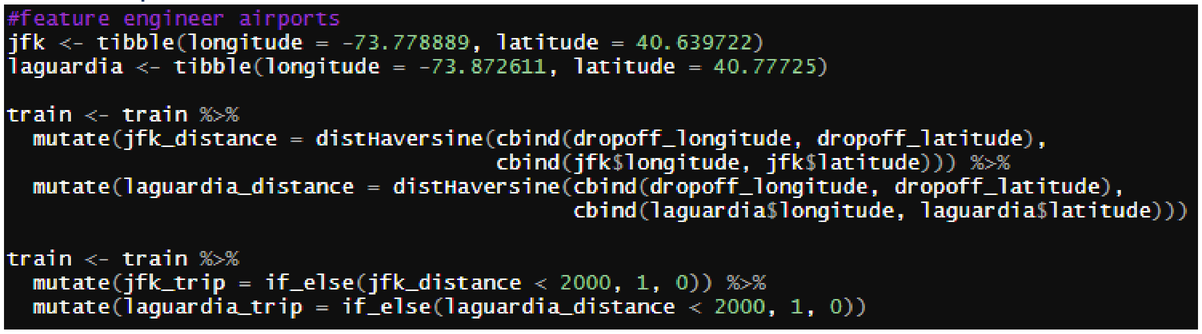
```
taxis1 <- taxis1 %>%
  mutate(pickup_hour = hour(pickup_datetime))
```

### b. Distance Feature

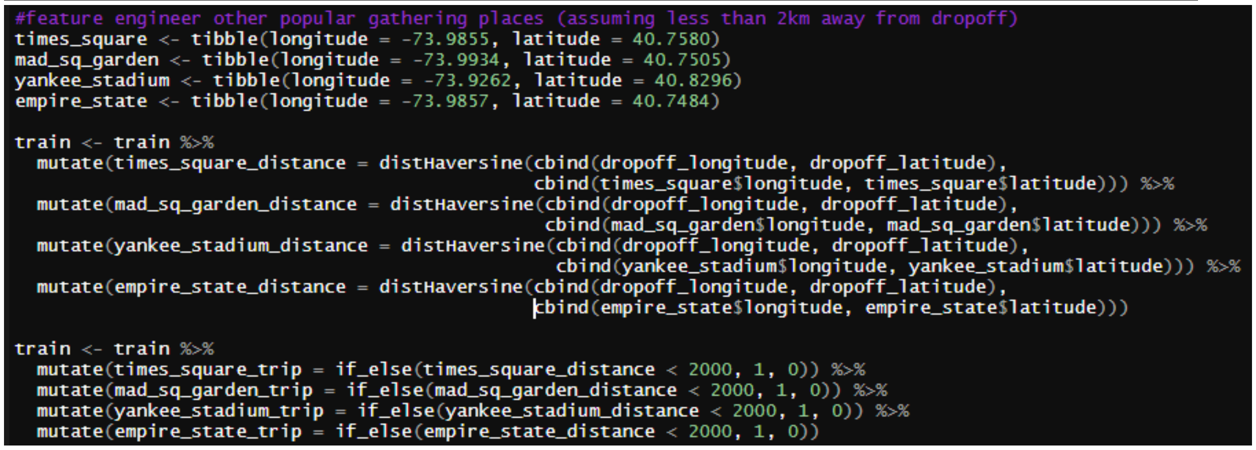


```
  #distance (metres) based off of longitude and latitude (using geodist package)
taxis1 <- taxis1 %>%
  mutate(distance = distHaversine(cbind(pickup_longitude, pickup_latitude),
                                  cbind(dropoff_longitude, dropoff_latitude)))
```

### c. Popular Destination Features

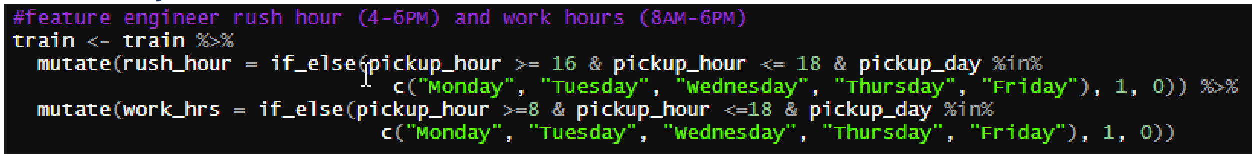


```
#feature engineer airports
jfk <- tibble(longitude = -73.778889, latitude = 40.639722)
laguardia <- tibble(longitude = -73.872611, latitude = 40.77725)

train <- train %>%
  mutate(jfk_distance = distHaversine(cbind(dropoff_longitude, dropoff_latitude),
                                      cbind(jfk$longitude, jfk$latitude))) %>%
  mutate(laguardia_distance = distHaversine(cbind(dropoff_longitude, dropoff_latitude),
                                            cbind(laguardia$longitude, laguardia$latitude)))

train <- train %>%
  mutate(jfk_trip = if_else(jfk_distance < 2000, 1, 0)) %>%
  mutate(laguardia_trip = if_else(laguardia_distance < 2000, 1, 0))
```



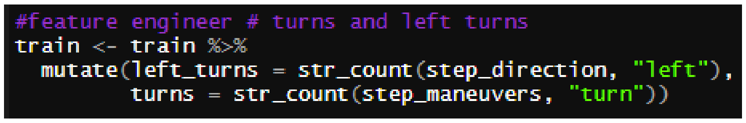
```
#feature engineer other popular gathering places (assuming less than 2km away from dropoff)
times_square <- tibble(longitude = -73.9855, latitude = 40.7580)
mad_sq_garden <- tibble(longitude = -73.9934, latitude = 40.7505)
yankee_stadium <- tibble(longitude = -73.9262, latitude = 40.8296)
empire_state <- tibble(longitude = -73.9857, latitude = 40.7484)

train <- train %>%
  mutate(times_square_distance = distHaversine(cbind(dropoff_longitude, dropoff_latitude),
                                               cbind(times_square$longitude, times_square$latitude))) %>%
  mutate(mad_sq_garden_distance = distHaversine(cbind(dropoff_longitude, dropoff_latitude),
                                                cbind(mad_sq_garden$longitude, mad_sq_garden$latitude))) %>%
  mutate(yankee_stadium_distance = distHaversine(cbind(dropoff_longitude, dropoff_latitude),
                                                 cbind(yankee_stadium$longitude, yankee_stadium$latitude))) %>%
  mutate(empire_state_distance = distHaversine(cbind(dropoff_longitude, dropoff_latitude),
                                               cbind(empire_state$longitude, empire_state$latitude)))

train <- train %>%
  mutate(times_square_trip = if_else(times_square_distance < 2000, 1, 0)) %>%
  mutate(mad_sq_garden_trip = if_else(mad_sq_garden_distance < 2000, 1, 0)) %>%
  mutate(yankee_stadium_trip = if_else(yankee_stadium_distance < 2000, 1, 0)) %>%
  mutate(empire_state_trip = if_else(empire_state_distance < 2000, 1, 0))
```

### d. Busy Hour Features



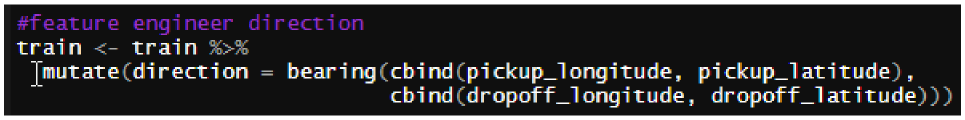
```
#feature engineer rush hour (4-6PM) and work hours (8AM-6PM)
train <- train %>%
  mutate(rush_hour = if_else(pickup_hour >= 16 & pickup_hour <= 18 & pickup_day %in%
                               c("Monday", "Tuesday", "Wednesday", "Thursday", "Friday"), 1, 0)) %>%
  mutate(work_hrs = if_else(pickup_hour >=8 & pickup_hour <=18 & pickup_day %in%
                              c("Monday", "Tuesday", "Wednesday", "Thursday", "Friday"), 1, 0))
```

### e. OSRM Turn Features



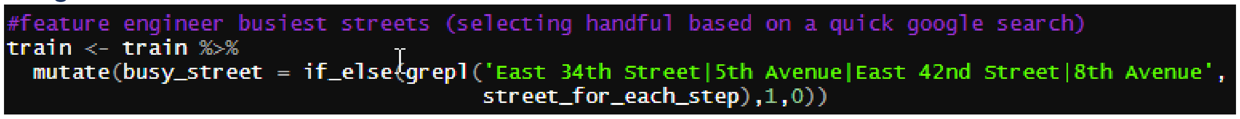
```
#feature engineer # turns and left turns
train <- train %>%
  mutate(left_turns = str_count(step_direction, "left"),
         turns = str_count(step_maneuvers, "turn"))
```

### f. Direction of Travel Feature



```
#feature engineer direction
train <- train %>%
  mutate(direction = bearing(cbind(pickup_longitude, pickup_latitude),
                             cbind(dropoff_longitude, dropoff_latitude)))
```

### g. Busiest Streets Feature



```
#feature engineer busiest streets (selecting handful based on a quick google search)
train <- train %>%
  mutate(busy_street = if_else(grepl('East 34th Street|5th Avenue|East 42nd Street|8th Avenue',
                                     street_for_each_step),1,0))
```

## External Data Sources

### a. Open Source Routing Machine (OSRM) Data

Credit to Kaggle user *oscarleo*

| **Variable** | **Description** |
|---|---|
| starting_street | Street where trip starts |
| end_street | Street where trip ends |
| total_distance | Total distance (metres) of the trip using optimal street route |
| total_travel_time | Total travel time (seconds) of the trip based on the optimal street route |
| number_of_steps | Number of steps in trip (see step_maneuvers) |
| street_for_each_step | List of streets where each step occurs |
| distance_per_step | Distance for each step in trip |
| travel_time_per_step | Travel time for each step in trip |
| step_maneuvers | The action performed in each step (turn, merge, on ramp, etc.) |
| step_direction | The direction for each maneuver (left, right, etc.) |
| step_location_list | Coordinates for each maneuver |

### b. 2016 NYC Weather Data

Obtained from https://w2.weather.gov/climate/xmacis.php?wfo=okx

| **Variable** | **Description** |
|---|---|
| date | Date |
| maximum.temperature | Highest temperature (°F) for the day |
| minimum.temperature | Lowest temperature (°F) for the day |
| average.temperature | Average temperature (°F) for the day |
| precipitation | Amount of rain fall (inches) for the day |
| snow_fall | Amount of snow fall (inches) for the day |
| snow_depth | Depth of snow level (inches) for the day |