Predict Taxi Fares with Random Forests

Introduction

To drive a yellow New York taxi, you have to hold a "medallion" from the city's *Taxi and Limousine Commission*. Recently, one of those changed hands for over one million dollars, which shows how lucrative the job can be. But this is the age of business intelligence and analytics! Even taxi drivers can stand to benefit from some careful investigation of the data, guiding them to maximize their profits. We will also use regression trees and random forests to build a model to predict the locations and times when the biggest fares can be earned.

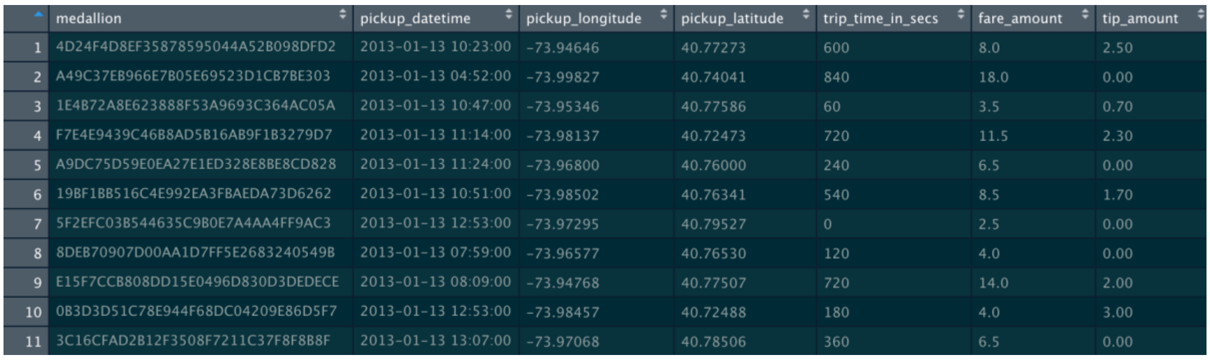
The dataset

In this project, we will analyze a random sample of 49999 New York journeys made in 2013. It has the following columns:

- medallion: Medallion ID.
- pickup_datetime: The time when the taxi driver pick up the customer.
- pickup_longitude: The longitude of the location where the taxi driver pick up the customer.
- pickup_latitude: The latitude of the location where the taxi driver pick up the customer.
- Trip_time_in_secs: Customer trip duration in second.
- fare_amount: The amount of fare paid by the customer.
- tip_amount: The amount of tip paid by the customer.

Let’s load the dataset and packages, and take a look at the first few rows.

~~~
```{r}
library(tidyverse)
library(ggmap)
library(viridis)
library(tree)
library(lubridate)
library(randomForest)
library(party)
taxi <- read_csv("taxi.csv")
head(taxi)
```
~~~

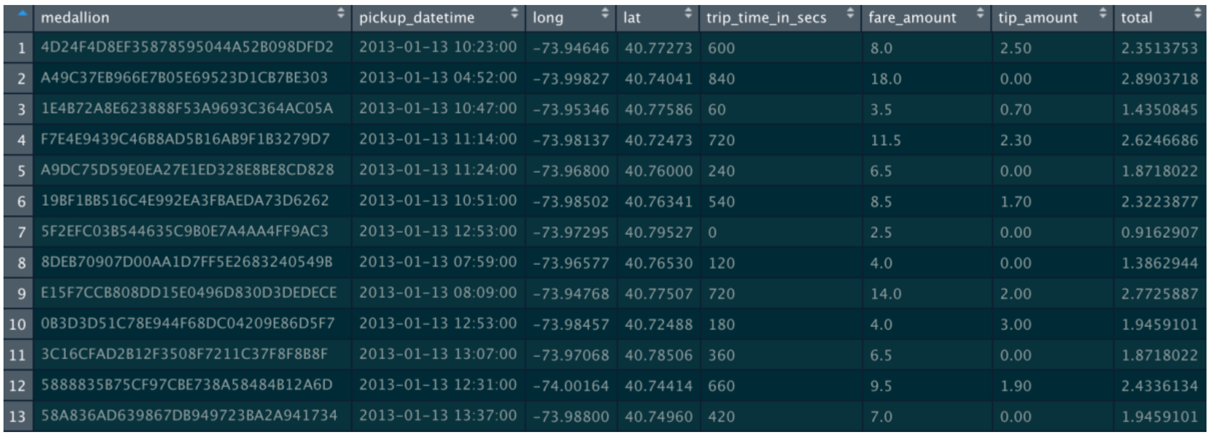


	medallion	pickup_datetime	pickup_longitude	pickup_latitude	trip_time_in_secs	fare_amount	tip_amount
1	4D24F4D8EF35878595044A52B098DFD2	2013-01-13 10:23:00	-73.94646	40.77273	600	8.0	2.50
2	A49C37EB966E7B05E69523D1CB7BE303	2013-01-13 04:52:00	-73.99827	40.74041	840	18.0	0.00
3	1E4B72A8E623888F53A9693C364AC05A	2013-01-13 10:47:00	-73.95346	40.77586	60	3.5	0.70
4	F7E4E9439C46B8AD5B16AB9F1B3279D7	2013-01-13 11:14:00	-73.98137	40.72473	720	11.5	2.30
5	A9DC75D59E0EA27E1ED328E8BE8CD828	2013-01-13 11:24:00	-73.96800	40.76000	240	6.5	0.00
6	19BF1BB516C4E992EA3FBAEDA73D6262	2013-01-13 10:51:00	-73.98502	40.76341	540	8.5	1.70
7	5F2EFC03B544635C9B0E7A4AA4FF9AC3	2013-01-13 12:53:00	-73.97295	40.79527	0	2.5	0.00
8	8DEB70907D00AA1D7FF5E2683240549B	2013-01-13 07:59:00	-73.96577	40.76530	120	4.0	0.00
9	E15F7CCB808DD15E0496D830D3DEDECE	2013-01-13 08:09:00	-73.94768	40.77507	720	14.0	2.00
10	0B3D3D51C78E944F68DC04209E86D5F7	2013-01-13 12:53:00	-73.98457	40.72488	180	4.0	3.00
11	3C16CFAD2B12F3508F7211C37F8F8B8F	2013-01-13 13:07:00	-73.97068	40.78506	360	6.5	0.00

As you can see above, the taxi dataset contains the times and prices of a large number of taxi trips. We also got to know the location, the longitude and latitude, where the trip was started.

Cleaning data is a large part of any data scientist's daily work. It may not seem glamorous, but it makes the difference between a successful model and a failure. The taxi dataset needs a bit of polishing before we're ready to use it.

~~~
```{r}
# Renaming the location variables,
# dropping any journeys with zero fares and zero tips,
# and creating the total variable as the log sum of fare and tip
taxi <- taxi %>%
  rename(long = pickup_longitude, lat = pickup_latitude) %>%
  filter(fare_amount>0| tip_amount>0) %>%
  mutate(total = log(fare_amount + tip_amount)) # We use log to reduce the skew
```
~~~



	medallion	pickup_datetime	long	lat	trip_time_in_secs	fare_amount	tip_amount	total
1	4D24F4D8EF35878595044A52B098DFD2	2013-01-13 10:23:00	-73.94646	40.77273	600	8.0	2.50	2.3513753
2	A49C37EB966E7B05E69523D1CB7BE303	2013-01-13 04:52:00	-73.99827	40.74041	840	18.0	0.00	2.8903718
3	1E4B72A8E623888F53A9693C364AC05A	2013-01-13 10:47:00	-73.95346	40.77586	60	3.5	0.70	1.4350845
4	F7E4E9439C46B8AD5B16AB9F1B3279D7	2013-01-13 11:14:00	-73.98137	40.72473	720	11.5	2.30	2.6246686
5	A9DC75D59E0EA27E1ED328E8BE8CD828	2013-01-13 11:24:00	-73.96800	40.76000	240	6.5	0.00	1.8718022
6	19BF1BB516C4E992EA3FBAEDA73D6262	2013-01-13 10:51:00	-73.98502	40.76341	540	8.5	1.70	2.3223877
7	5F2EFC03B544635C9B0E7A4AA4FF9AC3	2013-01-13 12:53:00	-73.97295	40.79527	0	2.5	0.00	0.9162907
8	8DEB70907D00AA1D7FF5E2683240549B	2013-01-13 07:59:00	-73.96577	40.76530	120	4.0	0.00	1.3862944
9	E15F7CCB808DD15E0496D830D3DEDECE	2013-01-13 08:09:00	-73.94768	40.77507	720	14.0	2.00	2.7725887
10	0B3D3D51C78E944F68DC04209E86D5F7	2013-01-13 12:53:00	-73.98457	40.72488	180	4.0	3.00	1.9459101
11	3C16CFAD2B12F3508F7211C37F8F8B8F	2013-01-13 13:07:00	-73.97068	40.78506	360	6.5	0.00	1.8718022
12	5888835B75CF97CBE738A58484B12A6D	2013-01-13 12:31:00	-74.00164	40.74414	660	9.5	1.90	2.4336134
13	58A836AD639867DB949723BA2A941734	2013-01-13 13:37:00	-73.98800	40.74960	420	7.0	0.00	1.9459101

While the dataset contains taxi trips from all over New York City, the bulk of the trips are to and from Manhattan, so let's focus only on trips initiated there.

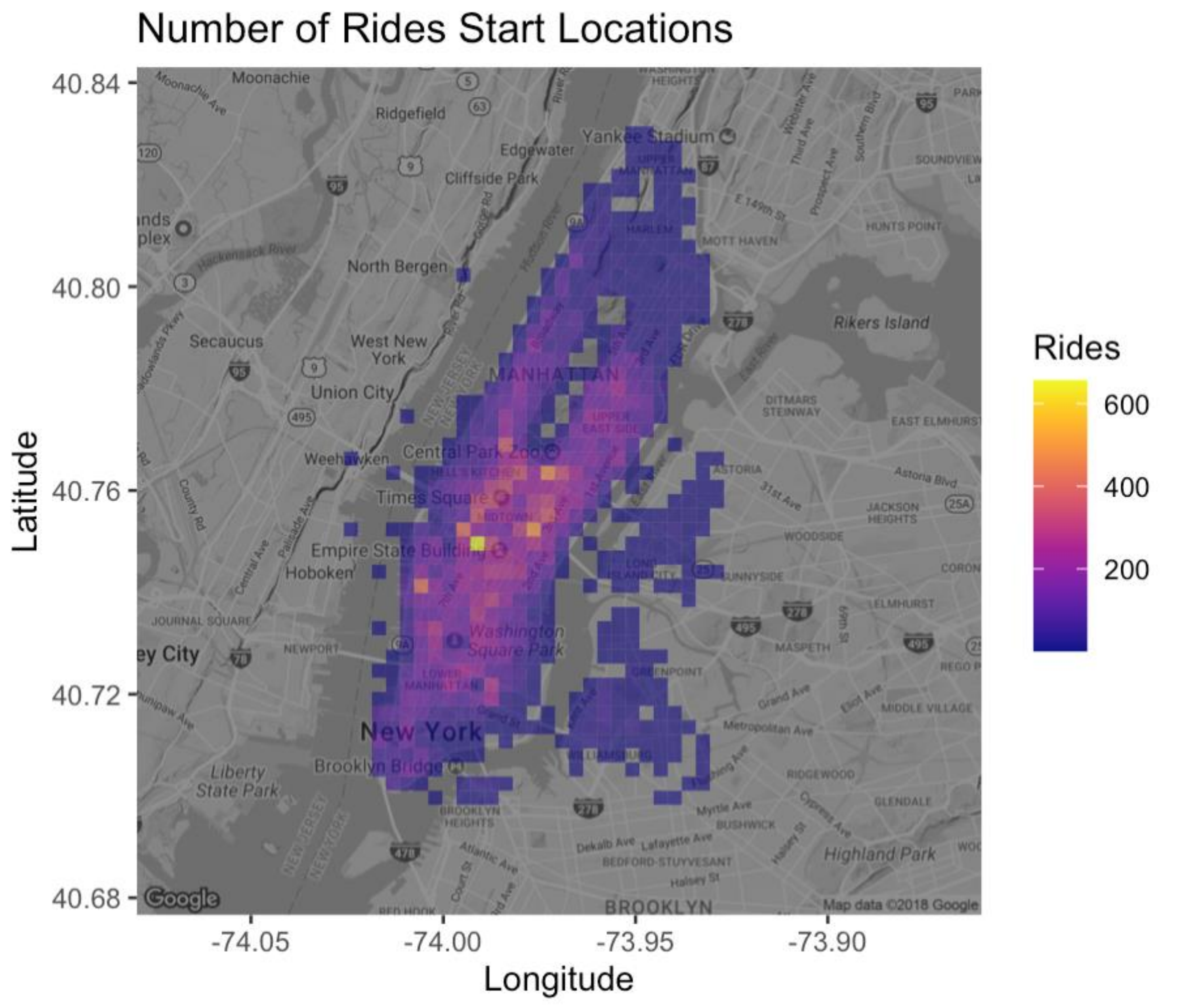
~~~
```{r}
# Reducing the data to taxi trips starting in Manhattan
# Manhattan is bounded by the rectangle with
# latitude from 40.70 to 40.83 and
# longitude from -74.025 to -73.93
taxi <- taxi %>%
  filter(between(lat, 40.70, 40.83), between(long, -74.025, -73.93))
```
~~~

Empirical Analysis

Now we're going to draw a map using ggmap and ggplot2 package to visualize location in Manhattan where people tend to start their taxi trip.

~~~
```{r}
# Drawing a density map with the number of journey start locations
ggmap(manhattan, darken = 0.5) +
  scale_fill_viridis(option = "plasma") +
~~~

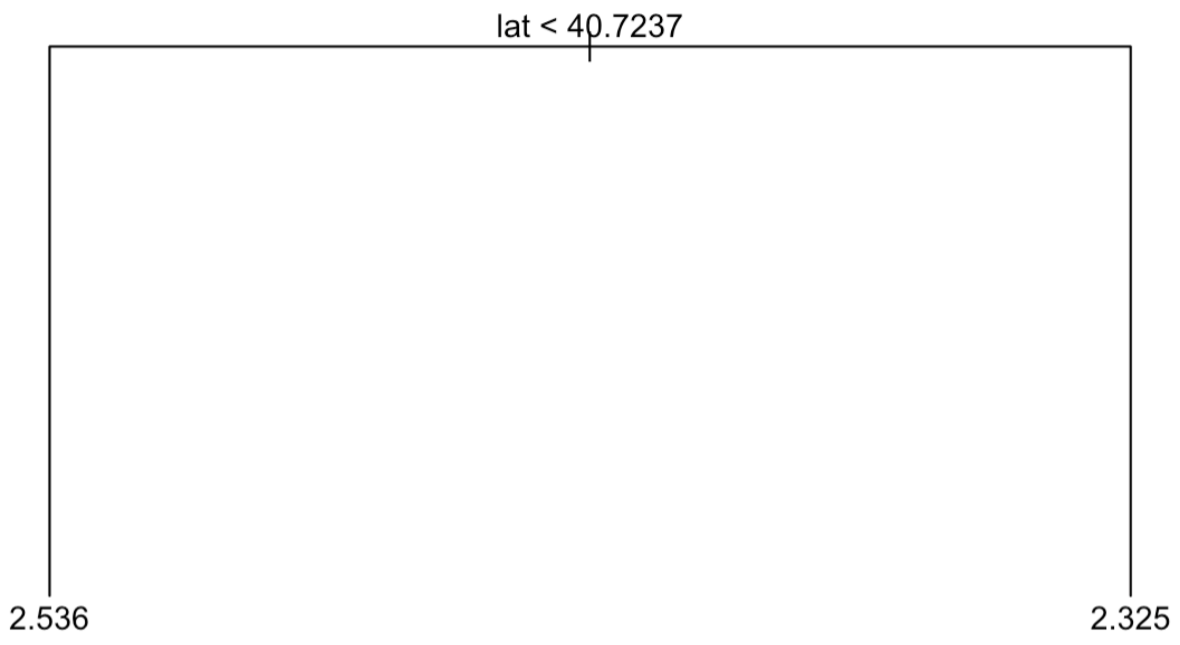
~~~
 geom_bin2d(data = taxi, aes(x = long, y = lat), bins = 60, alpha=0.6) +
 labs(x = "Longitude", y = "Latitude", fill = "Rides", title = "Number of Rides Start Locations")
```
~~~



The map from the previous task showed that the trips are highly concentrated in the business and tourist areas. We also see that some taxi trips originating in Brooklyn slipped through, but that's fine. We're now going to use a regression tree to predict the total fare, with lat and long being the predictors. The tree algorithm will find cutpoints in those predictors that result in the decision tree with the best predictive capability.

~~~
```{r}
# Fitting a tree to lat and long
fitted_tree <- tree(total ~ lat + long, data = taxi)

# Draw a diagram of the tree structure
plot(fitted_tree)
text(fitted_tree)
```
~~~



The tree above looks a bit frugal; it only includes one split: It predicts that trips where lat < 40.7237 are more expensive, which makes sense as it is downtown Manhattan. But that's it. It didn't even include long as tree deemed that it didn't improve the predictions. Taxi drivers will need more information than this, and any driver paying for your data-driven insights would be disappointed. As we know from Robert de Niro, it's best not to upset New York taxi drivers. So we will start by adding some more predictors related to the *time* the taxi trip was made.

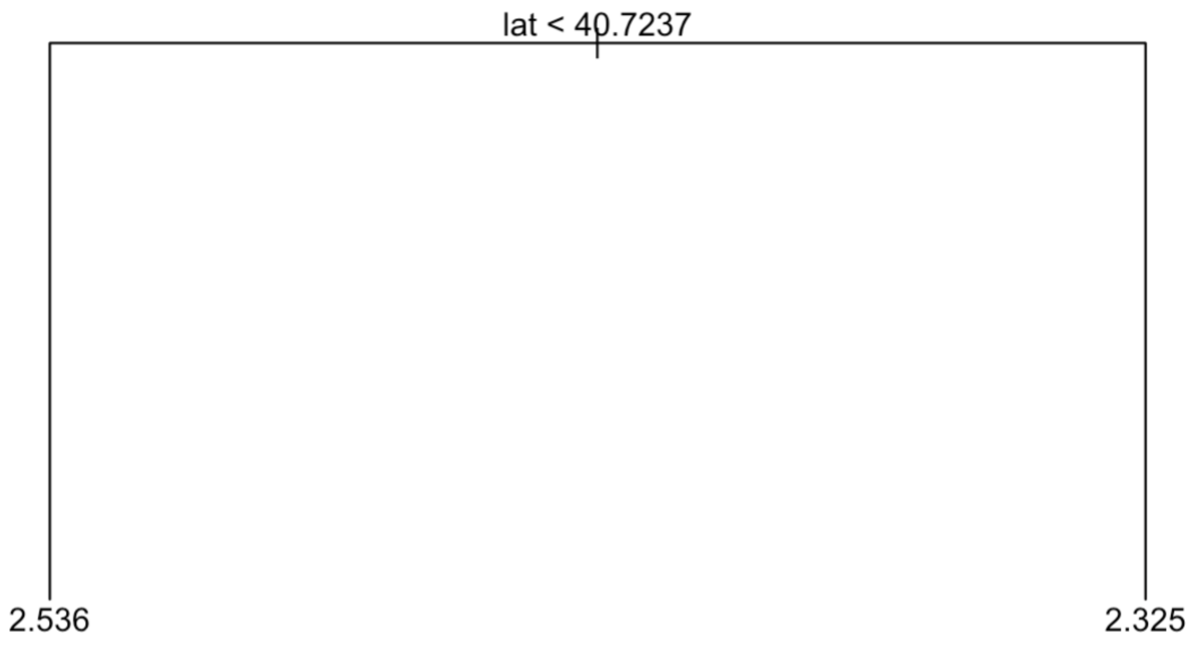
~~~
```{r}
# Generate the three new time variables
taxi <- taxi %>%
  mutate(hour = hour(pickup_datetime),
         wday = wday(pickup_datetime, label = TRUE),
         month = month(pickup_datetime,label = TRUE))
```
~~~

Let's try fitting a new regression tree where we include the new time variables.

~~~
```{r}
# Fitting a tree with total as the outcome and
# lat, long, hour, wday, and month as predictors
fitted_tree <- tree(total~lat + long+hour+wday+month, data = taxi)

# Draw a diagram of the tree structure
plot(fitted_tree)
text(fitted_tree)

# Summarizing the performance of the tree
summary(fitted_tree)
```
~~~



```
Regression tree:
tree(formula = total ~ lat + long + hour + wday + month, data = taxi)
Variables actually used in tree construction:
[1] "lat"
Number of terminal nodes:  2
Residual mean deviance:  0.3041 = 13910 / 45760
Distribution of residuals:
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-1.61900 -0.37880 -0.04244  0.00000  0.32660  2.69900
```

The regression tree has not changed after including the three-time variables. This is likely because latitude is still the most promising first variable to split the data on, and after that split, the other variables are not informative enough to be included. A random forest model, where many different trees are fitted to subsets of the data, may consist of the other variables that make it up in some of the trees.

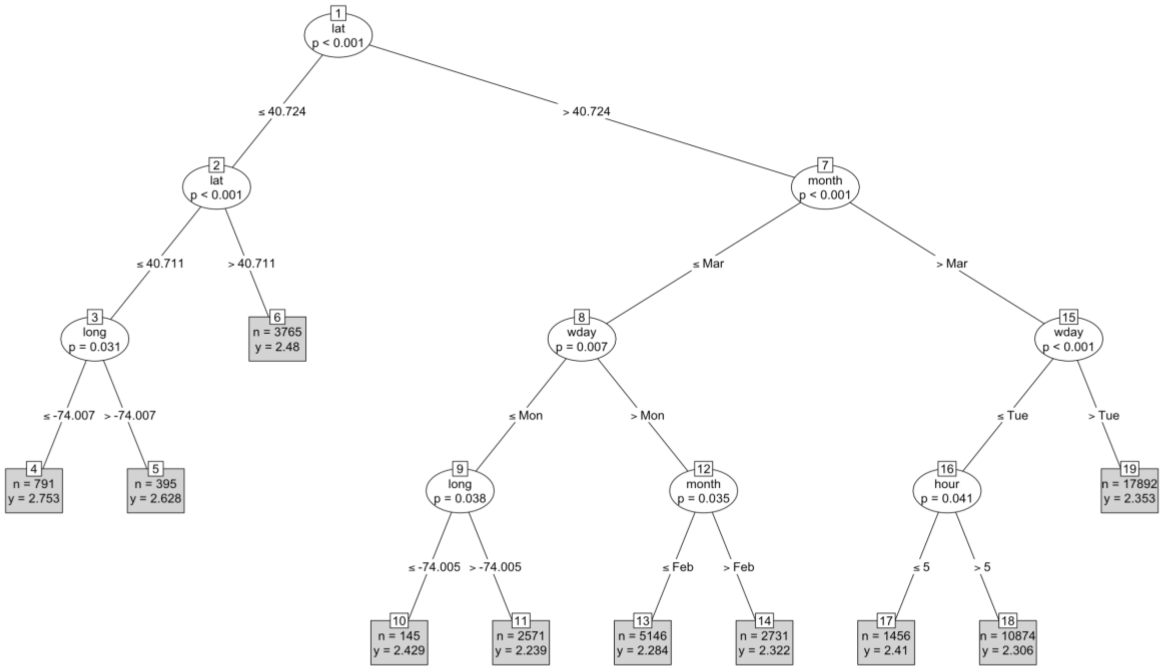
```{r}
# Fitting a random forest
fitted_forest <-  randomForest(total~lat + long+hour+wday+month, data=taxi, ntree=80,
sampsize = 10000)

# Printing the fitted_forest object
summary(fitted_forest)
```

```
                Length Class  Mode
call                5  -none- call
type                1  -none- character
predicted       45766  -none- numeric
mse                80  -none- numeric
```

```
rsq             80  -none- numeric
oob.times       45766  -none- numeric
importance         5  -none- numeric
importanceSD       0  -none- NULL
localImportance    0  -none- NULL
proximity          0  -none- NULL
ntree              1  -none- numeric
mtry               1  -none- numeric
forest            11  -none- list
coefs              0  -none- NULL
y            45766  -none- numeric
test               0  -none- NULL
inbag              0  -none- NULL
terms              3  terms  call
```

Let's draw the random forest plot.
```{r}
x <- ctree(total~lat + long+hour+wday+month, data=taxi)
plot(x, type = "simple")
```

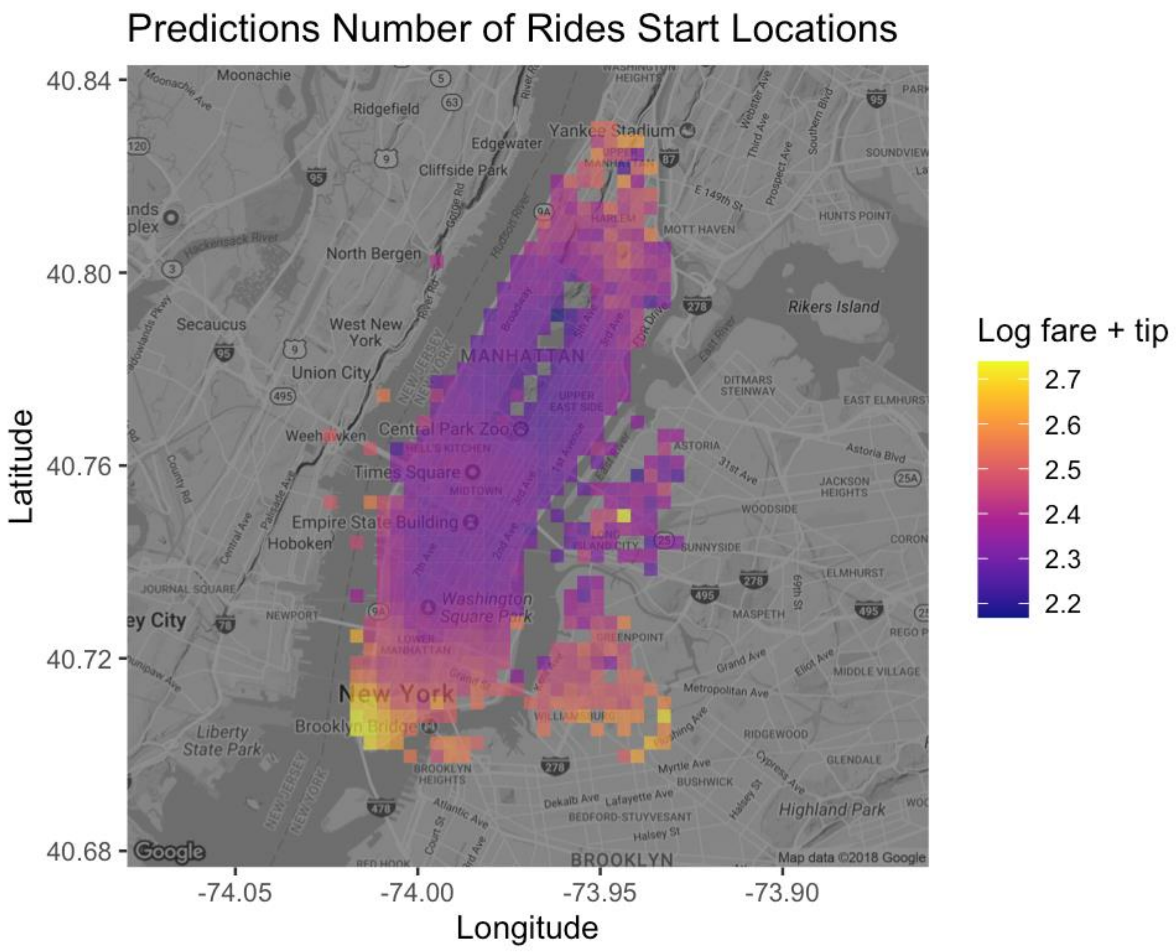


In the output of fitted_forest, you should see the Mean of squared residuals, that is, the average of the squared errors the model makes. If you scroll up and check the summary of fitted_tree, you'll find Residual mean deviance, which is the same number. If you compare these numbers, you'll see that fitted_forest has a slightly lower error. Neither predictive model is *that* good in statistical terms, they explain only about 3% of the variance.

Now, let's take a look at the predictions of fitted_forest projected back onto Manhattan.

```{r}
# Extracting the prediction from fitted_forest
taxi$pred_total <- fitted_forest$predicted

# Plotting the predicted mean trip prices from according to the random forest
# Drawing a density map with the number of journey start locations
ggmap(manhattan, darken = 0.5) +
  scale_fill_viridis(option = "plasma") +
  stat_summary_2d(data = taxi, aes(x=long, y = lat, z =pred_total), bins = 60, alpha = 0.6, fun
= mean)+
  labs(x="Longitude", y = "Latitude", fill = "Log fare + tip", title = "Predictions Number of
Rides Start Locations ")
```



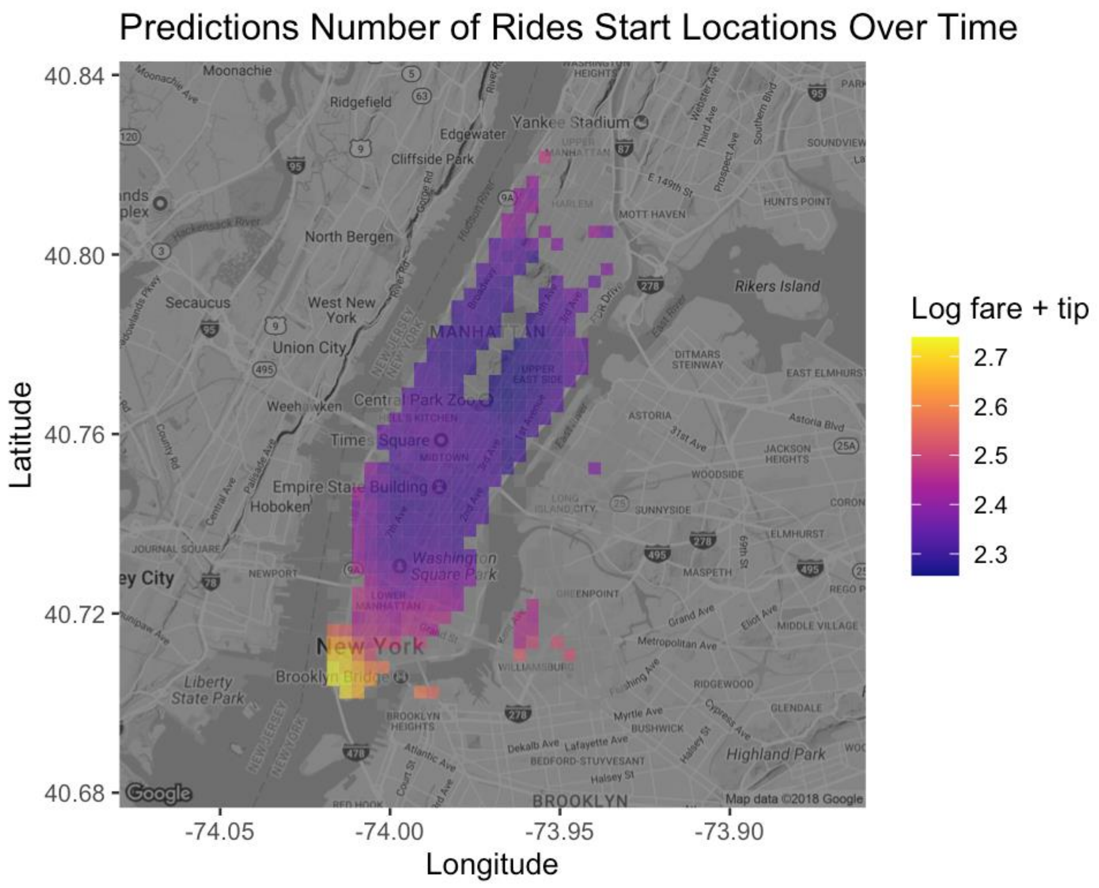
Looking at the map with the predicted fares, we see that taxi fares in downtown Manhattan are predicted to be high, while midtown is lower. This map only shows the prediction as a function of lat and long, but we could also plot the predictions over time or a combination of time and space, but we'll leave that for another time.

For now, let's compare the map with the predicted fares with a new map showing the mean fares according to the data.

```{r}
```

```
# Funtion that returns the mean *if* there are 15 or more datapoints
mean_if_enough_data <- function(x) {
  ifelse(length(x)>=15, mean(x), NA)
}

# Plotting the mean trip prices fron the data
ggmap(manhattan, darken = 0.5) +
  scale_fill_viridis(option = "plasma") +
  stat_summary_2d(data = taxi, aes(x=long, y = lat, z =pred_total), bins = 60, alpha = 0.6, fun
= mean_if_enough_data)+
  labs(x="Longitude", y = "Latitude", fill = "Log fare + tip", title = "Predictions Number of
Rides Start Locations Over Time")
```



Conclusion

So it looks like the random forest model captured some of the patterns in our data. At this point in the analysis, we could do many more things that we haven't done. We could add more predictors if we have the data. We could try to fine-tune the parameters of randomForest. And we should test the model on a hold-out test dataset. So from the map above, taxi drivers in NYC can expect to spend the most on a taxi ride in lower Manhattan.

Source

- https://en.wikipedia.org/wiki/Taxi_medallion