Data Science/Machine Learning Project

NYC Taxis Tip Prediction

Background:

I was taking a cab on my way to my house from the airport, being unaware that tipping was customary here in the US, I initially refused to pay the driver a tip. After much contemplation, I gave him a dollar on $59 fare amount. I realized this was not correct and was rude on my behalf. The culture here is completely different from where I grew up. After having several conversations with cab drivers, I came to realize that they usually aren't tipped well even after giving a perfect ride to riders.

Proposal:

I want to help cab drivers, earn a bit more on tips for their hard work and time spent driving. Hence, I started working on a dataset for NYC taxi drivers that would help them analyze and predict how much tip they would receive from drivers given their status (such as location, time of day, day of the week, distance, time traveled etc.).

```
df <- read.csv("C:\\Users\\Tejas\\Documents\\green_tripdata_2015-09.csv")

nrow(df) #number of rows in the dataset

## [1] 1494926

ncol(df) #number of columns in the dataset

## [1] 21

colnames(df) #column names

##  [1] "VendorID"              "lpep_pickup_datetime"
##  [3] "Lpep_dropoff_datetime" "Store_and_fwd_flag"
##  [5] "RateCodeID"            "Pickup_longitude"
```

```
## [7] "Pickup_latitude" "Dropoff_longitude"
## [9] "Dropoff_latitude" "Passenger_count"
## [11] "Trip_distance" "Fare_amount"
## [13] "Extra" "MTA_tax"
## [15] "Tip_amount" "Tolls_amount"
## [17] "Ehail_fee" "improvement_surcharge"
## [19] "Total_amount" "Payment_type"
## [21] "Trip_type"

# checking for NA
colSums(is.na(df[,]))

##              VendorID  lpep_pickup_datetime Lpep_dropoff_datetime
##                     0                     0                     0
##    Store_and_fwd_flag            RateCodeID      Pickup_longitude
##                     0                     0                     0
##       Pickup_latitude     Dropoff_longitude      Dropoff_latitude
##                     0                     0                     0
##       Passenger_count         Trip_distance           Fare_amount
##                     0                     0                     0
##                 Extra               MTA_tax            Tip_amount
##                     0                     0                     0
##          Tolls_amount             Ehail_fee improvement_surcharge
##                     0               1494926                     0
##          Total_amount          Payment_type             Trip_type
##                     0                     0                     4

# We can see Ehail_fee has no data hence we will eliminate it
df <- subset(df,select = -c(Ehail_fee))

# We can also see that only 4 observations are missing in Trip_type of 1.49 million observations
df <- df[complete.cases(df),]

# Checking the datatypes of each variable in the dataframe
sapply(df, class)
```

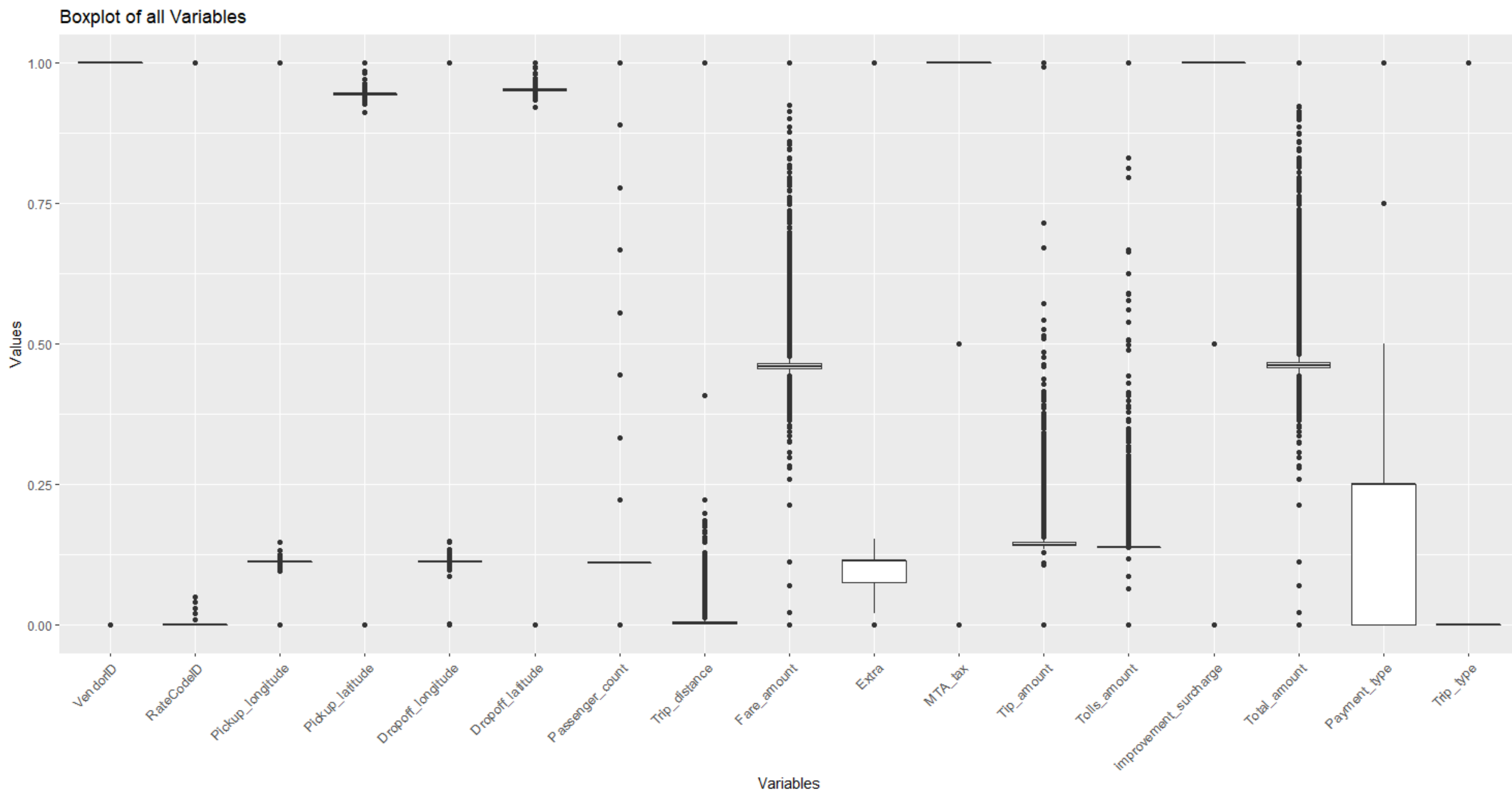
```
##              VendorID  lpep_pickup_datetime Lpep_dropoff_datetime 
##             "integer"              "factor"              "factor" 
##    Store_and_fwd_flag            RateCodeID      Pickup_longitude 
##              "factor"             "integer"             "numeric" 
##       Pickup_latitude     Dropoff_longitude      Dropoff_latitude 
##             "numeric"             "numeric"             "numeric" 
##       Passenger_count         Trip_distance           Fare_amount 
##             "integer"             "numeric"             "numeric" 
##                 Extra               MTA_tax            Tip_amount 
##             "numeric"             "numeric"             "numeric" 
##          Tolls_amount improvement_surcharge          Total_amount 
##             "numeric"             "numeric"             "numeric" 
##          Payment_type             Trip_type 
##             "integer"             "integer"
```

This gives us an understanding that the data types for 'lpep_pickup_datetime' and 'Lpep_dropoff_datetime' aren't in the format that we want. Rest of the variables are fine.

Data Preprocessing and Exploratory Data Analysis

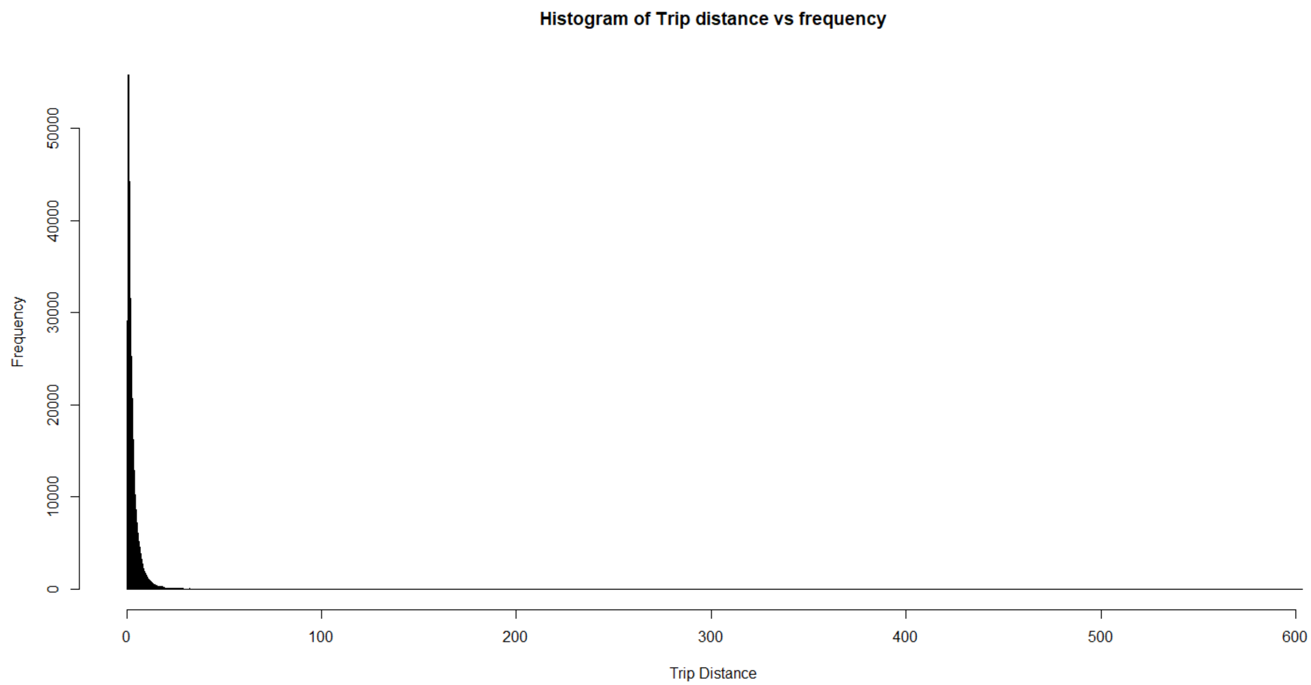
Check for outliers using boxplots. Since a normal boxplot without normalizing wasn't visually easily to interpret with all the variables simultaneously. All the numerical variables are plotted after normalizing them between 0 and 1.

```
# Normal variables were scaled to better understand their behavior

# Boxplot of all variables
```

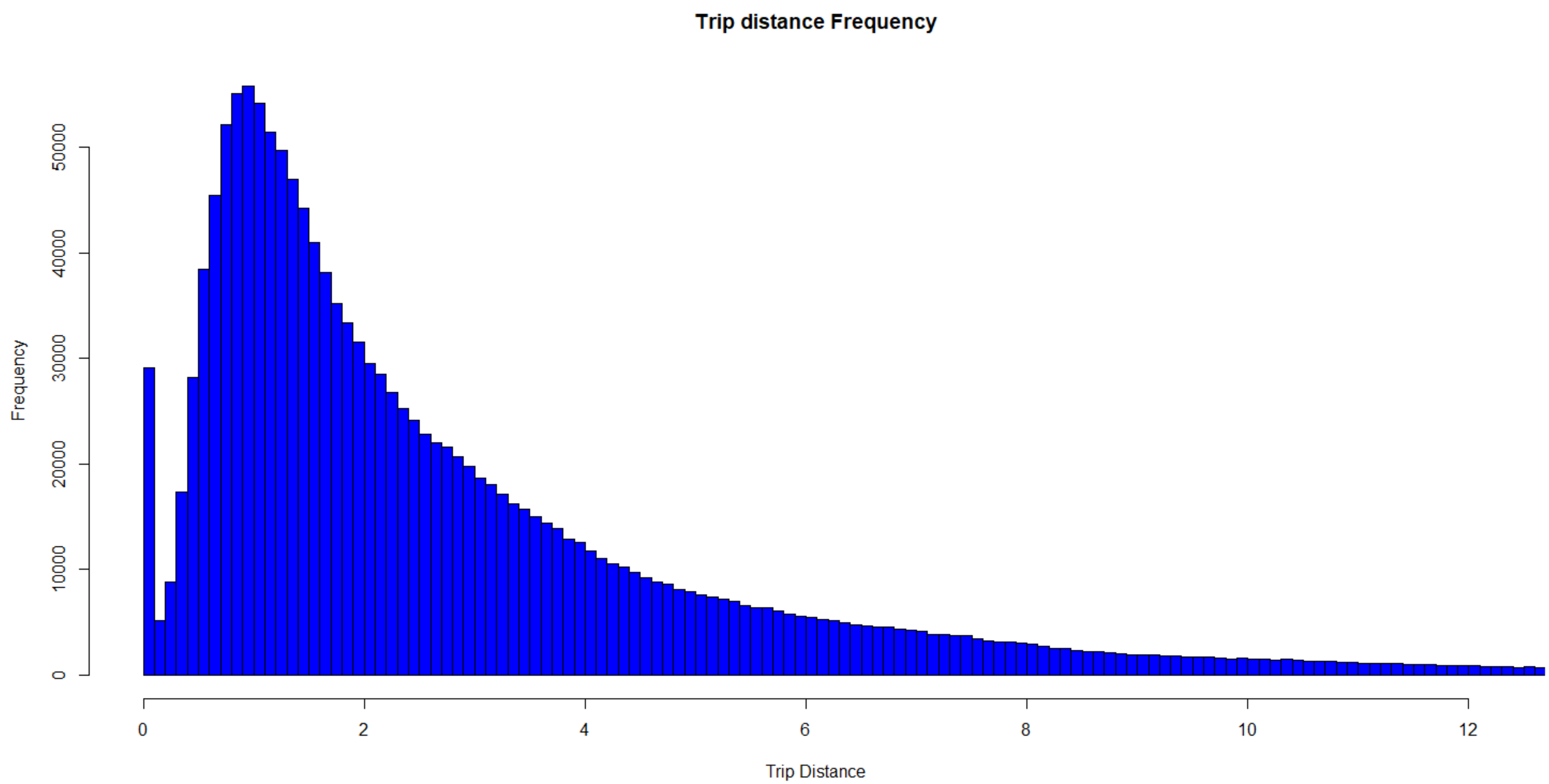


This plot shows us that, there are a number of outliers for variables such as: Trip Distance, Fare Amount, Tip amount, Tolls Amount and Total Amount. We will look into Trip Distance first.

```r
gc()

#Lets look at Trip Distance as it shows a high percentage of outliers
```



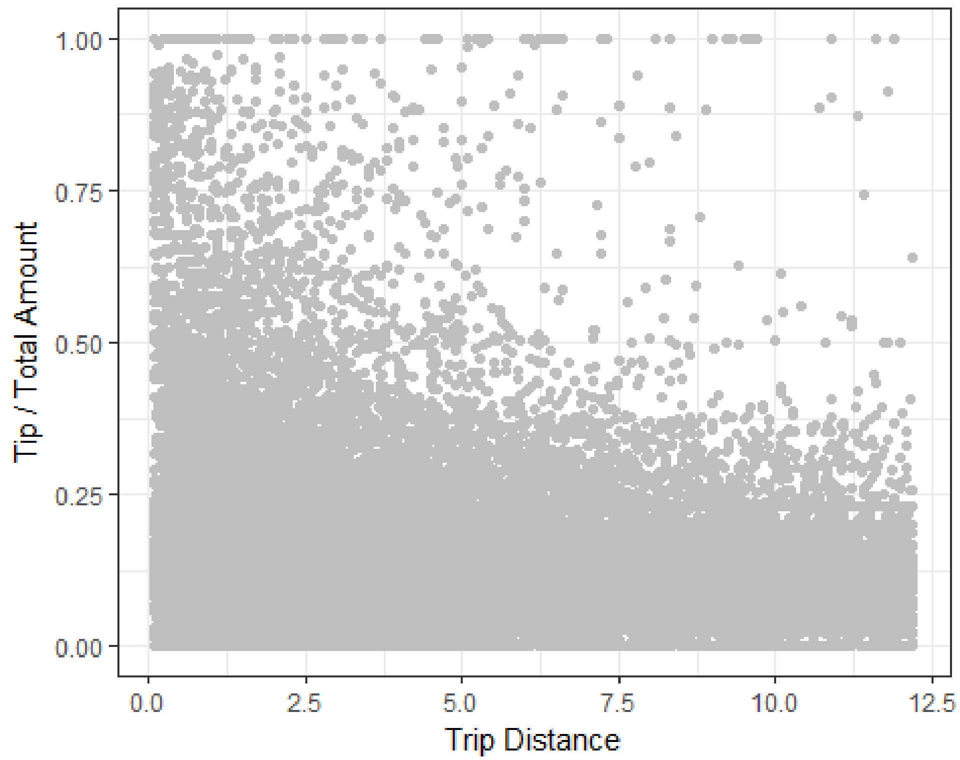
Since the figure is quite vague, for visualization purposes, all the outliers are excluded by limiting the x values to x = mean +/- standard deviation*3 to give a clear idea about the distribution.



It can be inferred that the graph is non-normal i.e. right skewed and the mode < median < mean. Here the t-statistic will hold valid because of its robustness to deviations from normality. The skewness can be explained as the distance starts from zero and will always have a very high number of rides that have a low travelling distance.

Hypothesis: The distance data isn't random. For it to be random the graph would have been normally distributed. The relation might be affected by the fact that people that travel long distances can't afford to pay the fare for a cab. Instead they use the public transportations system such as bus, metro etc. People who go to work daily would prefer a cheaper option. Moreover, the cab drivers wouldn't want to travel to a remote place from where they wouldn't get a return fare. It can be pointed out that there is an anomaly seen in the graph. There are number of rides where the distance covered is zero which is incorrect. We thus need to clean the data through for better predictions/results.

Plot - Trip Distance v/s Tip/Total amt. From the graph, we notice that a good percentage of tip to total amount ratio is high in case of short trips. Drivers are better of driving short distances to let the tips keep coming.



Airport trips - According to the data dictionary, it is mentioned that the rate code ids are

1 = Standard rate

2 = JFK

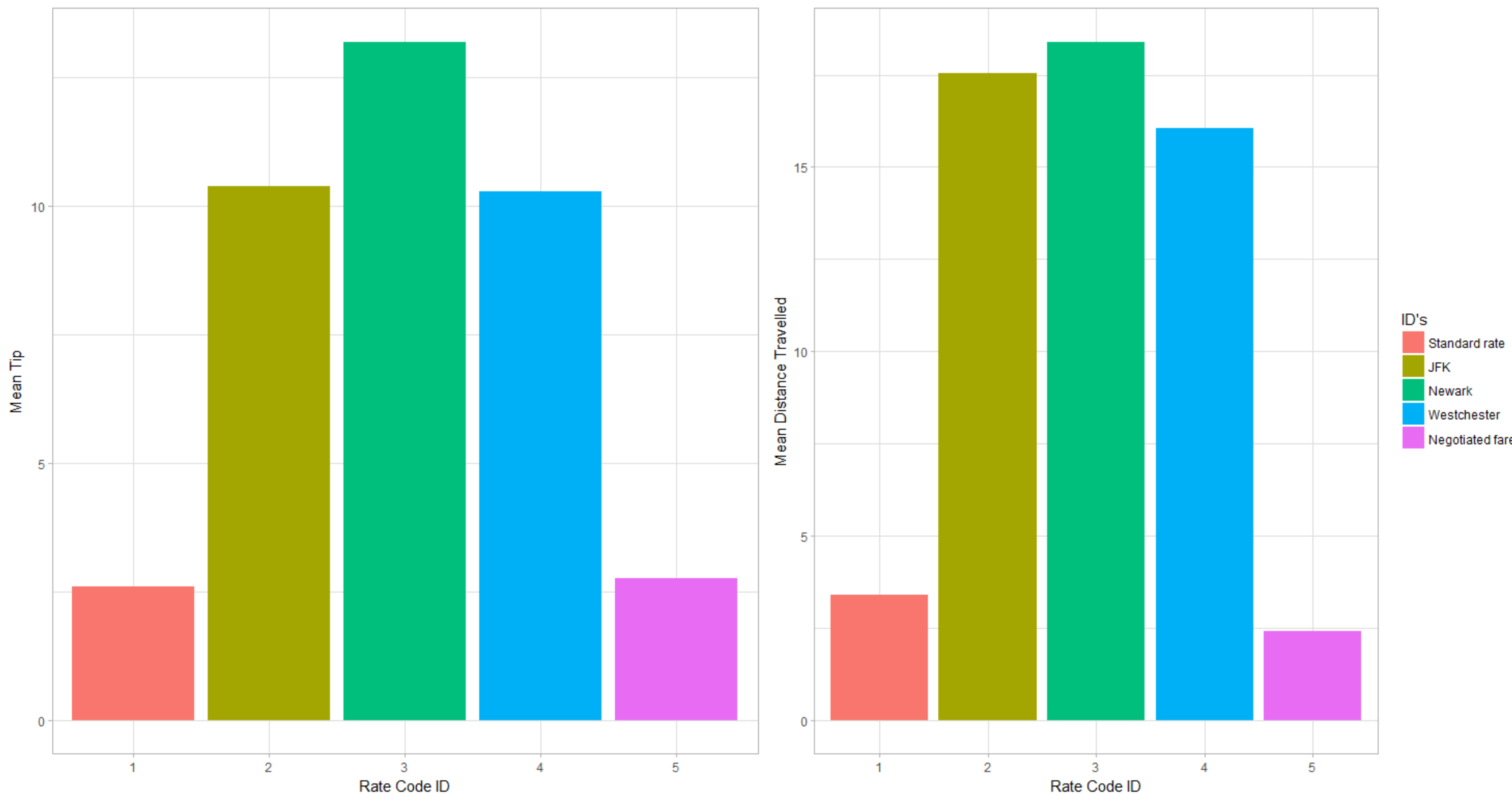
3 = Newark

4 = Westchester

5 = Negotiated fare

6 = Group ride

```r
plyr::count(df, 'RateCodeID')

##   RateCodeID    freq
## 1          1 1454464
## 2          2    4435
## 3          3    1117
## 4          4     925
## 5          5   33943
## 6          6      36
## 7         99       2

# Upon analysis it was found that ratecode id 99 was an error
```



*The graph plotted above is a filtered dataset with observations that have payment type as Credit Card. Later, it was revealed that other payment types had an anomaly that needs to be fixed.

This plot shows the average tip received from different rate code ID's. This gives us a better idea of how drivers can earn a bit more than their usual earnings. Tips received from airports are nearly as 10x their standard tip. This could be accounted for a number of reasons 1) The distance travelled from airports to their respective destinations will be higher than a normal ride, mainly because of the location of the airports being local. Thus, making the commuting distance more and subsequently higher fares. 2) The local taxes are levied on riders for catching a cab from the airport, thus contributing to the total amount and subsequently higher tips. 3) Tourists who come to visit, might be unaware of how much to tip the driver, driving the mean tip even higher.

After further analysis, it was found that rate code ID 6 though showing some distance travelled, is showing almost zero tip received. ID 6 corresponds to group ride. Let's dive into it a bit, by looking the type of payment for each rate code ID.

```
## Cross table of the count of rides w.r.t the payment type and rate code id.
```

	Credit card	Cash	No charge	Dispute	Unknown
Standard rate	687004	758515	4767	4108	70
JFK	1740	2498	138	58	1
Newark	460	499	110	47	1
Westchester	475	434	12	4	0
Negotiated fare	11605	21725	462	149	2
Group ride	***0***	***26***	***8***	***2***	***0***

Since a majority of the payment done in group rides (rate id 6) is done by cash, the tips aren't recorded.

Upon further analysis, I found that, the tip amount recorded is zero for all cases but one. All the transactions which are not electronic have zero tips.

	Payment_type	Tips Recorded	Tips (=$0) Recorded
1	Credit card	701284	98554
2	Cash	783697	783695
3	No charge	5497	5462
4	Dispute	4368	4365
5	Unknown	74	74

It is understood that if passengers pay by cash, their tips won't be recorded by drivers as they want to avoid taxes. Ironically, all the other types of trips also have zero tips. It would be sensible to remove those points later for modeling as they will have no good impact on our predictions.

From the data provided, we can derive a few variables of our own.

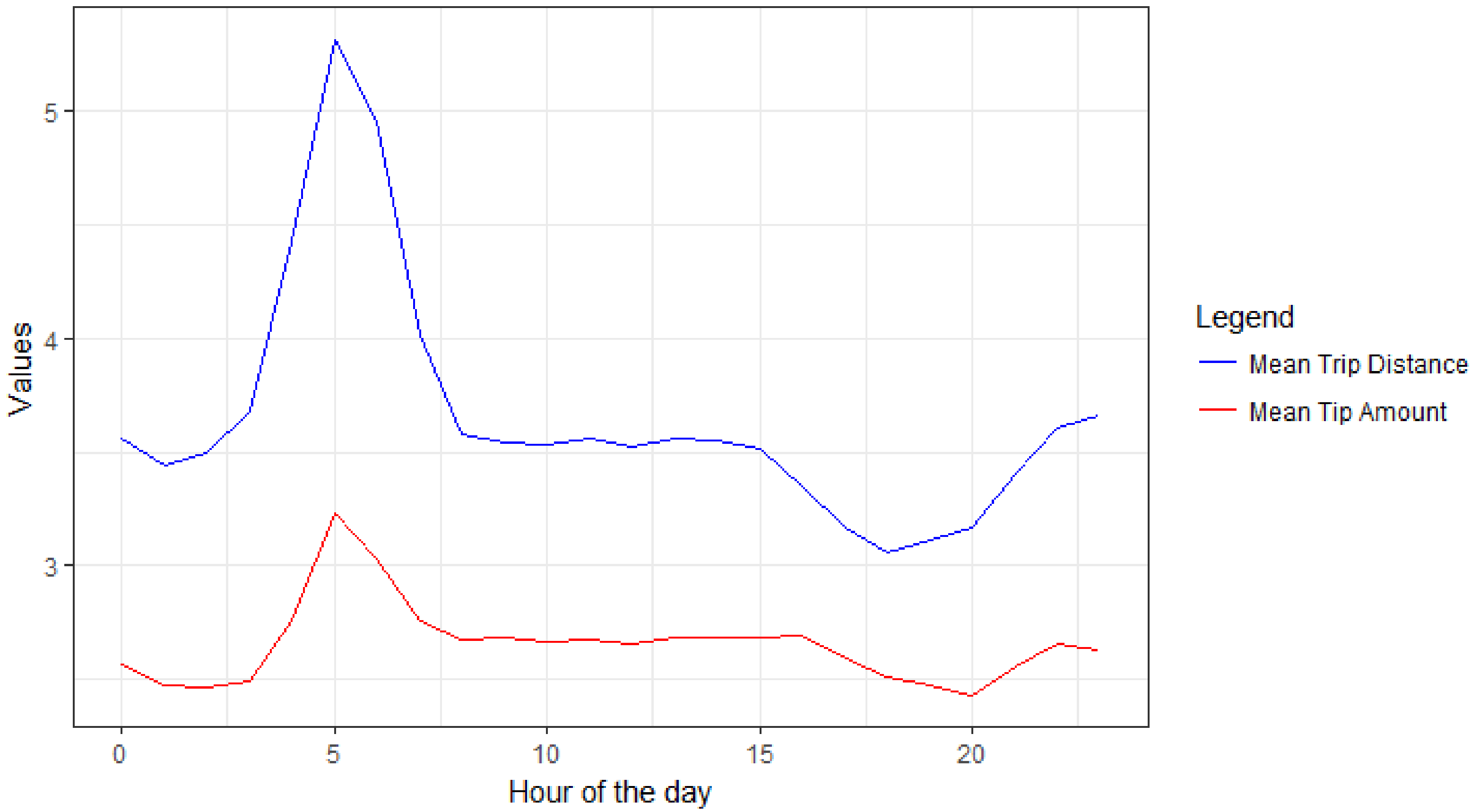
```
#Converting Trip Durations to secs
x1 <- strptime(df$lpep_pickup_datetime, "%Y-%m-%d %H:%M:%OS")
x2 <- strptime(df$Lpep_dropoff_datetime, "%Y-%m-%d %H:%M:%OS")
df$trip_duration <- as.numeric(x2-x1,units="secs") #this is a derived feature

#dividing into hours
time.category <- with(df, ifelse(trip_duration <= (4*3600), 1,
                                 ifelse(trip_duration >= 5*3600 & trip_duration <= 24*3600, 2, 3))
)
aggregate(df$trip_duration,by=list(time.category),FUN=length)

##   Group.1       x
## 1       1 1486351
## 2       2    8382
## 3       3     187
```

As can be seen 8382 rides are between 5 hrs and 24 hrs and 187 rides are above 24 hours. These are obviously outliers as passengers don't travel usually for more than 2 hours. As an exception I will consider 4 hours to be the upper limit while building my model.

Let's see if the time of the day has any impact on the distance covered.

```
##    hour Trip_distance Tip_amount
## 1     0      3.115276   1.260581
## 2     1      3.017347   1.189363
## 3     2      3.046176   1.149693
## 4     3      3.212945   1.089771
## 5     4      3.526555   1.058802
## 6     5      4.133722   1.414896
## 7     6      4.055686   1.579561
## 8     7      3.284394   1.411359
```

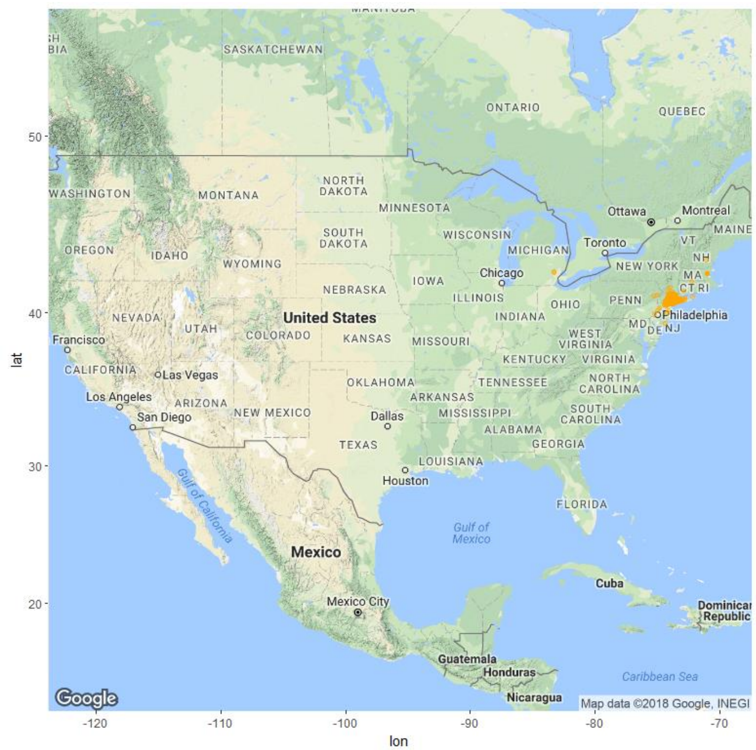
```
## 9     8       3.048450   1.454360
## 10    9       2.999105   1.421786
## 11   10       2.944482   1.250880
## 12   11       2.912015   1.172081
## 13   12       2.903115   1.163272
## 14   13       2.878294   1.136609
## 15   14       2.864304   1.090176
## 16   15       2.857040   1.109829
## 17   16       2.779852   1.170272
## 18   17       2.679114   1.163597
## 19   18       2.653222   1.186836
## 20   19       2.715597   1.212864
## 21   20       2.777052   1.177906
## 22   21       2.999223   1.286050
## 23   22       3.185394   1.394593
## 24   23       3.191538   1.343764
```



An interesting observation can be interpreted here.

The distance covered here during the 5th and 6th hour is highest along with the tip received. This could be true as people travelling early in the morning to work would need to get on time. While coming back from work they wouldn't mind catching a local transportation system. Also, subsequently it reduces throughout the day until it reaches late night till the morning. I believe this would be high mainly because of the number of rides seen on weekends, which definitely would have an impact on these late-night rides.

```r
#latitude and longtitude other than the area covered by Green Taxis
```



From this visualization it is clear that some of the coordinates fall out of the actual area of service. These coordinates need to be scrapped off and for future purposes these also need to be investigated. While plotting, it was noticed that 2110 observations had coordinates outside of USA. Interesting?

Coordinates outside of area of service (bounding box) are set to NA (reference taken from https://www.maptechnica.com/city-map/New%20York/NY/3651000). According to this, the bounding box limits are latitude=40.917577, longitude=-74.259090 at the northwest corner, and latitude=40.477399, longitude=-73.700272 at the southeast corner

```
##                   na_count
## Dropoff_longitude     2972
## Dropoff_latitude      3223
```

```
## Pickup_longitude      2312
## Pickup_latitude       2358
```

As it can be seen there are latitudes and longitudes outside of the area of service. These can be considered as outliers and can be scrapped of.

```r
# 0 passenger count
nrow(df[df$Passenger_count == 0,])

## [1] 436
```

Here we notice that the passenger count in some of these rides were zero. This is obviously incorrect, but instead of deleting these rows, it would be better to replace 0 with the median number of passengers, since there are fares recorded with those rides and tips to. Also, the median was chosen as the its histogram was skewed.

```r
# More than 7 passenger count
nrow(df[df$Passenger_count > 8,])

## [1] 16
```

Another observation was that since the number of max passengers that can travel is only 7, we can exclude those rows that have more than 7 passengers

Also, there were a few more anomalies that were observed which needed to be cleaned up. After reviewing the rates (from "http://www.nyc.gov/html/tlc/html/passenger/taxicab_rate.shtml"), it was observed that the minimum fare for a ride is $2.5. From the data it can be observed that some of the rides were less than the minimum amount.

```r
# Fare Amount less than 2.5
nrow(df[df$Fare_amount < 2.50,])

## [1] 7455
```

7455 rows have fare amount < 2.5 which can't be possible.

Digging a bit deeper, fare amount from different vendors, Creative Mobile Technologies and VeriFone Inc. have an interesting observation.

```r
nrow(df[df$Fare_amount < 0 & df$VendorID == 2,])
```

```
## [1] 2417
```

```
nrow(df[df$Fare_amount < 0 & df$VendorID == 1,])
```

```
## [1] 0
```

I see that vendor ID 2 has an issue recording fares at times. All the fares that have values less than 0 (wrongly recorded negative values) are derived only from the second vendor (i.e. Vendor ID = 2, VeriFone Inc). This should be further investigated, to avoid loss and errors in data. As of now, I'll convert all the negative data to positive to avoid loss in data.

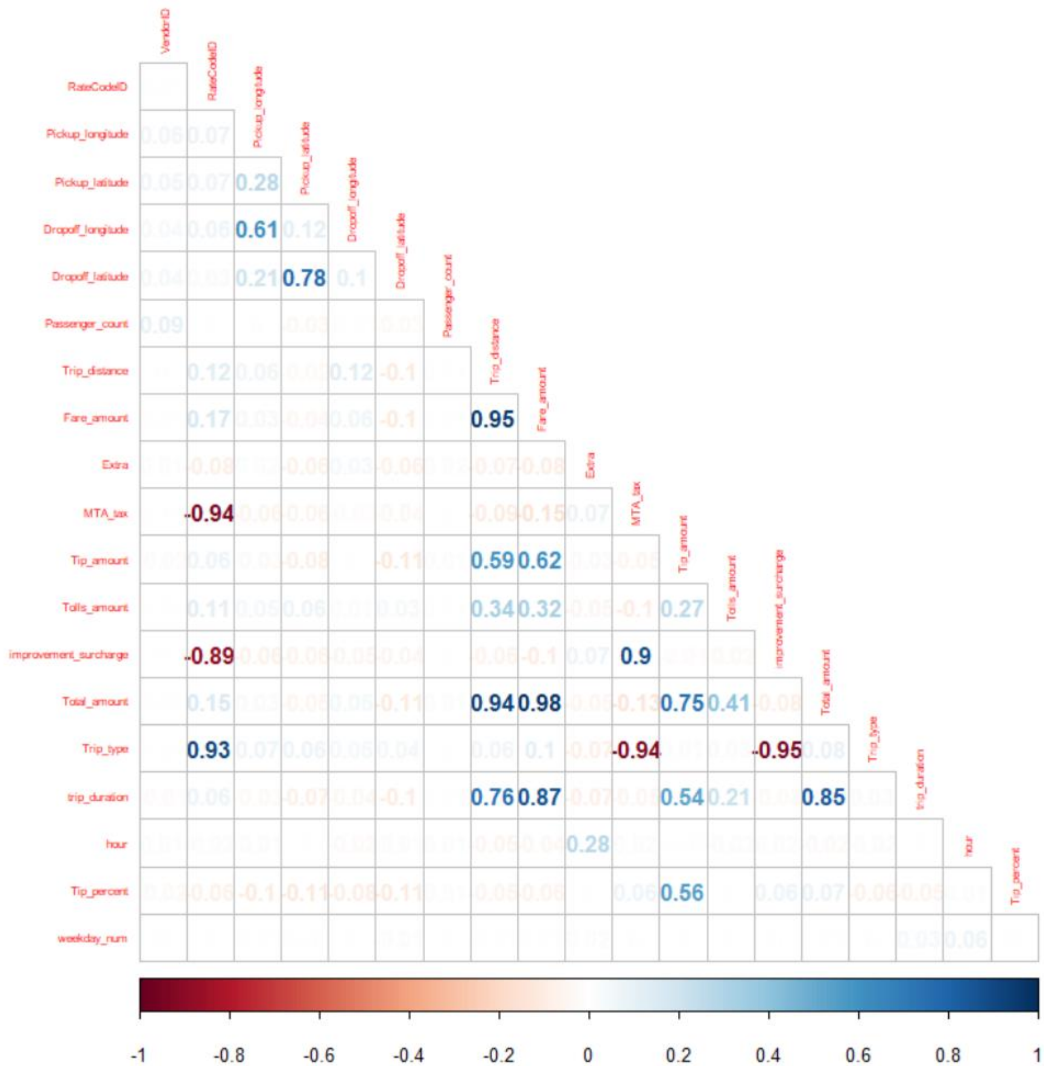
```
# Data Cleaning

neg.vars <- c('Fare_amount','Extra','improvement_surcharge','Total_amount','MTA_tax','Tip_amount')

df[df$Fare_amount < 0,][neg.vars] <- df[df$Fare_amount < 0,][neg.vars]*-1

# Removing Fare amount less than 2.5
df <- subset(df,df[,'Fare_amount'] >= 2.5)

# Distances greater then 0
df <- subset(df,df[,11] > 0)


# Trip Durations greater then 4 hrs
df <- subset(df,df[,21] < (4*3600))

# Remove trip Durations less then 2min
df <- subset(df,df[,21] > (2*60))


# Set coordinates outside of NYC bounding box to NA

nw <- list(lat = 40.917577, lon = -74.259090)
se <- list(lat = 40.477399, lon = -73.700272)
ind <- which(df$Dropoff_longitude < nw$lon | df$Dropoff_longitude > se$lon)
df$Dropoff_longitude[ind] <- NA
```

```r
ind <- which(df$Pickup_longitude < nw$lon | df$Pickup_longitude > se$lon)
df$Pickup_longitude[ind] <- NA
ind <- which(df$Dropoff_latitude < se$lat | df$Dropoff_latitude > nw$lat)
df$Dropoff_latitude[ind] <- NA
ind <- which(df$Pickup_latitude < se$lat | df$Pickup_latitude > nw$lat)
df$Pickup_latitude[ind] <- NA


df <- df[complete.cases(df),]

# passengers < 7
df <- subset(df,df[,10] < 7)

# Replace passengers with zero count with the median value i.e. 1
df$Passenger_count[df$Passenger_count == 0] <- 1

# Since payment types other than credit card have zero tips 99% of the time
df <- subset(df, Payment_type == 1)
```

Collinearity plot

Looking at relations we might have missed

```r
nums <- sapply(df, is.numeric) #taking only numeric class
num.df <- df[ , nums]
corrplot(cor(num.df), method = "number",tl.cex = 0.5,type="lower",diag=FALSE)
```


VendorID
RateCodeID
Pickup_longitude
Pickup_latitude
Dropoff_longitude
Dropoff_latitude
Passenger_count
Trip_distance
Fare_amount
Extra
MTA_tax
Tip_amount
Tolls_amount
improvement_surcharge
Total_amount
Trip_type
trip_duration
hour
Tip_percent
weekday_num
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1

Relations

According to this correlation plot,

1) The trip distance and fare amount are positively corelated, and so is the total amount. It's understandable as higher the distance travelled, the higher the fare amounts to.

2) Surprisingly the tip amount isn't that highly related to the fare amount, this might be because of the number of tips recorded for cash type payment is 0.

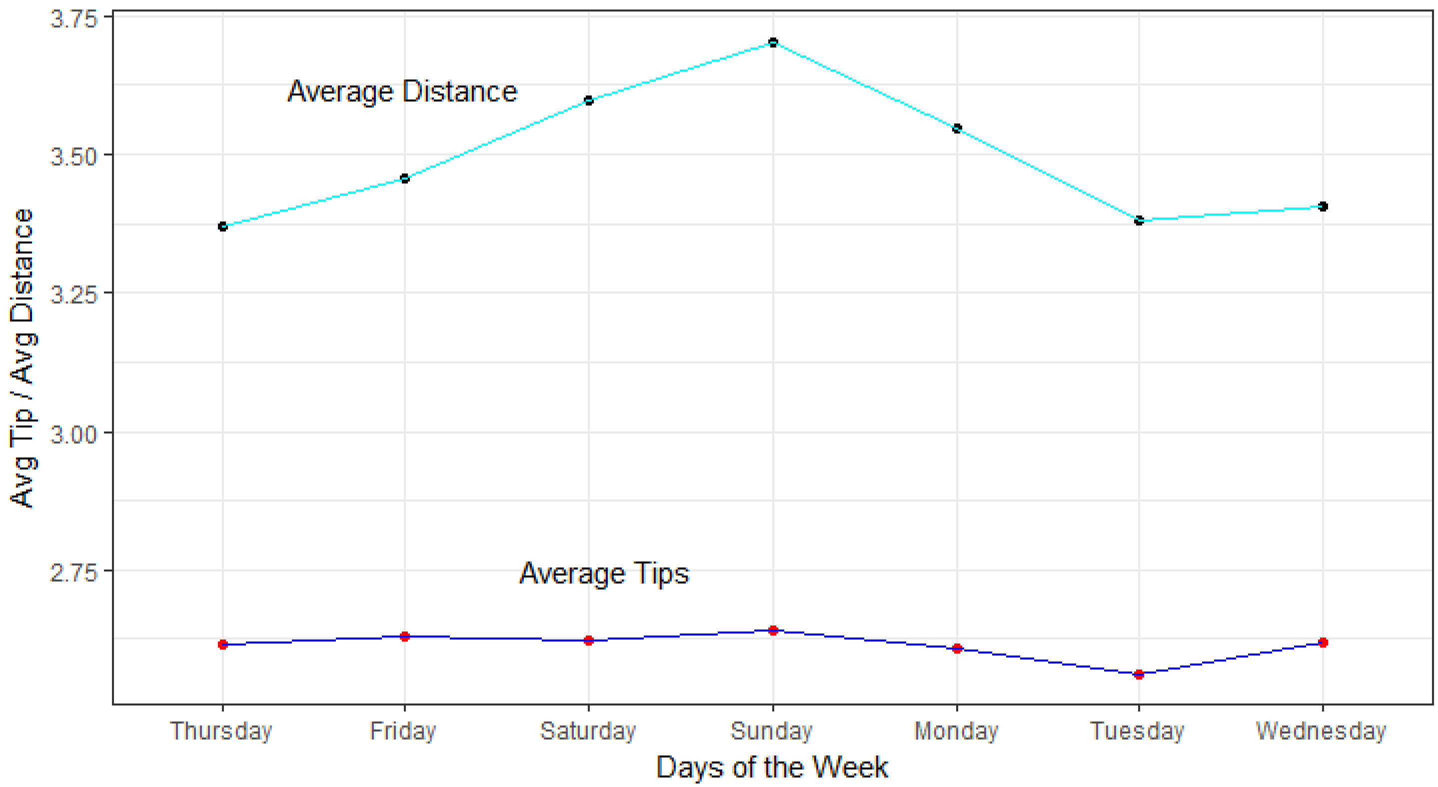
3) The derived variable trip duration has high correlation coefficients with trip distance, fare, total and tip amounts.

4) The trip type is highly related to the ratecode id and the other taxes, which makes perfect sense as pointed out earlier.

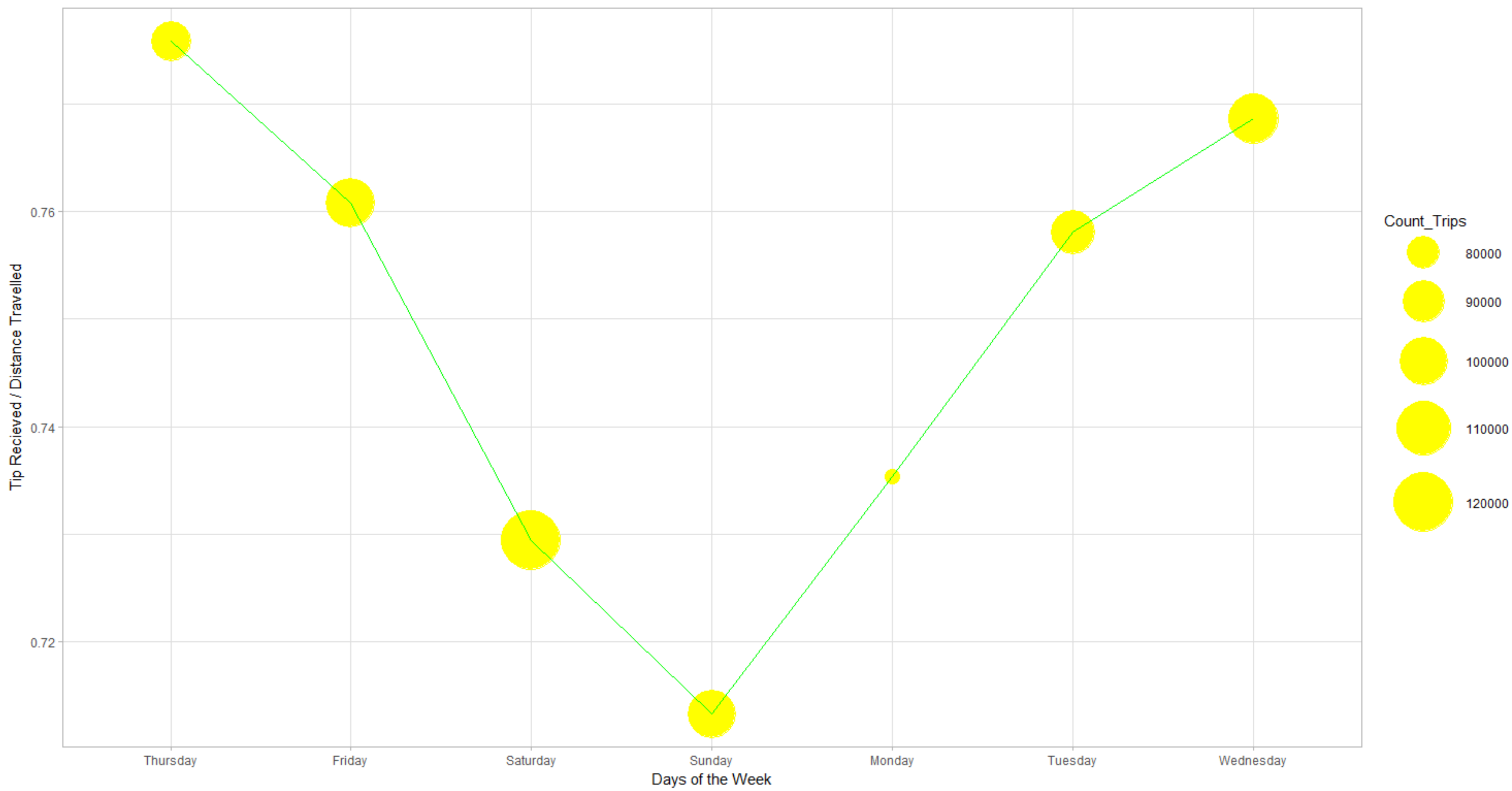
Feature engineering: New derived features

```
df$Tip_percent <- (df$Tip_amount/df$Total_amount)*100

clean_datetime <- df %>%
  mutate(lpep_pickup_datetime = ymd_hms(lpep_pickup_datetime)) %>%
  mutate(Lpep_dropoff_datetime = ymd_hms(Lpep_dropoff_datetime)) %>%
  mutate(weekday_pickup = weekdays(lpep_pickup_datetime)) %>%
  mutate(weekday_dropoff= weekdays(Lpep_dropoff_datetime))%>%
  mutate(hpick = hour(lpep_pickup_datetime)) %>%
  mutate(date1 = date(lpep_pickup_datetime))

#from the above code we get derived features such as weekday pickup, hour of pickup

temp <- clean_datetime %>%
  group_by(weekday_pickup) %>%
  summarize(Count_Trips = n(), avg_dist = mean(Trip_distance),
            avg_passengers = mean(Passenger_count),
            avg_price = mean(Total_amount),
```

```r
            avg_Tip = mean(Tip_amount),
            Total_tip = sum(Tip_amount))

temp$ratio <- temp$avg_Tip/temp$avg_dist

temp

## # A tibble: 7 x 8
##   weekday_pickup Count_Trips avg_dist avg_passengers avg_price  avg_Tip
##            <chr>       <int>    <dbl>          <dbl>     <dbl>    <dbl>
## 1       Thursday       88274 3.370301       1.353184  18.05979 2.614853
## 2         Friday      101211 3.457057       1.359655  18.27459 2.630239
## 3       Saturday      120402 3.598627       1.399802  18.11773 2.624716
## 4         Sunday       99571 3.703381       1.397706  18.11053 2.641506
## 5         Monday       72832 3.547673       1.364538  17.98952 2.608715
## 6        Tuesday       93683 3.381342       1.348153  17.76378 2.563375
## 7      Wednesday      102930 3.406467       1.354892  18.08792 2.618645
## # ... with 2 more variables: Total_tip <dbl>, ratio <dbl>
```

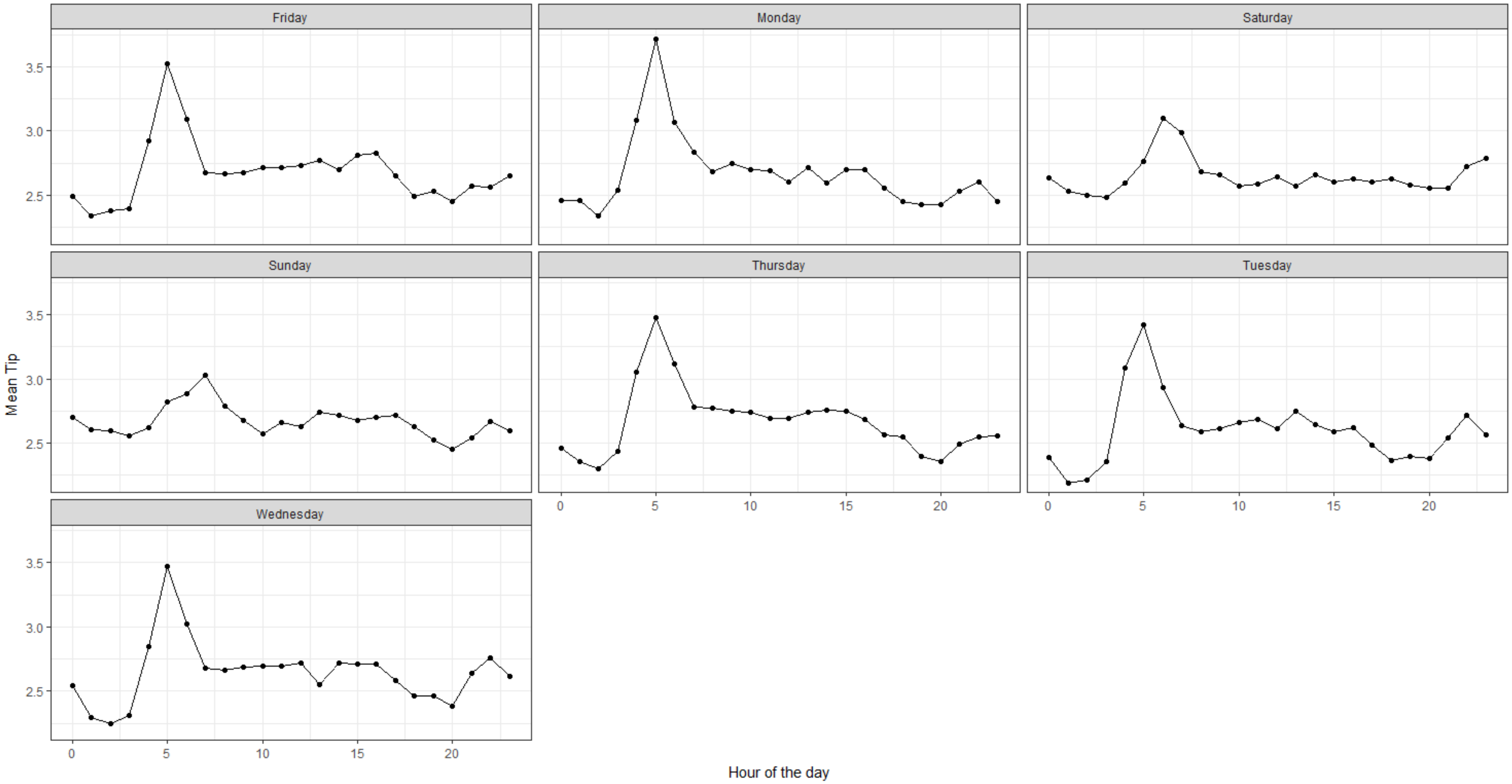


From the above graph and table above, we notice the maximum distance was on Sunday and the maximum average tip/ total tip amount received was on Saturday followed by Friday. There's a straight dip in tips on Monday. This can be accounted as people like to go out on weekends and won't hesitate to spend a little more, on the other hand they wouldn't do the same on a weekday. Let's compare the ratios of the distance to tip.



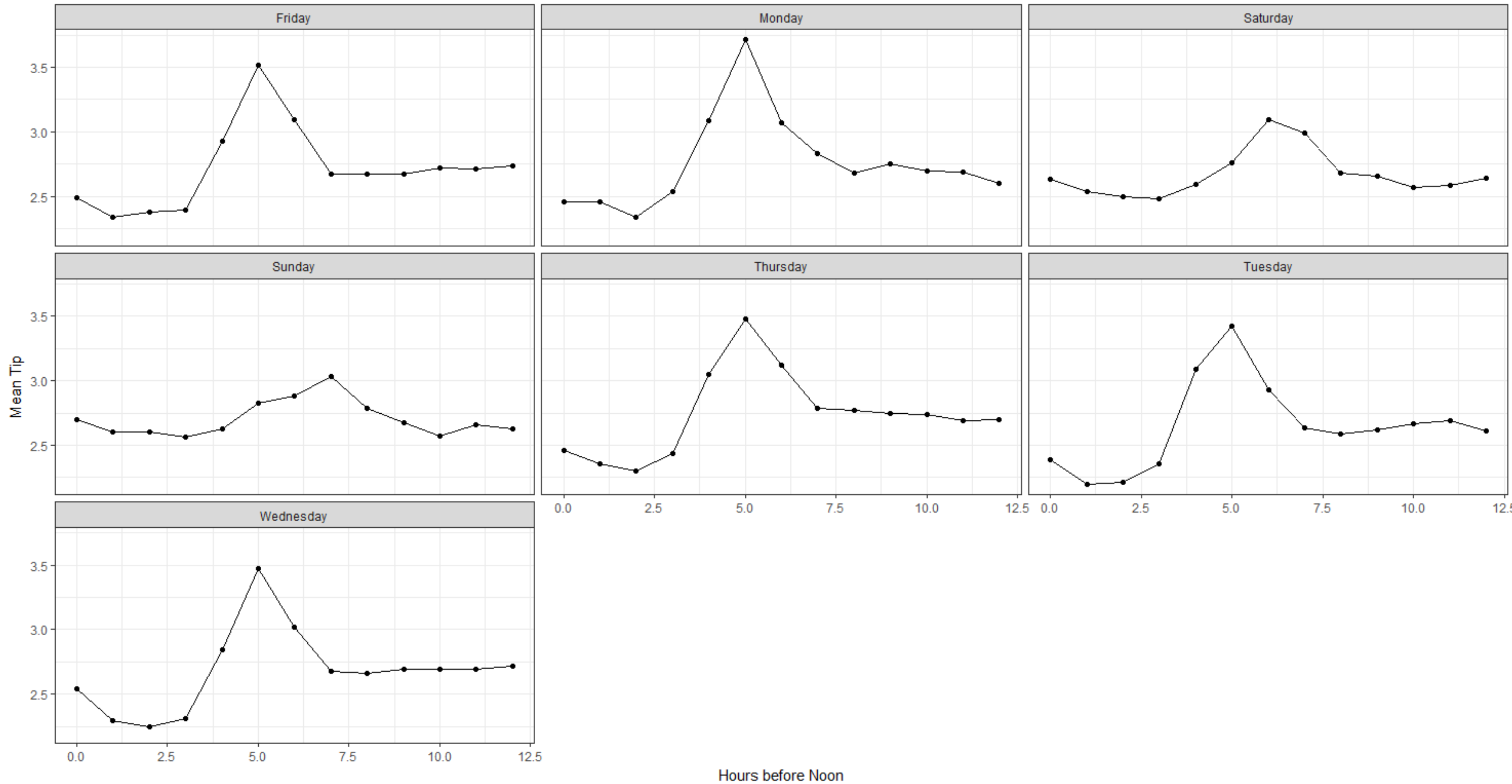
There’s a massive dip on Sunday’s, which drivers might be unaware of. They don’t get a good return on their trips on Sunday. The intensity of each point describes the average number of rides given at that day. From the graph, we see that Thursday and Wednesday have the best ratio’s, in turn giving the best returns, although it doesn’t show us the total tips received that day. If one looks to save time and wants the best on its return, they should work through the weekdays (Wednesday, Thursday); else, if the driver has time and wants to earn a bit more, it would be smarter to work on weekends as the ratio difference between Thursday and Sunday isn’t that high.

Combining this knowledge and our previous plot on tip received by the hour, we can get a detailed explanation on when and what time can one maximize their tips.



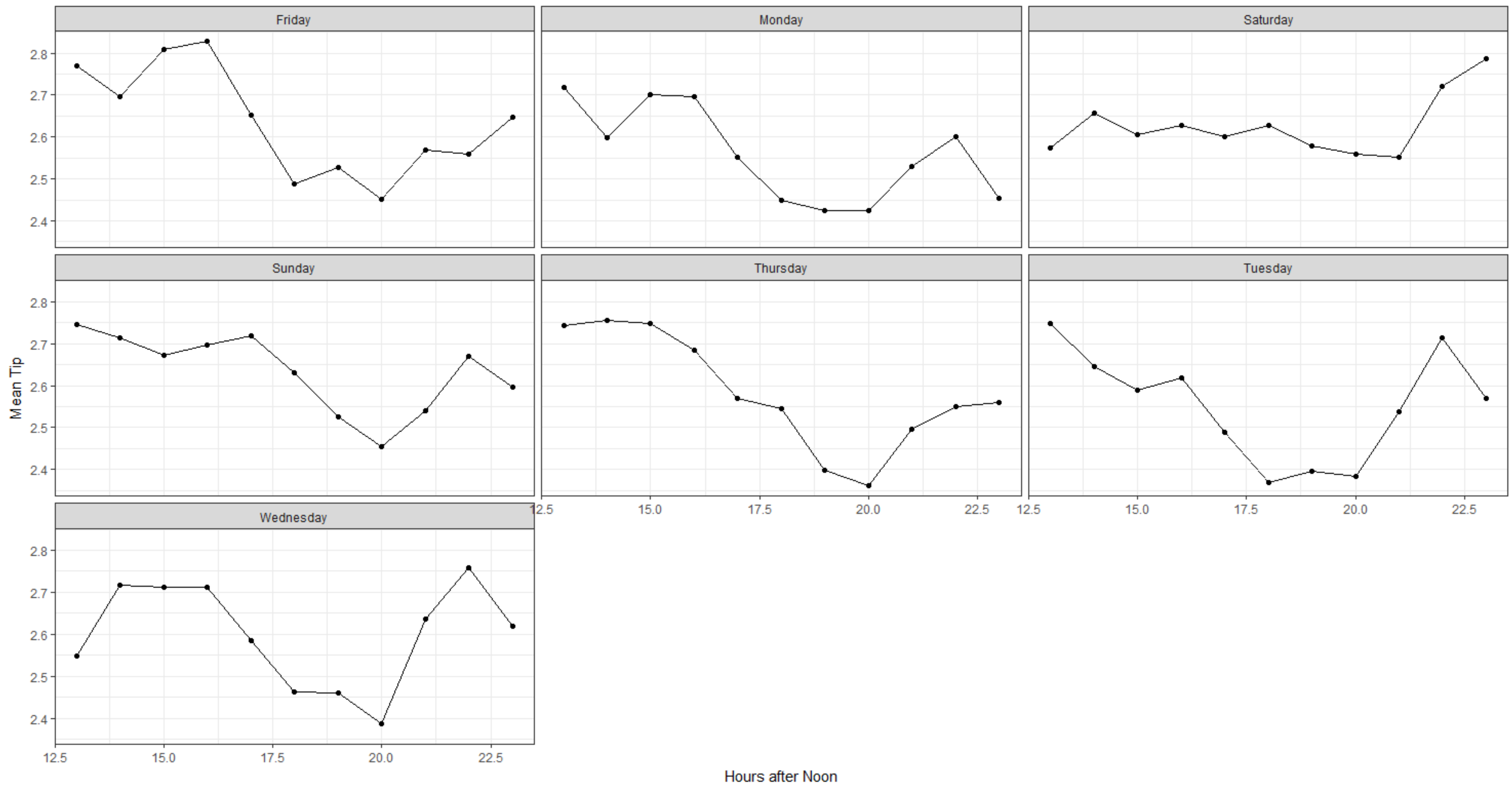
A closer investigation into these plots

Tips received before Noon during the week



As it can be see, morning 5 am can be a good time to earn some tips on days like Friday, Monday, Thursday, Tuesday and Wednesday.

Tips received after Noon during the week



From this we can confirm the best time to work at night after 9 pm is on Saturday, followed by Tuesday and Wednesday (Surprising?)

Data Modeling

Tejas Bawaskar

Model Building

```
# Selecting the top variables from the list by using p-value.
summary(lm(Tip_percent ~.,data=clean_datetime[sample(nrow(clean_datetime),100000) ,]))
##
## Call:
## lm(formula = Tip_percent ~ ., data = clean_datetime[sample(nrow(clean_datetime),
##     1e+05), ])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -423.25   -0.98    0.92    2.15   42.06
##
## Coefficients: (1 not defined because of singularities)
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)           -2.895e+02  3.545e+01  -8.168 3.17e-16 ***
## VendorID              -9.013e-02  3.700e-02  -2.436   0.0148 *
## Store_and_fwd_flagY    2.408e-01  2.248e-01   1.071   0.2842
## RateCodeID            -1.263e-01  1.661e-01  -0.761   0.4469
## Pickup_longitude      -3.949e+00  5.038e-01  -7.839 4.59e-15 ***
## Pickup_latitude       -3.076e+00  4.486e-01  -6.856 7.11e-12 ***
## Dropoff_longitude     -4.665e+00  3.987e-01 -11.701  < 2e-16 ***
## Dropoff_latitude      -5.153e+00  4.529e-01 -11.379  < 2e-16 ***
## Passenger_count        2.867e-02  1.440e-02   1.991   0.0465 *
## Trip_distance         -9.614e-02  1.794e-02  -5.359 8.40e-08 ***
## Fare_amount           -3.608e-01  7.732e-03 -46.658  < 2e-16 ***
## Extra                 -5.714e-01  4.330e-02 -13.195  < 2e-16 ***
## MTA_tax                3.797e-01  1.441e+00   0.263   0.7922
```

```
## Tip_amount             2.543e+00  7.218e-03 352.274  < 2e-16 ***
## Tolls_amount          -3.274e-01  1.611e-02 -20.322  < 2e-16 ***
## improvement_surcharge  4.016e+00  1.927e+00   2.084   0.0372 *
## Total_amount                  NA         NA      NA       NA
## Trip_type              4.670e-01  9.142e-01   0.511   0.6095
## trip_duration         -8.376e-04  4.998e-05 -16.758  < 2e-16 ***
## hour                   1.088e-02  2.271e-03   4.791 1.66e-06 ***
## weekday_num            7.976e-03  7.332e-03   1.088   0.2767
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.723 on 99980 degrees of freedom
## Multiple R-squared:  0.5668, Adjusted R-squared:  0.5667
## F-statistic:  6886 on 19 and 99980 DF,  p-value: < 2.2e-16
```

```
# Check for multicollinearity
# Check for alias
alias(lm(Tip_percent ~.,data=clean_datetime))
```

```
## Model :
## Tip_percent ~ VendorID + Store_and_fwd_flag + RateCodeID + Pickup_longitude +
##     Pickup_latitude + Dropoff_longitude + Dropoff_latitude +
##     Passenger_count + Trip_distance + Fare_amount + Extra + MTA_tax +
##     Tip_amount + Tolls_amount + improvement_surcharge + Total_amount +
##     Trip_type + trip_duration + hour + weekday_num
##
## Complete :
##              (Intercept) VendorID Store_and_fwd_flagY RateCodeID
## Total_amount 0           0        0                   0
##              Pickup_longitude Pickup_latitude Dropoff_longitude
## Total_amount 0                0               0
##              Dropoff_latitude Passenger_count Trip_distance Fare_amount
## Total_amount 0                0               0             1
##              Extra MTA_tax Tip_amount Tolls_amount improvement_surcharge
## Total_amount 1     1       1          1            1
```

```
##              Trip_type trip_duration hour weekday_num
## Total_amount 0         0             0    0

# After removing the alias, we check for vif
vif(lm(Tip_percent ~.,data=clean_datetime[,-c(16)]))

##              VendorID    Store_and_fwd_flag            RateCodeID
##              1.033153              1.017930             11.058885
##      Pickup_longitude       Pickup_latitude      Dropoff_longitude
##              1.745318              2.783382              1.745057
##      Dropoff_latitude       Passenger_count         Trip_distance
##              2.692893              1.010608             14.545188
##           Fare_amount                 Extra               MTA_tax
##             26.765581              1.107118             12.383903
##            Tip_amount          Tolls_amount improvement_surcharge
##              1.651563              1.209682             11.172619
##             Trip_type         trip_duration                  hour
##             20.627401              5.828800              1.094788
##           weekday_num
##              1.007344

# A vif value above 1 indicates the predictors are slightly correlated. A vif between 5 and 10 indicates
high correlation that maybe problematic. And anything above 10, it can be concluded that the regression
coefficients aren't correct/poorly estimated. To solve it standardizing the continuous predictors can be
used, if not, we would have to remove the highly correlated variables.

# From the variables we can notice that Trip distance, fare amount, mta tax, trip type and improvement
surcharge have high vif's. This is due to the fact that these variables are highly correlated and can be
seen from the correlation plot.

# Mean absolute error
MAE <- function(actual, predict){
error <- abs(actual - predict)
return(mean(error))
}
```

```r
#Root Mean squared error
RMSE <- function(actual, predict){
sqrt(mean((actual - predict)^2))
}

# Accuracy
accuracy <- function(actual, predict){
    count = 0
    ncount = 0
    error = abs(actual - predict)
      for( i in 1:length(actual)){
        if (error[i] <= 1) {count = count + 1}
        else( ncount = ncount + 1)
        }
    acc <- 100*count/(ncount + count)
  return(acc)
}
```

From this we notice that the best variables that have p-values less than 0.05 are; VendorID, Pickup_longitude, Pickup_latitude, Dropoff_longitude, Dropoff_latitude, Passenger_count, Trip_distance, Fare_amount, Extra, MTA_tax, Tip_amount, Tolls_amount

Creating training & validation subsets

Since there are nearly 678000 observations in this dataset, it would make sense to randomly subset a sample of these observations for our model.

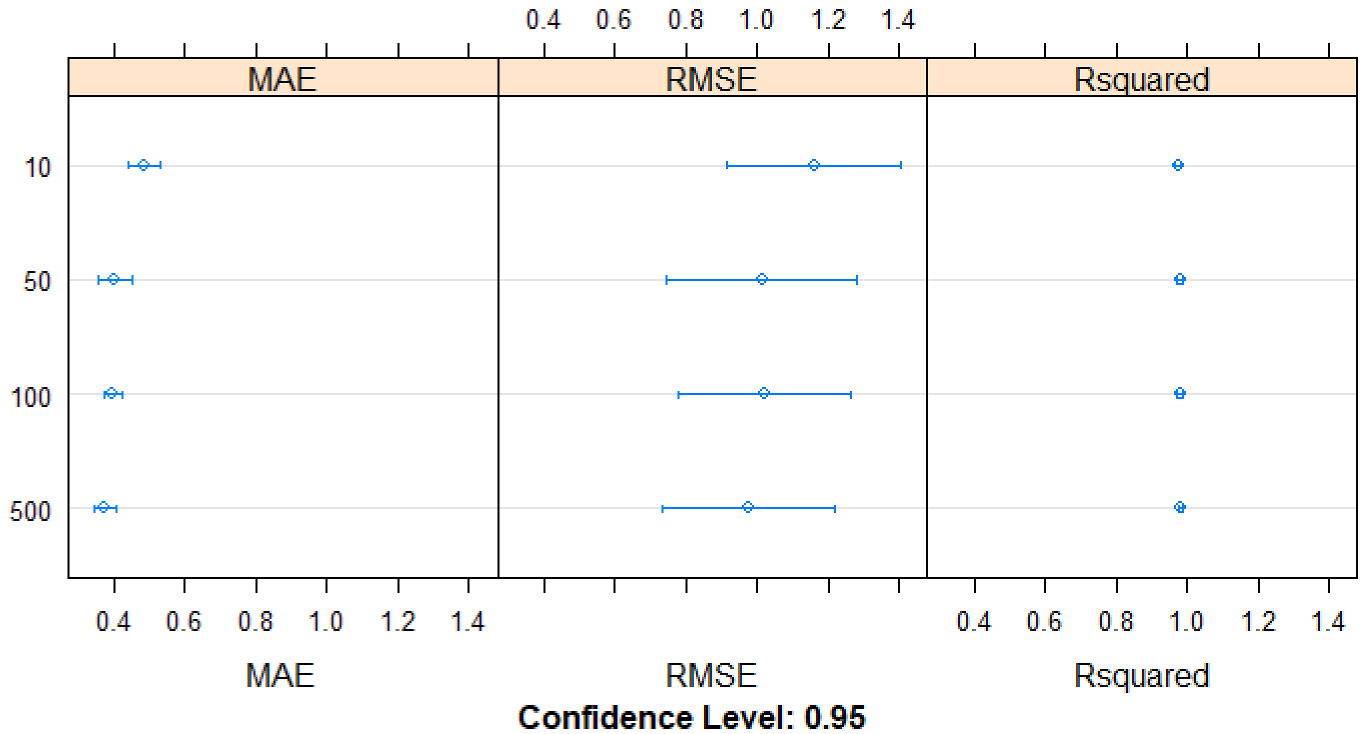
```r
set.seed(789)

#using just a sample of this to run the various models

clean_datetime.test <- clean_datetime[sample(nrow(clean_datetime), 10000), ]
clean_datetime.train <- clean_datetime[sample(nrow(clean_datetime), 10000), ]
clean_datetime.val <- clean_datetime[sample(nrow(clean_datetime), 2000), ]
```

Construction of related models and tuning them subsequently

Construction of stacked ensemble model (Random Forest)

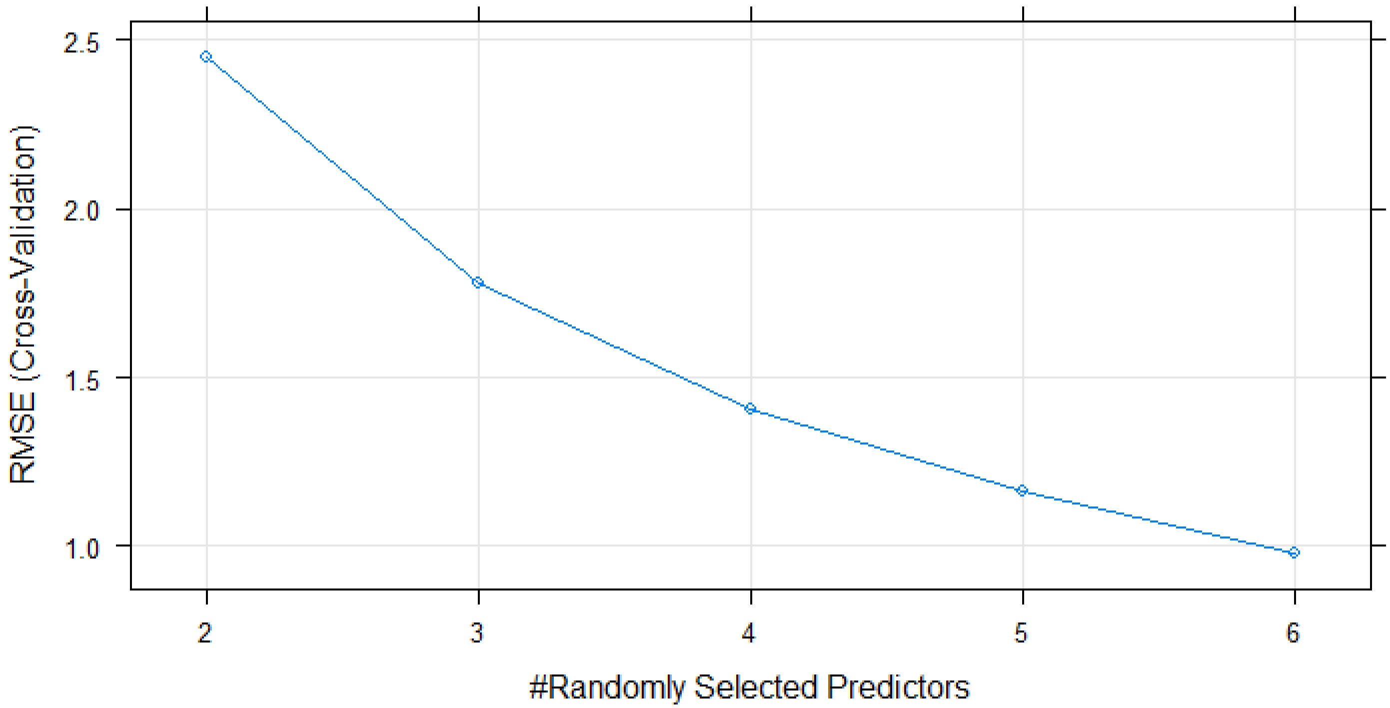
```
# Model 1 Random forest

# Grid/Manual Search
control <- trainControl(method="cv", number=5, search="grid")
x <- floor(sqrt(ncol(clean_datetime)))
tunegrid <- expand.grid(.mtry=c(x-2):(x+2))
modellist <- list()


for (ntree in c(10,50,100,500)) {
    set.seed(21)
    fit <- train(Tip_percent ~ VendorID + Pickup_longitude + Pickup_latitude + Dropoff_longitude + 
Dropoff_latitude + Passenger_count + Trip_distance + Fare_amount + Extra + MTA_tax + Tip_amount + 
Tolls_amount
                 , data=clean_datetime.train
                 , method="rf"
                 , metric='RMSE'
                 , tuneGrid=tunegrid
                 , trControl=control
                 , ntree=ntree)

    key <- toString(ntree)
    modellist[[key]] <- fit
}

# compare results
results <- resamples(modellist)
summary(results)

##
## Call:
## summary.resamples(object = results)
```

```
##
## Models: 10, 50, 100, 500
## Number of resamples: 5
##
## MAE
##          Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## 10  0.4297521 0.4664447 0.4748931 0.4802251 0.4885516 0.5414842    0
## 50  0.3881650 0.3926714 0.3967072 0.3964185 0.3968663 0.4076827    0
## 100 0.3721884 0.3776052 0.3795528 0.3836523 0.3907955 0.3981197    0
## 500 0.3536920 0.3595947 0.3732355 0.3692681 0.3743907 0.3854277    0
##
## RMSE
##          Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## 10  0.9870190 1.204684 1.223375 1.238236 1.283352 1.492750    0
## 50  0.8292506 1.074813 1.086968 1.087844 1.144499 1.303689    0
## 100 0.8242699 1.065944 1.072602 1.082802 1.126703 1.324491    0
## 500 0.7762704 1.042938 1.044375 1.054233 1.095353 1.312228    0
##
## Rsquared
##          Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## 10  0.9574350 0.9694954 0.9715551 0.9703449 0.9723520 0.9808869    0
## 50  0.9677749 0.9763544 0.9777504 0.9775200 0.9788675 0.9868526    0
## 100 0.9667102 0.9771491 0.9782160 0.9777438 0.9795662 0.9870774    0
## 500 0.9674711 0.9786173 0.9793126 0.9789165 0.9806515 0.9885303    0

dotplot(results)
```



MAE
RMSE
Rsquared
10
50
100
500
0.4
0.6
0.8
1.0
1.2
1.4
Confidence Level: 0.95


```r
plot(fit)
```




```
# Here it seems that once the ntree crosses 100, it would tend to overfit the data which is undesirable.
The number of variables selected is 6, which I believe is the best combination of mtry and ntree.

rf.model <- randomForest(Tip_percent ~ VendorID + Pickup_longitude + Pickup_latitude + Dropoff_longitude +
Dropoff_latitude + Passenger_count + Trip_distance + Fare_amount + Extra + MTA_tax + Tip_amount +
Tolls_amount
                      , data = clean_datetime.train
                      , ntree = 100
                      , mtry = 6
                      , replace = TRUE
                      , nodesize = 5)
```

```r
importance(rf.model)
```

```
##                    IncNodePurity
## VendorID                216.4473
## Pickup_longitude       4931.1488
## Pickup_latitude        5045.0790
## Dropoff_longitude      3314.4630
## Dropoff_latitude       5909.0924
## Passenger_count         340.5997
## Trip_distance         13129.2760
## Fare_amount           26545.8737
## Extra                   977.7810
## MTA_tax                 122.1303
## Tip_amount           444949.1398
## Tolls_amount            788.8441
```

Linear Model

```r
# Model 2 Linear Regression


# To solve the multicollinearity issue, standardizing the continuous predictors can be used, but after trying it (standardizing/normalizing) the vif's values were unaffected. While building the linear model, we would have to subset variables that are highly related.

lm.model <- lm(Tip_percent ~ Dropoff_longitude + Dropoff_latitude + Fare_amount + Extra + MTA_tax + Tip_amount + Tolls_amount, data = clean_datetime.train)

summary(lm.model)
```

```
##
## Call:
## lm(formula = Tip_percent ~ Dropoff_longitude + Dropoff_latitude +
##     Fare_amount + Extra + MTA_tax + Tip_amount + Tolls_amount,
##     data = clean_datetime.train)
```

```
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -174.360   -1.093    0.063    2.227   35.971
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)       -246.66346   82.82800  -2.978  0.00291 **
## Dropoff_longitude   -7.79159    0.96011  -8.115 5.41e-16 ***
## Dropoff_latitude    -7.75099    0.88910  -8.718  < 2e-16 ***
## Fare_amount         -0.40079    0.00619 -64.747  < 2e-16 ***
## Extra               -0.55662    0.13333  -4.175 3.01e-05 ***
## MTA_tax              0.69206    1.22284   0.566  0.02145 ***
## Tip_amount           2.28815    0.02124 107.731  < 2e-16 ***
## Tolls_amount        -0.27960    0.04936  -5.665 1.51e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.77 on 9992 degrees of freedom
## Multiple R-squared:  0.5509, Adjusted R-squared:  0.5506
## F-statistic:  1751 on 7 and 9992 DF,  p-value: < 2.2e-16

# All vif' are below 5
vif(lm.model)

## Dropoff_longitude  Dropoff_latitude       Fare_amount             Extra
##          1.021660          1.033235          1.580934          1.018123
##           MTA_tax        Tip_amount      Tolls_amount
##          1.044615          1.455963          1.131968

# this is the best model for linear regression

# -246.6 – 7.79*Dropoff_longitude - 7.75*Dropoff_latitude – 0.40*Fare_amount – 0.55*Extra + 0.69*MTA_tax +
2.288*Tip_amount – 0.2796*Tolls_amount
```

SVM Model

```r
# Model 3 SVM
# Using grid search technique, we can find the optimal cost and gamma values
obj <- tune(svm, Tip_percent~VendorID + Pickup_longitude + Pickup_latitude +
    Dropoff_longitude + Dropoff_latitude + Passenger_count +
    Trip_distance + Fare_amount + Extra + MTA_tax + Tip_amount +
    Tolls_amount, data = clean_datetime.train,
            validation.x = clean_datetime.val,
              ranges = list(gamma = 2^(-1:1), cost = 2^(2:4)),
              tunecontrol = tune.control(sampling = "fix")
             )

obj

##
## Parameter tuning of 'svm':
##
## - sampling method: fixed training/validation set
##
## - best parameters:
##  gamma cost
##    0.5   16
##
## - best performance: 6.671177

# Model tuned
# Since we have high multicollinearity among our features, we will be using the linear kernel. Also
selecting the features that have p-values less than 0.05
svm.model <- svm(Tip_percent ~ VendorID + Pickup_longitude + Pickup_latitude +
    Dropoff_longitude + Dropoff_latitude + Passenger_count +
    Trip_distance + Fare_amount + Extra + MTA_tax + Tip_amount +
    Tolls_amount, kernel="linear", cost=16, gamma=0.5, clean_datetime.train)
```

Rpart Model

```r
# Model 4 Rpart
# Recursive Partitioning and Regression Trees (rpart) model

rpartout <- rpart(Tip_percent ~ VendorID + Pickup_longitude + Pickup_latitude +
    Dropoff_longitude + Dropoff_latitude + Passenger_count +
    Trip_distance + Fare_amount + Extra + MTA_tax + Tip_amount +
    Tolls_amount
      ,data = clean_datetime.train
      ,method = "poisson")
```

Evaluation with k-fold cross-validation

```r
fitControl <- trainControl(method = "cv" #Cross Validation
                           , number = 10
                           , search = "random")

lm.kmodel <- train(Tip_percent ~ Dropoff_longitude + Dropoff_latitude + Fare_amount + Extra + MTA_tax +
Tip_amount + Tolls_amount, data=clean_datetime.train, trControl=fitControl, method="lm")

lm.kpredict <- predict(lm.kmodel,clean_datetime.test, se.fit = TRUE
,interval = "confidence"
,level = 0.95)
```

Comparing the MSE, RMSE and Accuracy across all models

```
compare.model<-data.frame(name = c("RPART","Linear Regression","SVM","Linear
Regression(Kfolds)","RandomForest")
                          ,MAE = c(rpart.mae, lm.mae, svm.mae, lmk.mae, rf.mae)
                          ,RMSE = c(rp.mse, lm.mse, svm.mse, lmk.mse, rf.mse)
                          ,Accuracy = c(rp.acc, lm.acc, svm.acc, lmk.acc, rf.acc))

compare.model

##                         name       MAE      RMSE Accuracy
## 1                      RPART 1.6747761 2.9550761    64.04
## 2          Linear Regression 3.1008717 4.4360816    22.62
## 3                        SVM 2.3313371 9.9711044    56.75
## 4 Linear Regression(Kfolds) 3.0955803 8.3443353    22.62
## 5               RandomForest 0.3475145 0.8815251    92.32

#From this we can see Random Forest is the best
```

Comparing all models, we find that random forest is the best. The accuracy determined here is basically checking if the predicted value is within 1% range of the actual value. Random forest gives the best response followed by RPART and SVM.