CitiBike Trip and Weather Data Analysis

Jash Gaglani (jg1700)

Overview

This is an analysis of the NYC CitiBike dataset to view how the different variables affect the duration of the bike rides and how different scenarios affect number of bike rides taken in a time period. I clubbed the 2019 Citibike dataset with weather data for that time period to augment our dataset. I also built a couple of models to predict the ride duration as it is directly proportional to how much a ride will cost. With such intelligence, a company can be better positioned to determine actions that can increase it's revenue stream.

With the joined datasets these are the hypothesis that I was able to test * Number of rides will go down during the winter months. * Number of rides go down during odd hours like after midnight and before dawn. * Number of rides taken will significantly increase on weekends as compared to weekdays. * Young riders will have longer bike ride durations and ride more frequently. * Yearly subscribers will have more number of rides and higher ride durations.

I recommend reading till page 13 and skipping to the last page for conclusion.

Data Processing

Data sources

We obtained data from the following sources:

1. CitiBike trip dataset

Since 2013, a shared bicycle system known as CitiBike has been available in New York City. CitiBike makes a vast amount of data available regarding system usage. The available usage data provides a wealth of information I used to find valuable trends.

For each month since the system's inception, there is a file containing details of (almost) every trip. There are currently 108 monthly data files for the New York City bikeshare system, spanning July 2013 through December 2019. I used data from the year 2019 as I wanted to do it for a time period before covid.

Each file contains a line for every trip. The number of trips per month varies from as few as 200,000 during winter months in the system's early days to more than 2 million trips in summer. Because of the computational limitations which this presented, I created samples of 1/1000. The samples were created non-deterministically, by randomly selecteing 'r nrow(file)/1000' from the file.

The column names for this are pretty much self explanatory.

```
citibike_2019 <- read.csv("citibike_2019.csv")
citibike_2019 <- citibike_2019[sample(nrow(citibike_2019), 20000),]
head(citibike_2019)
```

```
##        tripduration               starttime                stoptime
## 60660            90 2019-05-15 15:29:12.3040 2019-05-15 15:30:42.6320
## 93128           701 2019-07-02 08:41:47.3420 2019-07-02 08:53:28.8450
## 78548           859 2019-06-06 10:01:34.8520 2019-06-06 10:15:53.8730
## 114392          645 2019-08-18 15:34:33.9960 2019-08-18 15:45:19.2570
## 77740           435 2019-06-02 23:23:45.2940 2019-06-02 23:31:00.5320
## 164887          143 2019-10-05 21:08:12.9780 2019-10-05 21:10:36.9100
##        start.station.id          start.station.name start.station.latitude
## 60660               258 DeKalb Ave & Vanderbilt Ave               40.68941
## 93128               445          E 10 St & Avenue A               40.72741
## 78548              3255             8 Ave & W 31 St               40.75059
## 114392              368          Carmine St & 6 Ave               40.73039
## 77740               474             5 Ave & E 29 St               40.74517
## 164887              347 Greenwich St & W Houston St               40.72885
##        start.station.longitude end.station.id            end.station.name
## 60660                -73.96885            241 DeKalb Ave & S Portland Ave
## 93128                -73.98142            402          Broadway & E 22 St
## 78548                -73.99468            383  Greenwich Ave & Charles St
## 114392               -74.00215            328     Watts St & Greenwich St
## 77740                -73.98683            519       Pershing Square North
## 164887               -74.00859            127       Barrow St & Hudson St
##        end.station.latitude end.station.longitude bikeid   usertype birth.year
## 60660              40.68981             -73.97493  16179 Subscriber       1960
## 93128              40.74034             -73.98955  30170 Subscriber       1979
## 78548              40.73524             -74.00027  25541 Subscriber       1949
## 114392             40.72406             -74.00966  35038 Subscriber       1986
## 77740              40.75187             -73.97771  30977 Subscriber       1994
## 164887             40.73172             -74.00674  38358 Subscriber       1984
##        gender
## 60660       1
## 93128       1
## 78548       1
## 114392      2
## 77740       1
## 164887      2
```

2. Central Park daily weather data

I obtained historical weather information for the year 2019 from the NCDC (National Climatic Data Center) by submitting an online request to https://www.ncdc.noaa.gov/cdo-web/search. Although the weather may vary slightly within New York City, I used just the data associated with the Central Park observations as proxy for the entire city's weather.

I believe that the above data provides a reasonable representation of the target population (all CitiBike rides) and the citywide weather.

```
weather <- read.csv("2812766.csv",header=FALSE)
names(weather)<-weather[1,]
weather<-weather[-1,]
weather<-subset(weather[weather$NAME == "NY CITY CENTRAL PARK, NY US" & year(weather$DATE) == 2019,],)
attr_indexes <- names(weather) %>% grep("ATTR",x = .)
weather <- weather[,-attr_indexes]
weather <- weather[,-c(1:5)]
weather<-weather %>%mutate(across(c(where(is.character), -DATE), as.numeric))
```

```
weather$DATE<-as.Date(weather$DATE)
weather <- weather %>% select("DATE","PRCP","SNOW","SNWD","TMAX","TMIN","WT01","WDF2","WDF5","WSF2","WS
weather[is.na(weather)] <- 0
head(weather)
```

```
##             DATE PRCP SNOW SNWD TMAX TMIN WT01 WDF2 WDF5 WSF2 WSF5 WT08
## 58385 2019-01-01 0.06    0    0   58   39    1    0    0    0    0    0
## 58386 2019-01-02 0.00    0    0   40   35    0    0    0    0    0    0
## 58387 2019-01-03 0.00    0    0   44   37    0    0    0    0    0    0
## 58388 2019-01-04 0.00    0    0   47   35    0    0    0    0    0    0
## 58389 2019-01-05 0.50    0    0   47   41    1    0    0    0    0    0
## 58390 2019-01-06 0.00    0    0   49   31    0    0    0    0    0    0
```

PRCP = Precipitation (mm or inches as per user preference, inches to hundredths on Daily Form pdf file) SNOW = Snowfall (mm or inches as per user preference, inches to tenths on Daily Form pdf file) SNWD = Snow depth (mm or inches as per user preference, inches on Daily Form pdf file WT01 = Fog, ice fog, or freezing fog (may include heavy fog) WT08 = Smoke or haze WDF2 = Direction of fastest 2-minute wind (degrees) WDF5 = Direction of fastest 5-second wind (degrees) WSF2 = Fastest 2-minute wind speed (miles per hour or meters per second as per user preference WSF5 = Fastest 5-second wind speed (miles per hour or meters per second as per user preference)

Data Exploration

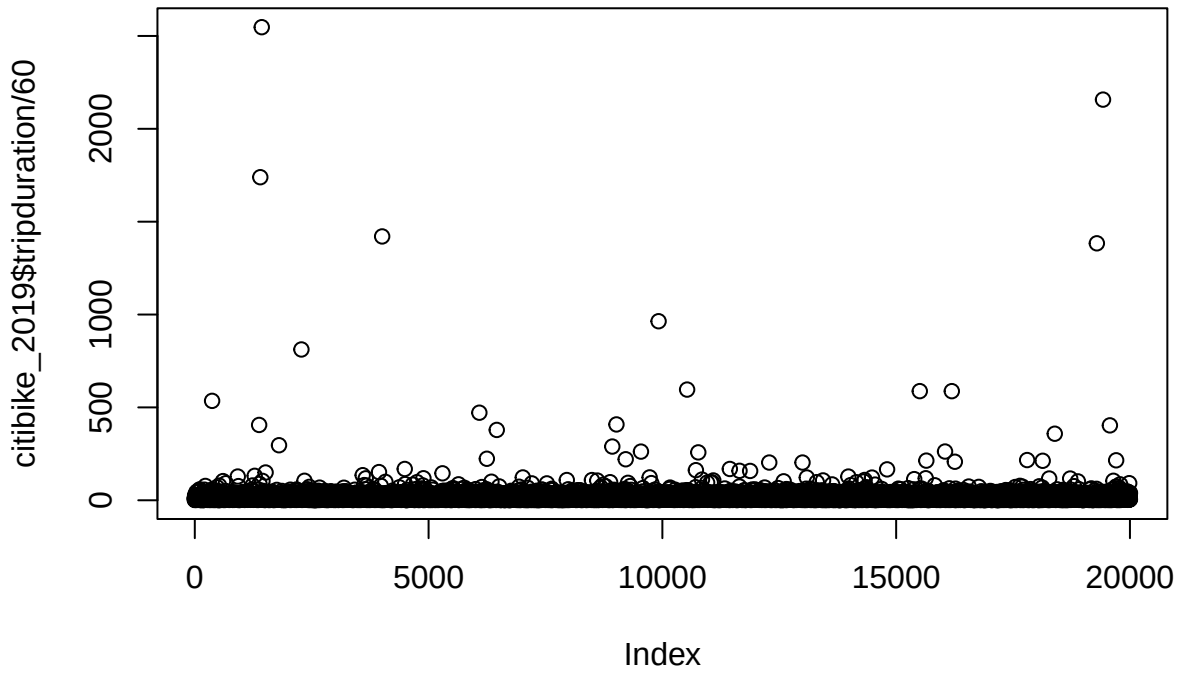
In this section, I examine selected individual variables from the CitiBike and Weather datasets. These items require transformation and/or cleaning as there are missing values or outliers which impede analysis otherwise.

```
print(paste0("The citibike data consists of ", nrow(citibike_2019), " data points spread across ", ncol
```

```
## [1] "The citibike data consists of 20000 data points spread across 15 features"
```

Examine variable trip_duration: The trip_duration is specified in seconds, but there are some outliers which may be incorrect, as the value for Max is quite high: 1.5283×10^5 seconds, or 1.7688657 days. I assumed that this data is bad, as nobody would willingly rent a bicycle for this period of time, given the fees that would be charged. Here is a histogram of the original data distribution:

```
plot(citibike_2019$tripduration/60)
```



Delete cases with unreasonable trip_duration values Let's assume that nobody would rent a bicycle for more than a specified timelimit (say, 3 hours), and drop any records which exceed this:

```
num_long_trips_removed<- nrow(citibike_2019[citibike_2019$tripduration > 7200,])
citibike_2019<-citibike_2019[citibike_2019$tripduration<=7200,]
print(paste0("Removed ", num_long_trips_removed, " trips of longer than 2 hours."))
```

```
## [1] "Removed 48 trips of longer than 2 hours."
```

Examine birth_year Other inconsistencies concern the collection of birth_year, from which we can infer the age of the participant. There are some months in which this value is omitted, while there are other months in which all values are populated. However, there are a few records which suggest that the rider is a centenarian – it seems highly implausible that someone born in the 1880s is cycling around Central Park – but the data does have such anomalies. Thus, a substantial amount of time was needed for detecting and cleaning such inconsistencies.

The birth year for some users is as old as 1888, which is not possible:

```
summary(citibike_2019$birth.year)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1888    1969    1983    1980    1990    2003
```

```
citibike_2019$age <- 2019 - citibike_2019$birth.year
```

```
num_old_age_removed<- nrow(citibike_2019[citibike_2019$age>90,])
citibike_2019<-citibike_2019[citibike_2019$age<90,]
print(paste0("Removed ", num_old_age_removed, " trips of people older thatn 90 years"))
```

Remove trips associated with very old users (age>90)

```
## [1] "Removed 10 trips of people older thatn 90 years"
```

```
library(geosphere)
citibike_2019$distance <- distHaversine(citibike_2019[,6:7], citibike_2019[,10:11])
citibike_2019$dist.lat <- abs((citibike_2019$start.station.latitude - citibike_2019$end.station.latitud
citibike_2019$dist.long <- abs((citibike_2019$start.station.longitude - citibike_2019$end.station.longi
summary(citibike_2019$distance)
```

Compute distance between start and end stations

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       0     571    1030    1323    1753   10671
```

Data Wrangling

```
is_weekday = function(timestamp){
  lubridate::wday(timestamp, week_start = 1) < 6
}
```

Feature Extraction

Add columns for the hour when the trip started, day of the week and if it was a week day

```
citibike_2019$start_date<-as.Date(citibike_2019$starttime)
citibike_2019$Hour<-hour(citibike_2019$starttime)
citibike_2019$dayofweek <- as.factor(wday(citibike_2019$starttime))
citibike_2019$weekday<-as.factor(as.numeric(sapply(citibike_2019$starttime, is_weekday)))
head(citibike_2019 %>% select("Hour","dayofweek","weekday"))
```

```
##        Hour dayofweek weekday
## 60660    15         4       1
## 93128     8         3       1
## 78548    10         5       1
## 114392   15         1       0
## 77740    23         1       0
## 164887   21         7       0
```

Convert into factor variables

```
citibike_2019$usertype<-as.factor(citibike_2019$usertype)
citibike_2019$gender<-as.factor(citibike_2019$gender)
```

Convert trip duration from seconds to minutes

```
citibike_2019$tripduration<-floor(citibike_2019$tripduration/60)
head(citibike_2019$tripduration)
```

```
## [1]  1 11 14 10  7  2
```

Join Citibike and Weather data

```
citibike_2019 <- citibike_2019 %>% inner_join(weather, by = c("start_date" = "DATE" ))
head(citibike_2019)
```

```
##   tripduration               starttime                stoptime
## 1            1 2019-05-15 15:29:12.3040 2019-05-15 15:30:42.6320
## 2           11 2019-07-02 08:41:47.3420 2019-07-02 08:53:28.8450
## 3           14 2019-06-06 10:01:34.8520 2019-06-06 10:15:53.8730
## 4           10 2019-08-18 15:34:33.9960 2019-08-18 15:45:19.2570
## 5            7 2019-06-02 23:23:45.2940 2019-06-02 23:31:00.5320
## 6            2 2019-10-05 21:08:12.9780 2019-10-05 21:10:36.9100
##   start.station.id          start.station.name start.station.latitude
## 1              258 DeKalb Ave & Vanderbilt Ave               40.68941
## 2              445          E 10 St & Avenue A               40.72741
## 3             3255             8 Ave & W 31 St               40.75059
## 4              368          Carmine St & 6 Ave               40.73039
## 5              474             5 Ave & E 29 St               40.74517
## 6              347 Greenwich St & W Houston St               40.72885
##   start.station.longitude end.station.id            end.station.name
## 1               -73.96885            241 DeKalb Ave & S Portland Ave
## 2               -73.98142            402          Broadway & E 22 St
## 3               -73.99468            383  Greenwich Ave & Charles St
## 4               -74.00215            328     Watts St & Greenwich St
## 5               -73.98683            519       Pershing Square North
## 6               -74.00859            127       Barrow St & Hudson St
##   end.station.latitude end.station.longitude bikeid   usertype birth.year
## 1             40.68981             -73.97493  16179 Subscriber       1960
## 2             40.74034             -73.98955  30170 Subscriber       1979
## 3             40.73524             -74.00027  25541 Subscriber       1949
## 4             40.72406             -74.00966  35038 Subscriber       1986
## 5             40.75187             -73.97771  30977 Subscriber       1994
## 6             40.73172             -74.00674  38358 Subscriber       1984
##   gender age  distance   dist.lat   dist.long start_date Hour dayofweek weekday
## 1      1  59  676.5607 0.00040288 0.006076630 2019-05-15   15         4       1
## 2      1  40  988.4800 0.01293526 0.008131030 2019-07-02    8         3       1
```

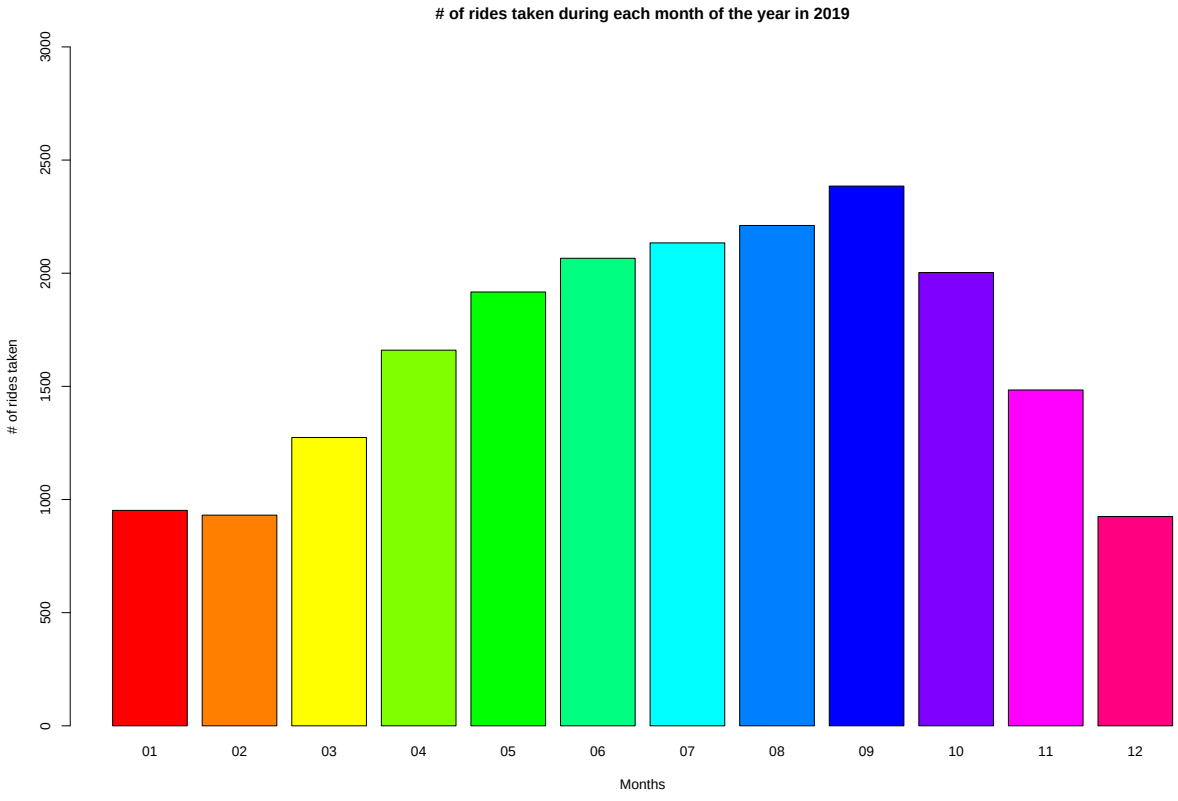
```
## 3      1  70  780.0821 0.01534735 0.005586185 2019-06-06   10         5       1
## 4      2  33  858.2380 0.00633050 0.007509770 2019-08-18   15         1       0
## 5      1  25 1036.4362 0.00670530 0.009124770 2019-06-02   23         1       0
## 6      2  35  223.7193 0.00287828 0.001846640 2019-10-05   21         7       0
##   PRCP SNOW SNWD TMAX TMIN WT01 WDF2 WDF5 WSF2 WSF5 WT08
## 1 0.01    0    0   69   44    1  250  300 10.1 19.9    0
## 2 0.02    0    0   85   71    0  270  330  8.9 14.1    0
## 3 0.04    0    0   83   68    1   50  270 12.1 21.0    0
## 4 0.64    0    0   90   70    1  240  230 15.0 23.9    1
## 5 0.86    0    0   81   61    1  260  220 12.1 25.1    1
## 6 0.00    0    0   61   45    0   30   20 13.0 21.9    0
```

Data Analysis to answer questions (hypotheses)

Number of rides taken during each month

Here we can clearly see a trend that number of rides taken during December, January and February are lower as compared to other seasons. This shows the seasonality of the rides i.e. number of rides during winters go down.

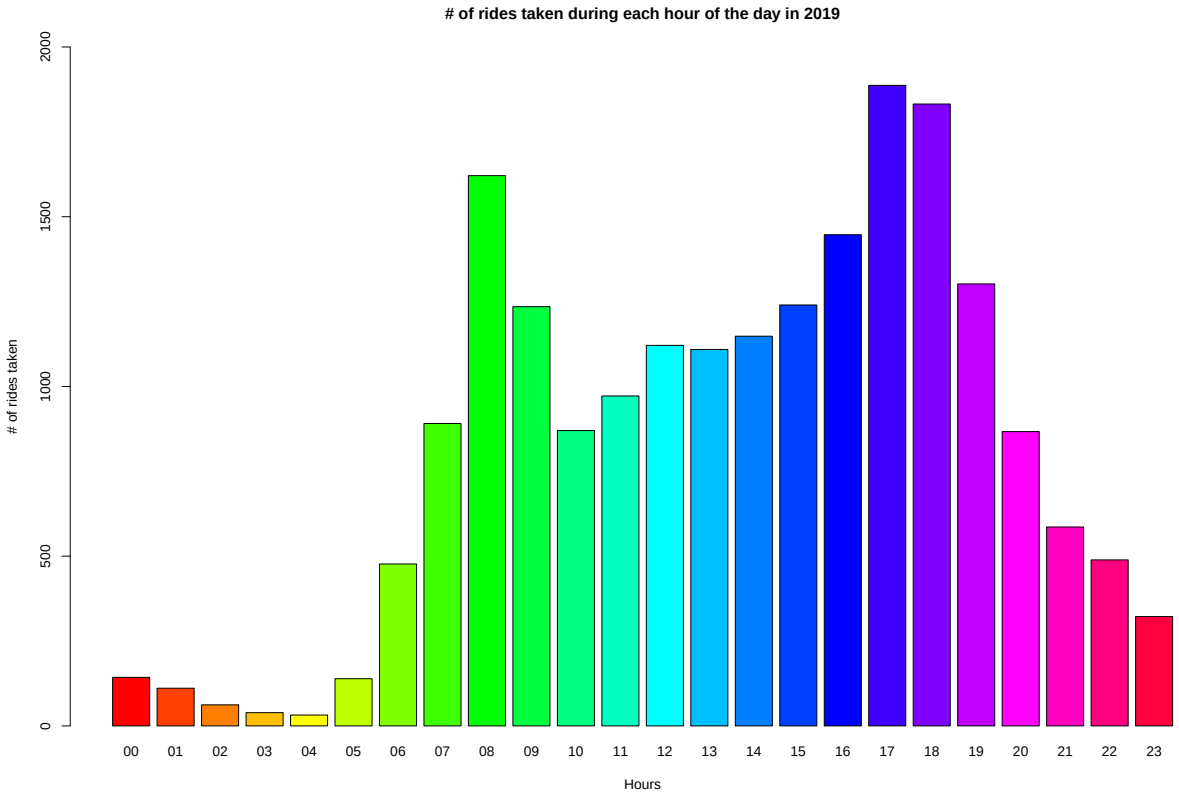
```
barplot(table(substring(citibike_2019$starttime, 6, 7)), col=rainbow(12), xlab="Months", ylab="# of r
```



Number of rides during each hour of the day

Here we can clearly see that the number of rides taken after midnight fall drastically and the number of rides go up only around 7 am.

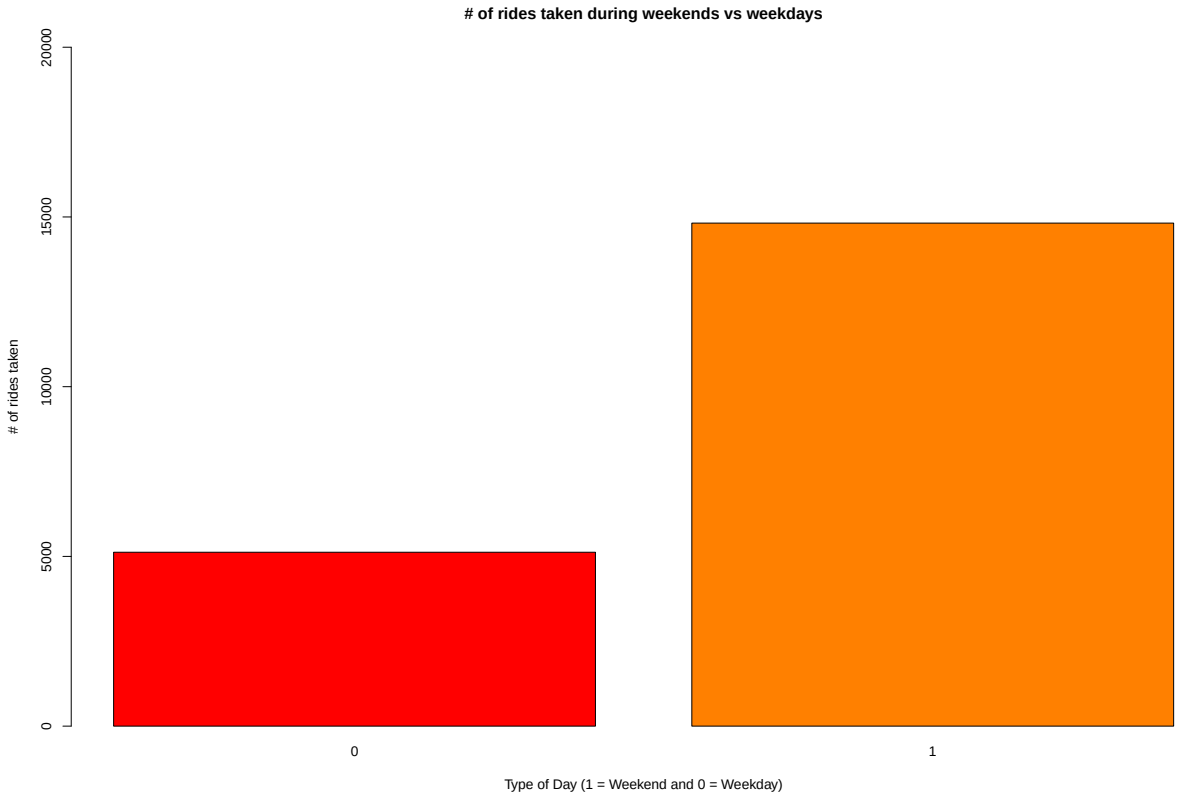
```
barplot(table(substring(citibike_2019$starttime, 12, 13)), col=rainbow(24), xlab="Hours", ylab="# of
```



Number of rides depending on if it's a weekend

Here we can clearly see that the number of rides taken during weekends is a lot more than on weekdays.

```
barplot(table(citibike_2019$weekday), col=rainbow(12), xlab="Type of Day (1 = Weekend and 0 = Weekday
```

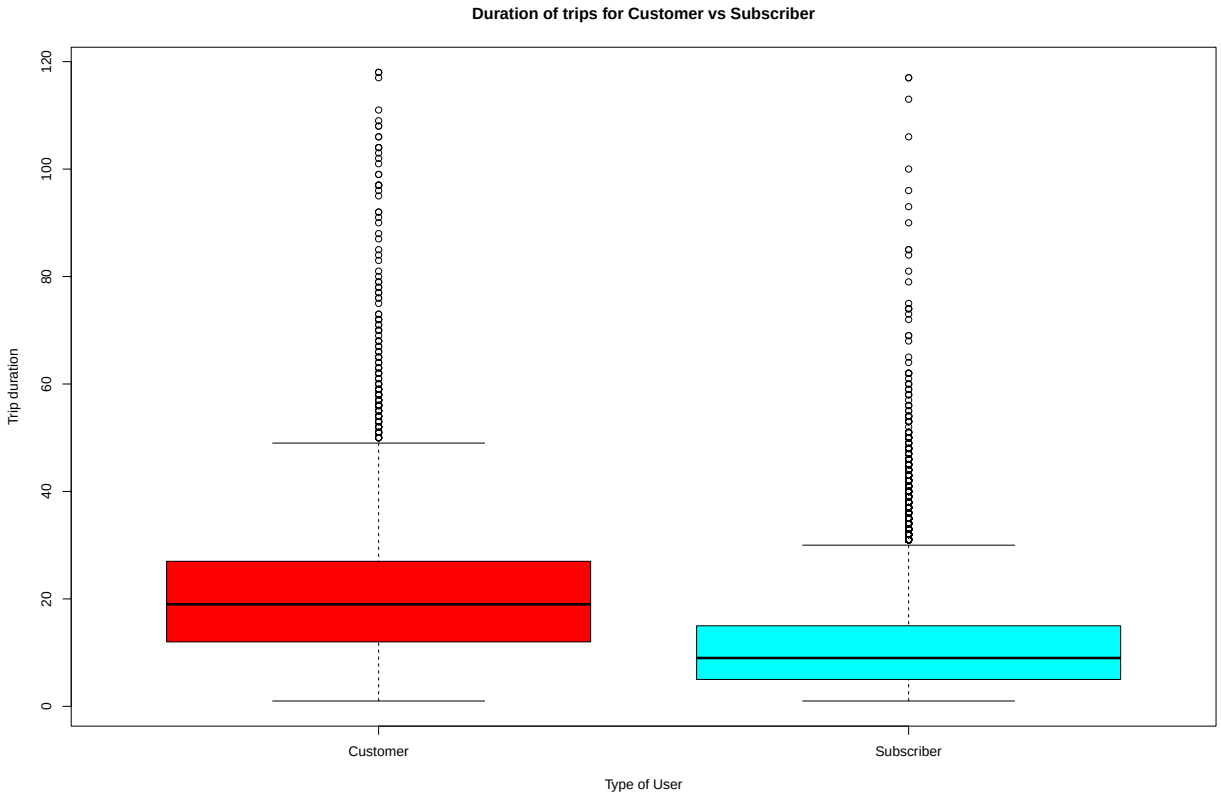


Ride duration based on user type (Customer = 24-hour pass or 3-day pass user; Subscriber = Annual Member)

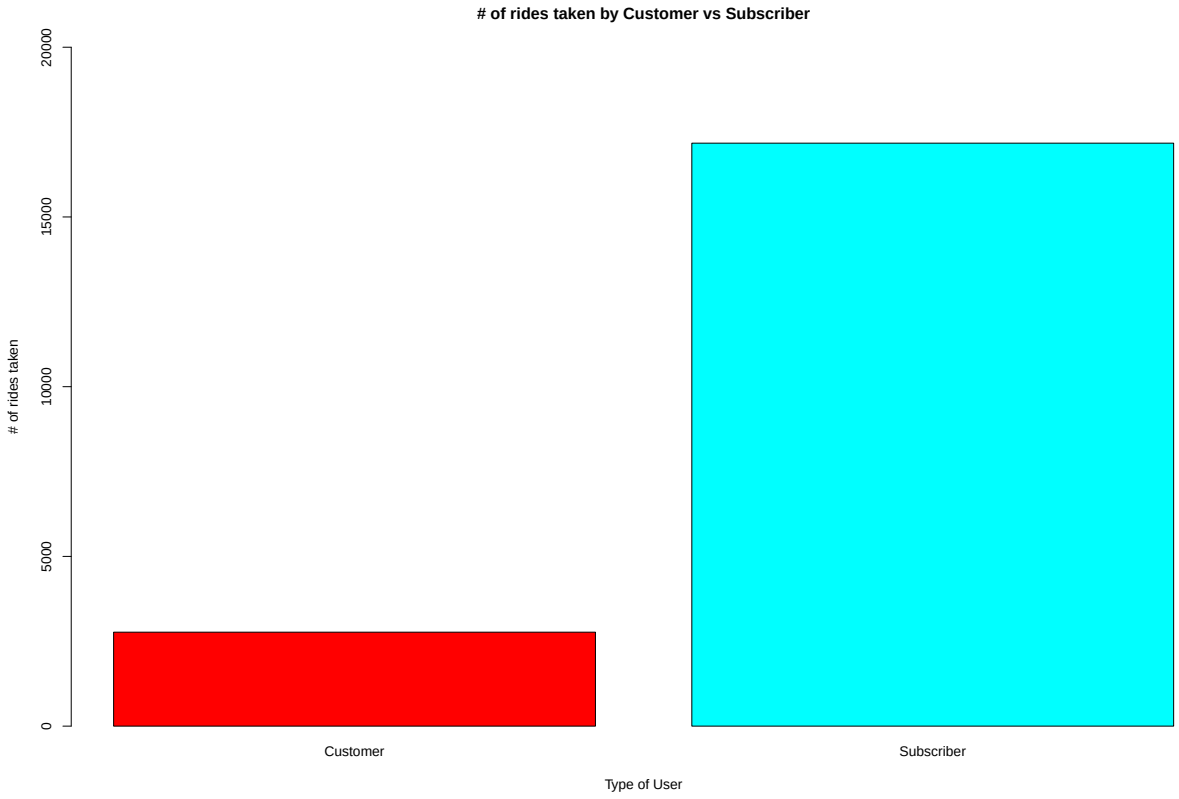
The first plot reveals something interesting and counter intuitive. Initially I thought annual members would have longer ride duration but the plot proves my hypothesis incorrect. It makes sense because people with a pass for a couple of days might be tourists or took the pass specifically to explore the city so they have higher ride durations. At the same time annual members are regular commuters and probably need it to commute to office or school and hence ride for a short duration.

The second plot shows that annual subscribers take lot more rides as compared to customers which makes sense.

```
boxplot(citibike_2019$tripduration ~ citibike_2019$usertype, col=rainbow(2), xlab="Type of User", ylab
```




```
barplot(table(citibike_2019$usertype), col=rainbow(2), xlab="Type of User", ylab="# of rides taken", m
```

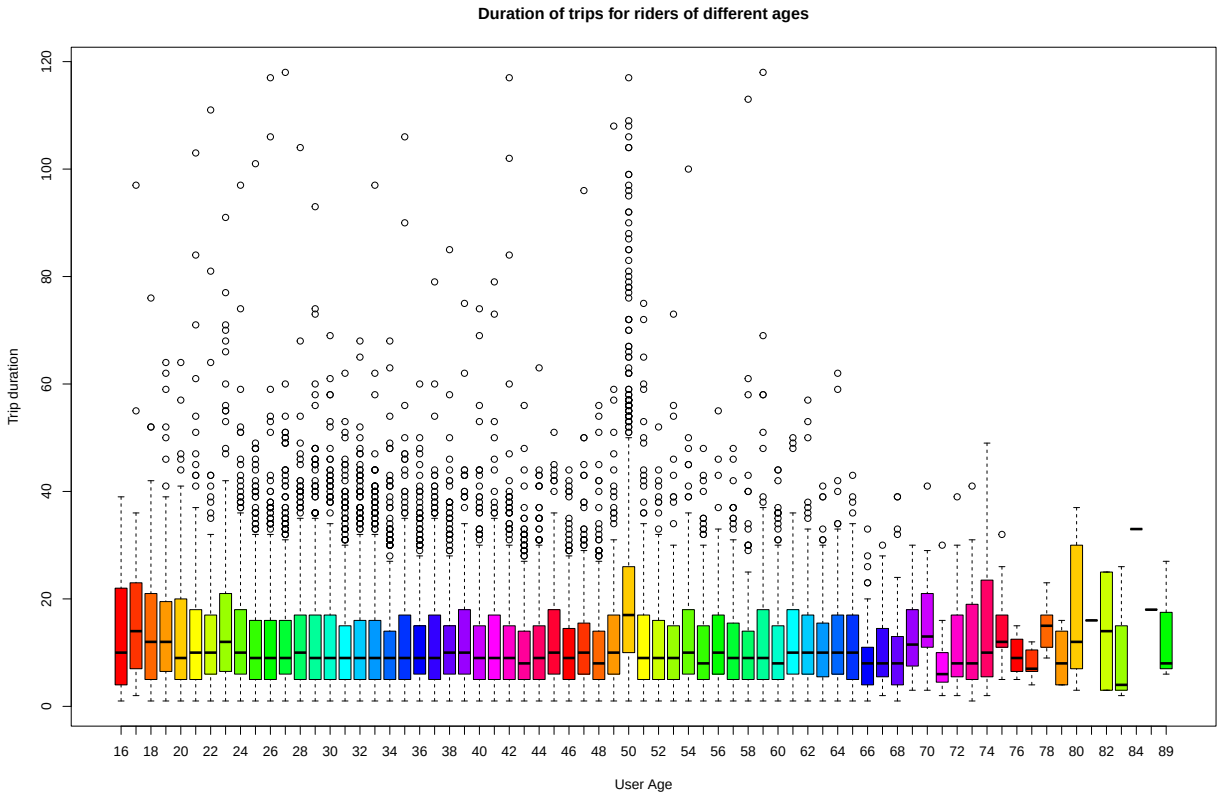


Ride duration and number of rides based on user age

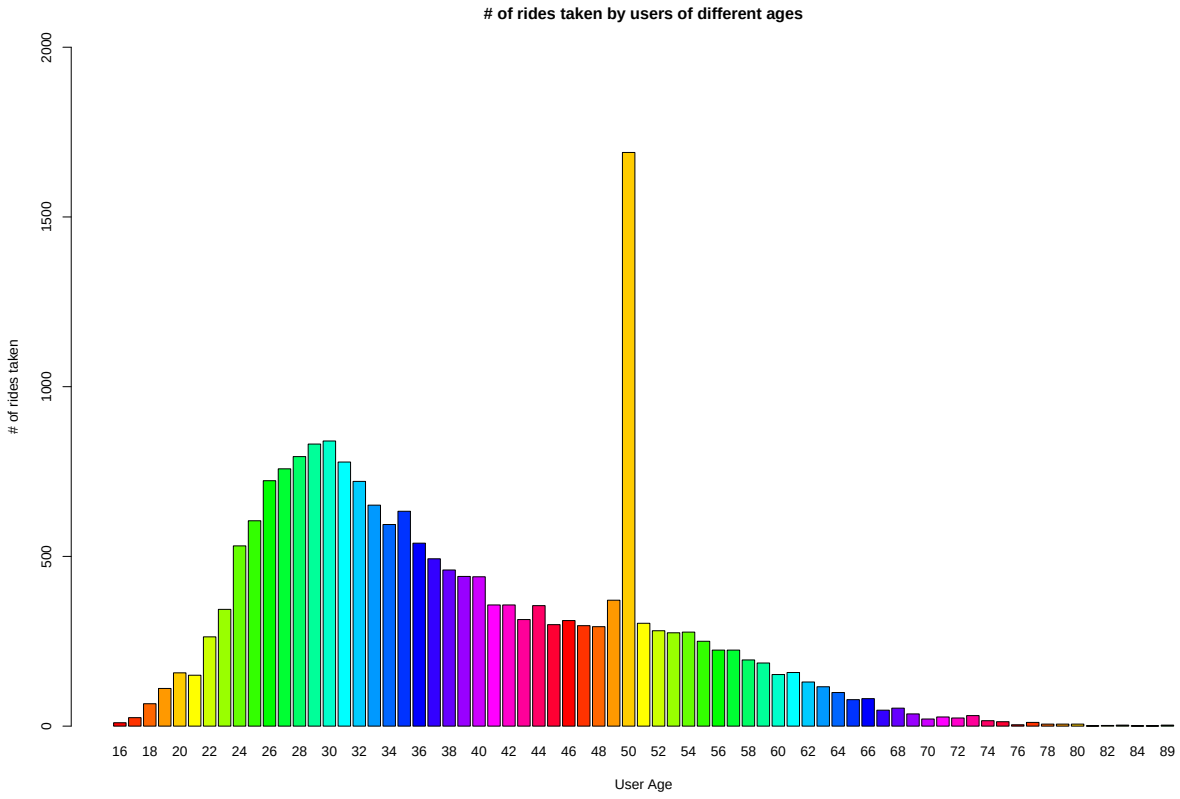
The first plot is surprisingly uniform. I expected younger riders to ride for longer durations but old people also ride for equally long durations.

Looking at the second graph we can conclude that young people use the bikes more frequently and hence they are the prime demographic the company could target for ad campaigns.

```
boxplot(citibike_2019$tripduration ~ citibike_2019$age, col=rainbow(30), xlab="User Age", ylab="Trip
```




```
barplot(table(citibike_2019$age), col=rainbow(30), xlab="User Age", ylab="# of rides taken", main="# o
```

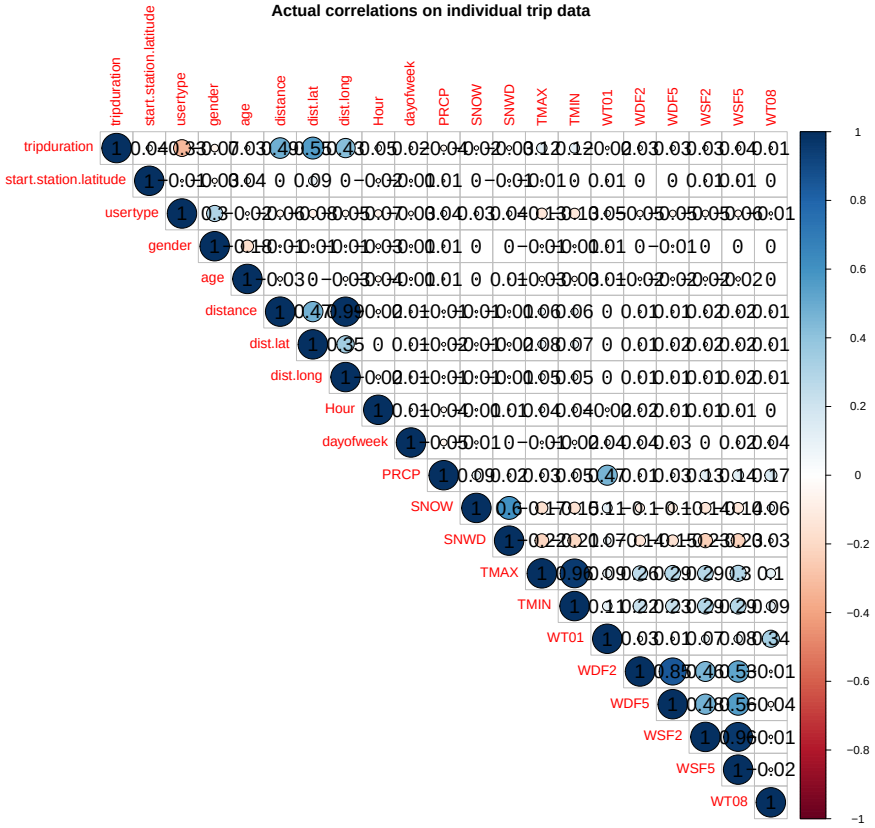


Prediction Models

I used a lot of models below to fit the data to get a good estimation for the ride durations. I got the lowest root mean squared error (RMSE) of 7.5 with Random Forest Regressor so to maintain a concise document I have only shown the code for it below. Due to the short nature of this course and not having access to high level computing resources I built a very basic version of these models below. But the RMSE can be significantly brought down by performing hyper-parameter tuning and having better computing resources to increase the model complexity.

Correlations of individual trip data features

We can examine the correlations between variables to understand the relationship between variables, and also to help be alert to potential problems of multicollinearity. Here we compute actual correlations between key variables. Here we compute the correlations between key variables on the individual CitiBike Trip data



Train - Test Split

```
smp_size<- floor(0.8*nrow(citibike_2019))
set.seed(123)
train_index<-sample(seq_len(nrow(citibike_2019)), size = smp_size)
train_data <- citibike_2019[train_index,]
test_data <-  citibike_2019[- train_index,]
```

Random Forest Model

```
colstouse <-setdiff(names(train_data),c("starttime","start.station.id","end.station.id",
                                        "stoptime","start.station.name","end.station.name",
                                        "start.station.longitude","start_date",
                                        "end.station.latitude","end.station.longitude",
                                        "weekday","bikeid","birth.year","DATE","SNOW","SNWD"))
X_train<-train_data[ , which(names(train_data) %in% colstouse)]
names(X_train)
```

```
##  [1] "tripduration"           "start.station.latitude" "usertype"
##  [4] "gender"                 "age"                    "distance"
##  [7] "dist.lat"               "dist.long"              "Hour"
## [10] "dayofweek"              "PRCP"                   "TMAX"
```

```
## [13] "TMIN"                  "WT01"                  "WDF2"
## [16] "WDF5"                  "WSF2"                  "WSF5"
## [19] "WT08"
```

```
library('randomForest')

control <- trainControl(method='repeatedcv',
                        number=10,
                        repeats=3)

#Metric compare model is Root Mean Squared Error
metric <- "RMSE"
set.seed(123)

#Number randomly variable selected is mtry
mtry <- sqrt(ncol(X_train)-1)
tunegrid <- expand.grid(.mtry=mtry)
rfmodel <- train(tripduration ~ .,
                      data = X_train,
                      method='rf',
                      metric=metric,
                      tuneGrid=tunegrid,
                      trControl=control)
```

```
rfmodel
```

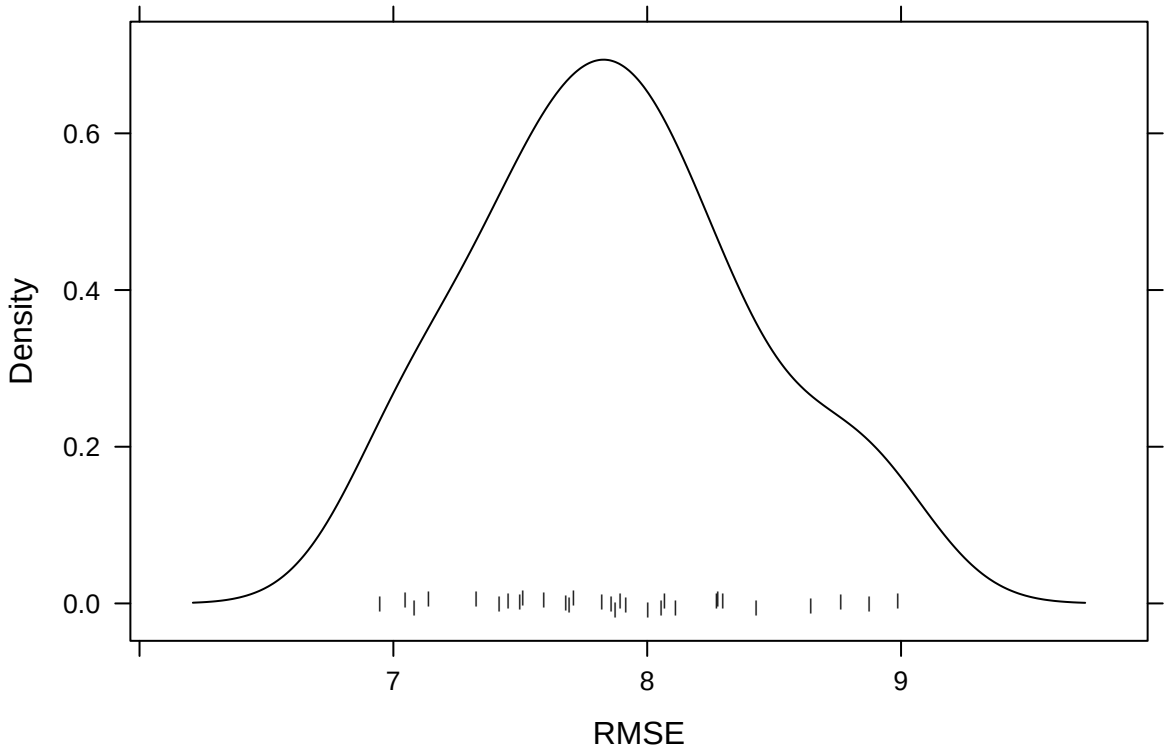
```
## Random Forest
##
## 15947 samples
##    18 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 14352, 14351, 14352, 14350, 14352, 14354, ...
## Resampling results:
##
##   RMSE      Rsquared   MAE
##   7.874974  0.5089116  4.489987
##
## Tuning parameter 'mtry' was held constant at a value of 4.242641
```

```
summary(rfmodel)
```

```
##                 Length Class      Mode
## call               4  -none-     call
## type               1  -none-     character
## predicted      15947  -none-     numeric
## mse              500  -none-     numeric
## rsq              500  -none-     numeric
## oob.times      15947  -none-     numeric
## importance        24  -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               15947  -none-     numeric
## test                0  -none-     NULL
## inbag               0  -none-     NULL
## xNames             24  -none-     character
## problemType         1  -none-     character
## tuneValue           1  data.frame list
## obsLevels           1  -none-     logical
## param               0  -none-     list
```

```
trellis.par.set(caretTheme())
densityplot(rfmodel, pch = "|")
```



Evaluate Performance on the train data

```
y_test<-train_data$tripduration
predicted = predict(rfmodel,train_data)
residuals = y_test - predicted
RMSE = sqrt(mean(residuals^2))
cat('The root mean square error of the train data is ', round(RMSE,3),'\n')
```

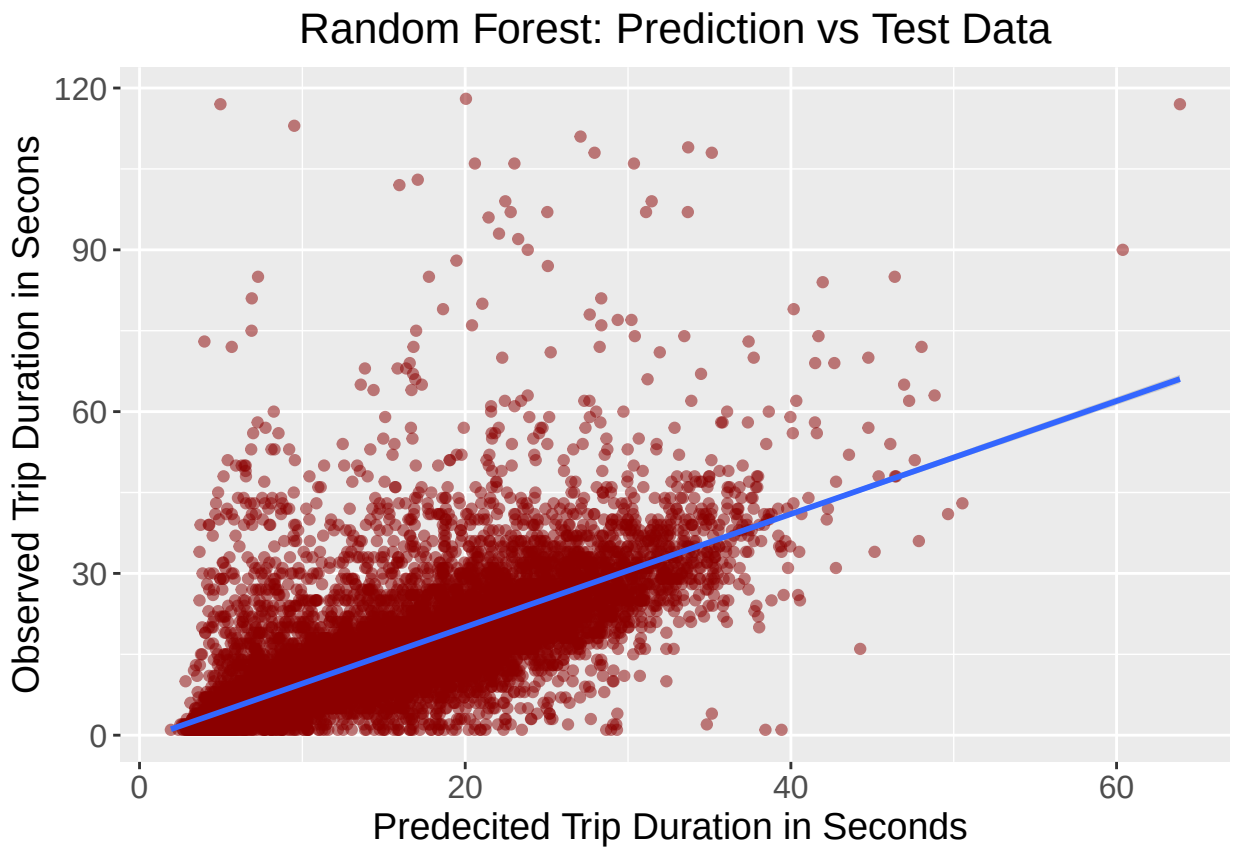
```
## The root mean square error of the train data is  7.518
```

```
y_test_mean = mean(y_test)
# Calculate total sum of squares
tss =  sum((y_test - y_test_mean)^2 )
# Calculate residual sum of squares
rss =  sum(residuals^2)
# Calculate R-squared
rsq  =  1 - (rss/tss)
cat('The R-square of the train data is ', round(rsq,3), '\n')
```

```
## The R-square of the train data is  0.529
```

```
options(repr.plot.width=8, repr.plot.height=4)
my_data = as.data.frame(cbind(predicted = predicted,
                              observed = y_test))
# Plot predictions vs test data
ggplot(my_data,aes(predicted, observed)) + geom_point(color = "darkred", alpha = 0.5) +
    geom_smooth(method=lm)+ ggtitle('Linear Regression ') + ggtitle("Random Forest: Prediction vs Test D
      xlab("Predecited Trip Duration in Seconds ") + ylab("Observed Trip Duration in Secons") +
        theme(plot.title = element_text(size=16,hjust = 0.5),
         axis.text.y = element_text(size=12), axis.text.x = element_text(size=12,hjust=.5),
         axis.title.x = element_text(size=14), axis.title.y = element_text(size=14))
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



Evaluate Performance on the test data

```
y_test<-test_data$tripduration
predicted = predict(gbmmodel,test_data)
residuals = y_test - predicted
RMSE = sqrt(mean(residuals^2))
cat('The root mean square error of the test data is ', round(RMSE,3),'\n')
```

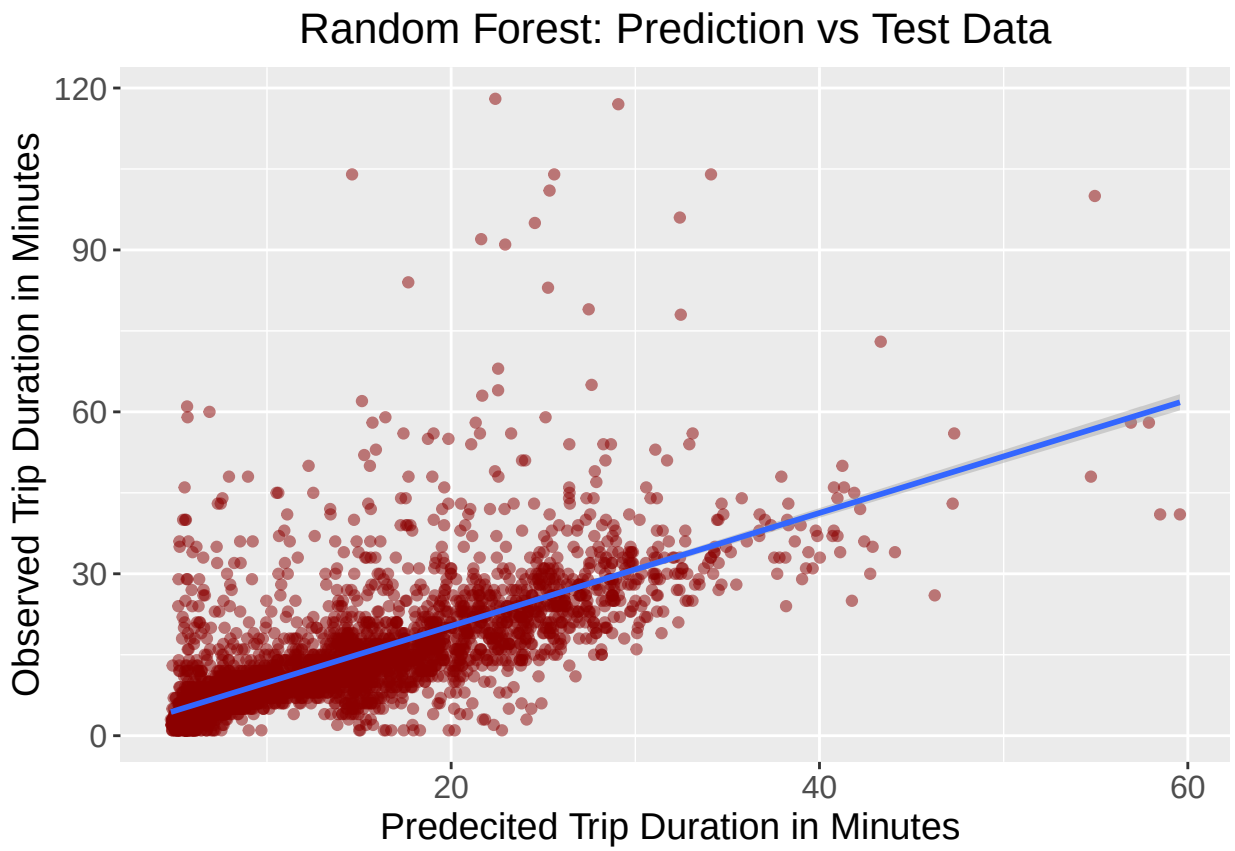
```
## The root mean square error of the test data is  8.074
```

```
y_test_mean = mean(y_test)
# Calculate total sum of squares
tss =  sum((y_test - y_test_mean)^2 )
# Calculate residual sum of squares
rss =  sum(residuals^2)
# Calculate R-squared
rsq  =  1 - (rss/tss)
cat('The R-square of the test data is ', round(rsq,3), '\n')
```

```
## The R-square of the test data is  0.51
```

```
options(repr.plot.width=8, repr.plot.height=4)
my_data = as.data.frame(cbind(predicted = predicted,
                              observed = y_test))
# Plot predictions vs test data
ggplot(my_data,aes(predicted, observed)) + geom_point(color = "darkred", alpha = 0.5) +
    geom_smooth(method=lm)+ ggtitle('Gradient Boosted Machines ') + ggtitle("Random Forest: Prediction
      xlab("Predecited Trip Duration in Minutes ") + ylab("Observed Trip Duration in Minutes") +
        theme(plot.title = element_text(size=16,hjust = 0.5),
         axis.text.y = element_text(size=12), axis.text.x = element_text(size=12,hjust=.5),
         axis.title.x = element_text(size=14), axis.title.y = element_text(size=14))
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



Conclusion

I was able to analyse a lot of information from the hypotheses I tested and even by building the predictive models. Cleaning the dataset and extracting features was also fun. I can make a lot of recommendations based on the hypothesis I tested. For instance, the citibike subscriptions should be marketed to the young demographic as they take more number of rides in general. Business would be slow during the winter months so inventory can be managed effectively and maybe repairs can be scheduled during this time. Demand is high during weekends so price can be surged whereas during weekdays discount offers should be pushed to drive users to the system. The prediction models I built can also be used to effectively manage inventory and also to predict sales for the future.

A future scope for this could be to improve the predictive models and also perform more feature extraction. This can be extended to predicting demand during hours of the days. We can also predict the rider destination using the source and their previous ride patterns. All of these can be together used by the company to manage their inventory very effectively and cut costs and increase profits.