TMAAP_Taxi

Tanushree Chungi

Michael Hansen

Sai Alekhya Tadikonda

Arjun Soni

Prathibha Chowdary Kolli

30/04/2020

Business Idea

In a busy city such as Chicago, driving and finding parking spots for private vehicles is very difficult. So, people who commute daily take a taxi because it is an easy and efficient means of transportation. Also, taxis are an always available option for tourists and other travelers to move around the city.

The name of our taxi service is unique too, TMAAP is the starting letter of each of our names, T stands for Tanushree, M stands for Micheal , A stands for Alekhya, A stands for Arjun and P stands for Prathibha. We as a team, want to start a new taxi service called the TMAAP taxi for people whose office is situated in the downtown of Chicago. Additionally, we plan to expand our business operations for people during times of high volume (such as holidays or breaks). In order to have a competitive edge and to better understand how taxi service in Chicago is, we are going to analyze the Chicago taxi trip dataset. We would want to know the frequency, the miles covered by a trip, information regarding the fare, the mode of payment and tolls.

Why did we choose this dataset!

We choose this dataset because we understand that before we take off and start with a taxi service business, we need to understand the market we are getting into, we need to understand, analyze and use this real time data to build our company strategies better.

This dataset provides us with a lot of information regarding the trips, such as trip miles and the duration of the trip along with the financials involved.

How will these R results be useful for our taxi service?

Looking at the visualizations we will get a brief idea of how the various variables in our dataset are related to each other. The data explorations and the summary statistics are important for us because we will get to know the numbers and variables that we are interested in like fare, trip miles, and trip minutes.

The special days analysis will help us come up with strategies that we will offer so that our taxi services will be effective. The regression analysis will give us the power to predict. By understanding the model, we will be able to make predictions and understand how one variable affects the other variable.

The results from clustering and SVM will give us an extra leap in understanding the data better and use this understanding and knowledge that we gained to implement our taxi service.

A little about our dataset:

Our dataset has observations that are over 1 million. Each observation belongs to one trip, some of the variables are trip miles, trip minutes, fare and tips. This dataset has the company names that own the taxis. We have performed the data exploration queries to understand the data better.

What do columns in our dataset mean!?

Column name	Datatype	Description
trip_id	chr	It is the unique id of the trip.
taxi_id	chr	This variable represents the unique id of the taxi trip_start_timestamp
trip_end_timestamp	dttm	The time stamp tells us the time stamp of ending of the trip it includes the date and time
trip_seconds	dbl	The trip seconds gives the duration of the code in seconds.
trip_miles	dbl	This is the distance in miles that the trip covers pickup_community_area
dropoff_community_area	dbl	This is a numerically coded column that represents the area code in the map of Chicago.
fare	dbl	This is a numerical column representing the fare of the trip in dollars.
tolls	dbl	This is a numerical column representing the tolls paid in the trip in dollars
extras	dbl	These are numerical variables representing the extra amount paid other than the fare, tips, tolls
trip_total	dbl	It is a numerical column that represents the sum of fare+ tips+ tolls+extras
payment_type	chr	This is a categorical variable that nine categories that include cash, credit card etc.
company	chr	This variable has a list of all local taxi companies of Chicago

Research Question:

As a team, we want to analyze the data of taxi trips in 2019 and use the results of the analysis to build models for prediction and also to come up with strategies that will help us start better and stay stronger as a successful taxi company in the market.

Link to the data:

Data source link "https://data.cityofchicago.org/Transportation/Taxi-Trips/wrvz-psew/data"

R-Studio Code:

Installing the required packages:

```
# install.packages("tidyverse")
# install.packages("ggmosaic")
# install.packages("forcats")
# install.packages("FactoMineR")
# install.packages("factoextra")
# install.packages("cluster")
# install.packages("lubridate")
# install.packages ("ISLR")
# install.packages("ROCR")
# install.packages("pscl")
# install.packages("caret")
# install.packages("e1071")
# install.packages("dbplot")
# install.packages("modelr")
# install.packages("broom")
# install.packages("gridextra")
# install.packages("sparkylr")
# install.packages("ggplot2")
# install.packages("kernlab")
# install.packages("RColorBrewer")
# install.packages("Scales")
```

Loading the libraries:

```
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------- tidyver
se 1.3.0 --

## v ggplot2 3.2.1     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   1.0.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0

## -- Conflicts ------------------------------------------- tidyverse_con
flicts() --
```

```
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(ggmosaic)
library(forcats)
library(RColorBrewer)
library(kernlab)
```

```
##
## Attaching package: 'kernlab'
```

```
## The following object is masked from 'package:purrr':
##
##     cross
```

```
## The following object is masked from 'package:ggplot2':
##
##     alpha
```

```
library(ggplot2)
library(sparklyr)
```

```
##
## Attaching package: 'sparklyr'
```

```
## The following object is masked from 'package:purrr':
##
##     invoke
```

```
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(modelr)
library(broom)
```

```
##
## Attaching package: 'broom'
```

```
## The following object is masked from 'package:modelr':
##
##     bootstrap
```

```
library(dbplot)
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following object is masked from 'package:base':
##
##     date

library(cluster)
library(factoextra)

## Welcome! Want to learn more? See two factoextra-related books at https://g
oo.gl/ve3WBa

library(pscl)

## Classes and Methods for R developed in the
## Political Science Computational Laboratory
## Department of Political Science
## Stanford University
## Simon Jackman
## hurdle and zeroinfl functions by Achim Zeileis

library(ROCR)

## Loading required package: gplots

##
## Attaching package: 'gplots'

## The following object is masked from 'package:stats':
##
##     lowess

library(ISLR)
library(caret)

## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift

library(e1071)
library(FactoMineR)
library(scales)

##
## Attaching package: 'scales'

## The following object is masked from 'package:kernlab':
##
##     alpha
```

```
## The following object is masked from 'package:purrr':
##
##     discard

## The following object is masked from 'package:readr':
##
##     col_factor
```

Data Preparation and data binding:

Our data is imported for each month in the year 2019 and each file has a little over 1 million rows and these 12 files that belong to 12 months have been combined to one file with "rbind" function taking 84500 observations from each month file. The Columns of the dataset are imported with spaces and they are capitalized. We have renamed the columns (with underscores) to avoid confusion.

```
#trips_jan_2019 <- read_csv("C:/Users/saial/Downloads/Taxi_Trips_January.csv"
)
#trips_feb_2019 <- read_csv("C:/Users/saial/Downloads/Taxi_Trips_February.csv
")
#trips_mar_2019 <- read_csv("C:/Users/saial/Downloads/Taxi_Trips_March.csv")
#trips_apr_2019 <- read_csv("C:/Users/saial/Downloads/Taxi_Trips_April.csv")
#trips_may_2019 <- read_csv("C:/Users/saial/Downloads/Taxi_Trips_May.csv")
#trips_jun_2019 <- read_csv("C:/Users/saial/Downloads/Taxi_Trips_June.csv")
#trips_jul_2019 <- read_csv("C:/Users/saial/Downloads/Taxi_Trips_July.csv")
#trips_aug_2019 <- read_csv("C:/Users/saial/Downloads/Taxi_Trips_August.csv")
#trips_sep_2019 <- read_csv("C:/Users/saial/Downloads/Taxi_Trips_September.cs
v")
#trips_oct_2019 <- read_csv("C:/Users/saial/Downloads/Taxi_Trips_October.csv"
)
#trips_nov_2019 <- read_csv("C:/Users/saial/Downloads/Taxi_Trips_November.csv
")
#trips_dec_2019 <- read_csv("C:/Users/saial/Downloads/Taxi_Trips_December.csv
")

#trips_jan <- trips_jan_2019 %>% sample_n(84500)
#trips_feb <- trips_feb_2019 %>% sample_n(84500)
#trips_mar <- trips_mar_2019 %>% sample_n(84500)
#trips_apr <- trips_apr_2019 %>% sample_n(84500)
#trips_may <- trips_may_2019 %>% sample_n(84500)
#trips_jun <- trips_jun_2019 %>% sample_n(84500)
#trips_jul <- trips_jul_2019 %>% sample_n(84500)
#trips_aug <- trips_aug_2019 %>% sample_n(84500)
#trips_sep <- trips_sep_2019 %>% sample_n(84500)
#trips_oct <- trips_oct_2019 %>% sample_n(84500)
#trips_nov <- trips_nov_2019 %>% sample_n(84500)
#trips_dec <- trips_dec_2019 %>% sample_n(84500)

#trips_2019 <- rbind(trips_jan, trips_feb, trips_mar, trips_apr, trips_may, t
rips_jun, trips_jul, trips_aug, trips_sep, trips_oct, trips_nov, trips_dec)
```

```
#trips_2019 <- trips_2019 %>%
#  rename(
#    trip_id = `Trip ID`,
#    taxi_id = `Taxi ID`,
#    trip_start_timestamp = `Trip Start Timestamp`,
#    trip_end_timestamp = `Trip End Timestamp`,
#    trip_seconds = `Trip Seconds`,
#    trip_miles = `Trip Miles`,
#    pickup_census_tract = `Pickup Census Tract`,
#    dropff_census_tract = `Dropoff Census Tract`,
#    pickup_community_area = `Pickup Community Area`,
#    dropoff_community_area = `Dropoff Community Area`,
#    fare = Fare,
#    tips = Tips,
#    tolls = Tolls,
#    extras = Extras,
#    trip_total = `Trip Total`,
#    payment_type = `Payment Type`,
#    company = Company,
#    pickup_centroid_latitude = `Pickup Centroid Latitude`,
#    pickup_centroid_longitude = `Pickup Centroid Longitude`,
#    pickup_centroid_location = `Pickup Centroid Location`,
#    dropoff_centroid_latitude = `Dropoff Centroid Latitude`,
#    dropoff_centroid_longitude = `Dropoff Centroid Longitude`,
#    dropoff_centroid_location = `Dropoff Centroid  Location`
#    )

#write_csv(trips_2019, "trips_2019.csv")
```

Data Import

```
trips_2019_1 <- read_csv("c:/Users/TANUSHREE/Desktop/OMIS_665/trips_2019.csv"
)
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   trip_id = col_character(),
##   taxi_id = col_character(),
##   trip_start_timestamp = col_character(),
##   trip_end_timestamp = col_character(),
##   payment_type = col_character(),
##   company = col_character(),
##   pickup_centroid_location = col_character(),
##   dropoff_centroid_location = col_character()
## )

## See spec(...) for full column specifications.
```

Tidying the dataset

Variables trip_start_timestamp and trip_end_timestamp are in “mm/dd/yyyy HH:MM” format. These variables are separated into different columns of respective day, month, year, hour, minute and weekdays (this is achieved through the lubridate package). Also, some taxi_id and trip_id values in the data set are not available (N/A) those values are omitted as well (we considered only the valid rides). For better understanding the trip duration we added the trip_minutes column to the dataset based on the trip_seconds.

```
trips_2019_1$trip_start_timestamp <-mdy_hms(trips_2019_1$trip_start_timestamp
)

trips_2019_1$trip_start_year <-year(trips_2019_1$trip_start_timestamp)

trips_2019_1$trip_start_month <- (month(trips_2019_1$trip_start_timestamp, la
bel = TRUE))

trips_2019_1$trip_start_day <-  (day(trips_2019_1$trip_start_timestamp))

trips_2019_1$trip_start_weekday <- (wday(trips_2019_1$trip_start_timestamp, l
abel = TRUE))

trips_2019_1$trip_start_hour <-  (hour(trips_2019_1$trip_start_timestamp))

trips_2019_1$trip_start_minute <-  (minute(trips_2019_1$trip_start_timestamp)
)

# trip_end_timestamp variable is divided into respective day, month, year, ho
ur and minute as follows

trips_2019_1$trip_end_timestamp <- mdy_hms(trips_2019_1$trip_end_timestamp)

trips_2019_1$trip_end_year <-  (year(trips_2019_1$trip_end_timestamp))

trips_2019_1$trip_end_month <- (month(trips_2019_1$trip_end_timestamp, label
= TRUE))

trips_2019_1$trip_end_day <-  (day(trips_2019_1$trip_end_timestamp))

trips_2019_1$trip_end_hour <-  (hour(trips_2019_1$trip_end_timestamp))

trips_2019_1$trip_end_minute <-  (minute(trips_2019_1$trip_end_timestamp))

# selecting only those trips which have a trip ID and taxi ID
trips_2019_1 <- trips_2019_1 %>%
  filter(!is.na(trip_id)) %>%
  filter(!is.na(taxi_id))
```

```
# Adding the trip_minutes column to the dataset
trips_2019_1 <- trips_2019_1%>%
  mutate(trip_minutes=trip_seconds/60)
```

Glimpse of our dataset

```
glimpse(trips_2019_1)
```

```
## Observations: 1,013,947
## Variables: 35
## $ trip_id                   <chr> "d3d7d2f65f61eea243cde9e92541aae9f7d161
8...
## $ taxi_id                   <chr> "b54dc19d3df07681ef496e54918ff42588f761
4...
## $ trip_start_timestamp      <dttm> 2019-01-22 15:15:00, 2019-01-19 22:30:
0...
## $ trip_end_timestamp        <dttm> 2019-01-22 15:30:00, 2019-01-19 22:30:
0...
## $ trip_seconds              <dbl> 457, 458, 720, 296, 1620, 422, 120, 101
7...
## $ trip_miles                <dbl> 1.19, 1.14, 0.90, 0.60, 0.00, 1.20, 0.0
0...
## $ pickup_census_tract       <dbl> NA, NA, 17031839100, 17031280100, NA, 1
7...
## $ dropff_census_tract       <dbl> NA, NA, 17031833000, 17031839100, NA, 1
7...
## $ pickup_community_area     <dbl> 8, 8, 32, 28, 28, 28, 32, 32, 77, 28, 2
8...
## $ dropoff_community_area    <dbl> 8, 8, 28, 32, 3, 8, 32, 7, 2, 32, 8, 76
,...
## $ fare                      <dbl> 6.75, 6.75, 7.75, 5.25, 19.50, 6.50, 4.
2...
## $ tips                      <dbl> 0.00, 2.00, 0.00, 0.00, 0.00, 0.00, 0.0
0...
## $ tolls                     <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0...
## $ extras                    <dbl> 0, 2, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
0...
## $ trip_total                <dbl> 6.75, 11.25, 7.75, 5.25, 19.50, 6.50, 4
....
## $ payment_type              <chr> "Cash", "Credit Card", "Credit Card", "
C...
## $ company                   <chr> "Chicago Carriage Cab Corp", "Sun Taxi"
,...
## $ pickup_centroid_latitude  <dbl> 41.89960, 41.89960, 41.88099, 41.88530,
...
## $ pickup_centroid_longitude <dbl> -87.63331, -87.63331, -87.63275, -87.64
2...
## $ pickup_centroid_location  <chr> "POINT (-87.6333080367 41.899602111)",
```

```
"...
## $ dropoff_centroid_latitude  <dbl> 41.89960, 41.89960, 41.88528, 41.88099,
...
## $ dropoff_centroid_longitude <dbl> -87.63331, -87.63331, -87.65723, -87.63
2...
## $ dropoff_centroid_location  <chr> "POINT (-87.6333080367 41.899602111)",
"...
## $ trip_start_year            <dbl> 2019, 2019, 2019, 2019, 2019, 2019, 201
9...
## $ trip_start_month           <ord> Jan, Jan, Jan, Jan, Jan, Jan, Jan, Jan,
...
## $ trip_start_day             <int> 22, 19, 23, 23, 22, 15, 10, 7, 15, 17,
2...
## $ trip_start_weekday         <ord> Tue, Sat, Wed, Wed, Tue, Tue, Thu, Mon,
...
## $ trip_start_hour            <int> 15, 22, 19, 7, 23, 20, 6, 16, 13, 12, 1
9...
## $ trip_start_minute          <int> 15, 30, 15, 30, 30, 15, 30, 15, 45, 0,
0...
## $ trip_end_year              <dbl> 2019, 2019, 2019, 2019, 2019, 2019, 201
9...
## $ trip_end_month             <ord> Jan, Jan, Jan, Jan, Jan, Jan, Jan, Jan,
...
## $ trip_end_day               <int> 22, 19, 23, 23, 23, 15, 10, 7, 15, 17,
2...
## $ trip_end_hour              <int> 15, 22, 19, 7, 0, 20, 6, 16, 13, 12, 19
,...
## $ trip_end_minute            <int> 30, 30, 30, 30, 0, 15, 30, 30, 45, 0, 1
5...
## $ trip_minutes               <dbl> 7.616667, 7.633333, 12.000000, 4.933333
,...
```

Data Explorations and Inferences

First, we have explored our data and tried to understand the trends, anomalies and consistent information our data provides us. This exercise helped us understand the data and the inferences we made can be useful to us to prepare the business plan for our taxi service

After we explored our dataset, we have encountered some anomalies such as there are trips that have covered more than 100 miles but have the trip duration as 0 minutes, which is unusual. So we have used the below filter criteria to remove such anomalies from our dataset.

Filter criteria

```
trips_2019 <- trips_2019_1 %>%
  filter(trip_miles < 100) %>%
  filter(trip_minutes != 0) %>%
  filter(trip_miles != 0) %>%
```

```
  filter(fare != 0) %>%
  filter(fare < 400) %>%
  filter(tips <= 50)
```

Let's look at our top 5 competitors first

Top 5 companies in descending order of the number of trips made

```
top_5_companies <- trips_2019 %>%
  group_by(company) %>%
  summarise(
    count = n()
    ) %>%
  arrange(desc(count)) %>%
  head(5)
```

As we can see from the output the taxi affiliation services, flash cab, Chicago carriage cab corporation and so on has the highest count of trips. Looking, at these companies' strategies we can adopt the same to be successful in the market.

Let's look at our competitors' average trip_total throughout the year

Reporting the average trip_total of top5 companies in the year 2019

```
trips_2019 %>%
  filter(!is.na(trip_total)) %>%
  filter(company == top_5_companies$company) %>%
  group_by(company) %>%
  summarise(
    Avg_trip_total = mean(trip_total)
    ) %>%
  arrange(desc(Avg_trip_total))
```

```
## # A tibble: 5 x 2
##   company                    Avg_trip_total
##   <chr>                               <dbl>
## 1 Sun Taxi                             19.5
## 2 Taxi Affiliation Services            19.0
## 3 Flash Cab                            18.3
## 4 Chicago Carriage Cab Corp            18.0
## 5 Medallion Leasin                     17.5
```

The above query is used to discover the average trip total of the top 5 taxi companies. The trip total includes the fares, tips, tolls, and extras. This is valuable information to our company as it allows to analyze some of our most profitable competition. We can see that the top earners are Sun Taxi with an average trip_total of 19.0 per trip followed by other companies. These companies may provide more service to and from airports which would explain the higher average trip total. We as a company could use this information to assist in determining pricing for these longer trips.

After we explored the trip_total we looked at the total taxis each of these companies own and the average fares and average miles

Number of taxis each of the top5 companies are providing and number of total trips made by these taxis and their average fares and average miles

```
trips_2019 %>%
  group_by(company)%>%
  summarise(
    Total_taxi = n_distinct(taxi_id),
    Total_trips_made = n(),
    Avg_fare = mean(fare, na.rm = T),
    Avg_miles = mean(trip_miles, na.rm = T)
    ) %>%
  arrange(desc(Total_trips_made)) %>%
  filter(company == top_5_companies$company) %>%
  head(5)
```

```
## # A tibble: 5 x 5
##   company                    Total_taxi Total_trips_made Avg_fare Avg_miles
##   <chr>                           <int>            <int>    <dbl>     <dbl>
## 1 Taxi Affiliation Services        1027           147891     15.6      3.38
## 2 Flash Cab                         776           147250     15.6      4.92
## 3 Chicago Carriage Cab Corp         557           108915     14.6      4.41
## 4 Sun Taxi                          396            73382     15.5      4.78
## 5 Medallion Leasin                  345            68454     14.3      4.27
```

Using the above query, we are able to find the number of taxis that the top 5 companies are utilizing. It also tells us the total trips those taxis made along with their average fare and miles. This information will help us estimate the amount of taxis that we will want to employ in our fleet. It also helps estimate, based on the number of taxis that we have deployed, what the average fare and miles might be Which is useful from a financial standpoint, but also a vehicle maintenance perspective too.

After analyzing the outputs of the above couple of queries, we reached a conclusion that the market trend is that the top companies are maintaining an average fare between $13 to $15 for an average distance of 3 to 5 miles. Looking at this market trend we decided to make a smart competitive move by setting an average fare of around $13 for an average distance of 5 miles which is slightly lower than the trend.

Let's get a little into the details, we now look at the average trips per day each of these top5 companies make.

Average trips per day made by the top 5 companies

```
trips_2019 %>%
  group_by(company) %>%
  summarise(
    avg_trips_per_day = n()/365
  ) %>%
```

```
  arrange(desc(avg_trips_per_day)) %>%
  filter(company == top_5_companies$company)
```

```
## # A tibble: 5 x 2
##   company                    avg_trips_per_day
##   <chr>                                  <dbl>
## 1 Taxi Affiliation Services               405.
## 2 Flash Cab                               403.
## 3 Chicago Carriage Cab Corp               298.
## 4 Sun Taxi                                201.
## 5 Medallion Leasin                        188.
```

The average number of trips per day made by our competitors is between 200 to 600 trips. By looking at these we can fix a goal for our company to make an average of 200 trips per day. So, in order to make 300 trips a day we would like to maintain 30 active taxis a day, to achieve our goal.

Now, let's dig deeper and look at the average trip_total these companies make per day.

Average trip_total by top five competitors per day

```
trips_2019 %>%
  filter(company == top_5_companies$company) %>%
  group_by(company) %>%
  summarise(
    avg_trip_total_per_day = sum(trip_total, na.rm = T)/365
  ) %>%
  arrange(desc(avg_trip_total_per_day))
```

```
## # A tibble: 5 x 2
##   company                    avg_trip_total_per_day
##   <chr>                                       <dbl>
## 1 Taxi Affiliation Services                    1519.
## 2 Flash Cab                                    1479.
## 3 Chicago Carriage Cab Corp                    1069.
## 4 Sun Taxi                                      792.
## 5 Medallion Leasin                              664.
```

Looking at the output, it is pretty evident that the company that has highest average number of trips per day is making the more money per day. The consolidated conclusion from all the above queries is that the

After we know who are competitors are and what their statistical summary of income terms is we shifted our focus in looking at the data year-wise. We looked at the prominent variables and their summary statistics throughout the year.

Let's start with our prime variable fare, lets look at the average, median and standard deviation of the fare variable throughout the year.

What are the average, median and standard deviation fares earned by trips per month?

```
trips_2019 %>%
  group_by(trip_start_month) %>%
  summarise(
    Avg_fare_per_month = mean(fare, na.rm = T),
    Median_fare_per_month = median(fare, na.rm = T),
    Sd_fare_per_month = sd(fare, na.rm = T)
    )
```

```
## # A tibble: 12 x 4
##    trip_start_month Avg_fare_per_month Median_fare_per_month Sd_fare_per_m
onth
##    <ord>                         <dbl>                 <dbl>             <
dbl>
##  1 Jan                            13.9                  8.25
13.4
##  2 Feb                            14.0                  8.25
13.2
##  3 Mar                            14.4                  8.5
13.8
##  4 Apr                            15.2                  8.5
14.4
##  5 May                            15.8                  9
14.6
##  6 Jun                            15.8                  9
14.7
##  7 Jul                            15.2                  8.75
14.2
##  8 Aug                            15.4                  9
14.3
##  9 Sep                            16.1                  9.25
14.8
## 10 Oct                            16.1                  9
14.8
## 11 Nov                            15.2                  8.75
14.1
## 12 Dec                            14.2                  8.5
13.3
```

Looking at the result, we can clearly come to a conclusion that the average fare per month is between $13 and $15.Now this can be because our taxis are the local taxis in the city of Chicago and it is obvious that the taxis are used for shorter trips and so the average fare too is less than $20.

After we looked at the fare, let's take a look if the average trip distance per month is small or not confirming that the trips are short distance trips.

What are the average, median and standard deviation trip_miles travelled by trips per month

```
trips_2019 %>%
  group_by(trip_start_month) %>%
  summarise(
    Avg_trip_miles_per_month = mean(trip_miles, na.rm = T),
    Median_trip_miles_per_month = median(trip_miles, na.rm = T),
    Sd_trip_miles_per_month = sd(trip_miles, na.rm = T)
    )
```

```
## # A tibble: 12 x 4
##    trip_start_month Avg_trip_miles_per_~ Median_trip_miles_p~ Sd_trip_mile
s_per~
##    <ord>                          <dbl>                <dbl>
<dbl>
##  1 Jan                             3.94                 1.41
5.66
##  2 Feb                             3.94                 1.43
5.58
##  3 Mar                             4.10                 1.5
5.75
##  4 Apr                             4.39                 1.6
6.01
##  5 May                             4.57                 1.64
6.12
##  6 Jun                             4.55                 1.66
6.14
##  7 Jul                             4.28                 1.59
5.94
##  8 Aug                             4.35                 1.6
5.96
##  9 Sep                             4.68                 1.7
6.24
## 10 Oct                             4.63                 1.6
6.23
## 11 Nov                             4.28                 1.52
5.94
## 12 Dec                             3.89                 1.48
5.54
```

The output of the query confirms that the average distance covered by the taxis per month is around 3.5 miles adding more support to the statement that the trips that taxi's cover in this dataset are short trips. As a taxi company we should keep in mind that we should be have active taxis and expect most revenue from the short trips.

Now, that we confirmed that we have more short trips, let's look at the long trips, short trips and medium trips based on the distance the trips cover.

As per our company criteria we have segregated shot trips as the trips that travelled less than or equal to 10 miles, medium duration trips between 10 to 25 miles and longer duration trips above 25 miles.

Display the number of short, medium and long duration trips per month.

```
trips_2019 %>%
  mutate(trip_duration = ifelse(trip_miles <= 10, "short duration trips",
                          ifelse(trip_miles > 10 & trip_miles <= 25, "medium
duration trips", "long duration trips"))) %>%
  group_by(trip_start_month, trip_duration) %>%
  count() %>%
  ungroup() %>%
  mutate(
    frac = n/sum(n)
  ) %>%
  arrange(desc(frac))
```

```
## # A tibble: 36 x 4
##    trip_start_month trip_duration            n   frac
##    <ord>            <chr>                <int>  <dbl>
##  1 Jan              short duration trips 62092 0.0720
##  2 Feb              short duration trips 62033 0.0719
##  3 Mar              short duration trips 61692 0.0715
##  4 Dec              short duration trips 61466 0.0713
##  5 Jul              short duration trips 60523 0.0702
##  6 Apr              short duration trips 60318 0.0699
##  7 Aug              short duration trips 59914 0.0695
##  8 Jun              short duration trips 59775 0.0693
##  9 May              short duration trips 59702 0.0692
## 10 Nov              short duration trips 59631 0.0691
## # ... with 26 more rows
```

From the output, it is clear that 73.8% trips are short duration, our company should achieve the target by taking up more short trips which in turn will result in more revenue.

We looked at individual months, lets combine these months into quarters. We divided the 12 months into 4 quarters, so we have three months per quarter.

Initially, let's look at the average of trip_total in each quarter.

Calculate the average of trip_total for each quarter and arrange results in descending order.

```
trips_2019 %>%
  mutate(quarter = ifelse(trip_start_month =="Jan"|trip_start_month =="Feb"|t
rip_start_month =="Mar","quarter1",
                   ifelse(trip_start_month =="Apr"| trip_start_month =="May"|
trip_start_month =="Jun" , "quarter2",
                   ifelse(trip_start_month =="Jul"|trip_start_month =="Aug"|t
rip_start_month =="Sep", "quarter3","quarter4")))) %>%
```

```
  group_by(quarter) %>%
  summarise(
    avg_mean_per_quarter = mean(trip_total, na.rm = T)
    ) %>%
  arrange(desc(avg_mean_per_quarter))
```

```
## # A tibble: 4 x 2
##   quarter  avg_mean_per_quarter
##   <chr>                   <dbl>
## 1 quarter2                 19.1
## 2 quarter3                 19.0
## 3 quarter4                 18.4
## 4 quarter1                 17.2
```

Looking at the average revenue earned in each quarter. We can see that the average range is between $16 to $18.This query will give us a better edge in the decision regarding which quarter to launch the business in. Based on this data, in the year 2019 quarter two is slightly higher revenue. That is the three months April, May and June. This might be due to the significant change in the weather of Chicago. The weather in these months is good and people are seen outdoor more often.

After we divided the year into quarters, we took a leap further and divided the year into weekdays and weekends. We came up with the analysis that will help us analyze one of our prime purpose of the project the revenue better.

Let's start by counting the trips on weekdays and weekends.

What is the average number of trips per weekday and per weekend?

```
trips_2019 %>%
  mutate(weekend = ifelse(trip_start_weekday == "Sat" | trip_start_weekday ==
"Sun", 1, 0)) %>%
  group_by(trip_start_month, trip_start_day) %>%
  select(trip_start_month, trip_start_day, weekend) %>%
  filter(weekend == 0) %>%
  summarise(
    trips_per_weekday = n()
    ) %>%
  ungroup() %>%
  summarise(
  avg_trips_per_weekday = mean(trips_per_weekday),
  )
```

```
## # A tibble: 1 x 1
##   avg_trips_per_weekday
##                   <dbl>
## 1                 2621.
```

```
trips_2019 %>%
  mutate(weekend = ifelse(trip_start_weekday == "Sat" | trip_start_weekday ==
"Sun", 1, 0)) %>%
```

```
  group_by(trip_start_month, trip_start_day) %>%
  select(trip_start_month, trip_start_day, weekend) %>%
  filter(weekend == 1) %>%
  summarise(
    trips_per_weekend = n()
    ) %>%
  ungroup() %>%
  summarise(
    avg_trips_per_weekend = mean(trips_per_weekend)
  )
```

```
## # A tibble: 1 x 1
##   avg_trips_per_weekend
##                   <dbl>
## 1                 1714.
```

From the above results, it is evident that average trips per day on a weekday are higher than the trips on weekends. From this we as a company can learn that we need to set more active cabs on the weekdays to earn better revenue and also become famous and get competitive advantage.

We did not want to stop by just looking at the count of the average number of trips on weekdays and weekends, we also wanted to go ahead and find out the average trip_total earned on each day.

What is the average number of trip_total on each day of the week.

```
trips_2019 %>%
  group_by(trip_start_weekday) %>%
  summarise(
    avg_trip_total = mean(trip_total, na.rm = T)
    )
```

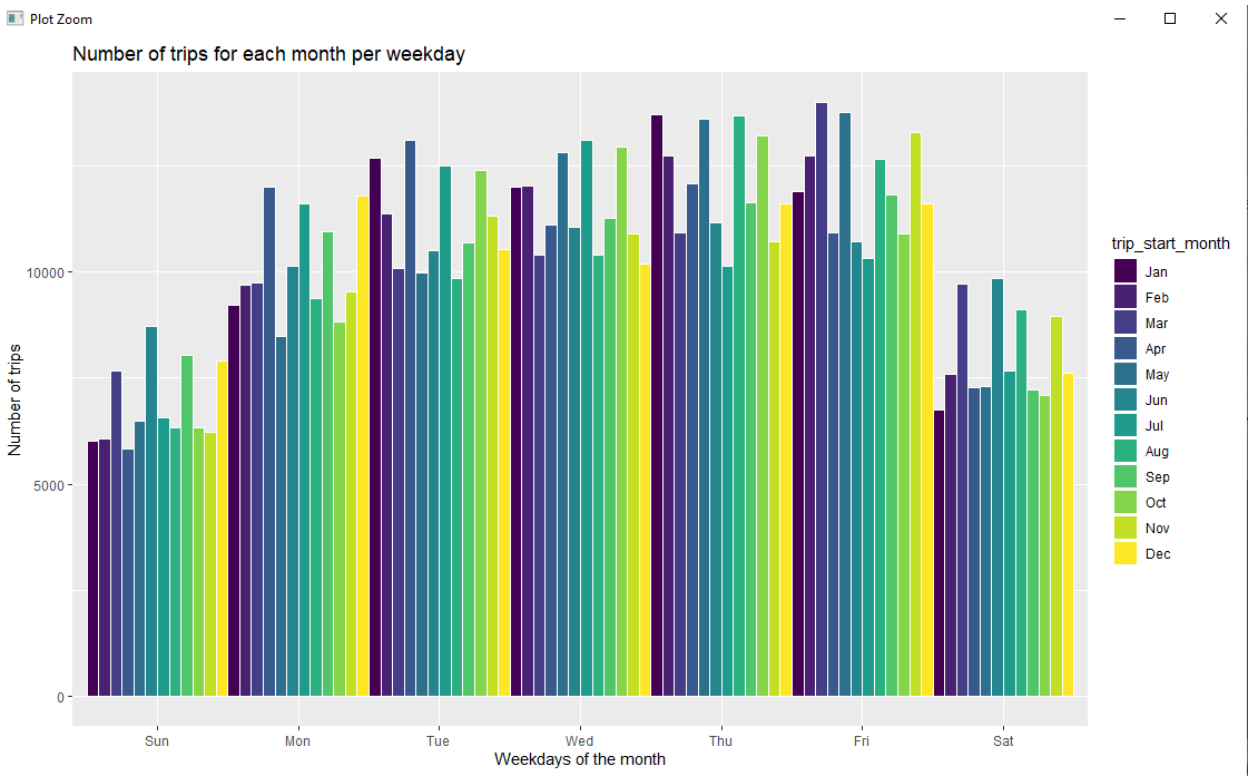
```
## # A tibble: 7 x 2
##   trip_start_weekday avg_trip_total
##   <ord>                       <dbl>
## 1 Sun                          21.6
## 2 Mon                          19.5
## 3 Tue                          17.9
## 4 Wed                          18.2
## 5 Thu                          18.4
## 6 Fri                          17.5
## 7 Sat                          16.8
```

From the result, we can deduct that the average trip total is highest for Sunday, which makes sense because weekends especially Sundays are the time when people like to go outdoors and take taxi rides to different places to spend their weekends with family and friends.

After we looked at the average trip total per week, we were curious to visualize the variation of number of trips for all months per weekday.

Visualizing number of trips for each month per weekday

```
trips_2019 %>%
  filter(!is.na(trip_start_month)) %>%
  filter(!is.na(trip_start_weekday)) %>%
  group_by(trip_start_month, trip_start_weekday) %>%
  summarise(
    total = n()
  ) %>%
  ggplot(aes(trip_start_weekday, total)) +
  geom_col(aes(fill = trip_start_month), stat ="identity", width = 1,color="w
hite",  position=position_dodge())+
  labs(x="Weekdays of the month",
       y="Number of trips",
       title = "Number of trips for each month per weekday"
       )
```



The above graph is both informative and easy to understand, the graph explains the variation in number of trips, it helps us analyze which day in a week has more prominence in every month. This data is useful for our company to judge which days in a month are the

most beneficial for us as a company, based on this we can decide our active fleet for those days. This analysis will also help us be prepared for rush days of the month.

For example, from the graph we can infer that in the month of January there are more number of trips on Tuesdays and Thursdays, so by this analysis we can keep our drivers informed about these opportunity days where there will be more trips and we as a company can generate more revenue and drivers can try their luck in accruing more tips.

We now know our competitors, we know their revenue, we have made our conclusions based on variables analyzed per month, per quarter and also based on weekday and weekends. Since we are most interested in the revenue made we will take a closer look at the payment type.

Query on number of payments per payment type.

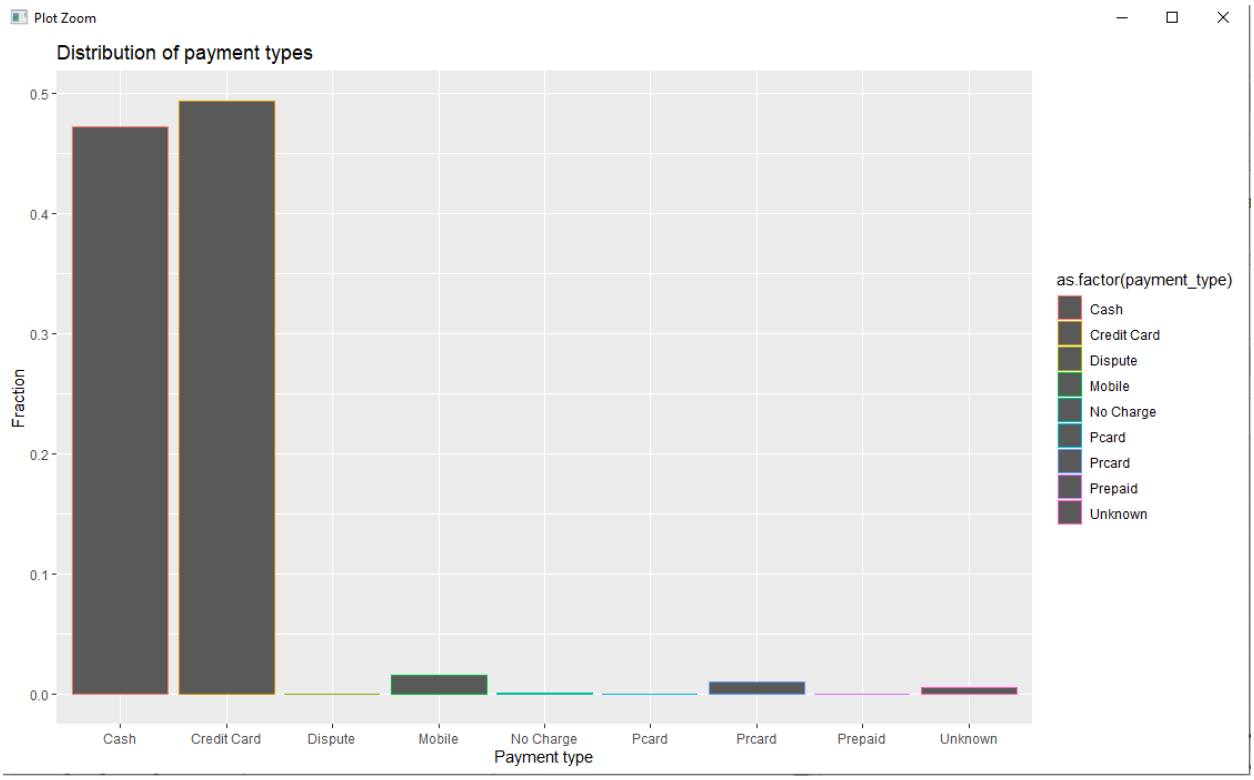
```
trips_2019 %>%
  group_by(payment_type) %>%
  count() %>%
  ungroup() %>%
  mutate(
    frac=n/sum(n)
    )
```

```
## # A tibble: 9 x 3
##   payment_type      n      frac
##   <chr>         <int>     <dbl>
## 1 Cash         406890 0.472
## 2 Credit Card  425566 0.493
## 3 Dispute         320 0.000371
## 4 Mobile        14147 0.0164
## 5 No Charge      1359 0.00158
## 6 Pcard            26 0.0000301
## 7 Prcard         9087 0.0105
## 8 Prepaid          29 0.0000336
## 9 Unknown        4963 0.00575
```

Assessing the above results visually

```
trips_2019 %>%
  group_by(payment_type) %>%
  count() %>%
  ungroup() %>%
  mutate(
    frac=n/sum(n)
    )%>%
  ggplot(mapping = aes(y=frac,x=as.factor(payment_type),color=as.factor(payme
nt_type)))+
  geom_histogram(stat = "identity")+
  labs(
    x="Payment type",
```

```
    y="Fraction",
    title="Distribution of payment types"
    )
```



So, basically the above queries help us to decide which is the most popular payment mode in the dataset. This information is valuable to our company because based on this we will decide which payment modes we need to enable for our taxis Looking at the fractions we can draw a conclusion that the most popular two payment modes are Cash and credit card both nearing to around 49% each. This result indicates us that we need to have these two payment modes for sure.

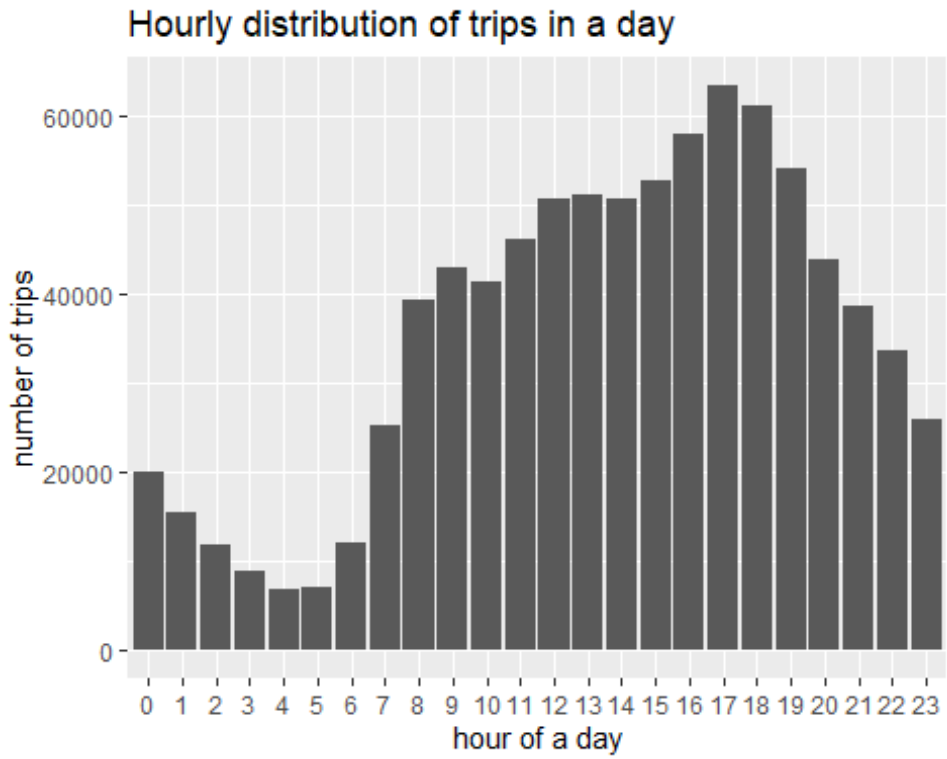
Now that we are following a logical flow of looking at our dataset, we looked at competitors, their summary statistics, then we looked at per month analysis of our prime variables and we looked at payment type. Let's dwell deeper and look at the per hour analysis.

Let's look at the number of trips per each hour of the day.

Calculate the number of trips in each hour of a day?

```
trips_2019 %>%
  group_by(trip_start_month,trip_start_hour) %>%
  summarise(
    n_trips = n()
  ) %>%
  arrange(desc(n_trips))%>%
  ggplot(mapping = aes(x=as.factor(trip_start_hour),y=n_trips))+
```

```
  geom_col() +
  labs(
    x="hour of a day",
    y="number of trips",
    title="Hourly distribution of trips in a day"
  )
```



In the above code we used the arrange at the end because we want to know the busiest hour in the day. As we can see most of the trips starting at 5pm in the evening or 6pm in the evening are prominent as this is the time most of offices close. So, by this we have to make sure to have enough active taxi's at these time so that we can generate more revenue.

After we looked at the busiest hour and found that it is in the evening, we were curious and we wanted to know the number of active taxis at different times of the day, say morning, noon and so on.

Distribution of trips over a day

```
trips_2019 %>%
  mutate(time_of_day = (ifelse(trip_start_hour > 4 & trip_start_hour <= 8, "e
arly morning",        ifelse(trip_start_hour > 8 & trip_start_hour <= 12, "mo
rning",              ifelse(trip_start_hour > 12 & trip_start_hour <= 16, "no
on",                 ifelse(trip_start_hour > 16 & trip_start_hour <= 20, "ev
ening",              ifelse(trip_start_hour > 00 & trip_start_hour <= 4, "mid
night", "night")))))) %>%
  group_by(time_of_day) %>%
  count() %>%
```

```
  ungroup() %>%
  mutate(frac=n/sum(n))
```

```
## # A tibble: 6 x 3
##   time_of_day        n   frac
##   <chr>          <int>  <dbl>
## 1 early morning  83755 0.0971
## 2 evening       223119 0.259
## 3 midnight       42948 0.0498
## 4 morning       181369 0.210
## 5 night         118257 0.137
## 6 noon          212939 0.247
```

As a team we came up with this threshold, because that is how the day is divided and this kind of division is easy to understand because it is intuitive.

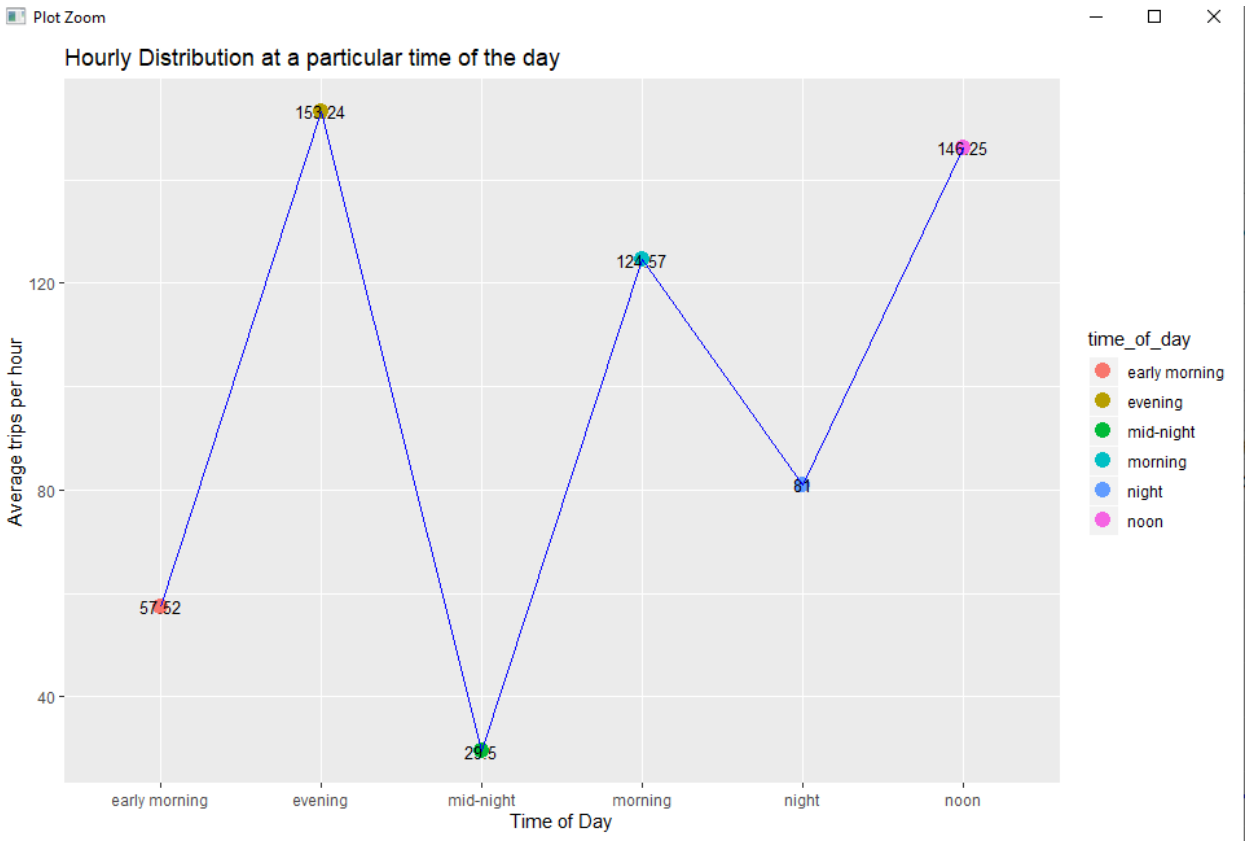
This query helps us discover the number of trips in a fraction (%) that occurred during the different "parts" of a day. We split the day up into five different parts. Early morning, which occurs from 4 AM to 8 AM, Morning, which occurs from 8AM to 12PM, Noon, which is 12PM to 4PM, Evening, which is 4PM to 8PM, and Night which is from 8PM to 12PM. What we found is the spread of trips throughout the day. 0.0588 or 5.88% of trips occurred in the early morning and 0.256 or 25.6% of trips occurred in the morning. At noon, 0.247 or 24.7% of trips occurred and during the evening 0.257 or 25.7% of trips occurred. 11.4% occurred during the night period and 6.73% of rides occurred during the else period named "Midnight". This information is very valuable to us as a company as it allows us to see when the most rides are occurring. It can be seen that the most advantageous time for the company to deploy a large majority of its feet is during the Morning, noon, and evening periods. Night, early morning, and midnight periods are still valuable times to have the fleet deployed, but not important enough to have the entire fleet operating.

Based on the time of day (morning, afternoon, so on) we were curious to find out the average number of trips per hour at a particular time of the day. Here, we have managed to get to the very deep detail, for example from the below query we will be able to judge how many taxis should be active at particular hour in that time of the day (Morning, afternoon, so on)

Hourly distribution at a particular time in the day

```
trips_2019 %>%
  mutate(time_of_day = (ifelse(trip_start_hour > 4 & trip_start_hour <= 8, "e
arly morning",
                        ifelse(trip_start_hour > 8 & trip_start_hour <= 12, "
morning",
                        ifelse(trip_start_hour > 12 & trip_start_hour <= 16,
"noon",
                        ifelse(trip_start_hour > 16 & trip_start_hour <= 20,
"evening",
                        ifelse(trip_start_hour > 0 & trip_start_hour <= 4, "m
id-night", "night")))))) %>%
```

```r
group_by(trip_start_month, trip_start_day, time_of_day) %>%
summarise(
  number_trips_per_day_per_time=n()
) %>%
ungroup() %>%
group_by(time_of_day) %>%
summarise(
  count_of_days = n(),
  sum_of_trips = sum(number_trips_per_day_per_time)
) %>%
mutate(
  avg_trips_per_hour = (sum_of_trips/count_of_days)/4
) %>%
ggplot(aes(as.factor(time_of_day),
           avg_trips_per_hour,
           group=1)) +
geom_point(aes(color = time_of_day),
           size = 4,
           show.legend = T) +
geom_text(aes(label = round(avg_trips_per_hour,2)),
          size = 3.5) +
geom_line(color="blue") +
labs(
  x = "Time of Day",
  y = "Average trips per hour",
  title ="Hourly Distribution at a particular time of the day"
)
```



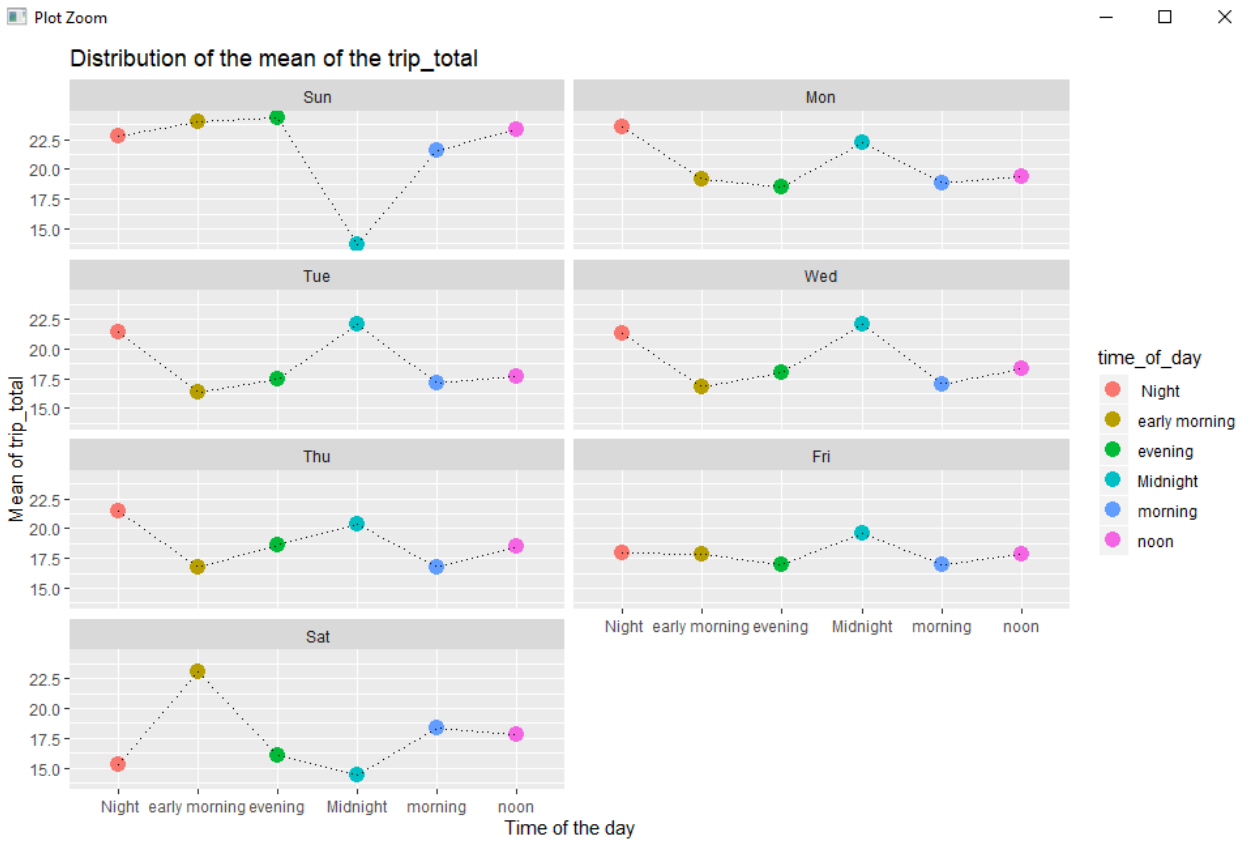
Looking at the output graph, we can clearly understand that there are more average number of trips that is 153.24 in the evening followed by 124.57 average trips in the morning this information is valuable to a startup like us because this indicates us that we need to maintain our full potential taxis during these rush hours.

After we explored the trips according to the time of the day and the hourly distribution we wanted to visualize the distribution of mean trip_total earned for every time period of the day (Morning, afternoon, so on) for all days in a week.

Distribution of the mean of the trip_total

```
trips_2019 %>%
  filter(!is.na(trip_miles)) %>%
  mutate(time_of_day = (ifelse(trip_start_hour > 4  & trip_start_hour <= 8, "
early morning",        ifelse(trip_start_hour > 8  & trip_start_hour <= 12, "
morning",              ifelse(trip_start_hour > 12 & trip_start_hour <= 16, "
noon",                 ifelse(trip_start_hour > 16 & trip_start_hour <= 20, "
evening",              ifelse(trip_start_hour > 0 & trip_start_hour <= 4, "Mi
dnight", " Night")))))) %>%
  group_by(trip_start_weekday, time_of_day) %>%
  summarise(
    mean_trip_miles = mean(trip_total, na.rm = T)
  ) %>%
  ggplot(aes(time_of_day, mean_trip_miles)) +
  geom_point(aes(color = time_of_day), size = 4,
```

```
            show.legend = T) +
  geom_line(aes(group = 1), linetype = 'dotted') +
  facet_wrap(~trip_start_weekday, nrow = 2)+
  labs(
    x="Time of the day",
    y="Mean of trip_total",
    title = "Distribution of the mean of the trip_total"
  )
```



Looking at the output, we can infer that that the highest mean of trip_total is generated on Sunday nights, this is completely intuitive and no brainer because Sunday evenings are the time when people in Chicago spend their time in restaurants or malls having a fun time.

Besides, Sunday nights comparatively Sunday on the whole has high values of trip_total when compared to other days. Also, one more stark observation that we made is the distribution follows a similar pattern for Monday to Fridays. Whereas, the pattern on Saturday and Sunday is a little different. So, this for our company indicates that at these times our drivers need to be proactive and take a greater number of trips because there is an opportunity to earn more revenue.

Let's, look at some other interesting trips now. One interesting trip to look at can be round trips.

Round trips-The trip that has pick_up_location and drop_off_location is same then that particular trip is called round trip.

Now, let's explore the our prime variables, that is trip_miles and trip duration for the top and bottom three round trips based on the duration of the trip.

Report the top 3 and bottom 3 round trips based on trip duration

```
trips_2019 %>%
  filter(pickup_centroid_latitude == dropoff_centroid_latitude &  pickup_cent
roid_longitude == dropoff_centroid_longitude) %>%
  select(trip_minutes,trip_miles,fare,tips) %>%
  arrange(desc(trip_miles)) %>%
  head(3)
```

```
## # A tibble: 3 x 4
##   trip_minutes trip_miles  fare  tips
##          <dbl>      <dbl> <dbl> <dbl>
## 1         203.       95.9  236.  35.5
## 2         202.       80.1  218    0
## 3         132.       73.9  182.  39.0
```

```
trips_2019 %>%
  filter(pickup_centroid_latitude == dropoff_centroid_latitude &  pickup_cent
roid_longitude == dropoff_centroid_longitude) %>%
  select(trip_minutes,trip_miles,fare,tips) %>%
  arrange(desc(trip_miles)) %>%
  tail(3)
```

```
## # A tibble: 3 x 4
##   trip_minutes trip_miles  fare  tips
##          <dbl>      <dbl> <dbl> <dbl>
## 1        8.18        0.01  5.75     0
## 2        0.683       0.01  3.25     0
## 3        0.717       0.01  3.25     0
```

This query helps us discover the trips with the highest trip minutes and the top 3 trips with the lowest amount of trip minutes. There are two trips past the three-hour mark. Those trips both have a fare over $200.It is interesting to see one of those long trips has a high tip and the second one doesn't. The customer could have paid by card and just opted to not give one. Looking at the lowest amounts, there are explanations for these occurrences. What could have occurred is the passenger got into the vehicle, decided that it was not what they wanted, and got out. Some taxi companies do start charging as soon as you get in the vehicle. These fees could have also occurred due to the cab being called and then the customer canceling it which would result in a fee. This information is very valuable to us as it helps us determine our cancel fare fee and it also gives us insight to the cost of higher minute/mile trips.

After we looked at the interesting round trip query, we shifted our focus and started to look at trips on special occasions or on special days.

One famous weekend in Chicago at the time of spring is the St. Patrick's weekend. We wanted to explore the average revenue the companies earned in this weekend.

What is the average fare and tips during St. Patrick's day weekend

```
trips_2019 %>%
  filter(trip_start_month == "Mar") %>%
  filter(trip_start_day == 16 | trip_start_day == 17) %>%
  summarise(
    Number_of_trips = n(),
    Avg_Fare = mean(fare, na.rm = T),
    Avg_Tips = mean(tips, na.rm = T)
    )
```

```
## # A tibble: 1 x 3
##   Number_of_trips Avg_Fare Avg_Tips
##             <int>    <dbl>    <dbl>
## 1            4503     13.7     1.62
```

This query helps us explore the average fair and tips during St. Patrick's Day weekend. St. Patrick's Day weekend in Chicago is a world-renowned holiday which also happens to be a very high taxi usage time. This is more related to the fact that alcohol is one of the center points of the holiday. So, we wanted to see what the average fare and average tips are for those days are. It came out to be that the average fare is 13.22 and the average tip is 1.56. This information is especially useful for encouraging our employees to work during these days as they have the opportunity to earn more as there is a higher volume of trips with a good fare and tip available.

After St. Patrick's day, we explored the holiday week that is the thanksgiving week, in this week many people travel and shop and this is the time for all taxi services to make a lot of money.

Total number of trips during Thanksgiving Week.

```
trips_2019 %>%
  filter(trip_start_month == "Nov") %>%
  filter(trip_start_day >= 24 | trip_start_day <= 30) %>%
  summarise(
    number_trips = n(),
    Avg_Fare = mean(fare, na.rm = T),
    Avg_Tips = mean(tips, na.rm = T)
    )
```

```
## # A tibble: 1 x 3
##   number_trips Avg_Fare Avg_Tips
##          <int>    <dbl>    <dbl>
## 1        70914     15.2     1.98
```

The above query is being used to find the total number of trips during Thanksgiving week. Thanksgiving is one of the busiest times of the year where tens of millions of Americans

travel to where their families are. The query tells us that there were 84,484 trips utilizing taxi's in Chicago during this time. The average fare is 14.3 and the average of the tips is 1.84, which is slightly higher than the results in the St. Patrick's weekend. This is valuable to us because, as a company planning to start a taxi business in the city, it is important to have data that shows us the amount of taxi usage during this time period. We can take this data and use it as basis for next year and whether our fleet will be able to meet the demand or if we need to expand.

After thanksgiving the next big holiday season is Christmas week, we did a different kind of analysis, instead of looking at the fare, trip_miles we looked at the tips column. Since people believe in charity and goodwill during the Christmas to New Years' time period, we wanted to explore if this is the time people were being generous and tipping the taxi rides higher or not.

Count of amount tipped from Christmas to New Years

```
trips_2019 %>%
  filter(trip_start_month == "Dec") %>%
  filter(trip_start_day >= 24 | trip_start_day <= 31) %>%
  group_by(tips) %>%
  summarise(count_of_tips_dec = n()) %>%
  arrange(desc(count_of_tips_dec))
```

```
## # A tibble: 881 x 2
##     tips count_of_tips_dec
##    <dbl>             <int>
##  1   0               37614
##  2   2               10757
##  3   3                3830
##  4   1                2487
##  5   4                1442
##  6   5                 675
##  7   1.5               539
##  8  10                 304
##  9   2.5               242
## 10   2.1               239
## # ... with 871 more rows
```

The above query is used to find the count of each amount tipped from Christmas to New Years. As can be seen above, there was a total of 46,400 customers who didn't tip any amount. Outside of customers just not tipping any amount, this could be the result of international travelers who do not know how tipping in the US works. The lack of knowledge here could help explain why the number of 0-dollar tips is so high as this time of year is the busiest travel season in the world. The next highest amount tipped is 2.00 dollars by 12,534 customers which is followed by 3.00 dollars and then 1.00 dollar. Outside of these amounts being useful information for accounting and tax purposes when it pertains to the business, it also is good information to share with employees and potential employees. This gives them the opportunity to estimate what they would be able to make and can also help encourage them to have excellent customer service.

After we explored the data based on the variables such as trip_miles, duration and special days, we decided to shift our focus to visualizing the pick-up and drop off locations of trips according to area codes in Chicago.

Distribution of various pickup and drop-off regions

In the below query we have divided the different pickups from each community areas into 9 different regions according to Chicago map then we have calculated the total number of pickups done each region.

```
community_pickups<-trips_2019 %>%
  filter(!is.na(pickup_community_area)) %>%
  transmute(pick_up_community_area =
              (ifelse(between(pickup_community_area,1,4)  | between(pickup_co
mmunity_area,9,14) |between(pickup_community_area,76,77), "Far North Side",
               ifelse(between(pickup_community_area,15,20),"North West Side",
               ifelse(between(pickup_community_area,5,7)|         between(pic
kup_community_area,21,22),"North Side",
               ifelse(between(pickup_community_area,23,31),"West side",
               ifelse(between(pickup_community_area,32,33)|pickup_community_a
rea==8,"Central ",
               ifelse(between(pickup_community_area,34,43)  | pickup_communit
y_area==60|pickup_community_area==69,"South side",
               ifelse(between(pickup_community_area,56,59)  | between(pickup_
community_area,61,68),"South west side",
               ifelse(between(pickup_community_area,70,75),"Far south west si
de","Far south east side"))))))))) %>%
  group_by(pick_up_community_area) %>%
  summarise(n=n())
```

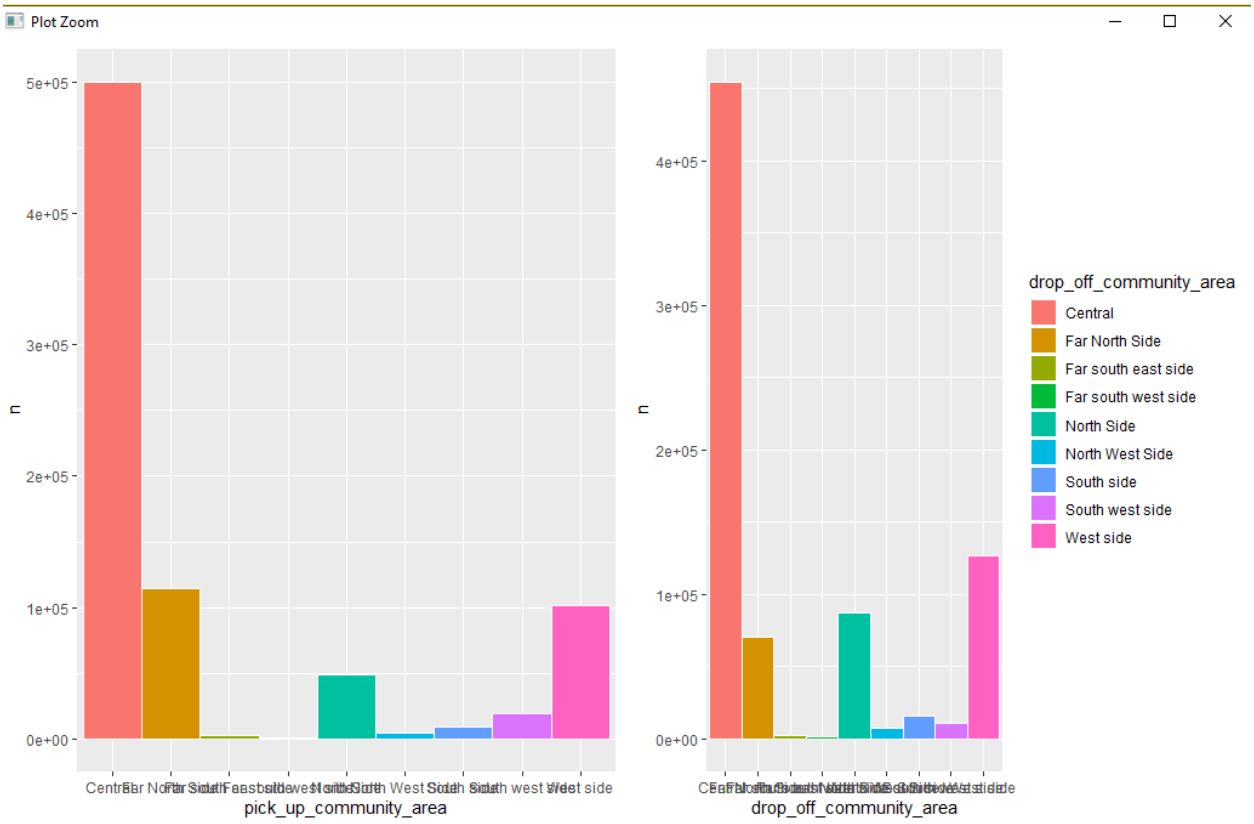
Similarly, we have done the same division for drop-off community areas

```
community_dropoffs<-trips_2019 %>%
  filter(!is.na(dropoff_community_area)) %>%
  transmute(drop_off_community_area=
              (ifelse(between(dropoff_community_area,1,4)  | between(dropoff_
community_area,9,14) |between(dropoff_community_area,76,77), "Far North Side"
,
              ifelse(between(dropoff_community_area,15,20),"North West Side",
              ifelse(between(dropoff_community_area,5,7)  | between(dropoff_c
ommunity_area,21,22),"North Side",
              ifelse(between(dropoff_community_area,23,31),"West side",
              ifelse(between(dropoff_community_area,32,33)|dropoff_community_
area==8,"Central ",
               ifelse(between(dropoff_community_area,34,43)  | dropoff_commun
ity_area==60|dropoff_community_area==69,"South side",
               ifelse(between(dropoff_community_area,56,59)  | between(dropof
f_community_area,61,68),"South west side",ifelse(between(dropoff_community_ar
ea,70,75),"Far south west side","Far south east side"))))))))) %>%
```

```
  group_by(drop_off_community_area) %>%
  summarise(n=n())
```

The above results are presented visually using bar-charts as shown below.

```
plot1 <- ggplot(community_pickups,
                aes(x=pick_up_community_area, y=n, fill=pick_up_community_area)
) +   geom_col(stat="identity", width=1, color="white",show.legend = F)

plot2 <- ggplot(community_dropoffs,
                 aes(x=drop_off_community_area, y=n, fill=drop_off_community_a
rea)) +
  geom_col(stat="identity", width=1, color="white")

gridExtra::grid.arrange(plot1, plot2, nrow = 1)
```



Now here we can see the bar charts show total no of pickups and drop offs done in each region. Looking at the bar chart it is evident that the highest number of pickups and drop offs are done in the “Central region”. Thus, if our company wants to generate good amount of revenue it has to make sure that at least 60% to 70% of the taxis should be active every time in the central region.

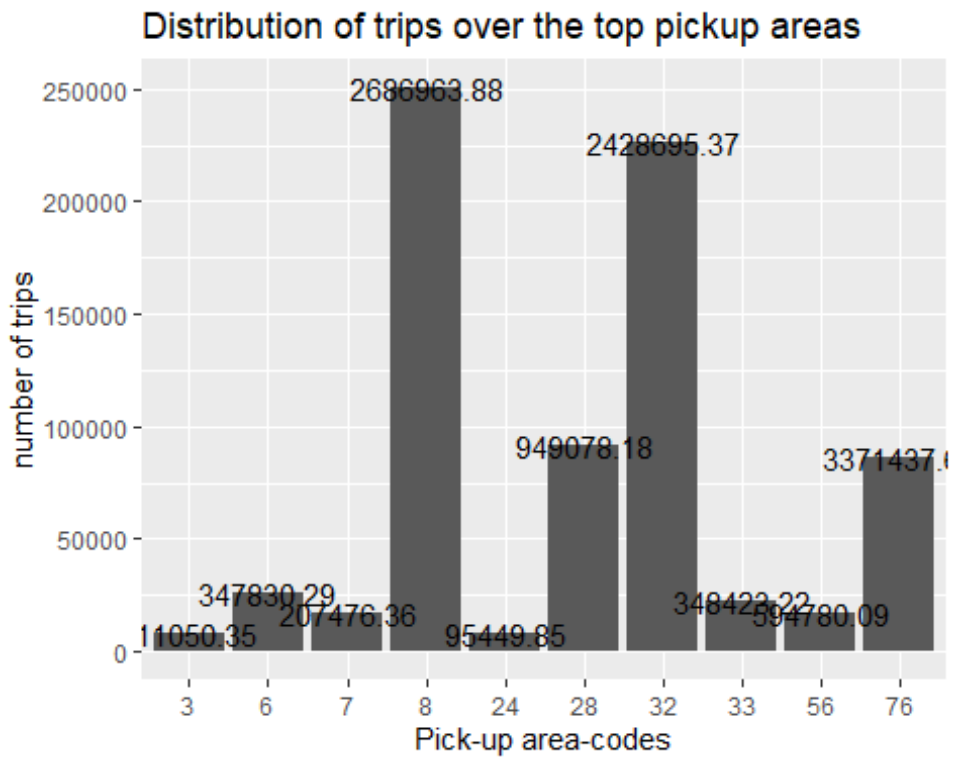
Followed by the central region is the far north side, West side and North side are the next best areas to focus on to make sure that the taxis are also available in those regions. This information is valuable to our company because it’s important to know what are the hot areas where the taxis are needed more so that it could generate most amount of the

revenue. If the company just randomly sends taxis to all the regions it wouldn't generate a high number of pickup rides whereas if it sends according to the areas shown in the graph it could generate more pick-ups and drop offs and thus generate good revenue.

After we got an idea about the regions, we further explored the top10 regions and the revenue generated from these regions.

Histogram showing top 10 pickup community areas and fare generated from it.

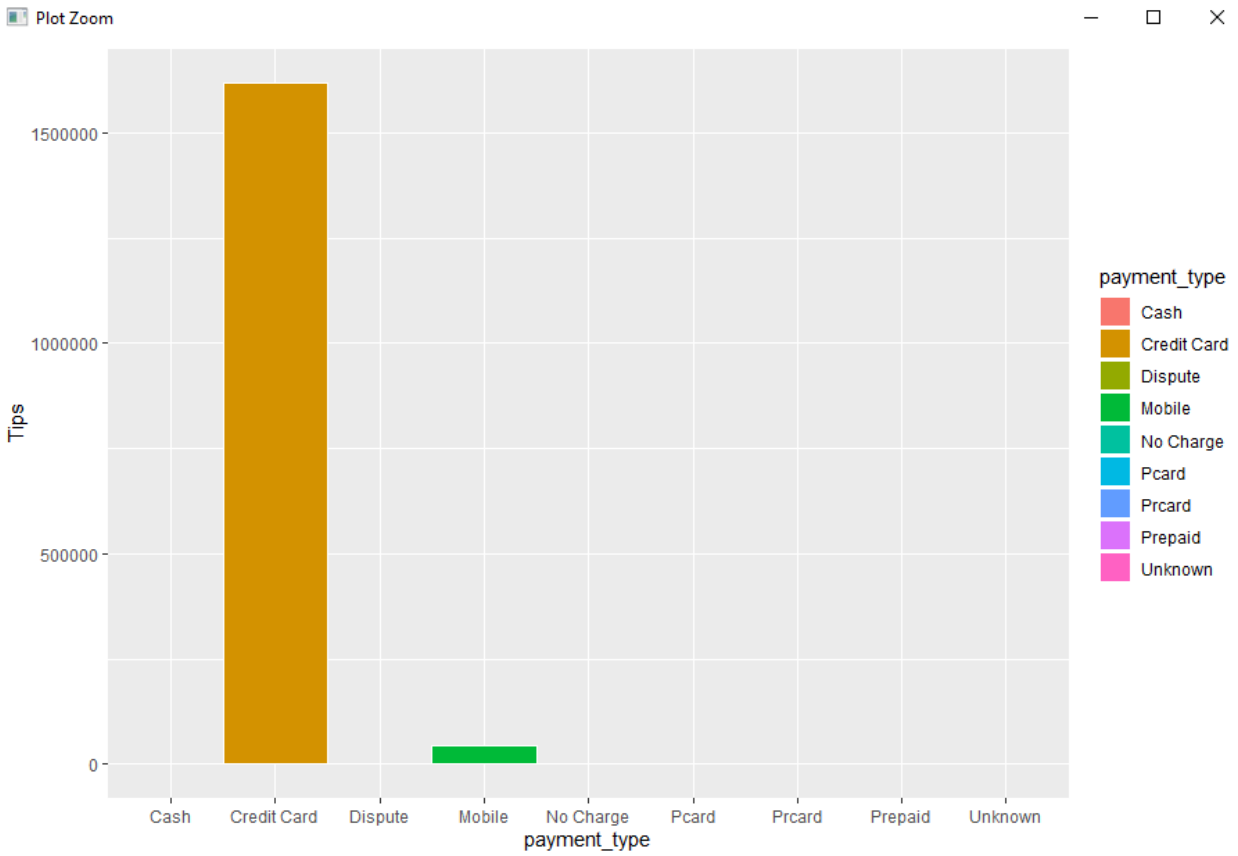
```
trips_2019 %>%
  filter(!is.na(pickup_community_area)) %>%
  filter(!is.na(fare)) %>%
  group_by(pickup_community_area) %>%
  summarise(
    n = n(),
    fare=sum(fare)
    ) %>%
  arrange(desc(n)) %>%
  head(10) %>%
  ggplot(mapping = aes(x=as.factor(pickup_community_area),y=n))+
  geom_histogram(stat = "identity")+
  geom_text(aes(y = n, label =fare ), color = "black", size=4)+
  labs(
    x="Pick-up area-codes",
    y="number of trips",
    title="Distribution of trips over the top pickup areas"
    )
```



The above histogram shows specific top 10 pickup community areas based on the count of number of pickups done from each area. From the Histogram we can clearly interpret the highest number of pickups are done from the community areas 8,32,28 and 76. So let's say if we want to start from low budget investment or with a smaller number of taxi's we could start with these areas first and then slowly expand.

Bar Graph showing the relation between the tips and payment type

```
trips_2019 %>%
  filter(!is.na(tolls)) %>%
  filter(!is.na(fare)) %>%
  filter(!is.na(extras)) %>%
  filter(!is.na(payment_type)) %>%
  filter(!is.na(tips)) %>%
  group_by(payment_type) %>%
  summarise(
    n = n(),
    Tips = sum(tips)
    ) %>%
  ggplot(mapping = aes(y=Tips,x=payment_type,fill=payment_type))+
  geom_col(width=1, color="white")
```

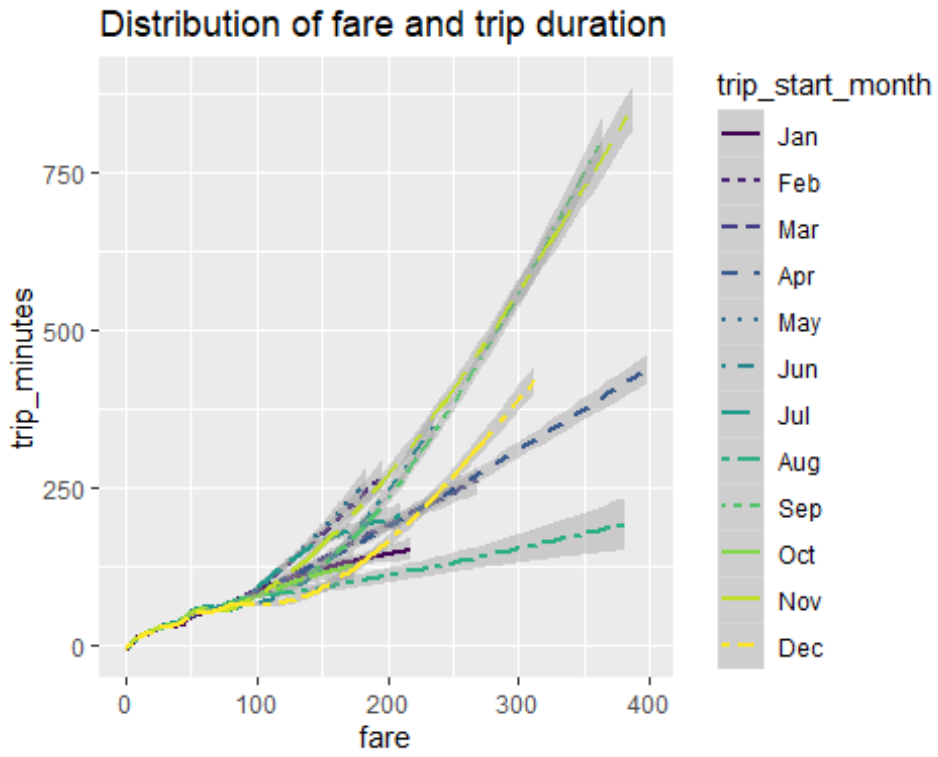


From the graph we can infer that tips are given most when the payment mode is through credit card. This information can be valuable for our drivers, because tipping is essential for them. The tip completely belongs to the drivers and having a credit card payment mode in our company will facilitate the tipping for our drivers as well.

To keep our drivers more informed about the relation between duration of the trip and the fare that they will make over the full year, we came up with a visual that will help us take our point across to the drivers well.

Plot between variation of fare and trip_minutes over all the months in a year.

```
trips_2019 %>%
  ggplot(mapping = aes(x=fare,y=trip_minutes,color=trip_start_month))+
  geom_smooth(aes(linetype=trip_start_month))+
  labs(
       title="Distribution of fare and trip duration"
       )

## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



From the graph, it is evident that the November month has significant number of long trips, so with this information we can tell our drivers that though our taxi service is going to be a local one, in the month of December, when people enjoy their vacation and shop, they will use the taxi service for long trips and long trips obviously will imply more fare. This information is useful for us as company to make our drivers more aware of all situations.

RELATIONSHIP BETWEEN VARIABLES USING CONTINGENCY TABLE

1. Assessing the relationship between top 5 companies and their fare ranges for the trips in 2019.

The top 5 companies are considered based on their total number of trips in the year 2019. This criterion is considered because if a company has a greater number of trips it means that more people tend to use them.

Data Preparation

Top 5 companies

```
company_top_5 <- trips_2019 %>%
  group_by(company) %>%
  summarise(
    trip_count = n()
  ) %>%
```

```
  arrange(desc(trip_count)) %>%
  head(5)
```

Defining a particular fare range for the trips by dividing the fare variable as low (up to $10), medium ($10-$25), high ($25-$50) and very high (above $50) based on values set by our company criteria.

Applying fare range, filtering only the required companies and selecting only the desired columns.

```
trips_chi2_1 <- trips_2019 %>%
  filter(!is.na(fare)) %>%
  mutate(fare_range = ifelse(fare <= 10, "low",
                      ifelse(fare > 10 & fare <= 25, "medium",
                      ifelse(fare > 25 & fare <= 50, "high",
                                                     "very high")))) %>%
  select(company, fare, fare_range) %>%
  filter(company == company_top_5$company)
```

Assessing the relationship in Spark

Connecting to spark

```
sc <- spark_connect(master = "local", version = "2.3")
```

Copying required data to spark

```
trips_chi2_spark <- copy_to(sc, trips_chi2_1, overwrite = TRUE)
```

The correlation between company and their fare range can be best explained using contingency table.

```
contingency_trips_comp_fare <- trips_chi2_spark %>%
  sdf_crosstab("fare_range","company") %>%
  collect()
```

Displaying the contingency table

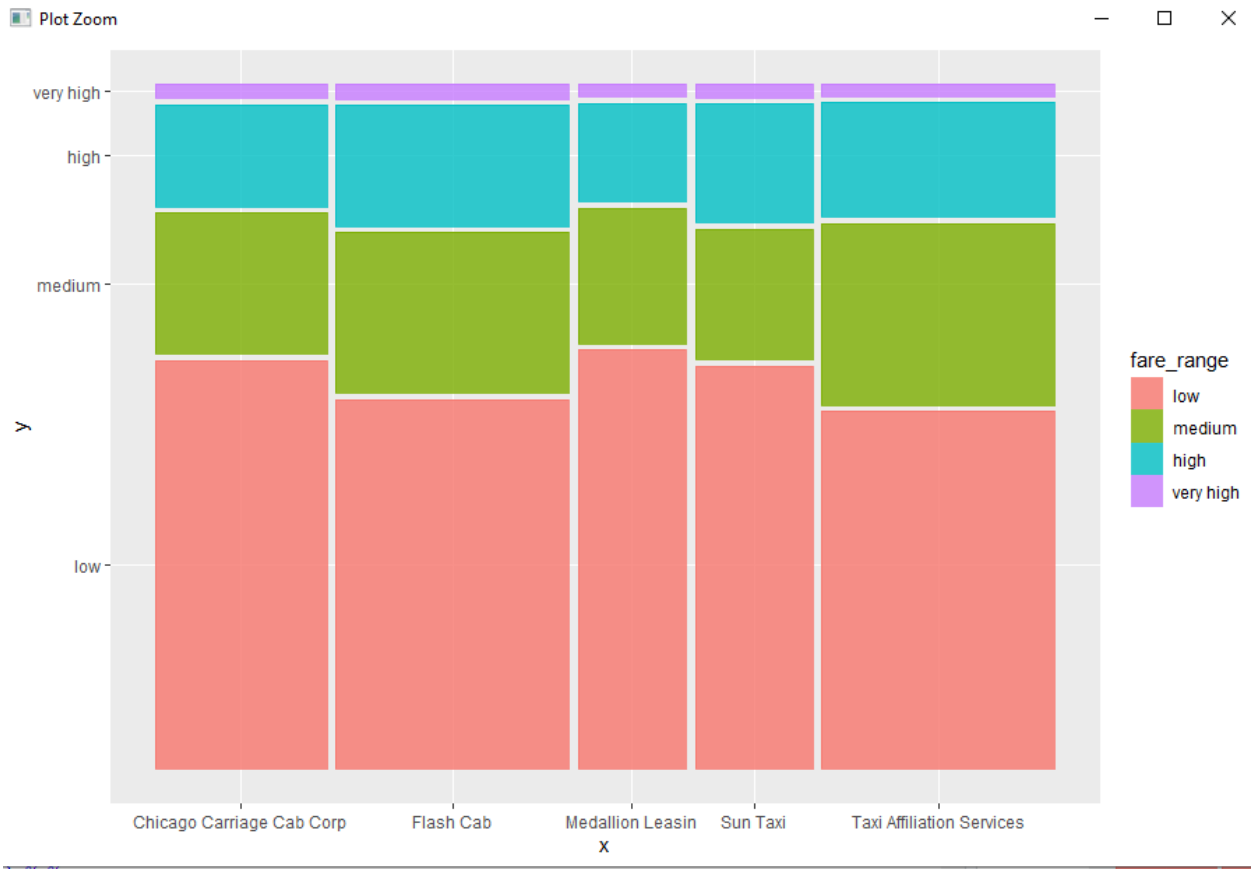
```
contingency_trips_comp_fare
```

```
## # A tibble: 4 x 6
##   fare_range_comp~ `Chicago Carria~ `Flash Cab` `Medallion Leas~ `Sun Taxi
`
##   <chr>                       <dbl>       <dbl>            <dbl>      <dbl
>
## 1 medium                       4574        7093             2786       291
0
## 2 high                         3284        5348             2029       265
1
## 3 very high                     460         641              250        28
6
## 4 low                         13275       16372             8646       897
```

```
5
## # ... with 1 more variable: `Taxi Affiliation Services` <dbl>
```

This contingency table can be visualized using mosaic plot as follows:

```
contingency_trips_comp_fare %>%
  rename(fare_range = fare_range_company) %>%
  gather("company", "count", 2:6) %>%
  mutate(
    fare_range = as_factor(fare_range) %>%
      fct_relevel("low", "medium", "high", "very high"),
    company = as_factor(company)
  ) %>%
  ggplot() +
    geom_mosaic(aes(x=product(fare_range, company), fill = fare_range, weight
= count))
```




```
trips_chi2_spark %>%
  group_by(fare_range) %>%
  count() %>%
  ungroup() %>%
  mutate(frac = n / sum(n))

## # Source: spark<?> [?? x 3]
##   fare_range     n   frac
```

```
##   <chr>      <dbl>  <dbl>
## 1 high       18381 0.169
## 2 medium     25399 0.233
## 3 low        63143 0.579
## 4 very high   2151 0.0197
```

Disconnecting the spark connection

```
spark_disconnect(sc)
```

Inference:

From the graph it is clear that most of the trips of top 5 companies belong to low fare range (61%) which is below $10. This will be a very useful interpretation for our company as this clearly explains us that most of the people are using cabs for shorter distances and paying less fares. Also, we can see from the graph that the top 5 company rides have very less "high" and "very high" fare ranges comparatively. This means that there are very a smaller number of trips whose fare ranges are high and very high for the top companies. With this interpretation we clearly know that our company should not start with higher base fare range. To compete with the top 5 companies, our company should maintain economic fare ranges by introducing a base fare range slightly less than the base fares of these companies in order to attract more customers thereby increasing its number of trips. Once our company becomes standardized in the market then we can plan on improving our base fare ranges.

2. Assessing the relationship between payment types and fare ranges for the trips in 2019.

We are planning to assess which payment types are more preferred based on our fixed values for fare ranges.

Data Preparation

Defining a particular fare range for the trips by dividing the fare variable as low (up to $10), medium ($10-$25), high ($25-$50) and very high (above $50) based on values set by our company criteria. Also, the most prominent payment types in the dataset are Cash and Credit Cards. So, only these payment types are considered for analysis.

Selecting the required data for analysis.

```
trips_chi2_2 <- trips_2019 %>%
  mutate(fare_range = ifelse(fare <= 10, "low",
                      ifelse(fare > 10 & fare <= 25, "medium",
                      ifelse(fare > 25 & fare <= 50, "high",
                                                     "very high")))) %>%
  filter(payment_type == "Cash" | payment_type == "Credit Card") %>%
  select(fare_range, payment_type)
```

Removing NA's from the data

```
trips_chi2_2 <- na.omit(trips_chi2_2)
```

Assessing the relationship in Spark

Connecting to Spark

```
sc <- spark_connect(master = "local", version = "2.3")
```

Copying the data to Spark

```
trips_chi2_2_spark <- copy_to(sc, trips_chi2_2, overwrite = T)
```

The correlation between payment type and fare range can be best explained using contingency table.

```
contingency_trips_ptype_fare <- trips_chi2_2_spark %>%
  sdf_crosstab("fare_range","payment_type") %>%
  collect()
```

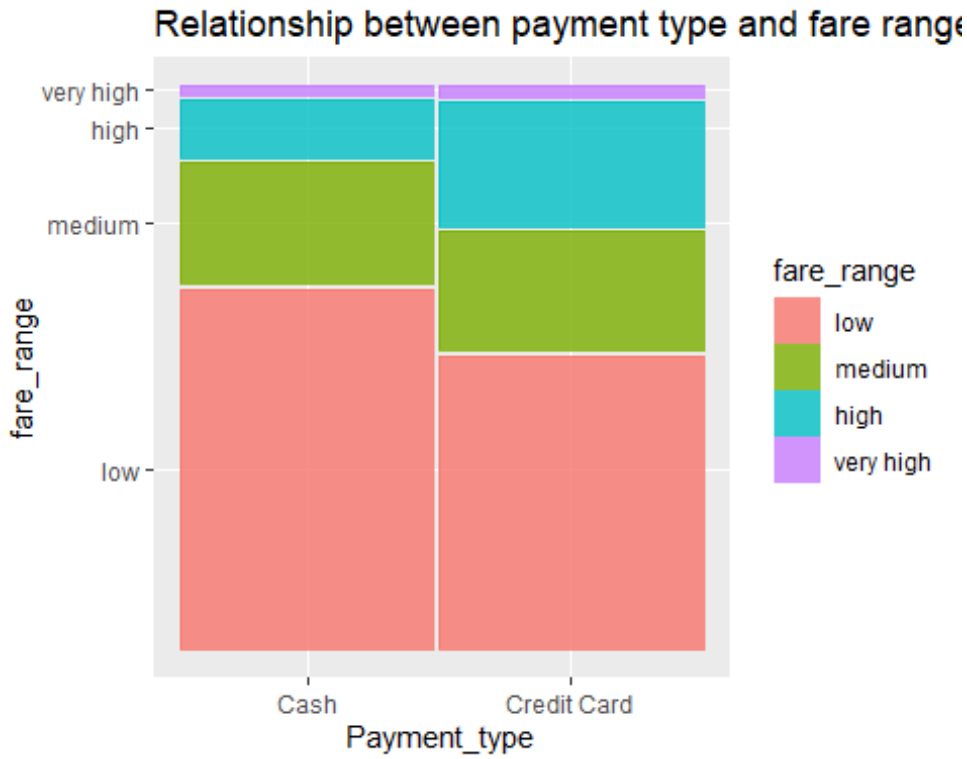
Displaying the contingency table

```
contingency_trips_ptype_fare
```

```
## # A tibble: 4 x 3
##   fare_range_payment_type   Cash `Credit Card`
##   <chr>                    <dbl>         <dbl>
## 1 medium                   89876         92707
## 2 high                     42487         96221
## 3 very high                 6951          9059
## 4 low                     267576        227579
```

This contingency table can be visualized using mosaic plot as follows:

```
contingency_trips_ptype_fare %>%
  rename(fare_range = fare_range_payment_type) %>%
  gather("payment_type", "count", 2:3) %>%
  mutate(
    fare_range = as_factor(fare_range) %>%
      fct_relevel("low", "medium", "high", "very high"),
    payment_type = as_factor(payment_type)
  ) %>%
  ggplot() +
    geom_mosaic(aes(x=product(fare_range, payment_type), fill = fare_range, w
eight = count)) +
  labs(
    x = "Payment_type",
    y = "fare_range",
    title = "Relationship between payment type and fare range"
  )
```



Most of the prominent payment types are cash and credit card. Also, the graph shows low and high fare ranges vary with payment types. We are checking the distribution of payment type with fare ranges.

Low fare ranges:

```
trips_chi2_2_spark %>%
  filter(fare_range == "low") %>%
  group_by(payment_type, fare_range) %>%
  count() %>%
  ungroup() %>%
  mutate(frac = n / sum(n)) %>%
  filter(payment_type == "Cash" | payment_type == "Credit Card")
```

```
## # Source: spark<?> [?? x 4]
##   payment_type fare_range      n  frac
##   <chr>        <chr>       <dbl> <dbl>
## 1 Credit Card  low        227579 0.460
## 2 Cash         low        267576 0.540
```

High fare ranges:

```
trips_chi2_2_spark %>%
  filter(fare_range == "high") %>%
  group_by(payment_type, fare_range) %>%
  count() %>%
  ungroup() %>%
```

```
  mutate(frac = n / sum(n)) %>%
  filter(payment_type == "Cash" | payment_type == "Credit Card")
```

```
## # Source: spark<?> [?? x 4]
##   payment_type fare_range     n  frac
##   <chr>        <chr>      <dbl> <dbl>
## 1 Cash         high       42487 0.306
## 2 Credit Card  high       96221 0.694
```

Disconnecting the spark connection

```
spark_disconnect(sc)
```

Inference:

From the graph it is clear that most of the prominent payment types customers are preferring to pay through are cash and credit cards. The graph clearly explains that payments for many of the low fare range trips are made through cash (54%). It is also clear that payments for many of the high range trips are made through credit cards (69%). With this analysis our company should make a note that customers are willing to pay through cash for trips less than $10 and through credit cards for trips between $25 to $50. It is clear that both online and offline payment modes should be integrated in our business for making it flexible for the customers in order to make payments.

LINEAR REGRESSION

Why do we go for a regression model

The relationship between trip_total and trip miles is very valuable data for our company to have. This is because, based on how we are planning to operate our company, knowing this relationship allows us to maximize profits and assist in setting our fare. As we are planning to primarily provide services to businessmen and women, it gives us the unique ability to adjust our rates in a way that is not only advantageous to the company, but also towards the customer. Knowing the relationship between trip_total and trip miles based on how other taxi companies operate is vital to us providing the highest profitability for the company.

The basic idea behind building various regression model is that we can keep these models as our base and if we get some other data in the future, we can use our regression models to predict.

Data Preparation

Removing all the NA's from the variables we use for performing regression analysis.

```
New2 <- trips_2019_1 %>%
  filter(!(is.na(fare))) %>%
  filter(!is.na(trip_total)) %>%
  filter(!(is.na(trip_minutes))) %>%
  filter(!(is.na(trip_miles)))
```

Applying filter for better results.

```
New3 <- New2 %>%
  filter(tips < 20)%>%
  filter(trip_miles != 0)%>%
  filter(trip_miles < 100) %>%
  filter(trip_seconds != 0)%>%
  filter(trip_total < 500) %>%
  filter(trip_total != 0) %>%
  filter(tolls < 900) %>%
  filter(fare < 400)
```

We applied these filters because these filters help our dataset, to eliminate the influential cases and the anomalies. These filters make the visualization of the graph better. The filter helped us remove the unusual entries in the data.

Data Splitting

```
set.seed(230)

sample <- sample(c(TRUE, FALSE),
                 nrow(New3),
                 replace = T,
                 prob = c(0.6,0.4))

train <- New3[sample, ]
test <- New3[!sample, ]
```

We have split the data into train and test data in the ratio 60 and 40, we have built all the regression models on the train data and finally we used the test data to predict if the model is doing a fair job or not.

Building the model

```
linear_reg<-lm(trip_total~trip_miles,
               data=train)
```

We have built a linear regression model between trip_total and tripmiles. Because, our dataset is about taxi trips and we are interested in finding the relation between trip_total and trip_miles because we want to know if more miles implies more trip_total or not. We want to see the relationship between these two variables whether one predicts the other or not. After the model is built, the output of this model will be an equation which will tells us the variation in our response variable based on our predictor variable.

Analyzing the model numerically

```
glance(linear_reg)

## # A tibble: 1 x 11
##   r.squared adj.r.squared sigma statistic p.value    df  logLik    AIC
```

```
BIC
##       <dbl>         <dbl> <dbl>     <dbl>   <dbl> <int>   <dbl>  <dbl>  <d
bl>
## 1     0.820         0.820  7.60  2311578.       0     2 -1.75e6 3.50e6 3.5
0e6
## # ... with 2 more variables: deviance <dbl>, df.residual <int>
```

```
summary(linear_reg)
```

```
##
## Call:
## lm(formula = trip_total ~ trip_miles, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -257.19   -2.93   -1.30    0.67  453.15
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 6.546305   0.013174   496.9   <2e-16 ***
## trip_miles  2.758148   0.001814  1520.4   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.599 on 508300 degrees of freedom
## Multiple R-squared:  0.8197, Adjusted R-squared:  0.8197
## F-statistic: 2.312e+06 on 1 and 508300 DF,  p-value: < 2.2e-16
```

```
tidy(linear_reg)
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic p.value
##   <chr>          <dbl>     <dbl>     <dbl>   <dbl>
## 1 (Intercept)     6.55   0.0132       497.       0
## 2 trip_miles      2.76   0.00181     1520.       0
```

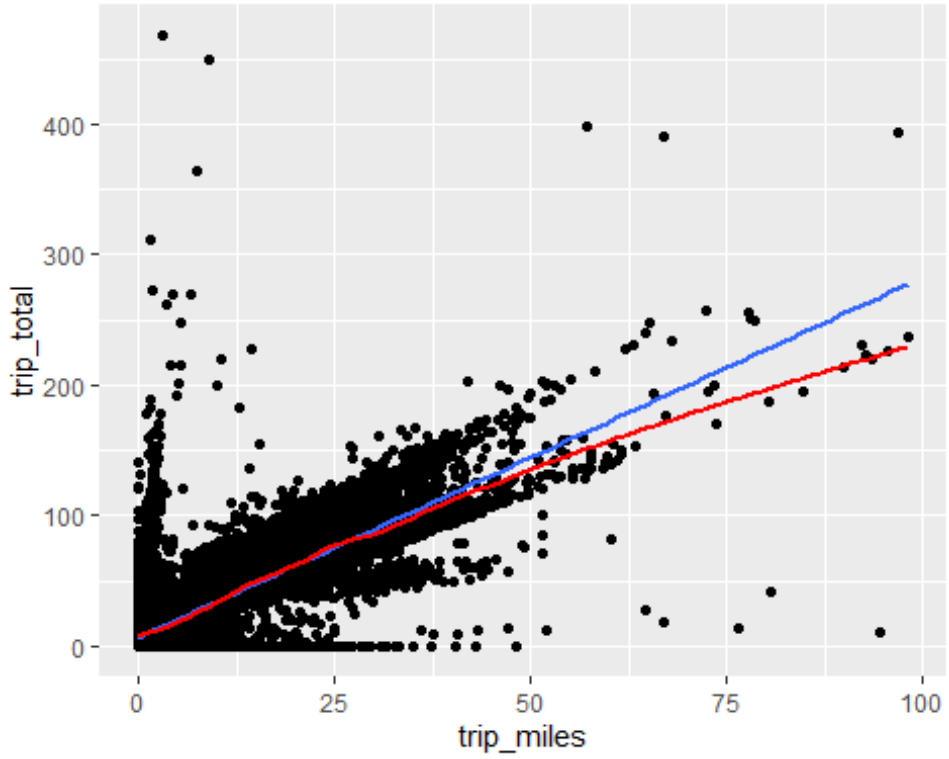
From the above output it is clear that, our model is significant as the value of p<alpha. The value of Rsq is 0.8196 or 82% of variance in trip_total is explained by the trip_miles. The value of the coefficient imply that if the trip miles increase by a value of 1 then the trip_total increases by 2.76. The F statistic value is very high implying that the model is fair. The value of RSE is 7.618, which implies that the residual square error for our model is low. The equation for this model is trip_total=6.53+(2.76)trip_miles, where 6.53 is the intercept value and 2.76 is the trip_miles coefficient.

Importance of this information for our company is that we are going to start our service keeping this relation in mind, the above information about the linear regression model is true for the entire dataset, that implies that it is true for all the taxi companies that are our competitors based on our market strategy we will set the trip total according to the trip miles following the linear relationship.

For example, based on this regression line equation if a taxi travels 10miles then the trip_total that will be charged is trip_total=6.53+2.76*10=$34.13. This supports one of our previous assumptions made during assessing the performance of the top five companies.

Analyzing the model visually

```
ggplot(train, aes(trip_miles,trip_total)) +
  geom_point() +
  geom_smooth(method = "lm") +
  geom_smooth(se = FALSE, color = "red")

## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



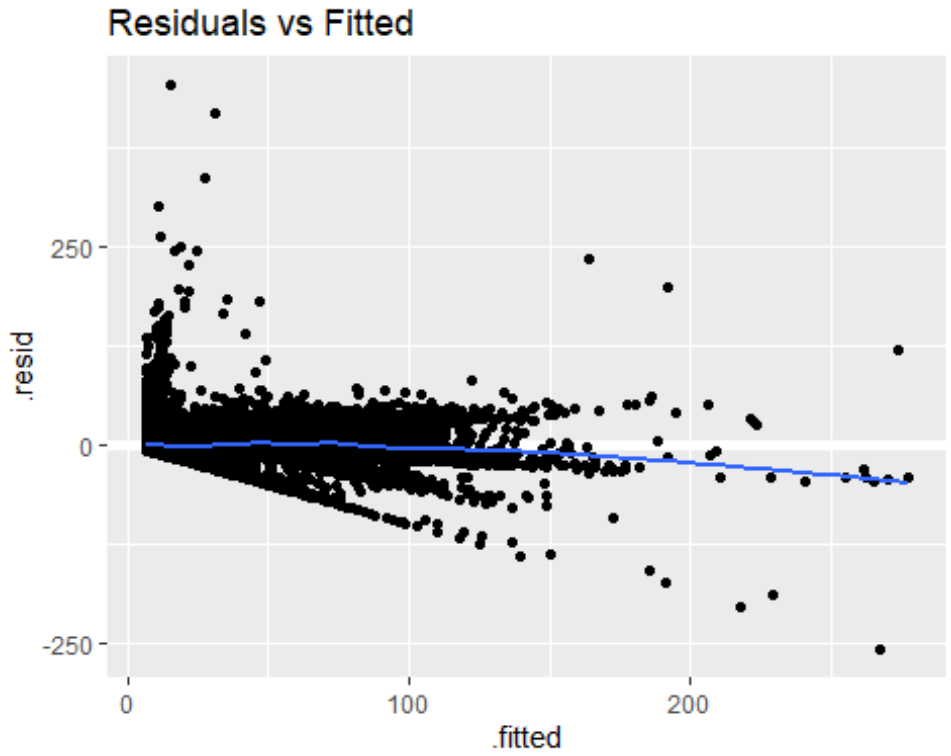
As displayed in the above graph, the blue line is the reference line and the red line is the line of our model, the red line almost follows the path of the blue line indicating that the model is good and there are no major anomalies. For smaller values of trip miles, the model perfectly follows the reference blue line. However, for higher values there is slight deviation implying that as trip miles increases the trip_total might not increase linearly and there may be some variations due to some factors.

Check for linearity and Homoscedasticity

```
linear_reg_result<-augment(linear_reg,train)

linear_reg_result<-linear_reg_result %>%
  mutate(model="linear_reg")
```

```
linear_reg_result %>%
  sample_n(500000) %>%
  ggplot(aes(.fitted, .resid)) +
  geom_ref_line(h = 0) +
  geom_point() +
  geom_smooth(se = FALSE) +
  ggtitle("Residuals vs Fitted")
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



For memory allocation warnings we have plotted the residual and fitted plot for 500K samples. Looking at the graph, we can deduct that the model has linearity and there is a funnel effect at the values that are small, that is there is funnel effect for the smaller values of trip_miles. We have tried eliminating the funnel effect using some transformations. So, we have applied the square root transformation and the logarithmic transformation.

Transformation using sqrt(x)

```
linear_reg_transformation1<-lm(sqrt(trip_total)~ trip_miles,
                               data=train)
```

Assesing the model numerically

```
glance(linear_reg_transformation1)
```

```
## # A tibble: 1 x 11
##   r.squared adj.r.squared sigma statistic p.value    df  logLik    AIC
BIC
##       <dbl>         <dbl> <dbl>     <dbl>   <dbl> <int>   <dbl>  <dbl>  <d
bl>
## 1     0.806         0.806 0.757  2118432.       0     2 -5.79e5 1.16e6 1.1
6e6
## # ... with 2 more variables: deviance <dbl>, df.residual <int>
```

```
summary(linear_reg_transformation1)
```

```
##
## Call:
## lm(formula = sqrt(trip_total) ~ trip_miles, data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -24.4769  -0.3761  -0.0962   0.2141  18.0269
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 2.7972256  0.0013116    2133   <2e-16 ***
## trip_miles  0.2628778  0.0001806    1455   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7566 on 508300 degrees of freedom
## Multiple R-squared:  0.8065, Adjusted R-squared:  0.8065
## F-statistic: 2.118e+06 on 1 and 508300 DF,  p-value: < 2.2e-16
```

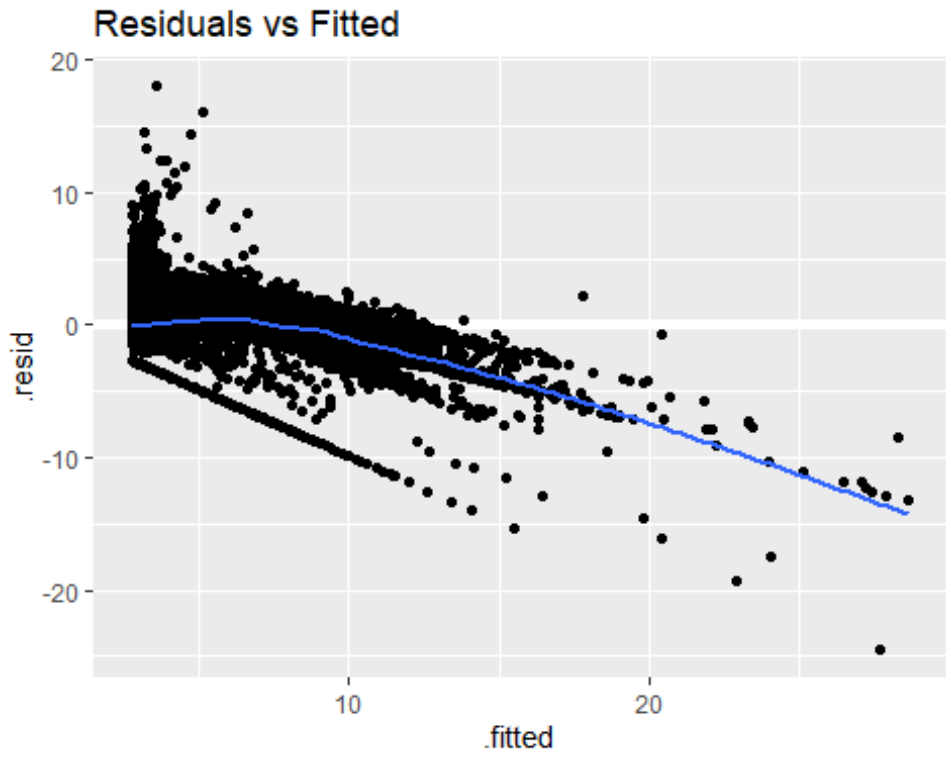
The values have changed after the transformation has been applied on the y variable, the Rsquare value has decreased a little bit and has come to 80.8%, the value of sigma has decreased remarkably it is 0.75, which is pretty low. P-value is still less than alpha implying that the the model is significant.

Visually analyzing the model

```
linear_reg_transformation1_result<-augment(linear_reg_transformation1,train)
```

```
ggplot(linear_reg_transformation1_result, aes(.fitted, .resid)) +
  geom_ref_line(h = 0) +
  geom_point() +
  geom_smooth(se = FALSE) +
  ggtitle("Residuals vs Fitted")
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



After the transformation of sqrt(x) the funnel effect has not reduced as much as expected, for lower values of the trip_miles there is still a funnel. This implies that there is a heteroskedasticity and sqrt transformation is not working well to get rid of it. Instead, the linearity of the model is effected which is not acceptable.

We then looked at the second transformation to eliminate the funnel effect, the log transformation.

Transformation using log(x):

```
linear_reg_transformation2<-lm(log(trip_total)~ trip_miles,data=train)
```

Assesing the model numerically

```
glance(linear_reg_transformation2)
```

```
## # A tibble: 1 x 11
##   r.squared adj.r.squared sigma statistic p.value    df  logLik    AIC
BIC
##       <dbl>         <dbl> <dbl>     <dbl>   <dbl> <int>   <dbl>  <dbl>  <d
bl>
## 1     0.693         0.693 0.430  1146000.       0     2 -2.92e5 5.84e5 5.8
4e5
## # ... with 2 more variables: deviance <dbl>, df.residual <int>
```

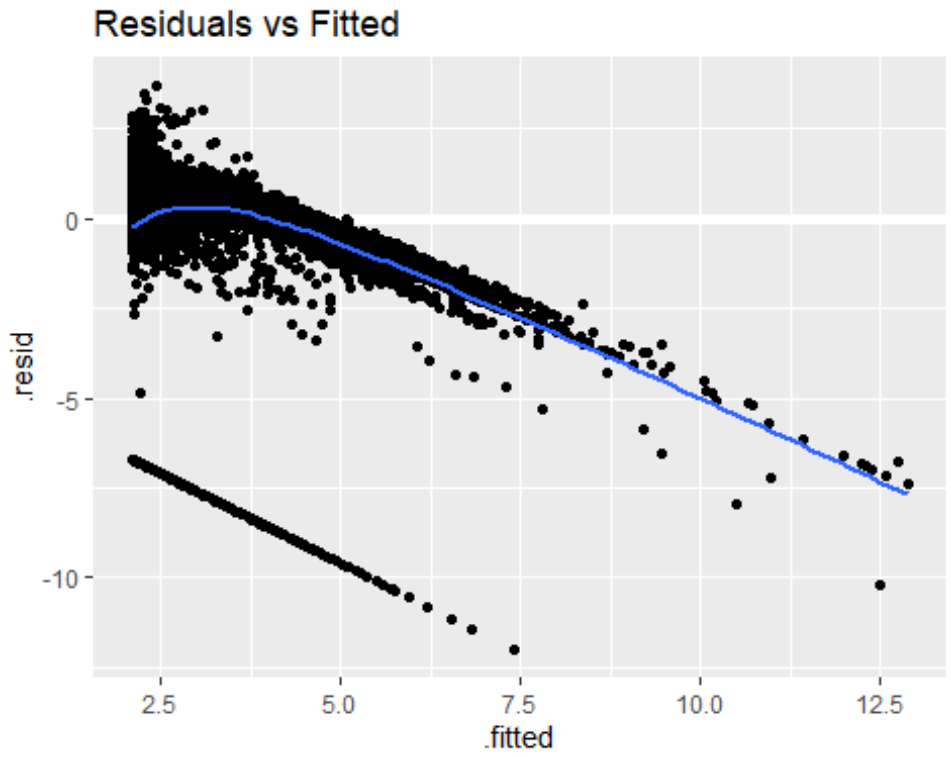
```
summary(linear_reg_transformation2)
```

```
##
## Call:
## lm(formula = log(trip_total) ~ trip_miles, data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -12.0068  -0.2094   0.0028   0.1851   3.7051
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 2.1035467  0.0007448    2824   <2e-16 ***
## trip_miles  0.1097827  0.0001026    1071   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4296 on 508300 degrees of freedom
## Multiple R-squared:  0.6927, Adjusted R-squared:  0.6927
## F-statistic: 1.146e+06 on 1 and 508300 DF,  p-value: < 2.2e-16
```

The log(x) transformation changes our R square drastically, it has changed from a 80% to a 69% which is not good. Let's assess the model's residual and fitted graph and check if the funnel effect is eliminated or not.

Visually analyzing the model

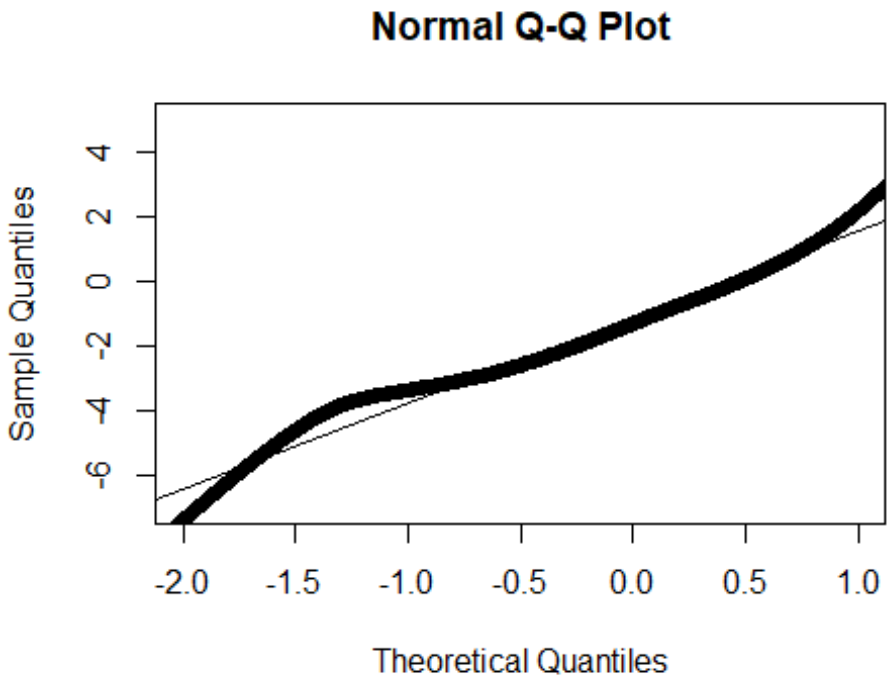
```
linear_reg_transformation2_result<-augment(linear_reg_transformation2,train)

ggplot(linear_reg_transformation2_result, aes(.fitted, .resid)) +
  geom_ref_line(h = 0) +
  geom_point() +
  geom_smooth(se = FALSE) +
  ggtitle("Residuals vs Fitted")

## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



From the graph it is evident that the funnel effect is eliminated, however the linearity of the model is affected that is indicated by the blue reference line, so this transformation is better than sqrt(x) transformation but it is not the best possible transformation for this model.

Q-Q plot to check for normality issues

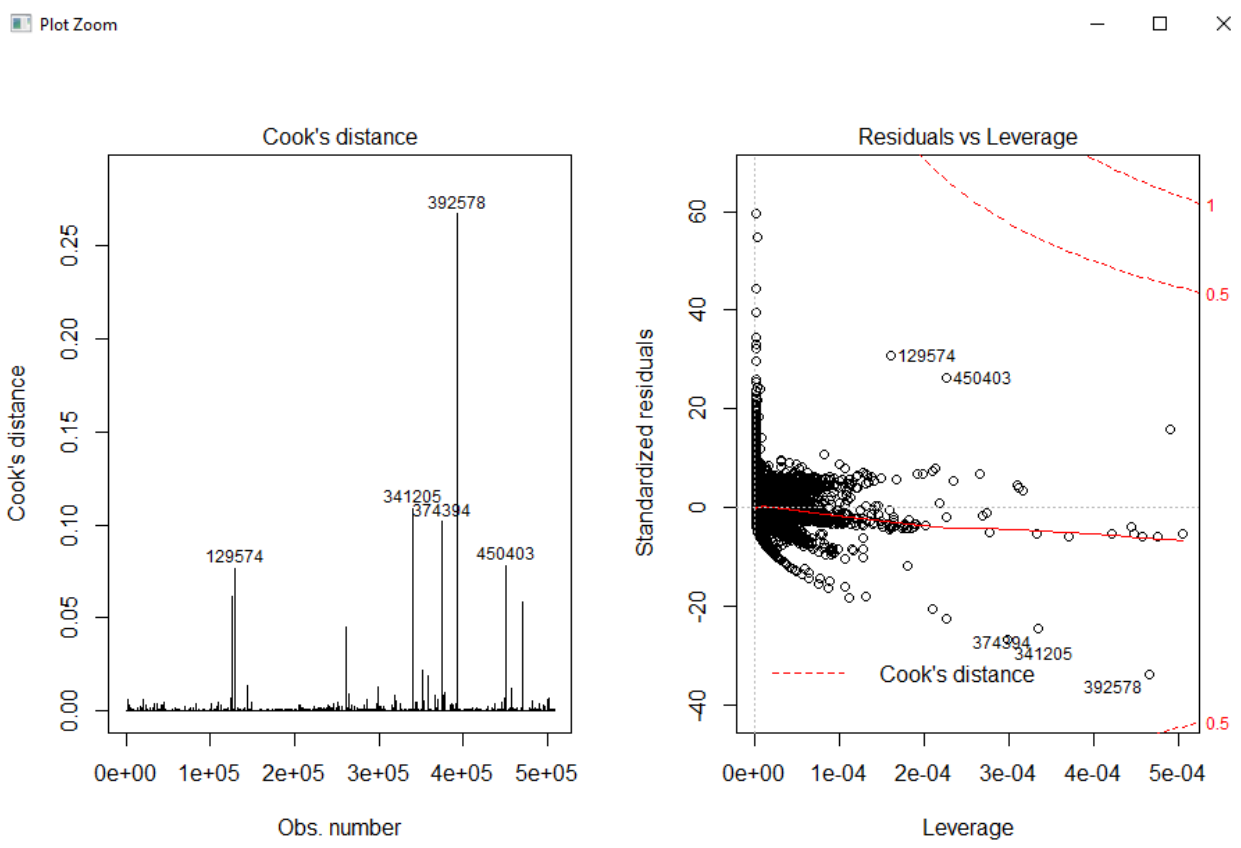
```
par(mfrow = c(1,1))
qqplot <- qqnorm(linear_reg_result$.resid,xlim=c(-2,1),ylim = c(-7,5))
qqplot <- qqline(linear_reg_result$.resid,xlim=c(-2,1),ylim = c(-7,5))
```



From the qq plot, it is clear that there are some normality issues because the curve is on the line only in some places and the top and bottom of the curve have most issues because those points are not the normality line.

Cook's D plot for influential cases

```
par(mfrow=c(1, 2))

plot(linear_reg, which = 4, id.n = 5)
plot(linear_reg, which = 5, id.n = 5)
```




```
linear_reg_result%>%
  top_n(5, wt = .cooksd)
```

```
## # A tibble: 5 x 43
##   trip_id taxi_id trip_start_timesta~ trip_end_timestamp  trip_seconds
##   <chr>   <chr>   <dttm>              <dttm>                      <dbl>
## 1 5d6083~ d03201~ 2019-04-16 01:45:00 2019-04-16 16:30:00        52860
## 2 a5c11c~ 74ec15~ 2019-09-11 13:30:00 2019-09-11 13:45:00         1034
## 3 78148e~ 6503f3~ 2019-09-05 07:00:00 2019-09-05 07:30:00         1440
## 4 4c22b8~ a4a925~ 2019-10-06 10:15:00 2019-10-06 10:15:00          300
## 5 44b31a~ 06d060~ 2019-11-24 21:45:00 2019-11-25 10:45:00        46624
## # ... with 38 more variables: trip_miles <dbl>, pickup_census_tract <dbl>,
## #   dropff_census_tract <dbl>, pickup_community_area <dbl>,
## #   dropoff_community_area <dbl>, fare <dbl>, tips <dbl>, tolls <dbl>,
## #   extras <dbl>, trip_total <dbl>, payment_type <chr>, company <chr>,
## #   pickup_centroid_latitude <dbl>, pickup_centroid_longitude <dbl>,
## #   pickup_centroid_location <chr>, dropoff_centroid_latitude <dbl>,
## #   dropoff_centroid_longitude <dbl>, dropoff_centroid_location <chr>,
## #   trip_start_year <dbl>, trip_start_month <ord>, trip_start_day <int>,
## #   trip_start_weekday <ord>, trip_start_hour <int>, trip_start_minute <in
t>,
## #   trip_end_year <dbl>, trip_end_month <ord>, trip_end_day <int>,
## #   trip_end_hour <int>, trip_end_minute <int>, trip_minutes <dbl>,
```

```
## #   .fitted <dbl>, .se.fit <dbl>, .resid <dbl>, .hat <dbl>, .sigma <dbl>,
## #   .cooksd <dbl>, .std.resid <dbl>, model <chr>
```

The top 5 influential points are the observations at 129574, 341205, 392578, 374394, 4540403 if these points are eliminated the model will become better.

Making predictions

```
test <-test %>%
  add_predictions(linear_reg)
```

We are building this model, so that we can use it for prediction purposes, we have built the model using the train dataset and now we are adding predictions using the test dataset.

MULTIPLE REGRESSION

Building the model

```
multiple_reg<-lm(trip_total~trip_miles+tips+tolls,
                 data=train)
```

After the linear regression, we shifted our focus to look multiple regression model, in multiple regression there are more than one predictors, in our model our response variable is trip_total, because we are interested in trip_total, as a booming taxi company we want to understand how the trip_total value can be predicted using these three prime variables. The three variables we considered are trip_miles, tolls and tips. In any taxi ride these three variables are crucial, the trip_total is affected by how long you have travelled in the taxi, that is the trip miles, the amount tipped by the customer and finally the tolls.

Assessing the model accuracy

```
glance(multiple_reg)
```

```
## # A tibble: 1 x 11
##   r.squared adj.r.squared sigma statistic p.value    df  logLik    AIC
BIC
##       <dbl>         <dbl> <dbl>     <dbl>   <dbl> <int>   <dbl>  <dbl>  <d
bl>
## 1     0.874         0.874  6.36  1173037.       0     4 -1.66e6 3.32e6 3.3
2e6
## # ... with 2 more variables: deviance <dbl>, df.residual <int>
```

```
summary(multiple_reg)
```

```
##
## Call:
## lm(formula = trip_total ~ trip_miles + tips + tolls, data = train)
##
```

```
## Residuals:
##     Min      1Q  Median      3Q     Max
## -217.99   -2.00   -1.17    0.05  455.85
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 5.314992   0.011337  468.80   <2e-16 ***
## trip_miles  2.285349   0.001825 1251.91   <2e-16 ***
## tips        1.678509   0.003605  465.59   <2e-16 ***
## tolls       1.589919   0.057219   27.79   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.359 on 508298 degrees of freedom
## Multiple R-squared:  0.8738, Adjusted R-squared:  0.8738
## F-statistic: 1.173e+06 on 3 and 508298 DF,  p-value: < 2.2e-16
```

```
tidy(multiple_reg)
```

```
## # A tibble: 4 x 5
##   term        estimate std.error statistic   p.value
##   <chr>          <dbl>     <dbl>     <dbl>     <dbl>
## 1 (Intercept)     5.31   0.0113      469.  0.
## 2 trip_miles      2.29   0.00183    1252.  0.
## 3 tips            1.68   0.00361     466.  0.
## 4 tolls           1.59   0.0572       27.8 8.48e-170
```

The adjusted Rsquare value for the model is 0.8733 that is 87.3%, that means that our multiple regression model explains 87.3% of variance in the trip_total due to trip_miles, tolls and tips. The F statistic for the model is huge implying that the model is good. The value of sigma is 6.384, that is a low value. Looking, at the parameter estimate coefficients and the pvalues, we can see that the predictor tolls is insignificant, because the pvalue is greater than alpha. The other two estimates for trip_miles and tips are 2.28 and 1.67, that is for every one value of increase in trip_miles there is 2.28 times increase in the trip_total. The equation that will help us predict the variation of the trip_total variable when there is a combined variation in the trip_miles+tolls+tips is

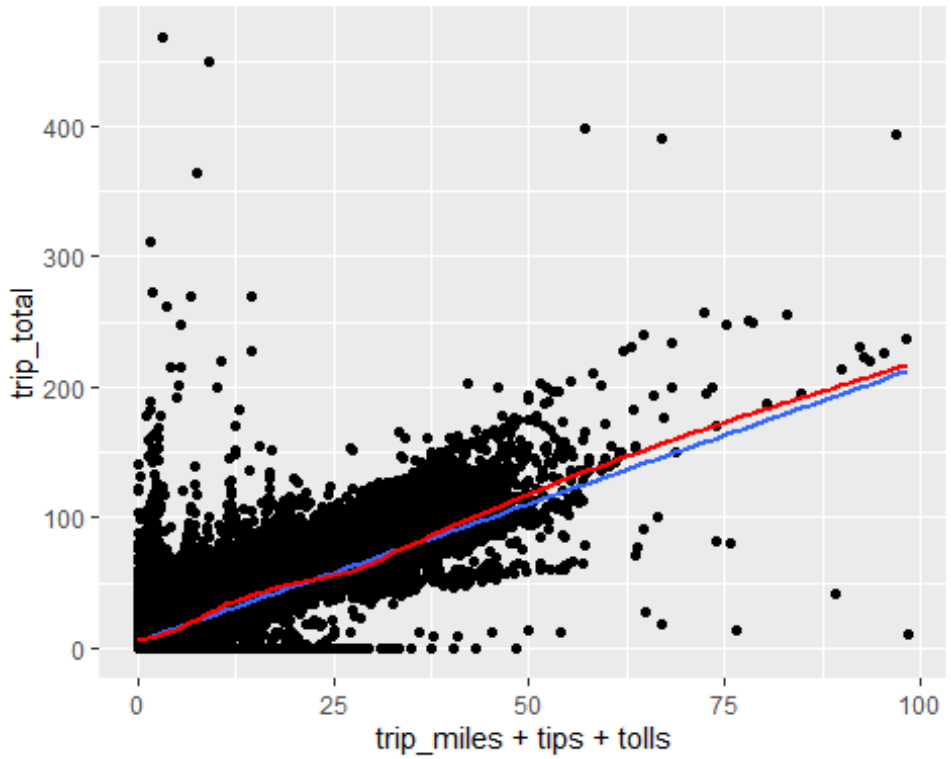
Equation: trip_total=5.31+(2.28)trip_miles+(1.67)tips+(1.70)tolls.

For example according to our model,if a trip covers 8 miles distance and the customer decides to give $2 tips and the tolls are zero,then the trip_total is 5.31+2.28*8+2*1.67+0*1.70=$26.89

Assessing the model visually

```
ggplot(train,aes(trip_miles+tips+tolls,trip_total)) +
  geom_point() +
  geom_smooth(method = "lm") +
  geom_smooth(se = FALSE, color = "red")
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



From the graph it is evident that there is a linear relationship between the combined predictors and the response variable. The red line that indicates our model is almost following the blue reference line, but however has some flaws at few points, especially at higher values of combined predictors. Now, let's look at the three assumptions.

Variation in the multiple regression model adding other possible predictors

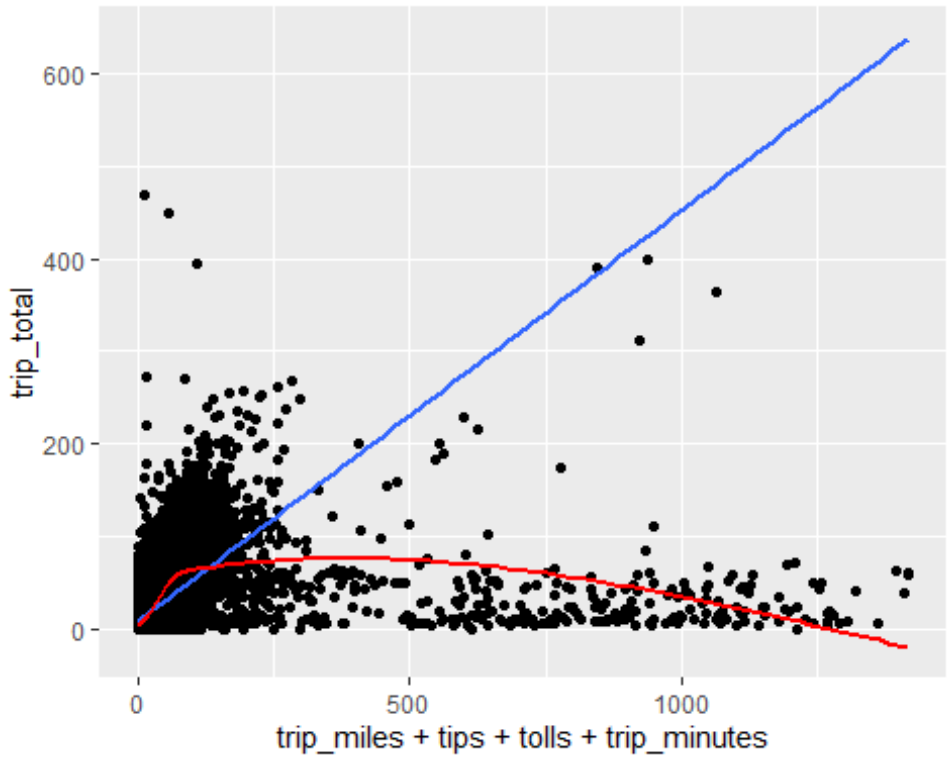
Building the model

```
multiple_reg1<-lm(trip_total~trip_miles+tips+tolls+trip_minutes,data=train)
```

We have just added the other possible predictor variable that is the trip_minutes to our model, let us have a look if it affects the linearity of the model.

Assessing the model visually

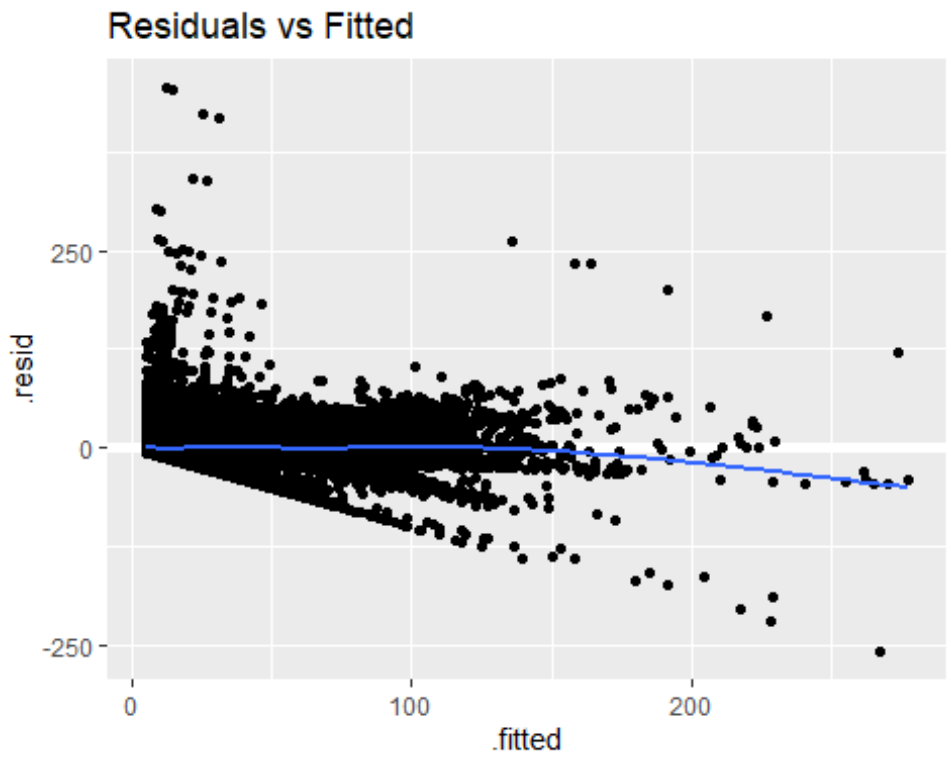
```
ggplot(train,aes(trip_miles+tips+tolls+trip_minutes,trip_total)) +
  geom_point() +
  geom_smooth(method = "lm") +
  geom_smooth(se = FALSE, color = "red")

## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



From the graph it is evident that adding this predictor affects our model negatively, so we have not further continued the analysis with this variable. Instead we have dropped this predictor and continued with the regression model without the trip_minutes.

Linearity and Homoskedasticity check

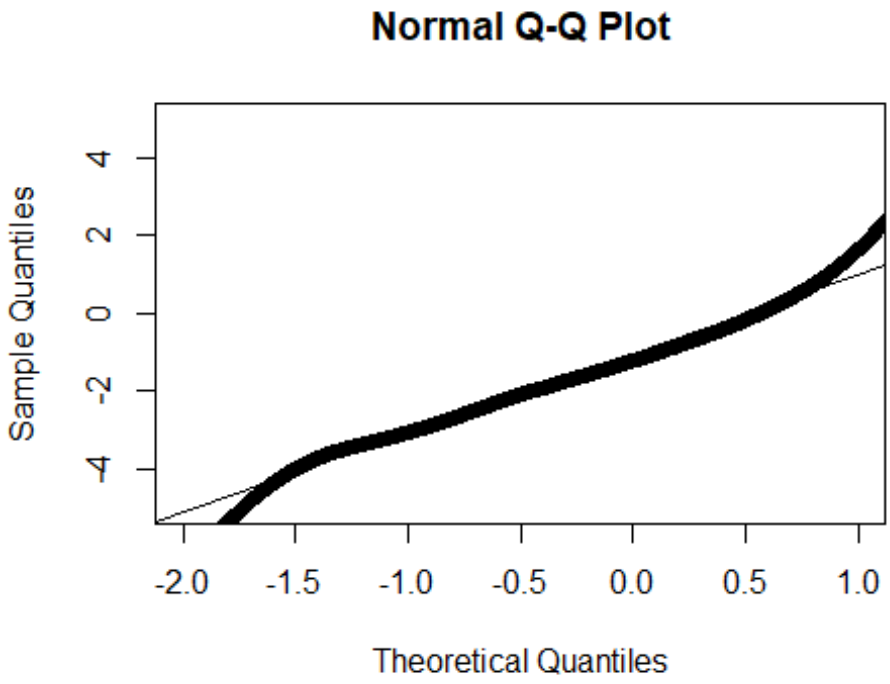
```
multiple_reg_result<-augment(multiple_reg,train)

multiple_reg_result<-multiple_reg_result%>%
  mutate(model="multiple_reg") %>%
  rbind(linear_reg_result)

ggplot(multiple_reg_result, aes(.fitted, .resid)) +
  geom_ref_line(h = 0) +
  geom_point() +
  geom_smooth(se = FALSE) +
  ggtitle("Residuals vs Fitted")

## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



From the graph, it is evident that there is some funnel effect for the negative values in the graph, but there is no funnel effect for the positive values, this implies that the assumption is violated but not to a larger extent.

Normality Check using Q-Q plot

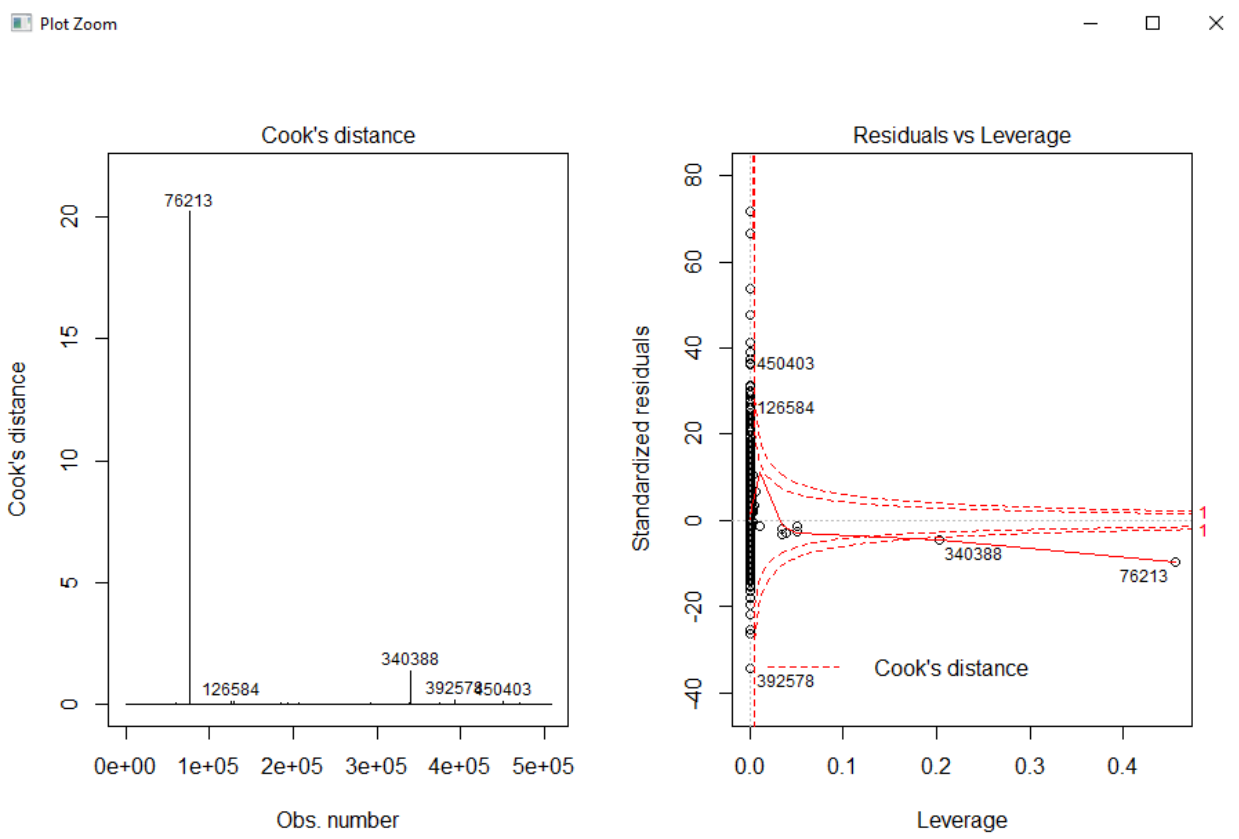
```
par (mfrow=c(1,1))
qqplot<-qqnorm(multiple_reg_result$.resid,xlim=c(-2,1),ylim = c(-5,5))
qqplot<-qqline(multiple_reg_result$.resid,xlim=c(-2,1),ylim = c(-5,5))
```



From the graph it is evident that there is definitely normality issue in the model, the higher values are not on the normal reference line, this implies that there are normality issues for higher values and no significant normality of lower values.

Cook's D plot for influential cases

```
par(mfrow=c(1, 2))

plot(multiple_reg, which = 4, id.n = 5)
plot(multiple_reg, which = 5, id.n = 5)
```




```
multiple_reg_result%>%
  top_n(5, wt = .cooksd)
```

```
## # A tibble: 5 x 43
##   trip_id taxi_id trip_start_timesta~ trip_end_timestamp  trip_seconds
##   <chr>   <chr>   <dttm>              <dttm>                      <dbl>
## 1 35e8e1~ 51e371~ 2019-02-14 12:00:00 2019-02-14 12:00:00          420
## 2 68786d~ 3d9dd8~ 2019-04-28 18:30:00 2019-04-28 18:45:00          596
## 3 afa418~ 51e371~ 2019-09-11 22:45:00 2019-09-11 23:15:00         1800
## 4 4c22b8~ a4a925~ 2019-10-06 10:15:00 2019-10-06 10:15:00          300
## 5 4c22b8~ a4a925~ 2019-10-06 10:15:00 2019-10-06 10:15:00          300
## # ... with 38 more variables: trip_miles <dbl>, pickup_census_tract <dbl>,
## #   dropff_census_tract <dbl>, pickup_community_area <dbl>,
## #   dropoff_community_area <dbl>, fare <dbl>, tips <dbl>, tolls <dbl>,
## #   extras <dbl>, trip_total <dbl>, payment_type <chr>, company <chr>,
## #   pickup_centroid_latitude <dbl>, pickup_centroid_longitude <dbl>,
## #   pickup_centroid_location <chr>, dropoff_centroid_latitude <dbl>,
## #   dropoff_centroid_longitude <dbl>, dropoff_centroid_location <chr>,
## #   trip_start_year <dbl>, trip_start_month <ord>, trip_start_day <int>,
## #   trip_start_weekday <ord>, trip_start_hour <int>, trip_start_minute <in
t>,
## #   trip_end_year <dbl>, trip_end_month <ord>, trip_end_day <int>,
## #   trip_end_hour <int>, trip_end_minute <int>, trip_minutes <dbl>,
```

```
## #   .fitted <dbl>, .se.fit <dbl>, .resid <dbl>, .hat <dbl>, .sigma <dbl>,
## #   .cooksd <dbl>, .std.resid <dbl>, model <chr>
```

Looking at the output the influential points that will affect the model are the observations at 76213, 126584, 340388, 450403, 392578.These are the top 5 influential points in our dataset, our model will become better if these observations are analyzed and further eliminated.

After analyzing the multiple regression model, we have reached a conclusion that all three variables that is trip_miles and tips affect the value of trip_total positively, that is if the value of trip miles, tolls and tips increases the trip_total increases.

Further we moved our focus to interaction and introduced an interaction term and analyzed how this would affect the relationship between trip_total, trip_miles and tips. The interaction term is trip_miles*tips.

Correlation between the predictors

```
cor(train$trip_miles,train$tips)

## [1] 0.5553304

cor(train$trip_miles,train$tolls)

## [1] 0.02012021
```

The correlation between trip_miles and tips is 0.55, which is a value that indicates that there is significantly low amount of correlation between these two terms. The correlation between trip_miles and tolls is 0.021, which is very less indicating that there is no much relation between the trip_miles that this the distance of the trip and the tolls doesn't really correlate with each other, indicating that most taxi trips are in the city and tolls are not so involved.

Correlation between Response and predictor variables

```
cor(train$trip_total,train$trip_miles)

## [1] 0.9053971

cor(train$trip_total,train$trip_minutes)

## [1] 0.5130485
```

From the above correlation result it is evident that there is 0.90 correlation between trip total and trip miles. But very less correlation between trip_total and trip_minutes

Making predictions

```
test <-test %>%
  add_predictions(multiple_reg)
```

We have added our predictions to the test dataset, since we have built three models, we will look at all the MSE values together and decide which model can be used best for prediction.

INTERACTION EFFECTS MODEL

Building the model

```
interaction_effect<-lm(trip_total ~ trip_miles*tips,
                       data=train)
```

After running the linear model and multiple regression model, we would like to look at the interaction effects, that is we know that there is a relationship between trip_total and trip_miles, Now as a taxi company we want to add variables to this relationship and check how the relationship varies, we are adding this moderator, and are interested to see if it makes the relation better or not. The variable that we added is the tips, now we are interested to see if tips, that are primarily considered or given based on the ride, customer satisfaction and drivers behavior really affect the relation between trip_total and trip_miles.

Assessing the model numerically

```
glance(interaction_effect)
```

```
## # A tibble: 1 x 11
##   r.squared adj.r.squared sigma statistic p.value    df  logLik    AIC
BIC
##       <dbl>         <dbl> <dbl>     <dbl>   <dbl> <int>   <dbl>  <dbl>  <d
bl>
## 1     0.881         0.881  6.17  1256506.       0     4 -1.65e6 3.29e6 3.2
9e6
## # ... with 2 more variables: deviance <dbl>, df.residual <int>
```

```
summary(interaction_effect)
```

```
##
## Call:
## lm(formula = trip_total ~ trip_miles * tips, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -211.89   -2.37   -0.66    0.57  456.42
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      4.1109677  0.0128689   319.5   <2e-16 ***
## trip_miles       2.4915778  0.0021069  1182.6   <2e-16 ***
## tips             2.5300198  0.0058800   430.3   <2e-16 ***
## trip_miles:tips -0.0735333  0.0004084  -180.1   <2e-16 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.17 on 508298 degrees of freedom
## Multiple R-squared:  0.8812, Adjusted R-squared:  0.8812
## F-statistic: 1.257e+06 on 3 and 508298 DF,  p-value: < 2.2e-16
```

Looking at the numerical estimates it is clear that the R-square has increased due to the interaction effect, the sigma however does not show considerable amount of decrease. The value of p is less than alpha so the model is significant. Coming to the coefficients the value is -0.07 implying that there is a negative effect, if there is an increase in interaction term by 1, then there is 0.07 decrease in the trip_total.

Adding diagnostics

```
interaction_effect_results <- augment(interaction_effect,train)
```

Checking the correlation

```
cor(train$trip_miles,train$tips)^2
```

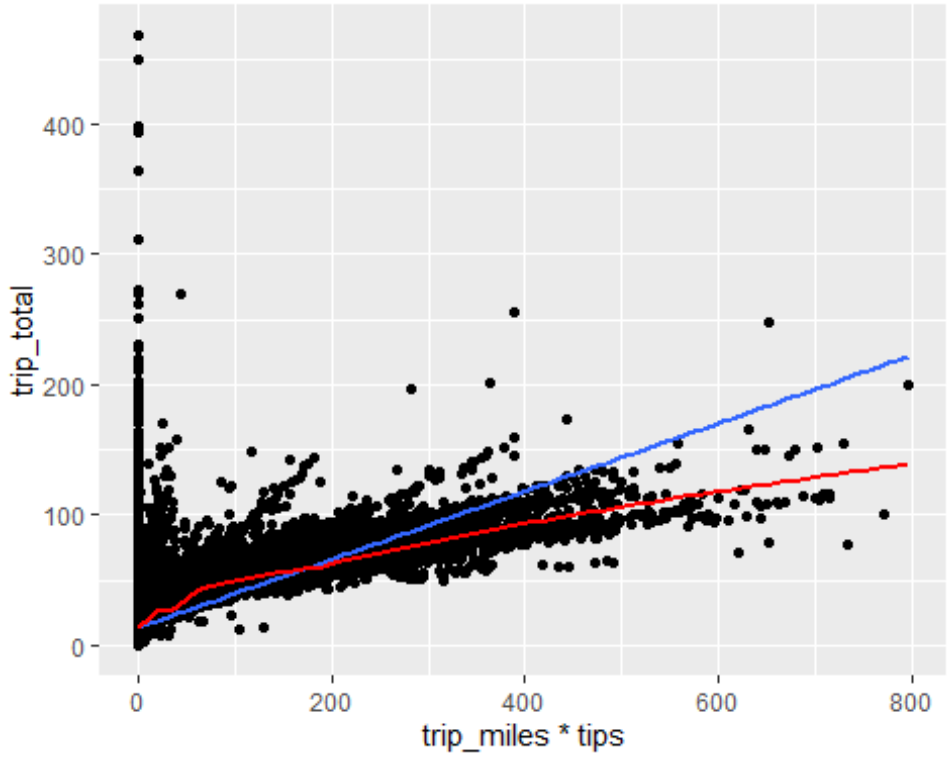
```
## [1] 0.3083919
```

In our dataset, it is no surprise if there is correlation between various variables, because the trip_miles and tips can have a correlation. For example, if the ride is really long, then the customer might feel that the driver has taken a lot of effort in driving safe and in time, so he might tip him more amount of money. The value of correlation turns out to be 0.30 which is a small value.

Assessing the model visually

```
train %>% sample_n(300000) %>%
ggplot(aes(trip_miles*tips,trip_total)) +
  geom_point() +
  geom_smooth(method = "lm") +
  geom_smooth(se = FALSE, color = "red")
```

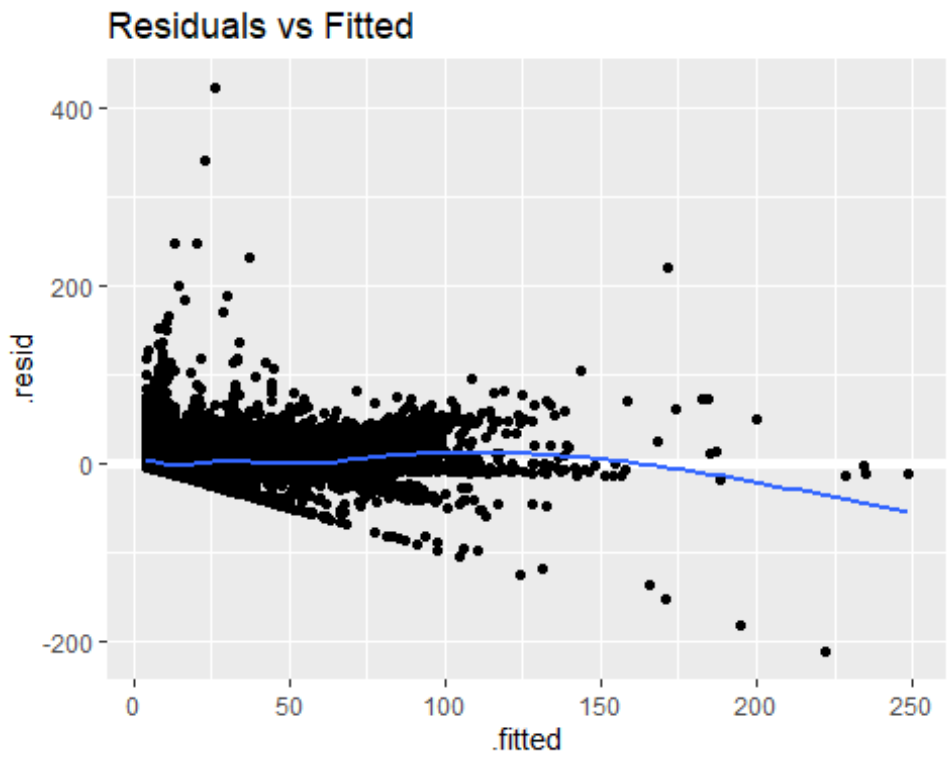
```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



We have visualized the model for only a sample of 300K observations, due to run time issues, the graph implies that there is a linear relationship between the predictor product and the response variable, however the relationship is perfect because the red line that indicates our model does follow the blue reference line.

Residual and Homoskedasticity check

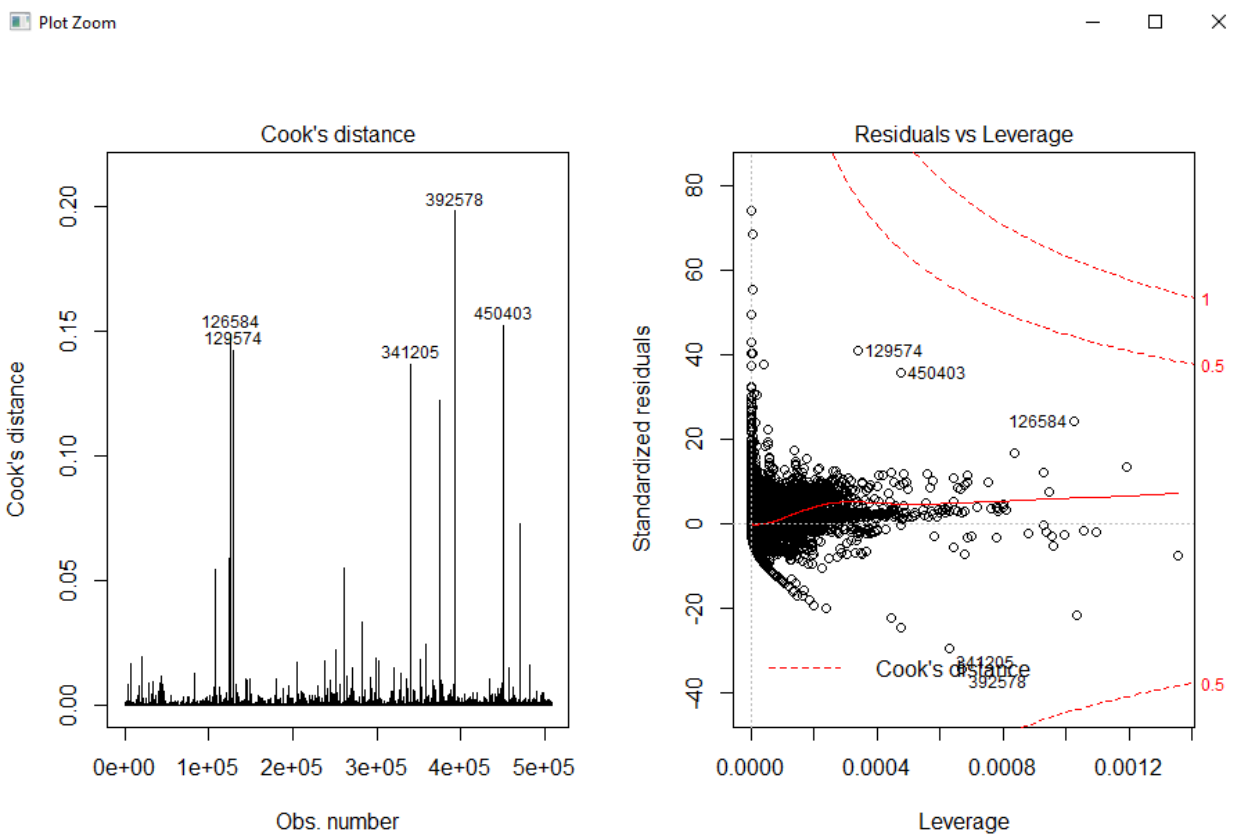
```
interaction_effect_results_comparison<-interaction_effect_results %>%
  sample_n(300000) %>%
  mutate(model="interaction_effect") %>%
  rbind(multiple_reg_result %>%
  sample_n(300000)) %>%
  rbind(linear_reg_result %>%
  sample_n(300000))

interaction_effect_results %>%
  sample_n(300000) %>%
  ggplot(aes(.fitted, .resid)) +
  geom_ref_line(h = 0) +
  geom_point() +
  geom_smooth(se = FALSE) +
  ggtitle("Residuals vs Fitted")

## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



We have taken only 300k samples from our dataset of one million rows to visualize the funnel effect due to runtime issues. From the graph it is clear that there is the funnel effect for the negative values, however there is no funnel effect for the positive values indicating that there is no heteroskedasticity.

Cooks D for influential cases

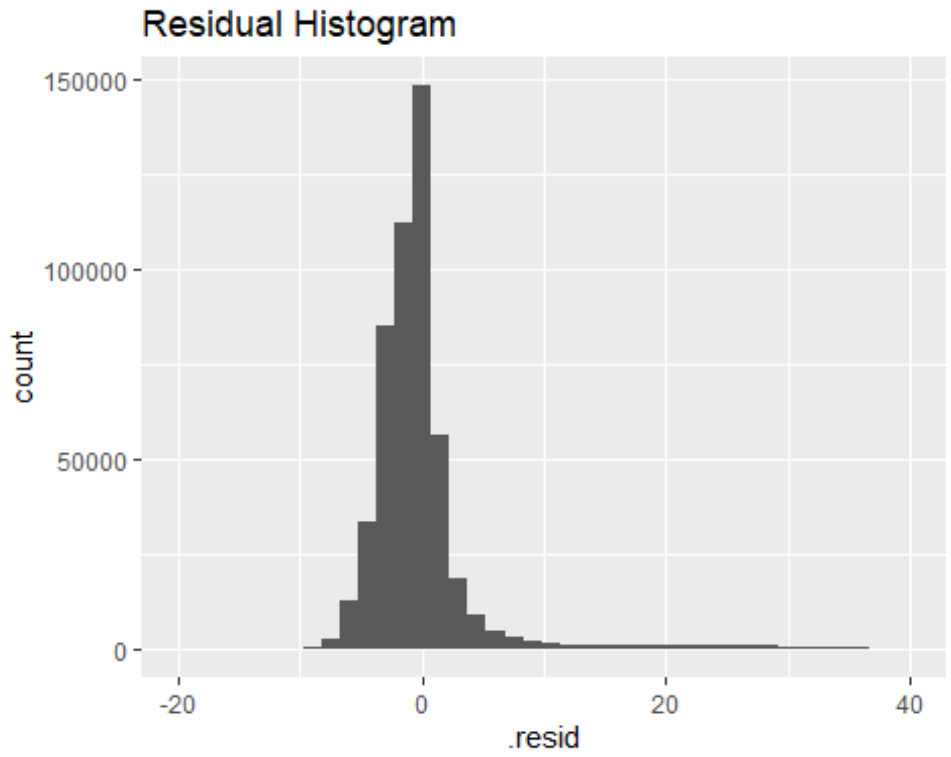
```
par(mfrow=c(1, 2))

plot(interaction_effect, which = 4, id.n = 5)
plot(interaction_effect, which = 5, id.n = 5)
```



From the graph it is evident that there some influential cases and our model can predict better if these observations are eliminated from the dataset. The influential cases mean that these points have the potential to affect the linearity of the model and so if these observations are not taken into consideration, the model will exhibit better linearity.

Residual histogram

```
ggplot(interaction_effect_results, aes(.resid)) +
  scale_x_continuous(limits = c(-20,40))+
  geom_histogram(binwidth = 1.5) +
  ggtitle("Residual Histogram")
```



From the graph it is evident that there are residues that have a count of 150K, from the residual histogram plot it can be deducted that our model normal distribution up to a certain level, but it is definitely not the perfect model.

Making prediction

```
test <-test %>%
  add_predictions(interaction_effect)
```

We are adding the predictions of this model to the train dataset, we will use the MSE value and decide out of the three models we built which one would be better for prediction and can be used further.

Comparison of the models

Assesing the models numerically

```
list(linear_reg = broom::glance(linear_reg),
     multiple_reg = broom::glance(multiple_reg),
     interaction_effect = broom::glance(interaction_effect))

## $linear_reg
## # A tibble: 1 x 11
##   r.squared adj.r.squared sigma statistic p.value    df  logLik    AIC
BIC
##       <dbl>         <dbl> <dbl>     <dbl>   <dbl> <int>   <dbl>  <dbl>  <d
bl>
```

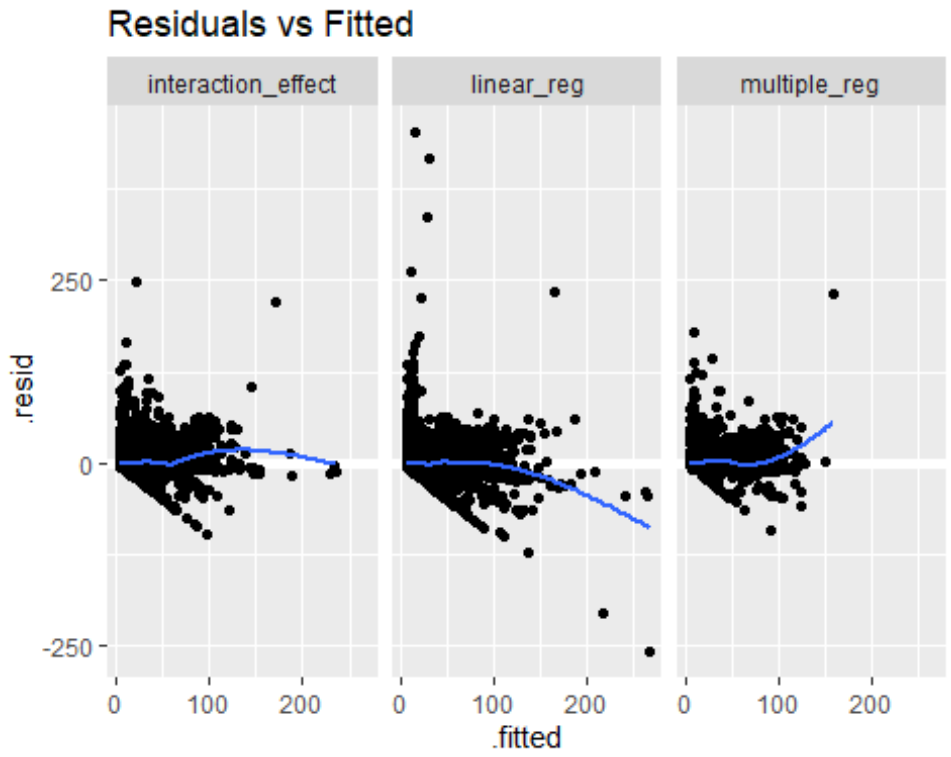
```
## 1     0.820         0.820  7.60  2311578.       0     2 -1.75e6 3.50e6 3.5
0e6
## # ... with 2 more variables: deviance <dbl>, df.residual <int>
##
## $multiple_reg
## # A tibble: 1 x 11
##   r.squared adj.r.squared sigma statistic p.value    df  logLik    AIC
BIC
##       <dbl>         <dbl> <dbl>     <dbl>   <dbl> <int>   <dbl>  <dbl>  <d
bl>
## 1     0.874         0.874  6.36  1173037.       0     4 -1.66e6 3.32e6 3.3
2e6
## # ... with 2 more variables: deviance <dbl>, df.residual <int>
##
## $interaction_effect
## # A tibble: 1 x 11
##   r.squared adj.r.squared sigma statistic p.value    df  logLik    AIC
BIC
##       <dbl>         <dbl> <dbl>     <dbl>   <dbl> <int>   <dbl>  <dbl>  <d
bl>
## 1     0.881         0.881  6.17  1256506.       0     4 -1.65e6 3.29e6 3.2
9e6
## # ... with 2 more variables: deviance <dbl>, df.residual <int>
```

Looking at the numerical summary, it is evident that the Rsquare value increases from linear to multiple and it is the highest for interaction effect model. Likewise, the interaction model has less RSE when compared to multiple and linear.

Funnel Effect comparison

```
interaction_effect_results_comparison %>%
  sample_n(300000) %>%
  ggplot(aes(.fitted, .resid)) +
  geom_ref_line(h = 0) +
  geom_point() +
  geom_smooth(se = FALSE) +
  facet_wrap(~ model) +
  ggtitle("Residuals vs Fitted")
```

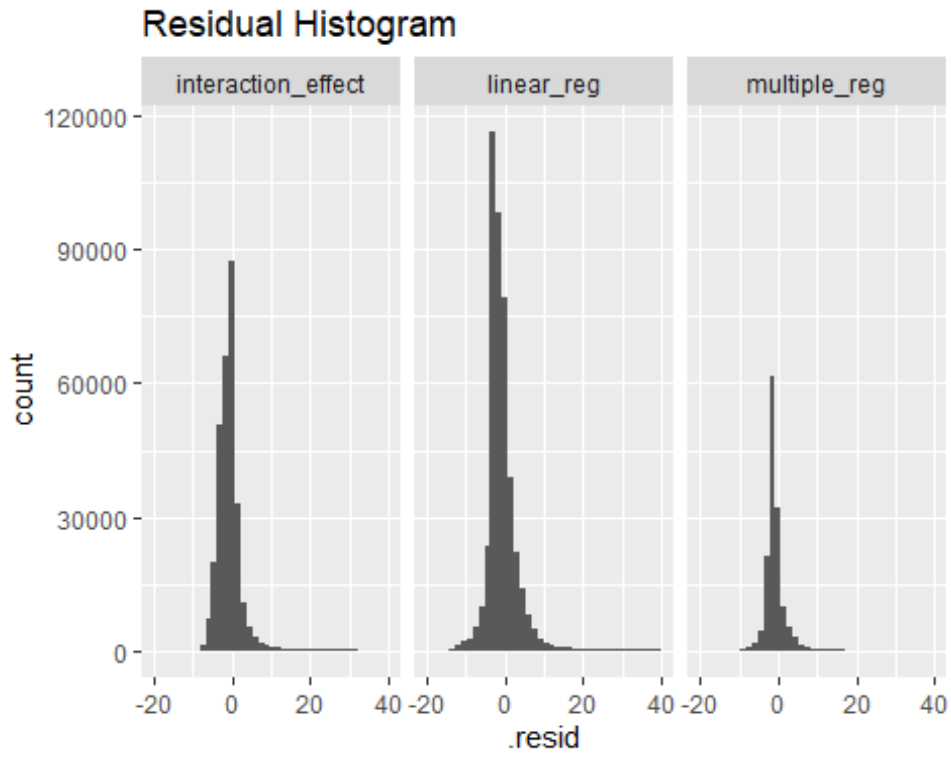
```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



From the graph, it is evident that there is heteroskedasticity in all the three models, one prominent observation we can make by looking at these graphs is that the funnel effect exists only in the negative values and there is no funnel effect in the positive values above the axis.

Residual Histogram comparison of all three models

```
ggplot(interaction_effect_results_comparison, aes(.resid)) +
 scale_x_continuous(limits = c(-20,40))+
 geom_histogram(binwidth =1.5) +
 facet_wrap(~ model, scales = "free_x") +
 ggtitle("Residual Histogram")
```



From the above graphs it is evident that the residual histograms follow the normal distribution up to a certain extent, the linear regression model has higher residual values when compared to the multiple regression model indicating that multiple regression does a better job than linear regression.

Making predictions and Calculating MSE for all the models to choose the best model.

```
test %>%
  gather_predictions(linear_reg,multiple_reg,interaction_effect) %>%
  group_by(model) %>%
  summarise(MSE = mean((trip_total-pred)^2)) %>%
  arrange(desc(MSE))
```

```
## # A tibble: 3 x 2
##   model                MSE
##   <chr>              <dbl>
## 1 linear_reg          57.0
## 2 multiple_reg        39.5
## 3 interaction_effect  37.2
```

The MSE values of the three models are 56.5, 39.0 and 36.6. The interaction effect regression model has the lowest MSE indicating that it is a good model and can be used for further analysis.

Calculating the VIF to check for multicollinearity

```
car::vif(multiple_reg)

## trip_miles       tips      tolls
##   1.446200   1.445934   1.000425
```

The VIF values for the multiple regression indicate the multicollinearity effect between the variables, according to the rule of thumb VIF>5 indicates that there is multicollinearity, but for our model the values are not even close to 5, indicating no multicollinearity.

LOGISTIC REGRESSION

Logistic Regression is done by analyzing which variables predict the probability of high and low fare values. The distribution of fare variable is considered such that if the fare value is above the mean fare value, it is assigned to 1 else 0. With this the fare variable becomes binary categorical with 2 values 0 and 1. This is done by creating an extra column in the dataset named "fare_high". The variables trip_minutes, tips and trip_miles are considered for predicting the distribution of fare_high variable using logistic regression.

Data Preparation

Selecting the required columns from the dataset and filtering the dataset for better and improved results.

```
trips_2019_logistic <- trips_2019_1 %>%
  filter(trip_miles < 100) %>%
  filter(trip_minutes < 10000/60) %>%
  filter(tips <= 30) %>%
  filter(fare < 400) %>%
  select(trip_minutes, trip_miles, fare)
```

Removing NA's from the data.

```
trips_2019_logistic <- na.omit(trips_2019_logistic)
```

Calculating the mean fare for the data

```
mean_fare <- trips_2019_logistic %>%
  summarise(
    mean_fare = mean(fare, na.rm = T)
  )
```

Creating the fare_high response variable based on the mean fare value.

```
trips_2019_logistic <- trips_2019_logistic %>%
  mutate(
    fare_high = ifelse(fare > mean_fare$mean_fare, 1, 0)
  )
```

Splitting our dataset into training (60%) and testing (40%) datasets.

```r
set.seed(1234)

trips_logistic <- sample(c(TRUE, FALSE),
                         nrow(trips_2019_logistic),
                         replace = T,
                         prob = c(0.6,0.4))

train_logistic <- trips_2019_logistic[trips_logistic, ]
test_logistic <- trips_2019_logistic[!trips_logistic, ]
```

Generating the Logistic Regression model

```r
logistic_model <- glm(fare_high ~ trip_miles + trip_minutes,
                      family = "binomial",
                      data = train_logistic)
```

Assessing the coefficients of the generated logistic model

```r
summary(logistic_model)
```

```
##
## Call:
## glm(formula = fare_high ~ trip_miles + trip_minutes, family = "binomial",
##     data = train_logistic)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -8.4904  -0.2453  -0.1324   0.0001   3.6113
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -6.5192375  0.0188574  -345.7   <2e-16 ***
## trip_miles    0.6372854  0.0031088   205.0   <2e-16 ***
## trip_minutes  0.2232991  0.0009623   232.0   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 696359  on 607448  degrees of freedom
## Residual deviance: 189441  on 607446  degrees of freedom
## AIC: 189447
##
## Number of Fisher Scoring iterations: 8
```

```r
glance(logistic_model)
```

```
## # A tibble: 1 x 7
##   null.deviance df.null  logLik     AIC     BIC deviance df.residual
##           <dbl>   <int>   <dbl>   <dbl>   <dbl>    <dbl>       <int>
## 1       696359.  607448 -94720. 189447. 189481.  189441.      607446
```

```
tidy(logistic_model)
```

```
## # A tibble: 3 x 5
##   term         estimate std.error statistic p.value
##   <chr>           <dbl>     <dbl>     <dbl>   <dbl>
## 1 (Intercept)    -6.52   0.0189       -346.       0
## 2 trip_miles      0.637  0.00311       205.       0
## 3 trip_minutes    0.223  0.000962      232.       0
```

```
exp(coef(logistic_model))
```

```
##  (Intercept)   trip_miles trip_minutes
##  0.001474793  1.891339586  1.250194429
```

From the model summary we can see that all the variables trip_minutes and trip_miles have P<alpha which means that all these three variables are significant predictors of the generated model. Also, we can see that the residual deviance is less than the null deviance implying that the model is a good fit model.

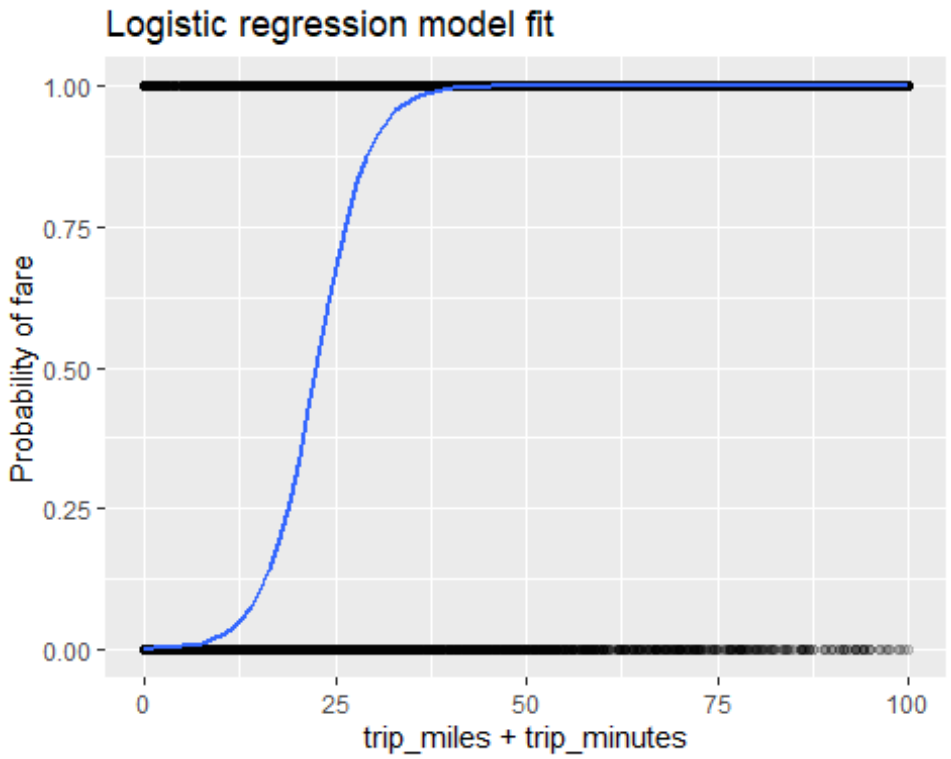
The coefficients indicate that increase in trip_miles and trip_minutes will contribute to the probability of high fare. One unit increase in trip_miles will increase the log odds of default by 0.64 units. Similarly, one unit increase in trip_minutes will increase the log odds of default by 0.22 units.

Also, from exponent of coefficients we can say that for one-mile increase in trip_miles variable, the odds of high fare rise by a factor 1.9; for one-minute increase in trip_minutes, the odds of high fare will rise by a factor 1.25.

These are the predicted values for all the trips. That means that these are the market trend values of the taxis for the year 2019. Assessing all the above values helps our company in defining the fare range based on miles and minutes travelled by a customer. Based on the current market fare values our company can decide and fix if the fare should be above the mean fare or below the mean fare based on the trip duration and miles travelled.

Assessing the logistic model visually:

```
train_logistic %>%
  ggplot(aes(trip_miles + trip_minutes, fare_high)) +
  scale_x_continuous(limits = c(0,100)) +
  geom_point(alpha = .15) +
  geom_smooth(method = "glm", method.args = list(family = "binomial")) +
  ggtitle("Logistic regression model fit") +
  xlab("trip_miles + trip_minutes") +
  ylab("Probability of fare")
```



Checking for multicollinearity issues between the predictor variables trip_miles and trip_minutes.

```
car::vif(logistic_model)
```

```
##   trip_miles trip_minutes
##     1.001737     1.001737
```

It is cleat that both the variables trip_miles and trip_minutes are not correlated. Assessing the importance of trip_miles and trip_minutes variables in predicting the fare_high variable.

```
caret::varImp(logistic_model)
```

```
##                Overall
## trip_miles   204.9946
## trip_minutes 232.0359
```

It is clear from the results that trip_minutes is the most important predictor of the model followed by trip_miles. There is only slight difference between them so our company should focus on both the variables.

Predictions with the model:

The model is predicted for the probability of high fare for the trips that travel 10-20 miles for 10-40 minutes

```
predict(logistic_model, data.frame(trip_miles = c(5,10), trip_minutes = c(10,
25)), type = "response")
```

```
##         1         2
## 0.2497736 0.9956624
```

As the number of miles increase from 5-10 and the corresponding trip travel time increases from 5min to 10min the probability of high fare is increasing from 25% to 99%. With this our company has a clear idea that as the trip miles becomes nearer to 10 miles and the trip duration becomes nearer to 25 min, the probability of high fare is 99%. Our company can easily decide upon setting up higher fares above the mean fare based on this relationship.

Goodness-of-fit:

Pseudo R-square:

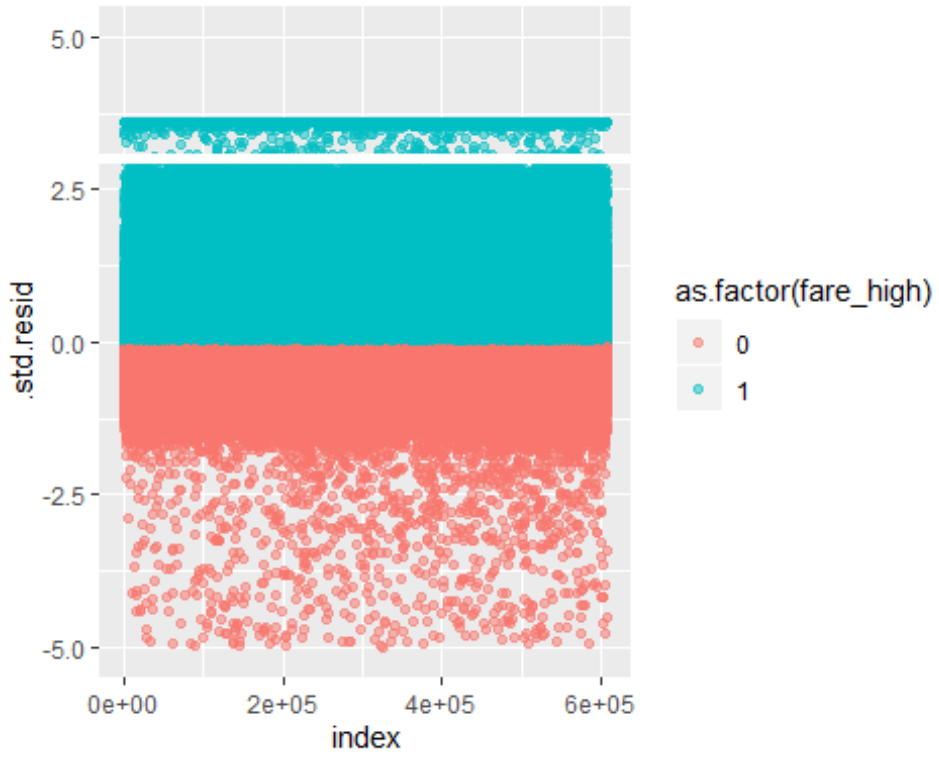
```
pscl::pR2(logistic_model)["McFadden"]

## fitting null model for pseudo-r2

##  McFadden
## 0.7279552
```

The pseudo R-square value for our logistic model is obtained as 73% which represents that our model is a very good fit model for assessing the probability of higher fares.

Residual Assessment:

We have done assessment of residuals to check how many residuals exceeded 3 standard deviations. The graph displays the residuals that exceeded 3 standard deviations.

```
logistic_model_data <- augment(logistic_model) %>%
  mutate(index = 1:n())

ggplot(logistic_model_data, aes(index, .std.resid, color = as.factor(fare_hig
h))) +
  scale_y_continuous(limits = c(-5,5)) +
  geom_point(alpha = .5) +
  geom_ref_line(h = 3)
```



Assessing the residuals that exceeded 3 standard deviations:

```
logistic_model_data %>%
  filter(abs(.std.resid) > 3)
```

```
## # A tibble: 3,126 x 11
##    fare_high trip_miles trip_minutes .fitted .se.fit .resid    .hat .sigma
##        <dbl>      <dbl>        <dbl>   <dbl>   <dbl> <dbl>   <dbl>  <dbl>
##  1         1          0            0       0   -6.52  0.0189   3.61 5.23e-7  0.558
##  2         1          0            0       0   -6.52  0.0189   3.61 5.23e-7  0.558
##  3         1          0            0       0   -6.52  0.0189   3.61 5.23e-7  0.558
##  4         1          0            3       0   -5.85  0.0166   3.42 7.86e-7  0.558
##  5         1          0            0.267   0   -6.46  0.0186   3.59 5.43e-7  0.558
##  6         1          0            1.98    0   -6.08  0.0173   3.49 6.86e-7  0.558
##  7         1          0            3       0   -5.85  0.0166   3.42 7.86e-7  0.558
##  8         1          0            0       0   -6.52  0.0189   3.61 5.23e-7  0.558
##  9         1          0            0.167   0   -6.48  0.0187   3.60 5.35e-7  0.558
## 10         1          0            0       0   -6.52  0.0189   3.61 5.23e-7  0.558
## # ... with 3,116 more rows, and 3 more variables: .cooksd <dbl>,
## #   .std.resid <dbl>, index <int>
```

```
logistic_model_data %>%
  mutate(std.resid_above_3 = ifelse(abs(.std.resid) >= 3, 1,0)) %>%
  group_by(std.resid_above_3) %>%
  count() %>%
  ungroup() %>%
  mutate(
```

```
    frac = n/sum(n)
  )
```

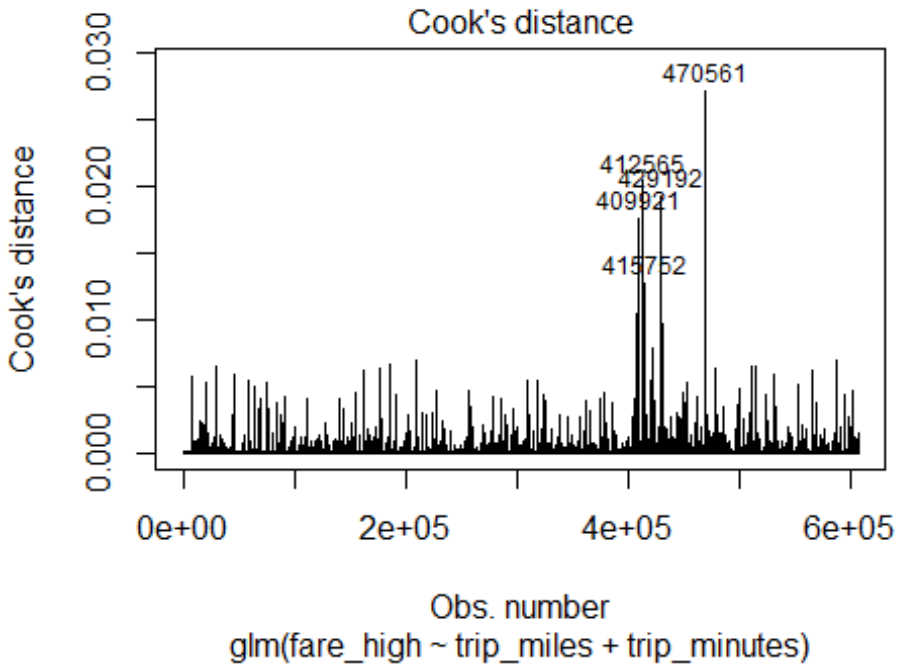
```
## # A tibble: 2 x 3
##   std.resid_above_3      n    frac
##               <dbl>  <int>   <dbl>
## 1                 0 604323 0.995
## 2                 1   3126 0.00515
```

There are 0.5% of observations that are inconsistent as the fare is high for trips with zero minutes and zero miles. So, removing these observations and re-running the whole model definitely yields a better model.

Cook's Distance Plot:

The top 5 most influential observations are plotted using Cook's D plot.

```
plot(logistic_model, which = 4, id.n = 5)
```




```
logistic_model_data %>%
  top_n(5, .cooksd)
```

```
## # A tibble: 5 x 11
##   fare_high trip_miles trip_minutes .fitted .se.fit .resid     .hat .sigma
##       <dbl>      <dbl>        <dbl>   <dbl>   <dbl>  <dbl>    <dbl>  <dbl>
## 1         0       76.5         17.2    46.1   0.229  -8.49 1.17e-17  0.558
## 2         0       69.8        145.     70.4   0.247  -8.49 1.36e-17  0.558
## 3         0       53.9        127.     56.2   0.195  -8.49 8.47e-18  0.558
```

```
## 4         0       80.1          7      46.1   0.240  -8.49 1.28e-17  0.558
## 5         0       94.5          5      54.8   0.285  -8.49 1.80e-17  0.558
## # ... with 3 more variables: .cooksd <dbl>, .std.resid <dbl>, index <int>
```

These are the trips that travelled very large distances (above 50 miles), but their fare value is below the mean value ($14). Omitting these values from the dataset definitely yields a better fit model.

Validation of Predicted Values

The test data is predicted based on the model generated by the training data.

```
test_logistic_predicted <- predict(logistic_model,
                                   newdata = test_logistic,
                                   type = "response")
```

Confusion Matrix:

```
table(test_logistic$fare_high, test_logistic_predicted > 0.5) %>% prop.table(
)

##
##           FALSE        TRUE
##   0 0.730924776 0.009970873
##   1 0.034838764 0.224265587
```

True-Positives:

The cases which we predicted the fare will be high, and it was actually high. ###### True-Negatives: The cases which we predicted no high fares, and the fares were as not high as per our prediction. ###### False Positives (type-I error): The cases which we predicted high fares, but they were not actually high fares ###### False Negatives (type-II error): The cases which we predicted there are no high fares, but there are high fares

For our data, 73% of the data are true negatives and 22% were true positives. type-I and type-II errors are less than 1% and type-II errors are less than 4%. This training model can be used to predict 73% of low fares (less than $14.3) and 22% of high fares.

Misclassification rate

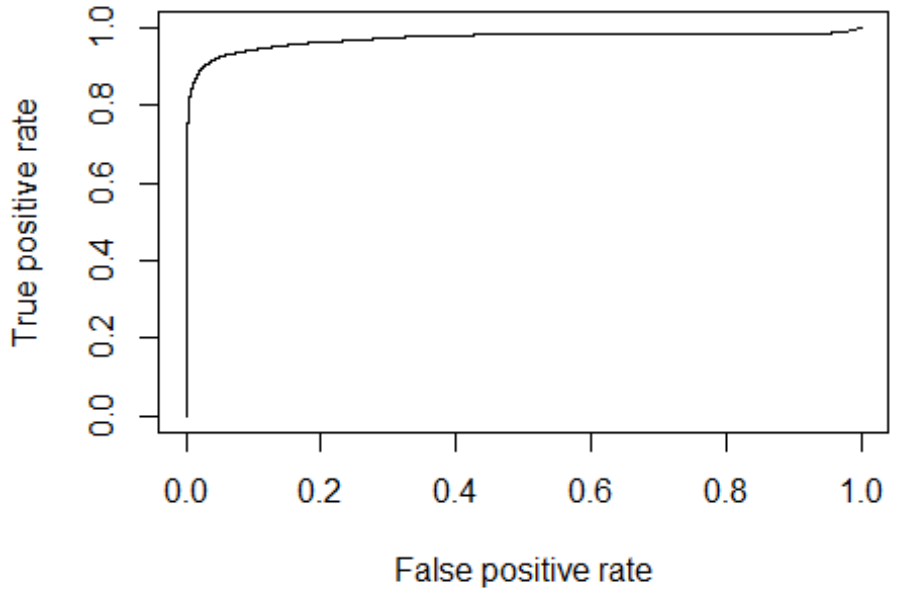
```
test_logistic %>%
  mutate(pred = ifelse(test_logistic_predicted > 0.5, 1, 0)) %>%
  summarise(
    error = mean(fare_high != pred)
  )

## # A tibble: 1 x 1
##    error
##    <dbl>
## 1 0.0448

table <- table(test_logistic$fare_high,
               test_logistic_predicted > 0.5)
```

```
low_fares_correctly_identified <- 1-table[1,2]/(table[1,1]+table[1,2])
```

We see that 98.6% of the low fare values are correctly identified (this is because more percentage of our data (73%) are low fares). For our company we consider this as "Specificity" as we are more concerned in increasing the fares to make profits.

Receiving Operation Characteristic (ROC)

```
library(ROCR)
prediction(test_logistic_predicted, test_logistic$fare_high) %>%
  performance(measure = "tpr", x.measure = "fpr") %>%
  plot()
```



Computation of AUC

```
prediction(test_logistic_predicted, test_logistic$fare_high) %>%
  performance(measure = "auc") %>%
  .@y.values

## [[1]]
## [1] 0.970925
```

The graph shows that the area under the curve is high and the corresponding AUC value is calculated as 97%. This means that our model is a very good classification model.
Conclusion: This training model can be used to predict 73% of low fares (less than $14.3) and 22% of high fares accurately.

K-MEANS CLUSTERING

K-means clustering is done to identify the clusters for the variables trip miles and trip minutes.

Data Preparation

Selecting required variables for clustering A random sample of 300K observations is taken due to memory and heavy run time issues

```
cluster_trips <- trips_2019_1 %>%
  select(trip_minutes, trip_miles) %>%
  filter(trip_miles < 70) %>% filter(trip_minutes<100) %>%
  sample_n(300000)
```

Removing NA's from the data

```
cluster_trips <- na.omit(cluster_trips)
```

Scaling/standardization of data as trip miles and minutes have different units

```
cluster_trips <- scale(cluster_trips)
```

(The distance matrix code is not used because we faced memory error "Cannot allocate vector of size 335.3 Gb". Even after sampling the code to 10K records, the error is the same).

Determine number of Optimal Clusters (Elbow method)

Elbow method is used for identifying the optimal number of clusters.

```
set.seed(1234)
```

Function to compute total within-cluster sum of square

```
wss <- function(k) {
  kmeans(cluster_trips, k, nstart = 10 )$tot.withinss
}
```

Compute and plot wss for k = 1 to k = 15

```
k.values <- 1:15
```

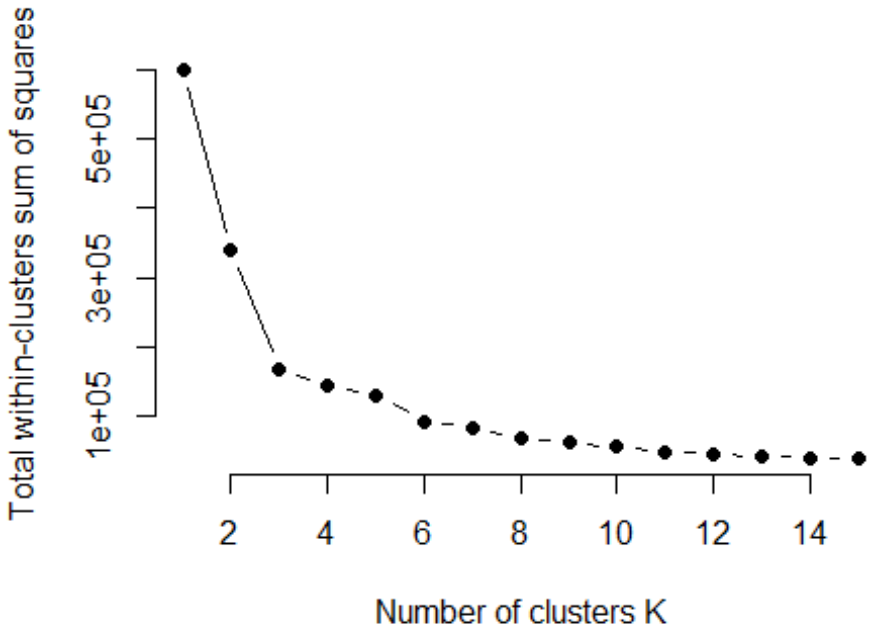
Extract wss for 2-15 clusters

```
wss_values <- map_dbl(k.values, wss)
```

Plot for identifying the optimal number of clusters based on Elbow method:

```
plot(k.values, wss_values,
     type="b",
     pch = 19,
     frame = FALSE,
```

```
     xlab="Number of clusters K",
     ylab="Total within-clusters sum of squares")
```



Inference: From the graph we estimated the optimal number of clusters as K=3.

K-means clustering with K=3 using Spark

K-Means Clustering is performed by pre-defining the number of clusters as K=3 in Spark.

Connecting to spark

```
sc <- spark_connect(master = "local", version = "2.3")
```

Copying the data to Spark

```
cluster_trips_spark <- sdf_copy_to(sc, cluster_trips, overwrite = TRUE)
```

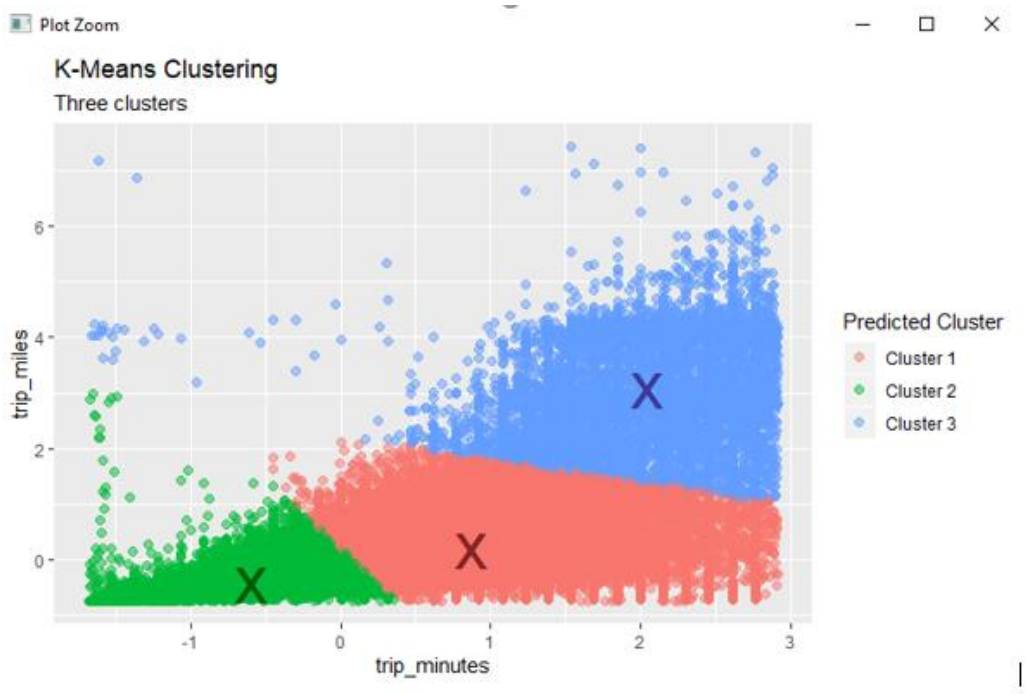
Running the code for K-means clustering in Spark with K=3

```
cluster_kmeans_spark <- ml_kmeans(cluster_trips_spark,
                                  trips ~ .,
                                  k=3,
                                  init_steps = 20)
```

Plot cluster membership for 3 clusters. (Due to heavy run time issues the code is commented and the results are posted below manually).

```
#ml_predict(cluster_kmeans_spark) %>%
#  collect()  %>%
```

```
#  ggplot(aes(trip_minutes, trip_miles)) +
#  geom_point(aes(trip_minutes, trip_miles, col = factor(prediction + 1)),
#             size = 2, alpha = 0.5) +
#  geom_point(data = cluster_kmeans$centers, aes(trip_minutes, trip_miles),
#             col = scales::muted(c("red", "green", "blue")),
#             pch = 'x', size = 12) +
#  scale_color_discrete(name = "Predicted Cluster",
#                       labels = paste("Cluster", 1:3)) +
#  labs(
#    x = "trip_minutes",
#    y = "trip_miles",
#    title = "K-Means Clustering",
#    subtitle = "Three clusters"
#  )
```



Disconnecting the Spark connection

```
spark_disconnect(sc)
```

Inference: Utilizing K-Means clustering, we are able to better determine the “clusters” that are found within the taxi market pertaining to trip minutes and trip miles. This information is valuable to us as it helps us predict our positioning within the clusters. We can also use this information to better figure out how long and far our competitors’ trips are lasting based on the sizing of the cluster which can also help us discover what the market demand is currently at.

The relationship between trip_miles and trip_minutes in predicting high fair values is valuable information for us so that we are able to set optimal pricing. Because in the city we will be transporting customers shorter distances, the traffic may result in higher minutes which would ultimately result in a higher fare. However, on days of high volume, we may also see an increase in trip miles which would also result in higher trip minutes which would also result in higher fares. Ultimately, this relationship helps us set better, more optimal pricing for both the company and the customer.

SUPPORT VECTOR MACHINE

Through SVM, we are trying to classify the variables trip_total and trip_miles based on the classification variable payment type. The classification variable is classified into two classes "Cash" and "Credit Card" (only 2 classes are considered because these are the prominent payment types customers are using in the current market). Pertaining to these classes we are trying to classify the trip_total and trip_miles variables using SVM.

Data Preparation

Only the necessary data is selected from the dataset. The analysis is done for a sample of 100K observations from the dataset due to heavy runtime and memory issues.

```
trips_svm <- trips_2019 %>%
  filter(payment_type == "Cash" | payment_type == "Credit Card") %>%
  select(trip_miles, trip_total, payment_type) %>%
  sample_n(100000)
```

Converting the classification variable into factors.

```
trips_svm$payment_type <- factor(trips_svm$payment_type,
                                 levels = c("Cash","Credit Card"))
```

Removing NA values from the data.

```
trips_svm <- na.omit(trips_svm)
```

Scaling the data as trip_total and tips are variables that have different units.

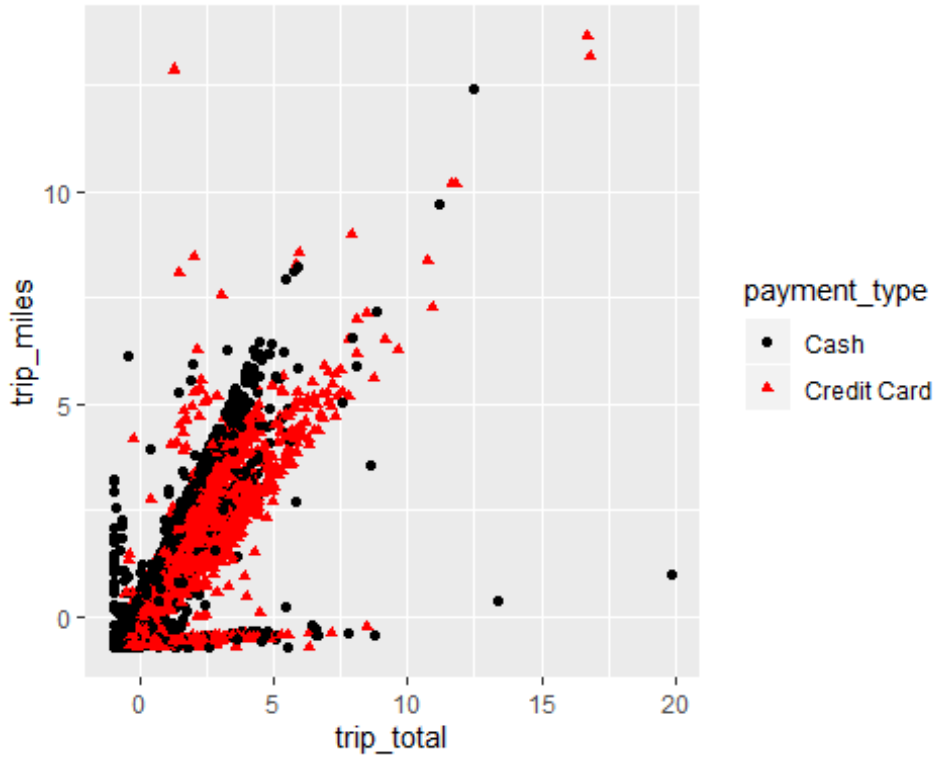
```
trips_svm[-3] <- scale(trips_svm[-3])
```

Partitioning the dataset into training and testing data with 70%/30% variation.

```
set.seed(1234)
svm_split <- sample(c(TRUE, FALSE), nrow(trips_svm), replace = T, prob = c(0.
7,0.3))
svm_training <- trips_svm[svm_split, ]
svm_test <- trips_svm[!svm_split, ]
```

Plotting the trip_total and trip_miles variables with respect to payment_type in order to see the distribution of the variables over the axis.

```
ggplot(data = svm_training, aes(x = trip_total, y = trip_miles, color = payme
nt_type, shape = payment_type)) +
  geom_point() +
  scale_color_manual(values=c("#000000", "#FF0000"))
```



Now the goal of SVM is to divide these scattered points using a boundary so that the total distance between the line and closest point in each class is maximized.

As the values are non-linearly arranged in the plane, the cost of having a point in the wrong side of the boundary can be determined using the tune() function. (As we are facing severe run time issue with the tune() function, we have just taken a sample from the training set in order to run the tune() function).

```
svm_training_sample<-svm_training %>%
  sample_n(5000)

tune.out <- tune(svm,
                 payment_type ~ .,
                 kernel = "linear",
                 data = svm_training_sample,
                 ranges = list(cost = c(0.1, 1, 5, 10, 100)))

(bestmod <- tune.out$best.model)

##
## Call:
## best.tune(method = svm, train.x = payment_type ~ ., data = svm_training_sa
```

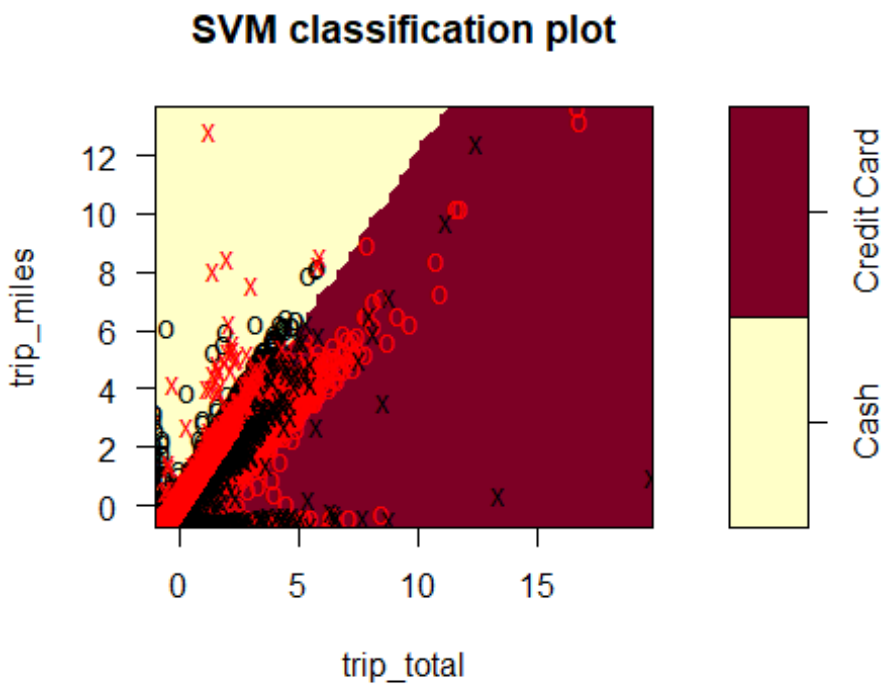
```
mple,
##     ranges = list(cost = c(0.1, 1, 5, 10, 100)), kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  100
##
## Number of Support Vectors:  3111
```

From the tune() function, the cost is assigned to '100', kernel to 'linear' and the SVM type to 'C-classification'. The SVM model is run with these inputs as follows:

```
svmfit_model <- svm(formula = payment_type ~ .,
                    data = svm_training,
                    kernel = "linear",
                    type = "C-classification",
                    cost = 100,
                    scale = F)
```

Visualising the results:

```
plot(svmfit_model, svm_training)
```



Running the prediction for the test dataset based on the model generated by the training data.

```
pred <- predict(svmfit_model,
                newdata = svm_test[-3])
```

Accuracy of the test model:

```
table(svm_test$payment_type, pred) %>%
  prop.table()

##              pred
##                     Cash Credit Card
##   Cash        0.41253576  0.07628585
##   Credit Card 0.11288176  0.39829663
```

The cases which the model predicted that the payment should be done by cash, and it is actually done by cash are 42%. The cases which the model predicted that the payment should be done by credit card, and it is actually done by credit card are 40%. The accuracy of our model is 81.5%

The cases which our model predicted that the payment should be done by credit card and it is actually done through cash are 11% and the cases which our model predicted that the payment type should be done by cash and it is actually done through credit card are 7%. Overall, there is an 18% chance that our model may not accurately predict the payments correctly.

For a given trip_miles and trip_total our model can predict whether a customer pays through Cash or Credit Card with 81.5% accuracy.

Inference:

Utilizing SVM, we we're able to set up the model to predict for a given trip_miles and trip_total, whether a customer will pay with Cash or Credit with an 81.5% accuracy. We can then utilize this valuable information to help us have a greater understanding of the payments methods that we can expect based on the trips our drivers may perform. We can map some potential routes from key businesses to see whether they will be paid for with cash or credit. This could help us perform some basic revenue projections for shareholders.

LESSONS LEARNT

Our take away from this project is mostly related to how well you understand the data and churn out the inferences from the output of the data. From the basic manipulations we learnt how to clean the data, wrangle the data and perform basic operations that help us to understand the trend or pattern the data follows.

From sophisticated models like regression and clustering we learnt the ability to use the data to build models that can be further in the future be used to predict any other data that is similar. Visualization techniques that we learnt through this project helped us understand the fact that pictures speak more than numbers or words and our information reaches across better due to these visualizations.

CONCLUSION

From our project, we have answered our research question that is analyzing the taxi trips in the year 2019, and using the results to our advantage to set up our taxi company TMAAP taxi in the year 2020. The results that we acquired from this project and the inferences that we concluded will help us achieve a competitive edge and also help us to keep our employees informed about all the trends and situations in the market.