Forecasting Lyft Fares Using Linear Regression

Jessica Padilla

December 13, 2019

Introduction

Lyft is one of the many rideshare services available to people today. Founded in 2012, Lyft allows users to order rides through the use of a phone app. There are 6 types of rides available: Shared Ride, Lyft, Lyft XL, Lux, Lux Black, and Lux Black XL (Lyft). This project will focus on the the basic Lyft service, which is the most commonly used ride.

Fares are generally determined by the distance for each ride. However, Lyft and other rideshare companies have adopted the practice of what is called price surging. This involves increasing the price of a ride because the demand for Lyft cars exceeds the number of Lyft cars that are actually in service (Lyft). This can result from many conditions such as densely populated locations, bad weather, and busy commute hours within the day. In this project, we will utilize a data set of Lyft rides within Boston, Massachusetts to see what factors contribute to price surges. We will, then, use the determined factors to forecast the price of any Lyft ride.

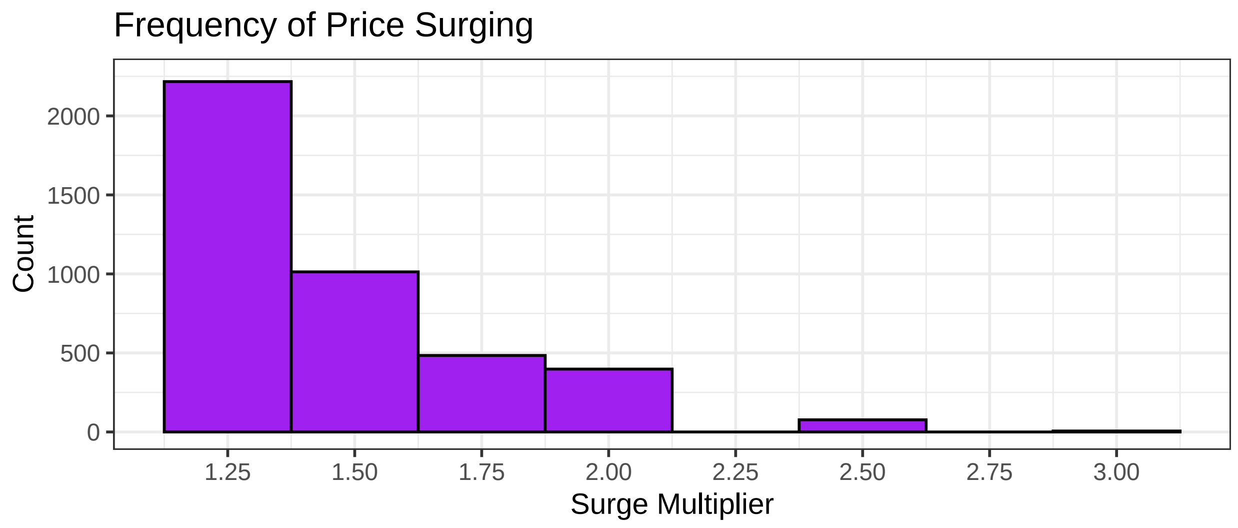
Methods

Lyft data was retrieved from Kaggle (https://www.kaggle.com/sliu65/data-mining-project-boston) and uploaded onto GitHub (https://github.com/jessicapadilla/lyft_fares/blob/master/lyft.csv.zip?raw=true). All of the data was imported into R. The tidyverse package was used for data cleaning, analysis, and visualization. The ggcorrplot and knitr packages was also used to enhance visualization.

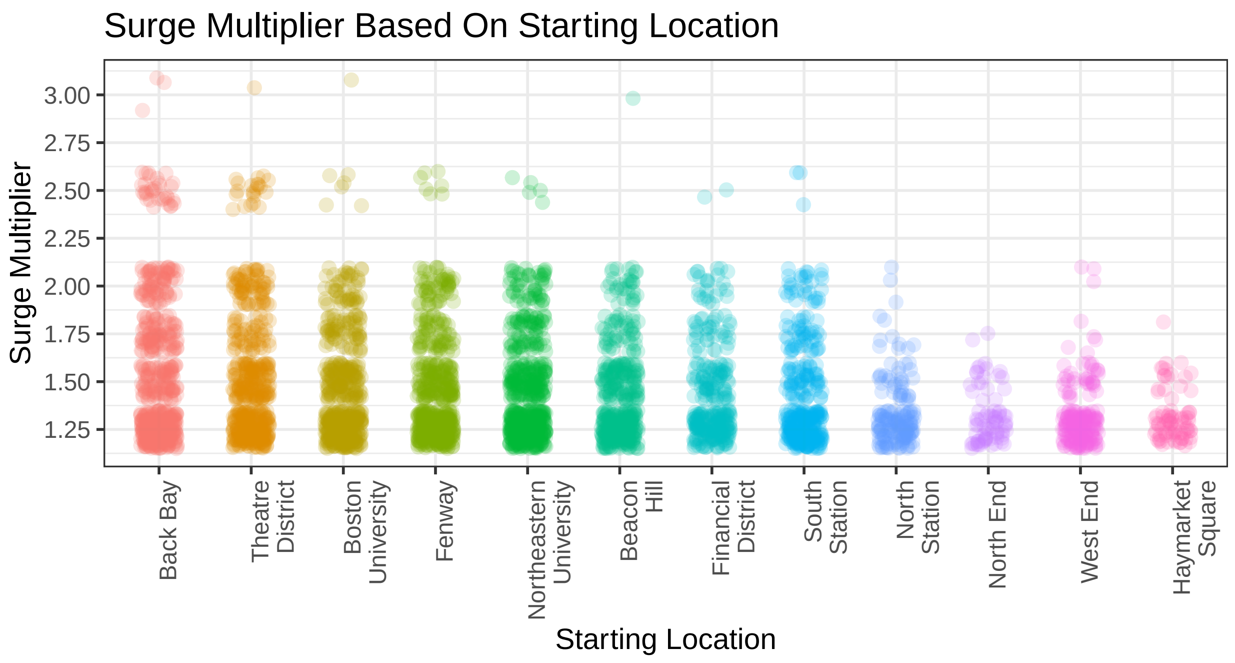
The R code for this project can be found within the Supplementary Materials section of this paper and on GitHub (https://github.com/jessicapadilla/lyft_fares/blob/master/code.R).

Results

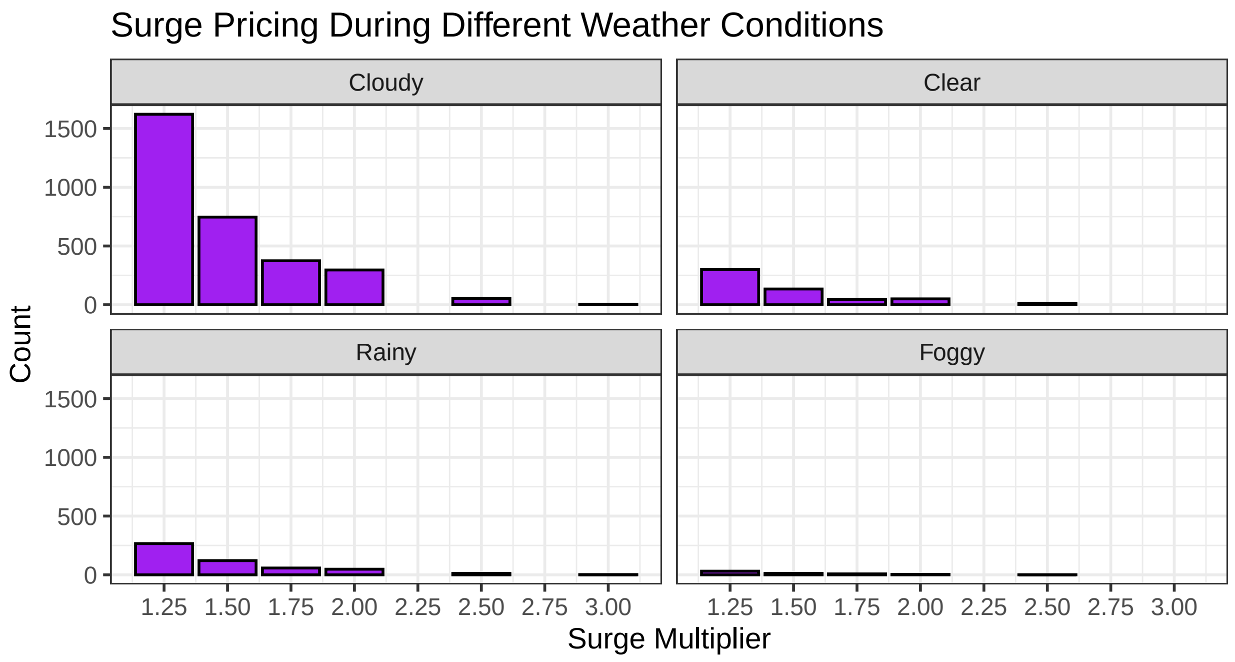
Lyft fare price surging is affected by what is termed the surge multiplier. If the surge multiplier is equal to 1, then it means that there are enough Lyft cars in the area to meet user demand and, therefore, the fare remains unchanged. If the surge multiplier is greater than 1, then there are more users than Lyft cars in service. As a result, the standard fare is increased by the surge multiplier. For this particular dataset, which focuses on the city of Boston, the most frequent surge multiplier is 1.25.



When surge multipliers are plotted against each Lyft ride's starting location, we see that the amount of surge multipliers is highest in the Boston district of Back Bay, which can be as high as roughly 3.00. Haymarket Square, on the other hand, has the fewest price surges and never goes above 1.75.



Thw weather can also affect surge multipliers. The number of rides that are affected by surge pricing are highest when weather conditions are cloudy. Furthermore, when the weather is either cloudy or rainy, surge multipliers can reach up to 3.00. It is interesting to note that clear weather conditions have nearly the same pattern of surge pricing as rainy conditions, indicating that there are other variables that impact surge multipliers.

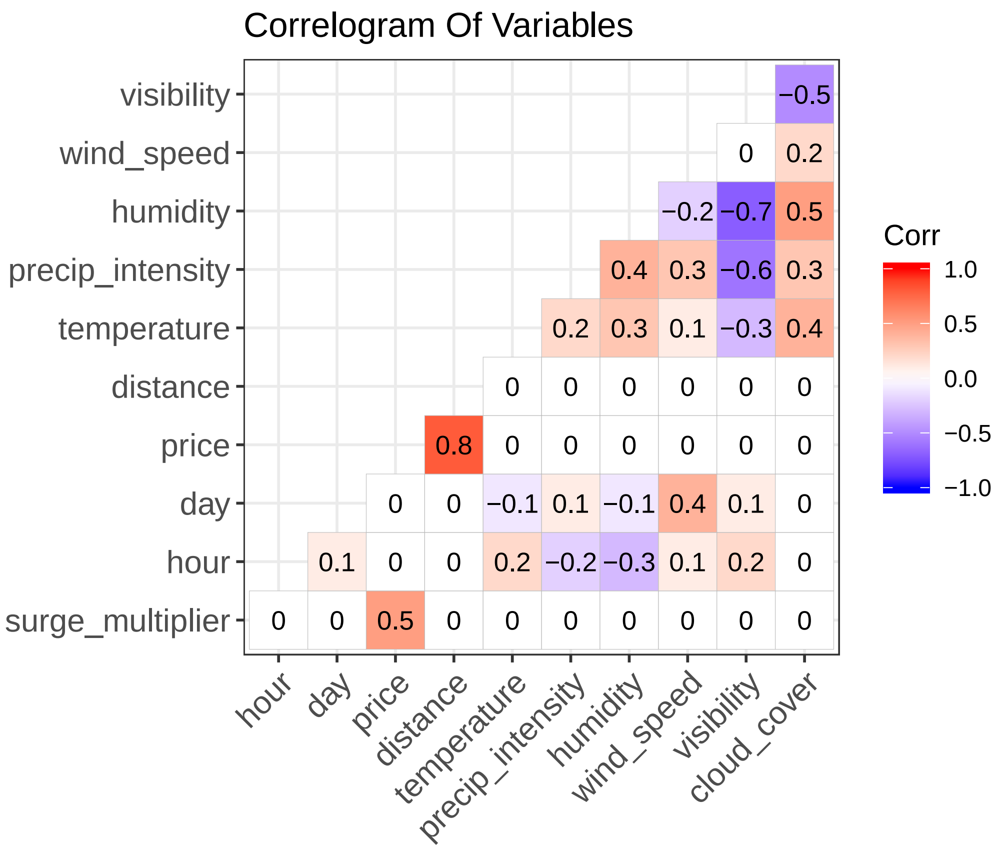


We, then, created a correlogram for the surge multiplier and other numeric variables of interest in the dataset. This allowed us to visualize any possible correlations between the variables through the use of color and a numeric scale.
If variables are positively correlated, then the intersecting box between variables in the graph would be a shade of red. If the variables are negatively correlated, then the intersecting box in the graph would be shade of purple. The darkness of the color represents the strength of the relationship: the darker the color, the stronger the relationship.

If there is a minus sign in front of the correlation coefficient, then there is a negative linear relationship (when one variable increases, the other decreases). If there is no sign, then there is a positive linear relationship (when one variable increases, the other increases).

The strength of the correlation coefficient is numerically defined as follows ("How To Interpret A Correlation Coefficient R"):

- 1.0: A perfect correlation.
- Between 0.7 to 1.0: A strong correlation.
- Between 0.5 to 0.7: A moderate correlation.
- Between 0.3 to 0.5: A weak correlation.
- 0: No correlation.



	hour	day	price	distance	temperature	precip_intensity	humidity	wind_speed	visibility	cloud_cover
visibility										−0.5
wind_speed									0	0.2
humidity								−0.2	−0.7	0.5
precip_intensity							0.4	0.3	−0.6	0.3
temperature						0.2	0.3	0.1	−0.3	0.4
distance					0	0	0	0	0	0
price				0.8	0	0	0	0	0	0
day			0	0	−0.1	0.1	−0.1	0.4	0.1	0
hour		0.1	0	0	0.2	−0.2	−0.3	0.1	0.2	0
surge_multiplier	0	0	0.5	0	0	0	0	0	0	0

The correlogram indicates that there is a moderate relationship between the surge multiplier and price, which have a correlation coefficient of 0.5. There is a strong correlation between price and distance, which have a correlation coefficient of 0.8. It is also clear that Lyft fares are not affected by the hour in the day, the day within the week, or other specific weather variables.

We built a linear regression model (see Supplementary Materials section for code) by splitting the data set into a training set (90% of the data) and a test set (10% of the data). The price of the Lyft ride was forecasted by using the surge multiplier, distance, source location, and weather summary. This model generated a multiple R-squared value of 0.845, which means it explains 84.5% of the variation in Lyft prices. In addition, the correlation accuracy of this model is 0.92, which means that the predicted prices of Lyft rides are very similar to the actual prices since its value is close to 1. This can be seen by examining the first few lines of data generated by this model in the table below.

Table 1: Comparing the Actual Prices to the Predicted Prices

Actual	Predicted
7.0	7.5
7.0	6.7
7.0	7.1
22.5	23.5
7.0	7.6
9.0	10.0

Conclusion

Price surging is a very common practice of rideshare companies. It occurs when the demand for rides exceeds the number of cars available for service. By analyzing a dataset of Lyft rides for the city of Boston, several variables were found to impact price surging, such as weather conditions and a ride's starting location. Using these findings plus the correlation of price and distance, we were able to build a linear regression model that forecasts Lyft prices with an accuracy of 0.92. Although this accuracy is very acceptable since it is close to a value of 1, this model could still potentially be improved by collecting more data throughout several locations and by including important events in each location (sporting events, concerts, etc.) which may affect the need for Lyft rides.

References

"Lyft." *Lyft*, Lyft, Inc., 2019, lyft.com.

Rumsey, Deborah. "How to Interpret a Correlation Coefficient r." *Dummies: A Wiley Brand*, John Wiley & Sons, Inc., 2019, dummies.com/education/math/statistics/how-to-interpret-a-correlation-coefficient-r.

Supplementary Material

This data science project was completed with the use of the R code below.

```
## load libraries
library(tidyverse)
library(ggcorrplot)
library(knitr)

## assign the link for the lyft zip folder
lyft_url <- "https://github.com/jessicapadilla/lyft_fares/blob/master/
lyft.csv.zip?raw=true"

## create a temporary file so the data can be downloaded into it
lyft_temp <- tempfile()

## download the zip folder containing the data
download.file(lyft_url, lyft_temp, mode = "wb")

## unzip the zip folder
unzip(lyft_temp, "lyft.csv")

## read the csv file within the zip folder
lyft <- read.csv("lyft.csv", sep = ",", header = TRUE)

## check the structure of the data
str(lyft)

## check if the visibility and visibility.1 columns are the same
identical(lyft$visibility, lyft$visibility.1)

## convert the data into a tibble
## remove unnecessary columns
## rename the remaining columns
lyft <- as_tibble(lyft) %>% select(-c(id, visibility.1)) %>%
  rename(company = cab_type,
         car_type = name,
         apparent_temperature = apparentTemperature,
         weather_summary = short_summary,
         precip_intensity = precipIntensity,
         precip_probability = precipProbability,
         wind_speed = windSpeed,
         temp_high = temperatureHigh,
         temp_low = temperatureLow,
         dew_point = dewPoint,
         wind_bearing = windBearing,
         cloud_cover = cloudCover,
         uv_index = uvIndex,
         moonphase = moonPhase)

## convert the integer columns to numeric
lyft[, c(1:3, 22, 24)] <- sapply(lyft[, c(1:3, 22, 24)], as.numeric)

## filter the data to only include the basic lyft services
lyft <- lyft %>% filter(car_type == "Lyft")
```

```r
## condense the categories in the weather summary column
lyft <- lyft %>% mutate(weather_summary = case_when(
  weather_summary == " Clear " ~ "Clear",
  weather_summary %in% c(" Overcast ", " Mostly Cloudy ", " Partly Cloudy ",
                       " Possible Drizzle ") ~ "Cloudy",
  weather_summary %in% c(" Light Rain ", " Rain ", " Drizzle ") ~ "Rainy",
  weather_summary == " Foggy " ~ "Foggy"))

## reorder the types of weather conditions
lyft$weather_summary <- factor(lyft$weather_summary,
                              levels = c("Cloudy", "Clear", "Rainy", "Foggy"))

## create a graph to show how frequent price surging occurs
lyft %>% filter(surge_multiplier > 1) %>%
  ggplot(aes(surge_multiplier)) +
  geom_histogram(binwidth = 0.25, col = "black", fill = "purple") +
  scale_x_continuous(breaks = seq(1.25, 3.00, 0.25)) +
  coord_cartesian(ylim = c(0, 2250)) +
  xlab("Surge Multiplier") + ylab("Count") +
  ggtitle("Frequency of Price Surging") +
  theme_bw()

## create a graph to show how surge multiplier varies by source location
## exclude surge_multiplier of 1 since it doesn't cause fare to increase
lyft %>% filter(surge_multiplier > 1) %>%
  mutate(source = reorder(source, -surge_multiplier)) %>%
  ggplot(aes(source, surge_multiplier, col = source)) +
  geom_jitter(alpha = 0.2, size = 2, width = 0.2) +
  scale_x_discrete(labels = function(x) str_wrap(x, width = 10)) +
  scale_y_continuous(breaks = seq(1.25, 3.00, 0.25)) +
  xlab("Starting Location") + ylab("Surge Multiplier") +
  ggtitle("Surge Multiplier Based On Starting Location") +
  theme_bw() +
  theme(legend.position = "none",
        axis.text.x = element_text(angle = 90, hjust = 1))

## create a graph to show how the surge multiplier changes with weather
lyft %>% filter(surge_multiplier > 1) %>%
  group_by(surge_multiplier) %>% count(weather_summary) %>%
  ggplot(aes(surge_multiplier, n, col = purple)) +
  geom_bar(stat = "identity", fill = "purple", color = "black") +
  facet_wrap(. ~ weather_summary) +
  scale_x_continuous(breaks = seq(1.25, 3.00, 0.25)) +
  xlab("Surge Multiplier") + ylab("Count") +
  ggtitle("Surge Pricing During Different Weather Conditions") +
  theme_bw()

## revise the data frame to only include numeric values of interest
lyft_num <- lyft %>% select(c(surge_multiplier, hour, day,
                              price, distance,
                              temperature, precip_intensity,
                              humidity, wind_speed,
                              visibility, cloud_cover))
```

```r
## round the data to one decimal place
corr <- round(cor(lyft_num), 1)

## create a correlogram
ggcorrplot(corr, type = "lower",
           lab = TRUE, lab_size = 3.5,
           title = "Correlogram Of Variables",
           ggtheme = theme_bw())

## set the seed
set.seed(123, sample.kind = "Rounding")

## there are 51235 observations
## split the data into training and testing sets
## set 90% of the data for the training set (46111 observations)
## set the remaining 10% of the data for the test set (5124 observations)
## randomize the data by first creating a vector of random integers
train_sample <- sample(51235, 46111)

## use the vector of random integers to randomly select observations from the data
## create the training set
lyft_train <- lyft[train_sample, ]

## create the test set
lyft_test <- lyft[-train_sample, ]

## create a linear regression model using the training set
lyft_model <- lm(price ~ surge_multiplier + distance +
                   source + weather_summary, data = lyft_train)

## check the summary to evaluate the model's performance
summary(lyft_model)

## use the linear regression model on the test set
lyft_pred <- predict(lyft_model, lyft_test)

## create a data frame to compare the predicted and the actual values
actual_predicted <- data.frame(cbind(Actual = lyft_test$price, Predicted = lyft_pred))

## check the correlation accuracy between the predicted and the actual values
correlation_accuracy <- cor(actual_predicted)
correlation_accuracy

## check the first few rows of the predicted and the actual values
kable(head(actual_predicted),
      caption = "Comparing the Actual Prices to the Predicted Prices", digits = 1)
```