# Fare Sight

2023-12-08

# INTRODUCTION

## DATA DESCRIPTION

The project is about the world's largest taxi company, Uber Inc. In this project, we are looking to predict the fare for their future transaction cases. Uber delivers services to millions of customers daily. Now, it becomes really important to manage their data properly to come up with new business ideas and achieve the best results. Eventually, it becomes crucial to estimate the fare prices accurately. This dataset mostly revolves around the New York demographic, but there are some other trips as well. With the help of the variables in our dataset, we want to understand what factors are contributing to the mean change in 'Fare'. We acquired this dataset from Kaggle.

- This dataset consists of 'Fare,' which is our response (the quantity we want to predict), pickup time, passenger count, **pickup location**, and **dropoff location**.
- The **pickup location** in our dataset is represented by two columns: pickup_latitude and pickup_longitude.
- The **dropoff location** in our dataset is represented by two columns: dropoff_latitude and dropoff_longitude.

## DATA DICTIONARY

- ID - a unique identifier for each trip
- fare - the cost of each trip in usd
- pickup_datetime - date and time when the meter was engaged
- passenger_count - the number of passengers in the vehicle (driver entered value)
- pickup_longitude - the longitude where the meter was engaged
- pickup_latitude - the latitude where the meter was engaged
- dropoff_longitude - the longitude where the meter was disengaged
- dropoff_latitude - the latitude where the meter was disengaged

## GOALS

Our primary goals are to:

1. Build a good prediction model to forecast the fare of an Uber ride.

2. Determine which attributes are significant in predicting the Uber fare.

3. Address the question: "Does pickup_datetime have a significant linear relationship with fare?"

The third question is particularly interesting because a key attribute missing from our dataset is the 'Number of drivers' present at the time of 'pick up.' A reasonable assumption would be that if the demand for rides is higher at a particular time than the supply of drivers in that locality, the fare should increase. The number

of drivers could vary throughout the day; for example, at 2 am, while the number of drivers is reduced, potential riders are also less, resulting in lower demand and, consequently, lower fare for Uber. On the other hand, during peak hours, such as 4-5 pm, the demand for Uber might increase. Although the number of available drivers is higher, the high demand could lead to higher fares. Therefore, given the lack of data for the number of available drivers at pickup time, we want to investigate whether pickup time plays a significant role in the increase or decrease of fare.

# REGRESSION ANALYSIS

## DATA PREPARATION

- First of all, our dataset included one redundant attribute, 'key,' which is the same as the pickup_datetime, and an attribute 'X,' which serves as the actual key. To address this, we changed the column name of 'X' to 'id,' and we dropped the 'key' column.

| | X <int> | key <chr> | fare_amo... <dbl> | pickup_datetime <chr> |
|---|---|---|---|---|
| 1. | 24238... | 2015-05-07 19:52:06.0000003 | 7.5 | 2015-05-07 19:52:06 UTC |
| 2. | 27835... | 2009-07-17 20:04:56.0000002 | 7.7 | 2009-07-17 20:04:56 UTC |
| 3. | 44984... | 2009-08-24 21:45:00.00000061 | 12.9 | 2009-08-24 21:45:00 UTC |
| 4. | 25894... | 2009-06-26 08:22:21.0000001 | 5.3 | 2009-06-26 08:22:21 UTC |
| 5. | 17610... | 2014-08-28 17:47:00.000000188 | 16.0 | 2014-08-28 17:47:00 UTC |
| 6. | 44470... | 2011-02-12 02:27:09.0000006 | 4.9 | 2011-02-12 02:27:09 UTC |
| 7. | 48725... | 2014-10-12 07:04:00.0000002 | 24.5 | 2014-10-12 07:04:00 UTC |
| 8. | 44195... | 2012-12-11 13:52:00.00000029 | 2.5 | 2012-12-11 13:52:00 UTC |
| 9. | 15822... | 2012-02-17 09:32:00.00000043 | 9.7 | 2012-02-17 09:32:00 UTC |
| 1. | 50611... | 2012-03-29 19:06:00.000000273 | 12.5 | 2012-03-29 19:06:00 UTC |

1–10 of 10 rows | 1–5 of 9 columns

- Another problem is the nature of pickup_datetime; it is a time series data. To address this, we split the pickup_datetime into pickup_day, pickup_month, pickup_year, pickup_hour, pickup_minute, and pickup_second. This approach solves two of our problems by providing numerical data instead of time series and increasing the number of columns for more detailed analysis.
- After this, we replaced all the missing values (NaN), which were only two in number, with mean values.
- Then, we added a new column for distances. We calculated the distances from pickup_latitude and pickup_longitude to drop_latitude and drop_longitude using the Haversine Distance formula.

The *haversine formula* allows the haversine of $\theta$ (that is, $\mathrm{hav}(\theta)$) to be computed directly from the latitude (represented by $\phi$) and longitude (represented by $\lambda$) of the two points:

$$\mathrm{hav}(\theta) = \mathrm{hav}(\varphi_2 - \varphi_1) + \cos(\varphi_1)\cos(\varphi_2)\,\mathrm{hav}(\lambda_2 - \lambda_1)$$

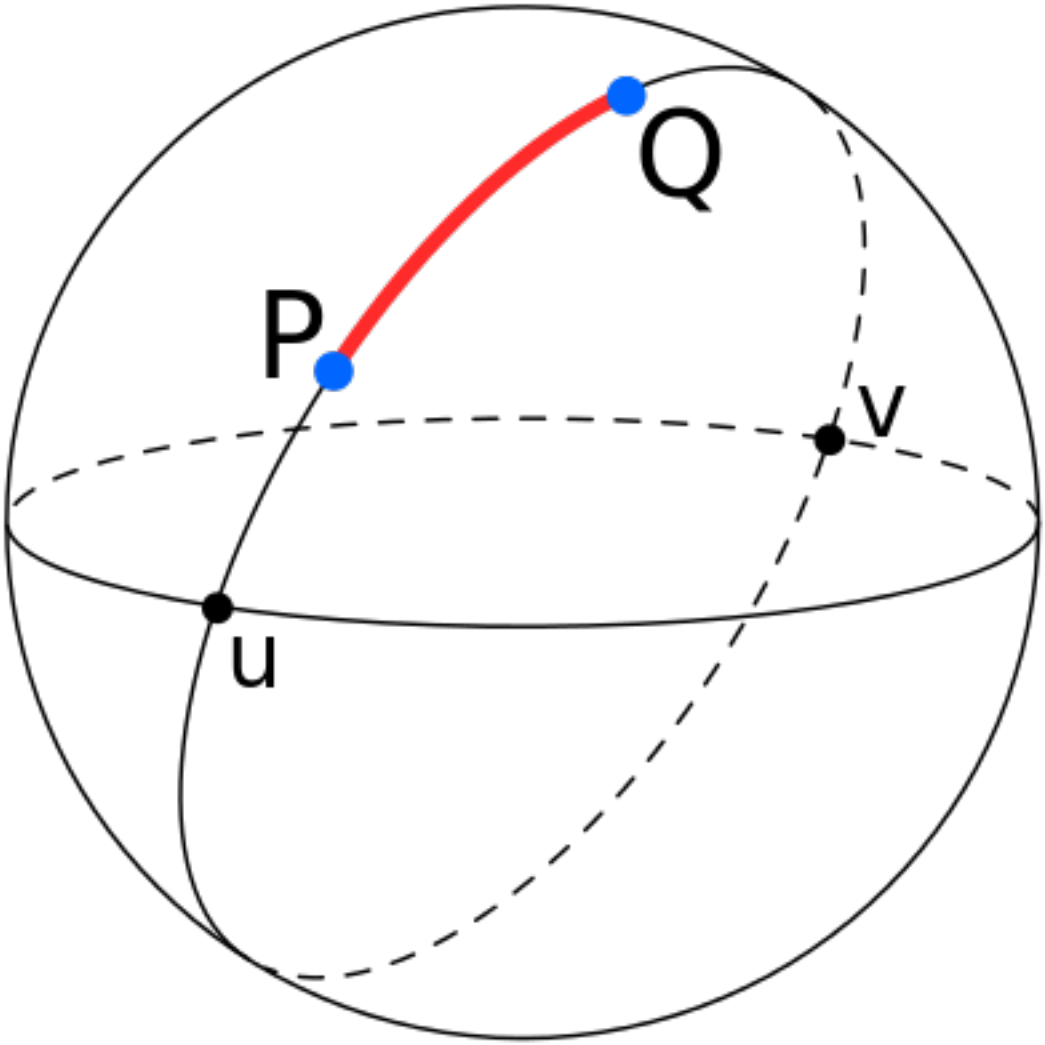
where

- $\phi_1$, $\phi_2$ are the latitude of point 1 and latitude of point 2,

- $\lambda_1$, $\lambda_2$ are the longitude of point 1 and longitude of point

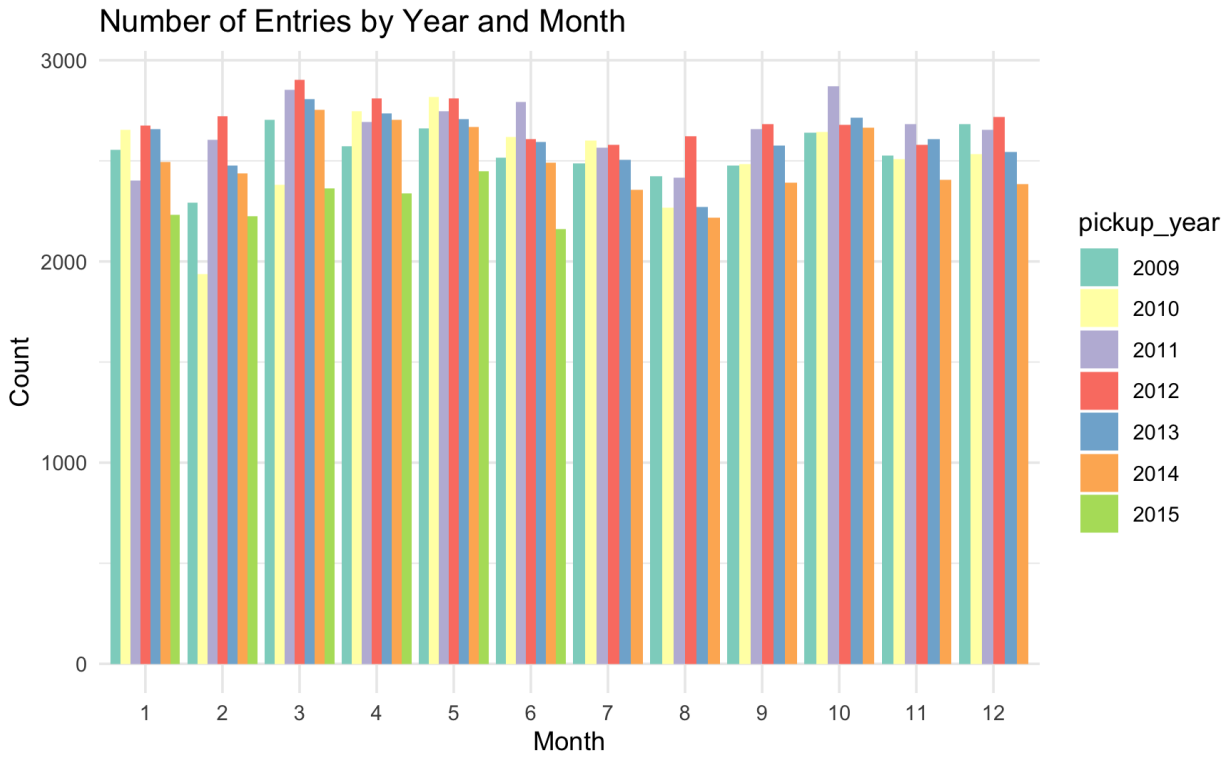
Finally, the haversine function hav( $\theta$), applied above to both the central angle $\theta$ and the differences in latitude and longitude, is

$$\text{hav}(\theta) = \sin^2\left(\frac{\theta}{2}\right) = \frac{1-\cos(\theta)}{2}$$




- At last we fit out model and we observe a very low $R^2$ value (0.01667). which is a very big problem even our initial model shouldn't perform this bad. So we checked for high influential points and high leverage points and we found 5727 highly influential points and 4148 high leverage points. We also assume the low $R^2$ could be due the time sensitivity of the data. We have 200,000 data points dating from 2009-2015 and the fare trend could change on hourly basis. So fitting a linear model for this big time series data might not be wise, and hence we norrow our data set to january 2013. First we plotted how many uber rides in every year and each month of those year, then we calculated the high influential

points in each year. The reason behind selecting 2013 was its lower number of high influential points while compared to other years.




| | year <chr> | influential_points_count <int> |
|---|---|---|
| 3 | 2011 | 1052 |
| 2 | 2010 | 1031 |
| 4 | 2012 | 993 |
| 1 | 2009 | 979 |
| 6 | 2014 | 865 |
| 7 | 2015 | 845 |
| 5 | 2013 | 799 |

We selected only one month of data because we assume that the trend might change month after month. This brought our data size down to 2658, but our $R^2$ only improved to 0.005834 which was still very bad so we concluded that we need to take care of high influential points.

## FINAL MODEL

$\hat{fare} = 6.529 - 9.103e{+}01\hat{pickup\_longitude} + 8.506e{+}01\hat{pickup\_latitude} + 9.596e{+}01\hat{drop\_longitude} - 7.618e{+}01\hat{drop\_latitu}$

The above is our final model equation, a result of feature selection and building a WLS model. We performed robust regression, and the Huber's model emerged as the best model. Using the confidence intervals from the model, we identified important and significant features. To validate our findings, we employed feature selection methods, comparing their AIC, BIC, and adjusted R2 scores. Through this process, we selected and finalized the significant features that we used in our weighted least squares model (WLS). The WLS

model yielded the best results for us. Despite the influence of time in the response, we were able to explain 82% of the data using our WLS model.

Starting with the LAD and Huber's model, we found that pickup_longitude, pickup_latitude, drop_longitude, drop_latitude, and distances are significant, as their confidence intervals did not include zero.

```
Intervals:
pickup_longitude [-1.089913e+02 -1.047619e+02 ]
pickup_latitude [8.317774e+01   8.656390e+01]
drop_longitude [1.032701e+02    1.088703e+02]
drop_latitude [-8.999490e+01    -8.292569e+01]
distances [1.284252e-03 1.335820e-03]
```

In our LAD model, we observe p-values of 0.0000 for the mentioned variables. While "0.0000" doesn't provide a clear explanation on its own, the Huber's model somewhat justifies the significance.

To further confirm our findings on significant features, we employed Forward, Backward, Best Subset, and Step-Wise selection methods. We utilized AIC, BIC, LOO-CV, and adjusted R2 criterion scores for evaluation. Below are the results:

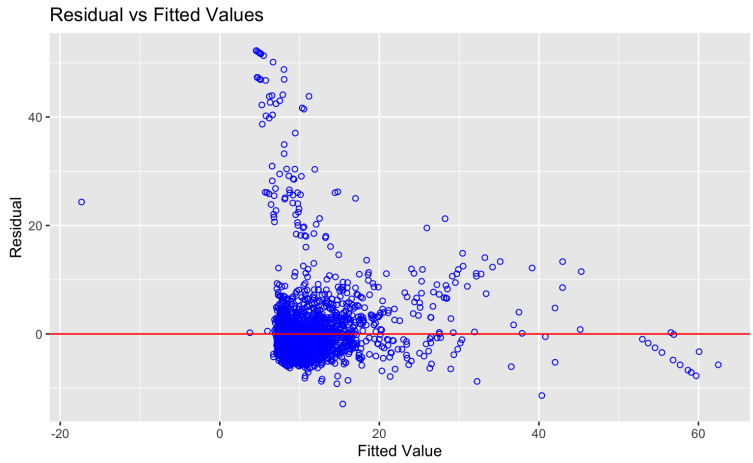
```
1. Backward AIC:
AIC = 10300.45 ; Adj_r2 = 0.3754496 ; LOOCV = 12.52515
Selected features = pickup_longitude, pickup_latitude, drop_longitude, drop_latitude, pickup_day,    dis

2. Backward BIC:
BIC = 10300.45 ; Adj_r2 = 0.3751913 ; LOOCV = 12.51689
Selected features = pickup_longitude, pickup_latitude, drop_longitude, drop_latitude, distances

3. Forward AIC and Forward BIC do not perform good. For Forward AIC we see all the predictors as signif

4. Best Subset(Exhaustive) selection using AIC:
Selected features = drop_longitude, drop_latitude, passenger_count, pickup_day, distances

5. Best Subset(Exhaustive) selection using BIC:
Selected features = drop_longitude, drop_latitude, passenger_count, pickup_day, distances
```

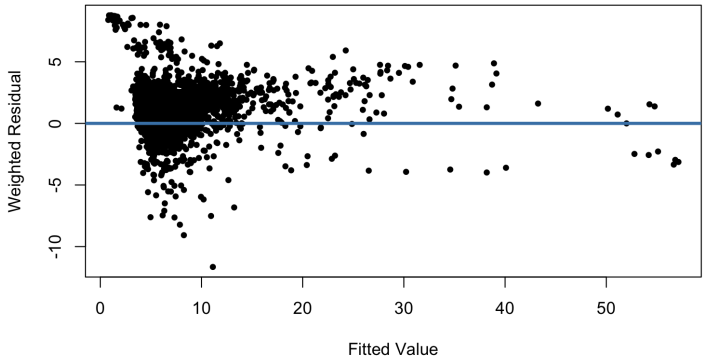
We should note that we have worked on feature selection with highly influential outliers removed. Therefore, we can now be confident that our features used in final model are highly significant.

Assumptions:

| | OLS(P-Value) | WLS(P-Value) |
|---|---|---|
| Shapiro-Wilk Test | 2.2e-16 | 2.2e-16 |
| BP-Test | 0.0004299 | 0.9993 |

It's reassuring to note that our constant variance assumption is met in the final WLS model, where the errors exhibit a consistent variance. Despite removing highly influential outlier points, it appears that the errors do not follow a normal distribution. However, the R2 score for the WLS model has increased. Below is a comparison of the fitted vs residual plot for the OLS model and the fitted vs weighted residual plot for the WLS model, illustrating the constant variance:



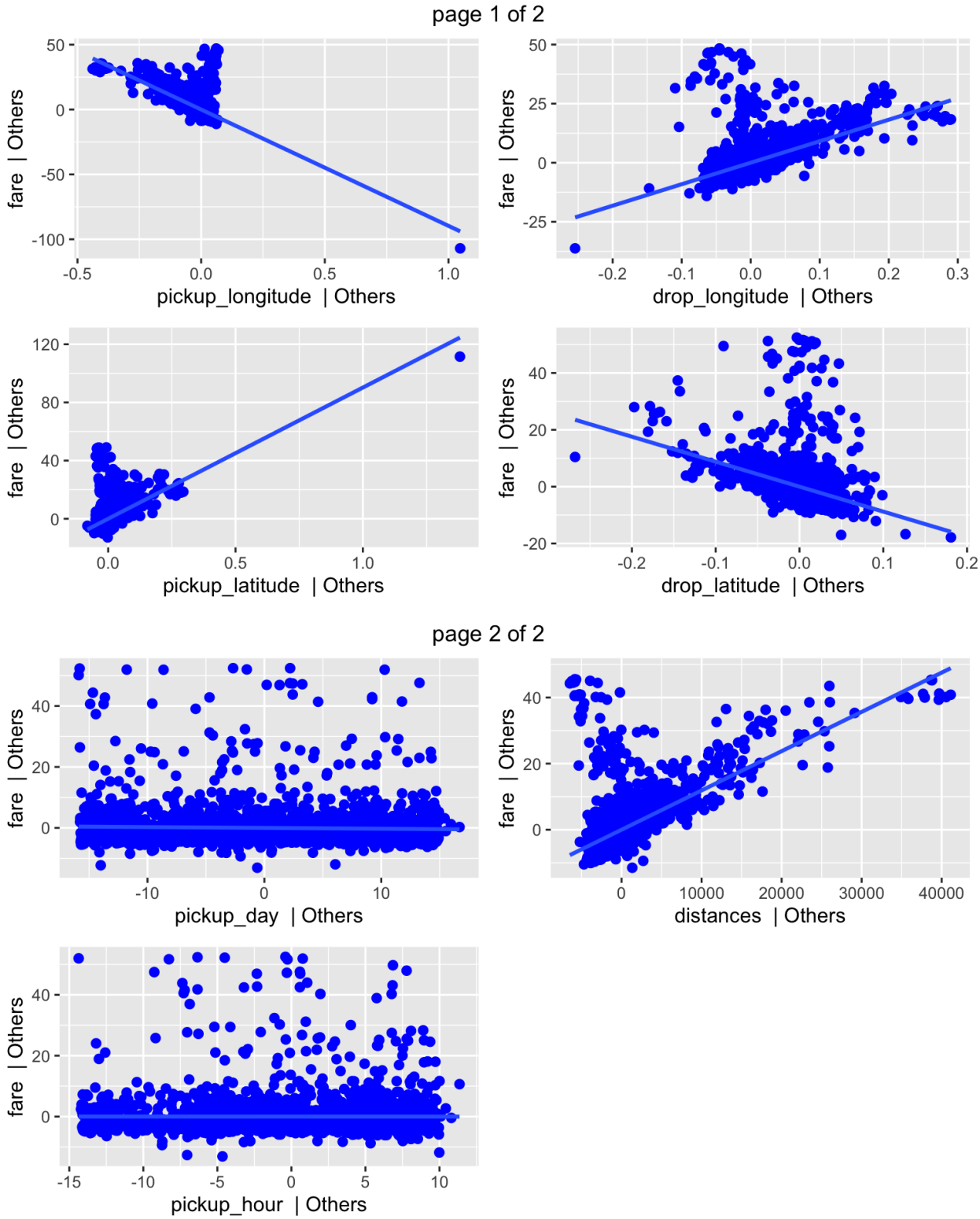

# DISCUSSION

With the Add Variable plots provided below, we can draw a couple of conclusions:

1. Answer to the Question about Pickup Datetime: The question of whether pickup_datetime has a significant linear relationship with the model is answered. The plot clearly indicates that pickup_datetime does not have a significant relationship with the fare. The points follow a horizontal trend, indicating a constant relationship between pickup_datetime and fare.

2. Significant Linear Relationships: On the other hand, pickup and drop latitudes and longitudes, as well as distances, show a significant linear relationship with the fare. This is particularly evident when examining the distances, drop_longitude, and drop_latitude.

It's valuable to observe and interpret these relationships to enhance our understanding of the factors influencing fare in the model.



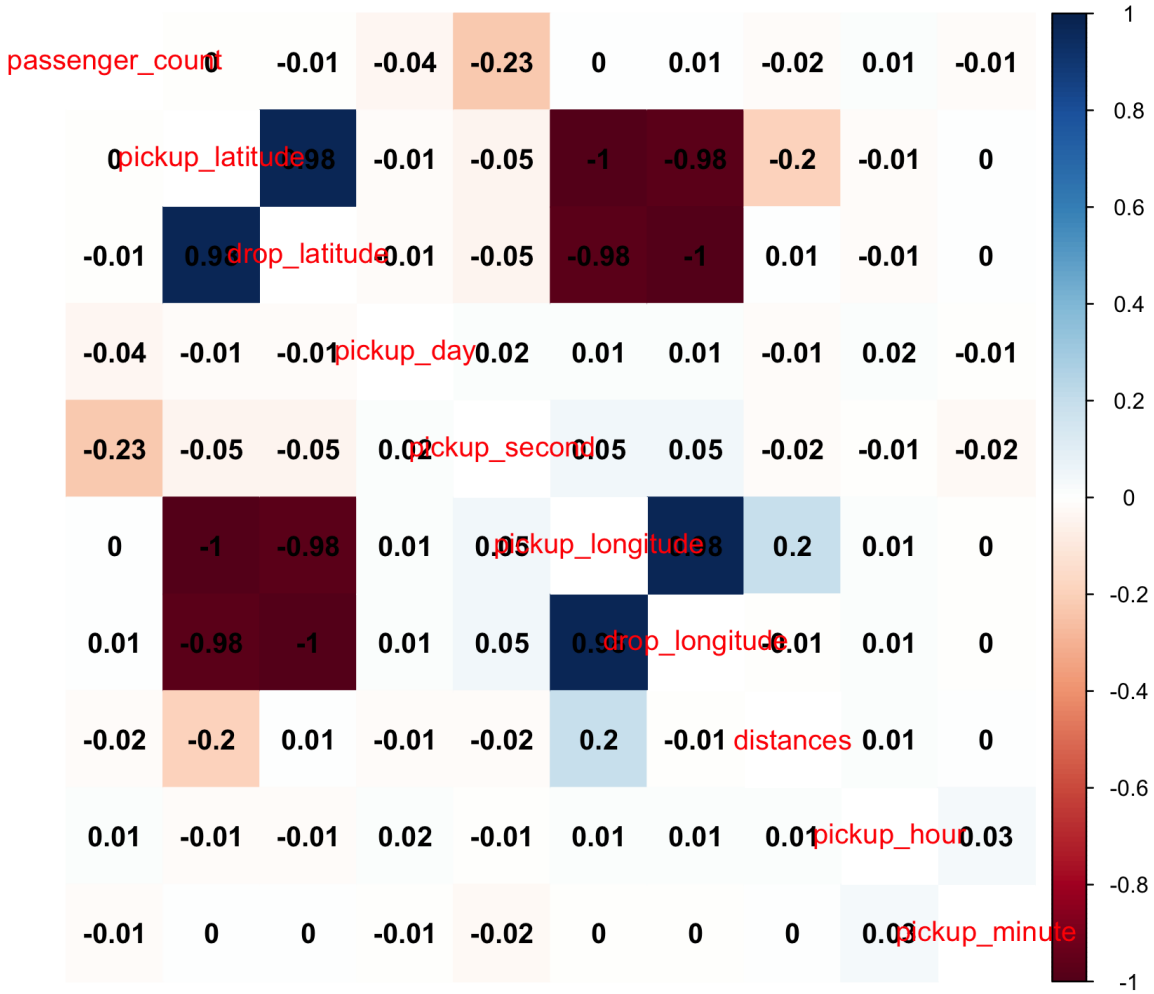

here we can also see some non linear relationship of distance with fare on the far left of their add variable plot but that could be due to highly influential points or outliers . we can also observe that it looks just like more spread out version of the pickup_latitude.

- We can also observe with the below mentioned summary of our WLS model that our $R^2$ has improved

to 0.8758 which is very good for our model, stating that 87.5% of the variability in the data set can be explained by our WLS model. Though because we did variable selection our significance tests don't mean much.

```
Weighted Residuals:
    Min      1Q  Median      3Q     Max
-9.1518 -0.3829  0.3197  0.9866  8.5712

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)       7.985e+00  3.601e-01  22.173  < 2e-16 ***
pickup_longitude -1.760e+01  1.981e+00  -8.886  < 2e-16 ***
pickup_latitude   1.403e+01  2.331e+00   6.021 2.00e-09 ***
drop_longitude    1.694e+01  2.009e+00   8.432  < 2e-16 ***
drop_latitude    -1.537e+01  2.097e+00  -7.327 3.19e-13 ***
distances         2.331e-03  2.130e-05 109.454  < 2e-16 ***
pickup_hour       5.111e-03  5.874e-03   0.870    0.384
pickup_day        4.317e-03  4.247e-03   1.017    0.309
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.338 on 2422 degrees of freedom
Multiple R-squared:  0.8758,    Adjusted R-squared:  0.8755
F-statistic:  2441 on 7 and 2422 DF,  p-value: < 2.2e-16
```

- Also a reasonable question to ask ourselves would be "Does pickup_day or pickup_hour have any collinearity with other predictors?" this is answered by the correlation matrix.




here we can see that pickup_hour or pickup_day or any of our time sensitive variables, none of them are correlated to any other predictor and hence are not collinear.

- We also need to check if there are any high influential points, and we found 73 high influential points in our WLS model

# LIMITATIONS

- It's crucial to acknowledge the collinearity in our model, which may obscure the clarity of inferences. While this condition may pose challenges for significance tests, it proves beneficial for predictions. The collinearity is reasonable, considering that distances are calculated with a linear combination of pickup and drop longitudes and latitudes.
- Due to variable selection, our significance tests are compromised, emphasizing the importance of understanding the trade-offs between model interpretability and prediction accuracy.
- Despite efforts to address the normality issue in our errors by removing highly influential points, it persisted. Consequently, we resorted to bootstrapping to derive confidence intervals.
- It's worth noting that the Haversine distances calculated for each ride serve as an estimate of the distance traveled by customers. The actual distance traveled is influenced by various factors such as

traffic, road conditions, and time of day. Hence, the calculated Haversine distances may differ from the actual distances, considering the existence of multiple possible paths between two points.

# CONCLUSION

- In conclusion, our exploration of this dataset revealed a significant presence of highly influential points. However, through effective feature engineering and variable selection, we successfully trained a robust predictive model that explains 87.5% of our dataset (January of 2013).
- The key insight from our modeling efforts is that pickup_longitude, pickup_latitude, drop_longitude, drop_latitude, and distance emerged as the only significant features. This aligns logically with the expectation that the fare of a ride should be strongly influenced by the distance the customer needs to travel. This finding underscores the importance of these spatial coordinates and distance in predicting Uber ride fares.

- Certainly, it's important to highlight the conclusion to your third question: whether there is a significant linear relationship between pickup_datetime and the fare of the ride. In the context of your subset of the dataset (January 2013), the answer is no. The analysis indicates that pickup_datetime does not have a significant linear relationship with the fare. This finding provides clarity on the role of pickup time in influencing ride fares within the specified dataset and time frame.

# Code Appendix

## Data Pre-Processing

```
# loading data set
uber = read.csv('uber.csv')
head(uber, n=5)
```

```
colnames(uber)
```

```
new_col_names = c("Id", "Drop_this", "fare", "pickup_datetime
", "pickup_longitude", "pickup_latitude", "drop_longitude", "drop_latitude", "passenger_count")
colnames(uber) = new_col_names
```

```
uber_data = uber[, !colnames(uber) %in% c("Drop_this")]
```

### Separating date and time from pickup_datetime

```
# Splitting pickup_datetime
library(lubridate)
```

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

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
uber_data$pickup_datetime = ymd_hms(uber_data$pickup_datetime)
uber_data$pickup_date = as.Date(uber_data$pickup_datetime)
uber_data$pickup_time = format(uber_data$pickup_datetime, "%H:%M:%S")
colnames(uber_data)

uber_data = uber_data[, !colnames(uber_data) %in% c("pickup_datetime", "pickup_datetime\n")]
colnames(uber_data)

sapply(uber_data, class)

uber_data$pickup_day = day(uber_data$pickup_date)
uber_data$pickup_month = month(uber_data$pickup_date)
uber_data$pickup_year = year(uber_data$pickup_date)
```

```
# dropping Pickup_datetime
uber_data = uber_data[, !colnames(uber_data) %in% c("pickup_date")]
colnames(uber_data)
```

```
# getting hour minute and seconds from date time
uber_data$pickup_time = as.POSIXct(uber_data$pickup_time, format = "%H:%M:%S", tz = "UTC")
uber_data$pickup_hour = hour(uber_data$pickup_time)
uber_data$pickup_minute = minute(uber_data$pickup_time)
uber_data$pickup_second = second(uber_data$pickup_time)
uber_data = uber_data[, !colnames(uber_data) %in% c("pickup_time")]
colnames(uber_data)
```

```
# dropping id
uber_data = uber_data[, !colnames(uber_data) %in% c("Id")]
sapply(uber_data, class)
```

```
nrow(uber_data)
ncol(uber_data)
colSums(is.na(uber_data))
```

```
library(tidyr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
uber_data = uber_data_filled <- uber_data %>%
  mutate_all(~replace_na(., mean(., na.rm = TRUE)))
colSums(is.na(uber_data))
nrow(uber_data)
```

**Calculating Haversine distances between pickup and dropoff locations.**

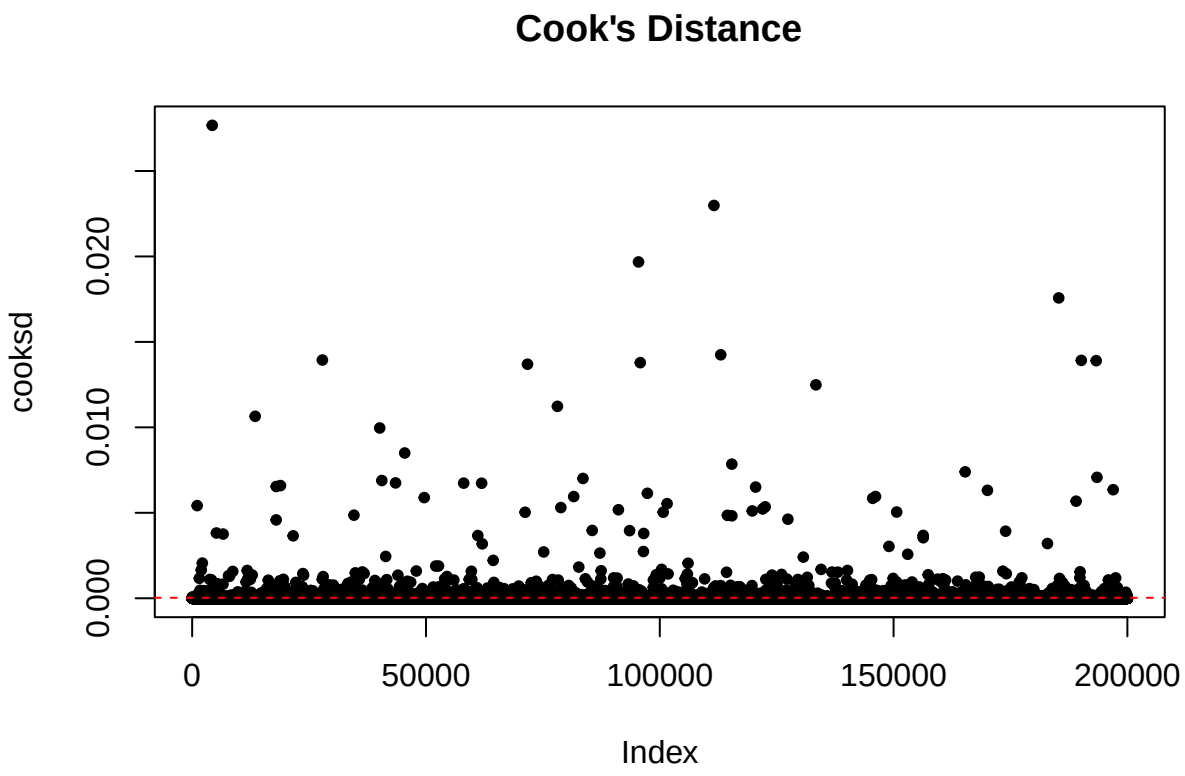
```
cleaned = subset(uber_data, pickup_longitude >= -180 & pickup_longitude <= 180 &
                           drop_longitude >= -180 & drop_longitude <= 180)
df_cleaned = subset(cleaned , pickup_latitude >= -90 & pickup_latitude <= 90 &
                           drop_latitude >= -90 & drop_latitude <= 90)
uber = df_cleaned
```

```
library(geosphere)
pickup_coordinates = cbind(uber$pickup_longitude, uber$pickup_latitude)
dropoff_coordinates = cbind(uber$drop_longitude, uber$drop_latitude)

distances = distHaversine(pickup_coordinates,dropoff_coordinates)
uber$distances = distances
```

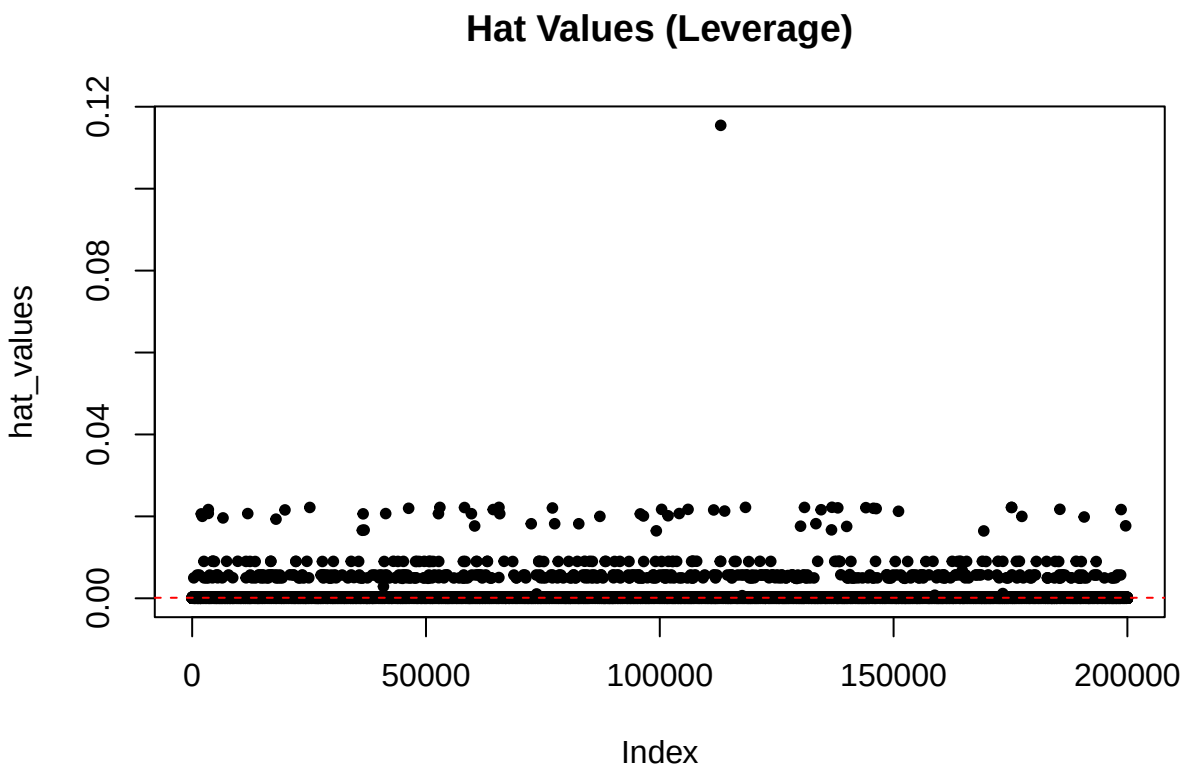
## Making model OLS for uber data, then sorting the year based on number of highly influential points.

```
model_ols = lm(fare~., uber)
summary(model_ols)
```

```
# Get Cook's distances
cooksd <- cooks.distance(model_ols)

# Identify influential points (considering a threshold, e.g., 4 times the mean)
influential_points <- which(cooksd > 4 * mean(cooksd))

# Print the influential points
cat("Influential points:", influential_points, "\n")

# Optionally, you can visualize Cook's distance
plot(cooksd, pch = 20, main = "Cook's Distance")
abline(h = 4 * mean(cooksd), col = "red", lty = 2)
```




```r
# Get hat values (leverage)
hat_values = hatvalues(model_ols)

# Identify high leverage points (considering a threshold, e.g., 2 times the mean)
high_leverage_points = which(hat_values > 2 * mean(hat_values))

# Print the high leverage points
cat("High leverage points:", high_leverage_points, "\n")

# Optionally, you can visualize hat values
plot(hat_values, pch = 20, main = "Hat Values (Leverage)")
abline(h = 2 * mean(hat_values), col = "red", lty = 2)
```




```
length(influential_points)
length(high_leverage_points)
```

```
uber$pickup_year = as.factor(uber$pickup_year)
uber$pickup_month = as.factor(uber$pickup_month)

# Group by year and month and count entries
result = uber %>%
  group_by(pickup_year, pickup_month) %>%
  summarise(count = n())

library(ggplot2)
# Plot the results
ggplot(result, aes(x = pickup_month, y = count, fill = pickup_year)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Number of Entries by Year and Month",
       x = "Month",
       y = "Count") +
  scale_fill_brewer(palette = "Set3") +
  theme_minimal()
```

```
library(kableExtra)
library(knitr)

result_table  =  kable(result, format = "html", caption = "Number of Entries by Year and Month") %>%
  kable_styling("striped", full_width = FALSE)

# Print the table
print(result_table)
```

**Identifying highly influential points in data for each month of every year.**

```
# Convert pickup_year and pickup_month to factors
uber$pickup_year = as.factor(uber$pickup_year)
uber$pickup_month = as.factor(uber$pickup_month)

# Function to identify influential points for each group
identify_influential_points <- function(data) {
  # Adjust the formula if needed
  cook_dist = cooks.distance(model_ols)
  threshold = 4 / nrow(data)  # You can adjust the threshold as needed
  influential_points = sum(cook_dist > threshold)
  return(influential_points)
}

# Apply the function to each year and month group
influential_points_count <- sapply(split(uber, list(uber$pickup_year, uber$pickup_month)), identify_infl

# Combine the results into a data frame
influential_points_df <- data.frame(year = rep(levels(uber$pickup_year), each = 12),
                                    influential_points_count = influential_points_count)

# Sum the counts for each year
total_influential_points <- aggregate(influential_points_count ~ year, data = influential_points_df, su

# Sort the result in decreasing order
sorted_result <- total_influential_points[order(total_influential_points$influential_points_count, decr

# Display the result
print(sorted_result)
```

**choosing 2013 1st month for it has lower influential points than other years and then making the ols model.**

```
uber$pickup_year = as.factor(uber$pickup_year)
data2013 = subset(uber, pickup_year == 2013)

data2013 = subset(data2013, pickup_month == 1)

data2013 = data2013[, !colnames(uber_data) %in% c("pickup_month", "pickup_year")]


row.names(data2013) = NULL
```

**OLS Model (Compare final with this OLS model)**

```
model_data2013_ols = lm(fare~., data2013)
summary(model_data2013_ols)
```

**Finding outliers, Highly influential points and highly influential outliers in data2013**

**outliers**

```
outlier_test_cutoff = function(model, alpha = 0.05) {
    n = length(resid(model))
    qt(alpha/(2 * n), df = df.residual(model) - 1, lower.tail = FALSE)
}

# vector of indices for observations deemed outliers.
cutoff = outlier_test_cutoff(model_data2013_ols, alpha = 0.05)

outlier_pts = which(abs(rstudent(model_data2013_ols)) > cutoff)
outlier_pts
```

**High influential**

```
high_infl_pts = which(cooks.distance(model_data2013_ols) > 4/length(resid(model_data2013_ols)))
high_infl_pts
length(high_infl_pts)
```

**Higly influential outliers**

```
highly_influ_outlier = intersect(outlier_pts, high_infl_pts)
highly_influ_outlier
```

## Data cleaned after removing highly influential outliers and highly influential points.

```
data_cleaned = data2013[-highly_influ_outlier, ]
data_cleaned = data_cleaned[-high_infl_pts,]
model_clean_ols = lm(fare~., data_cleaned)
summary(model_clean_ols)
```

## model ols with no highly influential outliers

```
model_ols_no_outliers = lm(fare ~ ., data = data_cleaned)

summary(model_ols_no_outliers)
```

## LAD regression with no higly influential outliers

```r
library(quantreg)
```

```
## Loading required package: SparseM
```

```
##
## Attaching package: 'SparseM'
```

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

```r
model_lad_ols_no_outliers = rq(fare ~ ., data = data_cleaned)
summary(model_lad_ols_no_outliers, alpha = 0.05)
```

```
## Warning in summary.rq(model_lad_ols_no_outliers, alpha = 0.05): 1 non-positive
## fis
```

### model hub no highly influential outliers

```r
library(MASS)
```

```
##
## Attaching package: 'MASS'
```

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

```r
library(car)
```

```
## Loading required package: carData
```

```
##
## Attaching package: 'car'
```

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

```r
model_hub_no_outliers = rlm(fare ~ ., maxit = 500, data = data_cleaned)

summary(model_hub_no_outliers)

set.seed(42)

Confint(Boot(model_hub_no_outliers, R = 2000, method = 'residual'))
```

```
## Loading required namespace: boot
```

```
## Warning in confint.boot(object, parm, level, type, ...): BCa method fails for
## this problem.  Using 'perc' instead
```

## Variable selection

### backward search using aic

```
mod_all_preds = lm(fare ~ ., data = data_cleaned)
mod_back_aic = step(mod_all_preds, direction = 'backward')
coef(mod_back_aic)
extractAIC(mod_back_aic)
```

### backward with bic

```
n = nrow(data_cleaned)
mod_back_bic = step(mod_all_preds, direction = 'backward', k = log(n))
coef(mod_back_bic)
extractAIC(mod_back_bic)
```

```
summary(mod_back_aic)$adj.r.squared
summary(mod_back_bic)$adj.r.squared
calc_loocv_rmse = function(model) {
  sqrt(mean((resid(model) / (1 - hatvalues(model))) ^ 2))
}
calc_loocv_rmse(mod_back_aic)
calc_loocv_rmse(mod_back_bic)
```

### forward using aic

```
mod_start = lm(fare ~ 1, data = data_cleaned)
mod_forwd_aic = step(
    mod_start,
    scope = fare ~ pickup_longitude + pickup_latitude + drop_longitude + drop_latitude + passenger_coun
    direction = 'forward')
coef(mod_forwd_aic)
```

### forward using bic

```
mod_forwd_bic = step(
    mod_start,
    scope = fare ~ pickup_longitude + pickup_latitude + drop_longitude + drop_latitude + passenger_coun
    direction = 'forward',
    k = log(n))
coef(mod_forwd_bic)
```

```
calc_loocv_rmse(mod_forwd_aic)
calc_loocv_rmse(mod_forwd_bic)
```

**stepwise with aic**

```
mod_stepwise_aic = step(
    mod_start,
    scope = fare ~ pickup_longitude + pickup_latitude + drop_longitude + drop_latitude + passenger_coun
    direction = 'both')
coef(mod_stepwise_aic)
```

**stepwise with bic**

```
mod_stepwise_bic = step(
    mod_start,
    scope = fare ~ pickup_longitude + pickup_latitude + drop_longitude + drop_latitude + passenger_coun
    direction = 'both',
    k = log(n))
coef(mod_stepwise_bic)
```

**best subset selection using aic**

```
library(leaps)

mod_exhaustive = summary(regsubsets(fare ~ ., data = data_cleaned, nvmax = 8))
best_r2_ind = which.max(mod_exhaustive$adjr2)
mod_exhaustive$which[best_r2_ind,]
```

```
model_exhaustive_adjr2 = lm(fare~ pickup_longitude+ pickup_latitude+ drop_longitude+ drop_latitude+ pic
summary(model_exhaustive_adjr2)
```

```
p = ncol(mod_exhaustive$which)

mod_aic = n * log(mod_exhaustive$rss / n) + 2 * (2:p)
```

```
## Warning in n * log(mod_exhaustive$rss/n) + 2 * (2:p): longer object length is
## not a multiple of shorter object length
```

```
best_aic_ind = which.min(mod_aic)

mod_exhaustive$which[best_aic_ind, ]
```

```
mod_exhaust_aic = lm(fare ~ drop_longitude+drop_latitude+passenger_count+pickup_day, data = data_cleane
summary(mod_exhaust_aic)
```

**best subset bic**

```
n = nrow(data_cleaned)
p = ncol(data_cleaned)

mod_bic = n * log(mod_exhaustive$rss / n) + log(n) * (2:p)
```

```
## Warning in n * log(mod_exhaustive$rss/n) + log(n) * (2:p): longer object length
## is not a multiple of shorter object length
```

```
best_bic_ind = which.min(mod_bic)
mod_exhaustive$which[best_bic_ind, ]
```

```
mod_exhaust_bic = lm(fare ~ pickup_longitude+pickup_latitude+pickup_latitude+drop_latitude, data_cleane
summary(mod_exhaust_bic)
```

## Model Diagnostics

**shapiro and bptest not good with our ols model**

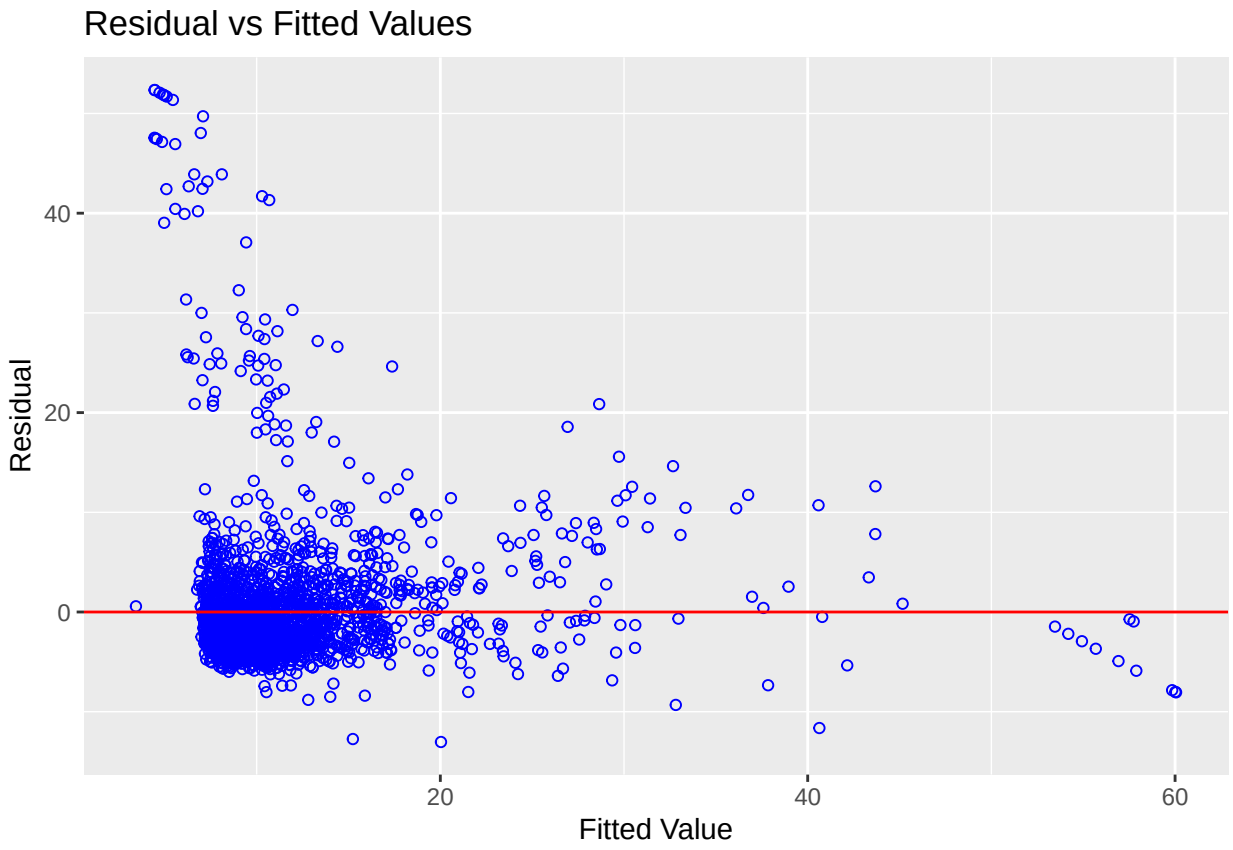
```
library(olsrr)
```

```
##
## Attaching package: 'olsrr'

## The following object is masked from 'package:MASS':
##
##     cement

## The following object is masked from 'package:datasets':
##
##     rivers
```

```
library(lmtest)
```

```
## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
ols_plot_resid_fit(model_clean_ols)
```

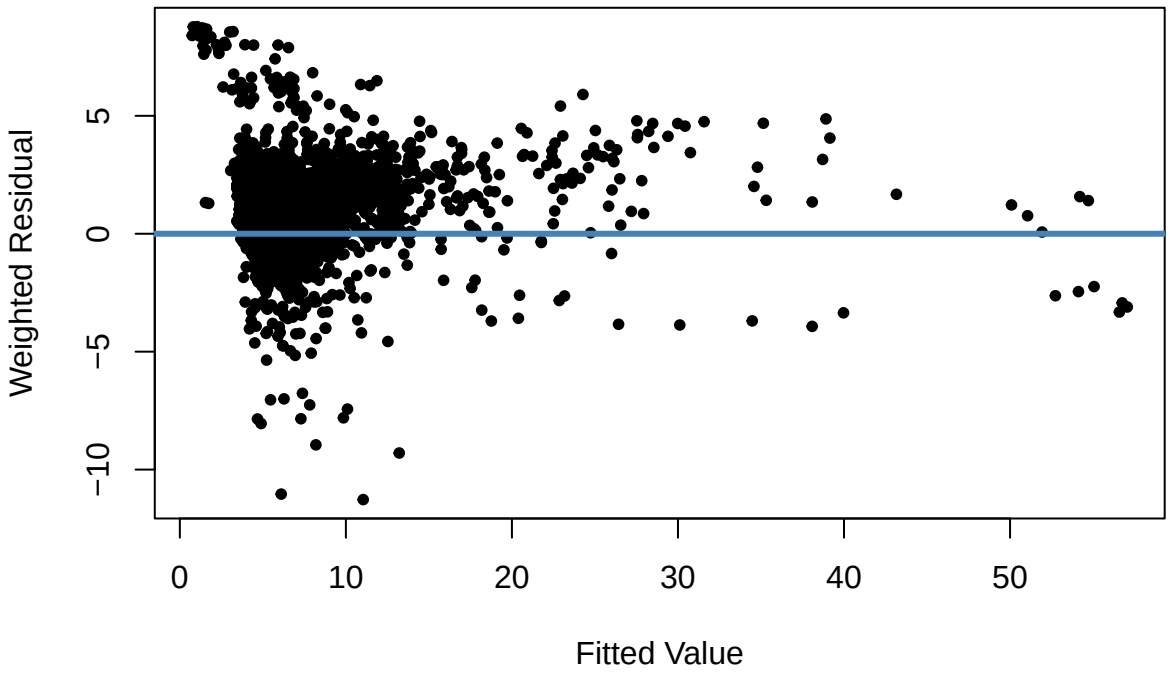



```
shapiro.test(resid(model_clean_ols))
bptest(model_clean_ols)
```

**Trying WLS on selected features and cleaned data**

```
model_wts = lm(abs(resid(model_clean_ols)) ~ . , data = data_cleaned)

# extract the coefficient estimates.
coef(model_wts)
```

```
# calculate the weights as 1 / (fitted values)
weights = 1/abs(fitted(model_wts))

# run WLS
model_wls = lm(fare ~pickup_longitude+ pickup_latitude+ drop_longitude+ drop_latitude + distances + pick
plot(fitted(model_wls), weighted.residuals(model_wls),
     pch = 20, xlab = 'Fitted Value', ylab = 'Weighted Residual')

abline(h=0, lwd=3, col='steelblue')
```



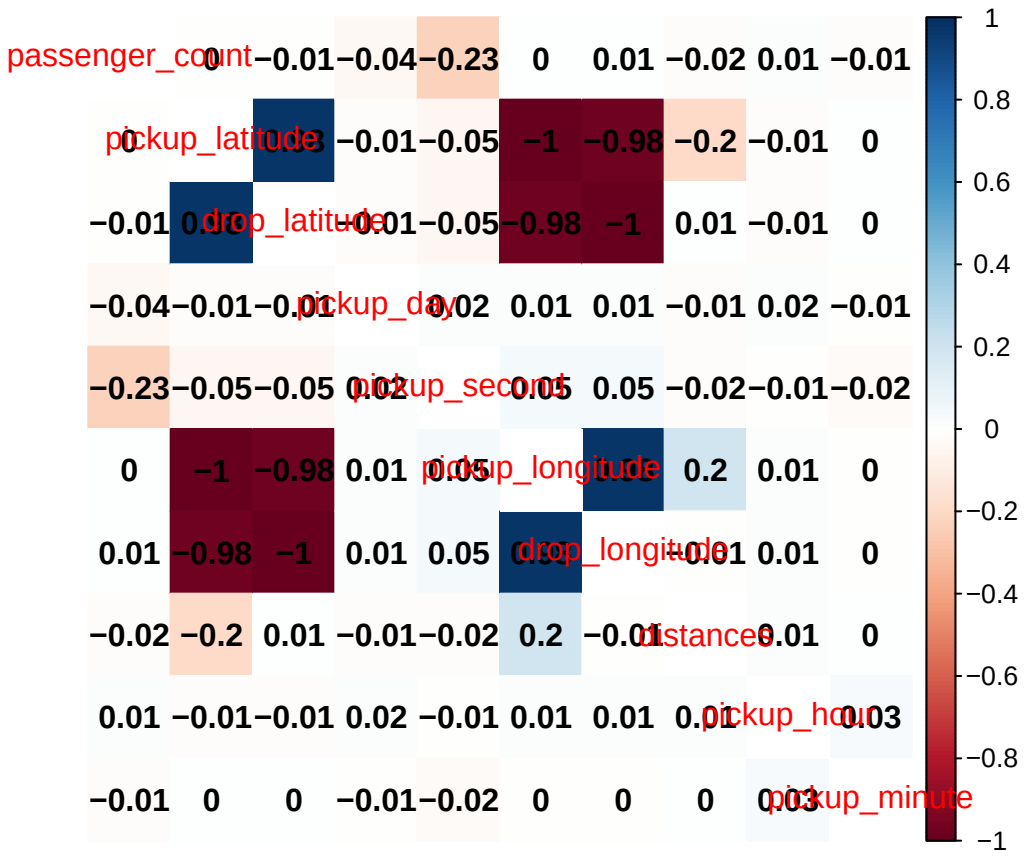

```
shapiro.test(resid(model_wls))
bptest(model_wls)
summary(model_wls)

Confint(Boot(model_wls, R = 2000, method = 'residual'))
```

```
## Warning in confint.boot(object, parm, level, type, ...): BCa method fails for
## this problem.  Using 'perc' instead
```

```
library(dplyr)
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
uber_predictors = dplyr::select(data_cleaned, -fare)
corrplot(cor(uber_predictors),
         method = 'color', order = 'hclust',  diag = FALSE,
         number.digits = 2, addCoef.col = 'black', tl.pos= 'd', cl.pos ='r')
```




```
high_infl_pts_wls = which(cooks.distance(model_wls) > 4/length(resid(model_wls)))
length(high_infl_pts_wls)
```

**Fitting a Weighted Least Squares model after remoing the highly influential points from model_wls**

```
model_wls_no_high_infl = lm(fare~pickup_longitude + pickup_latitude + drop_longitude + drop_latitude + d
summary(model_wls_no_high_infl)
shapiro.test(resid(model_wls_no_high_infl))
```