Appendix Part III: Advanced Linear Modeling

Introduction

In this section, we will use the processed data to build predictive models and in the meantime try to analyze the coefficients. Forward feature selection will also be used to reduce the multi-colinearity of the predictors.

This appendix will correspond to section 3.2 to 3.3 in the final report.

Simple Linear regression

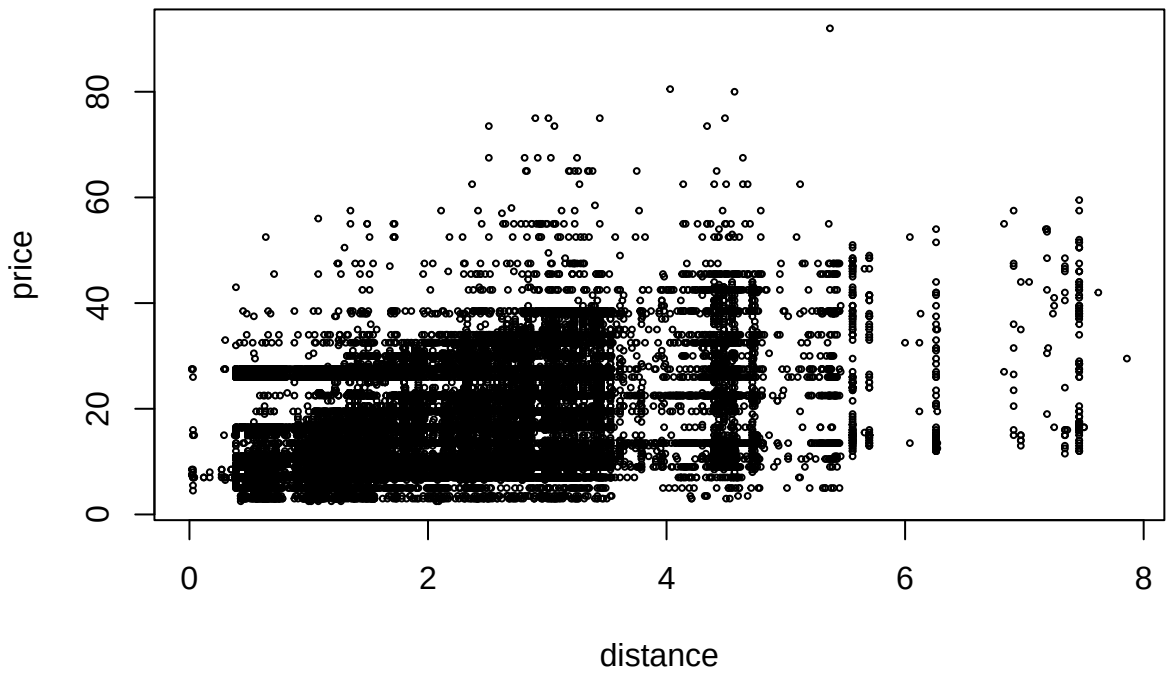
Visualization

In this section we will aim to build a very simple model to analyze the relationship between ride distance and ride price. As is stated earlier, we didn't use the entire dataset because it is too slow to process them all in R. So we randomly sampled 10% of the observations and use the downsampled version in the subsequent model building and analysis. We will start with some simple visualizations to explore the relationship between ride price and ride distance.

```
# helper function for calculating RMSE
RMSE = function(y, yhat) {
  return (mean((y - yhat) ^ 2))
}

# load data from the pre-processed csv
rideshare.train = read.csv("data/rideshare_train.csv")
rideshare.test = read.csv("data/rideshare_test.csv")

# scatter plot for ride price vs ride distance
plot(price ~ distance, data=rideshare.train, cex=0.4)
```



From the scatter plot we can see there is a positive relationship between ride distance and price, although we can see that the variability of the response at the same predictor value is fairly big, and we can expect that the R^2 for a simple linear model would not be very ideal.

Simple Linear model

The next step is to actually fit a simple linear regression model and check the assumptions for simple linear regression. We call it `lm.simple`.

```
lm.simple = lm(price ~ distance, data=rideshare.train)

# model summary
summary(lm.simple)
```

```
##
## Call:
## lm(formula = price ~ distance, data = rideshare.train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -20.615  -6.956  -1.708   4.966  66.608
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 10.44145    0.09484  110.09   <2e-16 ***
```

```
## distance     2.78417    0.03842   72.47   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.748 on 39998 degrees of freedom
## Multiple R-squared:  0.1161, Adjusted R-squared:  0.116
## F-statistic:  5251 on 1 and 39998 DF,  p-value: < 2.2e-16
```

```
# RMSE, which can also be calculated via residual standard error and dF
y.pred.train = predict(lm.simple)
y.pred.test = predict(lm.simple, newdata=rideshare.test)
lm.simple.train.rmse = RMSE(rideshare.train$price, y.pred.train)
lm.simple.test.rmse = RMSE(rideshare.test$price, y.pred.test)

# print out the RMSE result
data.frame(trainRMSE=lm.simple.train.rmse, testRMSE=lm.simple.test.rmse)
```

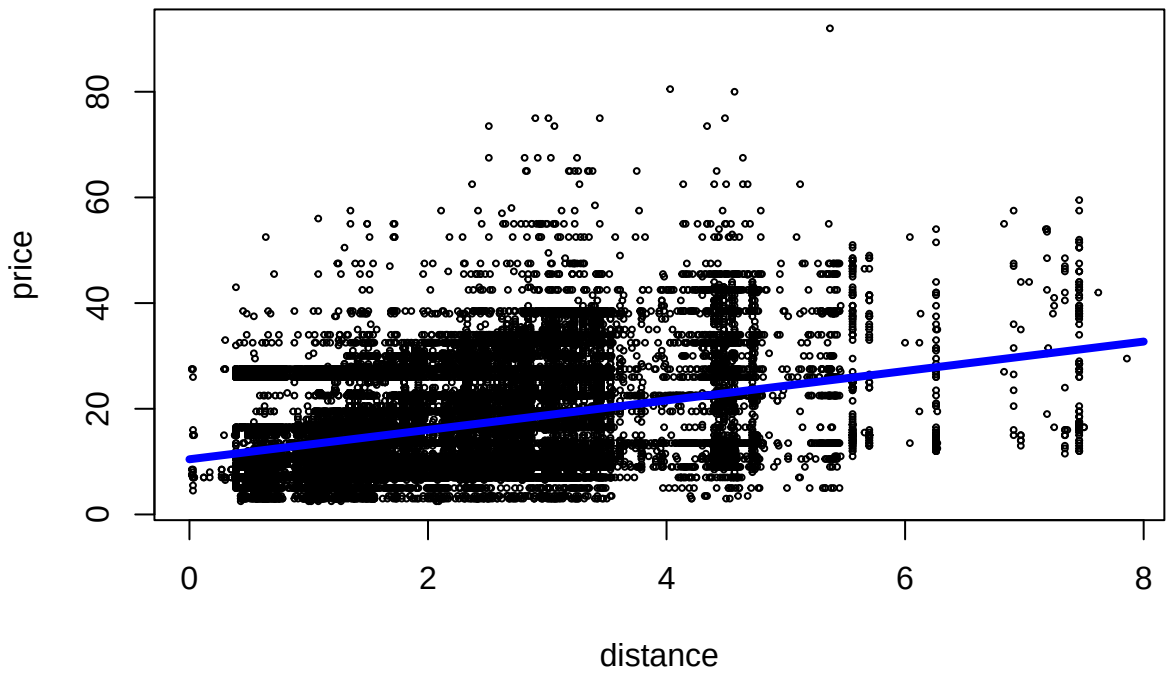
```
##   trainRMSE testRMSE
## 1  76.51672  78.0119
```

From the point estimates we can make the rough interpretation: first, the intercept is 10.33, indicating that a "starting price" would be aroun 10.33 dollar, even if it's a super short ride; second, the slope for the `distance` $is 2.82$ meaning that every 1 mile would be estimated to cost you 2.82 dollar more in general cases. The p-value for the slope is less than 2×10^{-16}, so we can say that the association is very very significant.

R^2 is roughly 0.1188. It's rather distant from 1. It means that nearly 90% of the variability is due the residual errors. We get a pretty high RMSE. This is not surprising as we expect more factors aside from sheer distance would affect the price as we learned they are considered into the pricing model. Also, we notice that the test RMSE is not significantly higher than that of the training, so the model is not overfitting at all (which is also within our expectation, as the model is rather simple and has a very low variance).

Further more we plot the prediction line to see if it matches the scatter plot we created before.

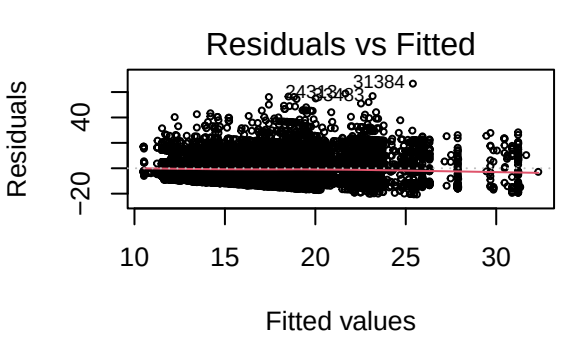
```
x.plot = seq(0, 8, 0.1)
y.pred.plot = predict(lm.simple, newdata=data.frame(distance=x.plot))
plot(price ~ distance, data=rideshare.train, cex=0.4)
lines(x.plot, y.pred.plot, col="blue", lwd=4)
```

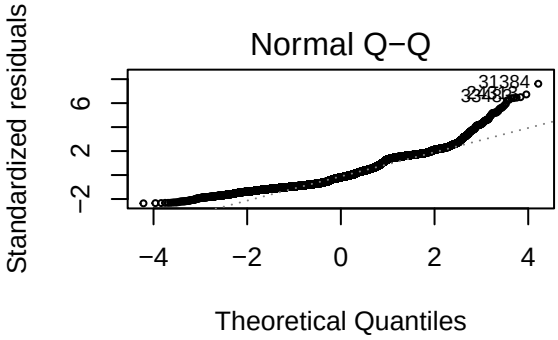


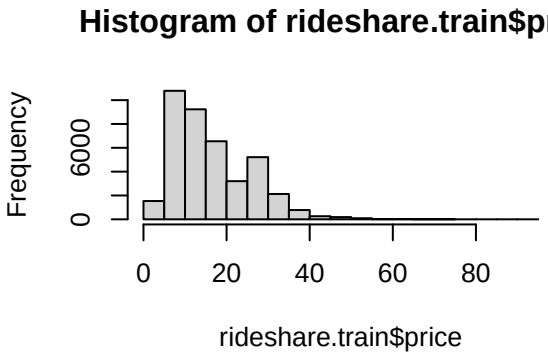
Next we check the assumptions for this linear model and see if we would need any transformations to relieve the skewness issue.

```
# residual and QQ plot
par(mfrow=c(2, 2))
plot(lm.simple, which=1:2, cex=0.5)

# y distribution plot
hist(rideshare.train$price)
```







According to the residual plot, the linearity and constant variance assumption are all poorly conformed. There a lot more positive residuals than negative ones, so `lm.simple` is underestimating the prices. The variability of residuals seem greater when the predicted prices are within range $[18, 23]$. From the QQ plot and the histogram we can see that the normality is also not met. The data looks quite right-skew, and slightly bi-modal.

Polynomial Regression

Since the linear model has a relatively not so satisfying performance, and the linearity is not well met, we think it might be worth a try to use polynomial model. We want to start with a quadratic model

```
lm.quadratic = lm(price ~ poly(distance, 2, raw=T), data=rideshare.train)

# quadratic model summary
summary(lm.quadratic)
```

```
##
## Call:
## lm(formula = price ~ poly(distance, 2, raw = T), data = rideshare.train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -20.565  -6.968  -1.707   4.972  66.654
##
## Coefficients:
```

```
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  10.396626   0.152021  68.389   <2e-16 ***
## poly(distance, 2, raw = T)1  2.827058   0.119990  23.561   <2e-16 ***
## poly(distance, 2, raw = T)2 -0.008059   0.021362  -0.377    0.706
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.748 on 39997 degrees of freedom
## Multiple R-squared:  0.1161, Adjusted R-squared:  0.116
## F-statistic:  2626 on 2 and 39997 DF,  p-value: < 2.2e-16
```

```
# F-test
anova(lm.simple, lm.quadratic)
```

```
## Analysis of Variance Table
##
## Model 1: price ~ distance
## Model 2: price ~ poly(distance, 2, raw = T)
##   Res.Df     RSS Df Sum of Sq      F Pr(>F)
## 1  39998 3060669
## 2  39997 3060658  1    10.891 0.1423  0.706
```

From the p-values from both the t-test of the quadratic term and the F-test between `lm.simple` and `lm.quadratic` we can see that the added quadratic term is not of much help, so we will not further more consider it.

Uber or Lyft?

Here we want to research into a very intriguing question: is Uber and Lyft exhibiting different pricing models? To answer this question, we will experiment with another linear model incorporating the type of the ride into the consideration. We call this model `lm.brand`. It considers the `distance` and `cab_type`, as well as their interaction

```
lm.brand = lm(price ~ (distance + cab_type) ^ 2, data=rideshare.train)
summary(lm.brand)
```

```
##
## Call:
## lm(formula = price ~ (distance + cab_type)^2, data = rideshare.train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -22.918  -6.609  -1.655   4.857  64.341
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)            10.28198    0.14079  73.030   <2e-16 ***
## distance                3.23597    0.05758  56.198   <2e-16 ***
## cab_typeUber            0.16302    0.18972   0.859     0.39
## distance:cab_typeUber -0.80685    0.07697 -10.483   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 8.699 on 39996 degrees of freedom
## Multiple R-squared:  0.1259, Adjusted R-squared:  0.1258
## F-statistic:  1920 on 3 and 39996 DF,  p-value: < 2.2e-16
```

```
# compare this to the simple model
anova(lm.simple, lm.brand)
```

```
## Analysis of Variance Table
##
## Model 1: price ~ distance
## Model 2: price ~ (distance + cab_type)^2
##   Res.Df     RSS Df Sum of Sq      F    Pr(>F)
## 1  39998 3060669
## 2  39996 3026649  2     34019 224.78 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# RMSEs
y.pred.brand.train = predict(lm.brand)
y.pred.brand.test = predict(lm.brand, new=rideshare.test)
lm.brand.train.rmse = RMSE(rideshare.train$price, y.pred.brand.train)
lm.brand.test.rmse = RMSE(rideshare.test$price, y.pred.brand.test)
data.frame(trainRMSE=lm.brand.train.rmse, testRMSE=lm.brand.test.rmse)
```

```
##   trainRMSE testRMSE
## 1  75.66623 77.47752
```

From the summary of the model we can derive some useful insights. When `cab_type` is `Lyft`, the regression formula is

$$\text{price} = 10.28 + 3.24 \times \text{distance}$$

; While if `cab_type` is `Uber`, the regression formula is

$$\text{price} = 10.44 + 2.43 \times \text{distance}$$

. This suggests that `Lyft` have a slightly lower starting price but would have a significantly higher fare rate w.r.t the distance compared to Uber rides. The pricing model for these two companies are in fact quite different. The t-test result supports this claim: the coefficients for `cab_typeUber` is not significant with a p-value as big as 0.55, while the coefficients for `distance:cab_typeUber` has a p-value less than 2×10^{-16}, indicating extreme significance. So we would have the impression that in Boston during 2018 in the winter, Lyft would be more expensive for long-distance rides and cheaper for short-distance ones.

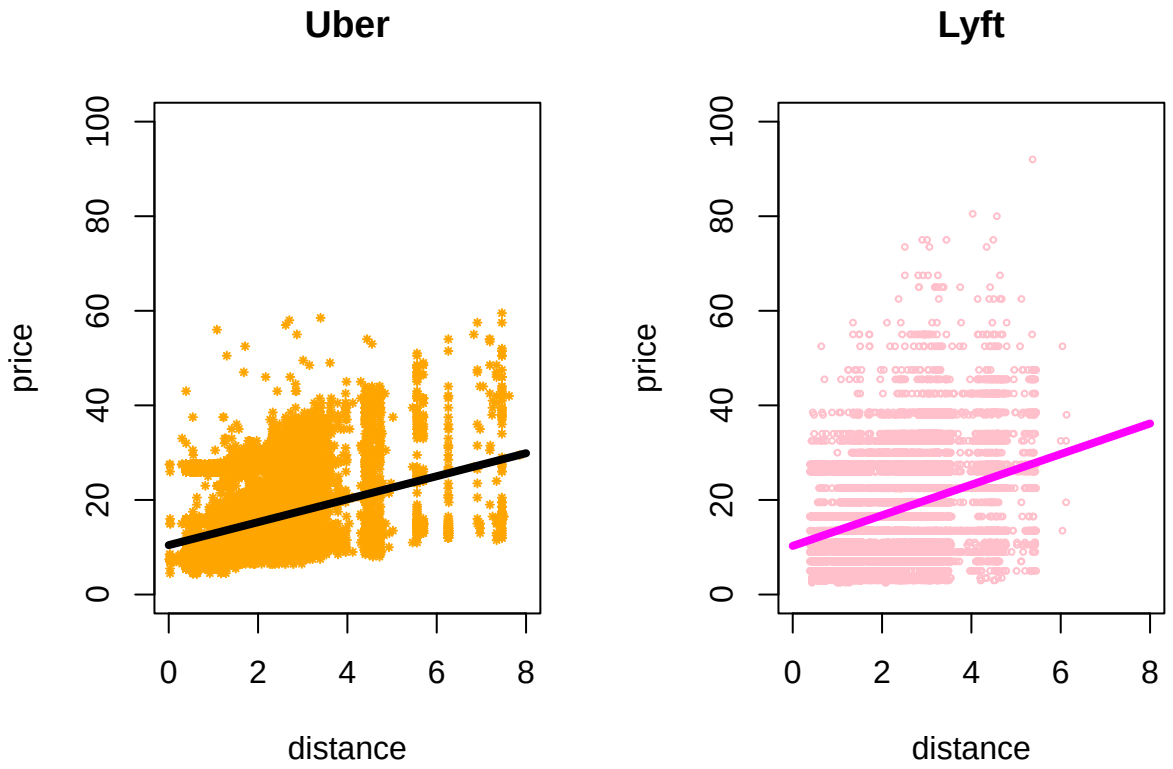
To explore whether this is an improvement compared to the naive linear regression, we perform an F-test and calculated the RMSEs. The p-value for the F-test is less than 2.2×10^{-16}, suggesting a significant improve. Also both train & test RMSEs are better compared to `lm.simple`.

Next, we will provide some visualizations to support this intuition. We will plot the scattered points as well as the prediction line, but in different colors to separate these two cab brands: pink for Lyft, and Orange for Uber.

```
par(mfrow=c(1,2))
rideshare.train.uber = rideshare.train[rideshare.train$cab_type=="Uber", ]
rideshare.train.lyft = rideshare.train[rideshare.train$cab_type=="Lyft", ]
```

```r
# Uber plot
plot(price ~ distance, data=rideshare.train.uber, col="orange",
     cex=0.4, pch=8,  xlim=c(0, 8), ylim=c(0, 100),
     main="Uber")
y.pred.plot.uber = predict(lm.brand,
                           newdata=data.frame(distance=x.plot, cab_type=rep("Uber",
                          length(x.plot))))
lines(x.plot, y.pred.plot.uber, col="black", lwd=4)

# Lyft plot
plot(price ~ distance, data=rideshare.train.lyft, col="pink",
     cex=0.4,
     xlim=c(0, 8),
     ylim=c(0, 100), main="Lyft")
y.pred.plot.lyft = predict(lm.brand,
                           newdata=data.frame(distance=x.plot, cab_type=rep("Lyft", length(x.plot))))
lines(x.plot, y.pred.plot.lyft, col="Magenta", lwd=4)
```



From the prediction line we can see that Lyft has a steeper prediction line, suggesting a significant higher fare rate. However, according to the scatter plots, we can easily find that Uber ride fares have better linearity w.r.t the ride distance, and has a smaller variance in the price.

Linear model using all the predictors

```
# a linear regression model using all predictors, without any interaction terms
lm.all = lm(price ~ ., data=rideshare.train)
summary(lm.all)
```

```
##
## Call:
## lm(formula = price ~ ., data = rideshare.train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -12.469  -1.652  -0.321   1.173  50.292
##
## Coefficients: (2 not defined because of singularities)
##                                  Estimate Std. Error  t value Pr(>|t|)
## (Intercept)                     5.715e-01  2.012e+00    0.284 0.776396
## distance                        2.916e+00  1.757e-02  165.975  < 2e-16 ***
## cab_typeUber                    1.462e+01  7.586e-02  192.760  < 2e-16 ***
## daycount                       -8.226e-04  2.501e-03   -0.329 0.742187
## weather Cloudy                 -1.414e-02  5.065e-02   -0.279 0.780160
## weather Foggy                  -2.826e-01  1.639e-01   -1.724 0.084676 .
## weather Rain                   -6.699e-02  9.465e-02   -0.708 0.479097
## sourceBeacon Hill              -5.248e-01  7.534e-02   -6.966 3.32e-12 ***
## sourceBoston University        -3.878e-01  1.054e-01   -3.679 0.000234 ***
## sourceFenway                   -3.456e-02  1.047e-01   -0.330 0.741320
## sourceFinancial District        7.055e-02  7.565e-02    0.933 0.351021
## sourceHaymarket Square         -5.752e-02  1.031e-01   -0.558 0.577054
## sourceNorth End                 1.851e-01  1.022e-01    1.812 0.070031 .
## sourceNorth Station            -3.046e-01  7.549e-02   -4.035 5.48e-05 ***
## sourceNortheastern University  -3.861e-01  1.046e-01   -3.690 0.000224 ***
## sourceSouth Station            -1.525e-02  1.020e-01   -0.149 0.881182
## sourceTheatre District          5.178e-01  7.587e-02    6.825 8.92e-12 ***
## sourceWest End                 -3.706e-01  7.613e-02   -4.868 1.13e-06 ***
## destinationBeacon Hill         -2.957e-01  7.578e-02   -3.902 9.57e-05 ***
## destinationBoston University    1.245e-02  7.944e-02    0.157 0.875444
## destinationFenway              -2.659e-01  7.841e-02   -3.391 0.000696 ***
## destinationFinancial District   4.576e-01  7.609e-02    6.014 1.82e-09 ***
## destinationHaymarket Square     2.426e-01  7.535e-02    3.220 0.001284 **
## destinationNorth End            1.036e-01  7.508e-02    1.380 0.167675
## destinationNorth Station        2.021e-01  7.570e-02    2.670 0.007599 **
## destinationNortheastern University  1.050e-02  7.730e-02    0.136 0.891915
## destinationSouth Station               NA         NA       NA       NA
## destinationTheatre District     2.212e-01  7.542e-02    2.933 0.003358 **
## destinationWest End             4.544e-03  7.580e-02    0.060 0.952191
## nameBlack SUV                   9.654e+00  7.438e-02  129.794  < 2e-16 ***
## nameLux                         1.189e+01  7.718e-02  154.119  < 2e-16 ***
## nameLux Black                   1.707e+01  7.671e-02  222.584  < 2e-16 ***
## nameLux Black XL                2.625e+01  7.738e-02  339.193  < 2e-16 ***
## nameLyft                        3.561e+00  7.697e-02   46.264  < 2e-16 ***
## nameLyft XL                     9.322e+00  7.713e-02  120.864  < 2e-16 ***
## nameShared                             NA         NA       NA       NA
```

```
## nameUberPool                       -1.182e+01  7.428e-02 -159.121  < 2e-16 ***
## nameUberX                          -1.084e+01  7.388e-02 -146.786  < 2e-16 ***
## nameUberXL                         -4.934e+00  7.444e-02  -66.282  < 2e-16 ***
## nameWAV                            -1.091e+01  7.381e-02 -147.847  < 2e-16 ***
## temperature                         3.885e-04  2.638e-03    0.147 0.882889
## humidity                           -2.705e-01  1.743e-01   -1.552 0.120660
## windSpeed                          -2.870e-03  7.780e-03   -0.369 0.712211
## visibility                         -2.039e-02  1.228e-02   -1.661 0.096691 .
## pressure                           -4.671e-04  1.969e-03   -0.237 0.812501
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.078 on 39957 degrees of freedom
## Multiple R-squared:  0.8907, Adjusted R-squared:  0.8906
## F-statistic:  7750 on 42 and 39957 DF,  p-value: < 2.2e-16
```

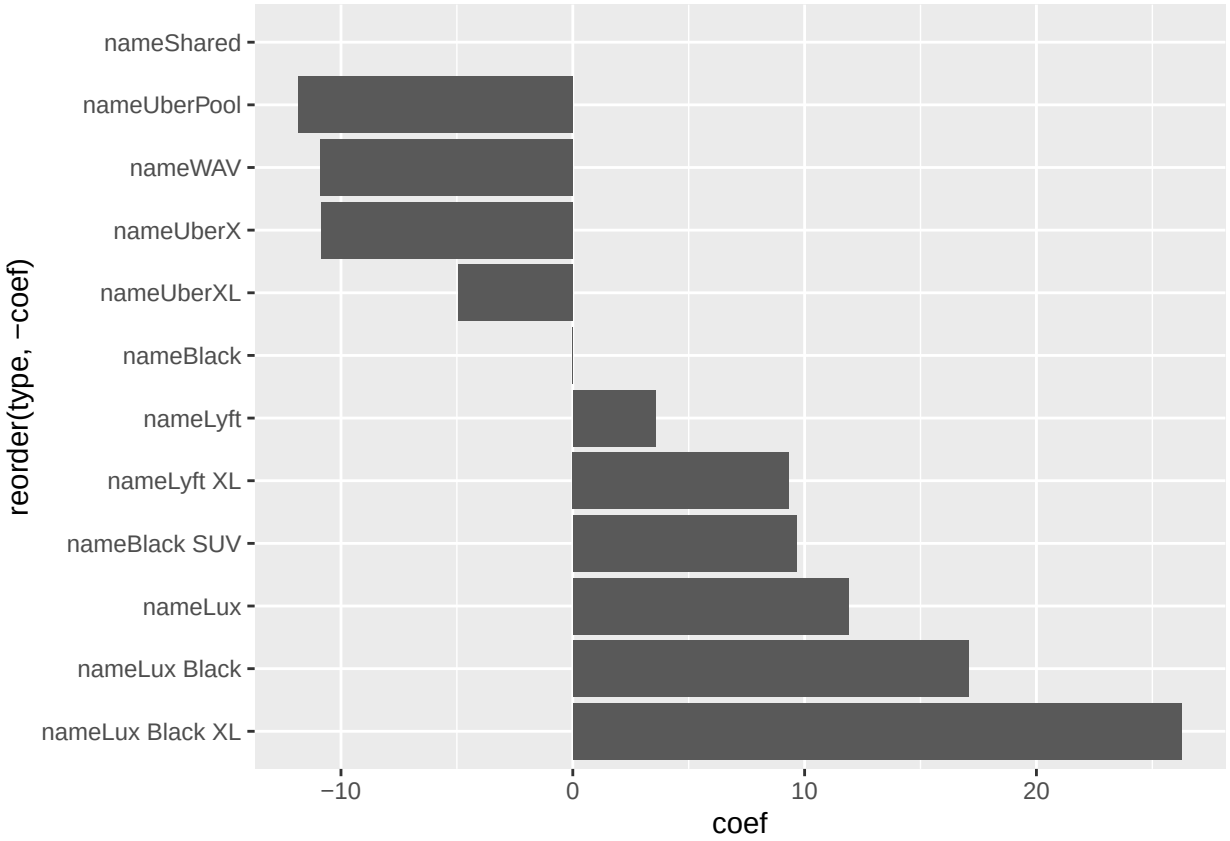
From the summary we can see that the significant predictors are `cab_type, name, distance, source, destination`. The coefficient for distance is 2.92, slightly higher than that we got in the simple linear regression. This means, holding every other factor constant, every mile would cost you 2.92 dollar on average.

The coefficient for `cab_typeUber` is 14.62, which is drastically different from the result we had when we used simple linear regression, indicating that generally Uber is more expensive compared to Lyft. The difference between the fare of these two brands do not seem to have so much gap, so it's probably due to the multicolinearities since we had another categorical predictor `name` which describes the specific type of the car. The reference group for this variable is `Black`, which is one of the Lyft's luxury ride type. Based on the coefficients, we can create a visual to see the fare ranking of these ride types.

```
lm.all.ridetype = lm.all$coefficients[30:40]
lm.all.ridetype['nameBlack'] = 0
lm.all.ridetype = data.frame(lm.all.ridetype)
lm.all.ridetype['type'] = rownames(lm.all.ridetype)
colnames(lm.all.ridetype) = c('coef', 'type')
rownames(lm.all.ridetype) = NULL

library(ggplot2)
ggplot(data=lm.all.ridetype, aes(x=coef, y=reorder(type, -coef))) +
  geom_bar(stat="identity")
```

```
## Warning: Removed 1 rows containing missing values (`position_stack()`).
```



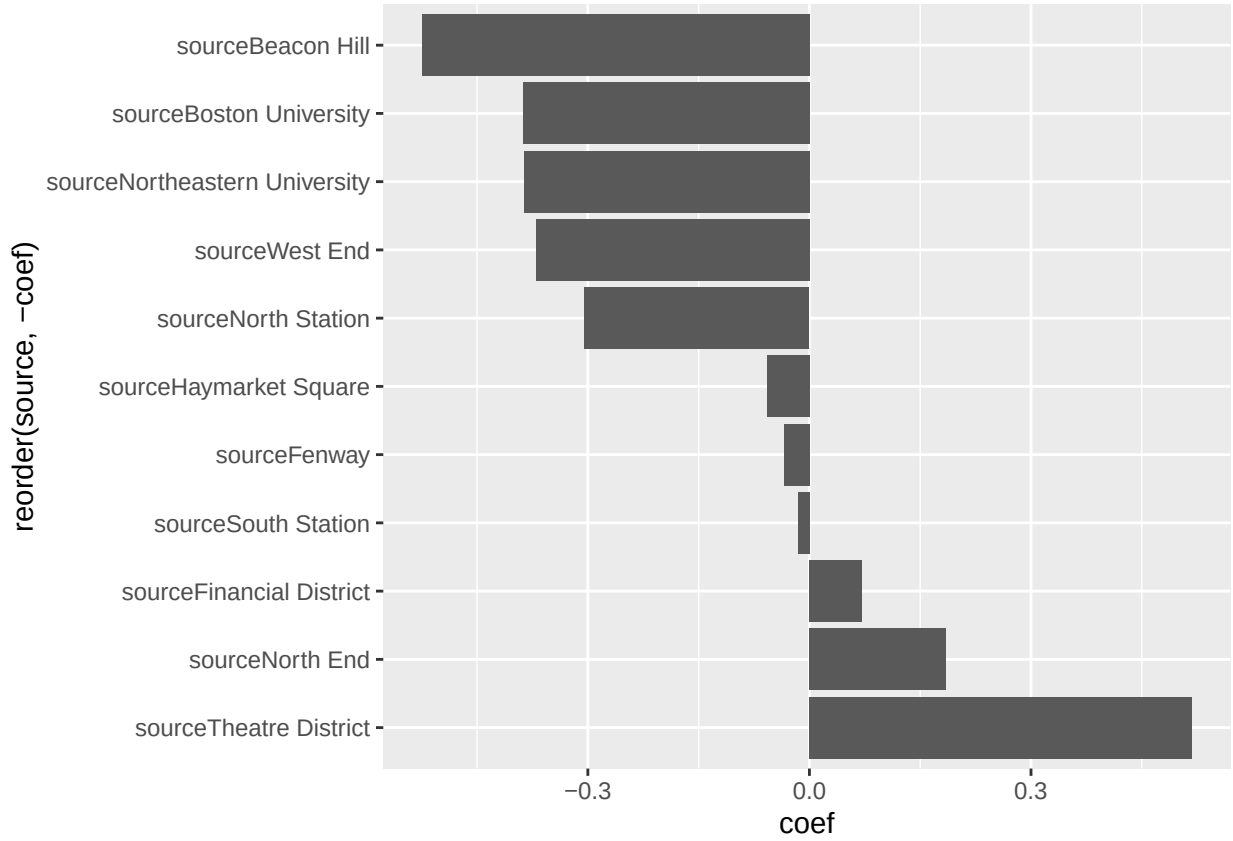
From the ranked barplot we can see that:

- shared rides have NA estimates, which is due to it is linearly related to other columns (prabably `cab_typeUber` and all other ride types of Uber will determine it).
- Uber Pool, Uber X and Lyft are the cheapest rides generally speaking.
- WAV is also very cheap because it is to certain extent a social welfare for the physically disabled citizens.
- The XL rides are more expensive, and the luxury models (Lyft Black SUV, Uber Lux, Lyft Black XL, etc) are much more expensive.
- It's surprising that Lyft Black actually seems to be cheaper than Lyft standard. But it may happen sometimes because the pricing model is probably due to machine learning and the over-demand of Lyft standard might cause a rise in the fare rate, while not many people choose Lyft Black and will cause a lower price.
- Generally for the same level (like Uber Lux compared to Lyft Lux Black, or UberX compared to Lyft standard), the coefficients for Uber rides are significantly lower, probably because of there is a large compensation coefficient for `cab_typeUber` already. Considering both factors, the Lyft ride still seems a better deal, but with marginal advantage. For example, the coefficient for `nameLyft` is 3.56, while the summation of `cab_typeUber` and `UberX` is 3.78.

Let's create two more visuals for the source and destination coefficients.

```
lm.all.source = lm.all$coefficients[8:18]
lm.all.source = data.frame(lm.all.source)
lm.all.source['source'] = rownames(lm.all.source)
colnames(lm.all.source) = c('coef', 'source')
rownames(lm.all.source) = NULL
```

```
ggplot(data=lm.all.source, aes(x=coef, y=reorder(source, -coef))) +
  geom_bar(stat="identity")
```

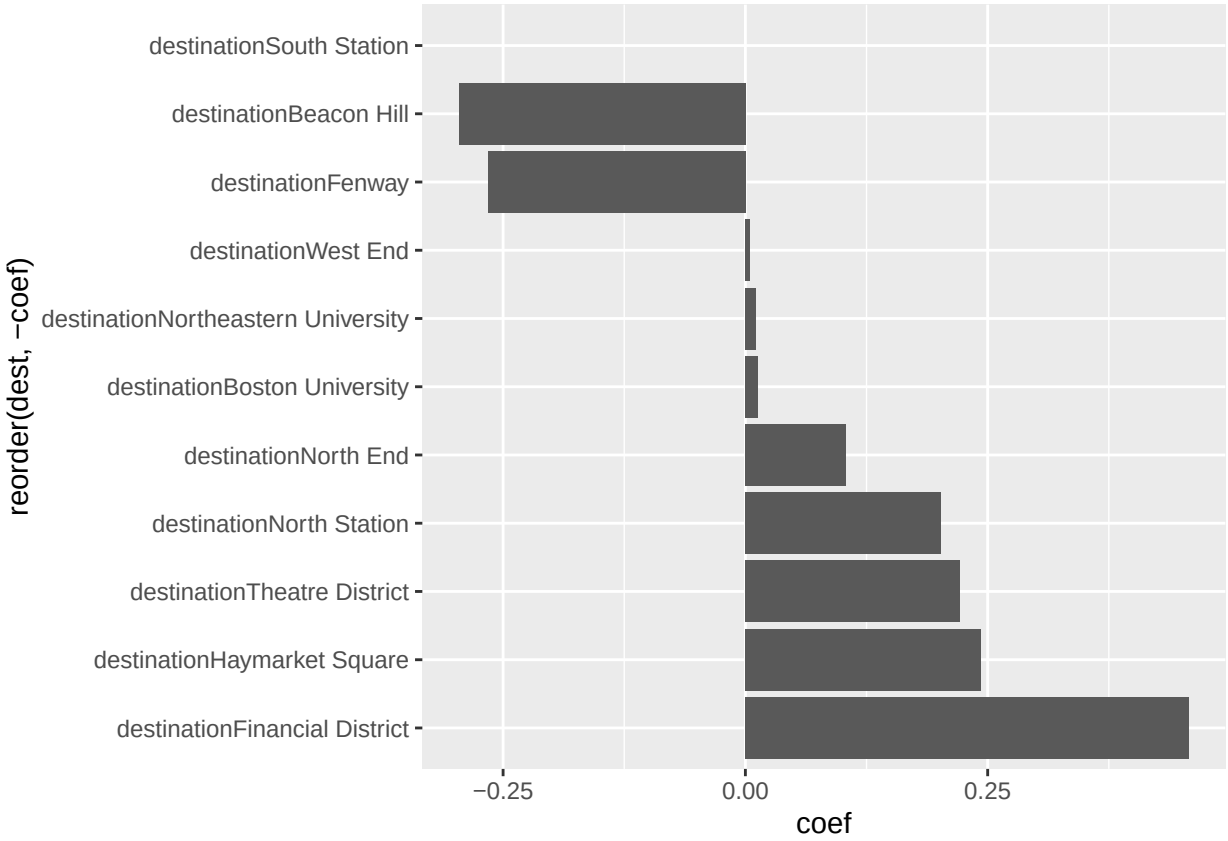


According to this visual, the seemingly most expensive source location is Theatre District near Downtown Crossing, followed by North Station where TD Garden is located. This makes perfect sense because Boston Downtown has a large population density and TD Garden has many games and performances which will induce heavy traffic, so the dynamic pricing system might yield a higher price due to the high demand.

The least expensive source locations are Beacon Hill, NEU and BU. Beacon Hill is located at Park Street, near Downtown, so it's a bit confusing. NEU and BU are located in Malden and Allston, and the riders should be mostly students nearby, and they might tend to choose cheaper rides.

```
lm.all.dest = lm.all$coefficients[19:29]
lm.all.dest = data.frame(lm.all.dest)
lm.all.dest['dest'] = rownames(lm.all.dest)
colnames(lm.all.dest) = c('coef', 'dest')
rownames(lm.all.dest) = NULL
ggplot(data=lm.all.dest, aes(x=coef, y=reorder(dest, -coef))) +
  geom_bar(stat="identity")
```

```
## Warning: Removed 1 rows containing missing values (`position_stack()`).
```



The visual for the destination shows a similar trend. Places near Downtown like Financial District and Theatre District, and places near TD Garden has a higher ride price in general.

We didn't see significances in the factor of weather, which is out of our expectation, because we presumed that during the Holiday seasons, the extreme weather conditions like storm would impact the ride prices positively. The estimated coefficients for `weatherFoggy, weatherRain, weatherCloudy` are negative, contradicting to our theory. But there is an explanation. These categorical predictors have a high correlation with the numerical predictors like `visibility`, `temperature`, `windSpeed`, `humidity`. And the coefficients are in accordance with our assumption: the worse the weather condition is, like lower visibility and higher wind speed, would have a positive impact on the ride price. The only exception is when the temperature is higher, the ride would cost more. Since the data is collected during November and December, it seems a little bit confusing.

The `daycount` has a negative coefficient, meaning the later it is, the lower the price is. This also contradicts our common sense, as we would expect a higher price during holiday season near Christmas, but since the p-value is super high (0.88), so maybe this can be neglegible.

The time period seems not so significant in the prediction, but generally during late night and early morning when there are less passengers, the price would be lower.

Let's calculate the RMSE as a comparison to other models

```
lm.all.rmse.train = RMSE(
  rideshare.train$price,
  predict(lm.all)
)

lm.all.rmse.test = RMSE(
```

```
  rideshare.test$price,
  predict(lm.all, new=rideshare.test)
)
```

```
## Warning in predict.lm(lm.all, new = rideshare.test): prediction from a rank-
## deficient fit may be misleading
```

```
lm.all.rmse = data.frame(trainRMSE=lm.all.rmse.train, testRMSE=lm.all.rmse.test)
lm.all.rmse
```

```
##   trainRMSE testRMSE
## 1   9.46417 9.081773
```

The RMSE is much lower (9.32 for train and 10.04 for test), but still not satisfying, the prediction can have up to 10 dollar error off the real price.

Forward Model Selection

In this section we will try to explore which predictors and interactions are important with forward model selection. Let's start with a full interaction model that takes into account all the predictors as well as their interaction terms.

```
lm.full = lm(price ~ .^2, data=rideshare.train)
lm.full.rmse.train = RMSE(
  rideshare.train$price,
  predict(lm.full)
)

lm.full.rmse.test = RMSE(
  rideshare.test$price,
  predict(lm.full, new=rideshare.test)
)

lm.full.rmse = data.frame(trainRMSE=lm.full.rmse.train, testRMSE=lm.full.rmse.test)
lm.full.rmse
```

```
##   trainRMSE testRMSE
## 1  6.686429 6.745499
```

Both train and test RMSEs are decreased (6.53 for train and 7.73 for test), indicating an improvement in the prediction capability. However, for the purpose of analysis, we also used AIC forward selection:

```
lm.aic = step(
  lm.simple,
  scope=list(upper=lm.full, lower=~1),
  direction="forward",
  trace=F
)

lm.aic.rmse.train = RMSE(
```

```
  rideshare.train$price,
  predict(lm.aic)
)

lm.aic.rmse.test = RMSE(
  rideshare.test$price,
  predict(lm.aic, new=rideshare.test)
)
```

```
## Warning in predict.lm(lm.aic, new = rideshare.test): prediction from a rank-
## deficient fit may be misleading
```

```
lm.aic.rmse = data.frame(trainRMSE=lm.aic.rmse.train, testRMSE=lm.aic.rmse.test)

# RMSE result for AIC forward selection model
lm.aic.rmse
```

```
##   trainRMSE testRMSE
## 1  6.751723 6.670986
```

The train RMSE increases a little bit compared to the full model (6.61), but test RMSE are decreased (7.68), indicating a slight improvement in the overfitting issue. We can see the formula of this forward selection model:

```
formula(lm.aic)
```

```
## price ~ distance + name + source + destination + distance:name +
##     name:source + distance:source + source:destination + name:destination +
##     distance:destination
```

From the formula we can see that the important predictors are just the same as what we observed in `lm.all`, the key factors are `distance`, `name` (which is the specific car type), `destination`, `source` and their pairwise interaction.