Regression

Derrick Martin

09/25/2022

Linear Regression Overview

Linear regression works by plotting the data and attempting to find a correlation for which the data best fits. The algorithm attempts to predict a target using a number of predictors, often the slope of the line represents the relational change between the variables. Linear Regression is used because it is simple and quite easy to do but it often over fits the data and has a hard time ignoring noise.

Choosing test and training data

This code block chooses 80 percent of the data from the file at random and assigns it to be the training data, and takes the remaining 20 percent of the data and assigns it to be the testing data. The data is on used audi cars and is sourced from Kaggle.

```
library(readr)
df <- read_csv("audi.csv", show_col_types = FALSE)

set.seed(1)

sample <- sample(c(TRUE,FALSE), nrow(df), replace = TRUE, prob = c(0.80,0.20))

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

Data Exploration

This code is showing different information about the data so that we can get a better idea of the values and counts of different items.

```
names(train)
```

```
## [1] "model"        "year"         "price"        "transmission" "mileage"
## [6] "fuelType"     "tax"          "mpg"          "engineSize"
```

```
dim(train)
```

```
## [1] 8456    9
```

```
summary(train)
```

```
##     model                year          price        transmission      
##  Length:8456        Min.   :1997   Min.   :  1490   Length:8456       
##  Class :character   1st Qu.:2016   1st Qu.: 15250   Class :character  
##  Mode  :character   Median :2017   Median : 20250   Mode  :character  
##                     Mean   :2017   Mean   : 22872                     
##                     3rd Qu.:2019   3rd Qu.: 27990                     
##                     Max.   :2020   Max.   :145000                     
##     mileage         fuelType              tax             mpg        
##  Min.   :     1   Length:8456        Min.   :  0.0   Min.   : 18.90  
##  1st Qu.:  5948   Class :character   1st Qu.:125.0   1st Qu.: 40.90  
##  Median : 18890   Mode  :character   Median :145.0   Median : 49.60  
##  Mean   : 24776                      Mean   :125.6   Mean   : 50.82  
##  3rd Qu.: 36361                      3rd Qu.:145.0   3rd Qu.: 58.90  
##  Max.   :323000                      Max.   :580.0   Max.   :188.30  
##    engineSize   
##  Min.   :0.000  
##  1st Qu.:1.500  
##  Median :2.000  
##  Mean   :1.929  
##  3rd Qu.:2.000  
##  Max.   :6.300  
```

```
str(train)
```

```
## tibble [8,456 × 9] (S3: tbl_df/tbl/data.frame)
##  $ model       : chr [1:8456] "A1" "A6" "A1" "A3" ...
##  $ year        : num [1:8456] 2017 2016 2016 2019 2016 ...
##  $ price       : num [1:8456] 12500 16500 11000 17300 11750 ...
##  $ transmission: chr [1:8456] "Manual" "Automatic" "Manual" "Manual" ...
##  $ mileage     : num [1:8456] 15735 36203 29946 1998 75185 ...
##  $ fuelType    : chr [1:8456] "Petrol" "Diesel" "Petrol" "Petrol" ...
##  $ tax         : num [1:8456] 150 20 30 145 20 20 30 145 125 145 ...
##  $ mpg         : num [1:8456] 55.4 64.2 55.4 49.6 70.6 60.1 55.4 58.9 57.6 52.3 ...
##  $ engineSize  : num [1:8456] 1.4 2 1.4 1 2 1.4 1.4 1.4 2 2 ...
```

```
head(train)
```

model <chr>	year <dbl>	price <dbl>	transmission <chr>	mileage <dbl>	fuelType <chr>	tax <dbl>	mpg <dbl>	engineSize <dbl>
A1	2017	12500	Manual	15735	Petrol	150	55.4	1.4
A6	2016	16500	Automatic	36203	Diesel	20	64.2	2.0

model	year	price	transmission	mileage	fuelType	tax	mpg	engineSize
<chr>	<dbl>	<dbl>	<chr>	<dbl>	<chr>	<dbl>	<dbl>	<dbl>
A1	2016	11000	Manual	29946	Petrol	30	55.4	1.4
A3	2019	17300	Manual	1998	Petrol	145	49.6	1.0
A4	2016	11750	Manual	75185	Diesel	20	70.6	2.0
A3	2015	10200	Manual	46112	Petrol	20	60.1	1.4

6 rows

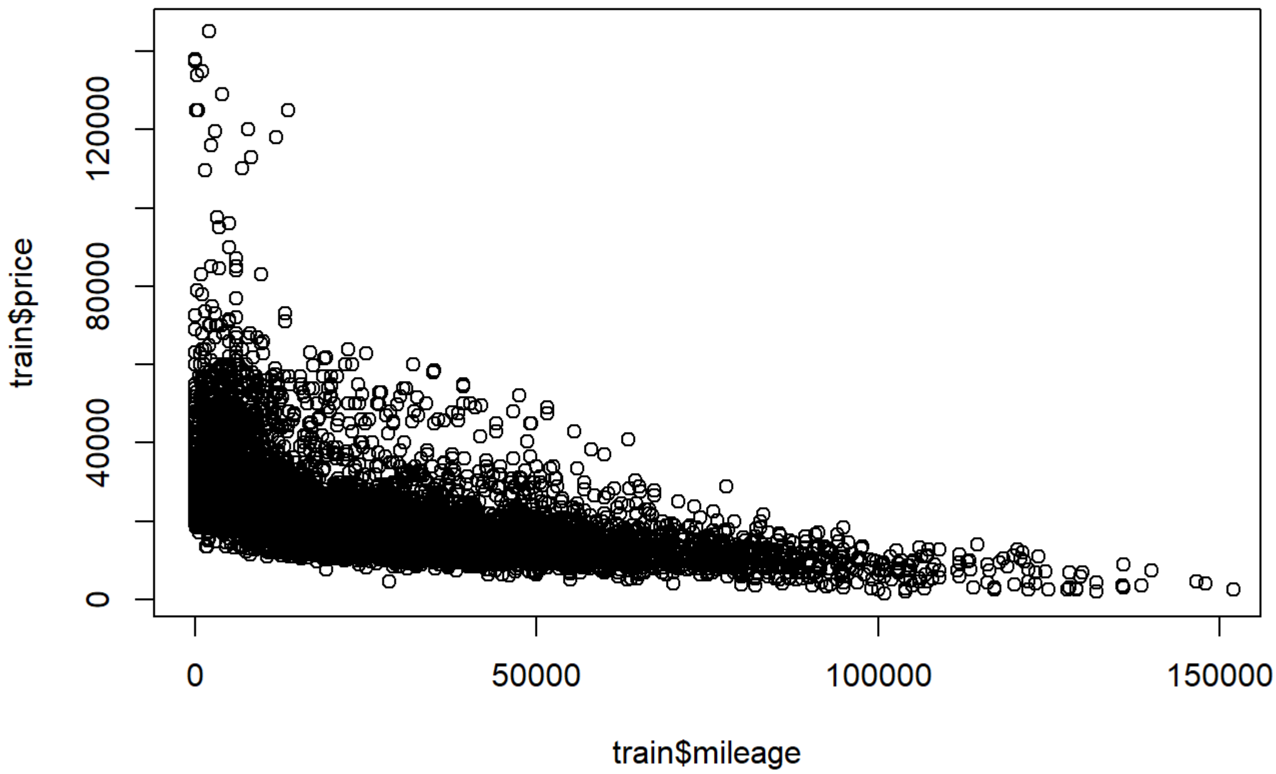
```
tail(train)
```

model	year	price	transmission	mileage	fuelType	tax	mpg	engineSize
<chr>	<dbl>	<dbl>	<chr>	<dbl>	<chr>	<dbl>	<dbl>	<dbl>
A3	2013	12695	Manual	31500	Petrol	125	53.3	1.4
A3	2020	16999	Manual	4018	Petrol	145	49.6	1.0
A3	2020	16999	Manual	1978	Petrol	150	49.6	1.0
A3	2020	17199	Manual	609	Petrol	150	49.6	1.0
Q3	2017	19499	Automatic	8646	Petrol	150	47.9	1.4
Q3	2016	15999	Manual	11855	Petrol	150	47.9	1.4

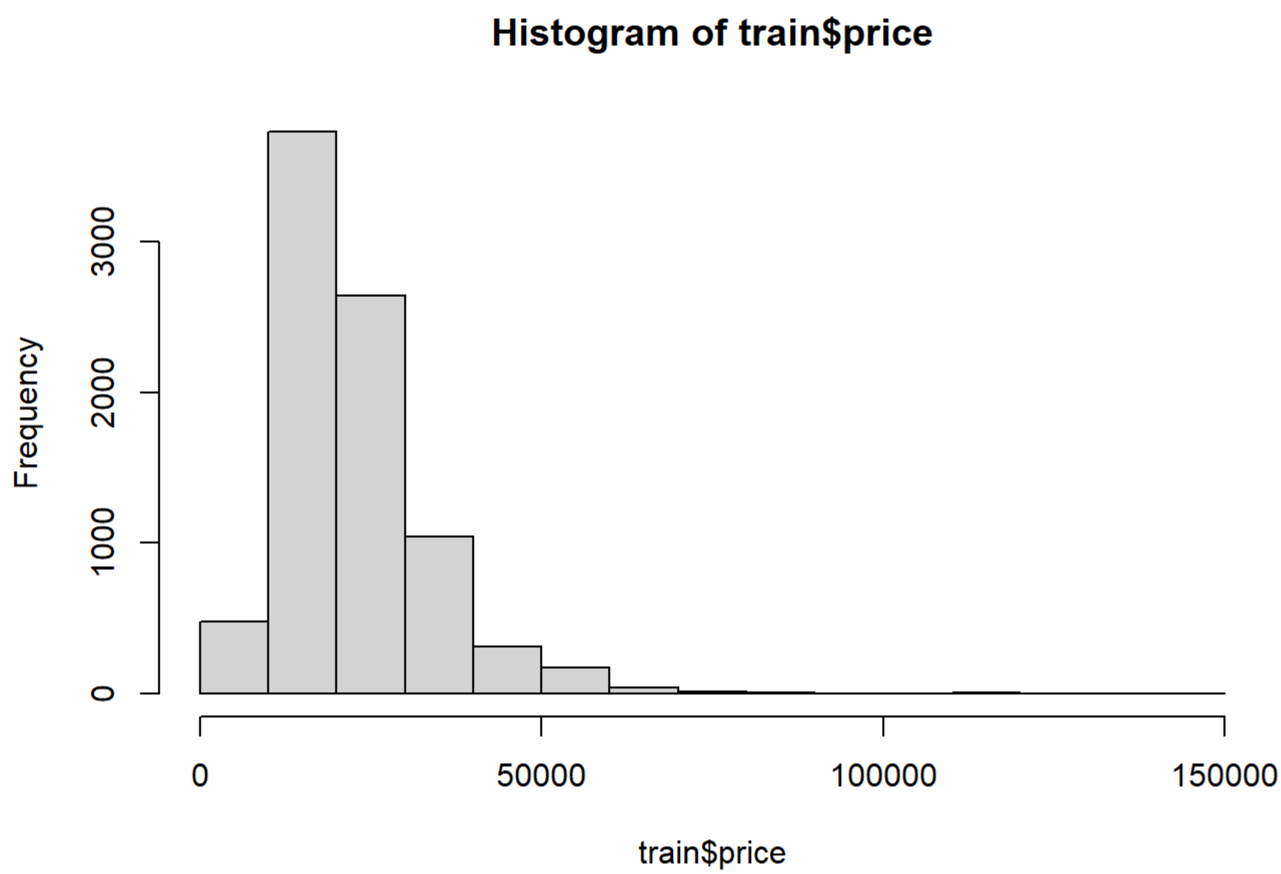
6 rows

Graphs

```
plot(x = train$mileage, y = train$price, xlim = c(0, 150000))
```




```
hist(train$price)
```



Linear Model

The Residuals in the linear model is the difference between the predicted value of y and the actual value of y. In this case the y value is the mileage, so a median difference of 1949 is actually not that bad considering the mileages are often in the hundreds of thousands. The three stars next to the mileage value indicates that it is a good predictor, however the relatively low R squared value indicates a low correlation.

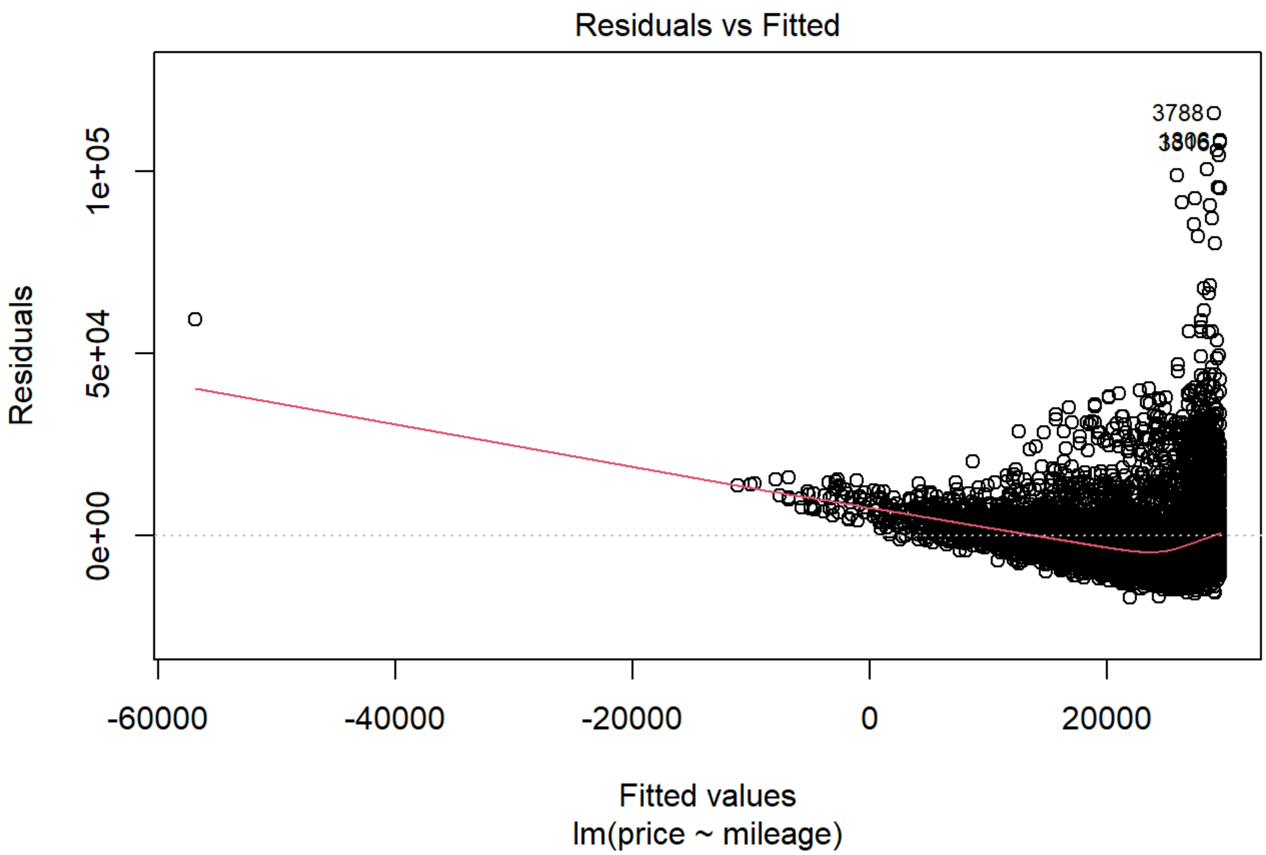
```
model1 <- lm(formula = price ~ mileage, data=train)
summary(model1)
```

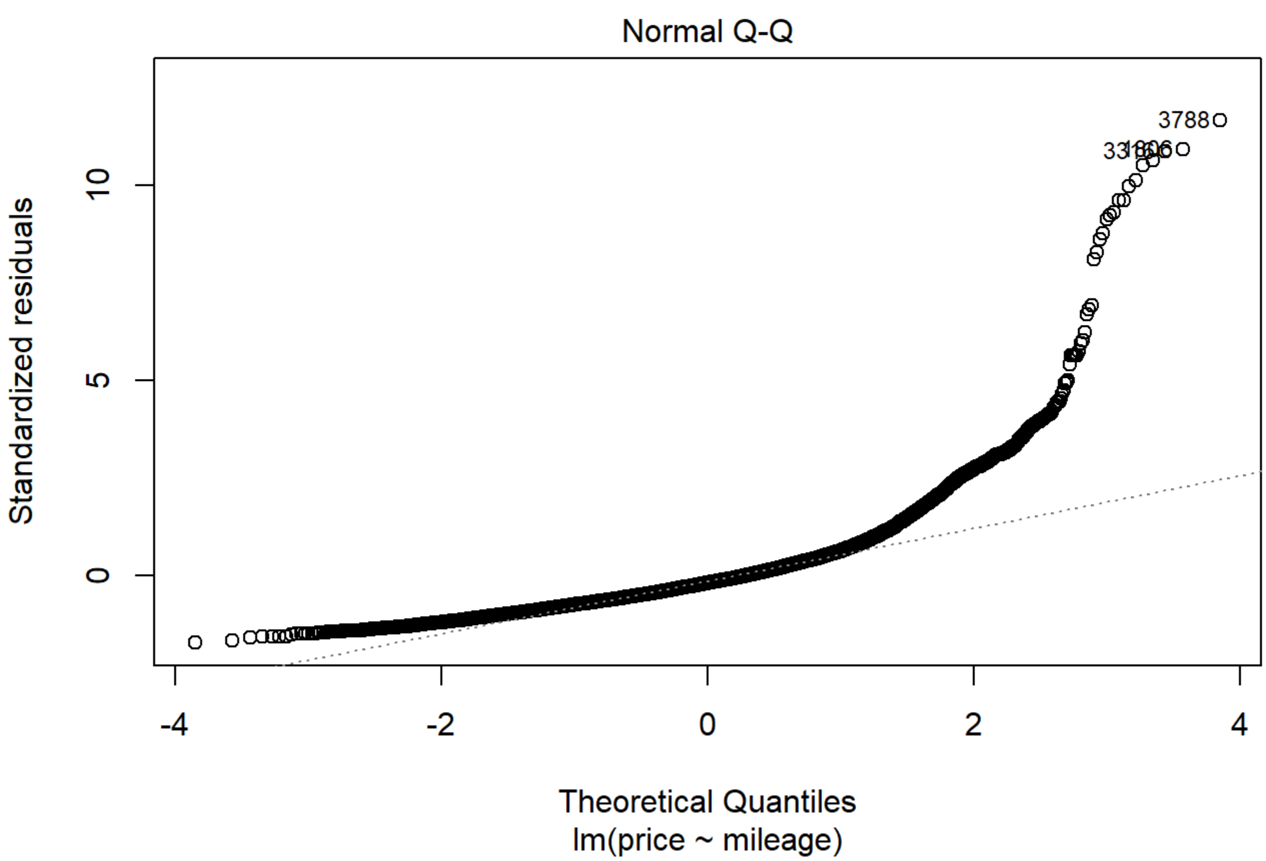
```
##
## Call:
## lm(formula = price ~ mileage, data = train)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -17193  -5934  -1949   3106 116038
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.950e+04  1.568e+02  188.12   <2e-16 ***
## mileage     -2.674e-01  4.589e-03  -58.27   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9929 on 8454 degrees of freedom
## Multiple R-squared:  0.2865, Adjusted R-squared:  0.2864
## F-statistic:  3395 on 1 and 8454 DF,  p-value: < 2.2e-16
```

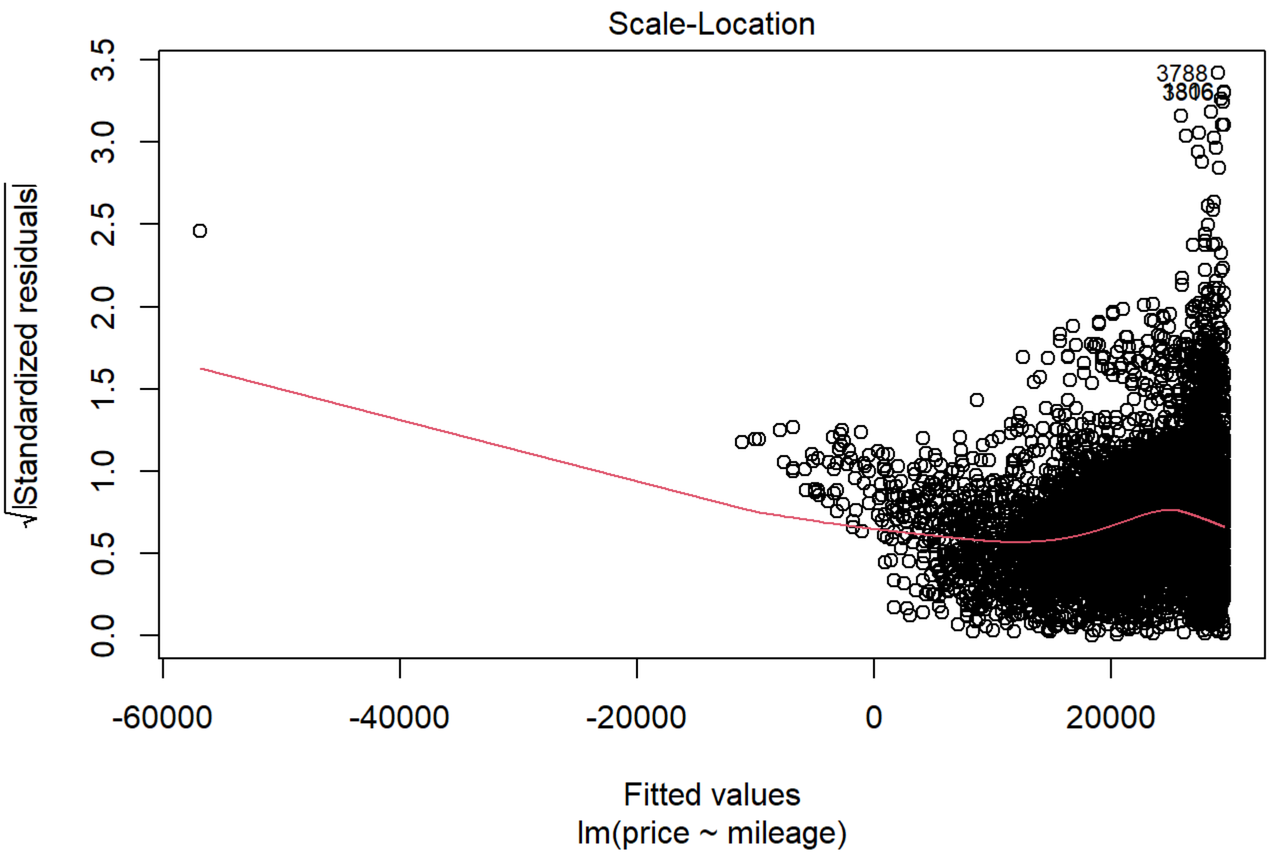
Residual Plots

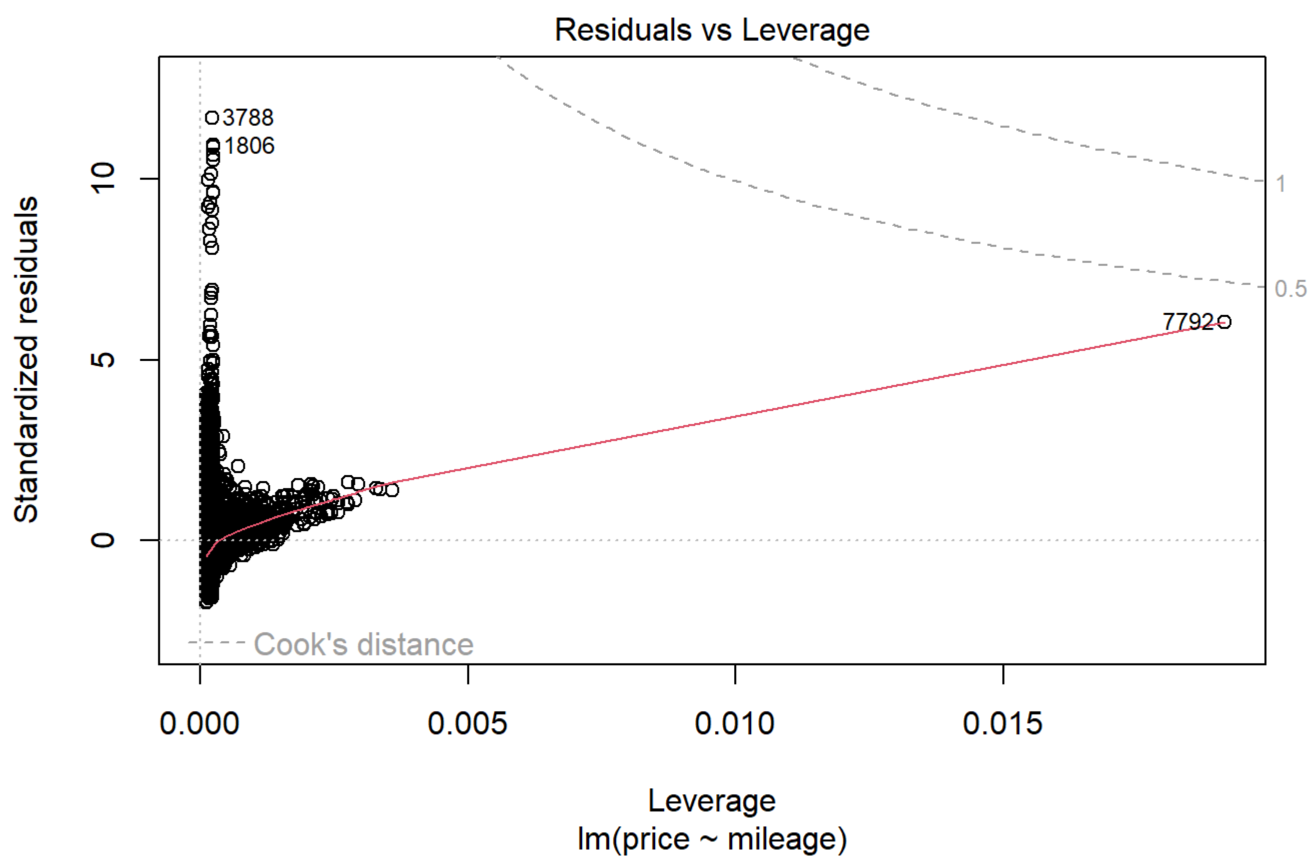
The first plot is meant to show if the model follows a non linear pattern. It seems to be clumped up at the end and have a somewhat curve so it could possibly not be a linear model. The second plot is meant to show if the residuals are normally distributed. In the plot it seems to follow a straight line for a while and then curve up heavily, which is concerning. The third plot is meant to check for equal variance, which it seems to match quite well. The last plot shows that there are some data points that influence the results, and would be changed if removed.

```
plot(model1)
```


Residuals vs Fitted
Residuals
1e+05
5e+04
0e+00
3788
-60000
-40000
-20000
0
20000
Fitted values
lm(price ~ mileage)


Normal Q-Q
Standardized residuals
10
5
0
3788
-4
-2
0
2
4
Theoretical Quantiles
lm(price ~ mileage)


Scale-Location
3788
1806
√|Standardized residuals|
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
-60000
-40000
-20000
0
20000
Fitted values
lm(price ~ mileage)

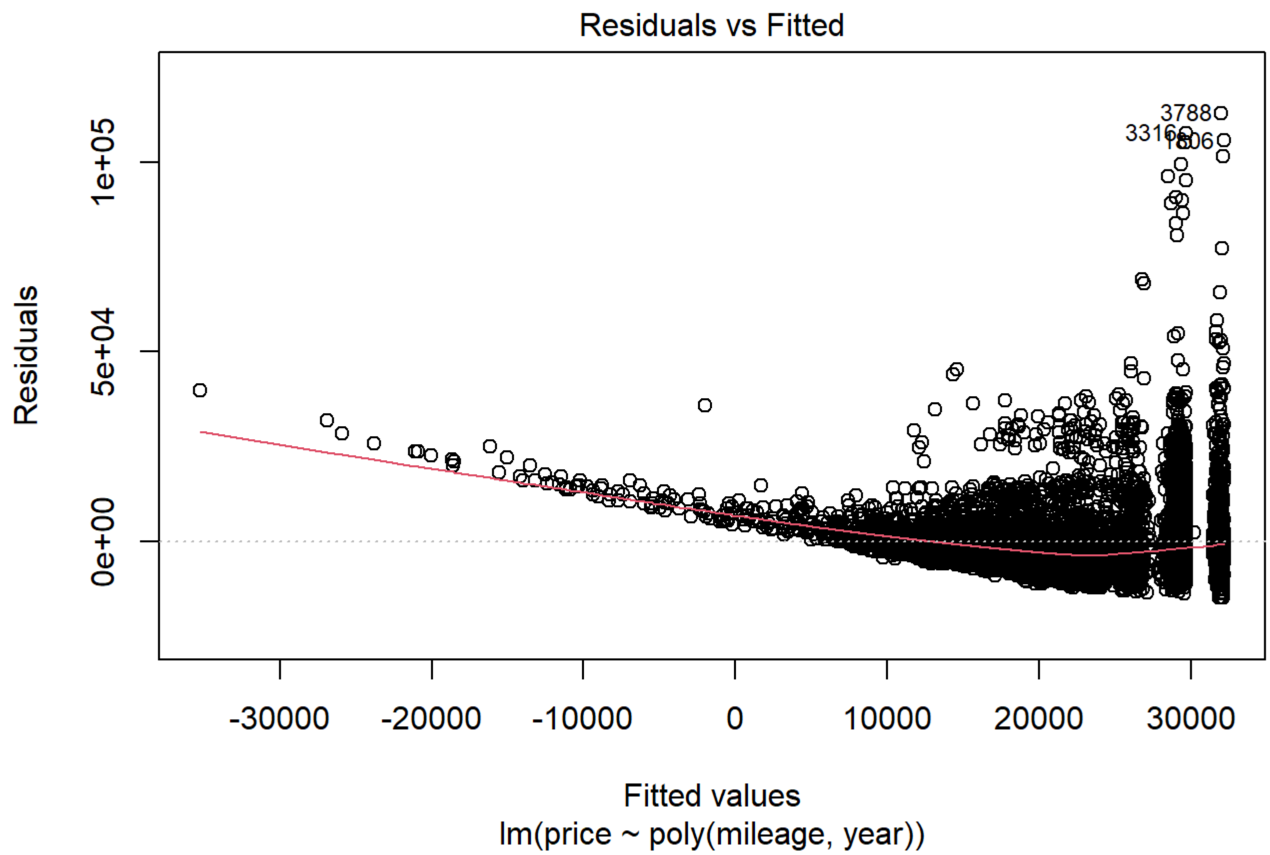

Residuals vs Leverage
3788
1806
7792
1
0.5
Standardized residuals
0
5
10
Cook's distance
0.000
0.005
0.010
0.015
Leverage
lm(price ~ mileage)

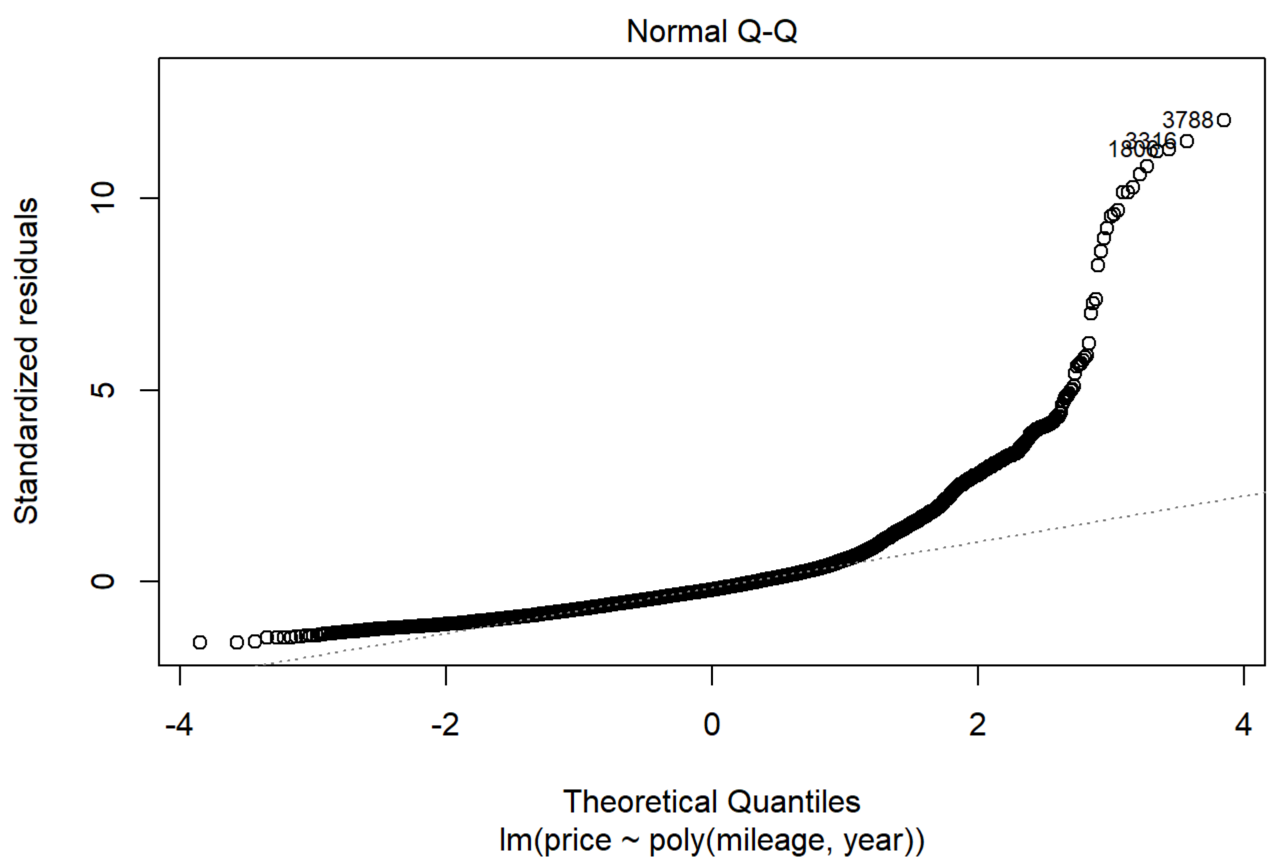
Multiple Linear Model and Residual plots

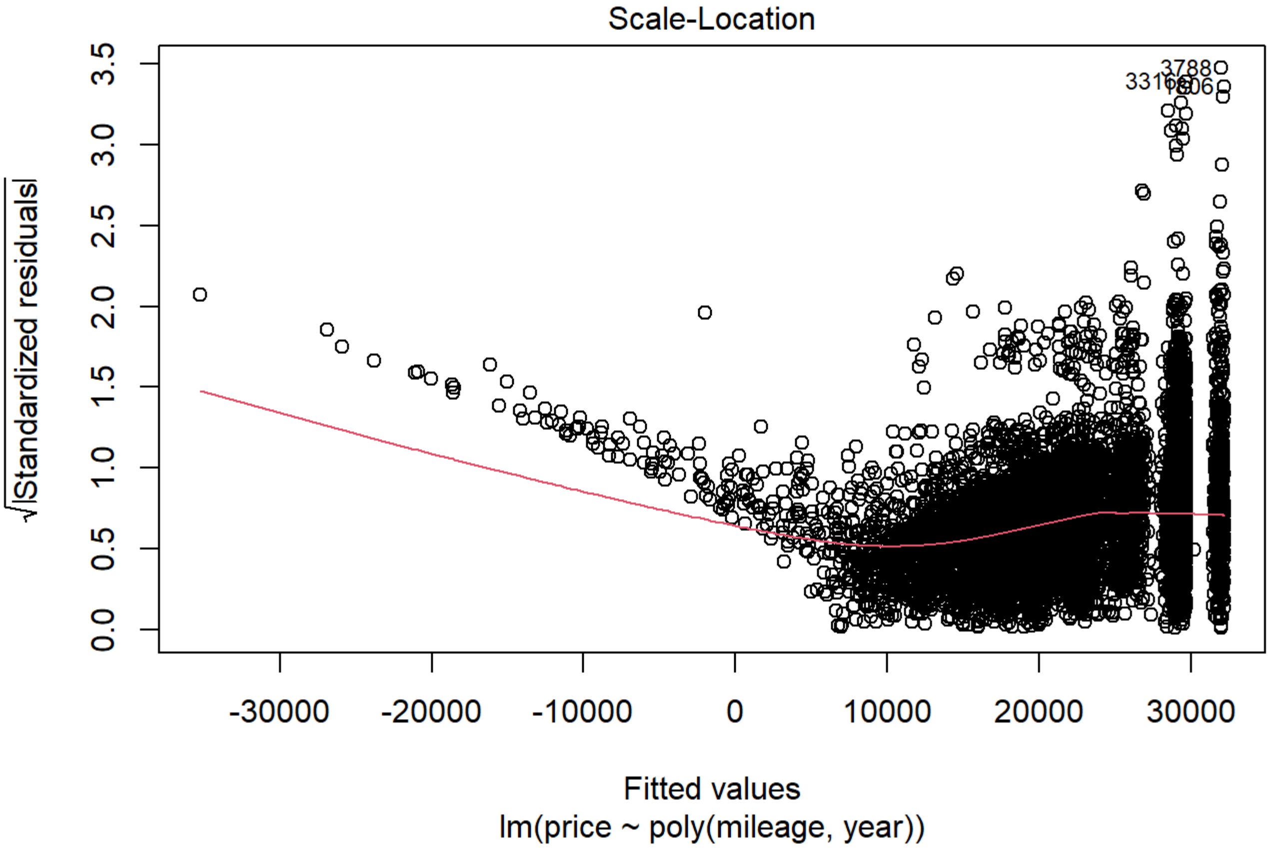
```
model2 <- lm(price ~ poly(mileage, year), data=train)
summary(model2)
```

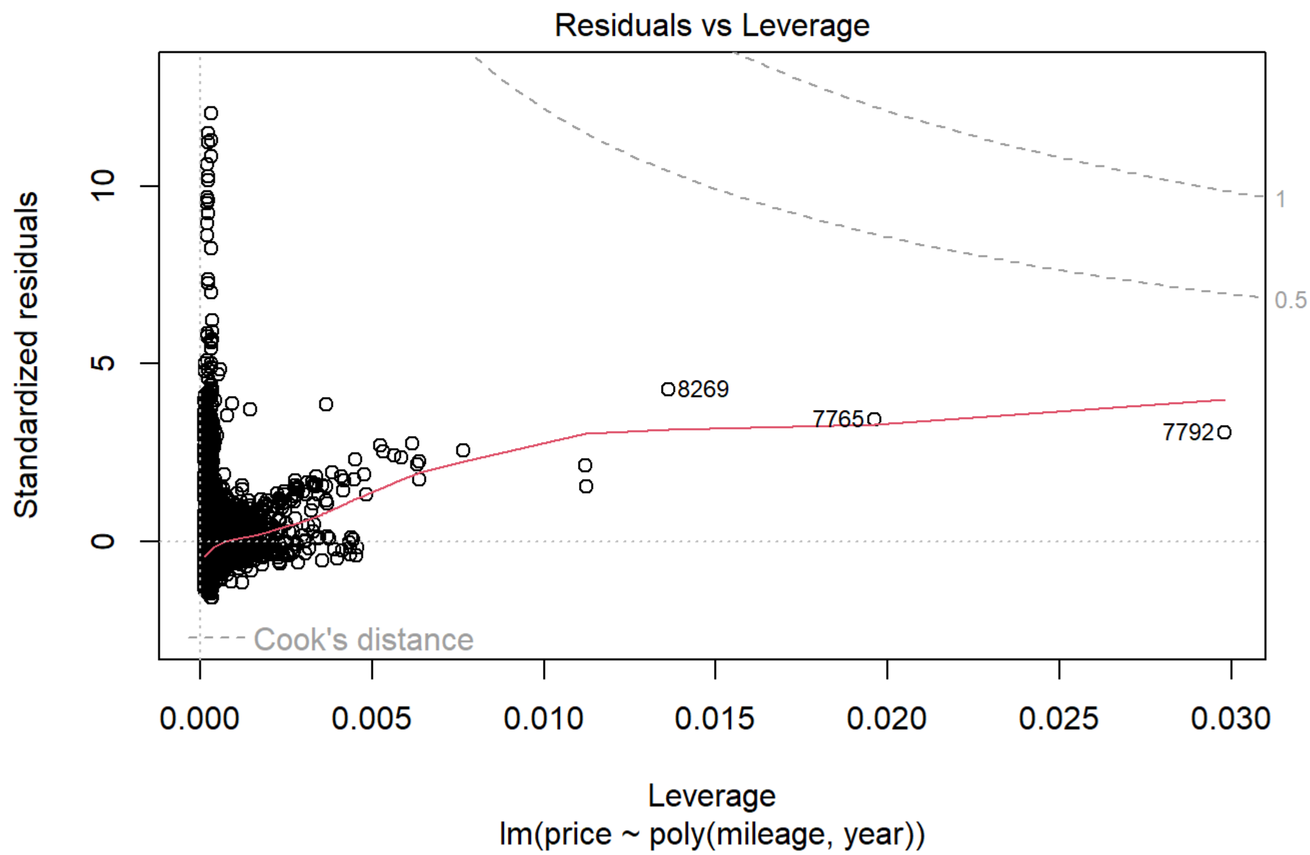
```
##
## Call:
## lm(formula = price ~ poly(mileage, year), data = train)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -15009  -5293  -2040   2240 112994
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)               22872        102  224.28   <2e-16 ***
## poly(mileage, year)1.0  -190803      15326  -12.45   <2e-16 ***
## poly(mileage, year)0.1   490174      15326   31.98   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9378 on 8453 degrees of freedom
## Multiple R-squared:  0.3635, Adjusted R-squared:  0.3634
## F-statistic:  2414 on 2 and 8453 DF,  p-value: < 2.2e-16
```

```
plot(model2)
```


Residuals vs Fitted
Residuals
1e+05
5e+04
0e+00
3788
3316
1806
-30000
-20000
-10000
0
10000
20000
30000
Fitted values
lm(price ~ poly(mileage, year))


Normal Q-Q
Standardized residuals
10
5
0
3788
3316
1806
-4
-2
0
2
4
Theoretical Quantiles
lm(price ~ poly(mileage, year))


Scale-Location
√|Standardized residuals|
Fitted values
lm(price ~ poly(mileage, year))
3788
331
06
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
-30000
-20000
-10000
0
10000
20000
30000

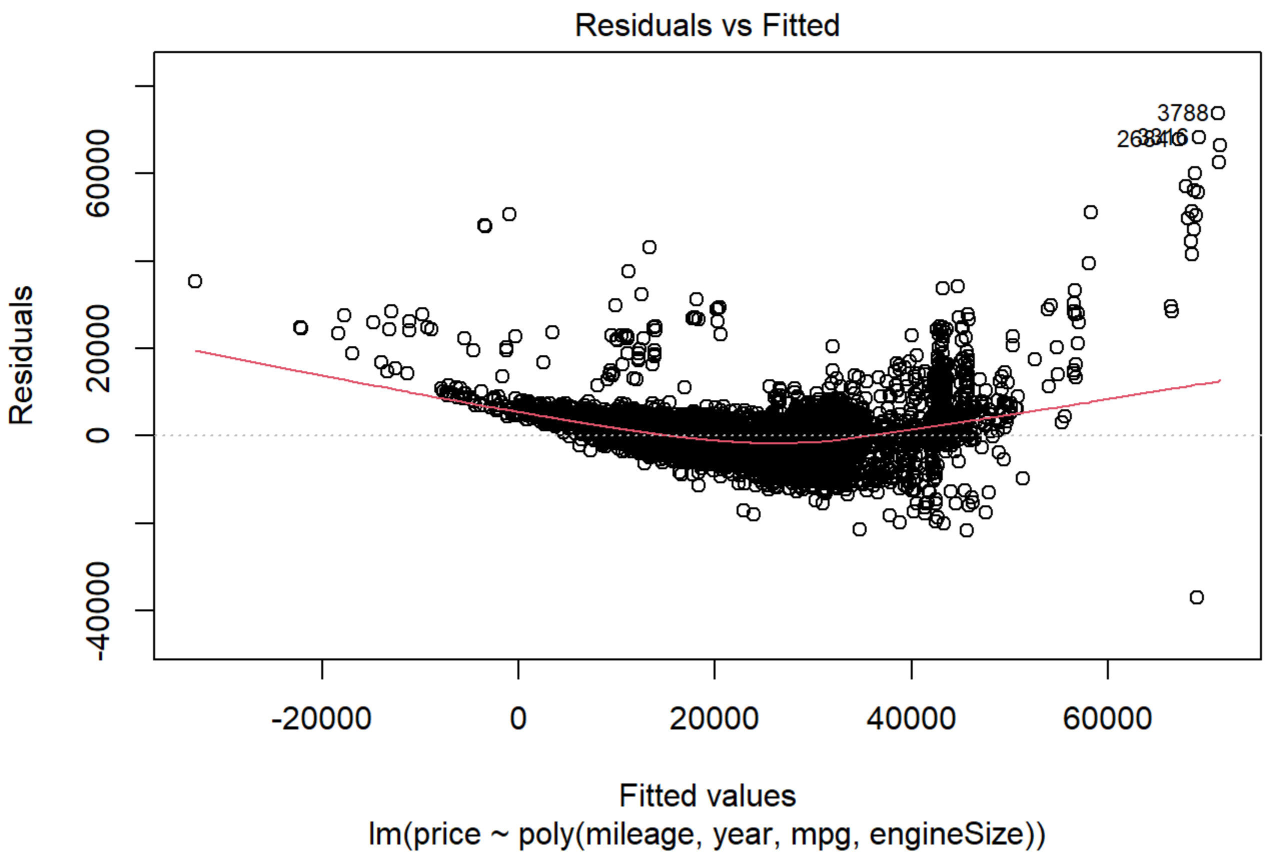

Residuals vs Leverage
Standardized residuals
Leverage
lm(price ~ poly(mileage, year))
8269
7765
7792
Cook's distance
1
0.5
0
5
10
0.000
0.005
0.010
0.015
0.020
0.025
0.030

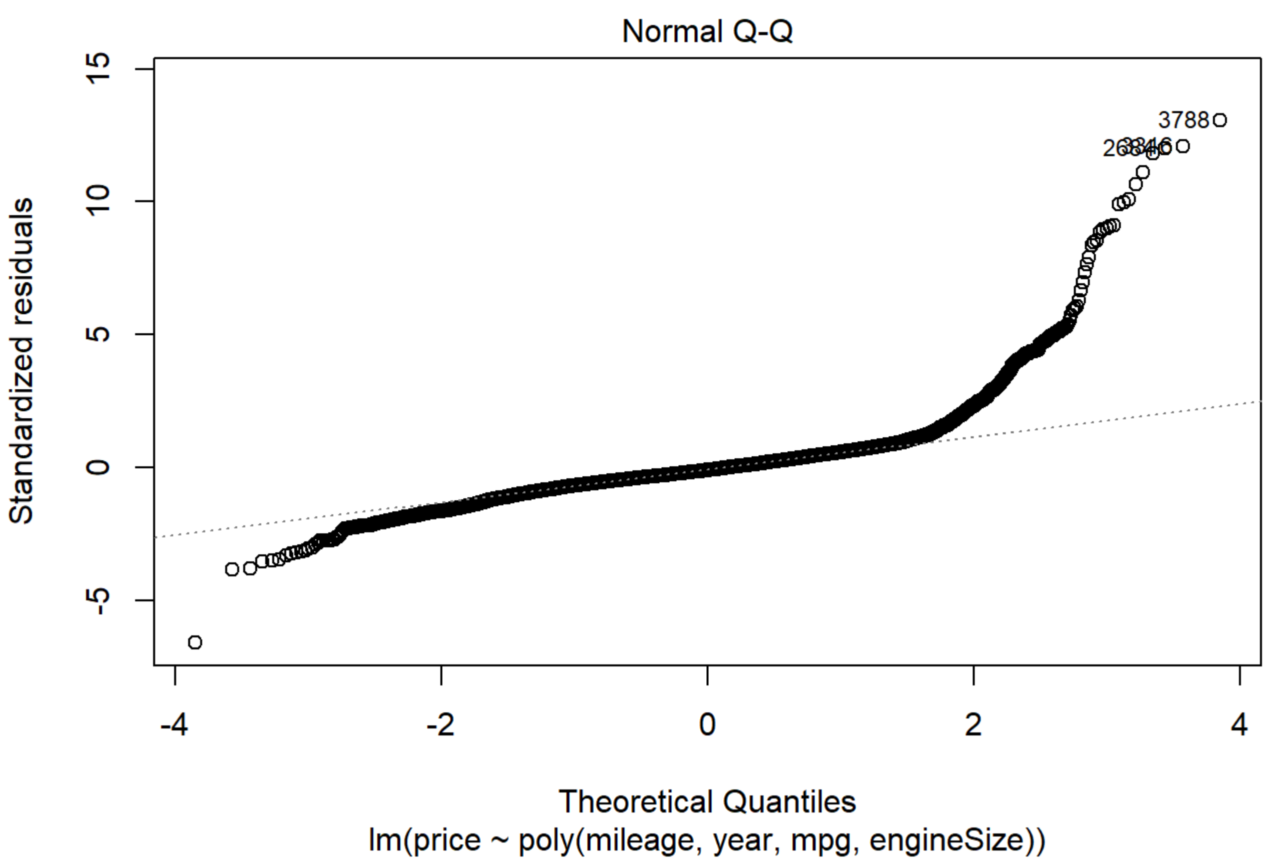
Third Model and Residual plots

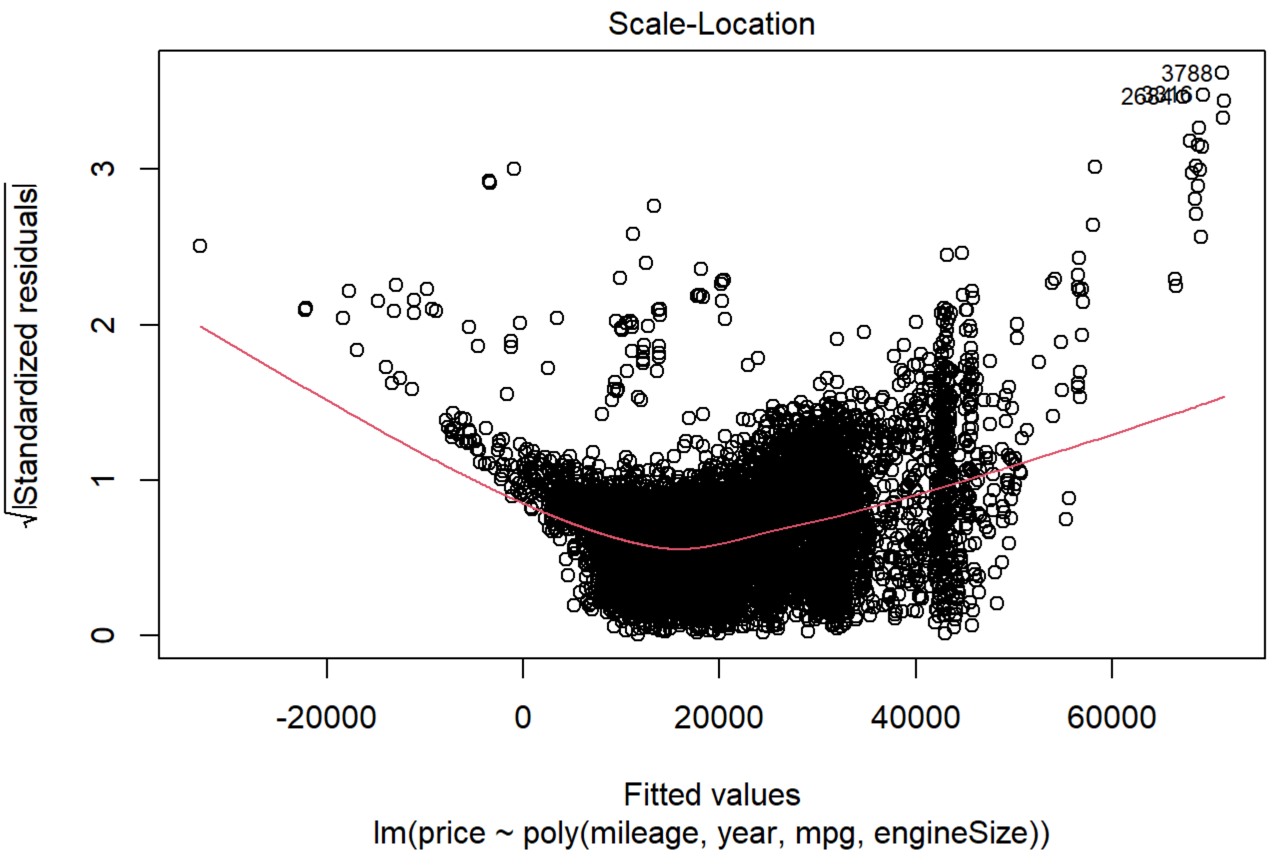
```
model3 <- lm(price ~ poly(mileage, year, mpg, engineSize),data=train)
summary(model3)
```

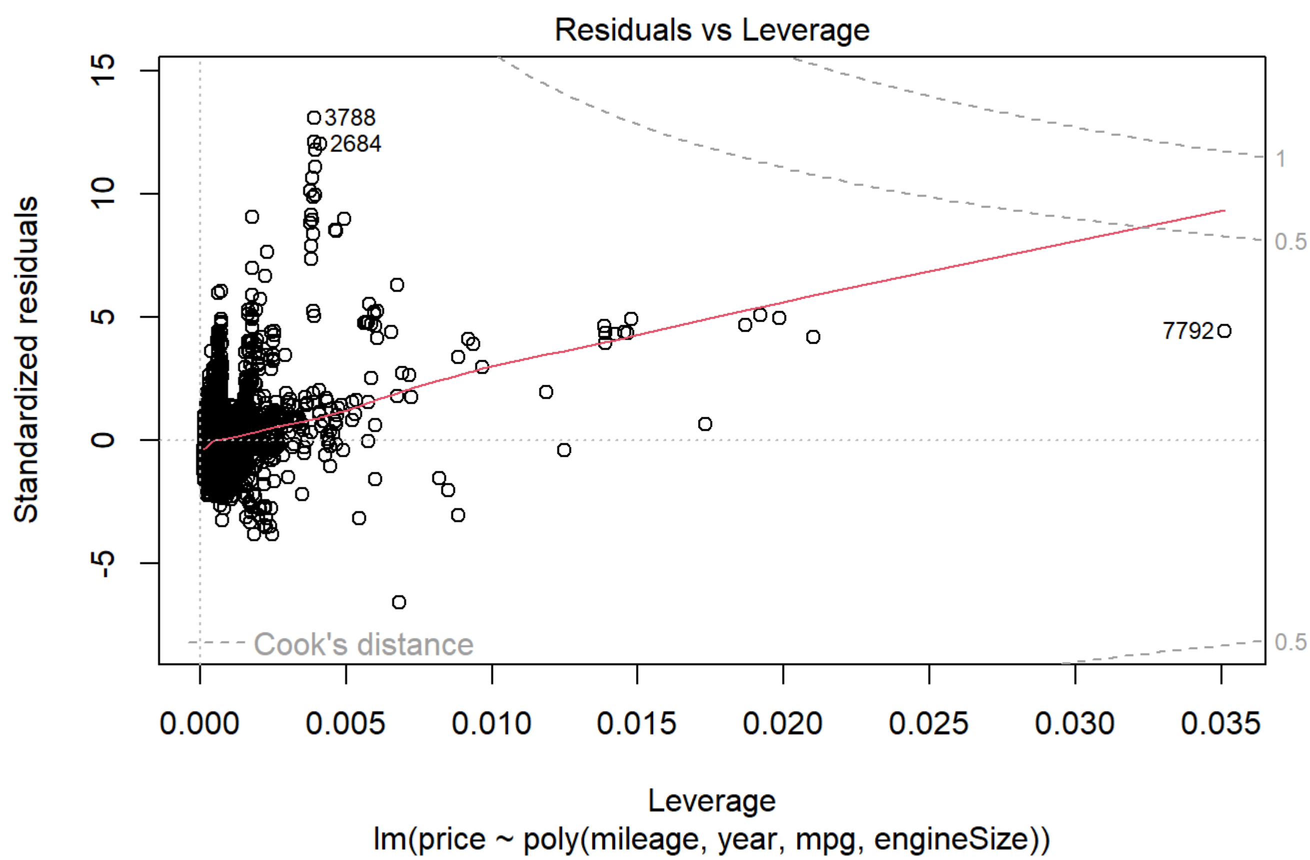
```
##
## Call:
## lm(formula = price ~ poly(mileage, year, mpg, engineSize), data = train)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -37102  -2683   -503   2012  73777
##
## Coefficients:
##                                              Estimate Std. Error t value
## (Intercept)                                  22872.02      61.46  372.17
## poly(mileage, year, mpg, engineSize)1.0.0.0 -200140.63    9595.89  -20.86
## poly(mileage, year, mpg, engineSize)0.1.0.0  422184.69    9261.34   45.59
## poly(mileage, year, mpg, engineSize)0.0.1.0 -201925.11    6865.26  -29.41
## poly(mileage, year, mpg, engineSize)0.0.0.1  589642.46    6289.80   93.75
##                                             Pr(>|t|)
## (Intercept)                                   <2e-16 ***
## poly(mileage, year, mpg, engineSize)1.0.0.0   <2e-16 ***
## poly(mileage, year, mpg, engineSize)0.1.0.0   <2e-16 ***
## poly(mileage, year, mpg, engineSize)0.0.1.0   <2e-16 ***
## poly(mileage, year, mpg, engineSize)0.0.0.1   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5651 on 8451 degrees of freedom
## Multiple R-squared:  0.7689, Adjusted R-squared:  0.7688
## F-statistic:  7030 on 4 and 8451 DF,  p-value: < 2.2e-16
```

```
plot(model3)
```


Residuals vs Fitted
Residuals
Fitted values
lm(price ~ poly(mileage, year, mpg, engineSize))
3788
26846
3316
-40000
0
20000
60000
-20000
40000


Normal Q-Q
Standardized residuals
Theoretical Quantiles
lm(price ~ poly(mileage, year, mpg, engineSize))
3788
26846
3316
15
10
5
0
-5
-4
-2
2
4


Scale-Location
3788
2684
3816
√|Standardized residuals|
0
1
2
3
-20000
0
20000
40000
60000
Fitted values
lm(price ~ poly(mileage, year, mpg, engineSize))


Residuals vs Leverage
3788
2684
7792
Standardized residuals
-5
0
5
10
15
1
0.5
0.5
Cook's distance
0.000
0.005
0.010
0.015
0.020
0.025
0.030
0.035
Leverage
lm(price ~ poly(mileage, year, mpg, engineSize))

As we added more variables into the regression, the R squared value increased. This is an indicator that the price can be more accurately predicted by multiple of the variables. I think this is more accurate to what should be expected, as when someone is trying to buy a car they often take all of the factors into consideration.

Test data using models

```
pred1 <- predict(model1, newdata=test)
correlation1 <- cor(pred1, test$price)
model_summ1 <- summary(model1)
mse1 <- mean((pred1 - test$price)^2)
print(correlation1)
```

```
## [1] 0.5358395
```

```
print(mse1)
```

```
## [1] 95392168
```

```
pred2 <- predict(model2, newdata=test)
correlation2 <- cor(pred2, test$price)
mse2 <- mean((pred2 - test$price)^2)
print(correlation2)
```

```
## [1] 0.6019817
```

```
print(mse2)
```

```
## [1] 85367279
```

```
pred3 <- predict(model3, newdata=test)
correlation3 <- cor(pred3, test$price)
mse3 <- mean((pred3 - test$price)^2)
print(correlation3)
```

```
## [1] 0.8845703
```

```
print(mse3)
```

```
## [1] 29104745
```

These results show a relatively low correlation that gets better as we move through the models and include more variables. Although the MSE is very high in each of the cases the correlation gets better at the end with a value of 0.88 roughly and the MSE decreases each time. This is further proof of what the training data results indicated that the price can be better predicted using multiple other variables.