Group Project 1

Benedikt Graf; Wong Jing Hei (Koby); Qiumeng He

11-18-2020

Contents

Introudction

With used car prices currently at an all time high due to a confluence of factors such as decreased vehicle production and safety concerns with public transport (Boudette 2020; Domonoske 2020), we wanted to learn more about the used car market and how the price of a used car is determined. The factors that influence the prices of used vehicles have been addressed both in the popular press (see D'Allegro 2020, for example) as well as in the academic literature. For instance, using a data set of 370,000 used cars sold in Germany, Pal et al. (2018) found that the car's odometer value, brand, and vehicle type are among the most important features. Erdem and Şentürk (2009) found that the "production year of the car is one of the principal characteristics influencing used car prices" (p.145). It is our hope to contribute to this area of research by specifying a model that relates a car's sale price to the car's feature such as its odometer value, engine capacity, et cetera.

The Data

For our analysis we turn to a data set called "Used-cars-catalog" which we retrieved from Kaggle. The data were webscraped from various websites by Kirill Lepchenkov in December of 2019 in Belarus, so the data are quite recent. The data set contains information on roughly 38,000 used cars including their odometer value, color, and about 20 other variables.
We decided to further explore the following variables:

- Price (price_usd)
  - The price of the car as listed in the catalog in USD.
- Odometer Value (odometer_value)
  - The car's current odometer reading in kilometers.
- Engine Capacity (engine_capacity)
  - The capacity of the engine in liters.
- Year Produced (year_produced, age)
  - The year the car was produced.
  - We created a new variable (age) by subtracting year_produced from the date of the webscraping (2019).
- Transmission (transmission, automatic)
  - 33 percent of the cars in the data have an automatic transmission, so we added an indicator which equals 1 if the car has automatic transmission and 0 if the car has manual transmission.
- Drivetrain (drivetrain, rear, all)
  - While most (72%) of the cars have front-wheel drive, some (14%) have rear-wheel drive and others (14%) have four-wheel drive.
  - We included two dummy variables to indicate whether a car has rear- or four-wheel drive with front-wheel drive being the baseline.
- Engine Fuel (engine_type, diesel)
  - The majority of the cars in the data set (~67%) run on gasoline, but some (33% run on diesel), so we added an indicator (diesel) which equals 1 if the car runs on diesel and 0 otherwise.
- Propane Equipped (engine_has_gas, propane)
  - A few (3%) of the cars have a propane tank and tubing, so we added an indicator (propane) which equals 1 if the car has propane adaptations and 0 if not.

- Manufacturer Name (manufacturer_name, luxury)
  - The cars in the data were built by 55 different manufacturers.
  - We decided to create an indicator variable based on these called "luxury" which equals 1 if the car brand is considered luxury and 0 i f not. We followed this (https://cars.usnews.com/cars-trucks/best-luxury-car-brands) to classify the brands.
  - We considered adding indicator variables for a car's country of origin (Germany, Italy, etc.), but decided against this because countries produce a range of brands in different price classes (e.g. for Germany: VW, Opel, Mercedes, BMW, Porsche)
- Body Type (body_type, sports, minivan, suv, pickup)
  - There are a number of different body types in our data set. We decided to include indicator variables for each different body type and keep sedan, universal, and hatchback as the reference group.
- Color (color, not_dark_color, not_popular_color)
  - The cars in this data set have 12 unique colors.
  - We did not want to include a unique indicator for each color so we tested two indicators:
  - An indicator for when a car's color is not dark (green, orange, red, violet, white, yellow, other), (not_dark_color).
  - An indicator for when a car has a non-popular color (any color other than blue, black, silver, white), (not_popular_color).
- State (state, new)
  - A few (~1%) cars in the data are actually new and not used. Thus, we added an indicator (new) which equals 1 if the car is new and 0 if not.
- Warranty (has_warranty, warranty)
  - A small number (~1%) of the cars has a warranty, so we included an indicator (warranty) which equals 1 if the car has a warranty and 0 if not.
- Exchange (is_exchangeable, exchange)
  - 35 percent of the car owners are open to exchange their car for another car with little or no additional payment. We added an indicator (exchange) which equals 1 if the current owner is open to exchange the vehicle and 0 if not.
- Number of Photos (number_of_photos)
  - The number of photos the car has on the website.
- Likes (up_counter)
  - The number of "ups" (likes) the car has on the website.
- Duration Listed (duration_listed)
  - The number of days the car has been listed on the website

Note: We were not able to use some of the original variables (Features 0-9), because the data set's author does not deem them consistent.

Importing the Data

```r
cars_raw <- read.csv("cars.csv")
```

Summary Statistics

```
## Number of rows:    38531
```

```
## Number of columns: 30
```

```
# Checking for Missing Values
sum(is.na(cars_raw))
```

```
## [1] 10
```

```
colSums(is.na(cars_raw))
```

```
## manufacturer_name        model_name      transmission             color
##                 0                 0                 0                 0
##    odometer_value     year_produced       engine_fuel    engine_has_gas
##                 0                 0                 0                 0
##       engine_type   engine_capacity         body_type      has_warranty
##                 0                10                 0                 0
##             state        drivetrain         price_usd   is_exchangeable
##                 0                 0                 0                 0
##   location_region  number_of_photos        up_counter         feature_0
##                 0                 0                 0                 0
##         feature_1         feature_2         feature_3         feature_4
##                 0                 0                 0                 0
##         feature_5         feature_6         feature_7         feature_8
##                 0                 0                 0                 0
##         feature_9   duration_listed
##                 0                 0
```

```
cars_raw <- cars_raw %>% drop_na()
```

There are 10 missing values in our data, which we dropped as our data set is so large.

```
##                         mean        sd median    min     max     range  skew
## manufacturer_name*     31.66     17.32     37    1.0      55      54.0 -0.36
## model_name*           572.31    330.58    584    1.0    1116    1115.0 -0.03
## transmission*           1.67      0.47      2    1.0       2       1.0 -0.70
## color*                  5.48      3.59      5    1.0      12      11.0  0.12
## odometer_value     248910.07 136059.50 250000    0.0 1000000 1000000.0  1.17
## year_produced        2002.94      8.06   2003 1942.0    2019      77.0 -0.39
## engine_fuel*            2.31      0.96      3    1.0       5       4.0 -0.48
## engine_has_gas*         1.03      0.18      1    1.0       2       1.0  5.06
## engine_type*            1.67      0.47      2    1.0       2       1.0 -0.70
## engine_capacity         2.06      0.67      2    0.2       8       7.8  2.05
## body_type*              7.79      2.93      9    1.0      12      11.0 -0.73
## has_warranty*           1.01      0.11      1    1.0       2       1.0  9.10
## state*                  2.97      0.22      3    1.0       3       2.0 -7.75
## drivetrain*             2.00      0.53      2    1.0       3       2.0  0.00
## price_usd            6637.16   6425.20   4800    1.0   50000   49999.0  2.24
## is_exchangeable*        1.35      0.48      1    1.0       2       1.0  0.62
```

```
## location_region*         4.30      1.38      5    1.0       6       5.0 -1.27
## number_of_photos         9.65      6.09      8    1.0      86      85.0  1.60
## up_counter              16.31     43.29      5    1.0    1861    1860.0 13.32
## feature_0*               1.23      0.42      1    1.0       2       1.0  1.29
## feature_1*               1.61      0.49      2    1.0       2       1.0 -0.44
## feature_2*               1.22      0.42      1    1.0       2       1.0  1.33
## feature_3*               1.28      0.45      1    1.0       2       1.0  1.00
## feature_4*               1.24      0.43      1    1.0       2       1.0  1.21
## feature_5*               1.36      0.48      1    1.0       2       1.0  0.60
## feature_6*               1.17      0.38      1    1.0       2       1.0  1.75
## feature_7*               1.26      0.44      1    1.0       2       1.0  1.07
## feature_8*               1.42      0.49      1    1.0       2       1.0  0.34
## feature_9*               1.58      0.49      2    1.0       2       1.0 -0.32
## duration_listed         80.58    112.84     59    0.0    2232    2232.0  6.82
##                    kurtosis
## manufacturer_name*    -1.22
## model_name*           -1.31
## transmission*         -1.51
## color*                -1.47
## odometer_value         4.90
## year_produced          0.65
## engine_fuel*          -1.29
## engine_has_gas*       23.63
## engine_type*          -1.51
## engine_capacity        6.37
## body_type*            -0.87
## has_warranty*         80.80
## state*                61.74
## drivetrain*            0.57
## price_usd              7.28
## is_exchangeable*      -1.62
## location_region*       0.37
## number_of_photos       4.96
## up_counter           308.48
## feature_0*            -0.33
## feature_1*            -1.81
## feature_2*            -0.24
## feature_3*            -0.99
## feature_4*            -0.54
## feature_5*            -1.64
## feature_6*             1.06
## feature_7*            -0.85
## feature_8*            -1.88
## feature_9*            -1.90
## duration_listed       76.87
```

Creating New Variables

Part 1: Variable Selection

```r
# Selecting relevant variables
cars <- subset(cars_raw, select = c(price_usd, odometer_value, engine_capacity,
```

```
                                          number_of_photos, up_counter,
                                         duration_listed, automatic, diesel, new,
                                         propane, luxury, age, drivetrain_rear,
                                         drivetrain_all, exchange, warranty,
                                         not_pop_color, sports, minivan,
                                         suv, pickup))
```

Boruta Algorithm

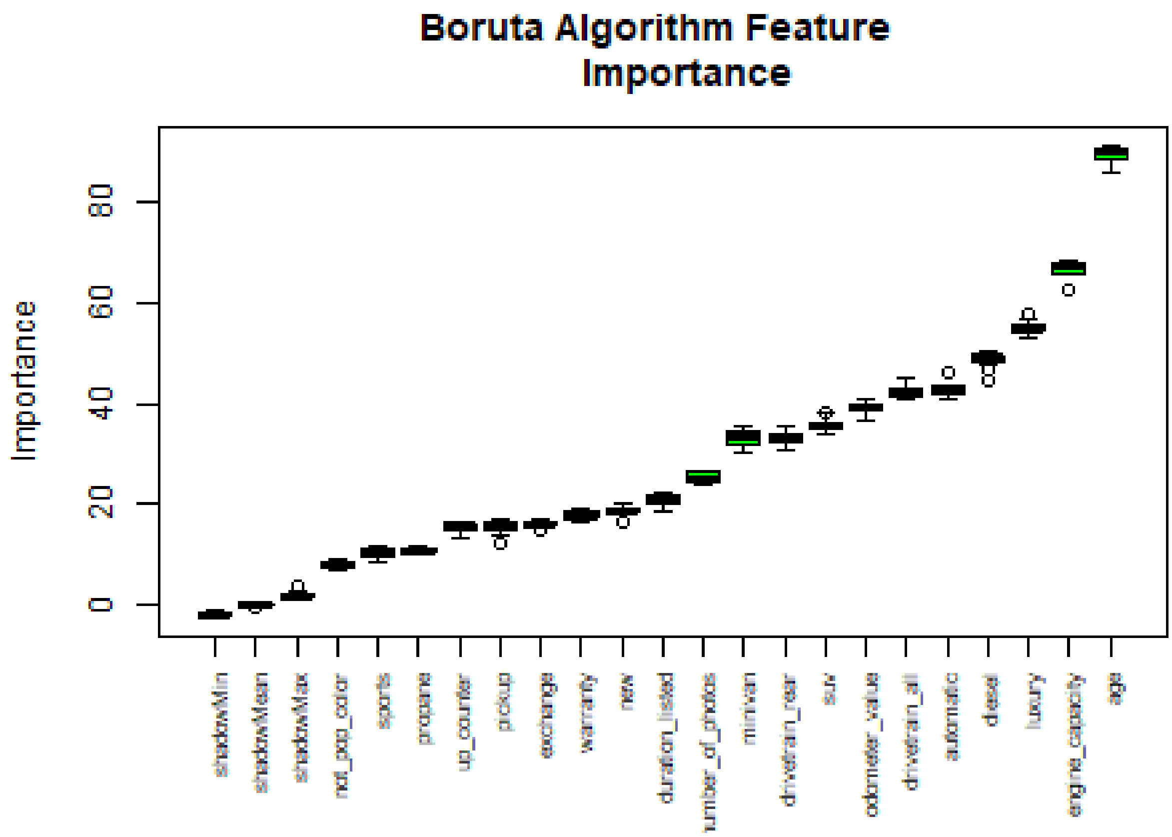
```
set.seed(123)
boruta.train <- Boruta(price_usd~., data = cars, doTrace = 2)
```

```
##  1. run of importance source...

##  2. run of importance source...

##  3. run of importance source...

##  4. run of importance source...

##  5. run of importance source...

##  6. run of importance source...

##  7. run of importance source...

##  8. run of importance source...

##  9. run of importance source...

##  10. run of importance source...

##  11. run of importance source...

## After 11 iterations, +5.7 mins:

##  confirmed 20 attributes: age, automatic, diesel, drivetrain_all, drivetrain_rear and 15 more;

##  no more attributes left.
```

```
print(boruta.train)
```

```
## Boruta performed 11 iterations in 5.655568 mins.
##  20 attributes confirmed important: age, automatic, diesel,
## drivetrain_all, drivetrain_rear and 15 more;
##  No attributes deemed unimportant.
```

```r
# Plotting the results
plot(boruta.train, xlab = "", xaxt = "n", main="Boruta Algorithm Feature
     Importance")
lz<-lapply(1:ncol(boruta.train$ImpHistory),function(i)
boruta.train$ImpHistory[is.finite(boruta.train$ImpHistory[,i]),i])
names(lz) <- colnames(boruta.train$ImpHistory)
Labels <- sort(sapply(lz,median))
axis(side = 1,las=2,labels = names(Labels),
at = 1:ncol(boruta.train$ImpHistory), cex.axis = 0.7)
```



From this graph we can easily identify the most important variables. Age, engine capacity, odometer reading, and the number of photos are among the most important continuous variables. Luxury, diesel, automatic, drivetrain_all, and suv are among the most important indicator variables.

Mallows CP

```r
full_model<-regsubsets(price_usd~., data=cars, nbest=1, nvmax=10)
info <- summary(full_model)
cbind(info$which, round(cbind(rsq=info$rsq, adjr2=info$adjr2, cp=info$cp, bic=info$bic, rss=info$rss), 3
```

```
##    (Intercept) odometer_value engine_capacity number_of_photos up_counter
## 1            1              0               0                0          0
## 2            1              0               0                0          0
## 3            1              0               0                0          0
```

```
## 4              1              0               0                0          0
## 5              1              0               1                0          0
## 6              1              1               1                0          0
## 7              1              1               1                0          0
## 8              1              1               1                1          0
## 9              1              1               1                1          0
## 10             1              1               1                1          0
##    duration_listed automatic diesel new propane luxury age drivetrain_rear
## 1                0         0      0   0       0      0   1               0
## 2                0         0      0   0       0      0   1               0
## 3                0         0      0   0       0      1   1               0
## 4                0         0      0   1       0      1   1               0
## 5                0         0      0   1       0      1   1               0
## 6                0         0      0   1       0      1   1               0
## 7                0         0      1   1       0      1   1               0
## 8                0         0      1   1       0      1   1               0
## 9                0         0      1   1       0      1   1               0
## 10               0         0      1   1       0      1   1               0
##    drivetrain_all exchange warranty not_pop_color sports minivan suv pickup
## 1               0        0        0             0      0       0   0      0
## 2               1        0        0             0      0       0   0      0
## 3               1        0        0             0      0       0   0      0
## 4               1        0        0             0      0       0   0      0
## 5               0        0        0             0      0       0   1      0
## 6               1        0        0             0      0       0   0      0
## 7               1        0        0             0      0       0   0      0
## 8               0        0        0             0      0       0   1      0
## 9               1        0        0             0      0       0   1      0
## 10              1        0        0             0      0       1   1      0
##      rsq adjr2        cp       bic          rss
## 1  0.498 0.498 27130.516 -26498.57 798859691888
## 2  0.584 0.584 15785.744 -33797.82 660781682480
## 3  0.620 0.620 11188.370 -37196.58 604812239439
## 4  0.648 0.648  7485.615 -40170.04 559729353178
## 5  0.664 0.664  5431.652 -41920.97 534710494013
## 6  0.673 0.672  4285.439 -42930.39 520737977612
## 7  0.680 0.680  3264.315 -43852.01 508287657600
## 8  0.687 0.687  2400.655 -44648.18 497753505627
## 9  0.692 0.692  1724.658 -45281.48 489503012025
## 10 0.695 0.695  1328.414 -45654.19 484656810618
```

Using Mallow's CP gives us similar results. The two notable differences are that the indicator for a new car is deemed important and that the indicator for automatic is deemed unimportant here.

Identifying Predictors

We combined the information from the Boruta Algorithm and Mallow's CP to specify the following basic model (before any transformations):

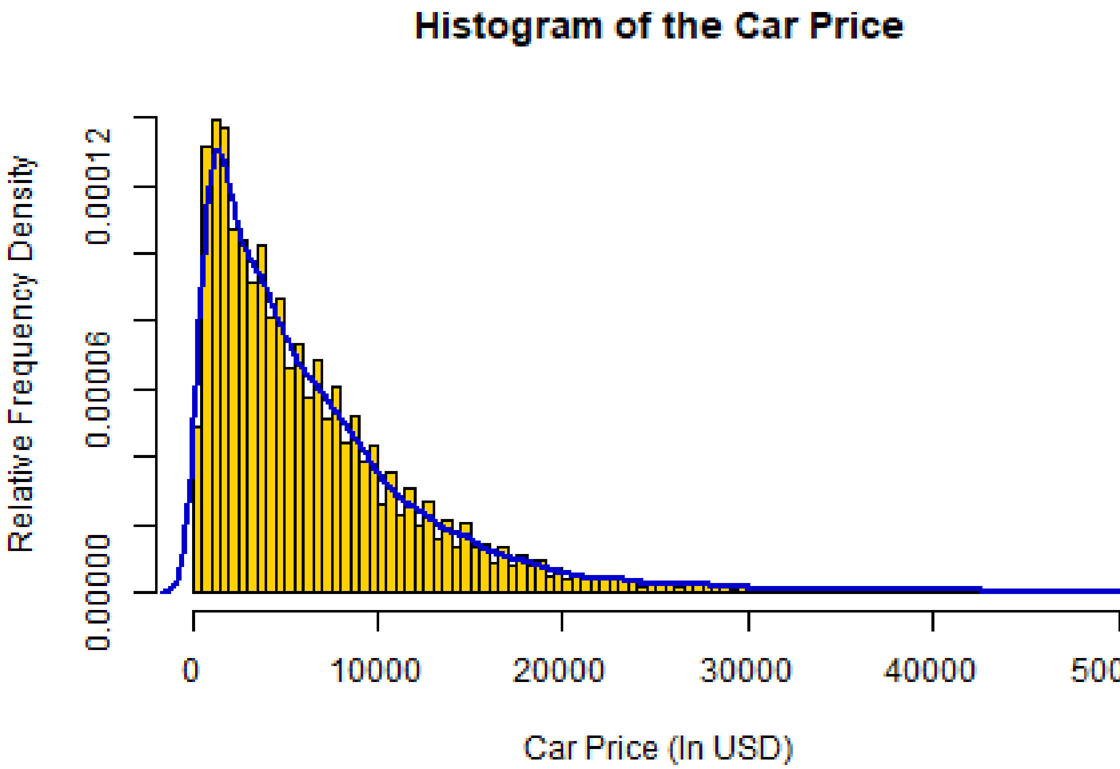
$$SalePrice = \beta_0 \ + \ \beta_1 OdometerValue \ + \ \beta_2 EngineCapacity \ + \ \beta_3 Age \ + \ \beta_4 NumberOfPhotos$$
$$+\beta_5 Diesel \ + \ \beta_6 Automatic \ + \ \beta_7 DrivetrainAll \ + \ \beta_8 SUV \ + \ \beta_9 Luxury \ + \ \beta_{10} New \ + \ \epsilon$$

We removed many of the indicators that have a low variation (warranty, propane) and those deemed unimportant (color, exchange). There are five different indicators for the car's body type (suv, sports, minivan, etc.). We originally wanted to include them all, but based on these tests we decided to only include an indicator for whether a car is a SUV or not. Interestingly, color was not deemed very important in either test with both color specifications (not_dark_color, not_pop_color). We kept most of the continuous variables with the exceptions of up-counter and duration listed, which are both variables unique to the online marketplace. The importance of the number of photos associated with a car's online listing initially surprised us, but we found that the importance of product photos in online marketplaces has been demonstrated before (Li et al. 2014). In an online marketplace pictures can help reduce the information asymmetry between buyer and seller, thereby influencing the price of the car.

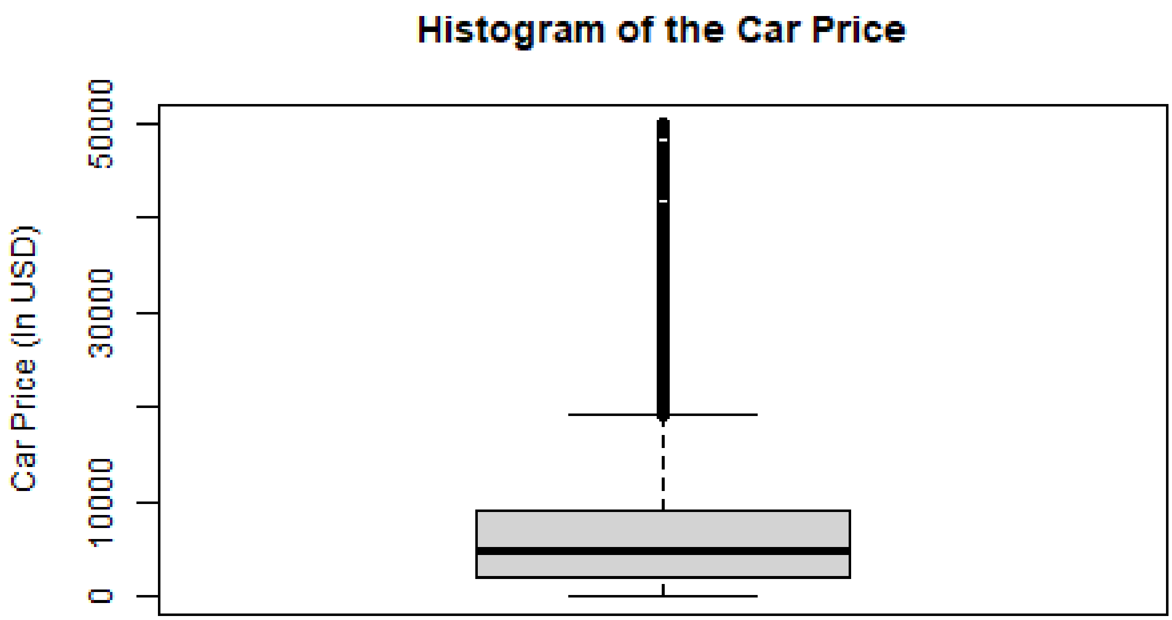
Part 2: Descriptive Analysis

Univariate Analysis

Note: We included the density curves here.

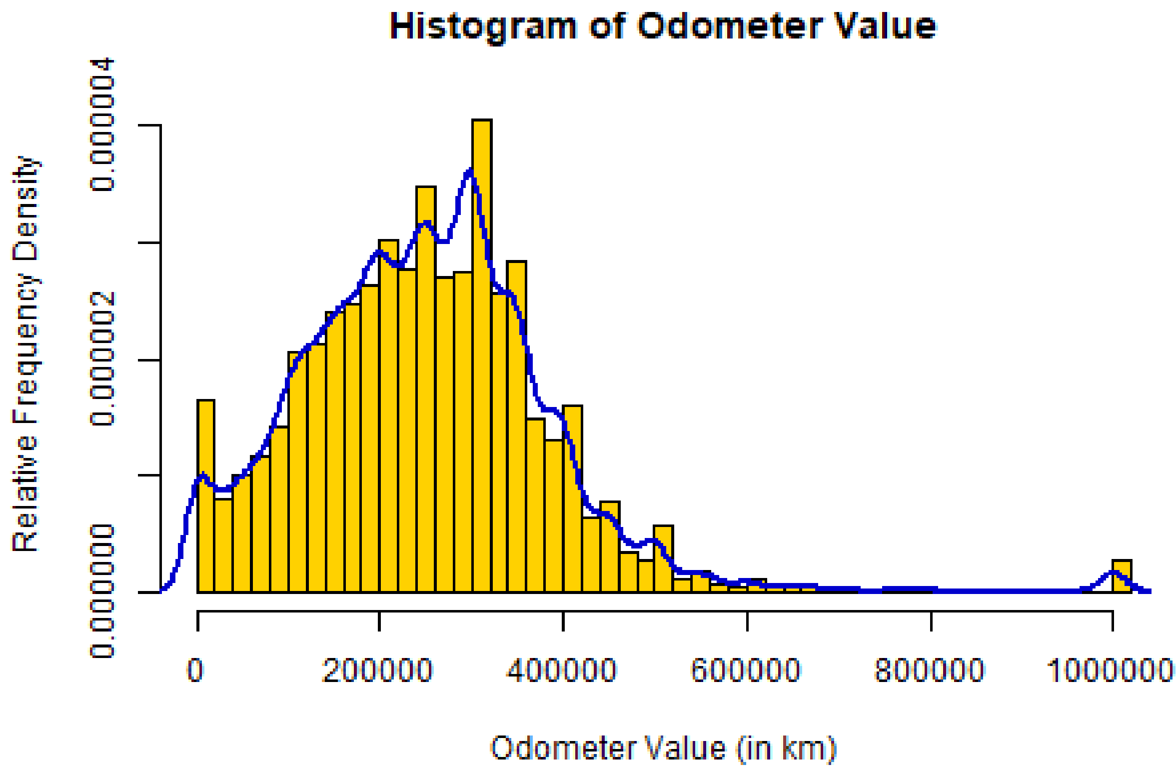


Y Variable: Sale Price

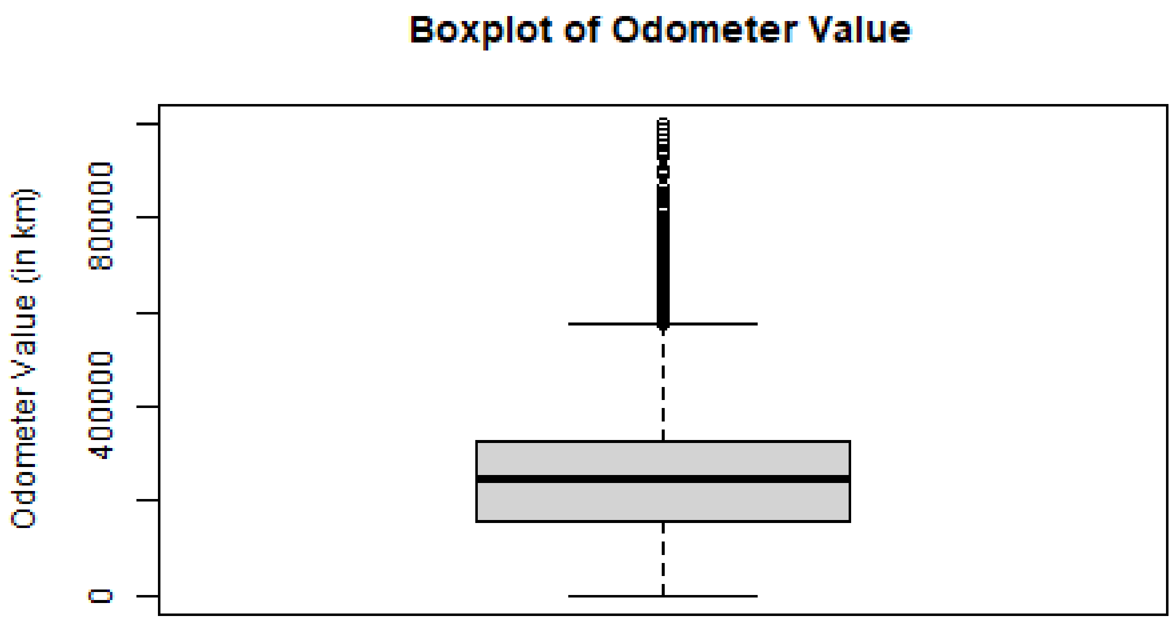


```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##       1    2100    4800    6637    8950   50000 
```

Heavy positive skew of sale price in the data set is observed. Potential transformation to linearity is needed, which will be explored in part c.

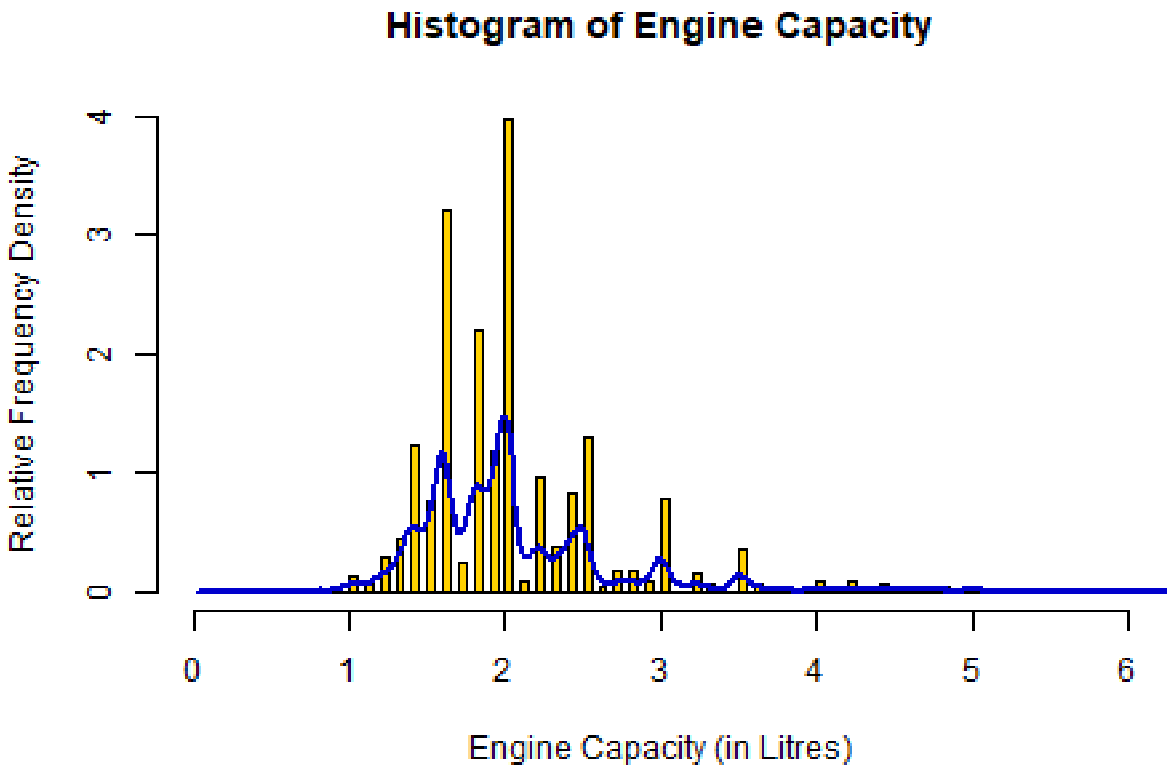


Odometer Value

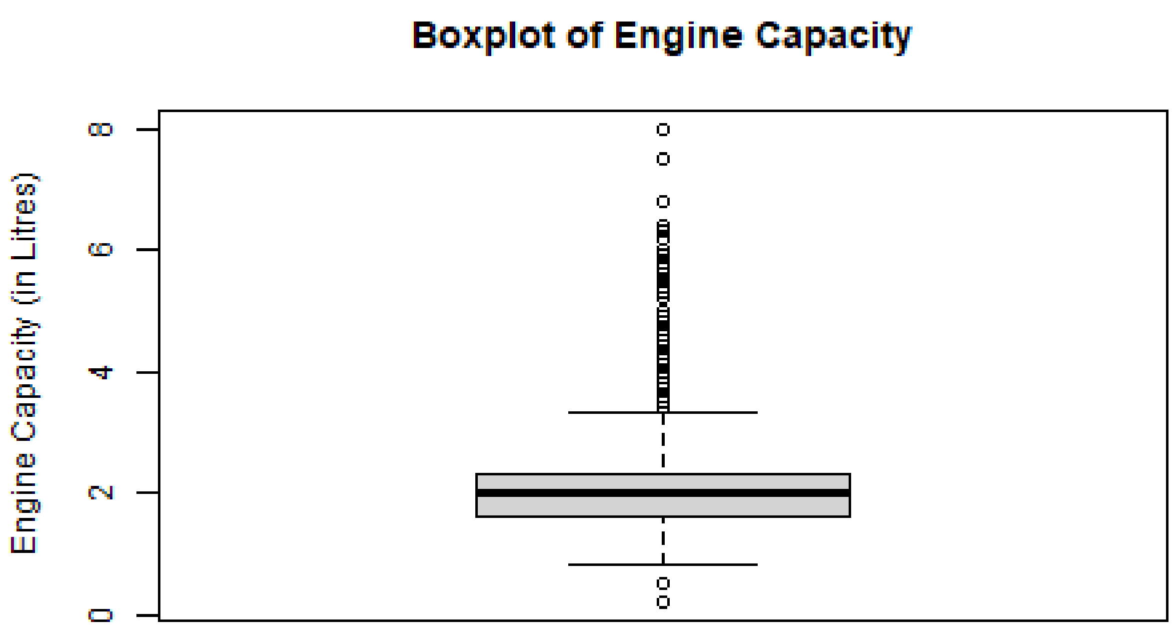


```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       0  158000  250000  248910  325000 1000000
```

Relatively normal distribution of odometer value for cars, but with a noticeable amount of outliers at odometer_value=1,000,000. Further tests are needed to determine whether they should be omitted from the data.

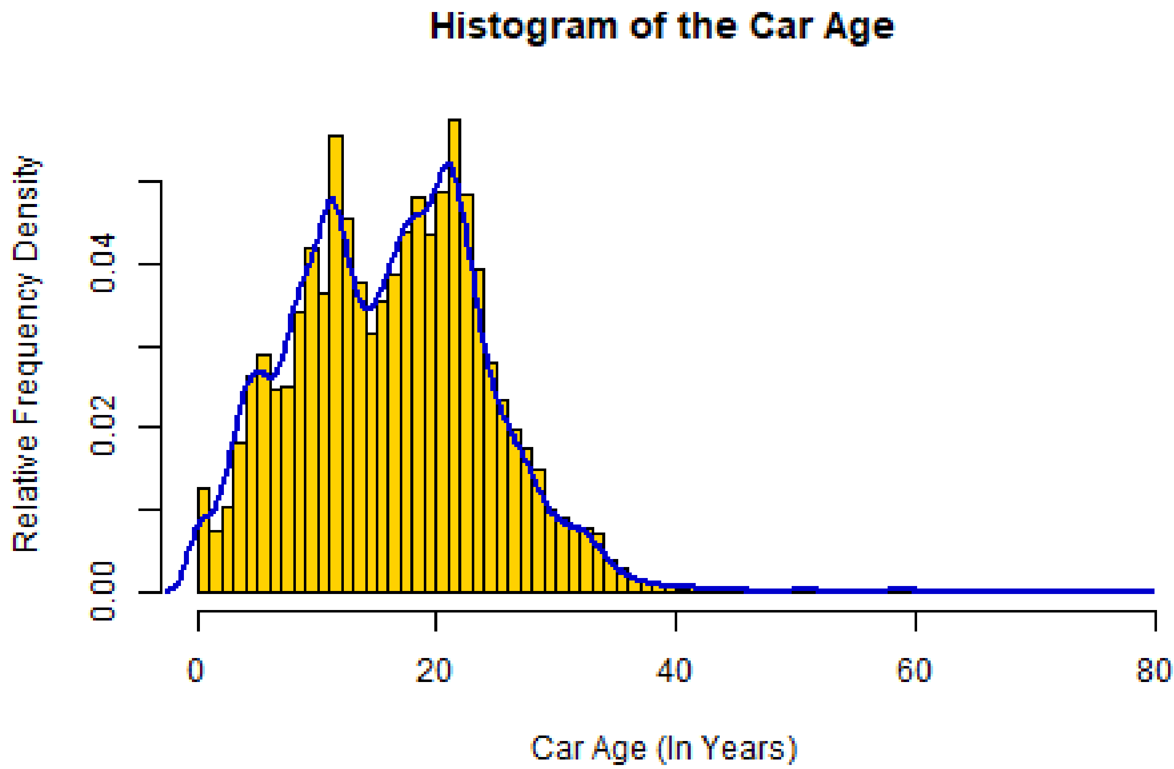


Engine Capacity

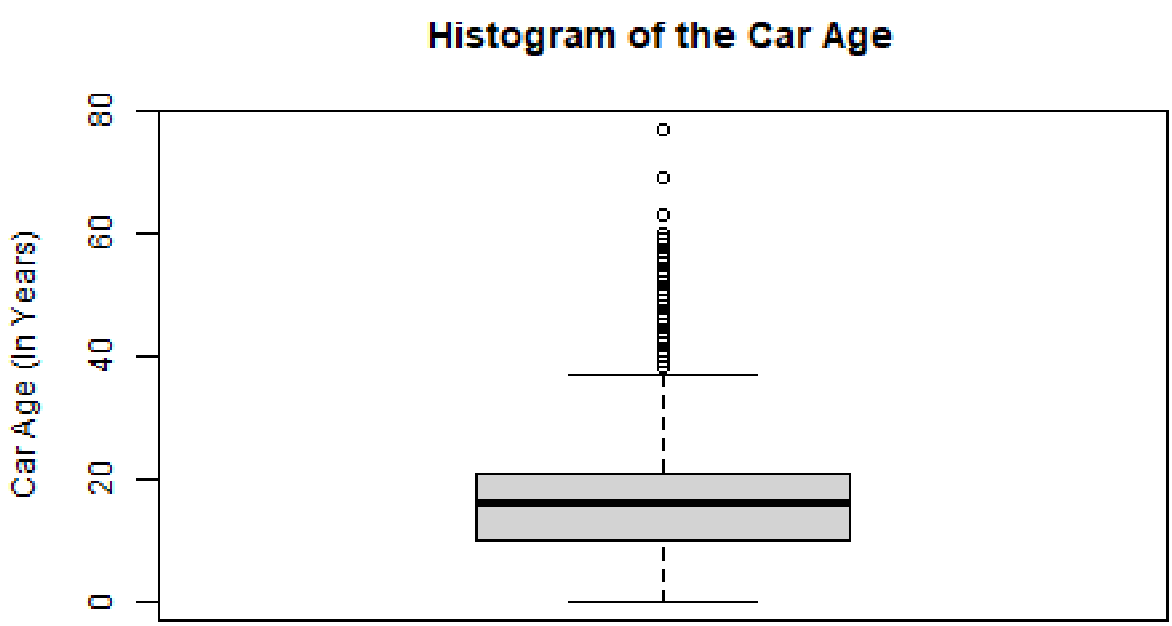


```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.200   1.600   2.000   2.055   2.300   8.000
```

The histogram shows that the distribution of engine capacity is skewed towards the middle at 2 liters, while the boxplot suggests that cars with an engine capacity above 3 and below 1 liter are outliers. Further tests are needed to determine whether they need to be removed.

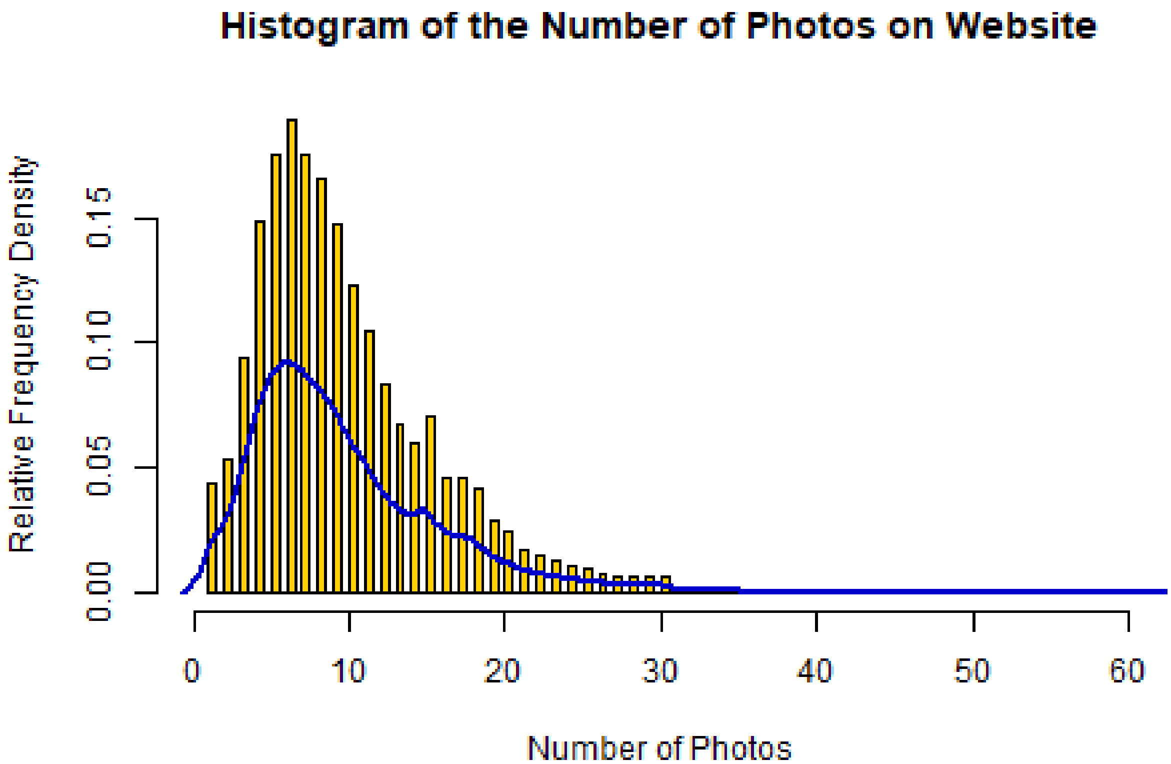

Histogram of the Car Age
Relative Frequency Density
0.00
0.02
0.04
0
20
40
60
80
Car Age (In Years)

Age

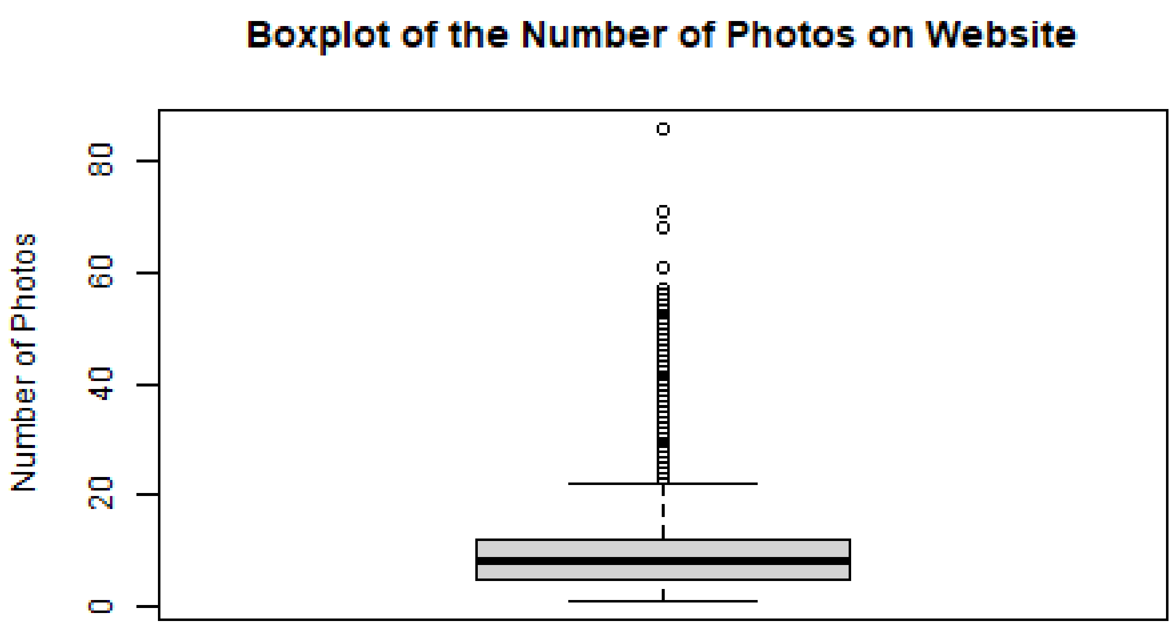

Histogram of the Car Age
Car Age (In Years)
0
20
40
60
80

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00   10.00   16.00   16.06   21.00   77.00
```

The variable for car age is sufficiently normally distributed, with a notable “dip” at the center of the distribution. Some outliers will be further examined in part d.



Number of Photos



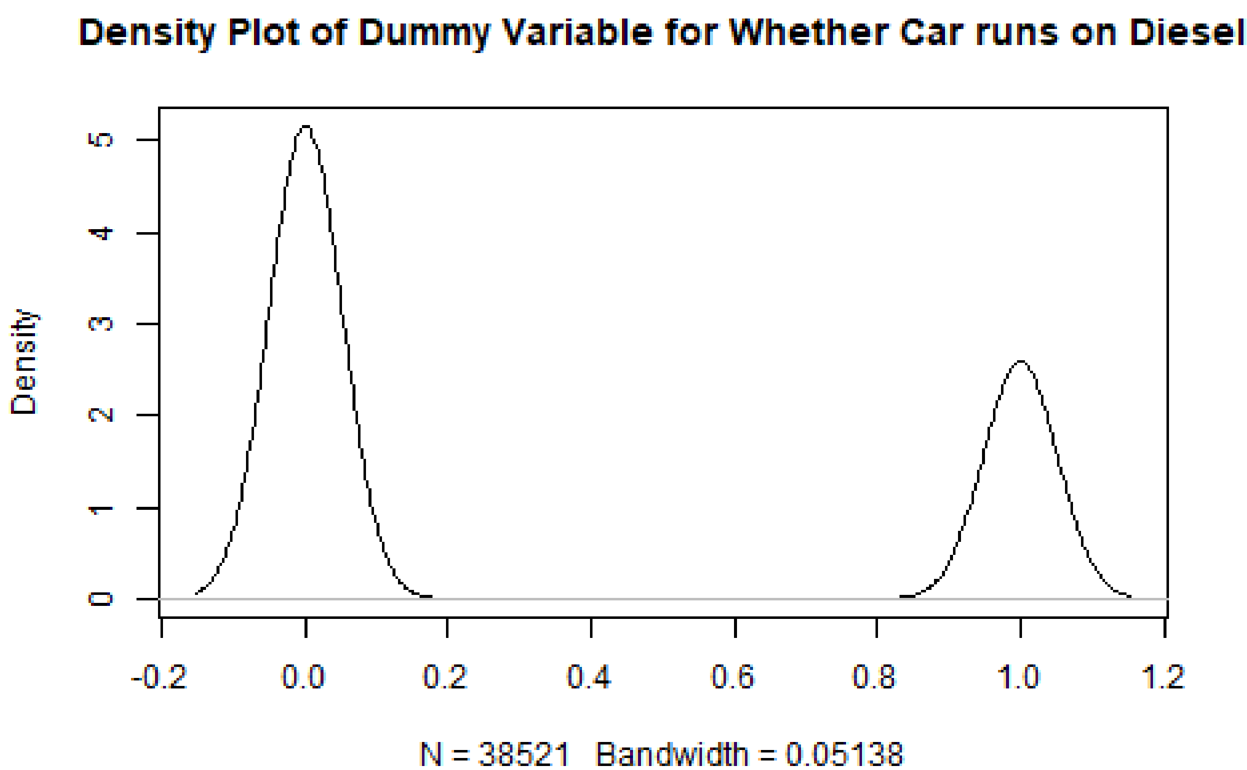
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   5.000   8.000   9.648  12.000  86.000
```

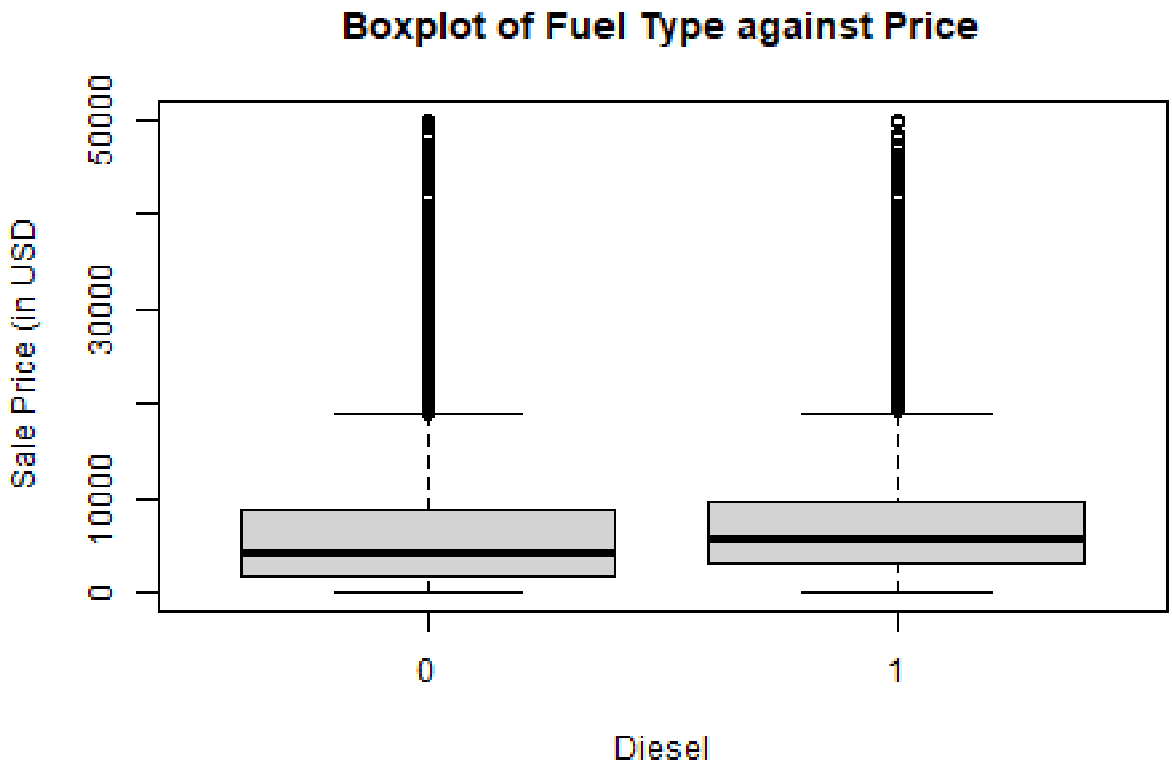
The histogram of the number of photos is slightly positively skewed. There are a few cars with an unusually high number of photos that are considered outliers in the boxplot. Again, further tests are needed to determine whether they are influential outliers that need to be removed.

Diesel

```
## Number of cars with diesel: 12874
```

```
## Percentage cars with diesel: 33.42
```



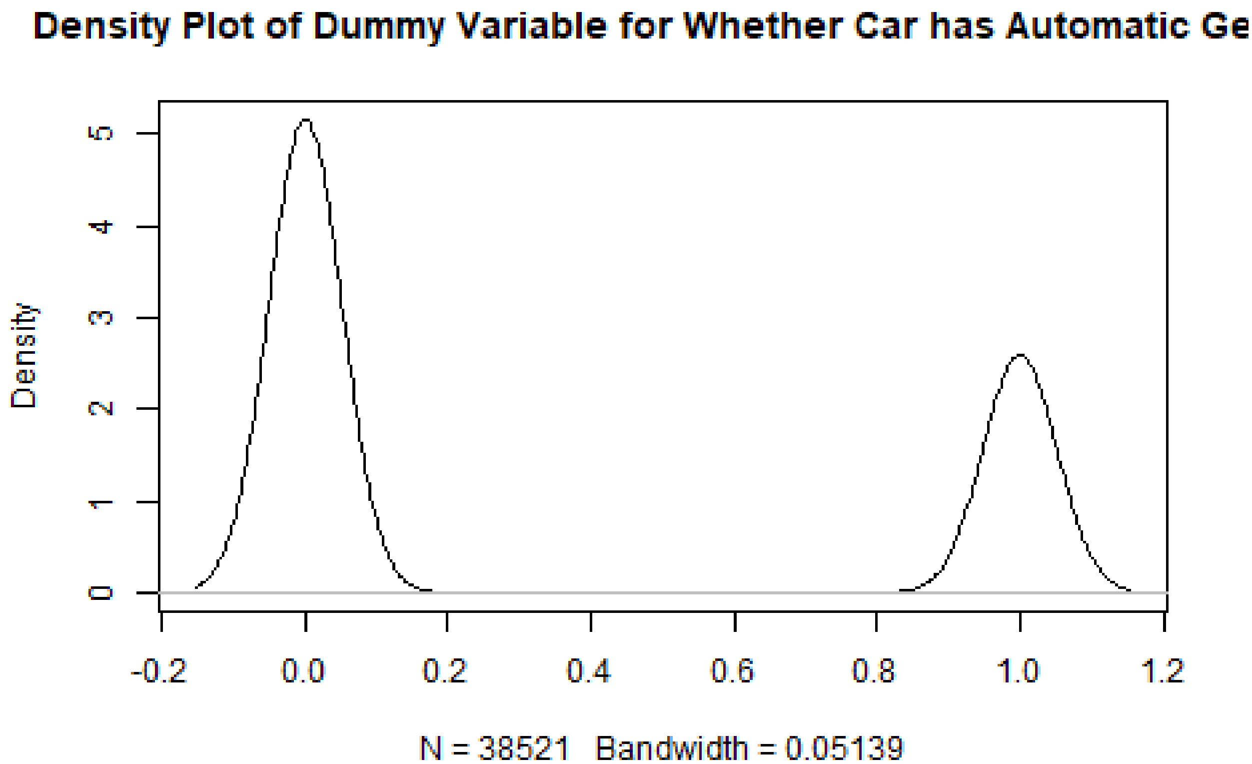


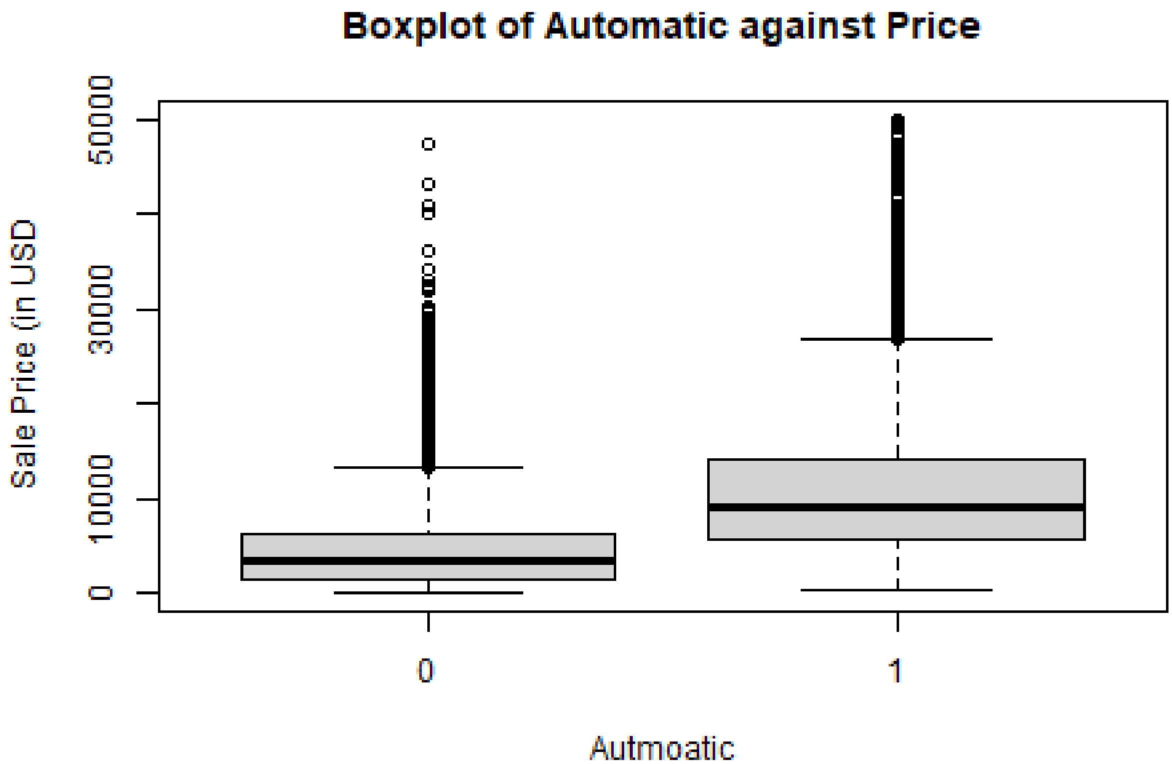
12,874 cars in the data set run on diesel, accounting for approximately 33%, which is a sufficiently large sample for analysis. From the boxplot we can see that there does not seem to be a significant difference in price bewteen gas and diesel vehicles.

Automatic

```
## Number of cars with automatic gear shift: 12888
```

```
## Percentage cars with automatic gear shift: 33.46
```


Density Plot of Dummy Variable for Whether Car has Automatic Ge
Density
0
1
2
3
4
5
-0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2
N = 38521 Bandwidth = 0.05139

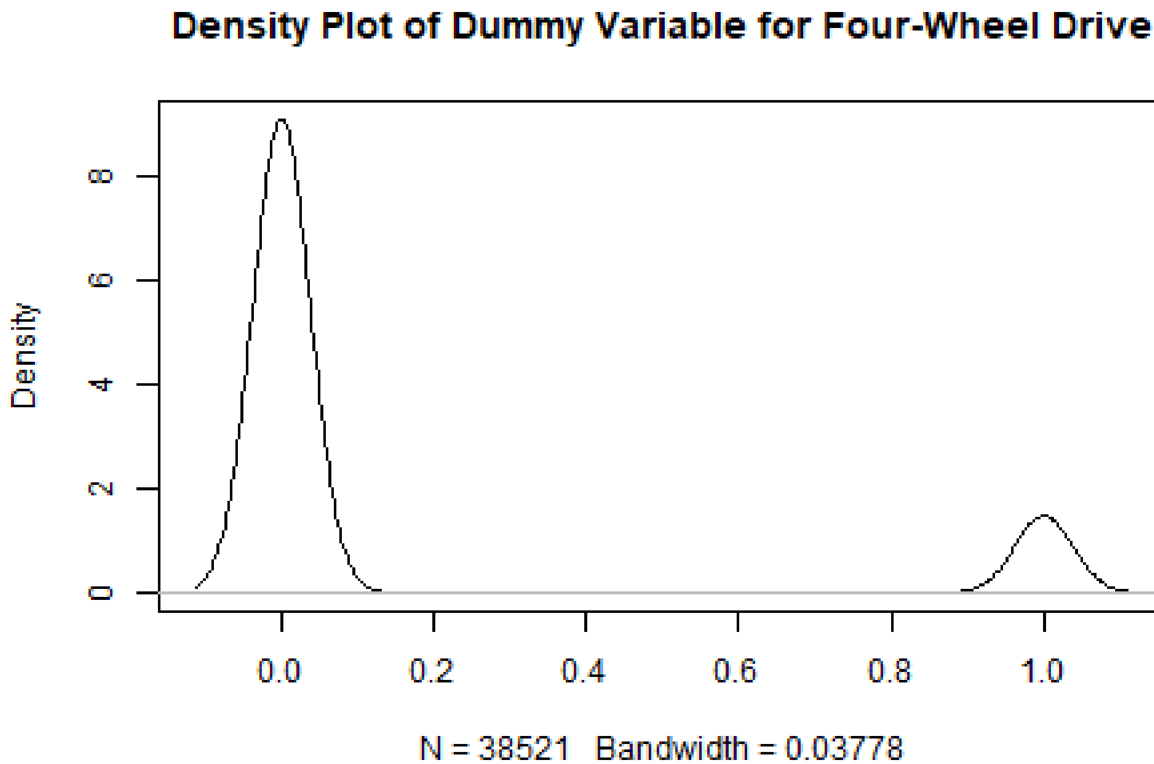


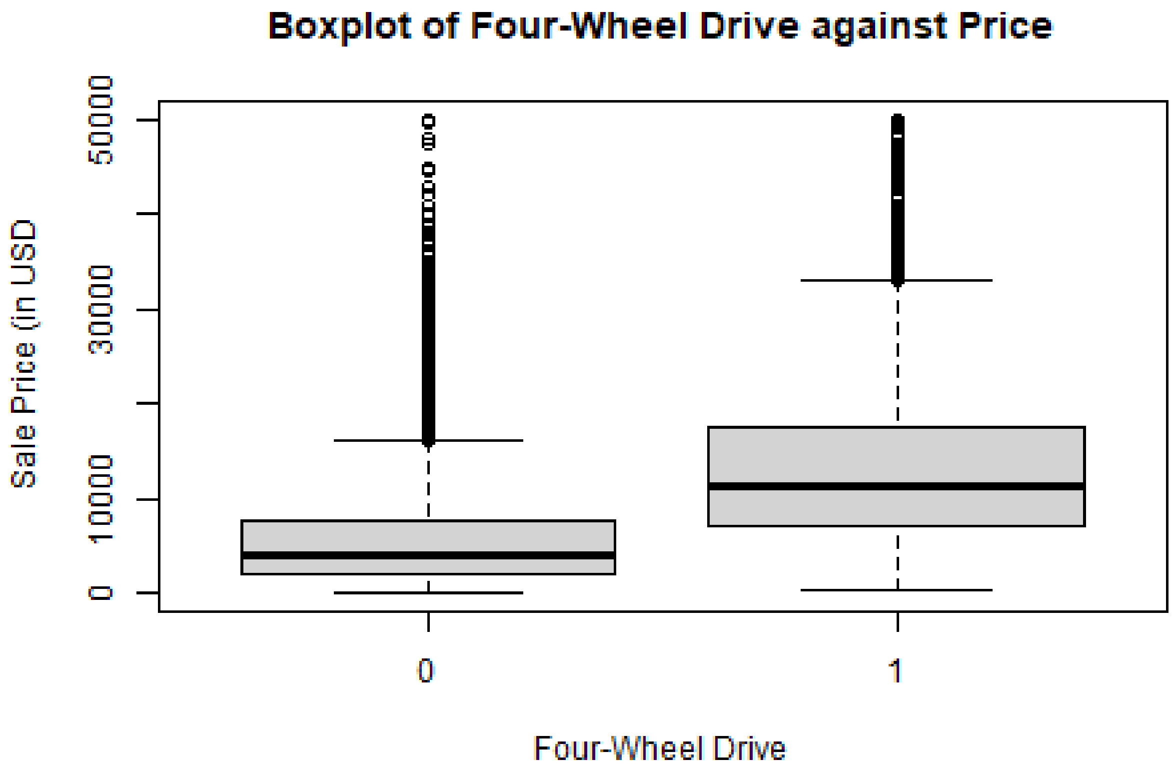
33% of the cars in the data set have an automatic gear shift, which is a sufficiently evenly distributed variable for analysis. The boxplot shows that automatic cars have a higher sale price than manual cars.

Drivetrain all

```
## Number of cars with four-wheel drive: 5387
```

```
## Percentage cars with four-wheel drive: 13.98
```


Density Plot of Dummy Variable for Four-Wheel Drive
Density
0
2
4
6
8
0.0
0.2
0.4
0.6
0.8
1.0
N = 38521 Bandwidth = 0.03778

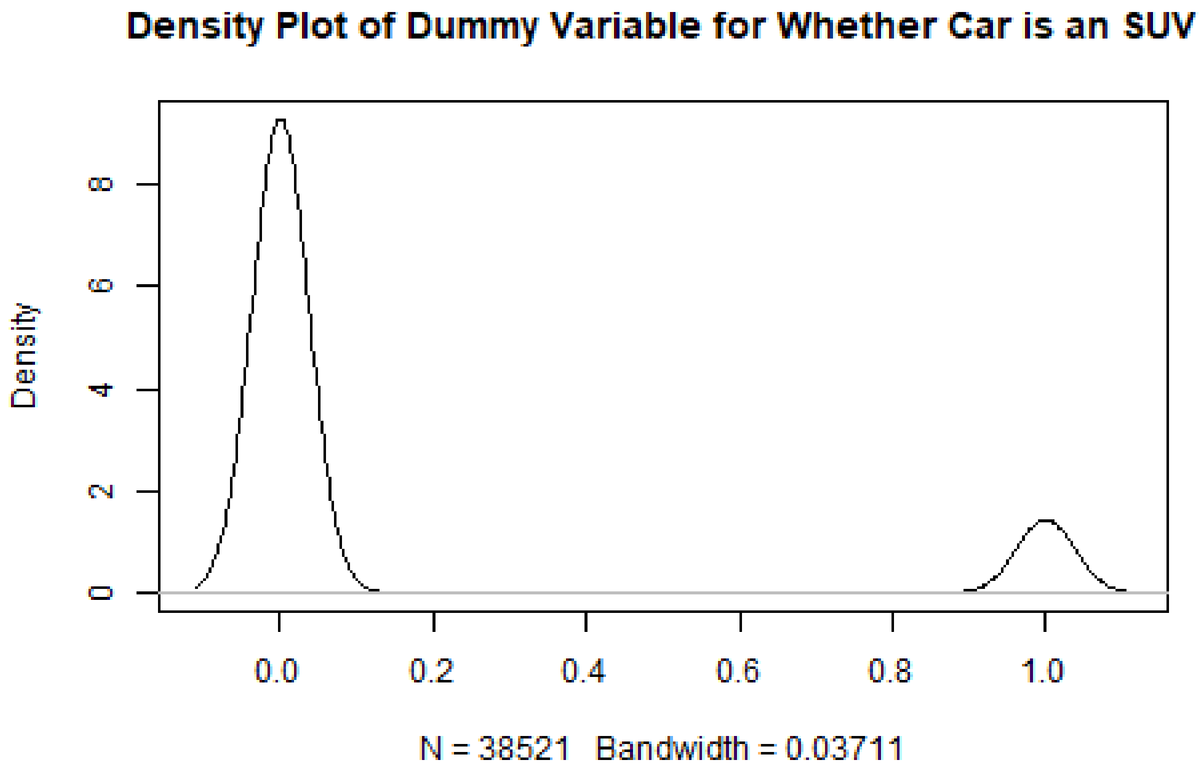


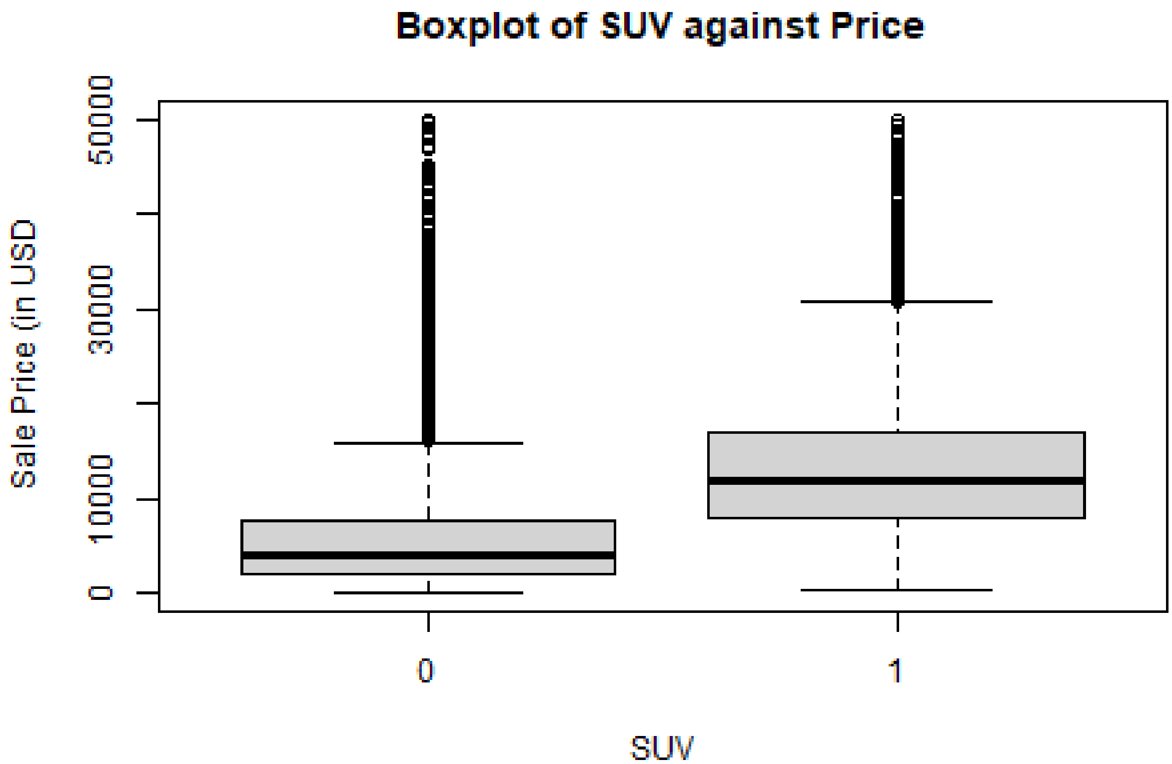
14% of the cars in the data set are propelled by all wheels of the car. Again, while a more even distribution would be preferred, both sides are still sufficiently sized for analysis. The boxplot shows that four-wheel drive vehicles have a higher sale price than front- or rear- drive vehicles.

SUV

```
## Number of SUVs: 5164
```

```
## Percentage SUVs: 13.41
```


Density Plot of Dummy Variable for Whether Car is an SUV
Density
0
2
4
6
8
0.0
0.2
0.4
0.6
0.8
1.0
N = 38521   Bandwidth = 0.03711

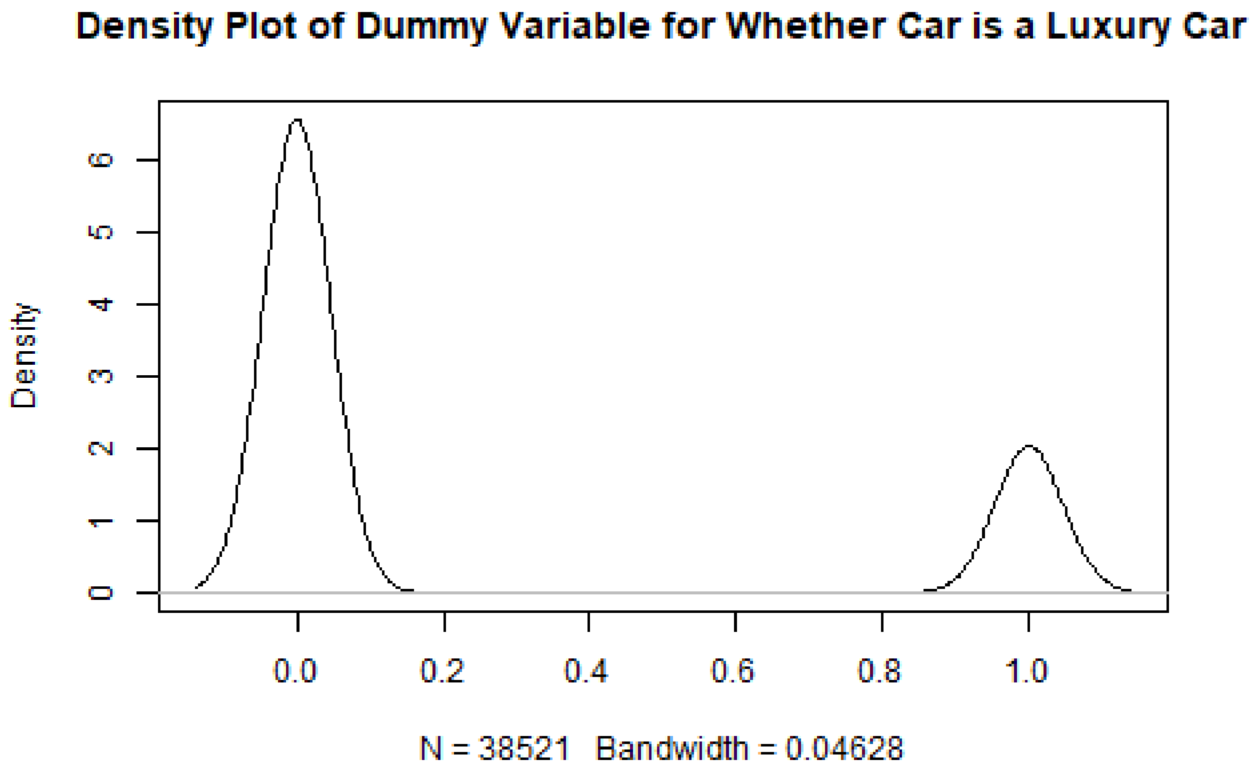


13% of the cars in the data set are an SUV. While the proportion of the variable is more skewed than preferred, it is still sufficiently sized. The boxplot shows that SUVs have a significantly higher sale price than other body types.

Luxury

```
## Number of luxury brand cars: 9106
```

```
## Percentage luxury brand cars: 23.64
```


Density Plot of Dummy Variable for Whether Car is a Luxury Car
Density
0
1
2
3
4
5
6
0.0
0.2
0.4
0.6
0.8
1.0
N = 38521   Bandwidth = 0.04628

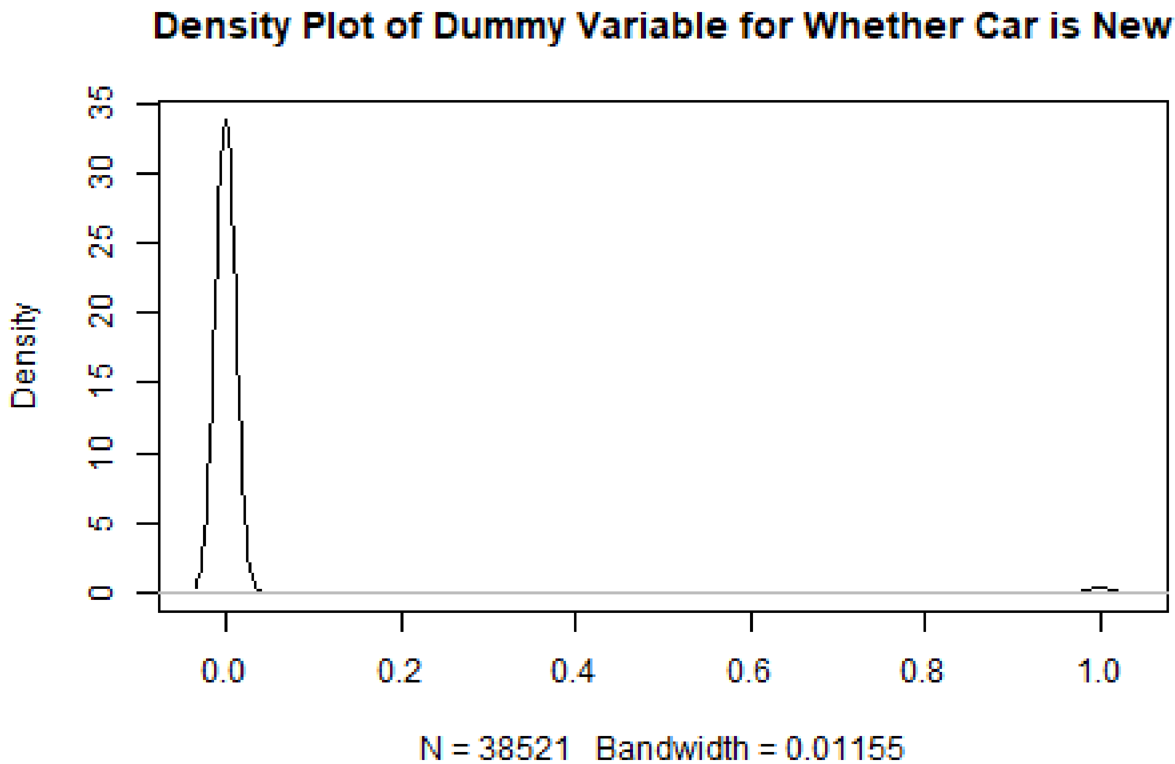
23% of cars in this data set are luxury brands, which is a sufficiently evenly distributed variable. From the boxplot, luxury brand cars seem to have a higher sale price than non-luxury brand cars.

New

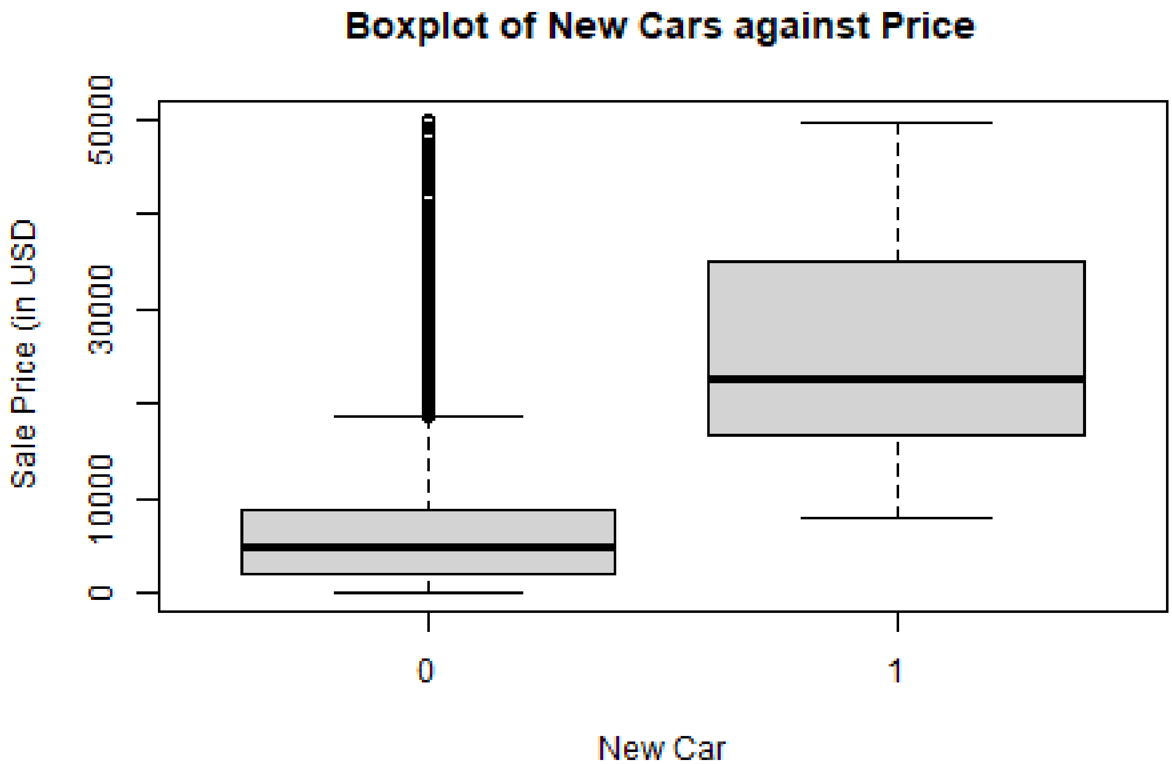
```
## Number of new cars: 438
```

```
## Percentage new cars: 1.14
```

```r
d <- density(cars$new)
plot(d, main="Density Plot of Dummy Variable for Whether Car is New")
```



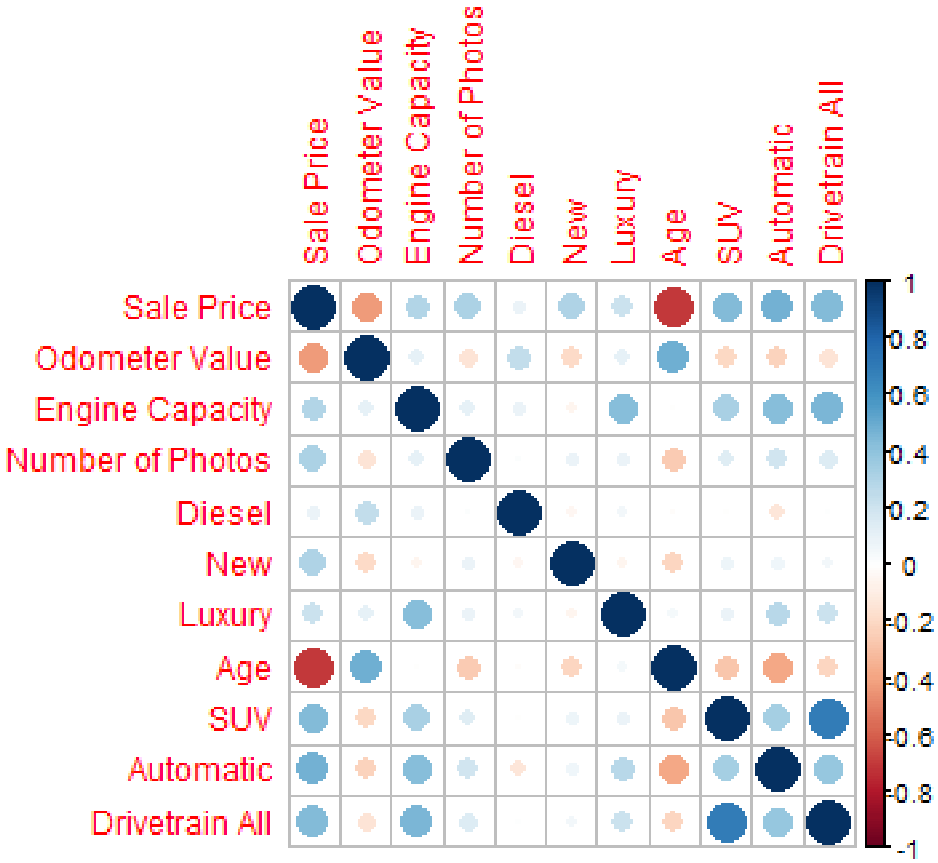

```r
boxplot(cars$price_usd~cars$new,
        main="Boxplot of New Cars against Price",
        xlab = "New Car", ylab="Sale Price (in USD)")
```



Only approximately 1% of cars in this data set is new. This may be a potential cause for concern as the distribution of the sample size is heavily skewed towards cars that aren't new. From the boxplot we can see that new cars have a much higher sale price than used cars.

Correlation Matrix

```
datac<-matrix(c(cars$price_usd, cars$odometer_value,cars$engine_capacity,
                cars$number_of_photos,cars$diesel,cars$new,cars$luxury,cars$age,
                cars$suv,cars$automatic,cars$drivetrain_all),nrow=nrow(cars),
              ncol=11)
colnames(datac)<-c("Sale Price", "Odometer Value","Engine Capacity",
                   "Number of Photos","Diesel","New","Luxury",
                   "Age","SUV","Automatic","Drivetrain All")
corrplot(cor(datac))
```



From this correlation matrix we can see that Sale Price is positively correlated with most of the explanatory variables with the notable exceptions of Age and Odometer Value. It makes intuitive sense that the car's price would be decreasing in age and odometer value. Four-wheel drive seems to be positively correlated with SUV, which we will have to further explore when we analyze the VIFs. Other than that, there is no obvious evidence for multicollinearity between the explanatory variables in the correlation matrix.

Linear Transformation

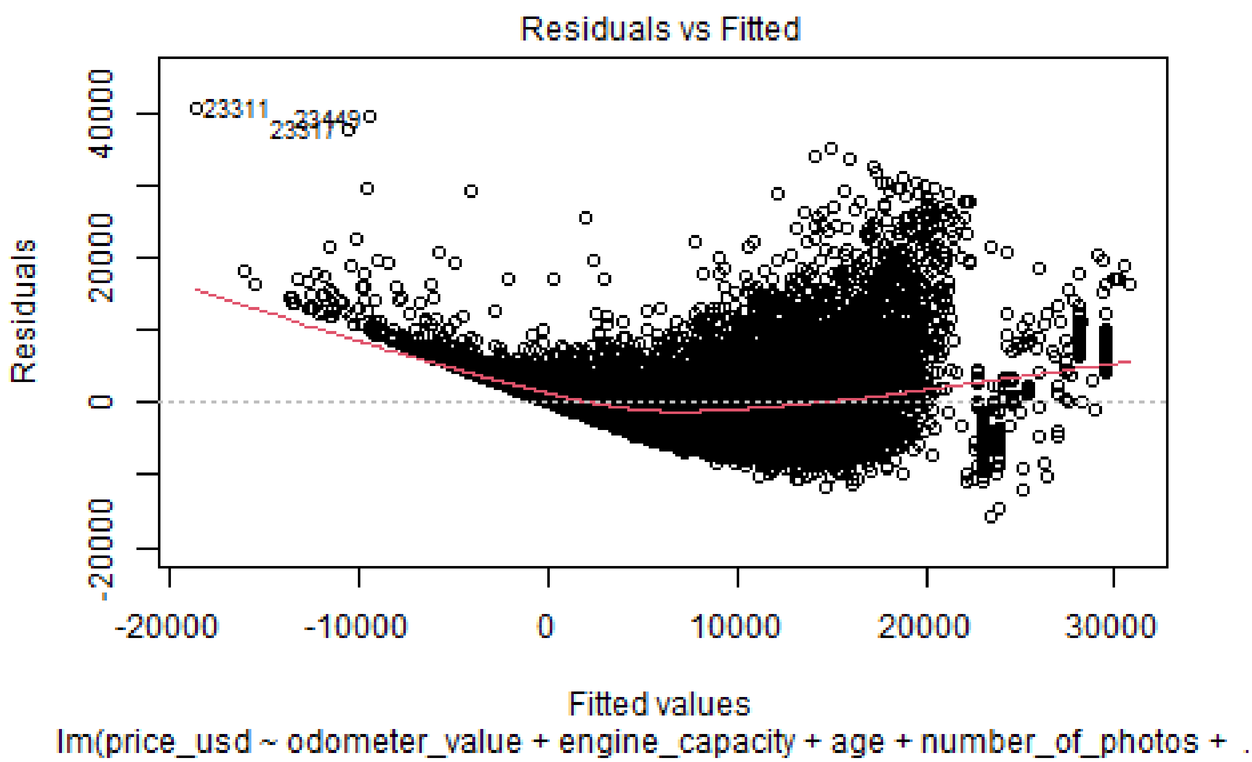
First, let us consider transformations to the y variable. Consider a regression that includes all variables with no transformations.

```
reg1<-lm(price_usd ~ odometer_value + engine_capacity + age + number_of_photos
         + diesel + automatic + drivetrain_all + suv + luxury + new, data=cars)
summary(reg1)
```

```
##
## Call:
## lm(formula = price_usd ~ odometer_value + engine_capacity + age +
##     number_of_photos + diesel + automatic + drivetrain_all +
##     suv + luxury + new, data = cars)
##
## Residuals:
##    Min      1Q Median      3Q     Max
## -15613  -1851   -419   1128  40682
##
```

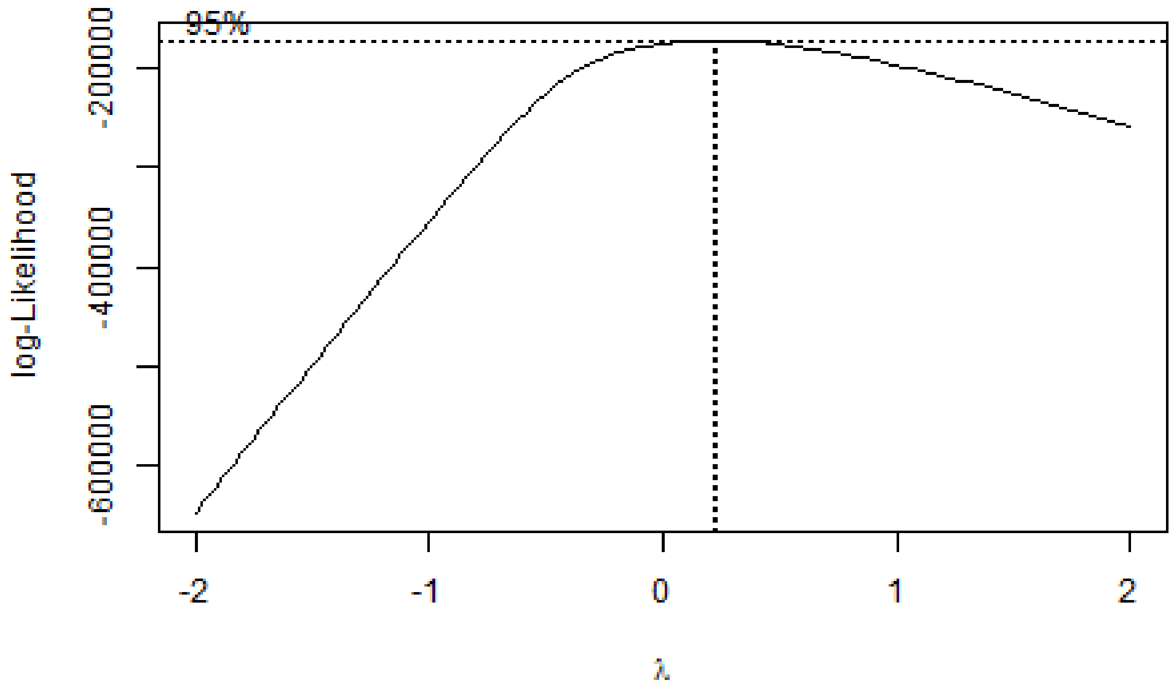
```
## Coefficients:
##                      Estimate    Std. Error t value             Pr(>|t|)    
## (Intercept)       9441.9192849    82.6272010  114.27 <0.0000000000000002 ***
## odometer_value      -0.0060604     0.0001627  -37.26 <0.0000000000000002 ***
## engine_capacity   1243.1096495    35.1612616   35.35 <0.0000000000000002 ***
## age               -411.2263017     2.8983230 -141.88 <0.0000000000000002 ***
## number_of_photos    86.2366322     3.1096378   27.73 <0.0000000000000002 ***
## diesel            1398.7918666    41.0404726   34.08 <0.0000000000000002 ***
## automatic          875.6511583    49.4222513   17.72 <0.0000000000000002 ***
## drivetrain_all    1897.3862287    78.4729324   24.18 <0.0000000000000002 ***
## suv               1885.5489111    76.1154950   24.77 <0.0000000000000002 ***
## luxury            2223.4738765    48.2204200   46.11 <0.0000000000000002 ***
## new              10263.3521623   176.0903563   58.28 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 3551 on 38510 degrees of freedom
## Multiple R-squared:  0.6947, Adjusted R-squared:  0.6946 
## F-statistic:  8762 on 10 and 38510 DF,  p-value: < 0.00000000000000022
```

```r
plot(reg1,1)
```



There is a large deviation from zero as seen in the residuals vs fitted plot, with a seemingly quadratic trend as seen in the red line.

```r
boxcox(reg1)
test<-boxcox(reg1)
```

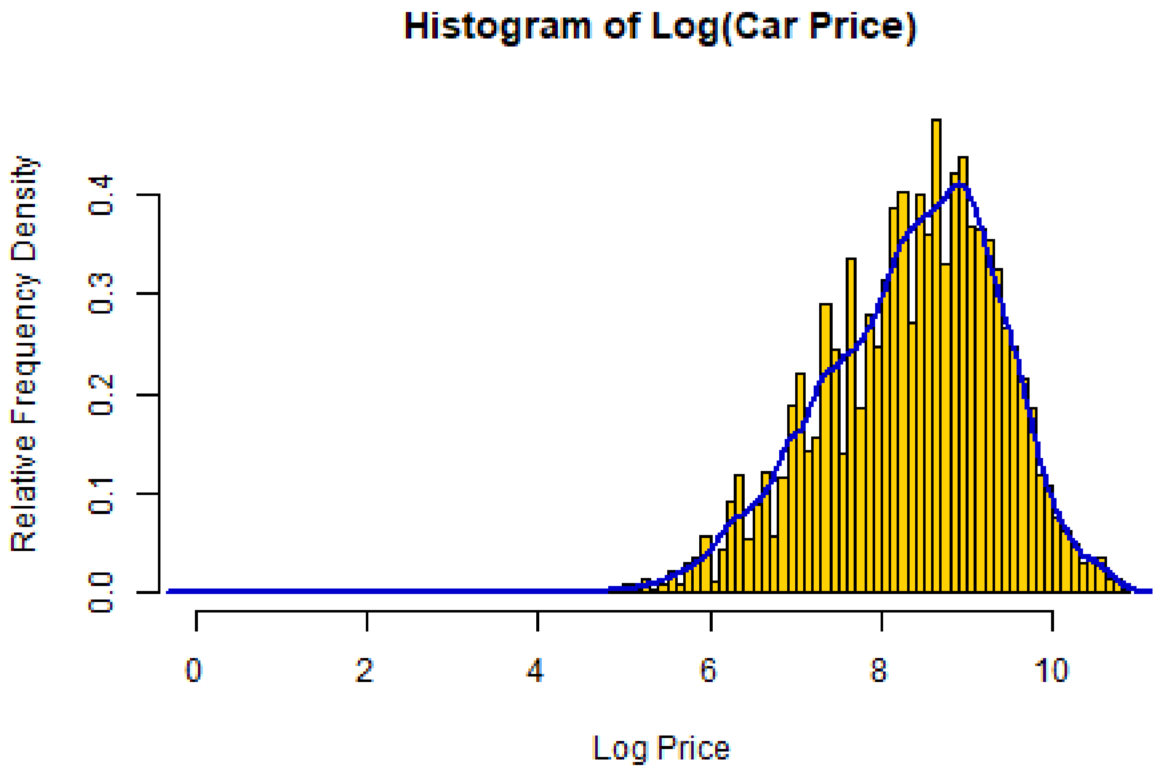



```r
test$x[which.max(test$y)]
```

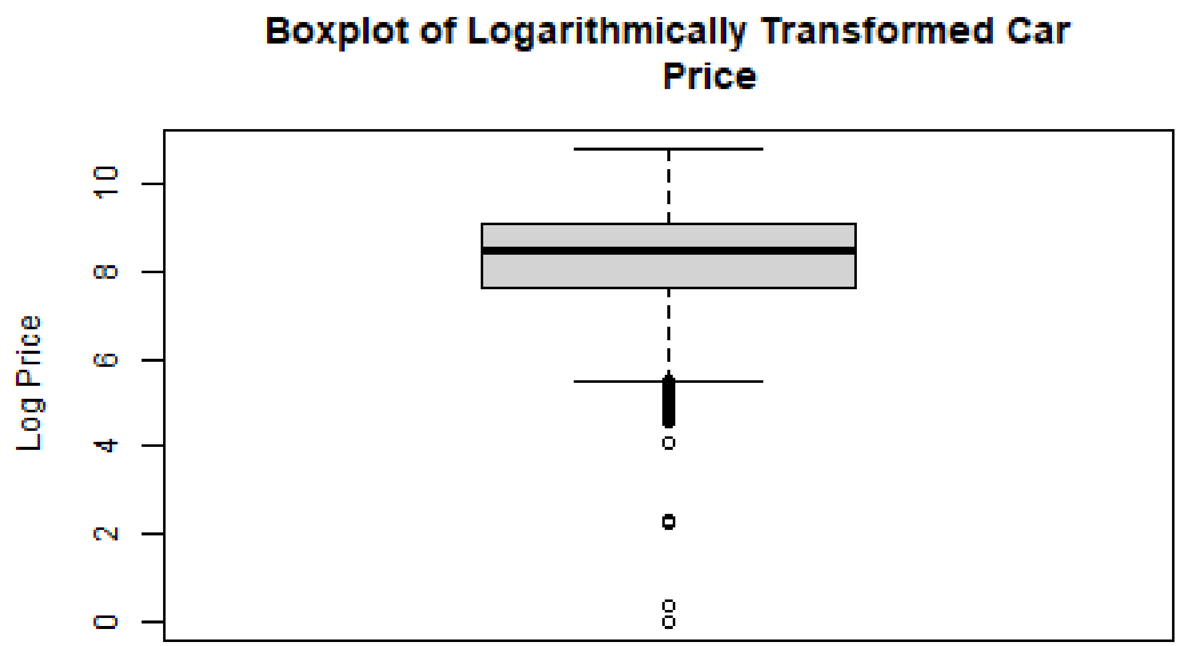
```
## [1] 0.2222222
```

The boxcox plot illustrates a log likelihood function that is maximized when the lambda value is close to 0, suggesting that the y variable should be logarithmically transformed.

```r
cars$transformed_price<-log(cars$price_usd)+0.000001
truehist(cars$transformed_price, col="#FFD100",
         main="Histogram of Log(Car Price)",xlab="Log Price",
         ylab="Relative Frequency Density")
lines(density(cars$transformed_price),lwd=2,col="blue3")
```




```
boxplot(cars$transformed_price, main="Boxplot of Logarithmically Transformed Car
        Price", ylab="Log Price")
```




```r
summary(cars$transformed_price)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
##  0.000001  7.649694  8.476372  8.351812  9.099410 10.819779
```

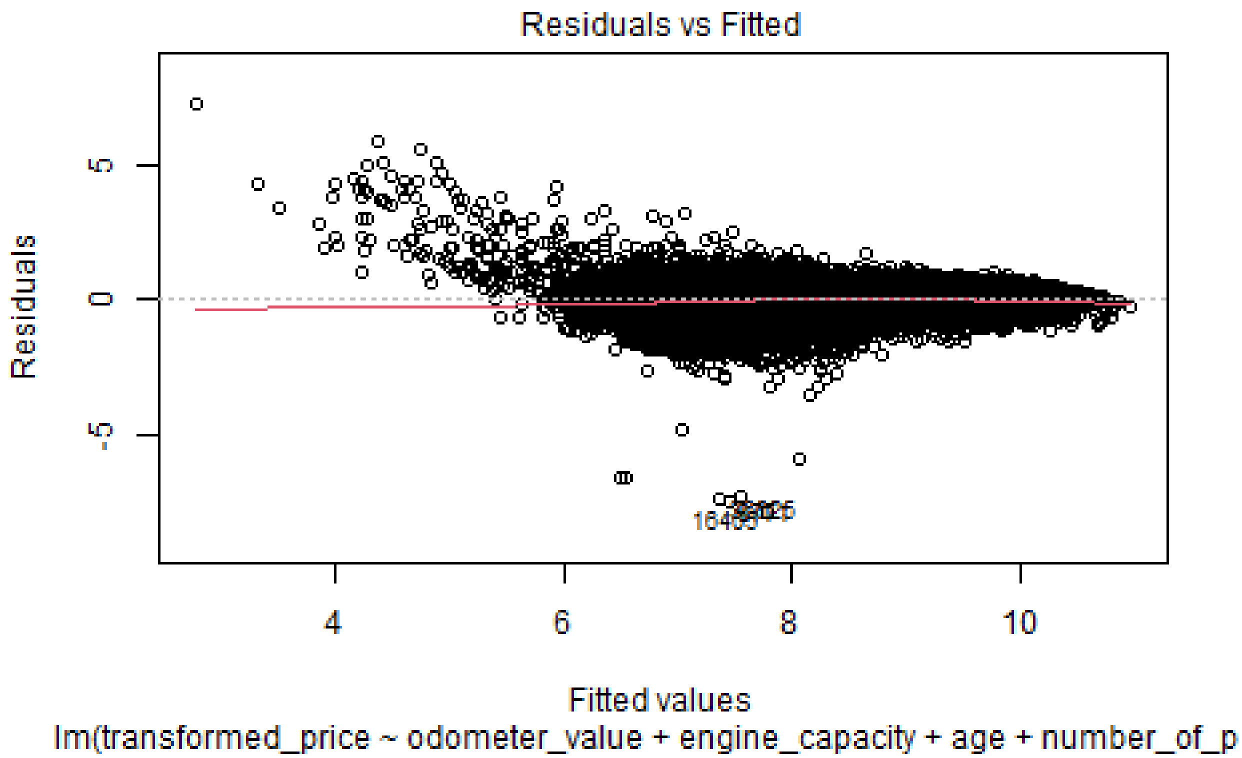
The y variable becomes much more normally distributed when y is logarithmically transformed, unlike the untransformed variable illustrated in part a which is positively skewed. 0.000001 is added to the y variable as the boxcox graph only works for y values > 0.

```r
reg2<-lm(transformed_price ~ odometer_value + engine_capacity + age +
           number_of_photos
         + diesel + automatic + drivetrain_all + suv + luxury + new, data=cars)
summary(reg2)
```

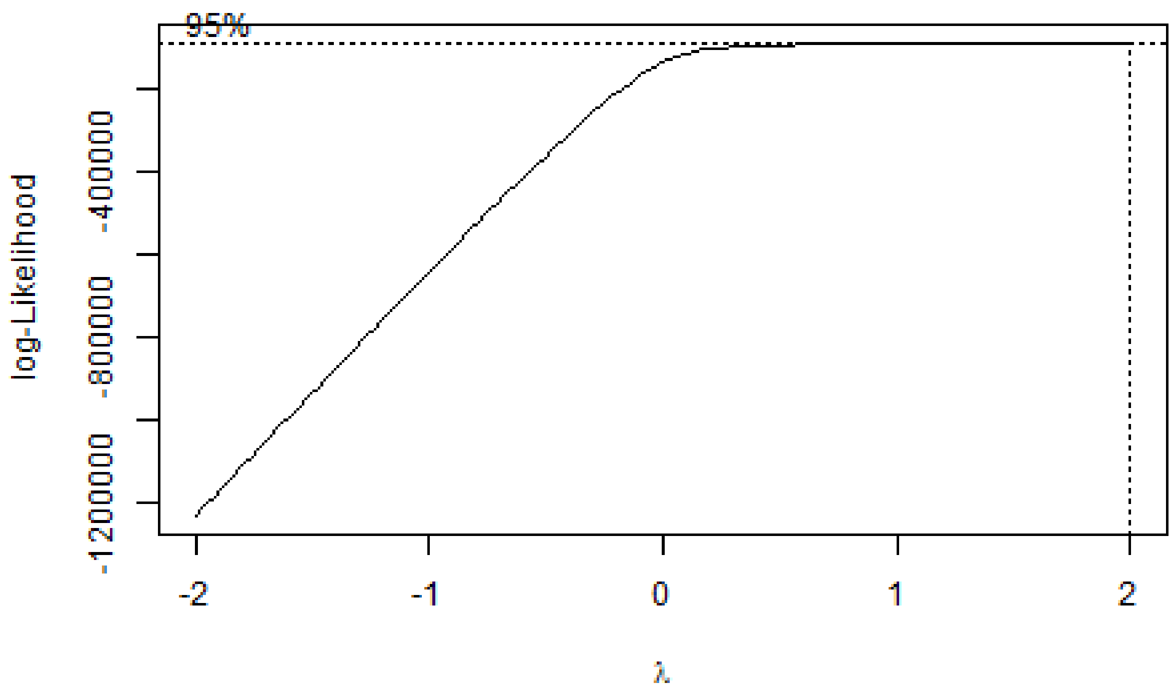
```
##
## Call:
## lm(formula = transformed_price ~ odometer_value + engine_capacity +
##     age + number_of_photos + diesel + automatic + drivetrain_all +
##     suv + luxury + new, data = cars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.7915 -0.2120  0.0345  0.2514  7.2321
##
```

```
## Coefficients:
##                      Estimate     Std. Error  t value             Pr(>|t|)
## (Intercept)       9.03861888881  0.01117883318  808.548 < 0.0000000000000002
## odometer_value   -0.00000036277  0.00000002201  -16.485 < 0.0000000000000002
## engine_capacity   0.26530320459  0.00475705183   55.771 < 0.0000000000000002
## age              -0.09246117733  0.00039212110 -235.798 < 0.0000000000000002
## number_of_photos  0.01070875438  0.00042071040   25.454 < 0.0000000000000002
## diesel            0.27098588068  0.00555246446   48.805 < 0.0000000000000002
## automatic         0.12117260099  0.00668645549   18.122 < 0.0000000000000002
## drivetrain_all    0.05949332746  0.01061679217    5.604         0.0000000211
## suv               0.18382261003  0.01029784878   17.851 < 0.0000000000000002
## luxury            0.31001022203  0.00652385685   47.519 < 0.0000000000000002
## new               0.21323308359  0.02382368874    8.950 < 0.0000000000000002
##
## (Intercept)      ***
## odometer_value   ***
## engine_capacity  ***
## age              ***
## number_of_photos ***
## diesel           ***
## automatic        ***
## drivetrain_all   ***
## suv              ***
## luxury           ***
## new              ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4804 on 38510 degrees of freedom
## Multiple R-squared:  0.7814, Adjusted R-squared:  0.7813
## F-statistic: 1.376e+04 on 10 and 38510 DF,  p-value: < 0.00000000000000022
```

```r
plot(reg2,1)
```


Residuals vs Fitted
Residuals
5
0
-5
4
6
8
10
Fitted values
lm(transformed_price ~ odometer_value + engine_capacity + age + number_of_p

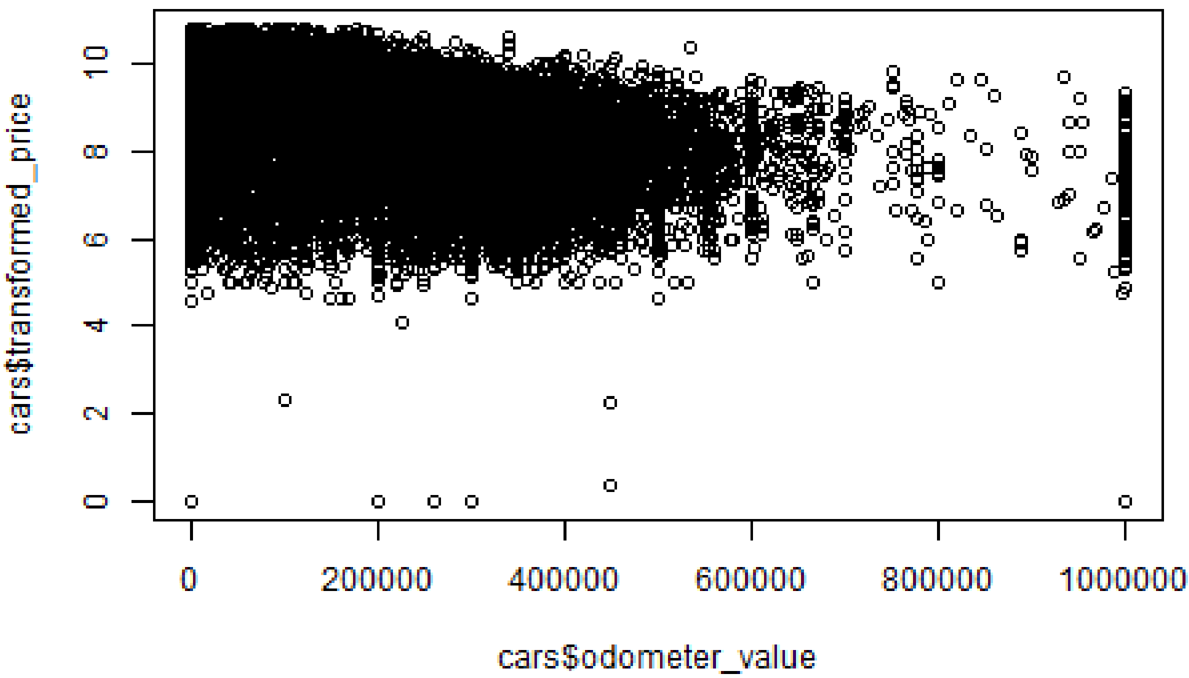
```
boxcox(reg2)
```



The residuals vs fitted plot also has a much closer value to zero, with a decently horizontal trend line, implying a successful transformation to linearity fo the y variable. While the new boxcox plot has a log likelihood function that is maximized when lambda = 2 after the transformation, this is mitigated by the fact that lambda=1 is also very close to the maximum log likelihood function, hence no further linear transformation may be needed. Additionally, we wanted to preserve the interpretability of the y variable which is why we kept the log transformation.

Odometer Value

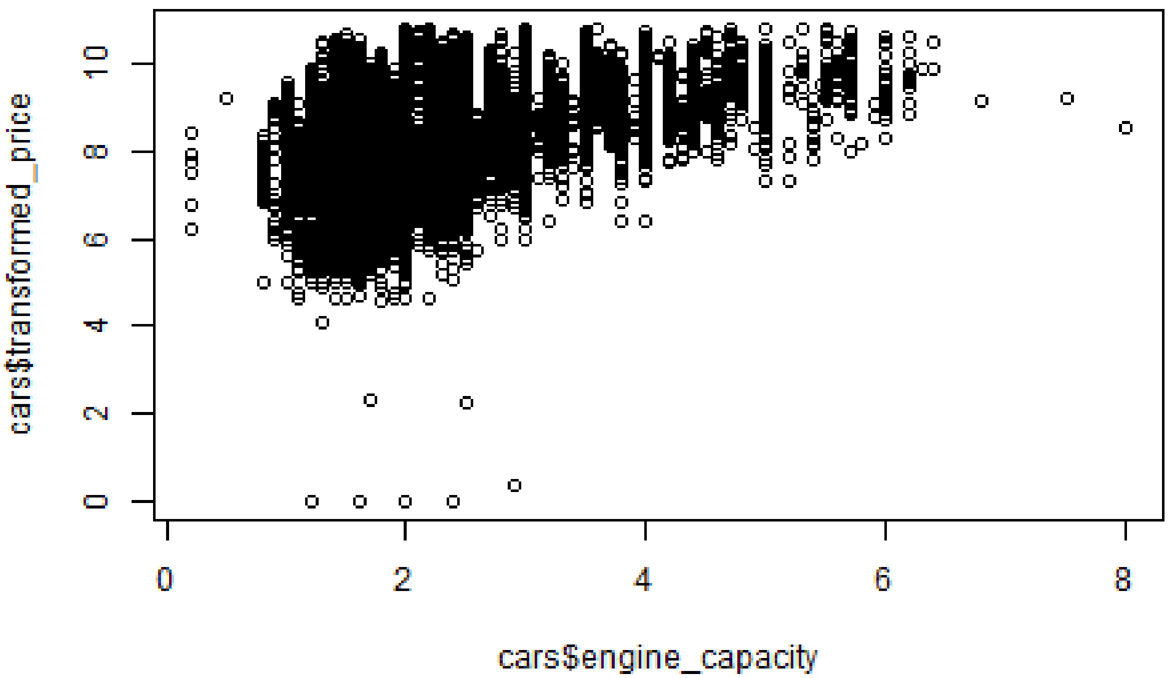
```
plot(cars$transformed_price~cars$odometer_value)
```



A somewhat negative, linear trend can be observed in this plot, before slightly increasing as odometer value increases, though the distribution seems a little random. Therefore, a square term will be considered.

Engine Capacity

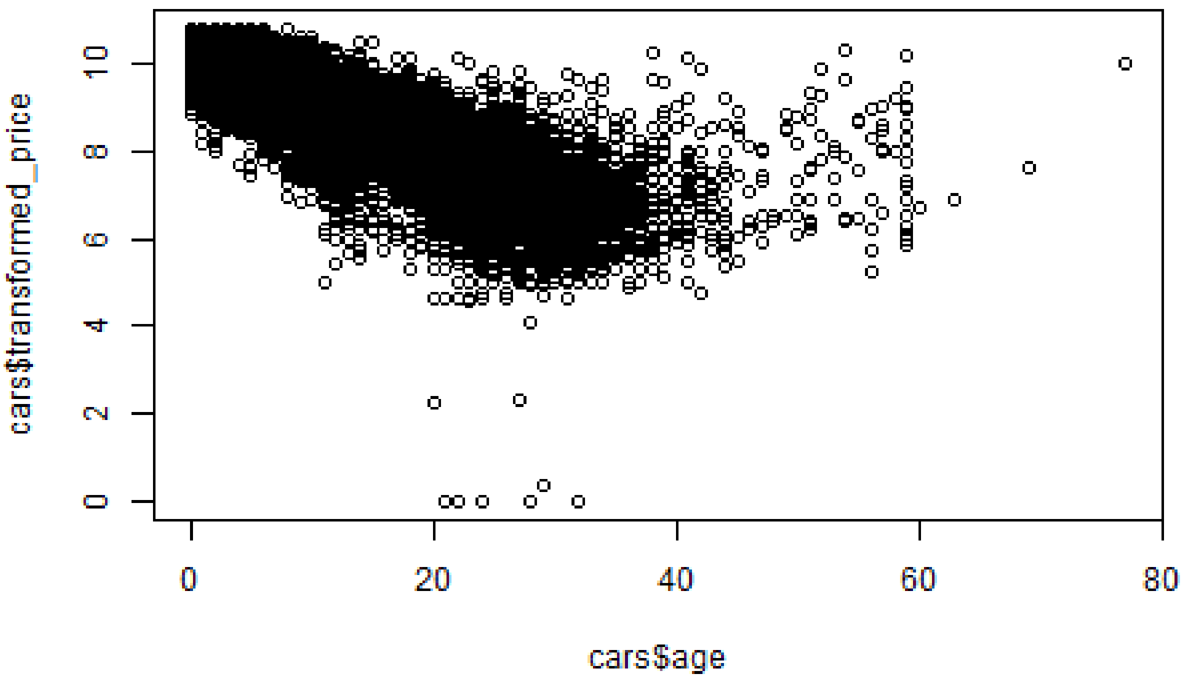
```
plot(cars$transformed_price~cars$engine_capacity)
```



The positive relationship between engine capacity and the transformed price seem linear enough. Hence no linear transformation is needed.

Age

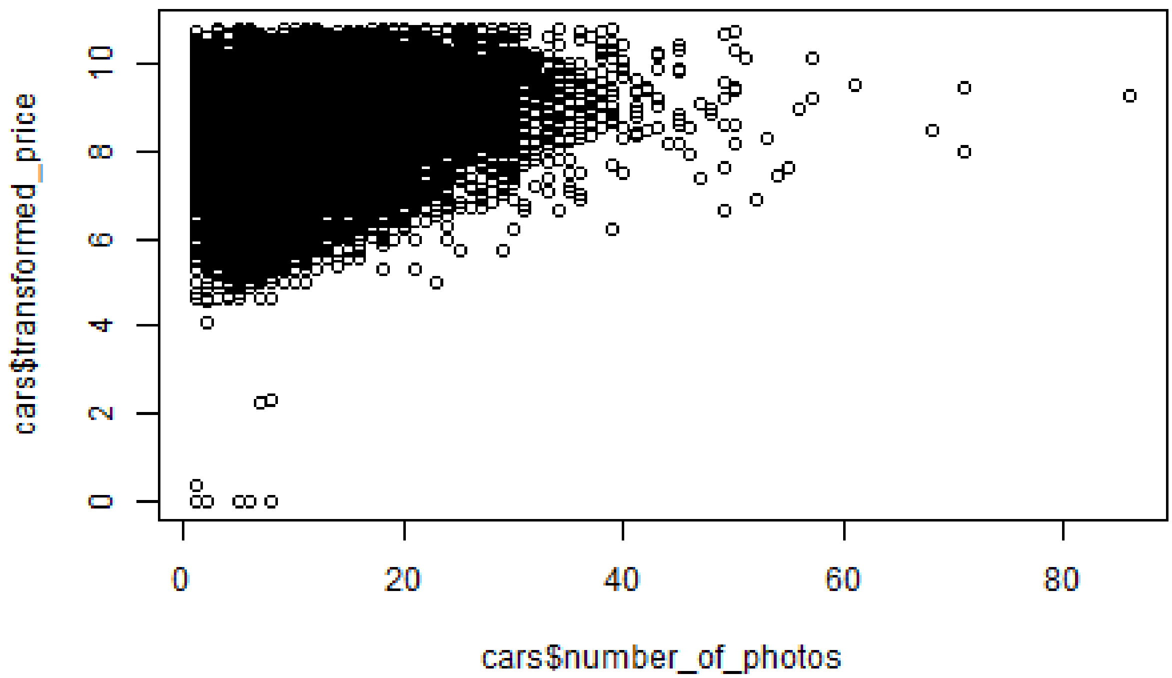
```
plot(cars$transformed_price~cars$age)
```



A clear negative trend is observed in the beginning of this plot, before showing signs of a somewhat increase in price after a certain car age is reached, suggesting a quadratic relationship. Hence a square term for age will be considered.

Number of Photos

```
plot(cars$transformed_price~cars$number_of_photos)
```



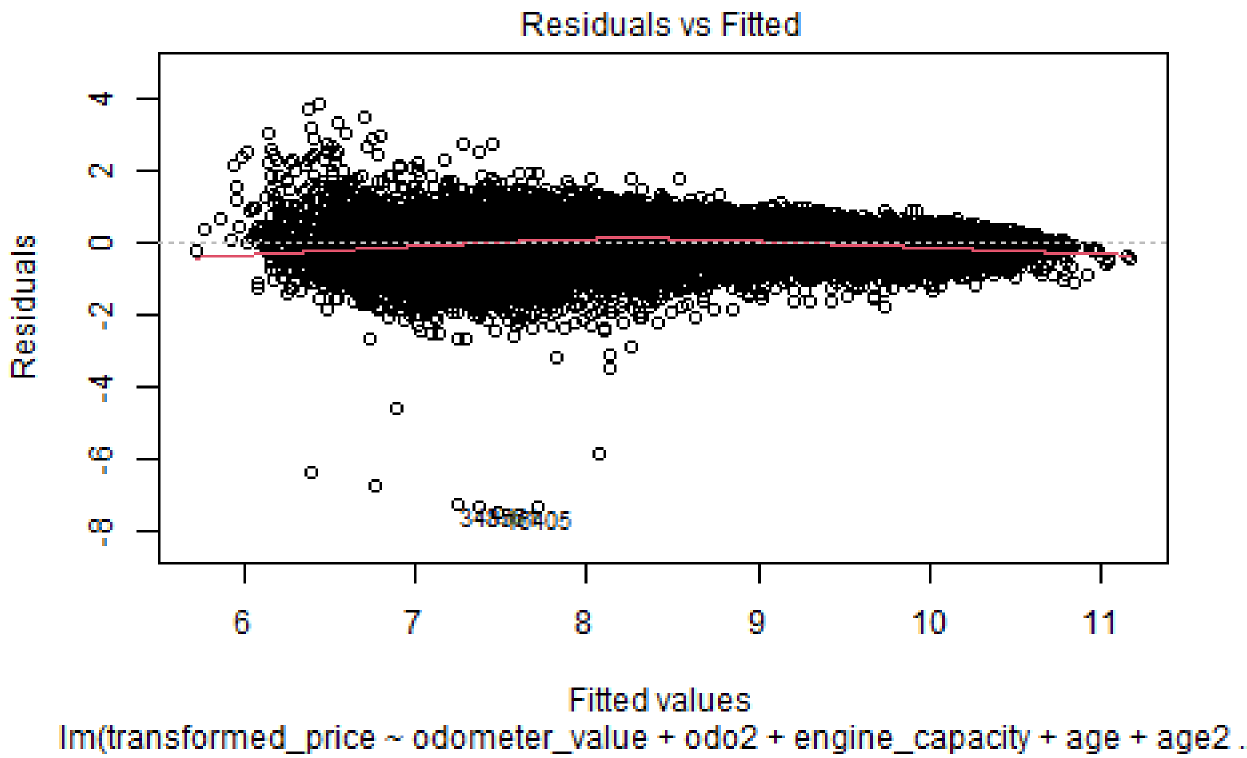
A positive relationship between the number of photos and transformed price can be observed. There is no implication that a different relationship exists, hence no transformation is needed.

```
cars$odo2<-cars$odometer_value^2
cars$age2<-cars$age^2
#Baseline Regression
regb<-lm(transformed_price ~ odometer_value + odo2 + engine_capacity + age +
           age2 + number_of_photos + diesel + automatic + drivetrain_all + suv +
           luxury + new, data=cars)
summary(regb)
```

```
##
## Call:
## lm(formula = transformed_price ~ odometer_value + odo2 + engine_capacity +
##     age + age2 + number_of_photos + diesel + automatic + drivetrain_all +
##     suv + luxury + new, data = cars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.4794 -0.2169  0.0381  0.2633  3.8683
##
## Coefficients:
##                             Estimate           Std. Error  t value
## (Intercept)        9.32857907150918209  0.01209123867800566  771.516
## odometer_value     0.00000094177727497  0.00000005065235097   18.593
## odo2              -0.00000000000115362  0.00000000000005991  -19.256
```

```
## engine_capacity   0.29530879471547417  0.00456291087543738   64.719
## age              -0.16079661321354155  0.00113602075396638 -141.544
## age2              0.00172661787544907  0.00002701635988304   63.910
## number_of_photos  0.00872851094264250  0.00040119848999976   21.756
## diesel            0.25426191528962272  0.00533350855306838   47.673
## automatic         0.10351860058011479  0.00636429372540221   16.266
## drivetrain_all    0.06892131794269928  0.01009521260704043    6.827
## suv               0.13957183784167254  0.00983361598230563   14.193
## luxury            0.28773648438037730  0.00621264634905603   46.315
## new              -0.06884911414028803  0.02342441676352775   -2.939
##                              Pr(>|t|)
## (Intercept)      < 0.0000000000000002 ***
## odometer_value   < 0.0000000000000002 ***
## odo2             < 0.0000000000000002 ***
## engine_capacity  < 0.0000000000000002 ***
## age              < 0.0000000000000002 ***
## age2             < 0.0000000000000002 ***
## number_of_photos < 0.0000000000000002 ***
## diesel           < 0.0000000000000002 ***
## automatic        < 0.0000000000000002 ***
## drivetrain_all      0.00000000000879 ***
## suv              < 0.0000000000000002 ***
## luxury           < 0.0000000000000002 ***
## new                          0.00329 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4567 on 38508 degrees of freedom
## Multiple R-squared:  0.8024, Adjusted R-squared:  0.8023
## F-statistic: 1.303e+04 on 12 and 38508 DF,  p-value: < 0.00000000000000022
```

```r
plot(regb,1)
```

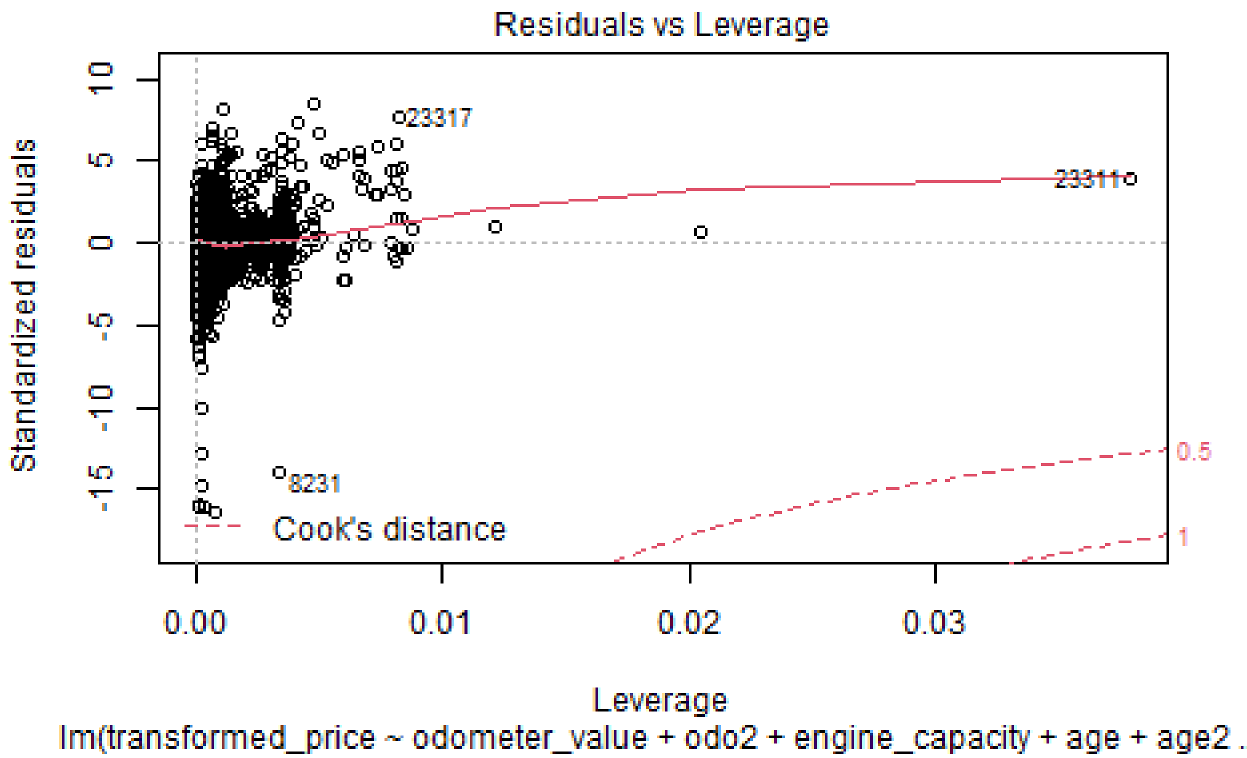


Transformation to linearity only works for non dummy variables, so only these variables are considered for linear transformation. As seen in the new baseline regression, all the added square terms are significant at the 0.1% significance level, and hence will remain included.

Linearity is a key assumption in OLS regression, as linear regression needs the relationship between the independent and dependent variables to be linear. If linearity is violated, then the regression estimates make no sense.

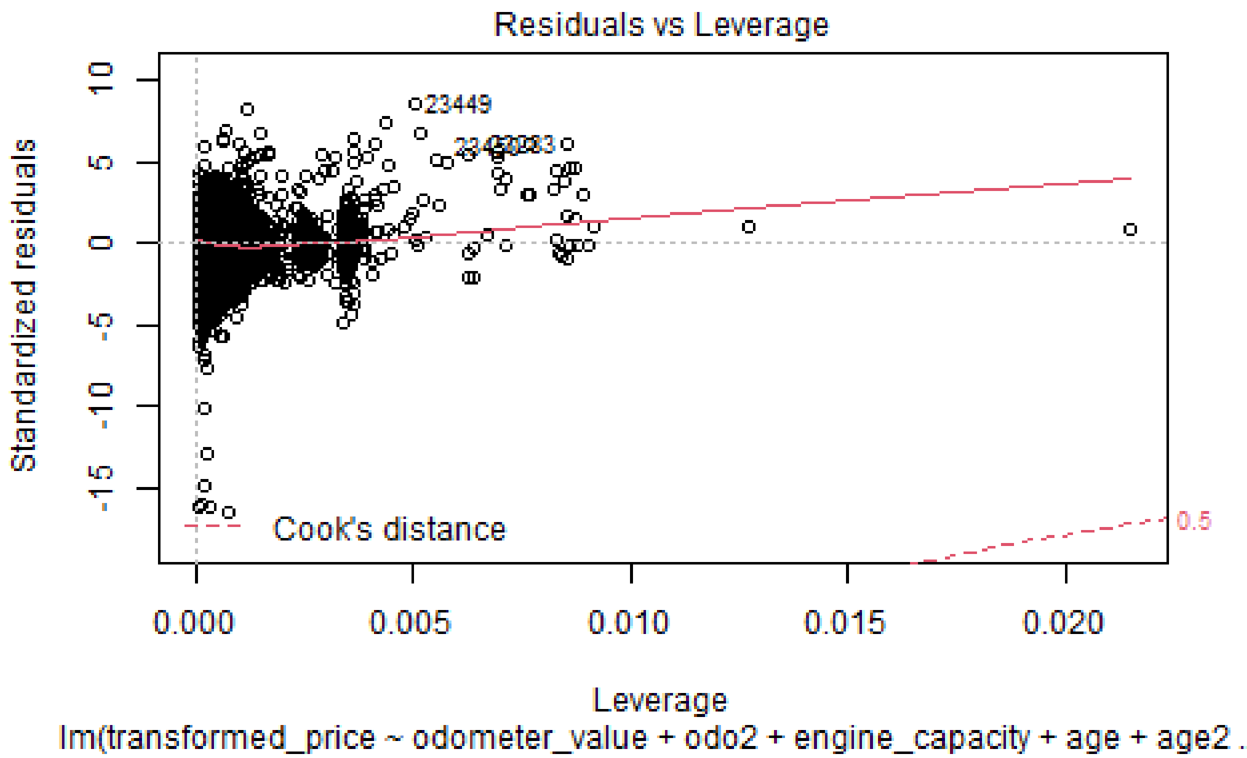
Outlier Analysis

```
plot(regb,5)
```



None of the data points in the dataset fall outside Cook's distance, so there are no influential outliers in the baseline regression. However, data points 8231 and 23311 and 23317 are identified as data points that are closest to the Cook's distance. As these datapoints deviate significantly from the data points and are very close to the Cook's distance, they will be treated as outliers and removed.

```
cars$test<-c(1:nrow(cars))
cars<-subset(cars,cars$test!=23311&cars$test!=8231&cars$test!=23317)
regb<-lm(transformed_price ~ odometer_value + odo2 + engine_capacity + age +
           age2 + number_of_photos + diesel + automatic + drivetrain_all + suv +
           luxury + new, data=cars)
plot(regb,5)
```



As seen in the new residuals vs leverage graph, the data points are much more compact and further away from the Cook's distance, implying that we have successfully removed the more influential outliers in the data.

```r
summary(regb)
```

```
##
## Call:
## lm(formula = transformed_price ~ odometer_value + odo2 + engine_capacity +
##     age + age2 + number_of_photos + diesel + automatic + drivetrain_all +
##     suv + luxury + new, data = cars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.4839 -0.2161  0.0377  0.2628  3.9150
##
## Coefficients:
##                              Estimate            Std. Error  t value
## (Intercept)       9.32614842823998380  0.01206804391715682  772.797
## odometer_value    0.00000090384755576  0.00000005059843672   17.863
## odo2             -0.00000000000109452  0.00000000000005986  -18.285
## engine_capacity   0.29396229050355310  0.00454945681627678   64.615
## age              -0.15940229110030485  0.00114953161369423 -138.667
## age2              0.00168682300614754  0.00002750080753257   61.337
## number_of_photos  0.00873139877354928  0.00039985660809647   21.836
## diesel            0.25392109969834403  0.00531490598140283   47.775
```

```
## automatic          0.10365352517868601  0.00634204568321061   16.344
## drivetrain_all     0.06933535695412331  0.01005990937682693    6.892
## suv                0.14034572283880409  0.00979991746083231   14.321
## luxury             0.28858275231365549  0.00619251159700137   46.602
## new               -0.06472726435685178  0.02335665667603536   -2.771
##                               Pr(>|t|)
## (Intercept)       < 0.0000000000000002 ***
## odometer_value    < 0.0000000000000002 ***
## odo2              < 0.0000000000000002 ***
## engine_capacity   < 0.0000000000000002 ***
## age               < 0.0000000000000002 ***
## age2              < 0.0000000000000002 ***
## number_of_photos  < 0.0000000000000002 ***
## diesel            < 0.0000000000000002 ***
## automatic         < 0.0000000000000002 ***
## drivetrain_all       0.00000000000558 ***
## suv               < 0.0000000000000002 ***
## luxury            < 0.0000000000000002 ***
## new                            0.00559 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4551 on 38505 degrees of freedom
## Multiple R-squared:  0.8034, Adjusted R-squared:  0.8033
## F-statistic: 1.311e+04 on 12 and 38505 DF,  p-value: < 0.00000000000000022
```

When looking at the coefficient for the "New" variable, unusually, cars that are new tend to be cheaper than cars that are not. A potential reason for this abnormality may be down to the small sample size of cars ithat are new, only account for approximately 1.1% of the entire data set. However, the Mallows CP estimate deemed this variable important, hence it will remain included in the regression.

NAs are removed in Part 1. This should be fine given that there are only 10 missing variables in the data set, with a sizable 38521 variables left for analysis.

Part 3: Model Building

By constantly adjusting the model, we decided to use regb as our final model. Compared with reg1, reg2 cleans up the data and discards many N/A and outliers, which makes our data more effective for problem analysis. Compared with reg2, Regb analyzes the data in more dimensions by adding some variables (odo2 and age2). These two variables are used to reflect the decreasing marginal effects caused by too large odometer_value and age, which makes the model more comprehensive.

Test for Multicollinearity

```
#summary(reg1)
tidy(vif(reg1))
```

```
## Warning: 'tidy.numeric' is deprecated.
## See help("Deprecated")
```

```
## Warning: `data_frame()` is deprecated as of tibble 1.1.0.
```

```
## Please use `tibble()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.

## # A tibble: 10 x 2
##    names                 x
##    <chr>             <dbl>
##  1 odometer_value     1.50
##  2 engine_capacity    1.70
##  3 age                1.67
##  4 number_of_photos   1.10
##  5 diesel             1.14
##  6 automatic          1.66
##  7 drivetrain_all     2.26
##  8 suv                2.05
##  9 luxury             1.28
## 10 new                1.06
```

```r
#summary(reg2)
tidy(vif(reg2))
```

```
## Warning: 'tidy.numeric' is deprecated.
## See help("Deprecated")

## # A tibble: 10 x 2
##    names                 x
##    <chr>             <dbl>
##  1 odometer_value     1.50
##  2 engine_capacity    1.70
##  3 age                1.67
##  4 number_of_photos   1.10
##  5 diesel             1.14
##  6 automatic          1.66
##  7 drivetrain_all     2.26
##  8 suv                2.05
##  9 luxury             1.28
## 10 new                1.06
```

```r
#summary(regb)
tidy(vif(regb))
```

```
## Warning: 'tidy.numeric' is deprecated.
## See help("Deprecated")

## # A tibble: 12 x 2
##    names                 x
##    <chr>             <dbl>
##  1 odometer_value     8.80
##  2 odo2               6.57
##  3 engine_capacity    1.73
##  4 age               15.9
##  5 age2              12.7
```

```
##   6 number_of_photos  1.10
##   7 diesel            1.17
##   8 automatic         1.66
##   9 drivetrain_all    2.26
##  10 suv               2.07
##  11 luxury            1.29
##  12 new               1.14
```

We can see that for reg1 and reg2 nothing should be removed since all VIF < 4. For regb, it makes sense that cars$odometer_value, cars$odo2, cars$age and cars$age2 as they are square terms of each other, and are expected to be highly correlated.

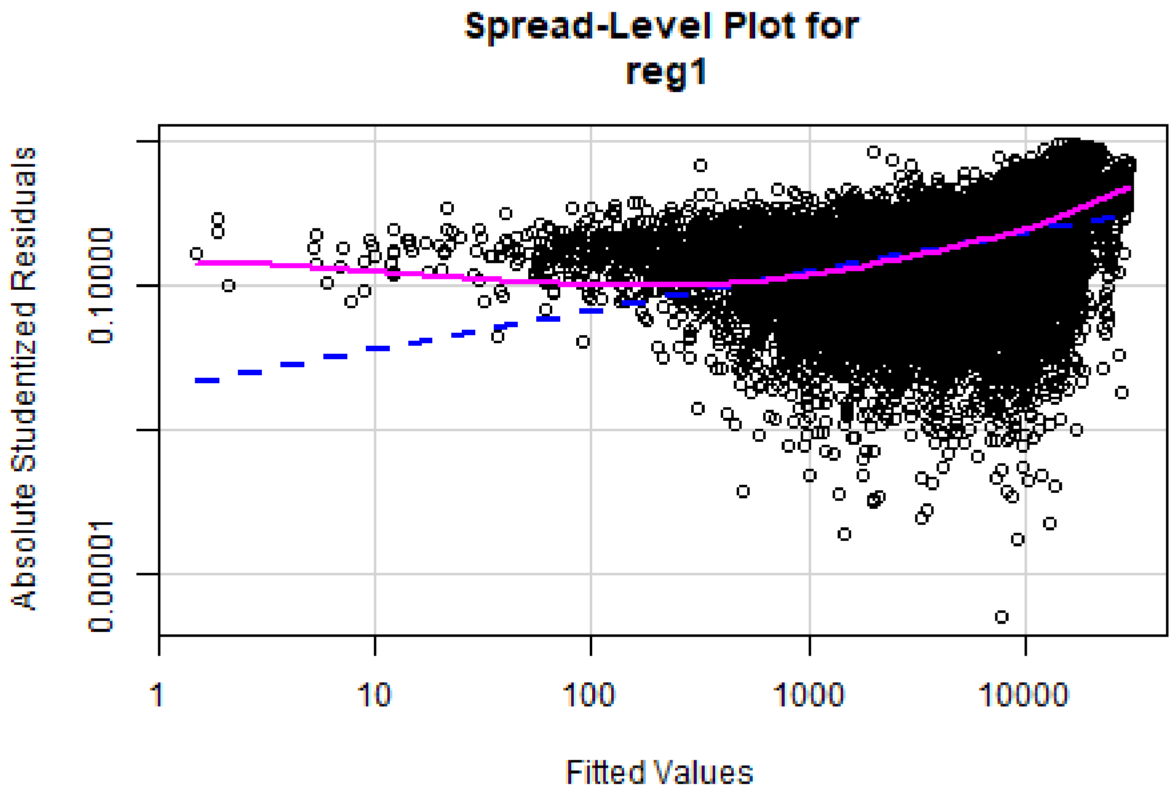
Test for Heteroskedasticity

```r
tidy(reg1)
```

```
## # A tibble: 11 x 5
##    term               estimate  std.error statistic   p.value
##    <chr>                 <dbl>      <dbl>     <dbl>     <dbl>
##  1 (Intercept)       9442.       82.6         114.  0.
##  2 odometer_value      -0.00606   0.000163     -37.3 1.59e-298
##  3 engine_capacity   1243.       35.2          35.4 1.78e-269
##  4 age               -411.        2.90        -142.  0.
##  5 number_of_photos    86.2       3.11          27.7 1.28e-167
##  6 diesel            1399.       41.0          34.1 7.12e-251
##  7 automatic          876.       49.4          17.7 5.80e- 70
##  8 drivetrain_all    1897.       78.5          24.2 3.36e-128
##  9 suv               1886.       76.1          24.8 2.02e-134
## 10 luxury            2223.       48.2          46.1 0.
## 11 new              10263.      176.           58.3 0.
```

```r
spreadLevelPlot(reg1)
```

```
## Warning in spreadLevelPlot.lm(reg1):
## 3031 negative fitted values removed
```




```
##
## Suggested power transformation:  0.4604066
```

```
bptest(reg1)
```

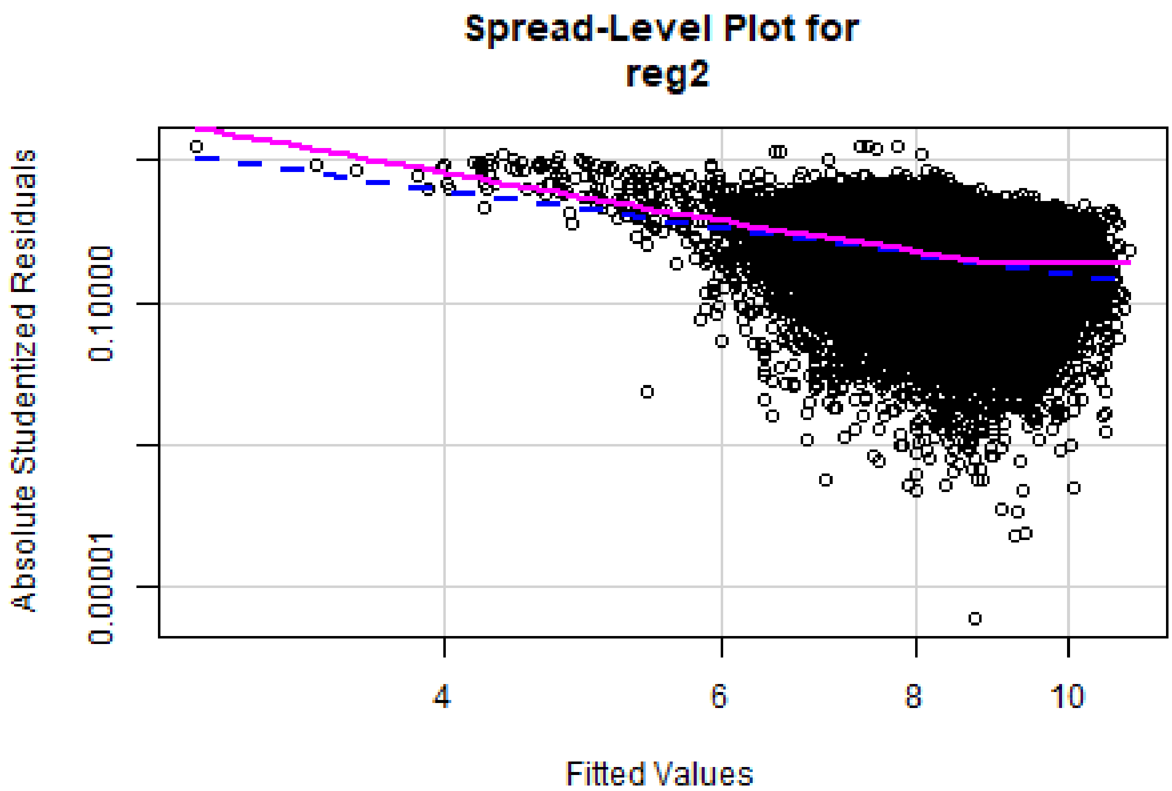
```
##
##  studentized Breusch-Pagan test
##
## data:  reg1
## BP = 3488.3, df = 10, p-value < 0.00000000000000022
```

```
tidy(reg2)
```

```
## # A tibble: 11 x 5
##    term                  estimate    std.error statistic   p.value
##    <chr>                    <dbl>        <dbl>     <dbl>     <dbl>
##  1 (Intercept)        9.04        0.0112           809.    0.
##  2 odometer_value   -0.000000363 0.0000000220     -16.5   7.66e- 61
##  3 engine_capacity    0.265       0.00476          55.8   0.
##  4 age               -0.0925      0.000392       -236.    0.
##  5 number_of_photos   0.0107      0.000421         25.5   9.52e-142
##  6 diesel             0.271       0.00555          48.8   0.
##  7 automatic          0.121       0.00669          18.1   4.30e- 73
##  8 drivetrain_all     0.0595      0.0106            5.60  2.11e-  8
```

```
##  9 suv                0.184      0.0103          17.9  5.53e- 71
## 10 luxury             0.310      0.00652         47.5  0.
## 11 new                0.213      0.0238           8.95 3.69e- 19
```

```r
spreadLevelPlot(reg2)
```




```
##
## Suggested power transformation:  3.901786
```

```r
bptest(reg2)
```

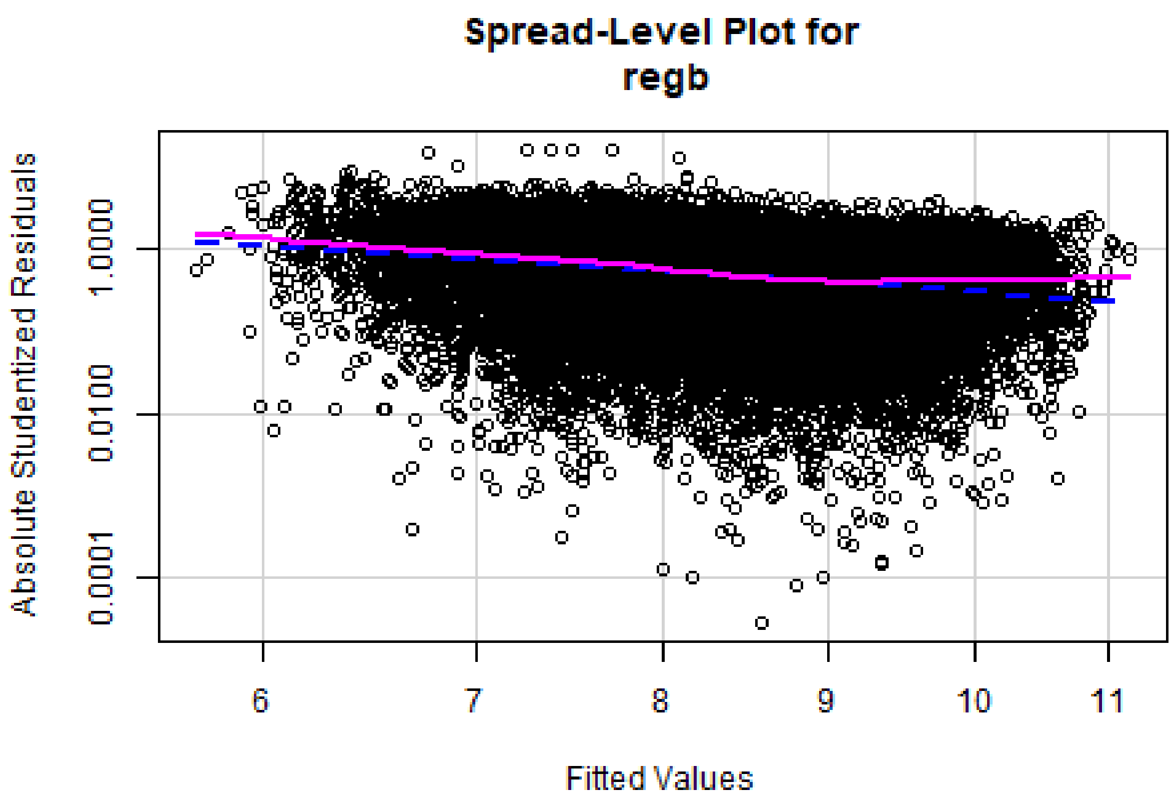
```
##
##  studentized Breusch-Pagan test
##
## data:  reg2
## BP = 3050.3, df = 10, p-value < 0.00000000000000022
```

```r
tidy(regb)
```

```
## # A tibble: 13 x 5
##    term               estimate std.error statistic   p.value
##    <chr>                 <dbl>     <dbl>     <dbl>     <dbl>
##  1 (Intercept)        9.33e+ 0  1.21e- 2    773.   0.
```

```
## 2 odometer_value     9.04e- 7  5.06e- 8      17.9  4.43e- 71
## 3 odo2              -1.09e-12  5.99e-14     -18.3  2.26e- 74
## 4 engine_capacity    2.94e- 1  4.55e- 3      64.6  0.
## 5 age               -1.59e- 1  1.15e- 3    -139.   0.
## 6 age2               1.69e- 3  2.75e- 5      61.3  0.
## 7 number_of_photos   8.73e- 3  4.00e- 4      21.8  4.56e-105
## 8 diesel             2.54e- 1  5.31e- 3      47.8  0.
## 9 automatic          1.04e- 1  6.34e- 3      16.3  7.66e- 60
## 10 drivetrain_all    6.93e- 2  1.01e- 2       6.89 5.58e- 12
## 11 suv               1.40e- 1  9.80e- 3      14.3  2.13e- 46
## 12 luxury            2.89e- 1  6.19e- 3      46.6  0.
## 13 new              -6.47e- 2  2.34e- 2      -2.77 5.59e-  3
```

```r
spreadLevelPlot(regb)
```




```
##
## Suggested power transformation:  3.538945
```

```r
bptest(regb)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  regb
## BP = 1962.7, df = 12, p-value < 0.00000000000000022
```

Since 2.2e-16 < 0.05, we reject the null hypothesis and conclude that heteroskedasticity is present.

```r
newregb<-coeftest(regb, vcov = vcovHC(regb, type = "HC0"))
newregb
```

```
##
## t test of coefficients:
##
##                               Estimate           Std. Error  t value
## (Intercept)       9.326148428239983801  0.012987673343553873 718.0769
## odometer_value    0.000000903847555756  0.000000068746965550  13.1475
## odo2             -0.000000000001094516  0.000000000000081175 -13.4834
## engine_capacity   0.293962290503553103  0.004844160110921305  60.6839
## age              -0.159402291100304849  0.002161716319814830 -73.7388
## age2              0.001686823006147536  0.000063072394750179  26.7442
## number_of_photos  0.008731398773549276  0.000375942392536317  23.2254
## diesel            0.253921099698344033  0.005423292289674700  46.8205
## automatic         0.103653525178686007  0.005408119879047162  19.1663
## drivetrain_all    0.069335356954123309  0.008105826691633450   8.5538
## suv               0.140345722838804088  0.007469685577317735  18.7887
## luxury            0.288582752313655488  0.005876295514993452  49.1096
## new              -0.064727264356851777  0.016259757601401782  -3.9808
##                               Pr(>|t|)
## (Intercept)      < 0.00000000000000022 ***
## odometer_value   < 0.00000000000000022 ***
## odo2             < 0.00000000000000022 ***
## engine_capacity  < 0.00000000000000022 ***
## age              < 0.00000000000000022 ***
## age2             < 0.00000000000000022 ***
## number_of_photos < 0.00000000000000022 ***
## diesel           < 0.00000000000000022 ***
## automatic        < 0.00000000000000022 ***
## drivetrain_all   < 0.00000000000000022 ***
## suv              < 0.00000000000000022 ***
## luxury           < 0.00000000000000022 ***
## new                          0.0000688 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

To mitigate this problem, robust standard errors will be used for the model, results of which can be seen in the table above.

Test for Model Misspecification

```r
resettest(reg1 , type="regressor")
```

```
##
##  RESET test
##
## data:  reg1
## RESET = 655.88, df1 = 20, df2 = 38490, p-value < 0.00000000000000022
```

```r
resettest(reg2 , type="regressor")
```

```
##
##  RESET test
##
## data:  reg2
## RESET = 371.81, df1 = 20, df2 = 38490, p-value < 0.00000000000000022
```

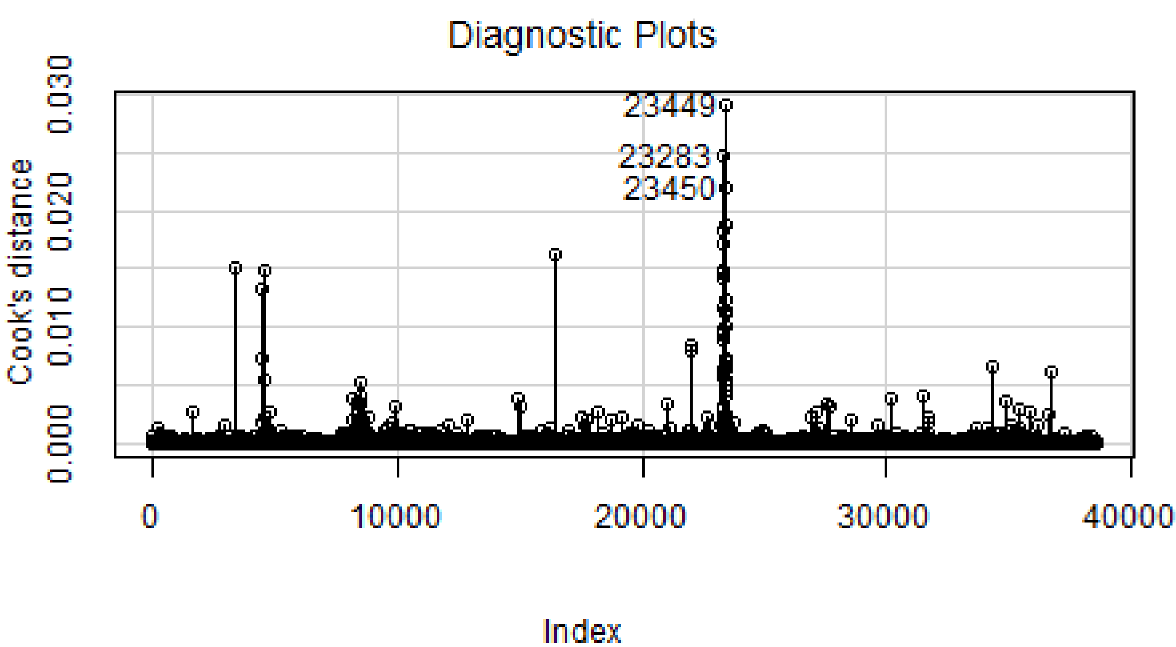
```r
resettest(regb , type="regressor")
```

```
##
##  RESET test
##
## data:  regb
## RESET = 164.36, df1 = 24, df2 = 38481, p-value < 0.00000000000000022
```

According to the test, all models have problems of model misspecification, we need to improve the model.

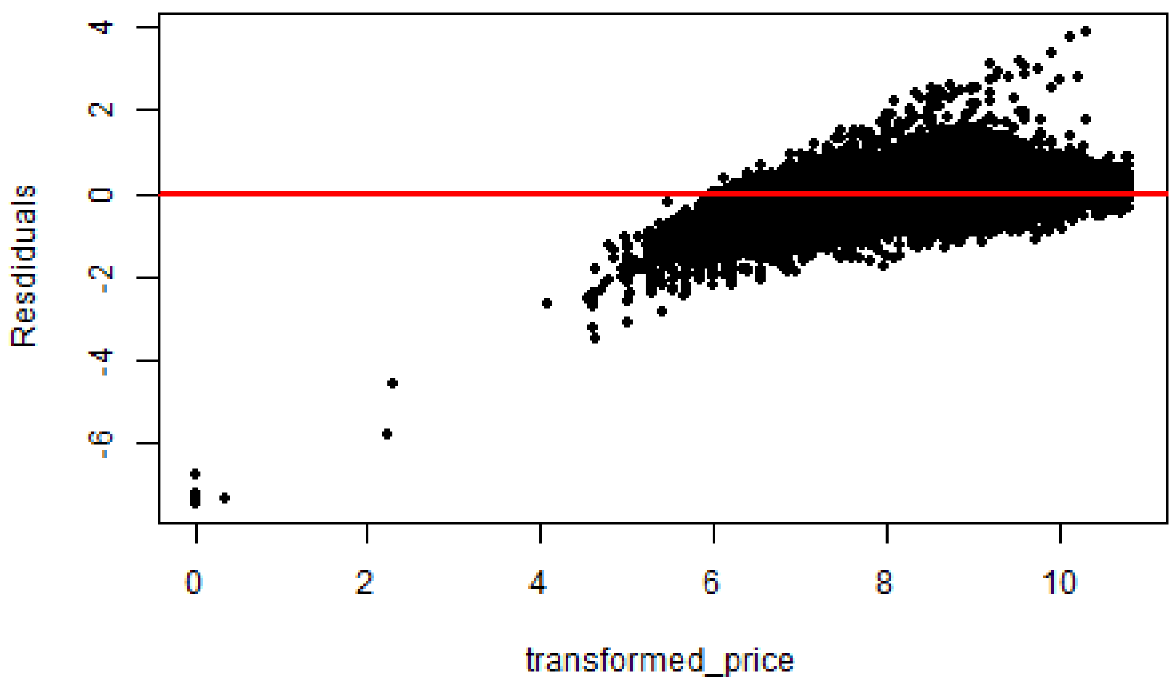
Cook's Distance Plot, Residual Plot

```r
influenceIndexPlot(regb, id=list(n=3),vars="Cook")
```



It looks like 23282, 23446 and 23447 are potential outliers, but all of the variables are within 0.03 of Cook's distance. Therefore, they do not need to be removed.

```r
#lm(cars$transformed_price~cars$odometer_value+cars$odo2+cars$engine_capacity+
#cars$number_of_photos+cars$diesel+cars$new+cars$luxury+cars$age+cars$age2+
#cars$suv+cars$automatic+cars$drivetrain_all)
cars_new <-subset(cars,select=c(transformed_price, odometer_value, odo2,
                                engine_capacity, age, age2, number_of_photos,
                                diesel, automatic, drivetrain_all, suv, luxury,
                                new))

plot(cars_new$transformed_price, regb$residuals, pch=20,ylab="Resdiduals",
     xlab="transformed_price")
abline(h=0,col="red", lwd=2)
```



It seems that the model has some issues with the cars that have very low and very high sale prices. For the low sale price cars, the model overestimates the sale price (negative residuals) and for high sale price cars, the model underestimates the sale price (positive residuals). However, towards the center of the sale price data the model performs relatively well. The spread in residuals seems constant here.

AIC and BIC

```r
AIC(reg1,reg2,regb)
```

```
##      df       AIC
## reg1 12 739144.00
## reg2 12  52848.31
```

```
## regb 14  48686.95
```

```
BIC(reg1,reg2,regb)
```

```
##      df       BIC
## reg1 12 739246.70
## reg2 12  52951.01
## regb 14  48806.77
```

As expected, Regb is the preferred choice.

Bootstrapping the Model

```
trans.mod = lm(transformed_price ~ odometer_value + odo2 + engine_capacity + age
               + age2 + number_of_photos
         + diesel + automatic + drivetrain_all + suv + luxury + new, data=cars)
betahat.boot = Boot(trans.mod, R=999)
usualEsts = summary(trans.mod)
summary(betahat.boot)
```

```
##
## Number of bootstrap replications R = 999
##                            original              bootBias
## (Intercept)       9.3261484282399838  0.0002235460935118283O
## odometer_value    0.0000009038475558 -0.0000000030469861713
## odo2             -0.0000000000010945  0.0000000000000056592
## engine_capacity   0.2939622905035531  0.0000272220804703460
## age              -0.1594022911003048 -0.0001451238801972177
## age2              0.0016868230061475  0.0000034048890320431
## number_of_photos  0.0087313987735493  0.00001354462009974487
## diesel            0.2539210996983440 -0.00032004558984366493
## automatic         0.1036535251786860 -0.00017003462025078075
## drivetrain_all    0.0693353569541233 -0.00004272330621057918
## suv               0.1403457228388041 -0.00004142002977602921
## luxury            0.2885827523136555  0.00008308214902874589
## new              -0.0647272643568518 -0.00005135127670909123
##                             bootSE            bootMed
## (Intercept)       0.012891720899430756  9.3262645710249963
## odometer_value    0.000000069566256901  0.0000009015598572
## odo2              0.000000000000082571 -0.0000000000010951
## engine_capacity   0.004946994419951000  0.2938991299796474
## age               0.002145781149237901 -0.1595155180338580
## age2              0.000062694349265010  0.0016899341486935
## number_of_photos  0.000385336649207392  0.0087487866805617
## diesel            0.005373841828897041  0.2537555955758399
## automatic         0.005367501808366094  0.1035112507841282
## drivetrain_all    0.007609717263969986  0.0691236351256663
## suv               0.007184447647613929  0.1405816589125978
## luxury            0.005770420397577211  0.2884570843110476
## new               0.016452115855679256 -0.0645162307492351
```

```r
confint(betahat.boot)
```

```
## Warning in confint.boot(betahat.boot): BCa method fails for this problem. Using
## 'perc' instead
```

```
## Bootstrap percent confidence intervals
##
##                                  2.5 %                97.5 %
## (Intercept)       9.300877501881865683  9.352306932695034547
## odometer_value    0.000000770115010215  0.000001036484881626
## odo2             -0.000000000001256538 -0.000000000000932749
## engine_capacity   0.284009296310763271  0.303852734690802484
## age              -0.163471445119473952 -0.155001369163692754
## age2              0.001557297025995580  0.001809373689538485
## number_of_photos  0.007930215891480324  0.009492676023535328
## diesel            0.243217029866002660  0.263923145082203880
## automatic         0.092950794714899712  0.113792827974729921
## drivetrain_all    0.054356278699150939  0.085124644767824706
## suv               0.125256315529731127  0.154601947042548982
## luxury            0.276975597214749625  0.300130487694851844
## new              -0.097391411329693298 -0.033792780631832095
```

The summary gives the original sample value for each component of the bootstrapped statistics, along with the bootstrap estimates of bias, the difference between the average bootstrapped value of the statistic and its original-sample value. The bootstrap estimates of standard error are computed as the standard deviation of the bootstrap replicates. These values are used to construct normal-theory confidence intervals for the regression coefficients.

There is a separate histogram for each bootstrapped quantity, here each coefficient. In addition to the histograms we also get kernel density estimates and the normal density based on the bootstrap mean and standard deviation. The vertical dashed line makes the original point-estimate, and the thick horizontal line gives a confidence interval based on the bootstrap. Whereas the two density estimates for the intercept are similar, the normal approximation is poor for the other coefficients, and confidence intervals are not close to symmetric about the original values.

We can see that regb fits very well.

```r
# Bootstrap for the estimated residual standard deviation:
sigmahat.boot <- Boot(regb, R=199, f=sigmaHat, labels="sigmaHat")

summary(sigmahat.boot)
```

```
##            R original    bootBias    bootSE bootMed
## sigmaHat 199  0.45515 -0.00052683 0.0043586 0.45456
```

```r
confint(sigmahat.boot)
```

```
## Warning in confint.boot(sigmahat.boot): BCa method fails for this problem. Using
## 'perc' instead
```

```
## Bootstrap percent confidence intervals
##
##             2.5 %    97.5 %
## sigmaHat 0.446973 0.4640195
```

The 95% confidence interval was 0.4496041-0.4688543, and the standard error was 0.019252.

Cross-validation

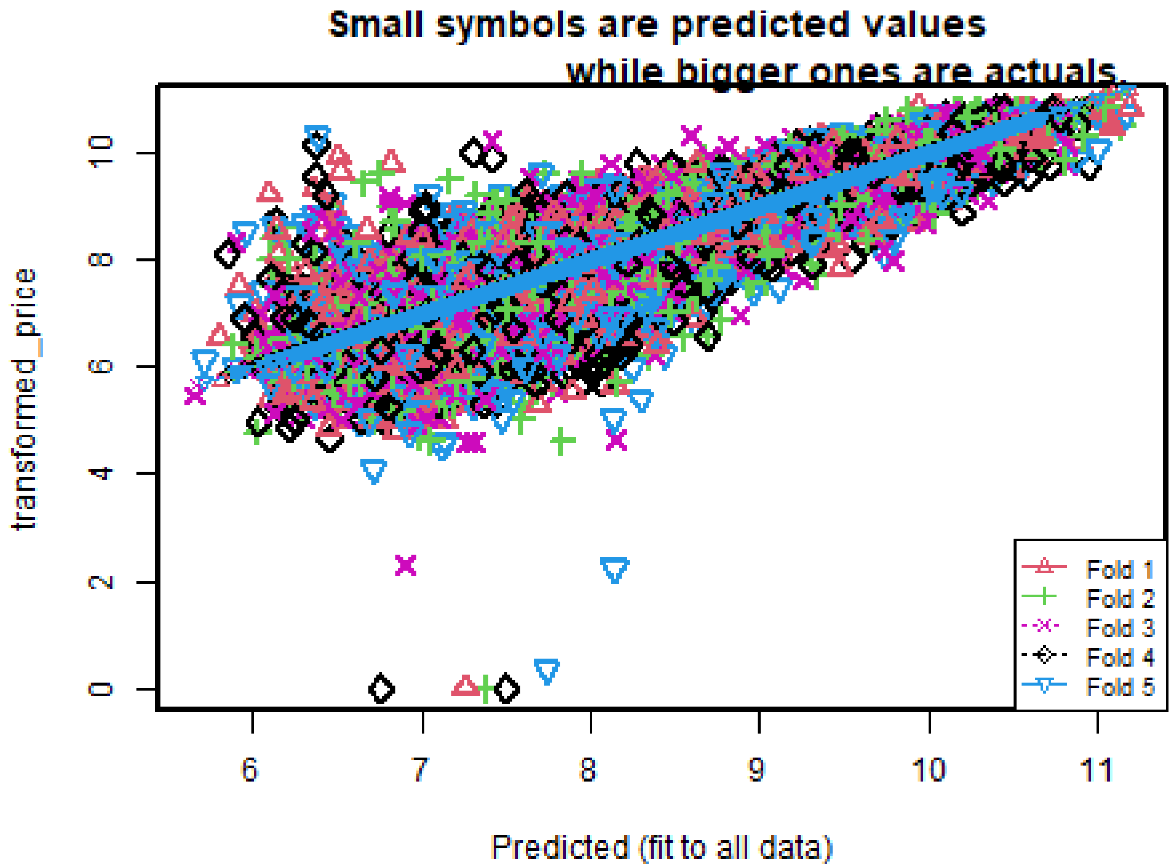
```r
train_control<- trainControl(method="cv", number=5, savePredictions = TRUE,
                             returnResamp = 'all')
model <- train(x=cars_new[,c(2,3,4,5,6,7,8,9,10,11,12,13)], y=cars_new[,1],
               trControl=train_control, method="rpart")
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo, :
## There were missing values in resampled performance measures.
```

```r
model
```

```
## CART
##
## 38518 samples
##    12 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 30813, 30816, 30815, 30814, 30814
## Resampling results across tuning parameters:
##
##   cp          RMSE       Rsquared   MAE
##   0.05487188  0.6235281  0.6306171  0.4762467
##   0.06249037  0.6517242  0.5963380  0.5004982
##   0.54056007  0.8291502  0.5369024  0.6574933
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was cp = 0.05487188.
```

```r
cvResults <- suppressWarnings(CVlm(data=as.data.frame(cars_new),
                                   form.lm=formula(transformed_price~
                                                     odometer_value+odo2+
                                                     engine_capacity
                                                   +number_of_photos+diesel+new+
                                                     luxury+age+age2+suv),
                                   m=5, dots=FALSE, legend.pos="bottomright",
                                   printit=FALSE,
                                   main="Small symbols are predicted values
                                   while bigger ones are actuals."))
```




```r
attr(cvResults, 'ms')
```

```
## [1] 0.2090873
```

The regb's prediction accuracy is not varying too much for any one particular sample, and the lines of best fit don't vary too much with respect the the slope and level.

Bootstrapping

```r
suppressMessages(library("tidyverse"))
training.samples <- cars$price_usd %>%
createDataPartition(p = 0.8, list = FALSE)
train.data <- cars[training.samples, ]
test.data <- cars[-training.samples, ]

# build the model
##reg1
reg1<-lm(price_usd ~ odometer_value + engine_capacity + age + number_of_photos
         + diesel + automatic + drivetrain_all + suv + luxury + new, data=cars)
##reg2
reg2<-lm(transformed_price ~ odometer_value + engine_capacity + age +
           number_of_photos + diesel + automatic + drivetrain_all + suv + luxury
         + new, data=cars)
```

```
##regb
regb<-lm(transformed_price ~ odometer_value + odo2 + engine_capacity + age + age2
         + number_of_photos + diesel + automatic + drivetrain_all + suv + luxury
         + new, data=cars)

# make predictions
predictions1 <- reg1 %>% predict(test.data)
predictions2 <- reg2 %>% predict(test.data)
predictionsb <- regb %>% predict(test.data)

# model performance
data.frame(RMSE = RMSE(predictions1, test.data$price_usd),
           R2 = R2(predictions1, test.data$price_usd))
```

```
##       RMSE        R2
## 1 3489.164 0.7010991
```

```
data.frame(RMSE = RMSE(predictions2, test.data$transformed_price),
           R2 = R2(predictions2, test.data$transformed_price))
```

```
##        RMSE        R2
## 1 0.4783017 0.7821586
```

```
data.frame(RMSE = RMSE(predictionsb, test.data$transformed_price),
           R2 = R2(predictionsb, test.data$transformed_price))
```

```
##        RMSE        R2
## 1 0.4578603 0.8003768
```

We can see that the regb has the largest R^2 and smallest RMSE, which means that regb fits the data better than reg1 and reg2. So, the regb is the best choice.

By testing the three models from different angles including multicollinearity, heteroskedasticity, model misspecification, Cook's distance Plot, Residuals Plot, AIC, BIC, robustness and cross-validation, we confirm that regb is the best choice, and by by splitting the data into testing and training sets, we got satisfied results on prediction.

Interpretation of the Final Model with Robust Standard Errors (newregb)

```
round(newregb,6)
```

```
##
## t test of coefficients:
##
##                    Estimate Std. Error  t value             Pr(>|t|)
## (Intercept)        9.326148   0.012988 718.0769 < 0.00000000000000022 ***
## odometer_value     0.000001   0.000000  13.1475 < 0.00000000000000022 ***
## odo2               0.000000   0.000000 -13.4834 < 0.00000000000000022 ***
## engine_capacity    0.293962   0.004844  60.6839 < 0.00000000000000022 ***
```

```
## age              -0.159402   0.002162 -73.7388 < 0.00000000000000022 ***
## age2              0.001687   0.000063  26.7442 < 0.00000000000000022 ***
## number_of_photos  0.008731   0.000376  23.2254 < 0.00000000000000022 ***
## diesel            0.253921   0.005423  46.8205 < 0.00000000000000022 ***
## automatic         0.103654   0.005408  19.1663 < 0.00000000000000022 ***
## drivetrain_all    0.069335   0.008106   8.5538 < 0.00000000000000022 ***
## suv               0.140346   0.007470  18.7887 < 0.00000000000000022 ***
## luxury            0.288583   0.005876  49.1096 < 0.00000000000000022 ***
## new              -0.064727   0.016260  -3.9808            0.000069 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The interpretation of the continuous variables must include their squared terms (if applicable). For example, the effect of an additional year of age on sale price would be the derivative of the equation with respect to age which would be: -0.159402 + 2(0.001687)(AGE). Unfortunately, in this model the effect of odometer value seems to negligible.

The baseline or reference car is a used non-suv, non-luxury brand car with diesel engine, automatic gear shift, and front or rear drivetrain. The indicator variables can all be interpreted along the same way: if the variable equals 1 ($/X_k = 1$), we expect the sale price of the respective car to increase on average by $100\beta_k\%$, all else equal. For example, all else equal, compared to a non-luxury brand car, the model predicts a luxury brand car to have a 28.85% higher sale price on average.

All of our variables are highly statistically significant and all their signs make economic sense (with the exception of the "new" variable). For instance, all else equal, we would expect an automatic car to have a higher sale price compared to the reference car which the model confirms. Similarly, all else equal, on average, we would expect a SUV to have a higher sale price compared to the reference car which the model confirms as well.

References

Boudette, Neal E. 2020. "Looking to Buy a Used Car in the Pandemic? So Is Everyone Else." *The New York Times*, Sep 07, 2020. https://www.nytimes.com/2020/09/07/business/used-cars-pandemic.html

D'Allegro, Joe. 2020. "Just What Factors Into The Value Of Your Used Car?" *Investopedia*, Feb 20, 2020. https://www.investopedia.com/articles/investing/090314/just-what-factors-value-your-used-car.asp.

Domonoske, Camila. 2020. "A Pandemic Sticker Shock: Used-Car Prices Are Through The Roof." *NPR*, Oct 28, 2020. https://www.npr.org/2020/10/28/927971920/a-pandemic-sticker-shock-used-car-prices-are-through-the-roof.

Erdem, Cumhur and Ismail Şentürk. 2009. "A Hedonic Analysis of Used Car Prices in Turkey." *International Journal of Economic Perspectives* 2009, 3(2): 141-149.

Lepchenkov, Kirill. 2019. "Used-cars-catalog." *Kaggle*, Dec 02, 2019. https://www.kaggle.com/lepchenkov/usedcarscatalog.

Li, Xin, Mengyue Wang, and Yubo Chen. 2014. "The Impact of Product Photo on Online Consumer Purchase Intention: An Image-Processing Enabled Empirical Study." *PACIS 2014 Proceedings*, 325.

Pal, Nabarun, Priya Arora, Sai Sumanth, Dhanasekar Sundararaman, and Puneet Kohli. 2018. "How Much Is My Car Worth? A Methodology for Predicting Used Cars Prices Using Random Forest." *Future of Information and Communications Conference (FICC).*