Gregorio Del Rio

ECON 494-02R-FA20

Professor Steven Levkoff

Nov. 18, 2020

Car Price Prediction

Source

The dataset used for this project is "Car Price Prediction Multiple Linear Regression." This dataset was downloaded from Kaggle and uploaded by Manish Kumar. I downloaded this dataset on Nov.11, 2020.

Executive Summary

This dataset has 205 observations. Each observation has 26 variables when initially downloaded. There are 10 categorical variables and 16 quantitative variables. One categorical variable built into the data set is CarName. I will not use this variable since the model of car maker has a huge range, however, I will manually input the Make of each model to bring the total of categorical variables to 11, and total variables to 27.

I plan to use the variables described below to build four different linear regression models to predict the price of a car.

Categorical variables

Carbody:	Labels the car on whether it's a convertible, hardtop, hatchback, sedan, or wagon.
Make:	The brand name of company which makes the car.

Quantitative Variables

Carlength:	Measurement of how long the car is in inches.
Curbweight:	The weight of a car measure in pounds (lbs).
Enginesize:	The size of the engine by its displacement.
Highwaympg:	Gas mileage efficiently when driving on the highway.
Horsepower:	A unit of measure of a cars power.
Price:	The for-sale price of a car.

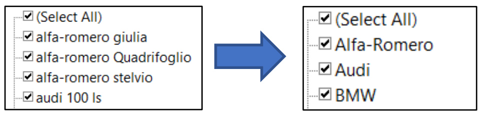
Goals

1. I would like to see the makeup of this data, so first I will see the count of cars by each Maker.
2. I will make scatter plots of price to each variable to see if we can identify any relationships.
3. Build a correlation between all quantitative variables to gauge positive and negative relationships.

4. I plan to build four models, 2 of which will require higher order polynomials, and 1 using logarithmic variables. To better organize my code, I will build these additional variables before the models.
    a. Model 1: Linear regression using variables provided by the data set.
    b. Model 2: Model 1 plus variables to the second power.
    c. Model 3: Model 2 plus variables taken to the third power.
    d. Model 4: Linear regression with original variables and log variables.
5. Assign 70% of the data in-sample training data, and 30% for out-of-sample testing data.
6. Model 1. Create a linear regression with all variables to see which are significant.
7. Model 2: Create a linear regression with all variables and variables taken to the second power.
8. Model 3: Repeat 6 but take quantitative variables to the third power.
9. Model 4: Logarithmic variables.
10. Put each Model's RMSE in-sample and out-of-sample error into a Dataframe.
11. Conclusion

Cleaning Data for Analysis

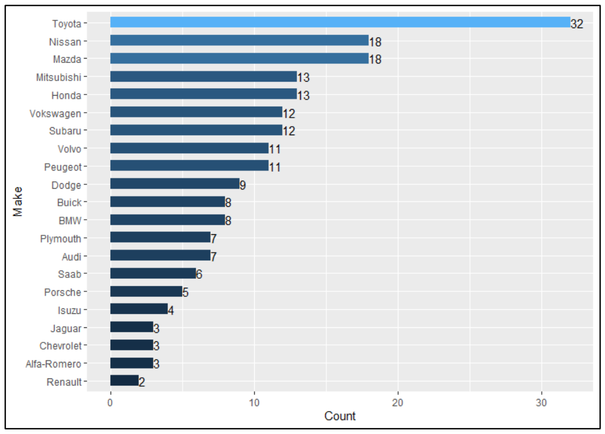
Fortunately, this dataset didn't need much cleaning. The only adjustment I made was adding a "make" variable representing the maker of the car/observation. Originally, the "make" of each car/observation was within the "CarName" variable. The CarName variable had both the maker and model of each car/observation. To better organize my code, I made a variable dedicated only to the "make" of each observation.



☑ (Select All)
☑ alfa-romero giulia
☑ alfa-romero Quadrifoglio
☑ alfa-romero stelvio
☑ audi 100 ls

☑ (Select All)
☑ Alfa-Romero
☑ Audi
☑ BMW

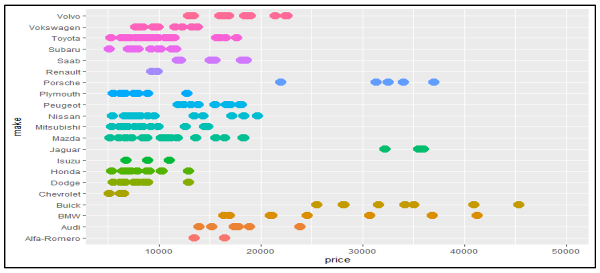
1. Number of Cars in Each Model



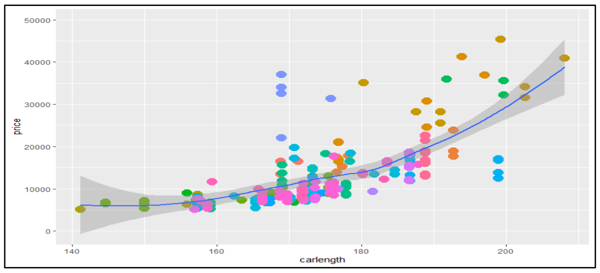
We can see the amount of observations (cars) for each car maker. Notably, Toyota makes up about 15% of the total dataset, meaning, the models created will be best at predicting prices for Toyotas and worst at predicting Renaults since there are only two Renault observations.
The make variable will be used when building models since each car maker tends to offer their cars at a certain price range.

2. Scatter Plot of Price to Each Variable

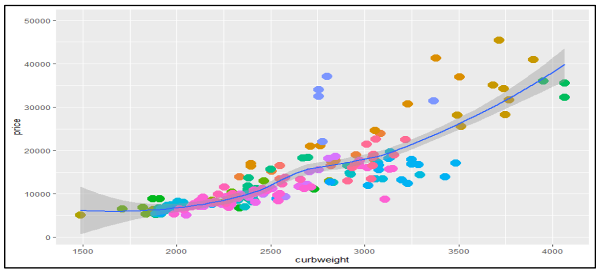
Price to Make (Legend)



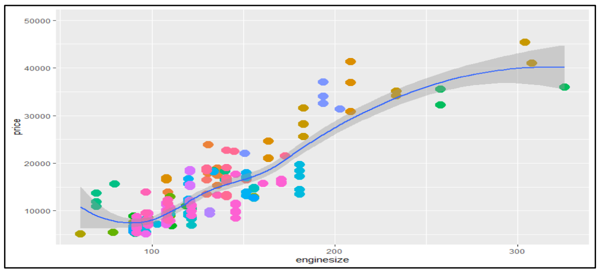
Price to Car Length



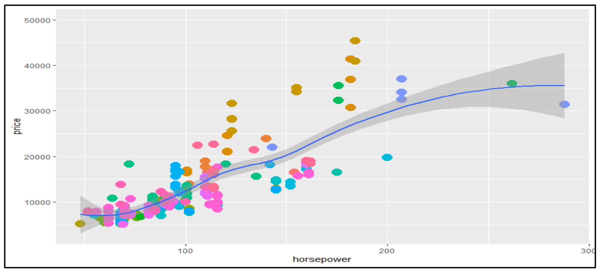
Price to Curb Weight



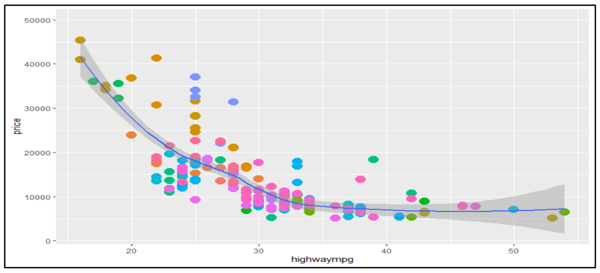
Price to Engine Size



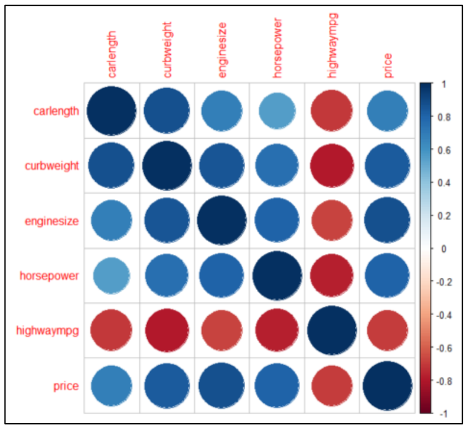
Price to Horsepower

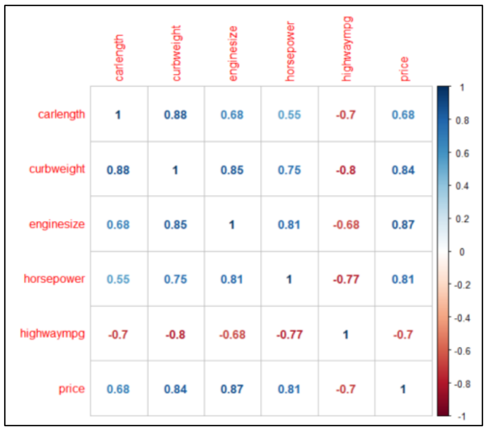


Price to Highway mpg



3. Column Variables Correlation





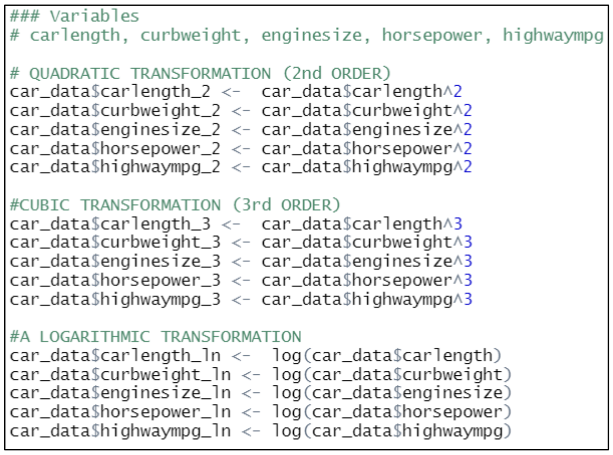
	carlength	curbweight	enginesize	horsepower	highwaympg	price
carlength	1	0.88	0.68	0.55	-0.7	0.68
curbweight	0.88	1	0.85	0.75	-0.8	0.84
enginesize	0.68	0.85	1	0.81	-0.68	0.87
horsepower	0.55	0.75	0.81	1	-0.77	0.81
highwaympg	-0.7	-0.8	-0.68	-0.77	1	-0.7
price	0.68	0.84	0.87	0.81	-0.7	1

I have reduced the amount of variables used in each model as a result of many variables being extremely correlated to one another. I removed variables to remove the likelyhood of multicollinearity obstructing my models. The variables I left to be used for the models are the ones in the correlationplots above. We can see from the corrleation plots above there are many variables which have positive and negative correlations with each other.

4. Creating Higher Order Polynomials and Logarithmic Variables

Variables Taken to the Second and Third Power:

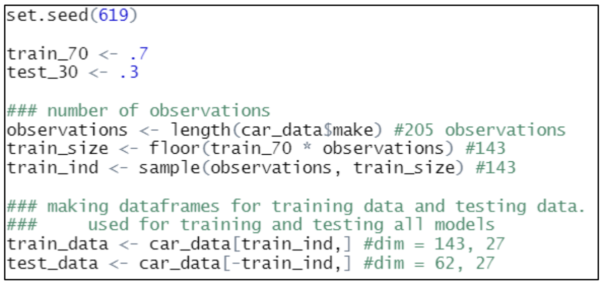
- carlength, curbweight, enginesize, horsepower, and highwaympg



```
### Variables
# carlength, curbweight, enginesize, horsepower, highwaympg

# QUADRATIC TRANSFORMATION (2nd ORDER)
car_data$carlength_2 <-  car_data$carlength^2
car_data$curbweight_2 <- car_data$curbweight^2
car_data$enginesize_2 <- car_data$enginesize^2
car_data$horsepower_2 <- car_data$horsepower^2
car_data$highwaympg_2 <- car_data$highwaympg^2

#CUBIC TRANSFORMATION (3rd ORDER)
car_data$carlength_3 <-  car_data$carlength^3
car_data$curbweight_3 <- car_data$curbweight^3
car_data$enginesize_3 <- car_data$enginesize^3
car_data$horsepower_3 <- car_data$horsepower^3
car_data$highwaympg_3 <- car_data$highwaympg^3

#A LOGARITHMIC TRANSFORMATION
car_data$carlength_ln <-  log(car_data$carlength)
car_data$curbweight_ln <- log(car_data$curbweight)
car_data$enginesize_ln <- log(car_data$enginesize)
car_data$horsepower_ln <- log(car_data$horsepower)
car_data$highwaympg_ln <- log(car_data$highwaympg)
```

I will take the quadratic, cubic, and logarithmic form of each quantitative variable before building the models to better organize my R-Script. The intent of using these higher order polynomials is to improve the prediction of variables who don't follow a straight linear trend.

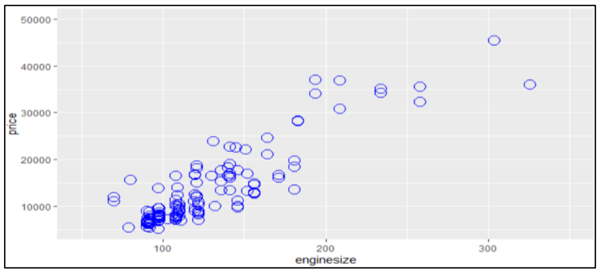
5. Partition Training and Testing Data



```
set.seed(619)

train_70 <- .7
test_30 <- .3

### number of observations
observations <- length(car_data$make) #205 observations
train_size <- floor(train_70 * observations) #143
train_ind <- sample(observations, train_size) #143

### making dataframes for training data and testing data.
###     used for training and testing all models
train_data <- car_data[train_ind,] #dim = 143, 27
test_data <- car_data[-train_ind,] #dim = 62, 27
```

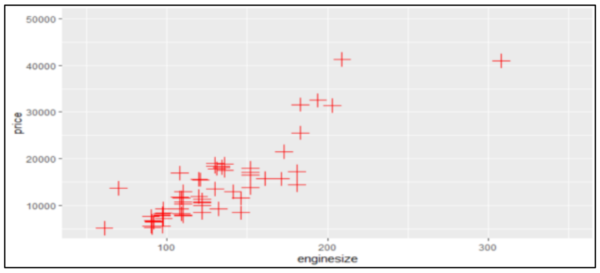
First, I'll use the seed function to make running my models over and over consistent. I then use the floor and sample function to help me create two separate dataframes for my training and testing data. Once my dataframes are made, I can then create scatterplots to display the training and testing data.

Training Data



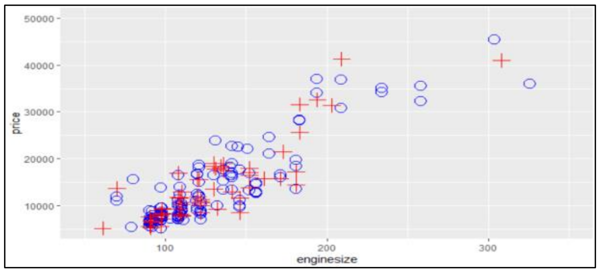
Of the 205 observations, 143 (69.75%) of them have been assigned to be used for training the four models. Each model will use the same training data in order to generate accurate comparisons of each model against each other.

Testing Data



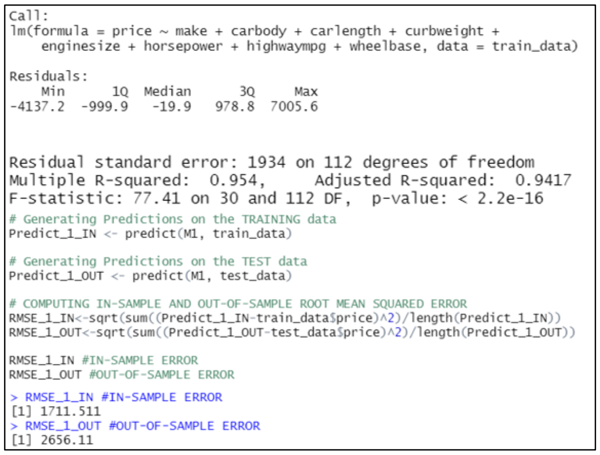
Of the 205 observation, 62 (30.25%) of them have been assigned to be used for testing all the models. After each model has been tested with this data, we can calculate each models Root Mean Square Error (RMSE) to gauge how accurate each model can predict car prices.

Training and Testing Data



In this scatter plot, we can see both the training and testing data. Viewing this scatter plot, we can visually see the spread of the testing data over the training data. This informs us the testing data is not do overly aligned with a section of the training data.

6. Model 1: Variables to Predict Price

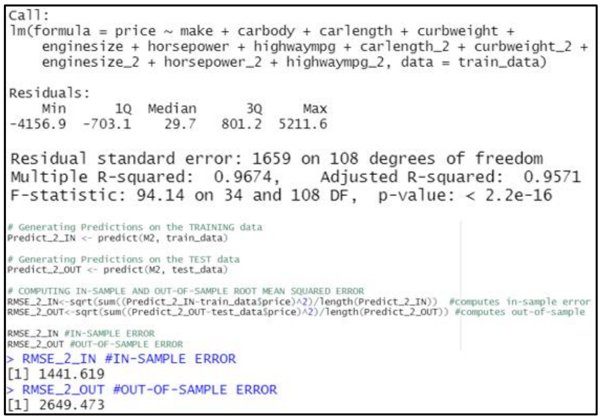


```
Call:
lm(formula = price ~ make + carbody + carlength + curbweight +
    enginesize + horsepower + highwaympg + wheelbase, data = train_data)

Residuals:
    Min      1Q  Median      3Q     Max
-4137.2  -999.9   -19.9   978.8  7005.6


Residual standard error: 1934 on 112 degrees of freedom
Multiple R-squared:  0.954,     Adjusted R-squared:  0.9417
F-statistic: 77.41 on 30 and 112 DF,  p-value: < 2.2e-16
# Generating Predictions on the TRAINING data
Predict_1_IN <- predict(M1, train_data)

# Generating Predictions on the TEST data
Predict_1_OUT <- predict(M1, test_data)

# COMPUTING IN-SAMPLE AND OUT-OF-SAMPLE ROOT MEAN SQUARED ERROR
RMSE_1_IN<-sqrt(sum((Predict_1_IN-train_data$price)^2)/length(Predict_1_IN))
RMSE_1_OUT<-sqrt(sum((Predict_1_OUT-test_data$price)^2)/length(Predict_1_OUT))

RMSE_1_IN #IN-SAMPLE ERROR
RMSE_1_OUT #OUT-OF-SAMPLE ERROR
> RMSE_1_IN #IN-SAMPLE ERROR
[1] 1711.511
> RMSE_1_OUT #OUT-OF-SAMPLE ERROR
[1] 2656.11
```

Model 1 only uses column variables provided with the dataset. I would expect this model to have the worst RMSE_OUT since it only uses provided data. Once I add higher order polynomials, these statistical measures should increase.

R-Squared:	0.954
RMSE_1_IN:	1711.511
RMSE_1_OUT:	2656.11

7. Model 2: Variables and Quadratic Variables

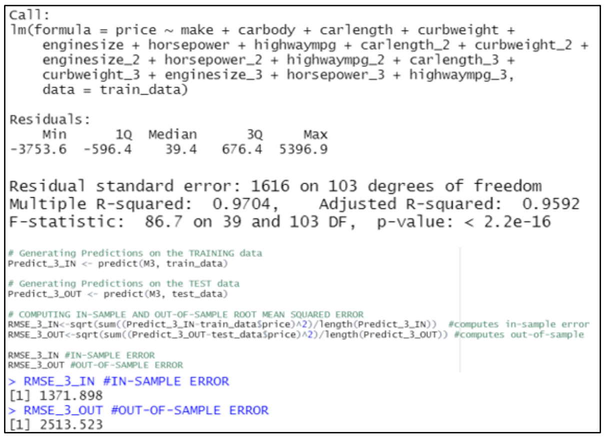


```
Call:
lm(formula = price ~ make + carbody + carlength + curbweight +
    enginesize + horsepower + highwaympg + carlength_2 + curbweight_2 +
    enginesize_2 + horsepower_2 + highwaympg_2, data = train_data)

Residuals:
    Min      1Q  Median      3Q     Max
-4156.9  -703.1    29.7   801.2  5211.6

Residual standard error: 1659 on 108 degrees of freedom
Multiple R-squared:  0.9674,     Adjusted R-squared:  0.9571
F-statistic: 94.14 on 34 and 108 DF,  p-value: < 2.2e-16

# Generating Predictions on the TRAINING data
Predict_2_IN <- predict(M2, train_data)

# Generating Predictions on the TEST data
Predict_2_OUT <- predict(M2, test_data)

# COMPUTING IN-SAMPLE AND OUT-OF-SAMPLE ROOT MEAN SQUARED ERROR
RMSE_2_IN<-sqrt(sum((Predict_2_IN-train_data$price)^2)/length(Predict_2_IN))  #computes in-sample error
RMSE_2_OUT<-sqrt(sum((Predict_2_OUT-test_data$price)^2)/length(Predict_2_OUT)) #computes out-of-sample

RMSE_2_IN #IN-SAMPLE ERROR
RMSE_2_OUT #OUT-OF-SAMPLE ERROR
> RMSE_2_IN #IN-SAMPLE ERROR
[1] 1441.619
> RMSE_2_OUT #OUT-OF-SAMPLE ERROR
[1] 2649.473
```

As expected, taking the quadratic form of each quantitative variable provided an increased R-Squared and a decreased RMSE_OUT. This is reasonable since based on the scatter plots in number 2, not all the variables follow a straight line. We can assume the addition of these quadratic variable means Model 2 is better than Model 1.

R-Squared:	0.9674
RMSE_2_IN:	1441.619
RMSE_2_OUT:	2649.473

8. Model 3: Variables, Quadratic, and Cubic Variables

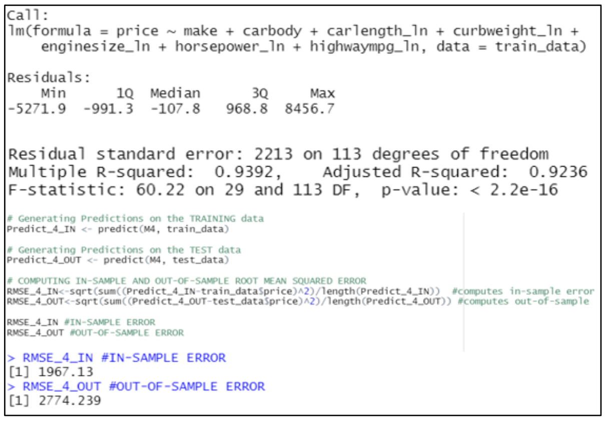


```
Call:
lm(formula = price ~ make + carbody + carlength + curbweight +
    enginesize + horsepower + highwaympg + carlength_2 + curbweight_2 +
    enginesize_2 + horsepower_2 + highwaympg_2 + carlength_3 +
    curbweight_3 + enginesize_3 + horsepower_3 + highwaympg_3,
    data = train_data)

Residuals:
    Min      1Q  Median      3Q     Max
-3753.6  -596.4    39.4   676.4  5396.9

Residual standard error: 1616 on 103 degrees of freedom
Multiple R-squared:  0.9704,    Adjusted R-squared:  0.9592
F-statistic:  86.7 on 39 and 103 DF,  p-value: < 2.2e-16

# Generating Predictions on the TRAINING data
Predict_3_IN <- predict(M3, train_data)

# Generating Predictions on the TEST data
Predict_3_OUT <- predict(M3, test_data)

# COMPUTING IN-SAMPLE AND OUT-OF-SAMPLE ROOT MEAN SQUARED ERROR
RMSE_3_IN<-sqrt(sum((Predict_3_IN-train_data$price)^2)/length(Predict_3_IN))  #computes in-sample error
RMSE_3_OUT<-sqrt(sum((Predict_3_OUT-test_data$price)^2)/length(Predict_3_OUT)) #computes out-of-sample

RMSE_3_IN #IN-SAMPLE ERROR
RMSE_3_OUT #OUT-OF-SAMPLE ERROR
> RMSE_3_IN #IN-SAMPLE ERROR
[1] 1371.898
> RMSE_3_OUT #OUT-OF-SAMPLE ERROR
[1] 2513.523
```

Using cubic variables increased the R-Squared from Model 2 and decreased the RMSE_OUT. Additionally, the Adj. R-Squared increased, meaning our model is considered less complex.

• R-Squared:	0.9704
• RMSE_3_IN:	1371.898
• RMSE_3_OUT:	2513.523

9. Model 4: Variables and Logarithmic Variables

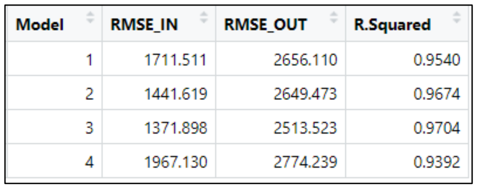


```
Call:
lm(formula = price ~ make + carbody + carlength_ln + curbweight_ln +
    enginesize_ln + horsepower_ln + highwaympg_ln, data = train_data)

Residuals:
    Min      1Q  Median      3Q     Max
-5271.9  -991.3  -107.8   968.8  8456.7

Residual standard error: 2213 on 113 degrees of freedom
Multiple R-squared:  0.9392,    Adjusted R-squared:  0.9236
F-statistic: 60.22 on 29 and 113 DF,  p-value: < 2.2e-16

# Generating Predictions on the TRAINING data
Predict_4_IN <- predict(M4, train_data)

# Generating Predictions on the TEST data
Predict_4_OUT <- predict(M4, test_data)

# COMPUTING IN-SAMPLE AND OUT-OF-SAMPLE ROOT MEAN SQUARED ERROR
RMSE_4_IN<-sqrt(sum((Predict_4_IN-train_data$price)^2)/length(Predict_4_IN))  #computes in-sample error
RMSE_4_OUT<-sqrt(sum((Predict_4_OUT-test_data$price)^2)/length(Predict_4_OUT)) #computes out-of-sample

RMSE_4_IN #IN-SAMPLE ERROR
RMSE_4_OUT #OUT-OF-SAMPLE ERROR

> RMSE_4_IN #IN-SAMPLE ERROR
[1] 1967.13
> RMSE_4_OUT #OUT-OF-SAMPLE ERROR
[1] 2774.239
```

Converting the quantitative variables to logarithmic variables produced the lowest R-Squared and the largest RMSE_OUT. Of the 4 models, Model 4 is the least accurate at predicting car prices.

• R-Squared:	0.9392
• RMSE_4_IN:	1967.13
• RMSE_4_OUT:	2774.239

10. Dataframe with Models and their RMSE and Prediction Plots



Model	RMSE_IN	RMSE_OUT	R.Squared
1	1711.511	2656.110	0.9540
2	1441.619	2649.473	0.9674
3	1371.898	2513.523	0.9704
4	1967.130	2774.239	0.9392

I built this Dataframe to easily compare the statistical performance of each model compared to each other. We can see Model 3 outperforms the other models in all areas. Where Model 4 performs the worst of the 4.

11. Conclusion

Based on the performance of the four models, we can conclude Model 3 performs the best when predicting car prices, based on its RMSE out-of-sample error. This is somewhat expected since it is provided the raw data given with the dataset and higher order polynomial variables to better predict non-linear trends. From the dataframe above, we can see on average Model 3's In-Sample Error is about $1,371.898 off when predicting the price of a car, and the Out-of-Sample Error is about $2,513.523 off.

Since Root Mean Square Error (RMSE) provides us with the same units of measurement as the output variable (price), we can better gauge how off our out-of-sample error is compared to the in-sample error.