Predicting Used Car Prices with Advanced Regression Techniques

University of Washington

Autumn 2021, IND E 427

17 December 2021

Andy Kim, Ekin Ugurel, Nick Shawger

Table of Contents

Abstract

We predict used car prices using a range of data analytics methods such as linear regression (LR) and random forests (RF), while also using methods like LASSO to determine the factors that are most significant in predicting the price. To create this model, we analyze a dataset containing used car listings based in the U.S. on craigslist.com. This dataset includes 25 independent variables for each listing as well as the selling price for 427,000 unique vehicles, containing a wide range of vehicle ages, manufacturers, and mileage. After data preprocessing, we proceed with 13 variables and roughly 276,000 unique listings. Using LASSO, we find that the most important factors that determine used car prices are (in order of decreasing importance): Odometer, Year, Transmission, and Cylinders. Further, we find that a random forest model boasts the highest accuracy at 92.7 percent, while a sparse linear regression model boasts an accuracy of 72 percent.

Introduction

Determining whether the listed price of a used car is reasonable is a tall task given the range of factors that affect a vehicle's value. Being more knowledgeable in the sensibility of listings in a used car market is important for both buyers and sellers. Buyers want to neither overpay nor buy a vehicle not up to their specification standards. Sellers want to price vehicles appropriately to not hold inventory too long.

Used car prices have significantly increased in the past two years due to supply chain shortages caused by the onset of COVID-19. As of October 2021, car prices in the United States have increased by 26% with a huge 32.7% spike in July compared to two years prior (Smith 2021). The focus of our project is to help car-buyers make more informed decisions during this turbulent economic period.

Some challenges that we encountered were the overwhelming number of factors that determine the prices of cars. We preprocessed these factors to omit any unnecessary variables that did not have a significant impact on the used car prices. The prices are pulled from listings on Craigslist and the Kaggle dataset, hence there is no certainty whether the prices in the data are the listed price or the price sold.

Dataset

The Kaggle dataset has the shape 426,880 x 26, providing enough entries to adequately train and internally test our model. Although the original dataset includes 26 columns (25 input variables and 1 output), our analysis is conducted with only 13 of these variables. We remove columns that either had a large proportion of missing values in the dataset (> 50%) or contained information not relevant to the price of a given listing. We removed the following variables despite having few missing values: 'ID' (arbitrary string of numbers), 'region_url' (website), 'url' (website), 'image_url' (website), 'lat' (latitude), 'long' (longitude), 'description' (subjective Craigslist description), 'posting_date' (date of listing), 'title_status' (clean or missing).

Data Preprocessing

In this section, we describe our procedure to preprocess the data. We thank Jose Portillo (2020), who open-source preprocessed a similar dataset, for teaching us various Pandas methods. First, we check for the percent of missing values in every column and remove those with too many, namely 'VIN', 'county', and 'size'. We avoid removing columns like 'cylinders', 'condition', and 'drive' (powertrain), as they are likely closely correlated with price.

In [5]:
```
# Check for missing values
null_count = pd.DataFrame({'Null': df.isnull().sum()})
# Check for percent of values missing
length=len(df)
percent_null = round((null_count['Null']/length)*100,1)
null_count['Percentage'] = percent_null
# Sort from highest percentage to lowest
null_count.sort_values(by='Null', ascending=False)
```

Out[5]:

	Null	Percentage
county	426880	100.0
size	306361	71.8
cylinders	177678	41.6
condition	174104	40.8
VIN	161042	37.7
drive	130567	30.6
paint_color	130203	30.5

Figure 1: Code to determine percent of missing values

We then drop duplicate rows and remove inconsistencies in data entries (e.g. spaces in the cell). This standardizes our entries for the model.

```
In [7]: # Check for duplicates
        df.duplicated().sum()

Out[7]: 56415

In [8]: # Drop duplicates and keep one of each
        df = df.drop_duplicates(keep='first')

In [9]: # remove inconsistent data entry (e.g. spaces in the cell)
        columns = ['manufacturer', 'condition','cylinders','fuel','tit

        for i in columns:
            df[i] = df[i].str.strip()
```

Figure 2: Code to remove duplicates and inconsistencies

We move on to deal with outliers. We begin by dropping 10 percent of listings on the low- and high-side of price. This corresponds to dropping any listing with a price below $1,200 and above $37,740.

```
In [11]: # Drop 10% of each side on price (outliers)
         sort = sorted(df['price'])
         q1, q2 = np.percentile(sort, [10,90])
         print(q1, q2)

         1200.0 37740.0

In [12]: df = df[(df.price <= 37740.0) & (df.price >= 1200)]
         df.shape

Out[12]: (296798, 16)
```

Figure 3: Code to remove 0-10 and 90-100 percentiles in price

We then eliminate outliers based on the odometer. A quick scatter plot shows exactly what we want to prune:

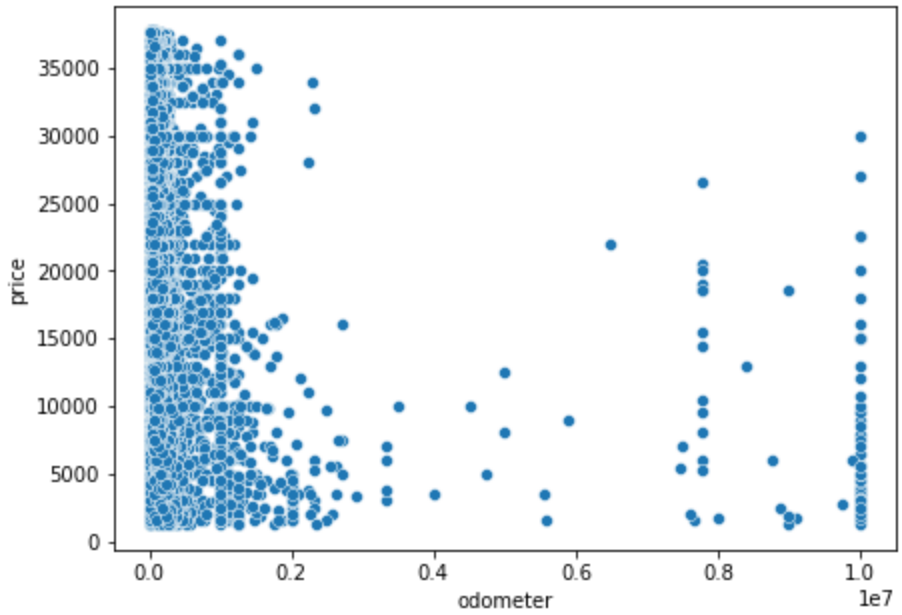

Figure 4: Scatter plot of odometer readings

We eliminate listings with an odometer greater than 300,000 miles, which we deem as obsolete. We also eliminate listings with an odometer reading of zero, as we are concerned with used cars (not new). Next, we deal with outliers in 'Year'. A quick scatter plot yields the following:

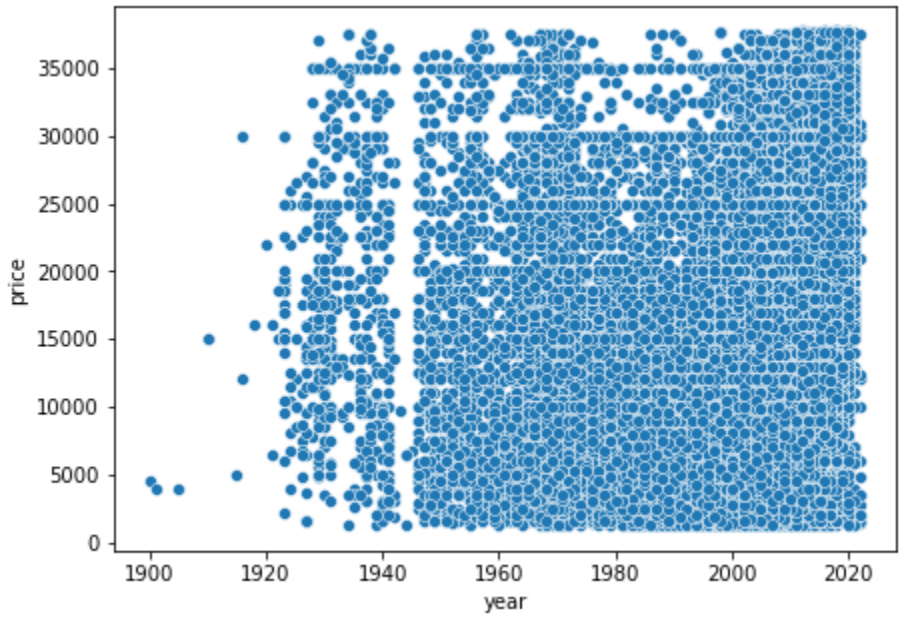

Figure 5: Scatter plot of model year of listings

We eliminate listings with a model year prior to 1945. Checking the distributions of odometer and price again, we get the following graphs:

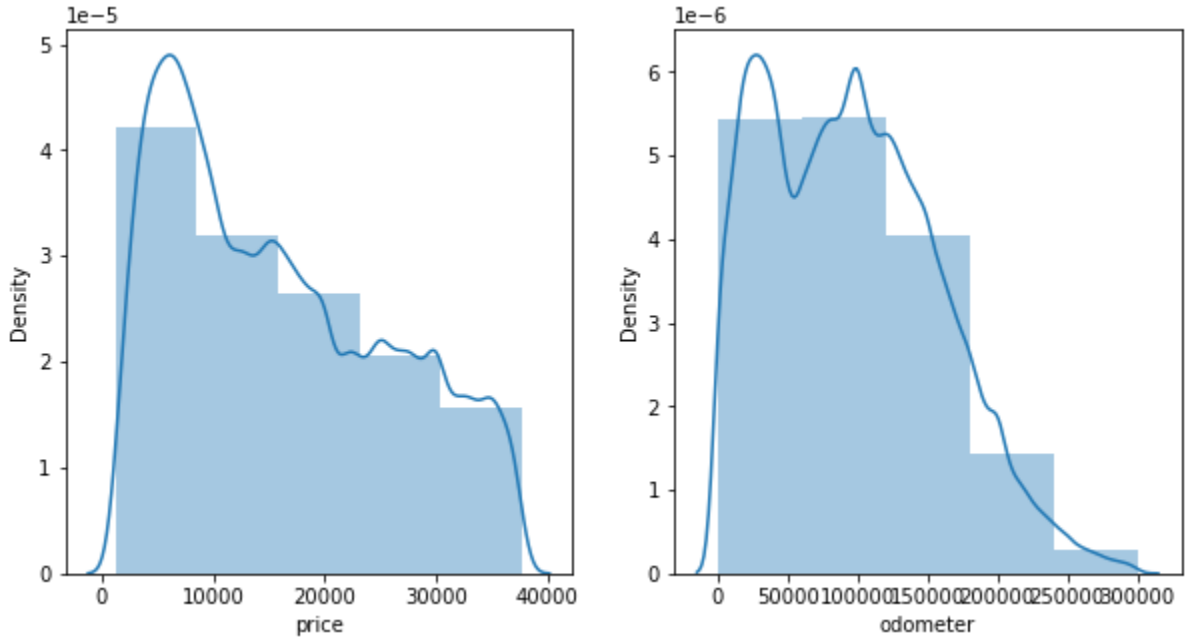

Figure 6: (a) Distribution of prices and (b) distribution of odometers

Next, we fill in missing values for columns like ‘cylinders’, for which 41.6 percent of values are not found. We infer missing values by looking at the existing distribution of condition for different bounds of odometer readings and selecting the median value.

```
In [24]: # Now, let's check the distribution for condition
         bins = [0,30000,60000,90000,115000,150000,1000000]
         groups = df.groupby(['condition', pd.cut(df.odometer,bins)])
         groups.size().unstack()
```

Out[24]:

odometer	(0, 30000]	(30000, 60000]	(60000, 90000]	(90000, 115000]	(115000, 150000]	(150000, 1000000]
condition						
excellent	4606	8272	13169	14048	16917	15101
fair	252	188	348	583	948	2916
good	30487	20022	12201	8393	10928	17649
like new	2719	2670	2692	2374	2563	1839
new	233	69	82	53	75	74
salvage	29	40	38	50	63	135

```
In [25]: # Replace missing entries with the median condition for each odometer level
g1 = (df['odometer'] > 60000) & (df['odometer'] <= 150000)
g2 = (df['odometer'] <= 60000) | (df['odometer'] > 150000)

df.loc[g1,'condition']=df.loc[g1,'condition'].fillna('excellent')
df.loc[g2,'condition']=df.loc[g2,'condition'].fillna('good')
```

Figure 7 and 8: Code to fill in missing values of condition

We use a similar process to fill in missing values for ‘cylinders’, ‘drive’, ‘type’, and ‘paint_color’. Finally, before extracting our pruned dataset for analysis, we create a factor correlation heat map to ensure no two factors are closely correlated with each other. This would make one of the factors redundant and add unnecessary computational load on our model.

Figure 9: Factor Correlation Heat Map

As no factors are strongly correlated (either positively or negatively) with another, we keep the dataset as is. To show the difference between the pre-pruned and pruned dataset, we can look at the differences in the preliminary statistics for the price of a used car:

Table 1: Statistical comparison between raw and preprocessed dataset

	Pre-Pruned Dataset	**Pruned Dataset**
Count	426,880 cars	275,754 cars
Mean	$75,199.03	$16,157.68
Standard Deviation	$12,182,280.00	$10,013.39
Minimum	$0.00	$1,200.00
25% Quartile	$5,900.00	$7,450.00
50% Quartile	$13,950.00	$14,500.00
75% Quartile	$26,857.50	$23,999.99
Maximum	$3,736,929,000.00	$37, 740.00

Methodology

In this section, we describe our methodology in detail, beginning with feature selection (LASSO), then linear regression, and finally random forest prediction.

Model 1 - LASSO

First in our analysis, we use the least absolute shrinkage and selection operator (LASSO) method to determine the most significant factors that influence price. We use the *glmnet* library on R to create the path solution trajectory, shown below in Figure 10.

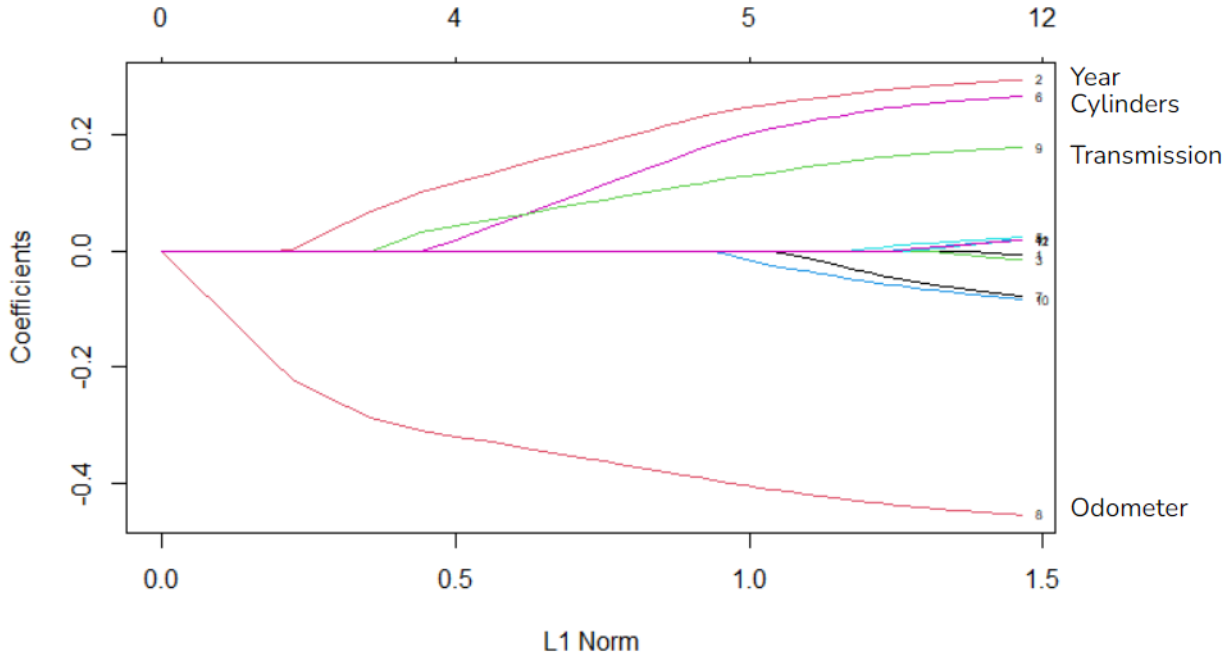

Figure 10: Path solution trajectory of factors

The path solution trajectory visualizes the coefficient solutions for a continuum of values of λ. Based on our analysis, the most important factors that determine used car prices are (in order of decreasing importance): Odometer, Year, Transmission, and Cylinders. Further, we use the command *cv.glmnet()* to obtain a cross-validation result, which identifies the best λ value for

the LASSO model. Unfortunately, the CV plot we obtain (Figure 11) does not have the desired U-shape, and thus is not helpful in determining the ideal value of λ.

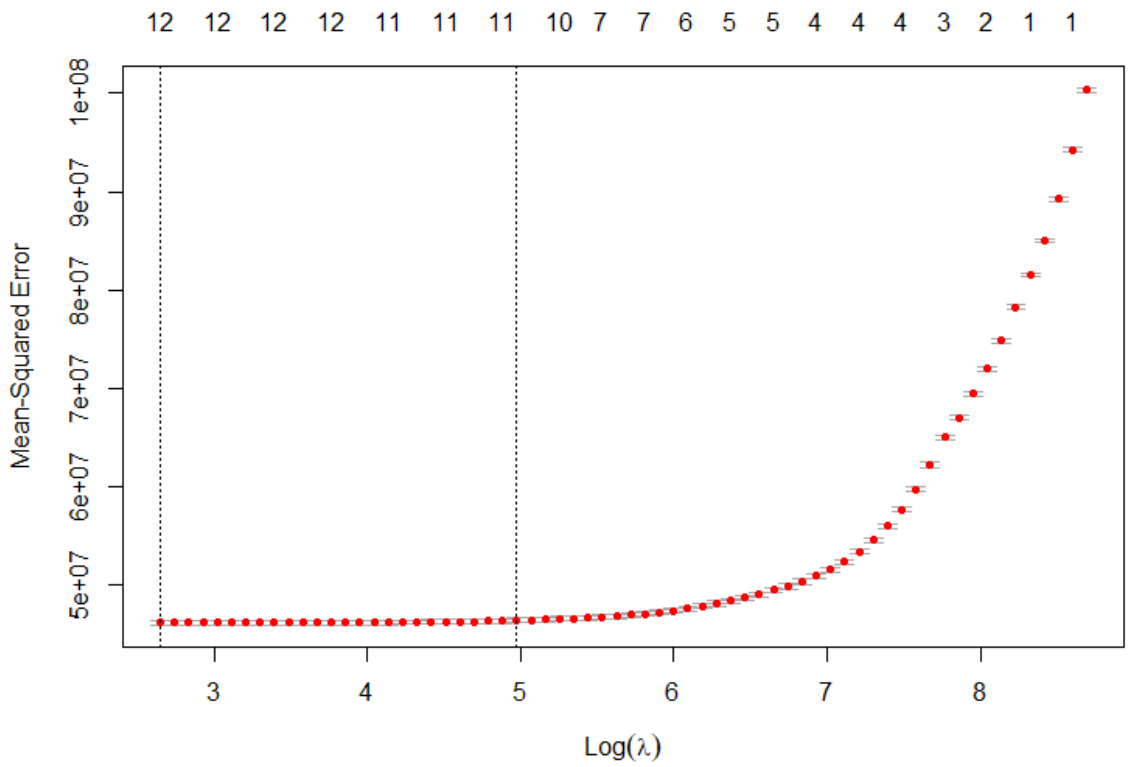

Figure 11: Cross-validation result

Model 2 - Linear Regression

The first model that we use is linear regression, as price is a continuous numerical variable. Once we preprocessed our data and got rid of unnecessary variables that didn't affect the cost, we had 13 variables remaining as shown here on the left side. We ended up getting an R^2 value of 0.534 and an accuracy of 73.4%. Even though it is shown here that all the values are statistically significant, the overall correlation is between the 0.4-0.7 benchmark, so there is positive correlation between the price and the dependent variables, but it is not strong. The original model with 13 independent variables is shown below.

```
Call:
lm(formula = price ~ ., data = data.train)

Residuals:
   Min     1Q Median     3Q    Max
-29289  -4613   -896   3843  40154

Coefficients:
                    Estimate      Std. Error  t value             Pr(>|t|)
(Intercept)  -709879.8654730    7109.9892022  -99.843 < 0.0000000000000002 ***
region            -0.8209133       0.2088712   -3.930   0.0000849452580869 ***
year             360.7102136       3.5234889  102.373 < 0.0000000000000002 ***
manufacturer     -13.6624645       2.3285347   -5.867   0.0000000044467525 ***
model              0.0425826       0.0050038    8.510 < 0.0000000000000002 ***
condition        191.1892324      25.7468071    7.426   0.0000000000001134 ***
cylinders       2168.7145468      21.6022298  100.393 < 0.0000000000000002 ***
fuel           -1243.3955949      41.0759125  -30.271 < 0.0000000000000002 ***
odometer          -0.0741417       0.0005067 -146.310 < 0.0000000000000002 ***
transmission    2368.0531974      40.3061875   58.752 < 0.0000000000000002 ***
drive          -1180.4121754      35.7731800  -32.997 < 0.0000000000000002 ***
type              49.6158594       6.6227864    7.492   0.0000000000000688 ***
paint_color       44.7341594       6.3482700    7.047   0.0000000000018496 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 6827 on 68925 degrees of freedom
Multiple R-squared:  0.534,     Adjusted R-squared:  0.534
F-statistic:  6583 on 12 and 68925 DF,  p-value: < 0.00000000000000022
```

Figure 12: Linear regression model

Based on the significant factors that our LASSO model was able to find, we ran a more concise model that consisted of the four key independent variables. The regression results are shown below.

```
Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)  -5.529e+01  3.825e-01 -144.55   <2e-16 ***
year          3.225e-02  1.895e-04  170.15   <2e-16 ***
cylinders     1.629e-01  1.190e-03  136.89   <2e-16 ***
transmission  1.647e-01  3.694e-03   44.58   <2e-16 ***
odometer     -5.735e-06  2.717e-08 -211.04   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.5326 on 137872 degrees of freedom
Multiple R-squared:  0.5136,    Adjusted R-squared:  0.5136
F-statistic: 3.639e+04 on 4 and 137872 DF,  p-value: < 2.2e-16
```

Figure 13: Pruned linear regression model

Our revised model indicates that newer cars and more cylinders in engines correlate with higher prices. Automatic transmission vehicles are also more expensive than other options such as manual stick shifts. Lastly, the more miles shown in the odometer, the lower the value of the car. It is also worth noting that the predictive correlation in this model was 72% which was only

1 percent less than the previous model. This shows that many of the variables that we used before were not as significant, and the significance levels shown in R are due to the vast amount of data that we were working with. The LASSO model proved to be effective showing us the more impactful factors when it came to used car prices and helping us find a better regression model. Below are some results of our second revised model.

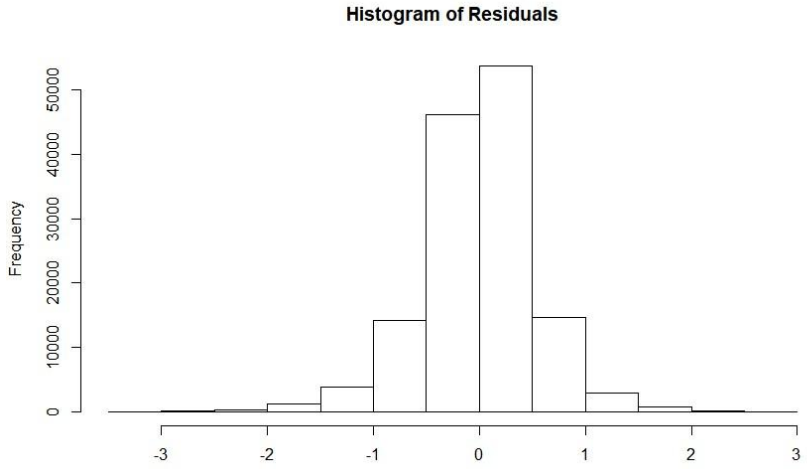

Figure 14: Histogram of residuals

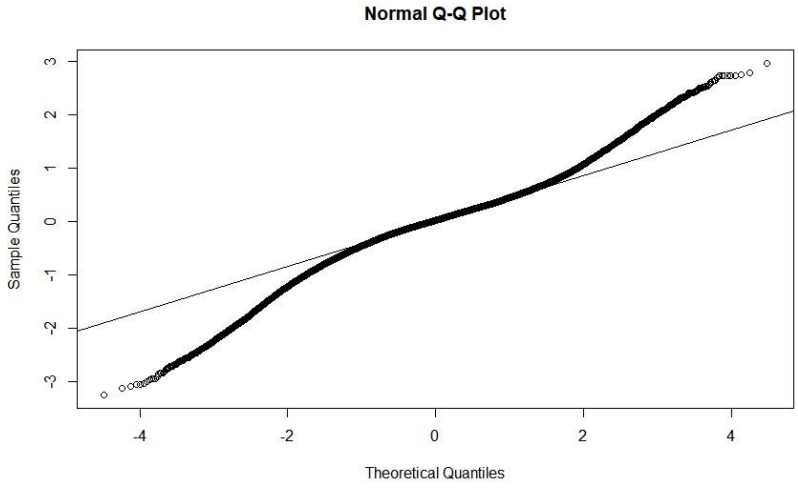

Figure 15: Normal Q-Q Plot

Model 3 - Random Forest

The second prediction model we created was a random forest model of our pre-processed data. This type of model is particularly useful for creating a prediction interface that can generate accurate output predictions based on your inputs. Because of this, it made sense to use this model to predict the price of a vehicle based on the various other factors of the vehicle like model, transmission, and year. Furthermore, when designing a client that could tell users whether or not a listing is a good deal, this model's prediction could be used as the primary comparison point.

The model was generated using these R commands:

1. Library for model generation: `library(randomForest)`
2. Model generation.
    a. Note parameters: 100 trees with a minimum size of 20 at each terminal node and 5 variables randomly sampled as candidates at each split.

```
rf.car <- randomForest( price ~ ., data = data.train, ntree = 100, nodesize = 20, mtry = 5)

rf.car
```

```
##
## Call:
##  randomForest(formula = price ~ ., data = data.train, ntree = 100,      nodesize = 20, mtry = 5)
##                Type of random forest: regression
##                      Number of trees: 100
## No. of variables tried at each split: 5
##
##          Mean of squared residuals: 14031115
##                    % Var explained: 86.01
```

These parameters were chosen because they resulted in the highest percentage of variables explained. We attempted to generate models with more trees and splits but they would have long generation times and did not show evidence of model improvement. With this prediction model, we achieved an accuracy statistic of **86% variance explained.**

3. Prediction on Testing Data

```
y_hat <- predict(rf.car, data.test,type="response")

A <- data.test$price
B <- y_hat
dataError <- data.frame(A, B, ((A - B)/A) * 100)

percentError <- dataError[3]
percentError <- t(percentError)

averagePercentError <- mean(percentError)
averagePercentError
```

```
## [1] -15.83611
```

Finally we used these commands here to generate an average percent error for the predictions made by our model. The average percent error of our model was approximately -16%. This can be interpreted as our model predicting cars to be **16% cheaper on average than they actually are**. However, this type of statistic is heavily affected by even relatively few outliers which could have resulted due to bugged individual predictions.

Because of this concern, we also decided to look at the correlation between predicted and real values. This is another prediction evaluation statistic that can be calculated using the following command: `cor(y_hat, data.test$price)`

Our calculated correlation value was 92.7%, which is a very strong correlation between predicted and actual values. This gives us more general insight into how difficult it would be to differentiate between our model's generated data and real world data.

Our conclusions from this model were very positive overall. All 3 of our evaluation statistics show that our model can generate predictions with very high accuracy across a wide range of factors. It is also worth noting that this model was generated using approximately 135,000 vehicles as reference. This means that it is a very thorough model that is not held back by region or time period specificity. Aside from some exceptions, we are confident this model could be used to accurately determine the price of a vehicle based on its specifications.

Conclusion

Overall, we found the results of our models to be very promising. Our original goal was to create models that could be used by a program to help users determine if a vehicle listing they are interested in is a good deal. Based on our preliminary analysis, we've found our models to be very good at that. Our linear regression model turned out to be the least valuable, semi-accurately noting which factors have the biggest influence on car prices. However, after evaluating its predictions, we found that the random forest model tends to predict the prices more accurately. In this situation with these exact models, we would most likely decide to use the random forest model as the main model of our program.

This doesn't mean our linear regression model is useless though, it just means that it's primary job would be to provide an additional prediction to test against if our random forest model starts giving predictions that do not make sense. Similarly, our LASSO model would not provide direct predictions that are more valuable than the random forest, but it could be used to "sanity check" many questionable predictions. For example, our LASSO model found transmission type to be an especially important factor for determining car price. If we were to find that either our regression or random forest model was undervaluing vehicles with valuable transmission types, then we could subsequently add or decrease the weight of some factors until our overall model more accurately reflects the real world. This would not be possible however, if we hadn't used LASSO to determine the most valuable factors.

Given more time and resources, we are confident that we could use these models to create a very useful interface for users looking to buy vehicles. In reality, many similar programs and models probably already exist for sites like CarFax and Carvana which claim to "always get you the best price."

References

Huang, S. (n.d.). *Data Analytics*. IND E 427. Retrieved December 17, 2021, from https://analytics.shuaihuang.info/

Smith, K. (2021, October 18). *Why used car prices have soared 26% during the pandemic*. Orange County Register. Retrieved December 4, 2021, from https://www.ocregister.com/2021/10/15/why-used-car-prices-have-soared-during-the-pandemic/

Portillo, J. (2020, October 20). Used Car Prices Project. GitHub. Retrieved December 5, 2021, from https://github.com/josem-gp/Used-Car-Prices/blob/main/Used%20vehicles.ipynb