BUAN 6337.007 - Predictive Analytics with SAS
Final Project

Prepared for:
Dr. Sourav Chatterjee

Group 6:
Nishanth Damarancha | nxd180021
Aneesa Noorani | axn180021
Prem Kumar Pulluri | pxp190001
Rohith Selvarajan | rds130030
April Wu | cxw180004

April 23, 2020

Executive Summary:

There are countless used car dealerships around the country. Each dealership must decide on an ideal price for each vehicle, taking into account factors such as: the year the car was manufactured, odometer reading, manufacturer, transmission, etc. In this analysis, we analyzed a dataset from Kaggle to build an accurate predictive model. This dataset contained information on nearly 510,000 used cars within the United States from the years 1980 to 2021. We used a number of statistical techniques and multiple linear regression methods to build our model. We conclude that a polynomial multiple linear regression model most accurately predicts the price a used car will sell for.

A. Introduction & Problem Statement:

The data comes from Kaggle, and includes every used vehicle entry within the United States on Craigslist. The data contains information on car sales, such as: price, condition, manufacturer, latitude/longitude, and 18 other columns, which we will discuss in more detail in the Dataset Description section.

This data can be potentially used for: (1) dealers who want to buy cars for their inventory to sell later, (2) dealers who want to decide how to price their cars, (3) dealers who have to think about not just what they can afford but also what their customer based can afford, or (4) automotive manufacturers using used car data to predict what the customers' needs for producing new vehicles.
Dealers have many aspects that they need to consider for how to price the used cars that they are selling, so we decided to use this data to build a predictive model that can help dealers decide on a reasonable selling price.

B. Dataset Description:

This data set shows used vehicles for sale from Craigslist. It contains 25 columns with 509,577 rows.

- *'ID'*: the unique ID number for each used vehicle
- *'URL'*: the listing URL link
- *'region'*: Craigslist region (i.e., Albany, Salt Lake City, etc.)
- *'region_URL'*: the URL for a certain 'region'
- *'Price'*: the entry price of a vehicle
- *'Year'*: the entry year
- *'Manufacturer'*: the manufacturers of a vehicle (i.e., Honda, Ford, etc.)
- *'Model'*: the model of a vehicle (i.e., Elantra, Civic, Tundra, etc.)
- *'Condition'*: the condition of the vehicle (i.e., excellent, like new, etc.)
- *'Cylinders'*: number of cylinders a vehicle has
- *'Fuel'*: fuel type (i.e., gas or diesel)
- *'Odometer'*: the miles traveled by a vehicle
- *'Title_status'*: whether a vehicle is clean (no accident) or rebuilt
- *'Transmission'*: whether a car is automatic or manual
- *'VIN'*: the VIN number of a vehicle
- *'Drive'*: whether a car is front-wheel drive, rear-wheel drive, or four-wheels drive
- *'Size'*: the size of a vehicle (i.e., full size, mid size, sub-compact, etc.)
- *'Type'*: the type of a vehicle (i.e., sedan, truck, pickup, suv, etc.)
- *'Paint_color'*: the color of a vehicle (i.e., white, silver, etc.)
- *'Image_URL'*: the link of the image
- *'Description'*: description of a vehicle
- *'County'*: a useless column that Kaggle accidentally left in by mistake

- *'State'*: the state of listing
- *'Lat'*: latitude
- *'Long'*: longitude

C. Data Preprocessing:

We preprocessed the data in a series of 5 steps:

1) Unnecessary column deletion

First, we dropped variables which we knew were not integral to our analysis and model building. We dropped specific variables for the following reasons:

- Since we are building an aggregate model, we are not interested in columns that identify rows: '*ID'*, '*VIN'*
- For the purposes of this report, we are not performing text analysis. So, we deleted such columns: '*URL'*, *'region_URL'*, '*image URL'*, '*description'*
- *'region'* and *'state'* are similar. Since *'region'* has more categories than '*state'*, this means that *'region'* has fewer observations within each category. To build a robust model, we prefer to include a variable with more categories. So, we deleted *'region'* and kept '*state'*
- We also deleted '*lat'* and '*long*,' since we are already capturing location information with '*state'*

2) Deletion based on Missing Values

Next, we checked the proportion of missing values, and used that as a benchmark to determine which additional columns to remove.

a) *'size'* column - has 67% missing values. It is hard to derive meaningful information from a column with such a high proportion of missing values, so we deleted it.
b) *'county'* column - had 100% missing values.

3) Outlier Deletion

a. We decided to only keep cars within a certain min-max '*price'* range. So, we deleted observation rows with a price below $750, or a price above $100,000; though we do understand that some high-tier brands like Aston Martin and Ferrari may warrant individual analysis to determine if their prices are plausible. When the goal is to predict the price for a car one would most commonly come across, observations in extreme ranges are considered noise.
b. We also dropped observations where the '*odometer'* reading was less than 50 miles, or more than 300,000 miles. When a car has less than 50 miles on it, it is still considered a new car, for our purposes. We felt that cars with more than 300,000 miles would be out of most people's scope for their search. Furthermore, dealers would also be less inclined to buy and sell these types of cars, as they may be less reliable, and have a lower ROI.
c. We also dropped observations for which the '*year'* was older than 1990. Cars older than ~30 years are probably considered antique, and may have inflated prices to reflect their antique value.
d. We also dropped observations for the year 2021, since our model is not aimed at predicting new model-year prices.

4) Cleaning categorical variables

a) We noticed that the *'state'* variable lists California twice, once with uppercase initials, and once with lowercase. We converted the uppercase 'CA' to lowercase 'ca.'

5) Missing value imputation

This dataset contained a high proportion of missing values in many columns. We considered many imputation options. For instance, we wrote code and tested the idea of imputing the missing values in the '*cylinder*' column with the mode, based on grouping by the '*manufacturer*' column. For example, we wanted to find the mode number of cylinders for Hondas, and then impute that value for all the other observations where '*manufacturer*' is Honda, and where the '*cylinders*' value is missing. However, after many discussions, we decided that was not the best approach for maintaining data integrity.

Instead, we coded all missing values as 'unspecified,' and treated them as a separate category for each variable when modeling.

After these preprocessing steps, we are left with: 364,573 rows, and 14 columns. The columns we are left with include the following: '*price*', '*year*', '*manufacturer*', '*model*', '*condition*', '*cylinders*', '*fuel*', '*odometer*', '*title_status*', '*transmission*', '*drive*', '*type*', '*paint_color*', and '*state*'.

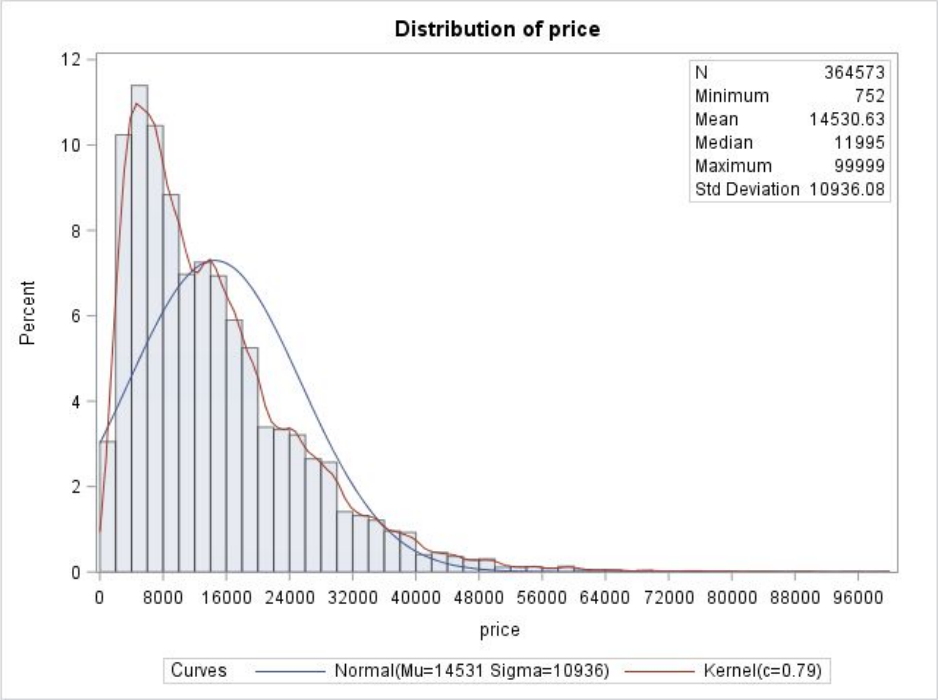
D. Exploratory Data Analysis (EDA):

In order to build a strong model, we performed exploratory analysis to gain a better understanding of the dataset we are working with.

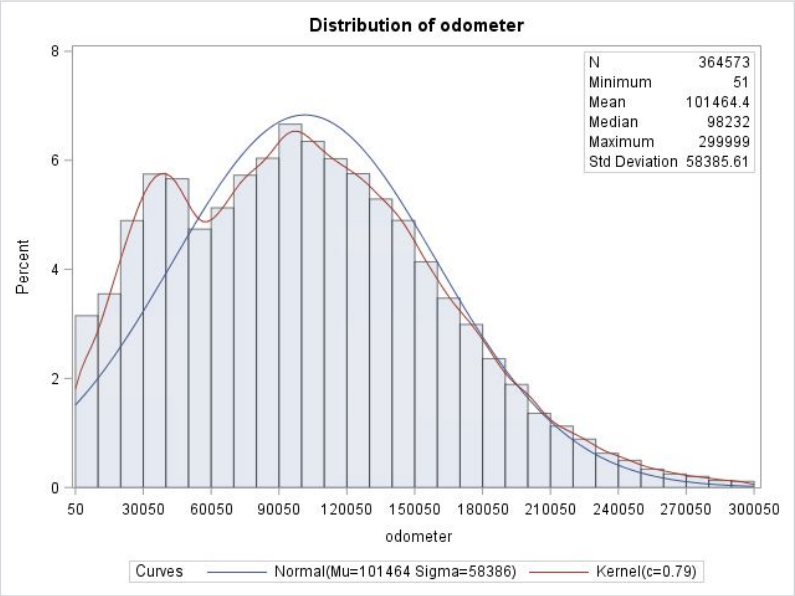
We split the EDA into two parts: one for categorical variables, and another for continuous variables.

I. Continuous Variables:

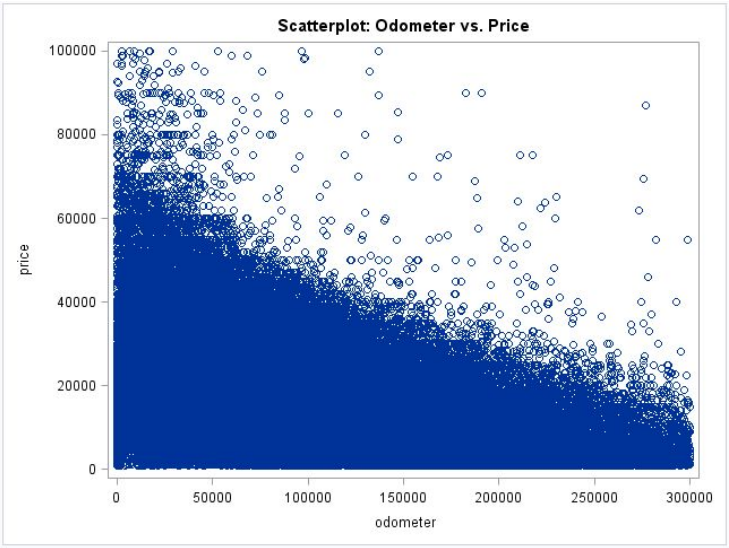
1. *'price'* - This histogram shows that the majority of prices fall in the $2,000 - $16,000 range. This distribution is right-skewed, which means there are a lot of extreme values in the upper price range. We can also infer that there are a lot of extreme prices when we observe that the mean is significantly greater than the median. This means that the mean is significantly influenced by a few extreme prices.



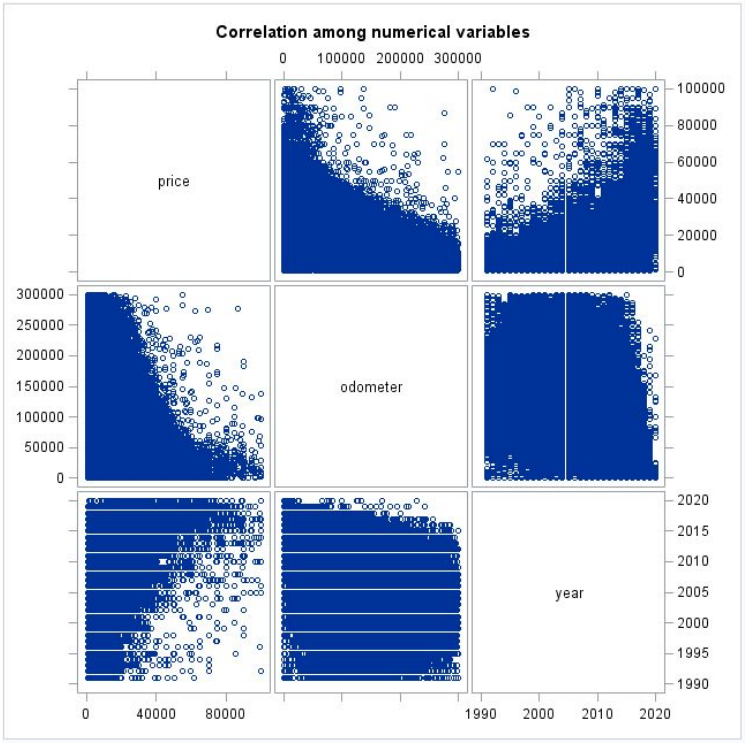
2. *'odometer'* - This histogram shows that most popular used cars have an odometer reading around 100,000 miles. We also see that this distribution looks almost like a normal distribution, and so is not very skewed. Although, there is a slight bulk in the lower end of values. But, the difference between the mean and median is only about 3,000 miles, which is not too large.



3. Scatterplot between *'price'* and *'odometer'* - From the scatterplot, we see an obvious correlation between these two variables: as the odometer reading increases, price decreases. This is confirmed later when we run a correlation procedure to gain a numerical understanding of the correlation between these two variables. We also see that for cars with lower odometer readings, the price range is large. But for cars with higher odometer readings, the prices are less scattered.



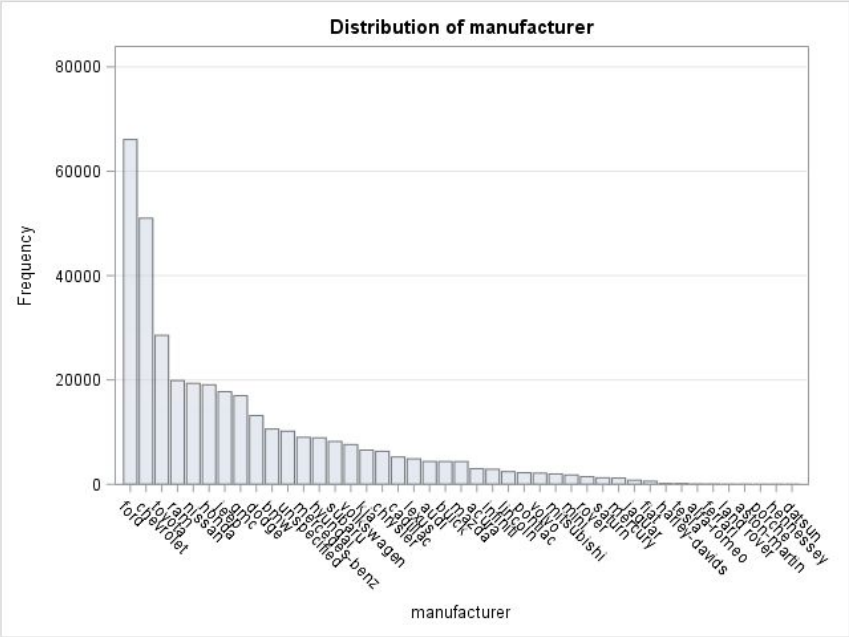
4. Correlation amongst *'price,' 'year'* and *'odometer'* - We see that the year is positively correlated with the price, which means that newer cars cost more. We also see that the odometer reading is negatively correlated with price, which means that cars with more mileage sell for less. These findings match with general domain knowledge of used cars.



Pearson Correlation Coefficients, N = 364573 Prob > |r| under H0: Rho=0			
	price	year	odometer
price	1.00000	0.57614 <.0001	-0.51121 <.0001
year	0.57614 <.0001	1.00000	-0.61298 <.0001
odometer	-0.51121 <.0001	-0.61298 <.0001	1.00000

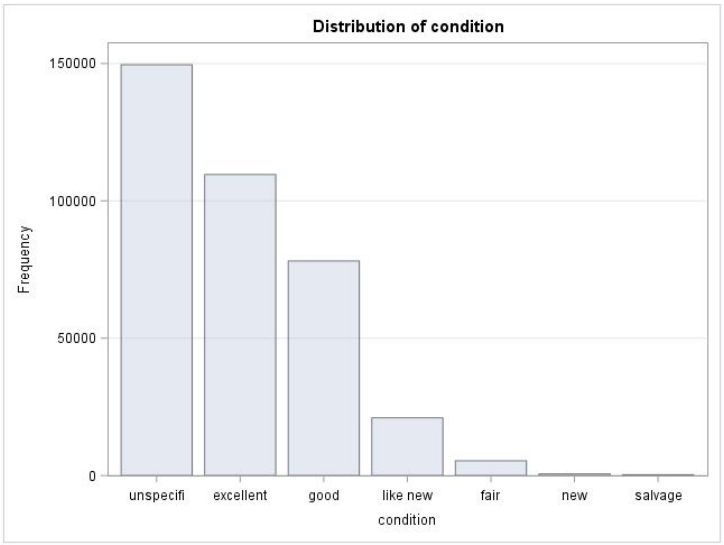
II. Categorical Variables

1. *'manufacturer'* - We see that the top five most frequent manufacturers in this data set are: ford, chevrolet, toyota, ram and nissan. We see that in this data set, luxury cars are sold less frequently than other, more common manufacturers. This tells us that we won't have an inordinate number of high-priced cars.

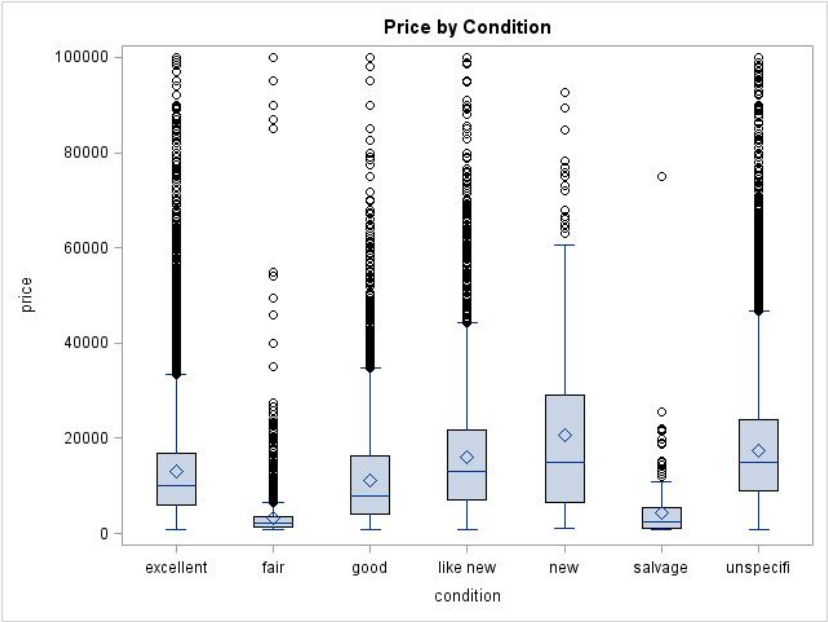


2. *'condition'* - Most of the observations for which we have values fall in the excellent and good categories. This is important to know because if most of the cars were in either the new or salvage categories, we

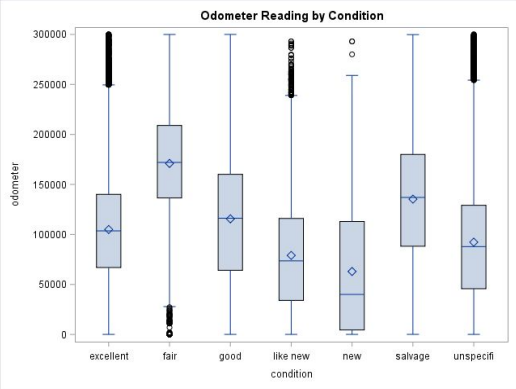
would have expected either a lot of very high or very low price values. We also note that cars with an unspecified distribution are the most common.



3. Median *'price'* by '*condition'* - it seems like cars in excellent, good, and like new condition have similar median price (around $10,000 to $15,000). For cars that are new, the median price is higher (closer to $20,000), and the range is also wider than for other conditions. The prices for cars in fair and salvage condition are drastically lower than those with other conditions.



4. Median *'odometer'* by '*condition'* - it is interesting to note that cars in a new condition have a high median odometer reading. This means that these cars are likely not new, and that putting them in this category is inaccurate. To ensure that we are making this judgment based on a sound sample size, we also counted the number of cars in each of these '*condition'* categories and saw that cars in a 'new' condition only account for 0.17% of the sample size. This information later informs our decision of dropping the 'new' category of cars.



Price by Condition

The FREQ Procedure

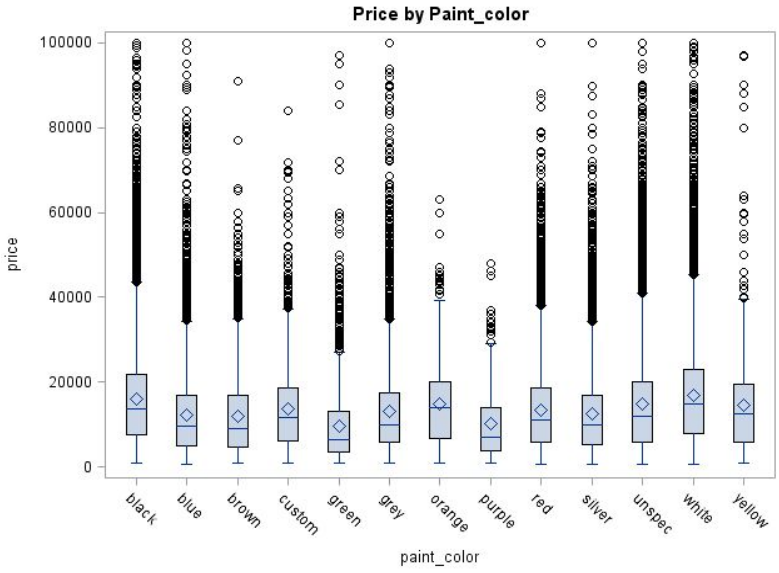
condition	Frequency	Percent	Cumulative Frequency	Cumulative Percent
excellent	109591	30.06	109591	30.06
fair	5379	1.48	114970	31.54
good	78114	21.43	193084	52.96
like new	21027	5.77	214111	58.73
new	609	0.17	214720	58.90
salvage	310	0.09	215030	58.98
unspecifi	149544	41.02	364574	100.00

5. Median *'price'* by '*condition'* and '*type'* - obviously, there is a difference in price by condition. However, it is interesting to note that the price difference by condition differs to varying extents based on the type of car. For instance, the difference between a fair versus good coupe is $11,500. But the difference between a fair versus good mini-van is $2,200.

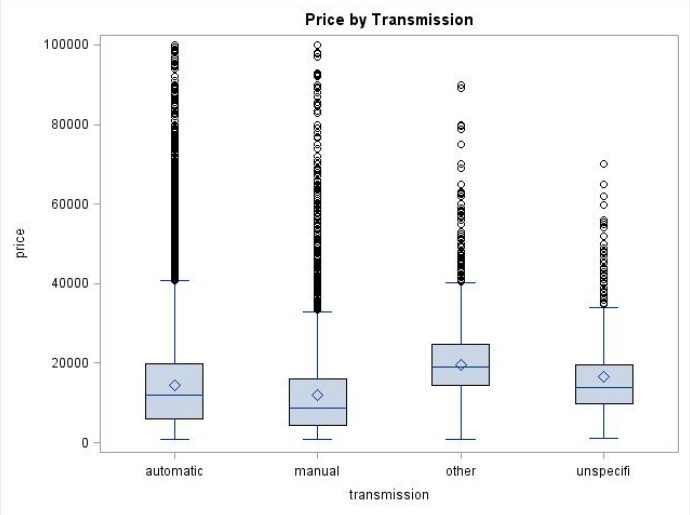
PROC Tabulate: Median Price by Condition & Type

Median Price	Median														
	price														
	type														All
	SUV	bus	convertib	coupe	hatchback	mini-van	offroad	other	pickup	sedan	truck	unspecifi	van	wagon	
condition															
excellent	$10,800	$19,950	$10,200	$10,200	$6,995	$7,990	$15,700	$9,497	$16,995	$7,490	$17,999	$9,750	$10,990	$8,999	$10,000
fair	$2,200	$5,900	$2,500	$1,800	$1,775	$1,800	$2,800	$2,500	$2,900	$1,800	$3,500	$2,100	$2,250	$1,500	$2,200
good	$6,000	$10,500	$6,450	$13,300	$8,500	$4,000	$8,500	$17,700	$19,900	$5,500	$13,250	$5,500	$8,990	$6,295	$8,000
like new	$13,900	$32,500	$12,948	$15,480	$8,995	$10,900	$21,950	$12,950	$22,998	$8,995	$23,995	$11,800	$16,500	$9,999	$12,995
new	$15,500	none	$21,125	$32,000	$12,923	$5,948	$30,000	$10,600	$35,000	$7,983	$29,473	$12,250	$36,800	$5,495	$14,900
salvage	$2,195	none	$2,999	$2,700	$2,900	$1,595	none	$10,000	$3,125	$2,500	$2,800	$2,900	$2,150	$1,600	$2,500
unspecifi	$15,979	$7,475	$14,223	$15,900	$9,410	$9,998	$11,000	$14,995	$22,999	$10,995	$24,995	$12,998	$14,500	$11,475	$14,998
All	$11,977	$15,950	$10,499	$12,999	$7,995	$6,900	$13,000	$17,000	$20,800	$7,995	$18,998	$10,500	$11,990	$9,995	$11,995

6. Median *'price'* by '*paint_color'* - we see that the median price does indeed differ by color, but not by a visibly significant amount. Although we acknowledge this could be due to vastly different sample sizes, we will still keep '*paint_color'* as a variable when modeling.



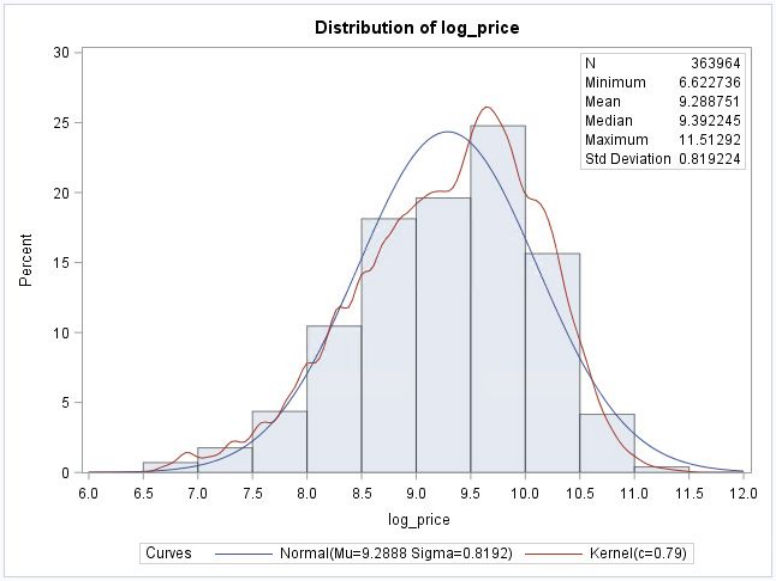
7. Median *'price'* by '*transmission'* - we see that the median price does indeed differ by '*transmission'*. But the difference between automatic versus manual does not seem visibly significant. We also see that there are several outliers for all three categories.



E. Empirical Analysis:

I. Pre-Modeling Data Pre-Processing

1) In the EDA, we noticed that the '*price'* variable is not normally distributed. This made us realize that it would be wise to create a log version of the '*price'* variable. This way, the variable is more likely to be normally distributed, and the model is not influenced by very extreme prices. Even though we see below that '*log_price'* is only slightly closer to normal distribution now, we still decide to create our models with *'log_price'* as the dependent variable. In addition, in some of the model variations we created, we got higher adjusted R-squared values when using *'log_price'* rather than *'price.'*



2) Also in the EDA, we saw the several cars that had a '*condition'* of 'new' actually had several thousand miles on them. Since this project is for predicting used car prices, we decide to exclude the cars with an assigned condition of new, because obviously labeling cars with several thousand miles on them as new is inaccurate.

3) Lastly, due to server constraints, we were forced to remove the '*model*' column. In addition, including this column in the modeling would have led to the data set becoming very wide, and sparse. There were several models that only had a handful of observations. So, we deemed it best to drop this column. Though, we acknowledge that for a car dealership, knowing the model of a car would be very valuable in being able to predict price.

II. Statistical Analyses

1) Correlation between '*log_price*' and '*odometer*' - in the EDA, we created a scatter plot to gain a general understanding of the relationship between these two numerical variables. Here, we obtained a Pearson correlation coefficient to understand the direction and strength of this relationship. The coefficient is -0.55512, with a p-value of <0.0001. Since the coefficient is negative, this means the variables move in opposite directions. (i.e. as '*odometer*' increases, '*log_price*' decreases). A value of -0.55512 means that we have a moderately strong correlation. The p-value corroborates this, indicating that the coefficient is significant.

Pearson Correlation Coefficients, N = 363965 Prob > |r| under H0: Rho=0		
	log_price	odometer
log_price	1.00000	-0.55512 <.0001
odometer	-0.55512 <.0001	1.00000

2) ANOVA for '*paint_color*' - from the EDA, we were not certain whether '*paint_color*' was influential in determining price or not. So, we performed an ANOVA test, which is appropriate when you have a continuous dependent variable, and categorical dependent variables. We see that the Model Pr > F is <0.0001, which means that the model explains more than a naive model. In addition, the Source '*paint_color*' Pr > F is <0.0001, which means that there is a difference in *'log_price'* by '*paint_color.'* This is helpful, because now we will include this variable in our model.

The GLM Procedure

Dependent Variable: log_price

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	12	7586.1254	632.1771	972.11	<.0001
Error	363952	236682.7667	0.6503		
Corrected Total	363964	244268.8921			

R-Square	Coeff Var	Root MSE	log_price Mean
0.031056	8.681688	0.806420	9.288746

Source	DF	Type I SS	Mean Square	F Value	Pr > F
paint_color	12	7586.125377	632.177115	972.11	<.0001

Source	DF	Type III SS	Mean Square	F Value	Pr > F
paint_color	12	7586.125377	632.177115	972.11	<.0001

However, it is also important to note that the assumption of the *'log_price'* variances being homogenous across the different '*paint_color*' categories is violated. We see this from the Levene and Welch Tests. For Levene's Test, the null hypothesis is that the variances are equal. Since the p-value is <0.0001, we reject the null and conclude that variances are unequal. On the other hand, the Welch test is applied when the variances are unequal (but still holds to the assumption that the variances are normal), which is applicable in this case. The null

in this case is that the mean *'log_price'*s vary by color. Since the p-value is <0.0001, we reject the null and conclude that the price does indeed differ by color. Thus, we will include *'paint_color'* in our modeling.

The GLM Procedure

Levene's Test for Homogeneity of log_price Variance
ANOVA of Squared Deviations from Group Means

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
paint_color	12	766.0	63.8336	79.57	<.0001
Error	363952	291960	0.8022		

Welch's ANOVA for log_price

Source	DF	F Value	Pr > F
paint_color	12.0000	960.39	<.0001
Error	16389.8		

3) ANOVA for '*title_status'* - we did not perform EDA for '*title_status'* , so we used an ANOVA test to help us determine whether to include this variable in the final model. Similar to above, we see that the model '*log_price'* = '*title_status'* explains more than a naive model, since p-value for the F-test is <0.0001. But, we also see that '*title_status'* by itself explains less than 1% of the variation in price.

The GLM Procedure

Dependent Variable: log_price

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	6	1828.7055	304.7842	457.55	<.0001
Error	363958	242440.1866	0.6661		
Corrected Total	363964	244268.8921			

R-Square	Coeff Var	Root MSE	log_price Mean
0.007486	8.786574	0.816163	9.288746

Source	DF	Type I SS	Mean Square	F Value	Pr > F
title_status	6	1828.705478	304.784246	457.55	<.0001

Source	DF	Type III SS	Mean Square	F Value	Pr > F
title_status	6	1828.705478	304.784246	457.55	<.0001

Based on the Levene's Test result, we conclude that the variances across the *'title_status'* categories are equal. Given this, we can have more faith in the ANOVA results above that *'title_status'* is indeed an important predictor variable, despite the small R-squared value. This is again confirmed with the Welch ANOVA test, which conducts an ANOVA test while assuming that variances are unequal. Since the p-value is <0.0001, we conclude that *'title_status'* is significant.

Levene's Test for Homogeneity of log_price Variance
ANOVA of Squared Deviations from Group Means

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
title_status	5	6.9244	1.3849	1.86	0.0982
Error	36606	27295.2	0.7456		

Welch's ANOVA for log_price

Source	DF	F Value	Pr > F
title_status	5.0000	68.30	<.0001
Error	19.2425		

4) ANOVA for '*type'* - just like with the previous two variables, we see that the *'log_price'* = '*type'* model explains more than a naive model, and that there are differences in price by '*type'* . But it is also important to note that '*type'* explains 12% of the variation in *'log_price'*. This means that we must definitely include '*type'* in our price models.

The GLM Procedure

Dependent Variable: log_price

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	13	29439.7562	2264.5966	3836.55	<.0001
Error	363951	214829.1358	0.5903		
Corrected Total	363964	244268.8921			

R-Square	Coeff Var	Root MSE	log_price Mean
0.120522	8.271191	0.768290	9.288746

Source	DF	Type I SS	Mean Square	F Value	Pr > F
type	13	29439.75624	2264.59663	3836.55	<.0001

Source	DF	Type III SS	Mean Square	F Value	Pr > F
type	13	29439.75624	2264.59663	3836.55	<.0001

5) Interaction Variables - we compared the adjusted R-squared values for '*type'* and '*condition'* versus '*type' *'condition'*. Based on the EDA, we thought there might be an interaction between '*type'* and '*condition'*. Put another way, we thought that the difference between the price for a car in excellent versus fair condition would differ based on the model. When we built a model with '*type'* and *condition* versus '*type' *'condition'* , we saw that the latter accounts for only a slightly greater adjusted R-squared than the former. So, we decided to not use the interaction term '*type' *'*condition*' when generating models.

The GLM Procedure

Dependent Variable: log_price

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F	R-Square
Model	18	58986.2689	3277.0149	6436.96	<.0001	0.241481

ANOVA: type*condition

The GLM Procedure

Dependent Variable: log_price

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F	R-Square
Model	81	62359.2716	769.8676	1540.00	<.0001	0.255289

III. Modeling

Before beginning to model, we thought about our overall goal again, which was to build a model with the highest adjusted R-squared value. This would give the car dealership greater confidence that the predictor variables put in the model can generate an accurate selling price for a used car. We chose adjusted R-squared as our metric of choice because we want a model that takes into account the number of predictor variables. Ideally, we want a model that uses the least amount of predictors, yet still makes accurate predictions.

Finally, we ran the following models:

1) PROC REG - we first ran a multiple linear regression model to estimate linear parameters. The dependent variable is *'log_price'*. However, we quickly discovered that this is a difficult model for SAS to run when you have a large number of dummy variables. Due to server constraints, we were unable to run a model that included all 119 predictor variables. Instead, we ran a model with just 4 predictor variables: *'odometer,' 'year', 'condition'* and *'cylinder.'* More specifically, we ran the model with the dummy variables for the latter two.
    a) Since the p-value is <0.0001, that means the model is significant.
    b) In addition, we achieve a better-than-average adjusted R-squared value of 60.83%.

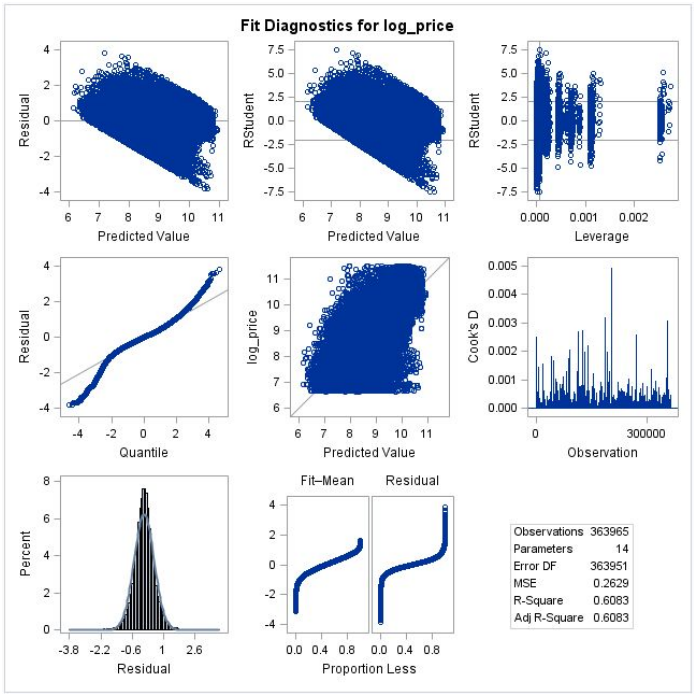
However, regarding the Fit Diagnostics:

a) Top-left graph: we see, from a glance, that more than 5% of the residuals lie outside a band. This means that our assumption that the errors are homoskedastic (i.e. variance of the error is constant) is violated. The errors are heteroskedastic. While this does not affect the parameter estimates, it causes the standard errors to be compromised.
b) Bottom-left graph: the assumption that the errors are normally distributed looks satisfied.
c) Bottom-center graph: ideally, the Fit-Mean line should be longer than the Residual line. This would mean that the model is capturing more information than the residuals. However, we see the reverse, which means that our model doesn't perform well at all.

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	13	148581	11429	43472.0	<.0001
Error	363951	95687	0.26291		
Corrected Total	363964	244269			

R-Square	0.6083
Adj R-Sq	0.6083



Since PROC REG is unable to handle so many categorical variables, we moved on to a different SAS function.

2) PROC GLMSELECT (No selection) - we next used PROC GLMSELECT to create another multiple linear regression model. Within the CLASS statement, we specified that the first category in the categorical variables as the one to be dropped. Again, we obtain a significant model, and also obtain a healthy adjusted R-squared of 77.02%. Below,

we interpret a few of the parameter estimates. Please note that the interpretations are in terms of percentages, because we used *'log_price.'*

a) *'year'* - for every increase in year (i.e. for every year the car is younger), we get a 7.7% increase in price
b) *'condition'* - as compared to a car in an excellent condition (dropped category), a car in a fair condition sells for 63.9% less
c) *'cylinders'* - as compared to a 10-cylinder car (the dropped category), a 12-cylinder car sells for 41.2% more. All cars with other engine sizes sell for less than a 10-cylinder car.
d) *'manufacturer'* - compared to an Acura (the dropped category), a Ferrari is most expensive, because its parameter estimate is the greatest amongst all the models, and has a p-value <0.0001.
e) *'paint_color'* - compared to black cars (the dropped category), white cars are not priced significantly differently, as the p-value is not significant and does not cause us to reject the null hypothesis.

The GLMSELECT Procedure
Least Squares Model (No Selection)

Analysis of Variance					
Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	149	188167	1262.86476	8189.97	<.0001
Error	363814	56099	0.15420		
Corrected Total	363963	244266			

Root MSE	0.39268
Dependent Mean	9.28875
R-Square	0.7703
Adj R-Sq	0.7702
AIC	-316325
AICC	-316325
SBC	-678670

(The below Parameter Estimates table only includes a few parameter estimates, to give an overview of how the rest of the rows would be interpreted)

Parameter Estimates					
Parameter	DF	Estimate	Standard Error	t Value	Pr > |t|
year	1	0.077192	0.000171	452.27	<.0001
condition fair	1	-0.638785	0.005594	-114.20	<.0001
cylinders 12 cylinder	1	0.412120	0.036986	11.14	<.0001
manufacturer ferrari	1	1.738579	0.059726	29.11	<.0001
paint_color white	1	0.000508	0.002271	0.22	0.8230

3) PROC GLMSELECT (Stepwise Selection) - we next used PROC GLMSELECT with Stepwise selection, which selects predictor variables in an automatic manner. In each iteration, a variable is either added or deleted from the set of independent variables based on a pre-described criterion. With this model, we obtained the exact same results as in the PROC GLMSELECT (no selection) model. This method picked the last model as the best one.

Below, we again interpret a few different parameter estimates. Please note that the interpretations are in terms of percentages, because we have *'log_price.'*

a) *'odometer'* - for every 1 mile increase in odometer reading, price decreases by 0.0003925%. While this is a small percentage, it is still significant, as the p-value is <0.0001. But we can interpret this in a more meaningful way by saying that for every 10,000-mile increase in odometer reading, the price decreases by nearly 4%.
b) *'type'* - compared to SUVs (dropped category), offroad cars sell for 38.9% more.
c) *'manufacturer'* - Aston-Martins sell for nearly double (96.2%) what Acura's sell for.
d) *'state'* - for this variable, we have an interesting result. All car prices in every state are in comparison to Alaska, which is the dropped category. We see that the majority of parameter estimates are negative,

which means that cars in nearly every state sell for a price below what they sell for in Alaska. For Texas in particular, cars sell for 6.9% less, in comparison to Alaska. This was not an expected result.

e) *'title_status'* - cars with missing titles sell for 33.8% less than cars with clean titles.

The GLMSELECT Procedure

Stepwise Selection Summary

Step	Effect Entered	Effect Removed	Number Effects In	Number Parms In	Adjusted R-Square
0	Intercept		1	1	0.0000
1	year		2	2	0.4469
2	type		3	15	0.5775
3	manufacturer		4	57	0.6230
4	odometer		5	58	0.6627
5	fuel		6	63	0.7063
6	cylinders		7	71	0.7322
7	condition		8	76	0.7459
8	drive		9	79	0.7578
9	state		10	129	0.7643
10	title_status		11	135	0.7677
11	transmission		12	138	0.7694
12	paint_color		13	150	0.7702*

* Optimal Value of Criterion

Selection stopped because all effects are in the final model.

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	149	188167	1262.86476	8189.97	<.0001
Error	363814	56099	0.15420		
Corrected Total	363963	244266			

Root MSE	0.39268
Dependent Mean	9.28875
R-Square	0.7703
Adj R-Sq	0.7702
AIC	-316325
AICC	-316325
SBC	-678670

(The below Parameter Estimates table only includes a few parameter estimates, to give an overview of how the rest of the rows would be interpreted)

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Pr > |t|
odometer	1	-0.000003925	1.5007551E-8	-261.56	<.0001
type offroad	1	0.388853	0.018882	20.59	<.0001
manufacturer aston-martin	1	0.962138	0.114237	8.42	<.0001
state tx	1	-0.069015	0.007635	-9.04	<.0001
title_status missi	1	-0.338231	0.051188	-6.61	<.0001

4) <u>PROC GLMSELECT (Lasso Regression):</u> Next we performed a Lasso regression. However, it resulted in a slightly smaller adjusted R-squared than the stepwise selection. The main surprising finding from this model was that the cars with a title status of clean did not have a significant effect, and so this variable was removed from the model. The previous models seemed to indicate that cars with a clean title status sell for more.

Based on this and the fact that the SBC for the stepwise model was less than it is for this lasso model, we decided that this lasso model is not the ideal model.

The GLMSELECT Procedure

LASSO Selection Summary

Step	Effect Entered	Effect Removed	Number Effects In	Adjusted R-Square
48		title_status_clean	47	0.7417
141	state_la		138	0.7701*

* Optimal Value of Criterion

The selected model, based on Adj R-Sq, is the model at Step 141.

Selection stopped at a local maximum of the AdjRSq criterion.

Dimensions	
Number of Effects	13
Number of Effects after Splits	160
Number of Parameters	160

Stop Details				
Candidate For	Effect	Candidate Adj-RSq		Compare Adj-RSq
Entry	paint_color_brown	0.7700	<	0.7701

Analysis of Variance				
Source	DF	Sum of Squares	Mean Square	F Value
Model	137	188118	1373.12404	8897.58
Error	363826	56148	0.15433	
Corrected Total	363963	244266		

Root MSE	0.39284
Dependent Mean	9.28875
R-Square	0.7701
Adj R-Sq	0.7701
AIC	-316032
AICC	-316032
SBC	-678507

5) PROC GLMSELECT (Polynomial Regression) - from intuition and also from our above models, we know that the odometer reading of a car has a significant influence on its selling price. So, we created a polynomial regression model that included *'odometer'* up to the fourth degree. This model is also significant and gives our highest adjusted R-squared value, at 77.57%, and also the lowest SBC value, at -687260.

We also note that the parameter estimates for every odometer value past the linear is infinitesimally small, but is still significant, according to the p-value.

The GLMSELECT Procedure
Least Squares Model (No Selection)

Analysis of Variance					
Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	152	189481	1246.58664	8278.29	<.0001
Error	363811	54785	0.15059		
Corrected Total	363963	244266			

Root MSE	0.38805
Dependent Mean	9.28875
R-Square	0.7757
Adj R-Sq	0.7756
AIC	-324948
AICC	-324947
SBC	-687260

Dimensions	
Number of Effects	16
Number of Parameters	163

Parameter Estimates					
Parameter	DF	Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	-140.621428	0.343682	-409.16	<.0001
odometer	1	0.000008934	0.000000160	55.98	<.0001
odometer^2	1	-1.7255E-10	2.418342E-12	-71.35	<.0001
odometer^3	1	8.158304E-16	1.367627E-17	59.65	<.0001
odometer^4	1	-1.24019E-21	2.551596E-23	-48.60	<.0001

F. Conclusions, Limitations, Further Explorations:

Using a used car sales data set from Craigslist, we performed exploratory analysis, statistical tests, and statistical modeling to produce a model that accurately predicts how much a used car will sell for, given predictor variables. We conclude that the polynomial regression model that includes the *'odometer'* value up to the fourth degree produces the best results, with an adjusted R-squared value of 77.57%.

Some important limitations of this analysis are:

1) We did not use any statistical or machine learning methods to determine the relative importance of the predictor variables. We conducted statistical tests to determine whether each variable, independently, is significant or not. However, our initial goal was to use the predictor variables we were left with after data preprocessing to build a model with the highest accuracy.
2) Before running the regression models, we did do rigorous testing to make sure all of the three GLM model assumptions were satisfied:
    a) The model will be accurately modeled by linear parameters
    b) The error term (i.e. residuals) are assumed to have a normal distribution, mean of zero and a constant variance
    c) The errors are independent
3) Another limitation of this analysis is that the prices likely do not take into account economic inflation over the years. A future report can incorporate this potential factor into the analysis.

Appendix:

Appendix A - The below code was for moving incorrect *'model'* values back to the appropriate *'manufacturer'* column. Though the logic is sound, we were unable to use this code due to technical glitches. But rest assured, every attempt was made to clean the *'manufacturer'* column to the best of our ability.

```
PROC sql;
CREATE TABLE work.vehicles5 AS
SELECT condition, cylinders, drive, fuel, manufacturer, model, odometer, paint_color, price, log_price, title_status,
        transmission, year, state, type,
                CASE WHEN UPCASE(MODEL) LIKE '%BENTLEY%' THEN 'BENTLEY'
                ELSE CASE WHEN UPCASE(MODEL) LIKE '%BMW%' THEN 'BMW'
                ELSE CASE WHEN UPCASE(MODEL) LIKE '%CADILLAC%' THEN 'CADILLAC'
                ELSE CASE WHEN UPCASE(MODEL) LIKE '%CAMARO%' THEN 'CHEVROLET'
                ELSE CASE WHEN UPCASE(MODEL) LIKE '%AUDI%' THEN 'AUDI'
                ELSE CASE WHEN UPCASE(MODEL) LIKE '%BENZ%' THEN 'MERCEDES'
                ELSE CASE WHEN UPCASE(MODEL) LIKE '%CAMRY%' THEN 'TOYOTA'
                ELSE CASE WHEN UPCASE(MODEL) LIKE '%CHEVROLET%' THEN 'CHEVROLET'
                ELSE CASE WHEN UPCASE(MODEL) LIKE '%PORSCHE%' THEN 'PORSCHE'
                ELSE CASE WHEN UPCASE(MODEL) LIKE '%SCION%' THEN 'SCION'
                ELSE CASE WHEN UPCASE(MODEL) LIKE '%HUMMER%' THEN 'HUMMER'
                ELSE CASE WHEN UPCASE(MODEL) LIKE '%MASERATI%' THEN 'MASERATI'
                ELSE CASE WHEN UPCASE(MODEL) LIKE '%OLDSMOBILE%' THEN 'OLDSMOBILE'
                ELSE CASE WHEN UPCASE(MODEL) LIKE '%SMART%' THEN 'SMART'
                ELSE CASE WHEN UPCASE(MODEL) LIKE '%SUZUKI%' THEN 'SUZUKI'
                ELSE CASE WHEN UPCASE(MODEL) LIKE '%ISUZU%' THEN 'ISUZU'
                ELSE CASE WHEN UPCASE(MODEL) LIKE '%TOYOTA%' THEN 'TOYOTA'
                ELSE CASE WHEN UPCASE(MODEL) LIKE '%AM%' THEN 'AM'
                ELSE CASE WHEN UPCASE(MODEL) LIKE '%PLYMOUTH%' THEN 'PLYMOUTH'
                ELSE CASE WHEN UPCASE(MODEL) LIKE '%POLARIS%' THEN 'POLARIS'
                ELSE CASE WHEN UPCASE(MODEL) LIKE '%FORD%' THEN 'FORD' ELSE 'OTHER'
                END END END END END END END END END END END END END END END END END
                END END END AS manufacturer_new,
                CASE WHEN UPCASE(MODEL) LIKE '%BENTLEY%' THEN SUBSTR(MODEL,LENGTH('BENTLEY ')+1,11)
                ELSE CASE WHEN UPCASE(MODEL) LIKE '%BMW%' THEN SUBSTR(MODEL,LENGTH('BMW/')+1,11)
                ELSE CASE WHEN UPCASE(MODEL) LIKE '%CADILLAC%' THEN SUBSTR(MODEL,LENGTH('CADILLAC/')+1,11)
                ELSE CASE WHEN UPCASE(MODEL) LIKE '%AUDI%' THEN SUBSTR(MODEL,LENGTH('AUDI/')+1,11)
                ELSE CASE WHEN UPCASE(MODEL) LIKE '%BENZ%' THEN SUBSTR(MODEL,LENGTH('BENZ')+1,11)
                ELSE CASE WHEN UPCASE(MODEL) LIKE '%CHEVROLET%' THEN
SUBSTR(MODEL,LENGTH('CHEVROLET/')+1,11)
                ELSE CASE WHEN UPCASE(MODEL) LIKE '%PORSCHE%' THEN SUBSTR(MODEL,LENGTH('PORSCHE')+1,11)
                ELSE CASE WHEN UPCASE(MODEL) LIKE '%SCION%' THEN SUBSTR(MODEL,LENGTH('SCION')+1,11)
                ELSE CASE WHEN UPCASE(MODEL) LIKE '%HUMMER%' THEN SUBSTR(MODEL,LENGTH('HUMMER')+1,11)
                ELSE CASE WHEN UPCASE(MODEL) LIKE '%MASERATI%' THEN SUBSTR(MODEL,LENGTH('MASERATI')+1,11)
                ELSE CASE WHEN UPCASE(MODEL) LIKE '%OLDSMOBILE%' THEN
SUBSTR(MODEL,LENGTH('OLDSMOBILE')+1,11)
                ELSE CASE WHEN UPCASE(MODEL) LIKE '%SMART%' THEN SUBSTR(MODEL,LENGTH('SMART')+1,11)
                ELSE CASE WHEN UPCASE(MODEL) LIKE '%SUZUKI%' THEN SUBSTR(MODEL,LENGTH('SUZUKI')+1,11)
                ELSE CASE WHEN UPCASE(MODEL) LIKE '%ISUZU%' THEN SUBSTR(MODEL,LENGTH('ISUZU')+1,11)
                ELSE CASE WHEN UPCASE(MODEL) LIKE '%TOYOTA%' THEN SUBSTR(MODEL,LENGTH('TOYOTA')+1,11)
```

```
        ELSE CASE WHEN UPCASE(MODEL) LIKE '%AM%' THEN SUBSTR(MODEL,LENGTH('AM')+1,11)
                ELSE CASE WHEN UPCASE(MODEL) LIKE '%PLYMOUTH%' THEN SUBSTR(MODEL,LENGTH('PLYMOUTH')+1,11)
                ELSE CASE WHEN UPCASE(MODEL) LIKE '%POLARIS%' THEN SUBSTR(MODEL,LENGTH('POLARIS')+1,11)

                ELSE CASE WHEN UPCASE(MODEL) LIKE '%FORD%' THEN SUBSTR(MODEL,LENGTH('FORD')+1,11) ELSE
'OTHER'
                END END END END END END END END END END END END END END END END END END END AS model_new
  FROM vehicles4_blanks
 WHERE manufacturer = ''
UNION
SELECT  condition, cylinders, drive, fuel, manufacturer, model, odometer, paint_color, price, log_price, title_status,
        transmission, year, state, type, manufacturer as manufacturer_new, model as model_new
 FROM vehicles4_blanks
 WHERE manufacturer > ''
 ;
quit;
```

Appendix B: One strategy we considered for imputing missing values was imputing the *'cylinder'* with the mode, based on grouping by the *'manufacturer'* and *'model'* column. After much debate, we decided this would interfere with data integrity and so decided not to use it.

```
cyl_grpby = cyl_grpby.groupby(["manufacturer", "model"], as_index = False)["cylinders"].agg(pd.Series.mode)
# In[12]:
cyl_grpby.cylinders = cyl_grpby.cylinders.astype(str)
# In[13]:
null_lkup_cyl_cor1 = cyl_grpby.where(cyl_grpby['cylinders'] != '999.0')
# In[14]:
#List of Manufacturers
manu_list = ['acura', 'alfa-romeo', 'aston-martin', 'audi', 'bmw',
     'buick', 'cadillac', 'chevrolet', 'chrysler', 'dodge', 'ferrari',
     'fiat', 'ford', 'gmc', 'harley-davids', 'hennessey', 'honda',
     'hyundai', 'infiniti', 'jaguar', 'jeep', 'kia', 'land rover',
     'lexus', 'lincoln', 'mazda', 'mercedes-benz', 'mercury', 'mini',
     'mitsubishi', 'nissan', 'pontiac', 'porche', 'ram', 'rover',
     'saturn', 'subaru', 'toyota', 'volkswagen', 'volvo']
# In[15]:
cyl_grpby.manufacturer = cyl_grpby.manufacturer.astype(str)
cyl_grpby.model = cyl_grpby.model.astype(str)
cyl_grpby.cylinders = cyl_grpby.cylinders.astype(str)
# In[16]:
null_lkup_cyl_cor1 = null_lkup_cyl_cor1[null_lkup_cyl_cor1.manufacturer.isin(manu_list)]
# In[17]:
#null_lkup_cyl_cor2 = null_lkup_cyl_cor1.where(null_lkup_cyl_cor1['cylinders'].str.len() < 6)
null_lkup_cyl_cor2 = null_lkup_cyl_cor1
# In[18]:
#Cylinders Approved Values List
cyl_list = ['4.0', '5.0', '6.0', '8.0', '10.0', '12.0', '16.0']
# In[19]:
null_lkup_cyl_cor2 = null_lkup_cyl_cor2[null_lkup_cyl_cor2.cylinders.isin(cyl_list)]
# ## Final LookUp Table
# In[20]:
null_lkup_cyl_cor = null_lkup_cyl_cor2
# ## Mode Imputation - Cylinders Column
# In[21]:
#Sorting Both Tables
#Sort Main Table
cars_dt = cars_dt.sort_values(by=['manufacturer', 'model'], ascending = True)
#Sort LookUp Table
null_lkup_cyl_cor = null_lkup_cyl_cor.sort_values(by=['manufacturer', 'model'], ascending = True)
# In[22]:
#Joining the two dataframes
join_view = pd.merge(cars_dt, null_lkup_cyl_cor, on = ['manufacturer', 'model'])
# In[23]:
#Dropping records where Model is null
join_view = join_view[join_view['model'] != 999]
# In[24]:
#Dropping records where Manufacturer is null
```

```
join_view = join_view[join_view['manufacturer'] != 999]
# In[25]:
valid_df = join_view[join_view['cylinders_x'].isin(cyl_list)]
# In[26]:
imp_df = join_view[join_view['cylinders_x'] == '999.0']
# In[27]:
del valid_df['cylinders_y']
del imp_df['cylinders_x']
# In[28]:
valid_df['cylinders'] = ''
imp_df['cylinders'] = ''
# In[29]:
valid_df['cylinders'] = valid_df['cylinders_x']
imp_df['cylinders'] = imp_df['cylinders_y']
# In[30]:
del valid_df['cylinders_x']
del imp_df['cylinders_y']
# In[31]:
FINAL_CYL_IMPUTE = pd.concat([valid_df, imp_df])
```

References

SAS Built-in Help

"SAS Customer Support Site." *SAS Customer Support Site | SAS Support*, SAS, support.sas.com/en/support-home.html.

"SAS Help." *Documentation.sas.com*, documentation.sas.com/?docsetId=helpcenterwlcm&docsetVersion=1.0&docsetTarget=home.htm&locale

"Where Developers Learn, Share, & Build Careers." *Stack Overflow*, stackoverflow.com/.