Predicting Vehicle Asking Price on Craigslist

Michael Nicodemus
MA5790
October 2020

Abstract:
Vehicle information from Craigslist is used to construct a predictive model of the vehicle's asking price. Before we begin, we give a thorough background and provide definitions to the available variables. The goal of this study is to use the data provided and proceed through the model building process for several different models including data exploration, data imputation, variable transformations, dimensional reduction, tuning parameters. We then select our top three performing models and compare their performance on the test data set and juxtaposing their diagnostic plots. We then conclude with the selection of our best performing model and a conversation on its strengths and limitations as well as recommendations for future studies. Since we are predicting the quantitative variable for the asking price, naturally we will only consider regression models, however we will consider multiple models from both the linear and non-linear variety.

Table of Contents

1 Background

Craigslist is an online classified ad website where one can buy, sell and trade just about anything which selling is legal. The only limitations for a buyer are the geography in which the buyer is willing to travel to and what sellers have posted to the site. That is, you cannot buy what is not for sale. This study looks at the vehicle section of the webpage.

The data are generated by Austin Reese as part of a web scraping project and the data is then posted to the Kaggle website. Reese states that the data are refreshed "every few months." The data for this project was pulled in September of 2020 and as of the writing of this paper was still current on the site.

Since the seller is solely responsible for the information posted to the website, no two adds are the same and the information contained in the ads vary widely. The only required information to post an ad is the asking price. This leads to very messy data posing several challenges such as data imputation for missing values and transformations to adjust for outliers. The nature of the data set also does not allow for scientifically rigorous data collection processes. These aspects are sure to lead to high model variance not matter which model is applied.

2 Exploratory Data Analysis

The data contained 25 different variables along with 423,857 observations. The response variable was the asking price labeled as "Price." Table 1 contains a complete list and description of each variable. The factor levels of the categorical variables that were retained for the study can be found in the appendix.

The Year variable may be viewed as either qualitative or quantitative. We chose to view it as quantitative due to the lack of such predictors. Therefore, our model could only be applied to vehicles in the specified range of the dataset corresponding to the year.

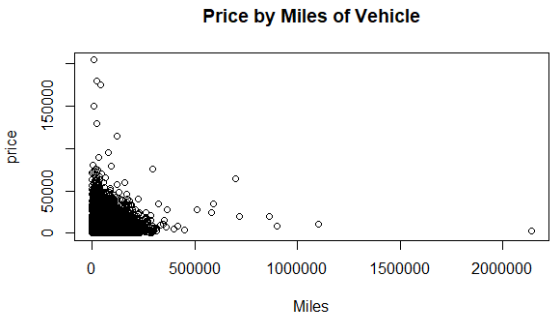


Figure 1: The odometer predictor plotted against the response price.

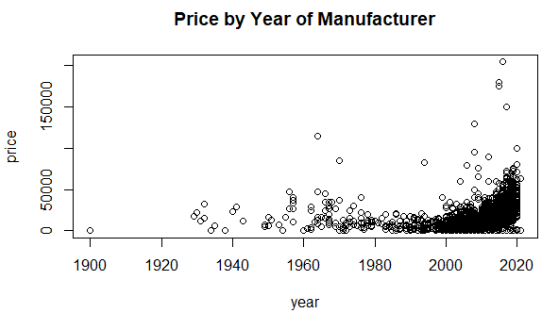


Figure 2: The model year predictor plotted against the response price.

Variable Name	Type	Description
Price (Response)	Continuous	US dollar amount the seller is asking for the vehicle.
ID	Integer	Unique number identifying each observation.
URL	Text	Wed address for the ad.
Region	Categorical	The geographical region.
Region Url	Text	The web address for the region.
Year	Continuous	Year vehicle was manufactured.
Manufacturer	Categorical	Company that manufactured the vehicle.
Model	Categorical	The model version of the vehicle.
Condition	Categorical	The condition the vehicle is in.
Cylinders	Categorical	Number of cylinders the engine operates on.
Fuel	Categorical	Type of fuel the vehicle uses.
Odometer	Continuous	The number of miles the vehicle has been driven.
Title Status	Categorical	The type of Title of Ownership.
Transmission	Categorical	Type of transmission in the vehicle.
VIN	Text	Unique identifying number belonging to the vehicle. The "number" can have a combination of letters and numbers.
Drive	Categorical	Types of drivetrain.
Size	Categorical	Sizes of vehicles.
Type	Categorical	Generic types of vehicles.
Paint Color	Categorical	Vehicle's paint color.
Image URL	Text	Web address of photos of the vehicle.
Description	Text	Seller's written description of the vehicle and any information they feel is pertinent.
County	Categorical	County of the state the vehicle is in.
State	Categorical	State the vehicle is in.
Lat	Double	The latitudinal coordinates in which the vehicle is being sold.
Long	Double	The longitudinal coordinates in which the vehicle is being sold.

Table 1: Full listing and description of variables obtained from the data source. Note that the VIN is a unique number assigned to a vehicle but may appear multiple times in the data if there are multiple listings of the vehicle.

Several variables were selected on the onset the be removed due to the researcher's belief that they lacked predictive power. Some of the variables that were dropped may have predictive

information but fell outside of the ability to model with regression. For instance, the Description field could be anywhere from zero to hundreds of characters. A natural language processing algorithm may find this data useful in a different type of project. A more thorough discussion of these issues can be found in the recommendations section.

A full list of variables that were initially dropped include:

- ID
- URL
- Region
- Region URL
- Image URL
- Description
- State
- County
- Longitude
- Latitude
- VIN

The two quantitative variables are plotted against the response variable in figures 1 and 2 above. There does not seem to be a discernible pattern in figure 1, while the year and price seem to be bimodal with a near exponential relationship in the newer vehicles. This makes sense as newer vehicles are generally worth more since they are likely to be in better shape, but older vehicles may demand higher values if they are sought as collector's items.

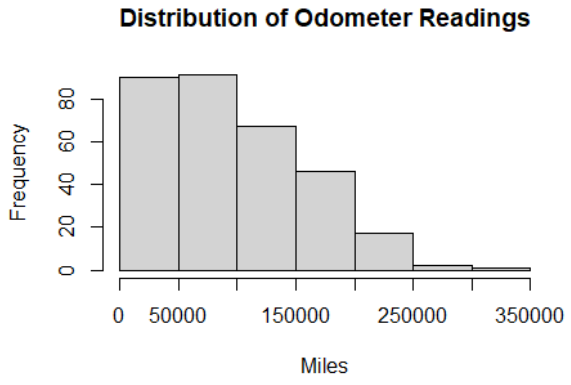


Figure 3: The distribution of miles observed on the vehicles. The distribution is clearly right skewed.

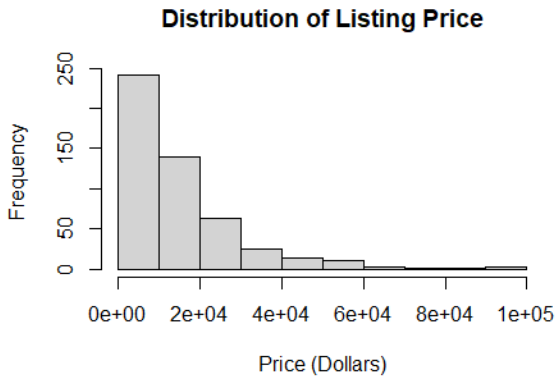


Figure 4: The distribution of asking price of the vehicles. The distribution is also clearly right skewed and even more distinctive than miles.

The distribution of the odometer and the response variable can be seen in figures 3 and 4. Both are right skewed and indicate that both will require some data transformations. Figure 5 below shows the distribution of the year which is clearly left skewed.

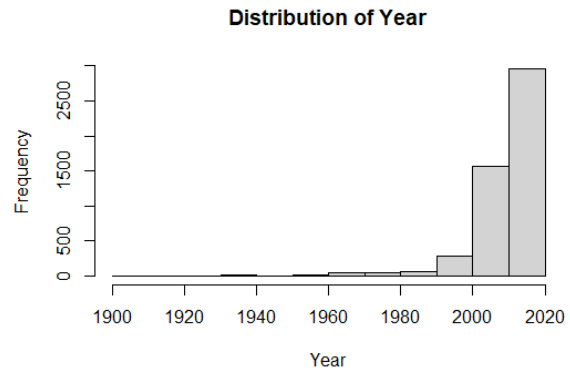


Figure 5: Distribution of the year of manufacture can be seen to be heavily left skewed.

Finally, while exploring the data we found that the categorical variable "model" too difficult to work with. There were more than 2,700 different categories. Since the data was submitted without a standard format, this caused the high number of different models. One example we found was a model named "S-10" and another named "S10." Data cleaning efforts were made but became too intensive under such short timelines. We made the decision that the inclusion of manufacturer and type contained much the same information. Thus, we felt comfortable removing the "model" predictor from the data.

3 Preprocessing

First, we must be mentioned a difficulty we had with the data. The large number of observations caused us to experience extremely long runtimes on some of our models. Thus, we randomly sampled 5000 accounts to build our models on. This caused us to perform the preprocessing over again. Every step that follows was conducted on the smaller sample.

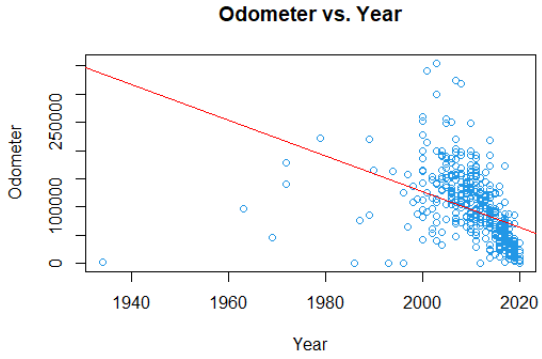


Figure 6: A linear model (red) was investigated in the hopes that it could be used to impute values in the odometer field.

The most important task in the preprocessing stage was the data imputation. Roughly 25% of the fields had missing values or values labeled as "NA." A relationship between odometer and year was initially investigated with the hope that a model could be generated to impute values into the odometer field. Figure 6 shows the relationship between year and odometer on a sample of 500 accounts. It seems that the relationship is fairly linear in more modern vehicles, but the variance increases as the vehicle ages. Although the *p*-value was highly significant at 2.2×10^{-16} the clear lack of the homoscedastic assumption and a less than impressive adjusted R-squared score of 0.18. Other variables were briefly added to this model and investigated but they did not return the desired results we were seeking. Thus, this method was abandoned for the KNN imputation method. Resampling also showed more outliers, reinforcing our decision.

Next, we created dummy variables for the categorical variables. Then the `preProcess()` function was used with the following methods indicated for the arguments:

- Near Zero Variance
- Box-Cox Transformation
- Center
- Scale
- KNN Impute
- PCA
- Spatial Sign

This process left us with 57 predictors, including 26 principal components. The dummy variable corresponding to the different levels of the title status were dropped as part of the near zero variance method. Figure 7 shows the correlation plot of the predictors after preprocessing. There are a few spots that indicate high variance but there does not seem to be any significant patterns of multicollinearity.

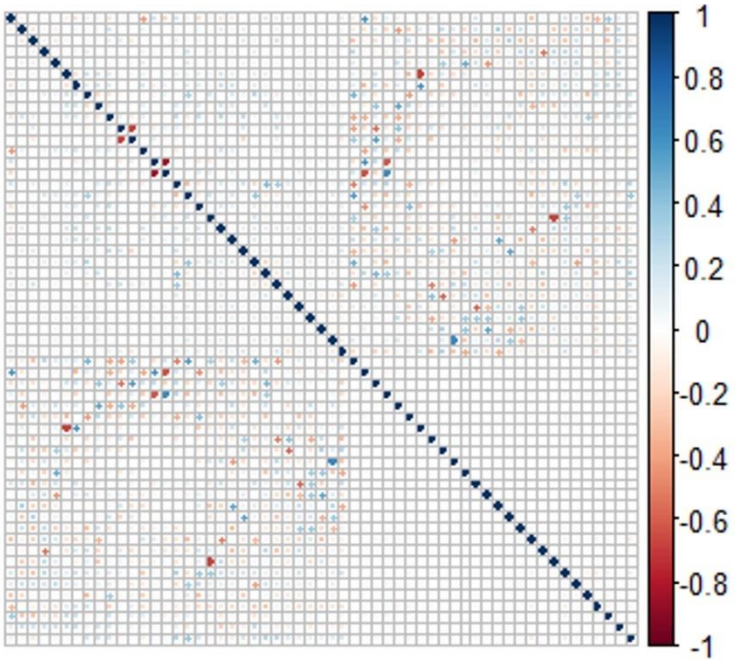


Figure 7: The correlation plot of the variables after pre-processing. The labels were excluded due to the number of variables in the model.

Next, we perform the principal component analysis on the pre-processed data. The variance in the data is quite spread out as the component accounting for the highest variance is just 6.18%. Figure 8 below shows the scree plot. The added variation drops off significantly after the 30th component. We chose to drop all components after the 37th component as this was equivalent to dropping all variable that contributed less than 0.5% of the variation. An argument could be made to drop more of the components but the researchers wanted to error on the side of conservative.

The R function `findCorrelation()` was then used to find high correlations with the threshold set to 0.75. The variable corresponding to the automatic transmission was returned and subsequently dropped. This was the final variable that was dropped in our process. The correlation plot containing all variables that models were built on can be seen in figure 9 below.

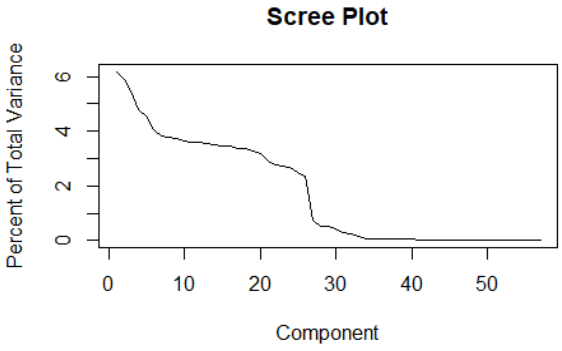


Figure 8: Scree plot. The variance is relatively spread out among the components.

The skewness values were checked after these variables were dropped as a final quality control measure. The minimum value returned was -2.689 while the maximum reported was 3.959. This satisfied the rule of thumb indicated by Kuhn (2016) and was acceptable for us.

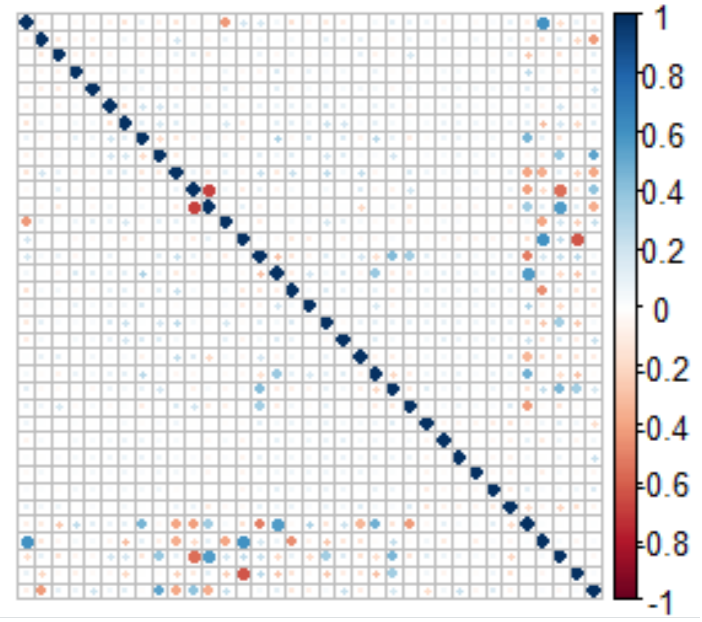


Figure 9: Correlation matrix of the remaining variables which models were built on.

Lastly several boxplots were investigated of the remaining transformed variables. Although most plots returned showed very well-behaved behavior, some variables seemed to exhibit a significant number of outliers. Figure 10 below is a sample of theses that we feel is representative of the data. A full list of box plots of these variables can be found in the appendix. Figure 11 shows the distribution of the two quantitative predictors after pre-processing. The distributions seem to have improved quite a bit through the pre-processing.

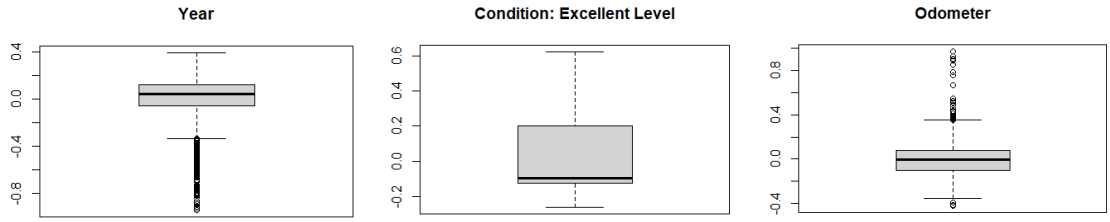


Figure 10: Selected box plots of transformed data. From left: Year, Condition at the Excellent factor, Odometer.

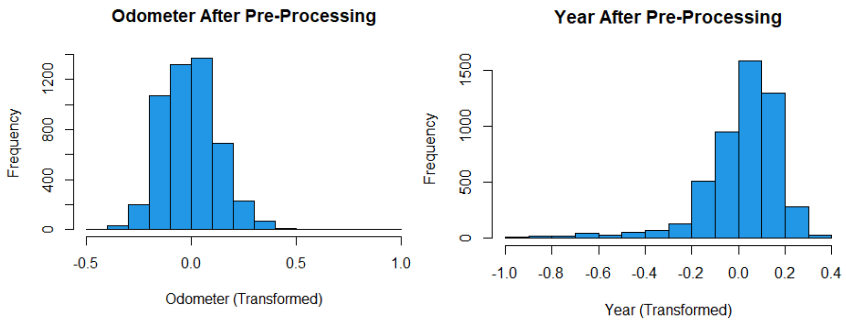


Figure 11: Distribution of the transformed odometer (left) and year(right) values.

Data Splitting and Resampling

The data was randomly split in the standard 80% training and 20% testing set sizes. The training set is therefore 4,000 observations while the testing set is 1,000. We use 10-fold cross validation as our resampling strategy throughout the study. Other data splitting strategies were discussed but were not incorporated in this study. A more detailed discussion follows in the recommendations section.

Model Fitting

Being As the goal of this project was to predict the price, a continuous response, we model several regression models including linear and nonlinear varieties. The statistic of main importance to analyzing our models will be the root mean squared error (RMSE) with the coefficient of determination (R^2) as our secondary statistic. We first compute these values on the training set and after selection of our top performing models, compute the values on the testing set.

Models		
Linear	**RMSE**	**R^2**
Linear Regression	75,477	0.090
Ridge Regression	74,932	0.108
LASSO	70,891	0.056
Elastic Net	72,713	0.105
Partial Least Squares	69,785	0.149
Non-Linear	**RMSE**	**R^2**
Neural Network	68,810	0.248
MARS	68,371	0.254
Support Vector Machine	67,821	0.327

Table 2: Summary of statistical results of the models that were fit to the training data.

The models and their performance on the training set can be found in table 2.

A. Linear Regression

The linear regression model was run as a base line model to show the benefits of the more advanced models. We did not pursue this model.

B. Ridge Regression

The ridge regression model was only slightly better performing than the linear regression model. Figure 12 shows the RMSE plot as the decay parameter increases.

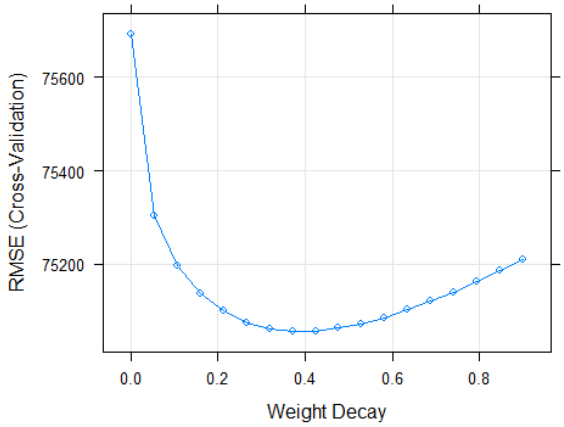


Figure 12: RMSE plotted against the decay parameter in the Ridge Regression model.

C. Least Absolute Shrinkage and Selection Operator

The LASSO model showed a decrease in accuracy compared to the Ridge Regression model. The RMSE plot, figure 13, did not show signs that widening the parameter range would improve the model. With such low R^2 scores we did not feel it was worth pursuing this model further.

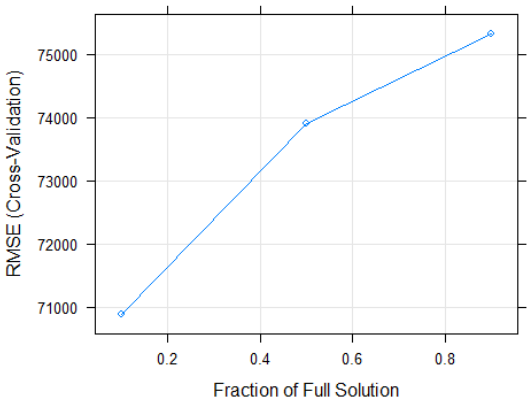


Figure 13: RMSE plot for LASSO model.

D. Elastic Net

The Elastic net model originally produced the results in table 2 above. The RMSE plot showed signs that expanding the tuning parameters may be helpful. Figure 14 and 15 shows the prior and post tuned RMSE plots. It seems that the appearance of possible improvement was not realized after tuning.

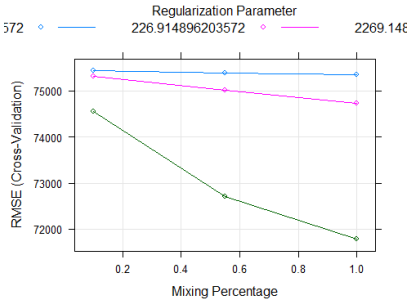


Figure 14: The RMSE plot for the Elastic Net Model.

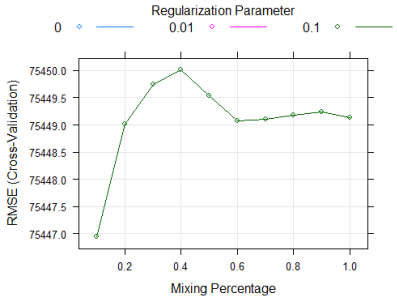


Figure 15: The RMSE plot for the Elastic Net Model with a wider margin of tuning parameters.

E. Partial Least Squares

The Partial Least Squares model also did not perform well. Similar to the LASSO model the RMSE plot, figure 16, did not show signs that the parameters could be successfully tuned. At this point we were ready to move on to the non-linear models.

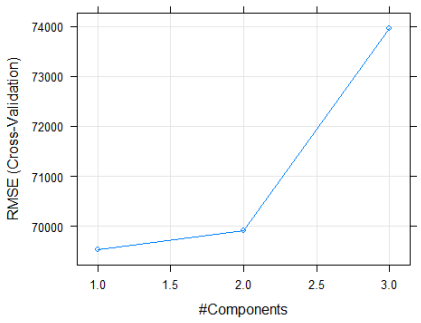


Figure 16: RMSE plot for the PLS model.

F. Neural Network

The Neural network algorithm was applied and it did return significantly improved performance over the linear models. The RMSE plot can be seen in figure 17. It took 2 to 3 hours to train the model and perhaps with more time a better tuned model could be obtained. The plot of the predicted values against the actual values in figure 18 does show some signs of linearity, but clearly there is an increase in variance.

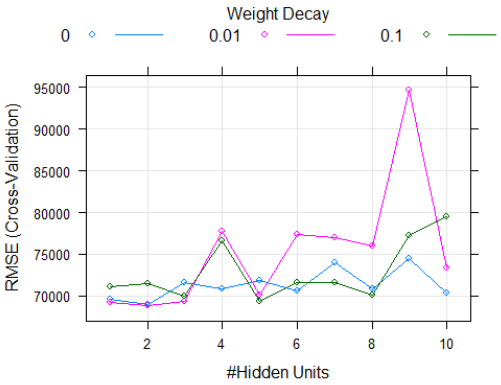


Figure 17: The RMSE plot for the Neural Network model.

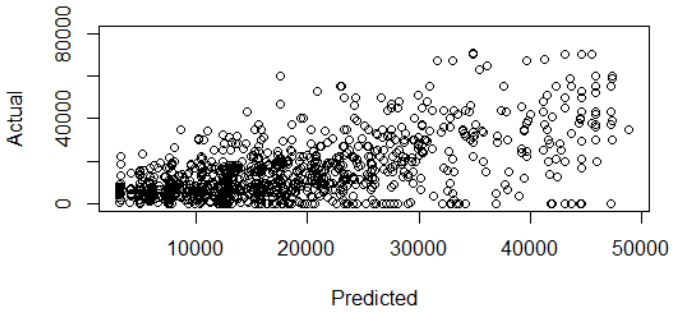


Figure 18: The actual values plotted against the predicted values for the Elastic Net Model.

G. Multivariate Adaptive Regression Splines

The MARS model was originally tuned on a range of degrees from 1 to 5. The 5th degree model was significantly worse than the other four. It was re-run on the range from 1 to 4 and produced the statistics in table 2. The MARS model was another improvement from the previous models and was eventually selected as one of the final two candidate models to analyze on the test set in the model selection phase. Figure 19 shows the RMSE plot for the model.

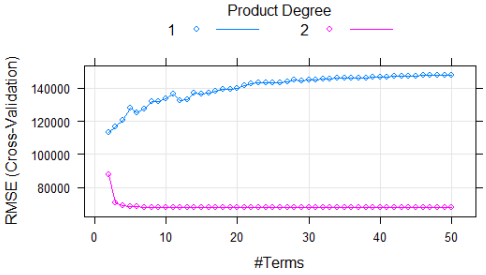


Figure 20: RMSE plot for the MARs model.

H. Support Vector Machine

The support vector machine model was the final model evaluated in this study and it performed the best on the training set. It was the second of the two models that we selected to analyze in the model selection phase. Figure 20 shows the RMSE plot for this model. Further diagnostic plots will be investigated in the next section.

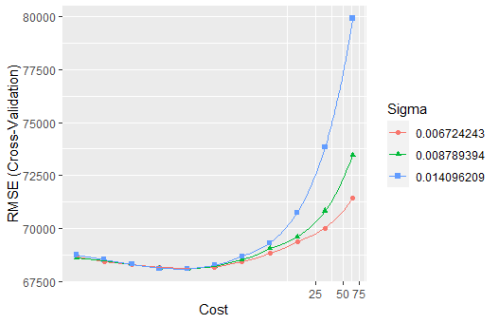


Figure 20: RMSE plot for the SVM model with three sigma values.

Model Selection

In this section we select the two models that best performed in the previous section. That is we compare the Support Vector Machine model with the MARS model. Figures 21 and 22 show their predicted values plotted against the actual values.

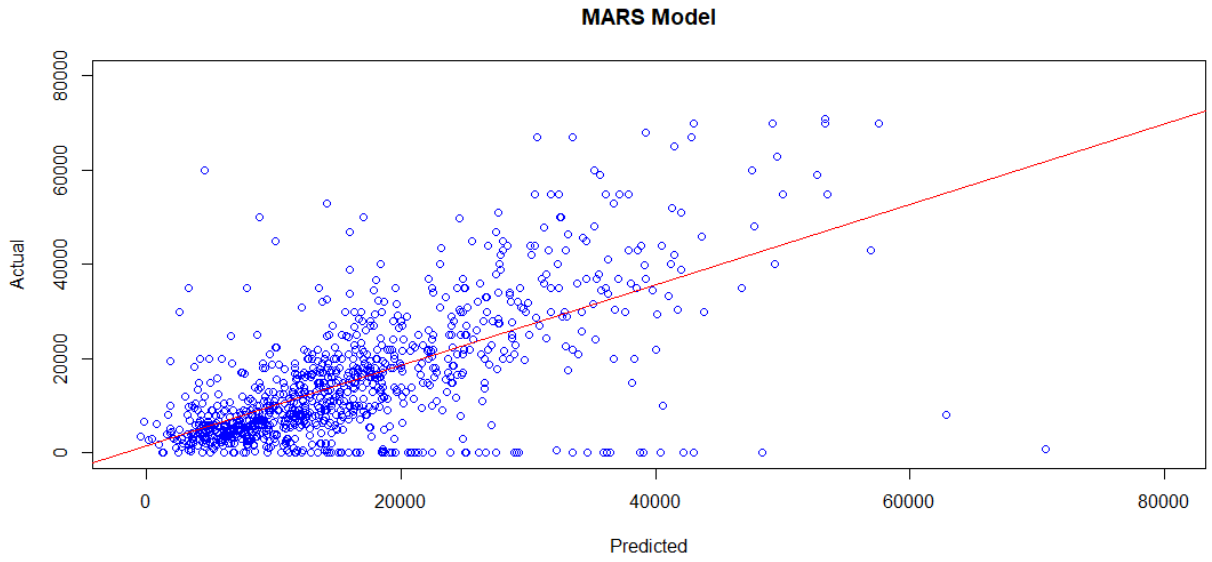


Figure 21: The actual values plotted against the predicted values of the MARS model.

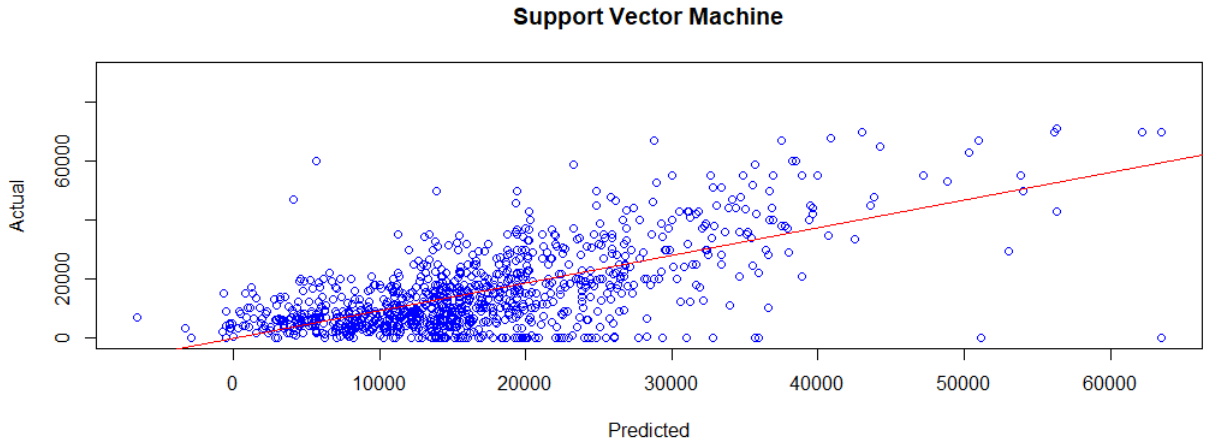


Figure 22: The actual values plotted against the predicted values of the SVM model.

Both models appear to perform much better than the Elastic Net model plotted above in figure 18. The primary issue with both models is the prediction of vehicles that turn out to be $0 or very near that. This may be acceptable since vehicles with such a low asking price might have a

severe issue that the model isn't recognizing. Next we plot the residuals against the fitted values below in figure 23.

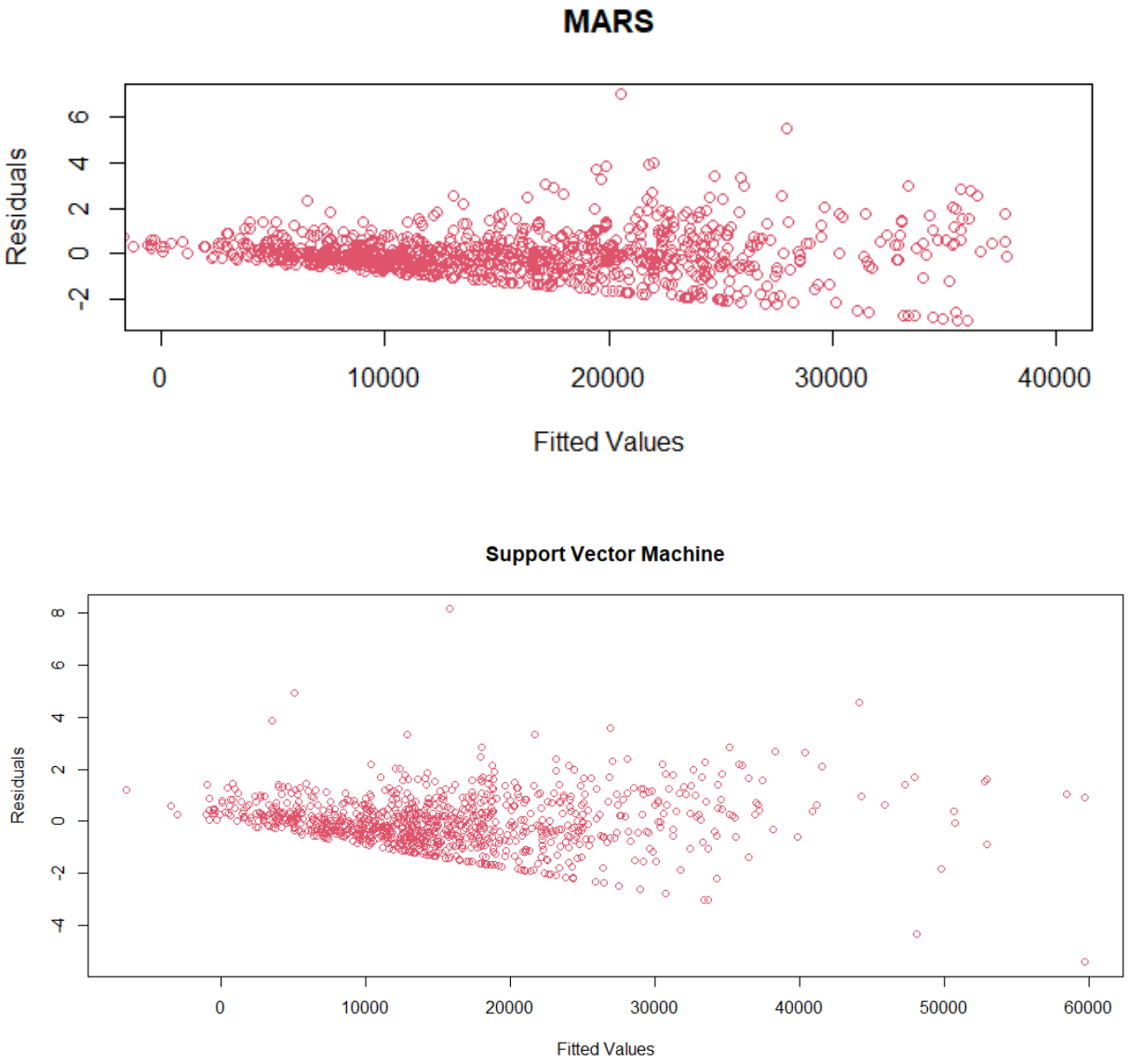


Figure 23: The residuals plotted against the fitted values for the MARS model (top) and the SVM model (bottom).

Both models show very similar patterns in that there is an increase in variance as the predicted prices increase but most points do seem to be within the (-2,2) range. The ridged boundary on the bottom of these plots indicate that the variance can only go so far since no one appears to be paying people to take their car away.

Finally we visualize the model performance with the following plots in figure 24 below.

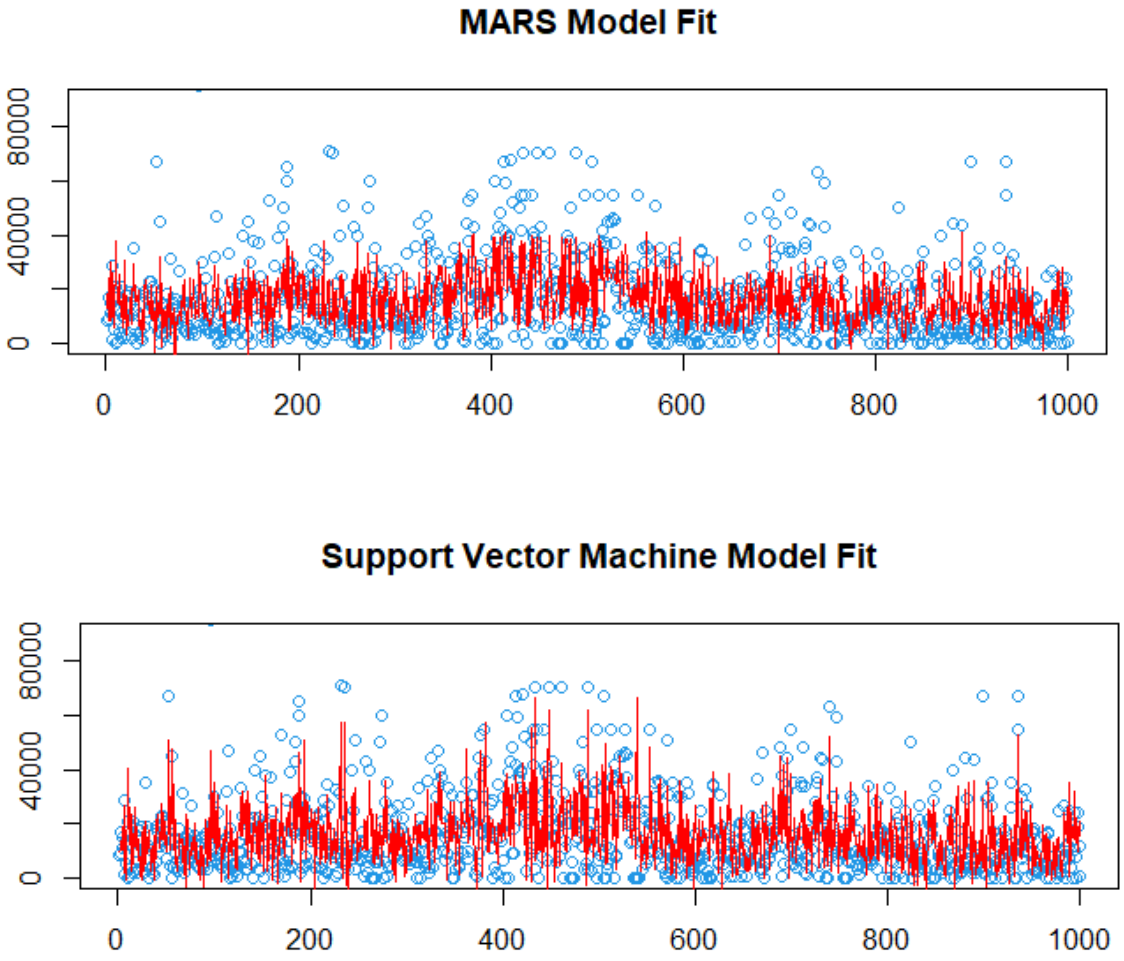


Figure 24: The ordered observations of the test set (blue) and the model prediction line (red) for the MARS model (top) and the SVM model (bottom).

Since the data has several dimensions it is hard to conclude whether either of these is over or under fit based on these plots but it does seem that the Support Vector Machine model chases the higher values more than the MARS model does.

The RMSE and R^2 scores for their performance on the test set is in table 3.

Model	RMSE	R^2
MARS	12,358	0.308
Support Vector Machine	11,418	0.415

Table 3: The RMSE and the R^2 scores for the Multivariate Adaptive Regression Splines model and the Support Vector Machine Model for their performance on the test set.

Since the plots do not provide any strong evidence that the MARS model is better suited for this data than the Support Vector Machine model, we conclude that the Support Vector Machine model is the best model developed for the data based on its superior RMSE and R^2 values.

Discussion and Recommendations

The most challenging part of this study was the data cleaning and imputation along with the enormous number of observations. Though the support vector machine model performed reasonably well, we believe that a more thorough hands-on approach to cleaning the data could yield better performance results. For example, we believe that the vehicles with asking price of $0 needs to be investigated meticulously to help reduce the overall variance of the model.

Nonetheless, we do believe that the model performed quite well on the middle range of vehicles. If one were in the market for a vehicle in the range of say $20,000-$30,000, you could input the various variables of a vehicle that you are interested in into the model to check if the asking price was fair or not.

We also believe that the data could be quite useful in other types of studies. Monitoring the change in model behavior over time may lend itself to economic studies as well as developing more sophisticated models. The description field is ripe for a natural language processing study that may be useful in the marketing field.

Another option future researchers may be interested in is binning the year variable into decades. It seems that this variable accounted for much of the variation in prices. One may be able to develop more accurate models by using this variable as a categorical variable. We chose to use it as a continuous variable mostly out of an academic need to retain the few continuous variables that we had.

Lastly, since the sample size was so large and we were forced to only use a small sample of observations due to runtime issues, resampling from the primary population and averaging models may also prove fruitful.

References

Kuhn, M. Johnson, K. (2016). *Applied Predictive Modeling.* Springer Books.

Reese, A (2020). "Used Car Datasets: Vehicles listings from Craigslist.org." Data Repository. Retrieved from: https://www.kaggle.com/austinreese/craigslist-carstrucks-data.

Appendix 1

This appendix contains the factor levels for all categorical variables that were included in our modeling.

Variable	Manufacturer
Levels	5
Levels:	
Ford	
Chevrolet	
Toyota	
Honda	
Dodge	

Variable	Condition
Levels	5
Levels:	
New	
Like New	
Excellent	
Good	
Salvage	

Variable	Cylinders
Levels	3
Levels:	
4	
6	
8	

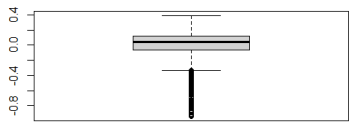
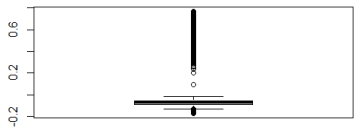
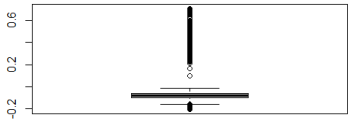
Variable	Fuel
Levels	2
Levels:	
Gas	
Diesel	

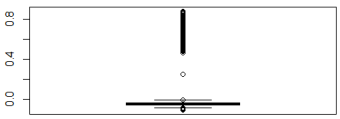
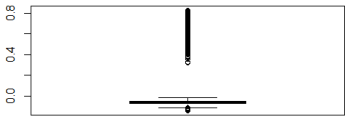
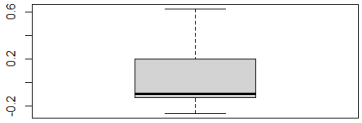
Variable	Drive
Levels	5
Levels:	
Front Wheel Drive	
Rear Wheel Drive	
Four Wheel Drive	
All Wheel Drive	

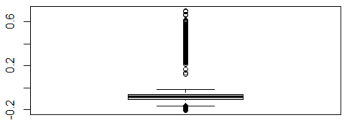
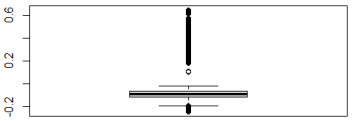
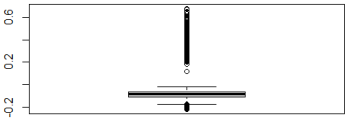
Variable	Type
Levels	7
Levels:	
Compact	
Mid-Size	
Full Size	
Sedan	
Pickup	
SUV	
Truck	

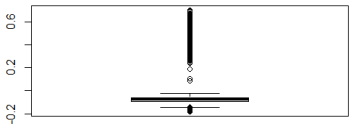
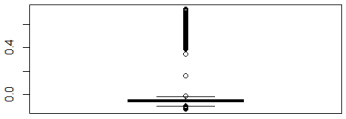
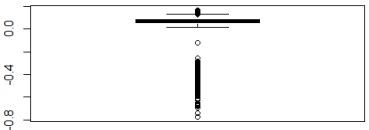
Appendix 2

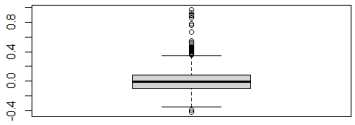
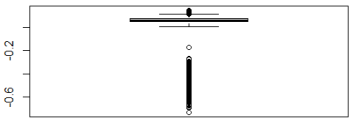
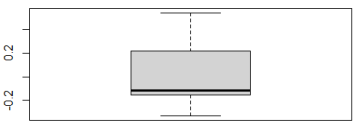
Box Plots of Transformed Variables

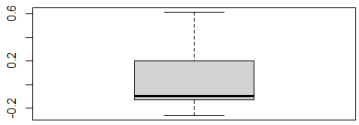
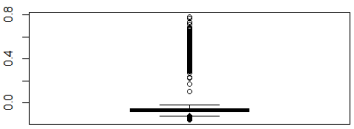
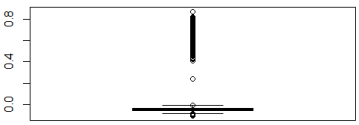




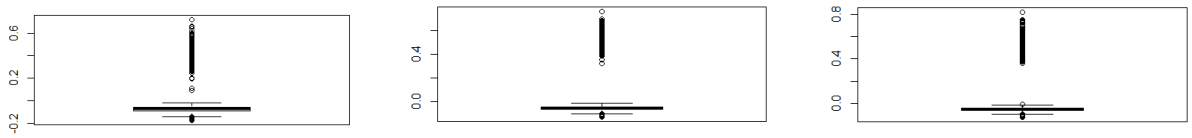




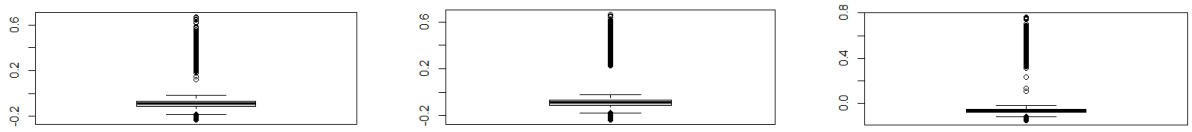




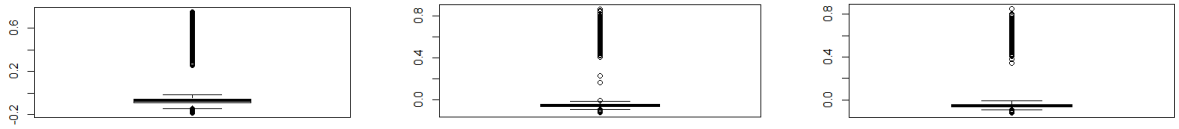




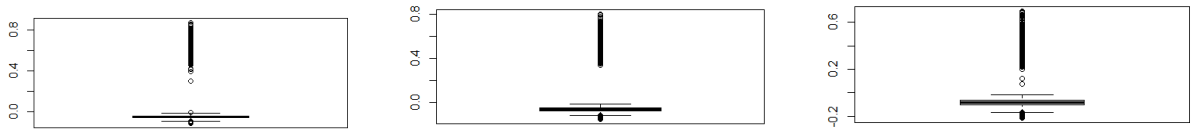




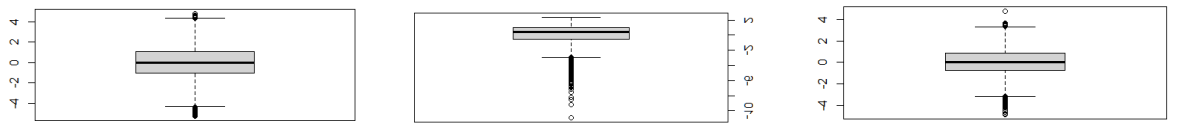




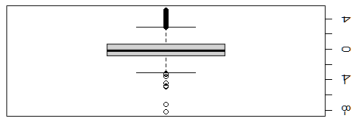

-0.2
0.2
0.6
0.0
0.4
0.8


-0.2
0.2
0.6
0.0
0.4
0.8


-0.2
0.2
0.6
0.0
0.4
0.8


0.0
0.4
0.8
-0.2
0.2
0.6


-4
-2
0
2
4
-10
-8
-6
-4


-8
-4
0
4
-4
0
2
4
6

Appendix 3

R Code

```
vehicle<-read.csv('../Data/vehicles.csv')

library(earth)
library(caret)
library(e1071)
library(kernlab)
library(corrplot)
library(elasticnet)
library(glmnet)


View(vehicle)
names(vehicle)
#Sample 5000 accounts and drop non-predictive data
dataSet = sample(nrow(vehicle), 5000)
df2=vehicle[dataSet,-c(1:4,8,15,20:25)]
names(df2)
head(df2)
write.csv(df2,"vehicle1.csv")
vehicle1<-read.csv('vehicle1.csv')
dim(vehicle1)
####Before Preprocessing
hist(vehicle1$odometer)
hist(vehicle1$price)
hist(vehicle1$year, main ="Distribution of Year", xlab = "Year")
boxplot(vehicle1$odometer, main="Odometer")
boxplot(vehicle1$price,main="Price")
skewness(vehicle1[vehicle1$odometer == "NA",])
skewness(vehicle1$price)


####


####Preprocess
```

```
price<-vehicle1$price
names(vehicle1)
pvehicle<-vehicle1[,-c(1:2)]

set.seed(1)
sampleData = vehicle1[sample(nrow(vehicle1),500), ]
plot(odometer~year, data = sampleData,
    ylab = "Odometer", xlab = "Year", col = 12,
    main = "Odometer vs. Year")
sd.mod = lm(odometer~year, data=sampleData)
abline(sd.mod, col = "red")
summary(sd.mod)



head(pvehicle)
nzv = nearZeroVar(pvehicle);nzv
dc =dummyVars(~.,data=pvehicle,fullRank = TRUE)
veh=predict(dc,pvehicle); veh
typeof(veh)
pp <- preProcess(veh,method=c("nzv", "BoxCox",
"center","scale","knnImpute","pca","spatialSign")) ## need {caret} package
veh=predict(pp,veh)
names(as.data.frame(veh))
head(veh, 2)

pcaObject = prcomp(veh, center = TRUE, scale =TRUE)
perVar = (pcaObject$sdev)^2/sum((pcaObject$sdev)^2)*100;perVar

#Scree Plot
x = 1:length(perVar)
plot(perVar~x, type = 'l', main = "Scree Plot",
    ylab = "Percent of Total Variance", xlab = "Component")

#veh = veh[,-c(37:57)] #Only do this once!!!
ncol(veh)
corrplot(cor(veh), labels = FALSE)



tooHigh = findCorrelation(cor(veh), cutoff = 0.75);tooHigh
#veh = veh[,-tooHigh] #Also just once!!!
```

```
corrplot(cor(veh), labels = FALSE)


names(as.data.frame(veh))


skewValues = sapply(as.data.frame(veh), skewness);skewValues
min(skewValues)
max(skewValues)

boxplot(veh[,1], main = "Year")
boxplot(veh[,2])
boxplot(veh[,3])
boxplot(veh[,4])
boxplot(veh[,5])
boxplot(veh[,6], main = "Condition: Excellent Level") #Condition Excellent
boxplot(veh[,7])
boxplot(veh[,8])
boxplot(veh[,9])
boxplot(veh[,10])
boxplot(veh[,11])
boxplot(veh[,12])
boxplot(veh[,13], main = "Odometer") #Odometer
boxplot(veh[,14])
boxplot(veh[,15])
boxplot(veh[,16]) #4wd
boxplot(veh[,17]) #fwd
names(as.data.frame(veh))
hist(as.data.frame(veh)$year, col = 12,
    main = "Year After Pre-Processing",
    xlab = "Year (Transformed)")
hist(as.data.frame(veh)$odometer, col = 12,
    main = "Odometer After Pre-Processing",
    xlab = "Odometer (Transformed)")
as.data.frame(veh)

set.seed(1)
train2 <- createDataPartition(price,  p = .80,  list= FALSE)
ptrain <- veh[train2,]
```

```
ytrain<-price[train2]
ptest <- veh[-train2,]
ytest<-price[-train2]

#Resampling strategy used thruought
ctrl<-trainControl(method="cv", number=10)

####linear
set.seed(1)
lmod<-train(ptrain,ytrain,method="lm",trControl=ctrl,
          tuneLength = 10)
lmod
lm.res = residuals(lmod)
plot(lm.res~fitted(lmod), ylim = c(-10000,10000))
#lmod_pred<-predict(lmod,ptest)
#postResample(pred=lmod_pred,obs=ytest)

#####ridge
set.seed(1)
ridgeGrid <- data.frame(.lambda = seq(0, .9, length = 18))
ridgeRegFit <- train(ptrain, ytrain, method = "ridge",
                     tuneGrid = ridgeGrid, trControl = ctrl)
rid_pred<-predict(ridgeRegFit,ptest)
postResample(pred=rid_pred,obs=ytest)
ridgeRegFit
plot(ridgeRegFit)

#####lasso
set.seed(1)
lass<-train(ptrain,ytrain,method="lasso",trControl=ctrl)
lass
plot(lass)
lass_pred<-predict(lass,ptest)
postResample(pred=lass_pred,obs=ytest)

####elastic net
set.seed(1)
enetGrid <- expand.grid(.lambda = c(0, 0.01, .1), .alpha = seq(0.1, 1, length = 10))
elas<-train(ptrain,ytrain,method="glmnet",trControl=ctrl)
elas
```

```
plot(elas)
elas_pred<-predict(elas,ptest)
postResample(pred=elas_pred,obs=ytest)
ptrain[1:4,]


######################
####Neural Network
set.seed(1)
nnetGrid <- expand.grid(.decay = c(0, 0.01, .1),
                        .size = c(1:10),
                        ## The next option is to use bagging (see the
                        ## next chapter) instead of different random
                        ## seeds.
                        .bag = FALSE)

nnetTune <- train(ptrain, ytrain,
                  method = "avNNet",
                  tuneGrid = nnetGrid,
                  trControl = ctrl,
                  ## Automatically standardize data prior to modeling
                  ## and prediction
                  preProc = c("center", "scale"),
                  linout = TRUE,
                  trace = FALSE,
                  MaxNWts = 10 * (ncol(ptrain) + 1) + 10 + 1,
                  maxit = 500)
nnetTune

nnet_pred<-predict(nnetTune,ptest)
postResample(pred=nnet_pred,obs=ytest)

plot(ytest~nnet_pred, xlab = "Predicted", ylab = "Actual", ylim = c(0,80000))
length(nnet_pred)
plot(ptest~nnet_pred)

#### Plots for Neural Network Model ###
#RMSE~Cost
plot(nnetTune)
```

```
nnet_Model = lm(ytest~nnet_pred)
#diagnostics
plot(nnet_Model, col = 4)
#Actual~Predicted
plot(ytest~fitted(nnet_Model), xlab = "Predicted", ylab = "Actual", col = "blue",
    main = "Neural Network Model", ylim=c(0,90000), xlim = c(5000,80000))
abline(nnet_Model, col = "red")

#Res~Fit for Actual~Predicted
marsRes = rstandard(mars_Model)
plot(marsRes~fitted(mars_Model), ylab = "Residuals", xlab = "Fitted Values",
    main = "MARS", col = 10, xlim = c(0,80000))

#Ordered Observations~Model
x = 1:length(ytest)
plot(x, ytest, pch=18, col=4, xlab ="", ylab = "", ylim=c(0,90000), main = "MARS Model Fit")
lines(x, mars_pred, lwd="1", col="red")
####
#####################################################
####
####




####
#MARS Model


set.seed(1)
marsGrid <- expand.grid(.degree = 1:4, .nprune = 2:50)

marsTuned <- train(ptrain, ytrain,
                   method = "earth",
                   tuneGrid = marsGrid,
                   trControl = trainControl(method = "cv"))
marsTuned
```

```
marsTuned$bestTune
mars_pred<-predict(marsTuned,ptest)
postResample(pred=mars_pred,obs=ytest)

#### Plots for MARS Model ###
#RMSE~Cost
plot(marsTuned)


mars_Model = lm(ytest~mars_pred)
#diagnostics
plot(mars_Model, col = 4)
#Actual~Predicted
plot(mars_pred~ytest, xlab = "Predicted", ylab = "Actual", col = "blue",
    main = "MARS Model", ylim=c(0,90000), xlim = c(0,50000))
abline(mars_Model, col = "red")

#Res~Fit for Actual~Predicted
marsRes = rstandard(mars_Model)
plot(marsRes~fitted(mars_Model), ylab = "Residuals", xlab = "Fitted Values",
    main = "MARS", col = 10, xlim = c(0,40000))

#Ordered Observations~Model
x = 1:length(ytest)
plot(x, ytest, col=12, xlab ="", ylab = "", ylim=c(0,90000), main = "MARS Model Fit")
lines(x, mars_pred, lwd="1", col="red")
####
#####################################################
####
####


###################################
# SVM Model Radial Basis Function #
###### Not included in paper ######
###################################
set.seed(1)
svmRTuned <- train(ptrain, ytrain,
                   method = "svmRadial",
                   preProc = c("center", "scale"),
```

```
            tuneLength = 14,
            trControl = trainControl(method = "cv"))
svmRTuned
svm_pred<-predict(svmRTuned,ptest);
postResample(pred=svm_pred,obs=ytest)

#### Plots for SVM Model ###
#RMSE~Cost
ggplot(svmRTuned)+coord_trans(x='log2')

#Actual~Predicted
svm_Model = lm(ytest~svm_pred)
plot(svm_Model, col = 4)
plot(svm_pred~ytest, xlab = "Predicted", ylab = "Actual")

#Ordered Observations~Model
x = 1:length(ytest)
plot(x, ytest, pch=18, col="red", xlab ="", ylab = "", ylim=c(0,90000), main = "Support Vector
Machine Model Fit")
lines(x, svm_pred, lwd="1", col="blue")
abline(lm(svm_pred~x), col = "black")

#Res~Fit for Actual~Predicted
svmRes = rstandard(svm_Model)
plot(svmRes~fitted(svm_Model), ylab = "Residuals", xlab = "Fitted Values",
    main = "Support Vector Machine", col = 10)

plot(svm_pred~ytest, xlab = "Predicted", ylab = "Actual", col = "blue",
    main = "Support Vector Machine", ylim = c(0,90000), xlim = c(0,80000))
abline(lm(ytest~svm_pred), col="red")
####
#####################################################
####
####

############################
### Second SVM modeling ###
#### Included in Paper ####
############################
```

```
sigmaRangeReduced <- sigest(as.matrix(ptrain)); sigmaRangeReduced
svmRGridReduced <- expand.grid(.sigma = sigmaRangeReduced,
                               .C = 2^(seq(-4, 6)))


svmR2Tuned <- train(ptrain, ytrain,
                    method = "svmRadial",
                    preProc = c("center", "scale"),
                    tuneLength = 14,
                    tuneGrid = svmRGridReduced,
                    trControl = trainControl(method = "cv"))


svmR2Tuned

svm2_pred<-predict(svmR2Tuned,ptest);
postResample(pred=svm_pred,obs=ytest)

#### Support Vector Machine 2 Plot ###
###
#RMSE
ggplot(svmR2Tuned)+coord_trans(x='log2')

svm2_Model = lm(ytest~svm2_pred)
plot(svm2_Model, col = 4)
plot(ytest~svm2_pred, xlab = "Predicted", ylab = "Actual",
    ylim = c(0,90000), main="Support Vector Machine", col = "blue")
abline(svm2_Model, col = "red")


#Ordered Observations~Model
x = 1:length(ytest)
plot(x, ytest, col=12, xlab ="", ylab = "", ylim=c(0,90000), main = "Support Vector Machine
Model Fit")
lines(x, svm_pred, lwd="1", col="red")
abline(lm(svm_pred~x), col = "black")

#Res~Fit for Actual~Predicted
svm2Res = rstandard(svm2_Model)
plot(svm2Res~fitted(svm2_Model), ylab = "Residuals", xlab = "Fitted Values",
```

```
    main = "Support Vector Machine", col = 10)



####PLS
set.seed(1)
plsmod<-train(ptrain,ytrain,method="pls",trControl=ctrl)
plsmod
plot(plsmod)
pls_pred<-predict(plsmod,ptest)
postResample(pred=pls_pred,obs=ytest)


####bestmodel
library("vip")
vip(svmR2Tuned)
vip(marsTuned)
vip(nnetTune)
vip(elas)
vip(lass)
vip(lmod)
vip(rid)
```