Assignment #2: Predictive Modeling in Regression

In this assignment we look at the diamonds dataset and aim to predict the response variable ‘price’ using available predictors such as a caret, color, and clarity. Since price is a numerical variable we are dealing with a regression problem. Before performing any statistical analysis, however, let’s perform a data quality check and an exploratory data analysis. I begin by loading the data into R using the function read.csv().

(1) Data Quality Check

In this section I provide a quick summary of the values of the diamonds data so that I can better understand the value ranges, shape of the distributions, and the number of missing values for each variable in our data set. In this assignment we are dealing with a regression problem since the goal of the assignment is to predict the price of each diamond based upon the 6 predictor variables. From the str() and summary() function outputs below we find that the diamonds data set consists of 425 observations and 7 variables. Of the 7 variables there are 4 numerical variables describing carat, color, clarity, and the price of the diamond, and there are 3 categorical variables that describe a diamond’s cut, channel, and store of purchase.

```
## 'data.frame':    425 obs. of  7 variables:
##  $ carat  : num  0.826 0.996 1.07 1.07 1.01 0.66 0.701 0.97 0.74 2.04 ...
##  $ color  : int  4 5 4 7 8 3 4 8 1 5 ...
##  $ clarity: int  7 6 7 7 6 4 8 6 9 6 ...
##  $ cut    : Factor w/ 2 levels "Ideal","Not Ideal": 1 1 1 2 2 1 1 2 2 2 ...
##  $ channel: Factor w/ 3 levels "Independent",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ store  : Factor w/ 12 levels "Ashford","Ausmans",..: 7 7 7 7 7 7 7 7 7 7 ...
##  $ price  : int  7775 9850 10950 7500 6995 6100 6300 4850 5895 23000 ...
```

```
##      carat           color          clarity              cut
##  Min.   :0.200   Min.   :1.000   Min.   : 2.000   Ideal    :154
##  1st Qu.:0.720   1st Qu.:3.000   1st Qu.: 5.000   Not Ideal:271
##  Median :1.020   Median :4.000   Median : 6.000
##  Mean   :1.041   Mean   :4.313   Mean   : 6.134
##  3rd Qu.:1.210   3rd Qu.:6.000   3rd Qu.: 7.000
##  Max.   :2.480   Max.   :9.000   Max.   :10.000
##
##         channel            store          price
##  Independent: 48   Blue Nile :211   Min.   :  497
##  Internet   :318   Ashford   :107   1st Qu.: 3430
##  Mall       : 59   Riddles   : 16   Median : 5476
##                    Fred Meyer: 15   Mean   : 6356
##                    Kay       : 14   3rd Qu.: 7792
##                    University: 13   Max.   :27575
##                    (Other)   : 49
```

Missing Data

The summary() function above confirms that there are no missing observations in the diamonds data.

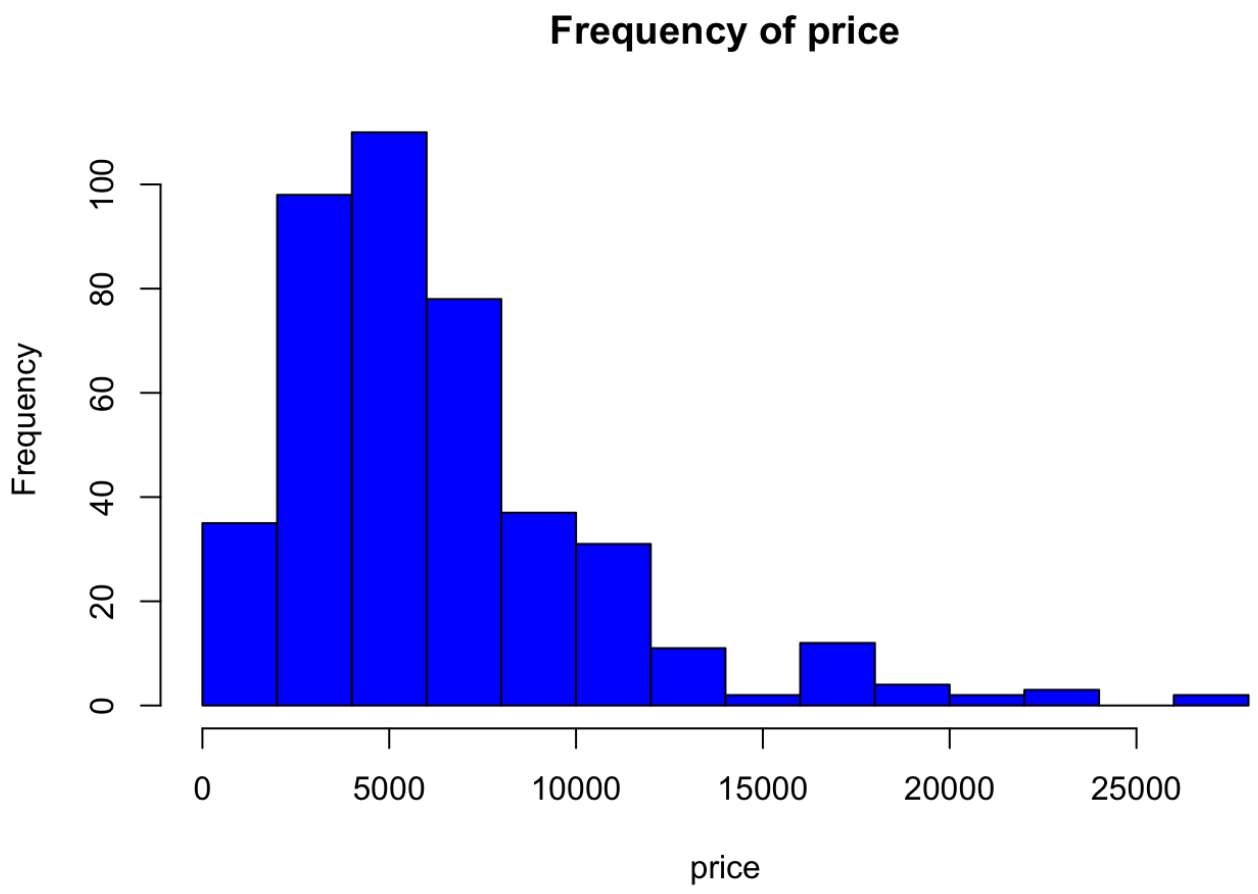
Data Ranges and Distributions

The summary() function output above also provides insight into the distribution of carat, color, and clarity. Each of these variables appear to be approximately normally distributed. In summary() above we see that the mean and median values are close to each other indicating an evenly dispersed data set. The ‘price’ variable shows signs of a positive skew since the median price of $5,476 is less that the mean price of

$6,356. Given that our data set contains the price of diamonds this may not be surprising, but nevertheless one of importance for data modeling since we may need to take the natural log of price to improve our model's predictive accuracy.

Outliers

When viewing the 'price' variable distribution we can see that there are also some outliers in the far-right tail. Above $15,000 the data does appear to show some outliers. The one observations in excess of $25,000 appears to be a clear outliers, however there is nothing to suggest that it is not a valid price. Again, a log transformation may be useful to reign in these higher priced diamonds.

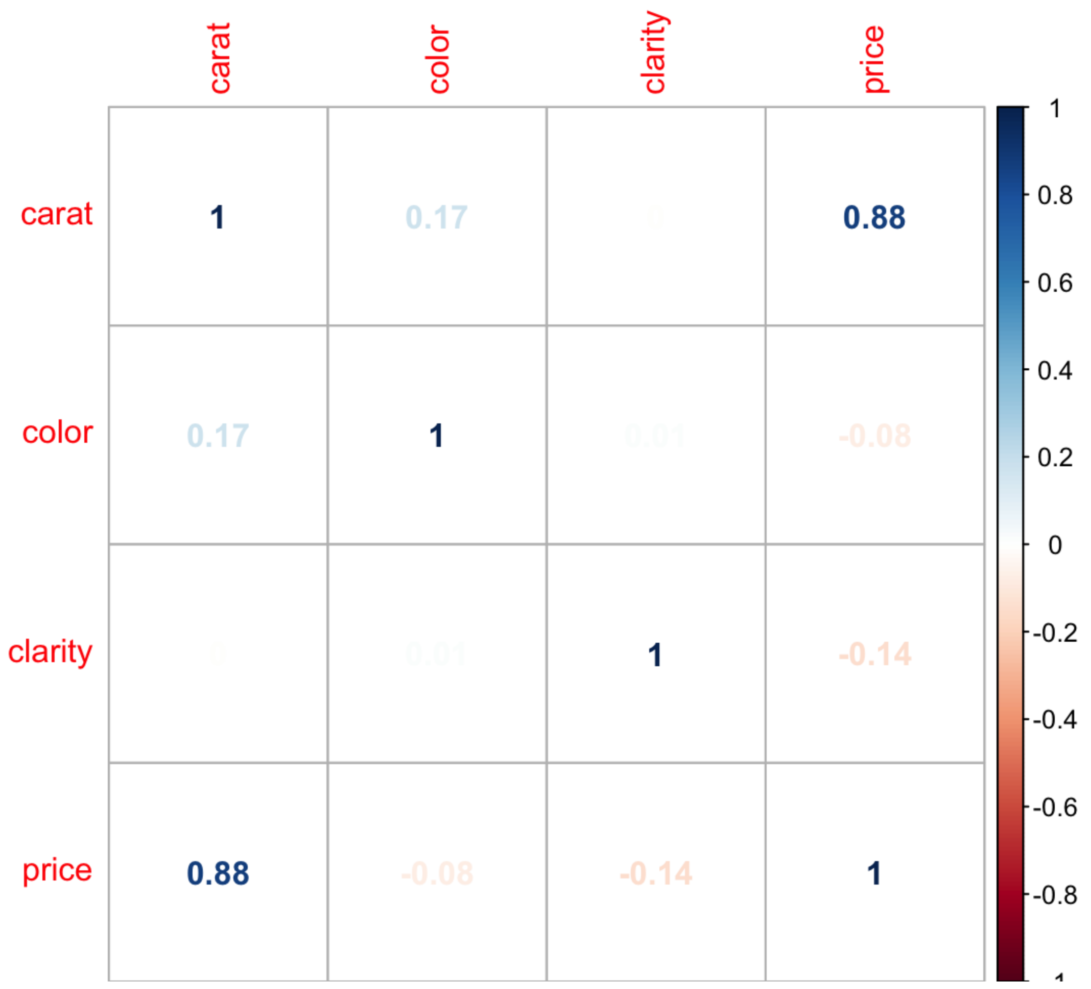


(2) Exploratory Data Analysis:

Next I begin to analyze the diamond data and glean information from it. In this next section I present the interesting relationships between the response variable and the predictor variables.

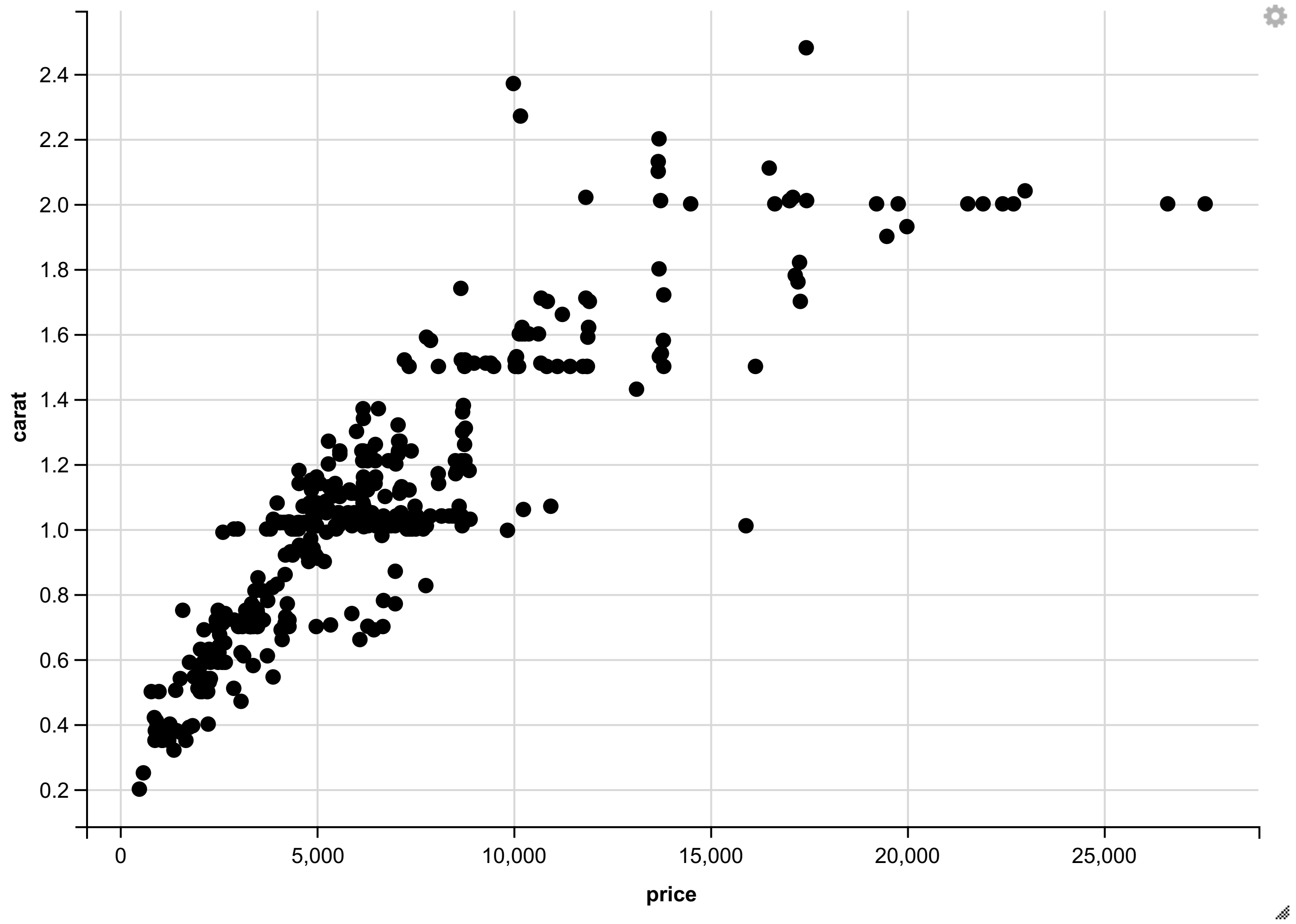
Amongst the numerical variables in the diamonds data set it appears that carat is most closely correlated with price. The correlation coefficient is 0.88 indicating a very high degree of correlation. As carat increases so too does price. The other numerical variables are less correlated with price. Both color and clarity have a correlation coefficient of -0.08 and -0.14 respectively. Of note from the negative correlation coefficients however is that an increase in either color or clarity is likely to lead to a decrease in price. I expect that a linear regression model will confirm that arat is the most predictive numerical variable.

Correlation Between Price, Caret, Color, and Clarity



The positive correlation between caret and price is more easily seen in the scatterplot produced below:

Scatterplot Caret against Price



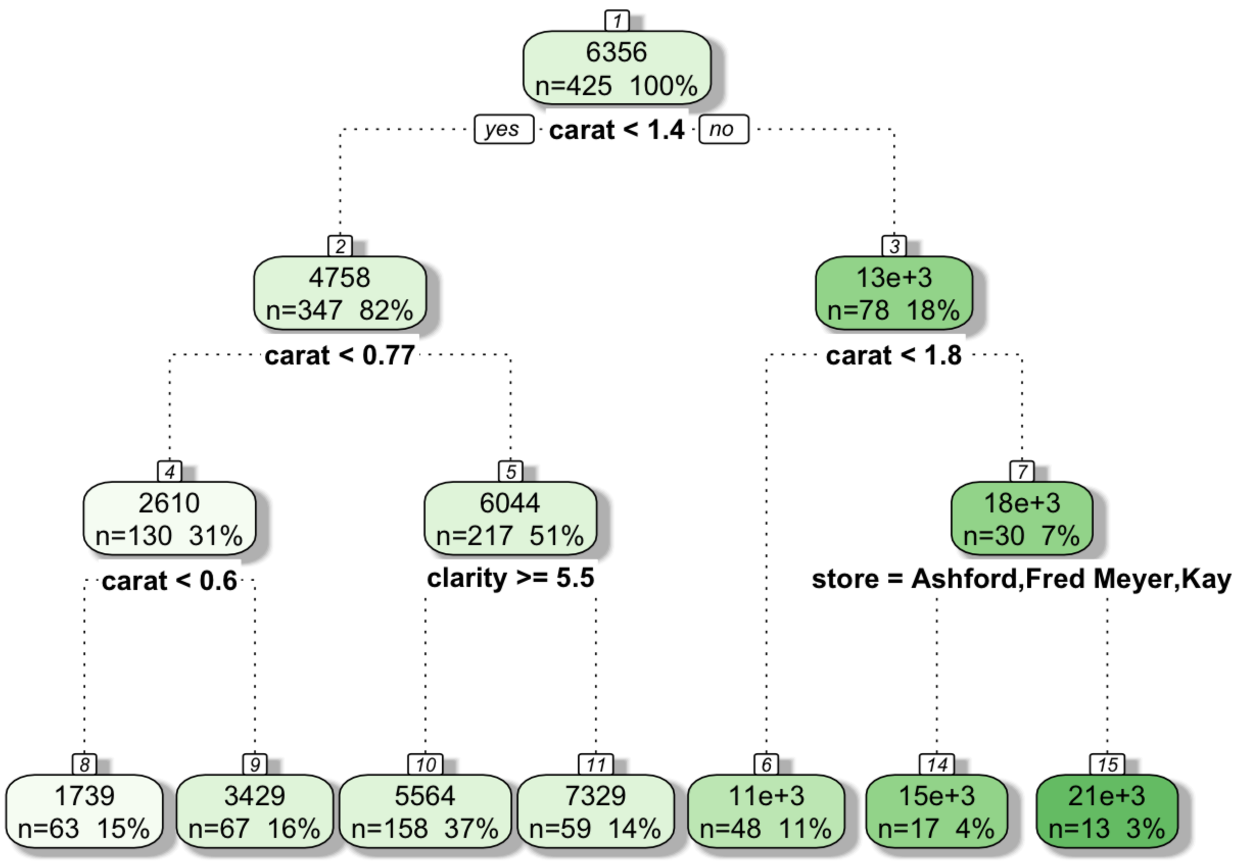
Frequency Plot of Bucketed Price and Store

From the frequency plot below we can see that the majority of observations come from Blue Nile and Ashford. The frequency plot also highlights that two of the stores, Blue Nile and Goodman’s have sole diamonds with a price greater than $20,000. Additionally, only five of the 12 stores have sold a diamond with a price greater thatn $10,000 (Ashford, Blue Nile, Goodmans, Kay, and Zales). This finding is of note because it highlights the fact that a decision tree may be able to split the data into branches that may lead to greater prediction accuracy.

```
##               store
##                Ashford Ausmans Blue Nile Chalmers Danford Fred Meyer
##   1-2000             0       0        22        0       0          0
##   2000-4000         27       0        47        1       2          6
##   4000-6000         18       6        56        5       4          5
##   6000-8000         18       1        39        2       1          2
##   8000-10000         9       0        20        0       2          2
##   10000-12000       17       0        12        0       0          0
##   12000-14000        9       0         2        0       0          0
##   14000-16000        0       0         2        0       0          0
##   16000-18000        9       0         1        0       0          0
##   18000-20000        0       0         4        0       0          0
##   20000-22000        0       0         2        0       0          0
##   22000-24000        0       0         2        0       0          0
##   24000-26000        0       0         0        0       0          0
##   26000-28000        0       0         2        0       0          0
##               store
##                Goodmans Kay R. Holland Riddles University Zales
##   1-2000              0   2          0       8          3     0
##   2000-4000           1   3          1       4          1     5
##   4000-6000           2   2          4       1          6     1
##   6000-8000           5   4          0       3          3     0
##   8000-10000          1   1          2       0          0     0
##   10000-12000         1   0          0       0          0     1
##   12000-14000         0   0          0       0          0     0
##   14000-16000         0   0          0       0          0     0
##   16000-18000         0   2          0       0          0     0
##   18000-20000         0   0          0       0          0     0
##   20000-22000         0   0          0       0          0     0
##   22000-24000         1   0          0       0          0     0
##   24000-26000         0   0          0       0          0     0
##   26000-28000         0   0          0       0          0     0
```

Tree Plot

I use the tree below to extract further information from the diamonds data set. As expected the carat variable is the most predictive. From the tree output below we can see that carat values of 0.77, 1.4, and 1.8 are significant in that they differentiate diamond prices at the highest two levels of the tree. These split values form the first two levels of the tree so are therefore the most predictive values in differentiating diamond pricing. There is also a split on the clarity variable at 5.5 when carat is greater than 0.77 but less than 1.4, so it appears that clarity may be another important predictor between these carat levels. Store values of Ashford, Fred Meyer, and Kay appear to be associated with high priceds diamonds.



Rattle 2017-Jan-30 23:13:23 stevenfutter

(3) The Model Build

In this problem we use a 70/30 training-testing split of the data. I split the data into the two separate data sets and then use the training data set for all of the model development and the testing data set to evaluate the model's out-of-sample predictive accuracy. Before we do this it is important that we make the necessary data transformations.

Log Transformation of the Response Variable & Break out Categorical Variables into Binary Value Columns

From the EDA section above it was determined that taking the log of price would help the model's predictive accuracy. Let's take another look at our data using forward, backward, stepwise, all subsets, and lasso methods.

From this point forward I use the natural logarithm of price as the response variable. Price has been removed from the diamonds, training, and test data sets. Let's continue to break out the data between a 70-30 train-test split.

Naïve Regression Model

Let's begin the model building process by fitting a naive model that uses the backward selection algorithm from the leaps library. This will give us a sense of which variables are important for predicting the response variable, price. Note that the backward selection algorithm produces the following error: '2 linear dependencies found'. The error indicates that some collinearity may exist in the predictor variables. For now I ignore this error and focus on the variables selected. The naive model produced is a 15 variable model with the following predictors:

1. carat,
2. color,
3. clarity,

4. cutNot Ideal,
5. channelInternet,
6. channelMall,
7. storeAusmans,
8. storeBlue Nile,
9. storeChalmers,
10. storeDanford,
11. storeFred Meyer,
12. storeGoodmans,
13. storeKay,
14. storeR. Holland,
15. storeRiddles

Note that storeUniversity and storeZales were not considered in the backward selection algorithm output.

```
## Subset selection object
## Call: regsubsets.formula(price ~ . - log.price, data = d.train, nvmax = 15,
##     method = "backward")
## 17 Variables  (and intercept)
##                 Forced in Forced out
## carat               FALSE      FALSE
## color               FALSE      FALSE
## clarity             FALSE      FALSE
## cutNot.Ideal        FALSE      FALSE
## channelInternet     FALSE      FALSE
## channelMall         FALSE      FALSE
## storeAusmans        FALSE      FALSE
## storeBlue.Nile      FALSE      FALSE
## storeChalmers       FALSE      FALSE
## storeDanford        FALSE      FALSE
## storeFred.Meyer     FALSE      FALSE
## storeGoodmans       FALSE      FALSE
## storeKay            FALSE      FALSE
## storeR..Holland     FALSE      FALSE
## storeRiddles        FALSE      FALSE
## storeUniversity     FALSE      FALSE
## storeZales          FALSE      FALSE
## 1 subsets of each size up to 15
## Selection Algorithm: backward
##           carat color clarity cutNot.Ideal channelInternet channelMall
## 1  ( 1 )  "*"   " "   " "     " "          " "             " "
## 2  ( 1 )  "*"   "*"   " "     " "          " "             " "
## 3  ( 1 )  "*"   "*"   "*"     " "          " "             " "
## 4  ( 1 )  "*"   "*"   "*"     " "          " "             "*"
## 5  ( 1 )  "*"   "*"   "*"     " "          " "             "*"
## 6  ( 1 )  "*"   "*"   "*"     " "          " "             "*"
## 7  ( 1 )  "*"   "*"   "*"     "*"          " "             "*"
## 8  ( 1 )  "*"   "*"   "*"     "*"          "*"             "*"
## 9  ( 1 )  "*"   "*"   "*"     "*"          "*"             "*"
## 10  ( 1 ) "*"   "*"   "*"     "*"          "*"             "*"
## 11  ( 1 ) "*"   "*"   "*"     "*"          "*"             "*"
## 12  ( 1 ) "*"   "*"   "*"     "*"          "*"             "*"
## 13  ( 1 ) "*"   "*"   "*"     "*"          "*"             "*"
## 14  ( 1 ) "*"   "*"   "*"     "*"          "*"             "*"
## 15  ( 1 ) "*"   "*"   "*"     "*"          "*"             "*"
##           storeAusmans storeBlue.Nile storeChalmers storeDanford
## 1  ( 1 )  " "            " "             " "            " "
## 2  ( 1 )  " "            " "             " "            " "
## 3  ( 1 )  " "            " "             " "            " "
## 4  ( 1 )  " "            " "             " "            " "
## 5  ( 1 )  " "            " "             " "            " "
## 6  ( 1 )  " "            " "             " "            " "
## 7  ( 1 )  " "            " "             " "            " "
## 8  ( 1 )  " "            " "             " "            " "
## 9  ( 1 )  " "            "*"             " "            " "
## 10  ( 1 ) " "            "*"             " "            " "
## 11  ( 1 ) " "            "*"             "*"            " "
## 12  ( 1 ) " "            "*"             "*"            " "
## 13  ( 1 ) " "            "*"             "*"            " "
## 14  ( 1 ) "*"            "*"             "*"            " "
## 15  ( 1 ) "*"            "*"             "*"            "*"
##           storeFred.Meyer storeGoodmans storeKay storeR..Holland
## 1  ( 1 )  " "             " "           " "      " "
## 2  ( 1 )  " "             " "           " "      " "
## 3  ( 1 )  " "             " "           " "      " "
## 4  ( 1 )  " "             " "           " "      " "
## 5  ( 1 )  " "             "*"           " "      " "
## 6  ( 1 )  "*"             "*"           " "      " "
## 7  ( 1 )  "*"             "*"           " "      " "
## 8  ( 1 )  "*"             "*"           " "      " "
## 9  ( 1 )  "*"             "*"           " "      " "
## 10  ( 1 ) "*"             "*"           " "      " "
## 11  ( 1 ) "*"             "*"           " "      " "
## 12  ( 1 ) "*"             "*"           " "      "*"
## 13  ( 1 ) "*"             "*"           "*"      "*"
## 14  ( 1 ) "*"             "*"           "*"      "*"
## 15  ( 1 ) "*"             "*"           "*"      "*"
##           storeRiddles storeUniversity storeZales
## 1  ( 1 )  " "          " "             " "
## 2  ( 1 )  " "          " "             " "
## 3  ( 1 )  " "          " "             " "
## 4  ( 1 )  " "          " "             " "
## 5  ( 1 )  " "          " "             " "
## 6  ( 1 )  " "          " "             " "
## 7  ( 1 )  " "          " "             " "
## 8  ( 1 )  " "          " "             " "
## 9  ( 1 )  " "          " "             " "
## 10  ( 1 ) "*"          " "             " "
## 11  ( 1 ) "*"          " "             " "
## 12  ( 1 ) "*"          " "             " "
## 13  ( 1 ) "*"          " "             " "
## 14  ( 1 ) "*"          " "             " "
## 15  ( 1 ) "*"          " "             " "
```

Below I remove the ‘price’ variable from the diamonds, d.train, and d.test data sets to avoid any confusion.

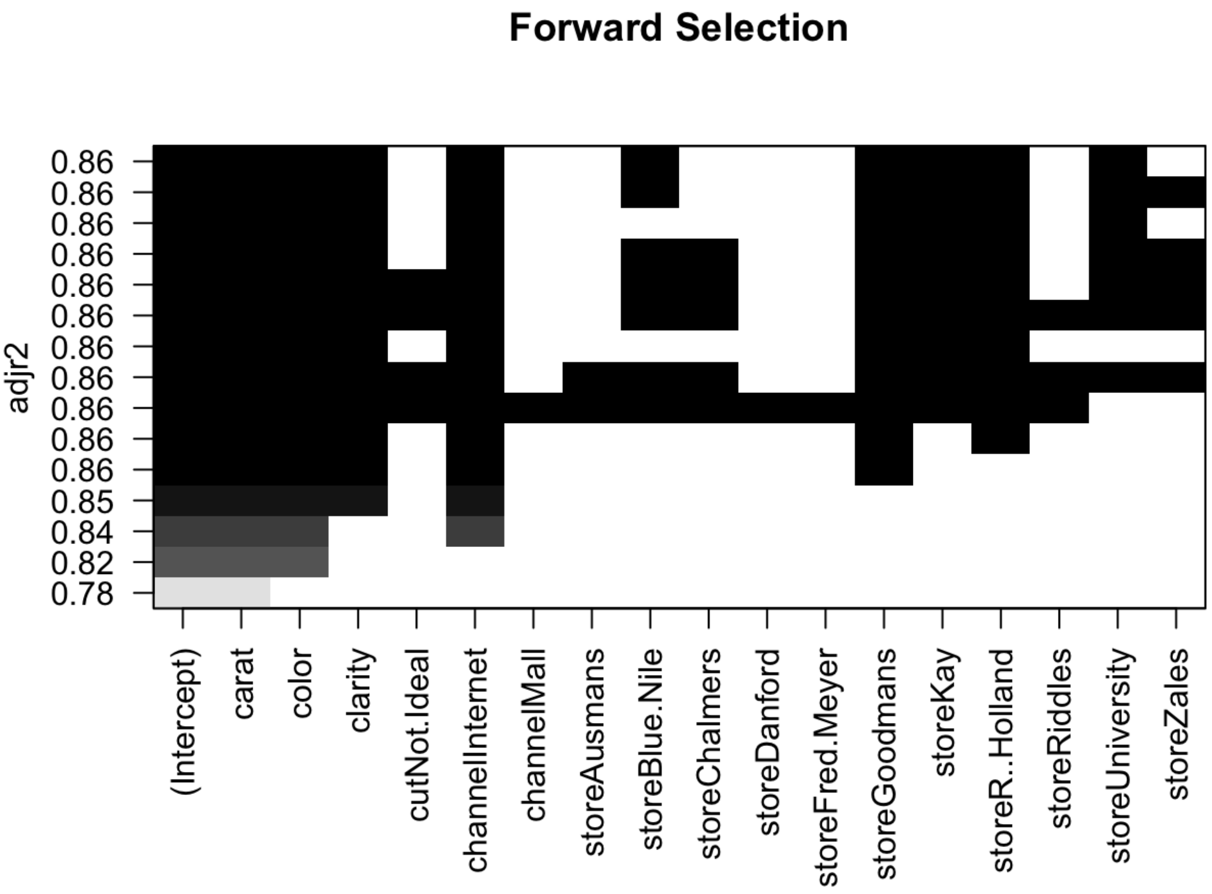
From this point forward I use the natural logarithm of price as the response variable. Price has been removed from the diamonds, training, and test data sets. Let’s take another look at our data using forward, backward, stepwise, all subsets, and lasso methods.

Forward Selection on Log(Price)

Using log(price) as the response the forward variable selection algorithm and evaluating by max adjusted

R^2 produces a 5 variable model, as below. Note that there are models with a higher number of variables but the adjusted R^2 value has a maximum value with only five predictors.

Forward Variable Selection Model: log(price) ~ carat + color + clarity + channelInternet + storeGoodmans

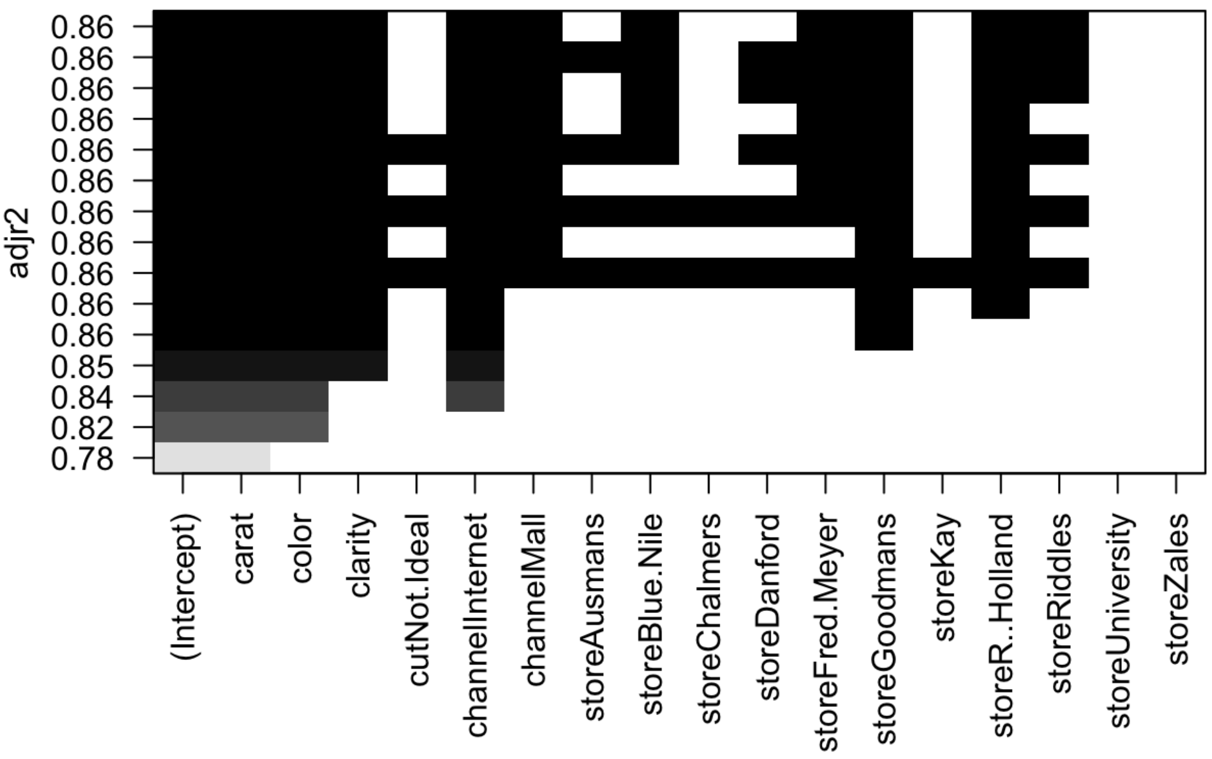


Backward Selection on Log(Price)

The backward variable selection algorithm also has a max adjusted R^2 at five predictor variables. Backward and forward variable selection algorithms return the same variable inputs, producing the model below.

Backward Variable Selection Model: log(price) ~ carat + color + clarity + channelInternet + storeGoodmans

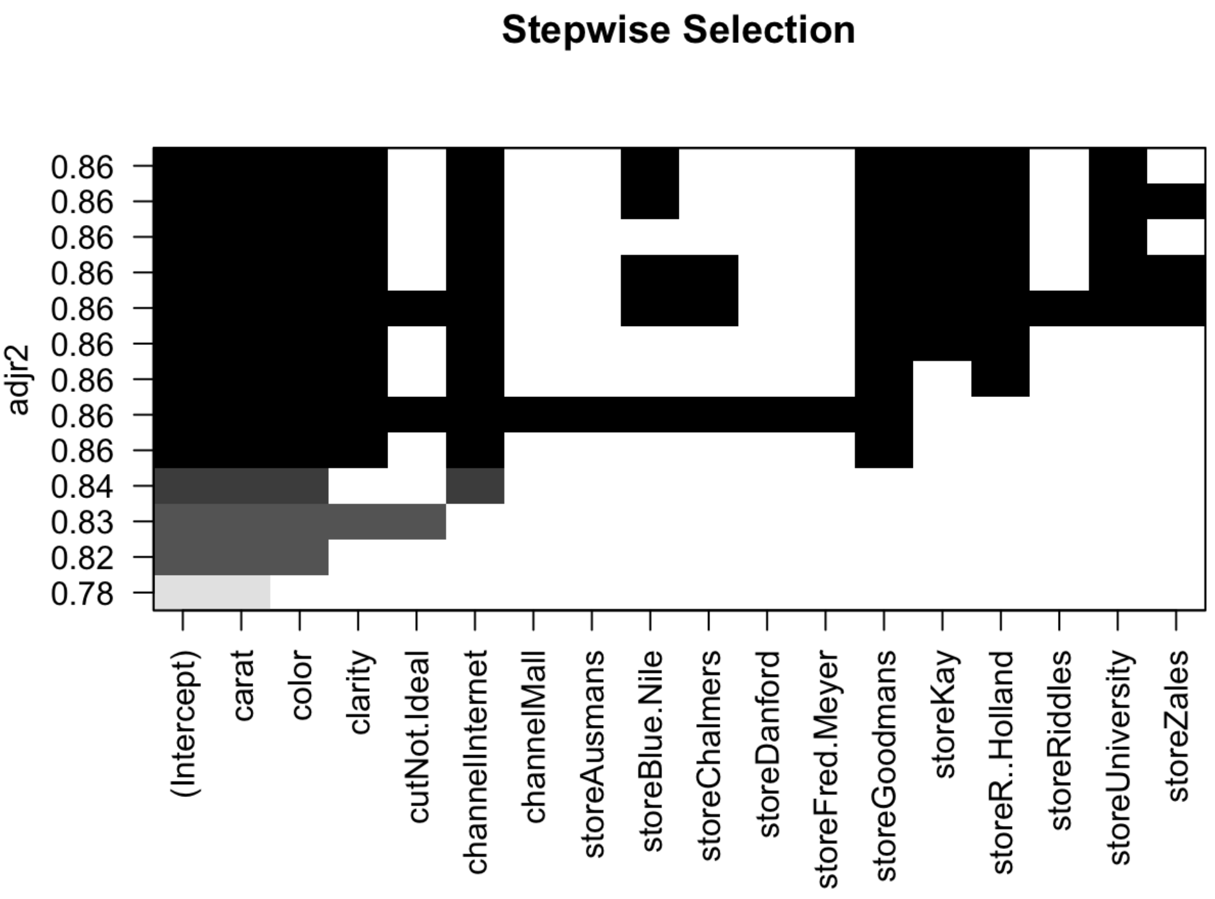
Backward Selection



Stepwise Selection on Log(Price)

The stepwise variable selection algorithm also produces a five variable model with 0.86 adjusted R^2 as its max value. It certainly appears that the Goodmans store stands out as being a good indicator of diamond price.

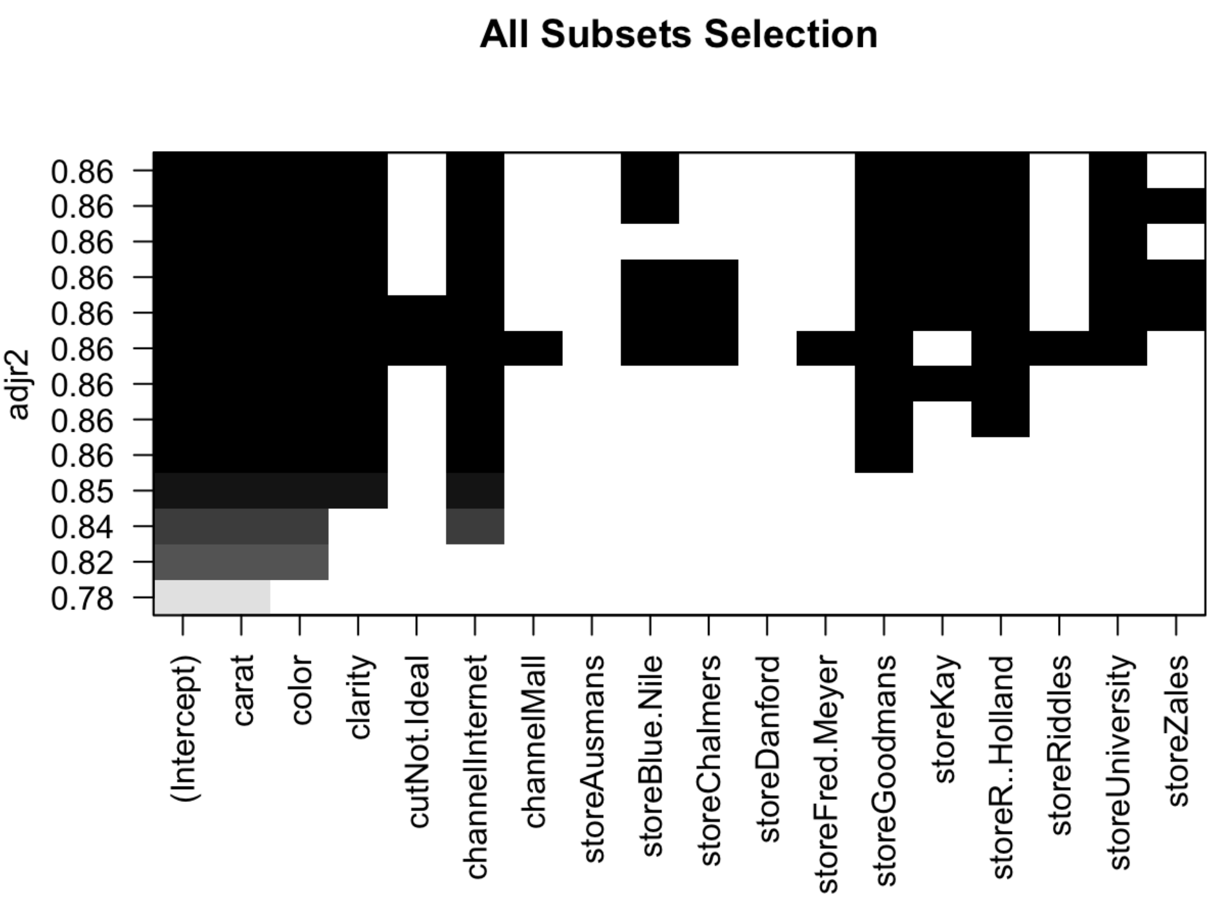
Stepwise Variable Selection Model: log(price) ~ carat + color + clarity + channelInternet + storeGoodmans



All Subsets Selection on Log(Price)

The all subsets variable selection algorithm returns the same five variables as each of the prior algorithms. The max adjusted R^2 occurs at 0.86 with these same five variables:

All Subsets Variable Selection Model: log(price) ~ carat + color + clarity + channelInternet + storeGoodmans



Let’s now use the lasso algorithm to see if that returns any other predictors of interest before we compare the outputs produced.

Lasso Selection on Log(Price)

The lasso selection algorithm produces a 7 variable model, as below. This is two more than the four models above. The additional variables added are storeKay and storeR..Holland.

All Subsets Variable Selection Model: log(price) ~ carat + color + clarity + channelInternet + storeGoodmans + storeKay + storeR..Holland

```
##           carat           color         clarity    cutNot.Ideal 
##      1.51874114     -0.07160654     -0.03810925      0.00000000 
## channelInternet     channelMall    storeAusmans  storeBlue.Nile 
##     -0.11579177      0.00000000      0.00000000      0.00000000 
##   storeChalmers    storeDanford storeFred.Meyer   storeGoodmans 
##      0.00000000      0.00000000      0.00000000      0.27616902 
##        storeKay storeR..Holland    storeRiddles storeUniversity 
##      0.07551717      0.24141412      0.00000000      0.00000000 
##      storeZales 
##      0.00000000
```

In summary, the forward, backward, stepwise, and all subsets models produce the same models when using adjusted R^2 for each model’s quality. The lasso model produces a slightly larger model with storeKay and storeR..Holland as two additional predictor variables. Let’s fit these priority variables into the following models:

Linear Regression Model

1. a linear regression model with no interactions using the lm() function, As can be seen from the output

of the linear regression model below, all input variables except for storeKay are significant at the 0.1% level. Note that storeKay is still significant but at a lower level of significance (5%). Adjusted R^2 = 0.8625.

```
##
## Call:
## lm(formula = log.price ~ carat + color + clarity + channelInternet +
##     storeGoodmans + storeKay + storeR..Holland, data = d.train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.15615 -0.08746  0.05446  0.14813  0.84347
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)      7.750965   0.092255  84.017  < 2e-16 ***
## carat            1.605160   0.037777  42.490  < 2e-16 ***
## color           -0.089856   0.008277 -10.856  < 2e-16 ***
## clarity         -0.062201   0.010041  -6.195 2.00e-09 ***
## channelInternet -0.190803   0.042511  -4.488 1.04e-05 ***
## storeGoodmans    0.379670   0.109235   3.476 0.000588 ***
## storeKay         0.173007   0.080674   2.145 0.032824 *
## storeR..Holland  0.436558   0.131072   3.331 0.000979 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2527 on 289 degrees of freedom
## Multiple R-squared:  0.8658, Adjusted R-squared:  0.8625
## F-statistic: 266.3 on 7 and 289 DF,  p-value: < 2.2e-16
```

Linear Regression Model with Interaction Terms Added

2. a linear regression model including some interaction terms. From the tree plot above in the EDA section we saw that carat and clarity appear to have some interaction in determining price. Let’s add this interaction term to the model as well as some other interactions to see how the model performs. In the model output below the added interaction terms are all significant. However, the newly added terms appear to account for the variation of the non-interactive terms: carat and color. These variables are no longer significant. Adjusted R^2 = 0.8789 (which is slightly higher than the linear regression model without interaction terms added).

```
##
## Call:
## lm(formula = log.price ~ carat + color + clarity + channelInternet +
##     storeGoodmans + storeKay + storeR..Holland + carat:color +
##     carat:clarity + carat:cutNot.Ideal, data = d.train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.04762 -0.07586  0.04933  0.14294  0.78336
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)         6.942246   0.170070  40.820  < 2e-16 ***
## carat               2.460314   0.149868  16.417  < 2e-16 ***
## color              -0.006799   0.020689  -0.329 0.742677
## clarity             0.011708   0.022555   0.519 0.604112
## channelInternet    -0.192096   0.041236  -4.658 4.89e-06 ***
## storeGoodmans       0.390674   0.103452   3.776 0.000194 ***
## storeKay            0.170258   0.075844   2.245 0.025543 *
## storeR..Holland     0.347422   0.124305   2.795 0.005542 **
## carat:color        -0.077438   0.018008  -4.300 2.34e-05 ***
## carat:clarity      -0.075141   0.020819  -3.609 0.000362 ***
## carat:cutNot.Ideal -0.072458   0.028491  -2.543 0.011510 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2372 on 286 degrees of freedom
## Multiple R-squared:  0.883,  Adjusted R-squared:  0.8789
## F-statistic: 215.8 on 10 and 286 DF,  p-value: < 2.2e-16
```

Tree Model

3. The output of the tree model is interesting. Note that the output of summary() indicates that only two of the 7 variables have been used in constructing the tree: carat and clarity. In the context of a regression tree, the deviance (Residual mean deviance = 0.05302) is simply the sum of squared error for the tree. Lets plot it:

```
##
## Regression tree:
## tree(formula = log.price ~ carat + color + clarity + channelInternet +
##     storeGoodmans + storeKay + storeR..Holland, data = d.train)
## Variables actually used in tree construction:
## [1] "carat"   "clarity"
## Number of terminal nodes:  8
## Residual mean deviance:  0.05302 = 15.32 / 289
## Distribution of residuals:
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -0.756000 -0.141500  0.003314  0.000000  0.131000  0.796800
```

Random Forest

4. The Random Forest model below is produced by creating 500 tree and trying 5 variables at each split.

```
##
## Call:
##  randomForest(formula = log.price ~ carat + color + clarity +      channelInter
net + storeGoodmans + storeKay + storeR..Holland,      data = d.train, mtry = 5, im
portance = TRUE)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 5
##
##           Mean of squared residuals: 0.0281323
##                     % Var explained: 93.92
```

(4) Model Comparison

In this next section I complare model performance using the mean squared error to evaluate each model's performance based upon both the training and test data sets.

MSE on the linear regression model without interaction terms added is 6.21% in-sample and 6.14% out-of-sample.

MSE on the linear regression model with the interaction terms added is 5.42% in-sample and 5.72% out-of-sample.

MSE on the tree regression model is 5.16% in-sample and 8.05% out-of-sample.

MSE on the random forest model is 0.63% in-sample and 4.57% out-of-sample.

Model MSE Results

```
##                              model   train.mse   test.mse
## 1                Linear Regression 0.062125673 0.06142574
## 2 Linear Regression w Interactions 0.054160813 0.05719870
## 3                             Tree 0.051587320 0.08054636
## 4                    Random Forest 0.006340628 0.04570370
```

In summary, the best performing model is the random forest model with an out-of-sample MSE of 4.57%.

(5) APPENDIX for R Code

```
# insert libraries that are used at beginning
library(dplyr)
library(ggvis)
library('lattice')

### Read Data from csv file
inPath = file.path("~/Dropbox","NU","ADVANCED_MODELING")
diamonds = read.csv(file.path(inPath,"two_months_salary.csv"),na.strings=c("NA"," "
))
attach(diamonds)


##############################
#### Quality Check Section ###
##############################
str(diamonds)
summary(diamonds)

# Outlier Check
```

```r
hist(price,col="blue",main='Frequency of price', xlab='price')

# Correlation Evaluation
library(corrplot)
M <- cor(diamonds[,sapply(diamonds, is.numeric)])
corrplot(M, method="number")

# Scatterplot Caret vs Price
diamonds %>% ggvis(x = ~price, y = ~carat) %>%
  layer_points()

# Frequency Plot Bucketed Price by Store front
diamonds$priceCut<-cut(price, seq(1,30000,2000), labels=c('1-2000','2000-4000','400
0-6000','6000-8000','8000-10000','10000-12000','12000-14000',
                                                              '14000-16000','16000-18
000','18000-20000','20000-22000','22000-24000','24000-26000',
                                                              '26000-28000'))
table(diamonds$priceCut,store)

# Tree Plot
require(rpart)
require(rattle)
require(rpart.plot)
diamonds = diamonds[,-8]  # remove priceCut variable as this will throw off the res
ults

form <- as.formula(price ~ .)     # Plot a more reasonable tree
tree <- rpart(form,diamonds)            # A more reasonable tree
fancyRpartPlot(tree)                     # A fancy plot from rattle




################################
#### The Model Build Section ###
################################

#### Data Transformation Steps
diamonds$log.price = log(price)
attach(diamonds) # making update to attached $ expanded cols now added log.price to
diamonds data.frame

# I use a trick here to break out the categorical variables into binary 1/0 values
diamondsMatrix = data.frame(model.matrix(log.price~., diamonds))

# add back the log.price variable
diamondsMatrix = cbind(log.price,diamondsMatrix)
diamonds = diamondsMatrix

# let's also remove price so that there is no confusion as to which response variab
le should be used
dropIdx = which(names(diamonds) %in% c("X.Intercept."))
diamonds = diamonds[,-dropIdx]

# Create train and test data sets using 70-30 split
smp.size = floor(0.7 * nrow(diamonds))
set.seed(1)
```

```
train = sample(seq_len(nrow(diamonds)), size = smp.size)
test = -train
d.train = diamonds[train,]
d.test = diamonds[-train,]

# Run naive model using the leaps libray
library(leaps)
regfit.bwd = regsubsets(price~.-log.price, data=d.train, nvmax = 15, method='backw
ard')
summary(regfit.bwd)

# let's also remove price so that there is no confusion as to which response variab
le should be used
dropIdx2 = which(names(diamonds) %in% c("price"))
diamonds = diamonds[,-dropIdx2]
d.train = d.train[,-dropIdx2]
d.test = d.test[,-dropIdx2]

##### Forward Selection on Log(Price)
regfit.fwd.log = regsubsets(log.price~., data=d.train, nvmax=17, method='forward',
nbest=1)
plot(regfit.fwd.log, scale = "adjr2", main = "Forward Selection")  # adjusted R2

##### Backward Selection on Log(Price)
regfit.bwd.log = regsubsets(log.price~., data=d.train, nvmax = 17, method='backwar
d',nbest=1)
plot(regfit.bwd.log, scale = "adjr2", main = "Backward Selection")  # adjusted R2

##### Stepwise Selection on Log(Price)
regfit.stepwise.log = regsubsets(log.price~., data=d.train, nvmax = 13, method='se
qrep', nbest=1)
plot(regfit.stepwise.log, scale = "adjr2", main = "Stepwise Selection")  # adjusted
R2

##### All Subsets Selection on Log(Price)
regfit.all.subsets.log = regsubsets(log.price~., data=d.train, nvmax = 13, method=
'exhaustive', nbest=1)
plot(regfit.all.subsets.log, scale = "adjr2", main = "All Subsets Selection")  # ad
justed R2

##### Lasso Selection on Log(Price)
library(lars)
y.train = d.train[,1]
x.train = d.train[,-1]
cv <- cv.lars(as.matrix(x.train), y.train, plot.it = FALSE, mode = "step")  # Use
cross validation to minimize error
idx <- which.max(cv$cv - cv$cv.error <= min(cv$cv))
coef(lars(as.matrix(x.train), y.train))[idx,]

##### Linear Regression Model
lm.fit = lm(log.price~ carat + color + clarity + channelInternet + storeGoodmans +
storeKay + storeR..Holland,data=d.train)
summary(lm.fit)
formula(lm.fit)

##### Linear Regression Model with Interaction Terms Added
```

```
lm.interact.fit = lm(log.price ~ carat + color + clarity + channelInternet + store
Goodmans + storeKay + storeR..Holland + carat:color + carat:clarity + carat:cutNot
.Ideal,data=d.train)

##### Tree Model
library(tree)
set.seed(1)
tree.fit=tree(log.price~carat + color + clarity + channelInternet + storeGoodmans +
storeKay + storeR..Holland,data=d.train)
summary(tree.fit)

##### Random Forest
library(randomForest)
set.seed(1)
randomForest.fit = randomForest(log.price~carat + color + clarity + channelInterne
t + storeGoodmans + storeKay + storeR..Holland, data=d.train, mtry=5, importance=T
RUE)
randomForest.fit


####################################
#### The Model Selection Section ###
####################################
yhat.lm.train = predict(lm.fit, newdata=d.train)
lm.mse.train = mean((yhat.lm.train-d.train$log.price)^2)  # MSE = 0.06212567 (in-s
ample)
yhat.lm = predict(lm.fit, newdata=d.test)
lm.mse.test = mean((yhat.lm-d.test$log.price)^2)         # MSE = 0.06142574 (out-o
f-sample)

yhat.lm.interact.train = predict(lm.interact.fit, newdata=d.train)
lm.interact.mse.train = mean((yhat.lm.interact.train-d.train$log.price)^2)  # MSE
= 0.05416081 (in-sample)
yhat.lm.interact = predict(lm.interact.fit, newdata=d.test)
lm.interact.mse.test = mean((yhat.lm.interact-d.test$log.price)^2)         # MSE =
0.0571987 (out-of-sample)

yhat.tree.train=predict(tree.fit,newdata=d.train)
tree.mse.train = mean((yhat.tree.train-d.train$log.price)^2)  # MSE = 0.05158732 (i
n-sample)
yhat.tree=predict(tree.fit,newdata=d.test)
tree.mse.test = mean((yhat.tree-d.test$log.price)^2)         # MSE = 0.08054636 (o
ut-of-sample)

yhat.randomForest.train = predict(randomForest.fit, newdata = d.train)
randomForest.mse.train = mean((yhat.randomForest.train-d.train$log.price)^2)  # MSE
= 0.006340628 (in-sample)
yhat.randomForest = predict(randomForest.fit, newdata = d.test)
randomForest.mse.test = mean((yhat.randomForest-d.test$log.price)^2)         # MSE
= 0.0457037 (out-of-sample)

##### Model MSE Results
model = c('Linear Regression', 'Linear Regression w Interactions', 'Tree', 'Random
Forest')
train.mse = c(lm.mse.train, lm.interact.mse.train, tree.mse.train, randomForest.mse
.train)
test.mse  = c(lm.mse.test, lm.interact.mse.test, tree.mse.test, randomForest.mse.te
st)
results = data.frame(model, train.mse, test.mse)
results
```

Loading [Contrib]/a11y/accessibility-menu.js