Derek Hughes
Assignment #2
Predict 454 – Sec 55
Winter 2016

Introduction

As most men will attest who have been confronted with the arduous task of selecting the right diamond for their soon-to-be bride, wading through the various characteristics of what defines a quality diamond can be a daunting and less-than-enjoyable experience. Yet, at the same time, they want to ensure they are getting a fair price. This study aims to discover which characteristics, jewelers, and locations will best predict the price of a diamond, so as to remove some of the drama from the newly engaged man.

The problem is approached by collecting data from stores, where the stores are located, and various characteristics of diamonds such as size, color, clarity, cut, and price. By using traditional statistical methods to find correlations between these characteristics to build models that predict a diamond's price, the data reveal interesting findings to aid the apprehensive buyer.

A few of these interesting findings are listed as follows: the size of the diamond is the single most important indicator to the price of the diamond.; the prices at some stores are significantly different than others; the pricing you receive whether you buy from a mall located jeweler or a jeweler on the Internet can be significantly different; and often only knowing a just a few of these characteristics can predict the price as well or better than knowing a plethora of characteristics to determine a diamond's price.

Modeling Problem

The problem for this study is to develop a model that best predicts the price of a diamond. Along the way we will learn interesting relationships between the variables and between the variables and the price. These inferences are used to create a larger understanding of the data and improve the value of the study, but the main goal is to develop a predictive model that predicts the price of a diamond based on its characteristics and the environment from which it is located. Hence, this is a regression modeling problem.

The data available consist of continuous and categorical variables representing various characteristics about a diamond and the diamond buying process. The categorical data are both nominal and ordinal. The nominal categorical data represent the type of cut used on the diamond (ideal or not ideal), the location of the jeweler (mall, independent, Internet), and the name of the jeweler. The ordinal categorical data are based on a rating scale for color and clarity devised by the Gemological Institute of America (GIA). The numeric data is based on the size of the diamond in carats and the price in US dollars ($).

The techniques we use in the study are implemented in a systematic fashion in order to initially understand the data and then, based on those findings, let the data guide us to developing the most predictive model.

Initially, we check the quality of the data and clean it if necessary. This initial step includes investigating the structure of the data, if we have missing values and if they should be imputed, if the values in the data are valid, if the range of values is appropriate, the presence of any outliers, the distributions of the data within each variable, etc.

Second, we apply traditional EDA to look for relationships in the data based on this being a regression problem. We look for correlations, relationships between variables, and interactions. We also run a naïve decision tree to guide our EDA investigation.

For creating models, we use the training paradigm to determine goodness of fit of our models. We do this by dividing the data into a training and test set based on a 70% to 30% division, respectively. Each model is fitted to the training data set and built based on the training set. The performance of the model is based on the MSE of the difference between the predicted price and the actual price of the training set. Finally, the model is fitted to the test data set and the MSE performance is determined and used as the ultimate indicator of model performance for comparing between models.

When we are ready to build our models, for easier interpretation of the results, we manually transform each level of every factor variable into a dummy variable. We then apply the variable selection techniques of backward selection, forward selection, stepwise selection, best subset selection, and Lasso. We do this to identify important variables and compare and contrast the performance of the models. Finally, we look at which variables are consistently identified within each of these variable selection models as significant and consider combining those variables when manually building other models.

Finally, we build three different models using linear regression, decision tree, and Random Forest algorithms. Each model is initially built with a simple, single variable for comparison purposes. Next each model is built using log transformations of the response variable (price) and the size variable (carat). Interactions between variables and various other combinations of variables are then added to each model. After each new subset of variables is applied to each model, its MSE performance on the in-sample training set and the out-sample test set is evaluated. Each model's MSE values on the test set are used as the final performance metric for comparing between models.

Data Quality Check

The data have continuous and categorical variables and is made up 425 observations and 7 variables.

VAR.NAME	VAR.CLASS	# LEVELS	DESCRIPTION
carat	numeric	0	weight of diamond
color	ordinal factor	10	clarity of diamond
clarity	ordinal factor	11	color of diamond
cut	factor	2	type of cut
channel	factor	3	jewler location
store	factor	12	name of jewler
price	integer	0	cost in US dollars

Table 1 – Data Description

The numeric data is comprised of two variables: price and size. There are not any missing values and all values appear valid, as does the range of values for each variable.

Based on values outside of either the 1st or 99th percentile, carat has eight outliers and price has ten outliers. By comparing the min and max values of each variable to these percentiles, and because they only account for approximately 2% of the total observations, we can reasonably presume these outliers are not having a significant effect on the overall metrics of either variable.

VAR.NAME	NOT.NA	IS.NA	PCT.NA	MIN	Q01	Q25	Q50	Q75	Q99	MAX	MEAN	SD	OUT
carat	425	0	0	0.2	0.35	0.72	1.02	1.21	2.125	2.48	1.041	0.422	8
price	425	0	0	497	895.16	3430	5476	7792	22311.48	27575	6355.99	4404.24	10

Table 2 – Numeric Data Descriptive Statistics

The distribution of our response variable, price, is right skewed. Since this is our response variable we apply a log transformation and can see it makes a more normal distribution. Carat, while less right skewed than price, is also considered for a log transformation.

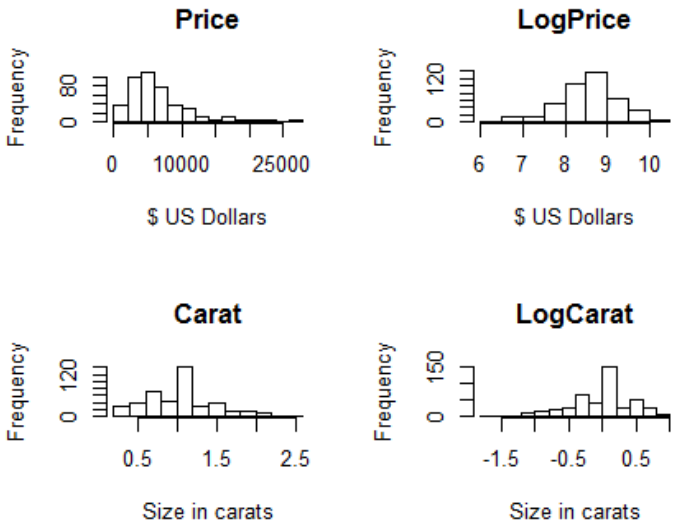

Figure 1 – Log Transformations (Price, Carat)

Our categorical variables (color, clarity, cut, channel, store) distributions have interesting traits as well. The levels "M" for color and "FL" and "I3" for clarity never appear in the data. These represent the extreme minimum or maximum levels in each variable. For store, Blue Nile and Ashford represent almost 75% of the store values and, since both are Internet stores, the Internet value in the channel variable is almost 75% of those values as well. This could indicate some inherent bias in the data.

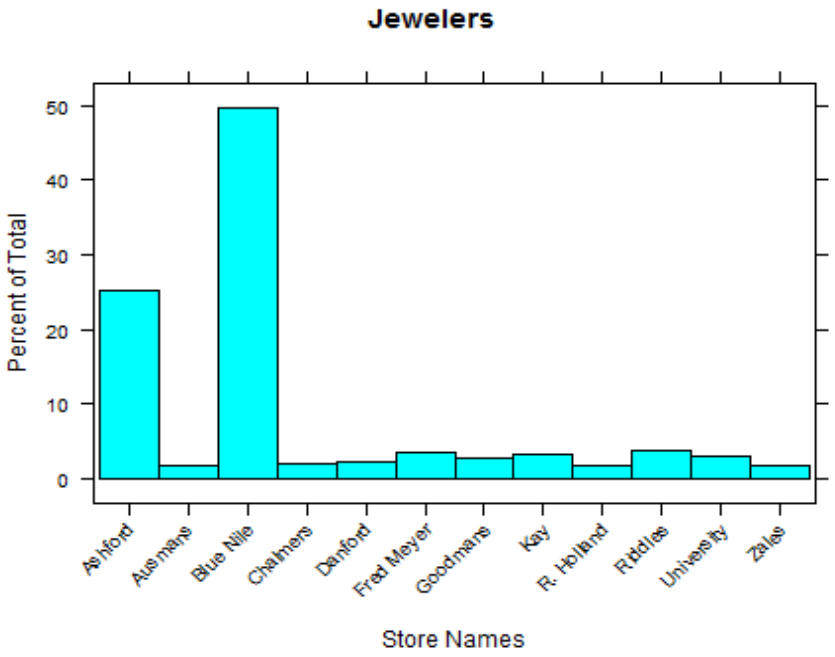

Figure 2 – Jewelers by Percentage

Exploratory Data Analysis

Exploratory data analysis is the process of looking for relationships in the data driven by the type of problem. In our case, we look for relationships related to the response variable, price.

In terms of correlation, carat has the strongest correlation with price at .8796 – a strong correlation. Furthermore, by log transforming price and carat their correlation increases to a very strong correlation of **0.9288** (logprice, logcarat).

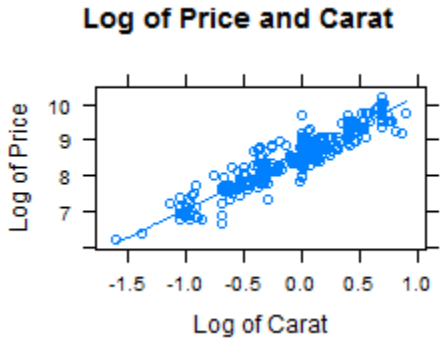

Figure 3 – Log Price/LogCarat Correlation

We can find categorical relationships between logprice and levels in each categorical variable. We find this by first visually inspecting boxplots of each categorical variable on price (Figure 4).

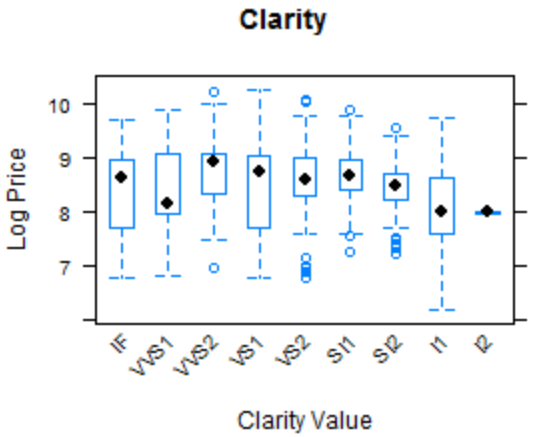

Figure 4 – Clarity Means of LogPrice

If the mean looks significantly different we run an ANOVA on logprice to determine if any factor level means are significantly different (Table 3). If the ANOVA is significant we run pairwise T-test between each factor level to identify which show significant differences in regard to logprice (Table 4).

	Df	Sum Sq	Mean Sq	F value	Pr(>F)	
clarity	8	12.73	1.5910	3.388	0.000881	***
Residuals	416	195.36	0.4696			

Table 3 – ANOVA Results for Clarity on LogPrice

The ANOVA results show significant mean differences between the levels in Clarity.

	IF	VVS1	VVS2	VS1	VS2	SI1	SI2	I1
VVS1	1.00000	-	-	-	-	-	-	-
VVS2	1.00000	1.00000	-	-	-	-	-	-
VS1	1.00000	1.00000	0.18603	-	-	-	-	-
VS2	1.00000	1.00000	1.00000	1.00000	-	-	-	-
SI1	1.00000	1.00000	1.00000	1.00000	1.00000	-	-	-
SI2	1.00000	1.00000	0.33137	1.00000	1.00000	1.00000	-	-
I1	1.00000	1.00000	0.00056	0.37010	0.01241	0.00615	0.27360	-
I2	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000

Table 4 – Pairwise T-test of Levels of Clarity

The pairwise T-tests show there are significant logprice differences between these levels in each factor.

Interactions are when the value of one variable "depends on" the value of another. We find interactions by initially using a graph grouping two categorical variables on our response variable, logprice. If an interaction appears (often indicated by diverging lines of a factor level over the levels of another factor's levels), we can compare the means of the factor levels with a t-test for significant differences. If the differences are significant then we can identify that interaction as one to test in our model to improve the model's performance.

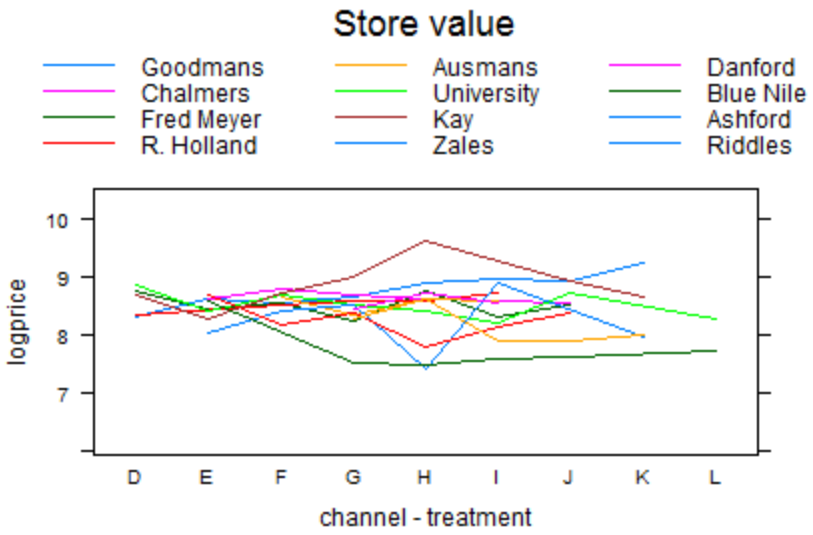

Figure 5 – Interactions of Channel and Store on LogPrice

In Figure 5, we see there may be interactions between both Blue Nile and Ashford and the color levels G, H, and I. Investigation of other interactions show Blue Nile and Ashford and the clarity level VS2, and Blue Nile and Ashford and the channel level Internet (Table 5) may be interactions as well.

```
Welch Two Sample t-test
data:  internet.bluenile$logprice and internet.ashford$logprice
t = -3.5262, df = 237.972, p-value = 0.0005057
```

Table 5- T-test for Means Difference Between Levels of Store (Internet) and Levels of Channel (Blue Nile and Ashford) on LogPrice

Finally we run a naïve decision tree on logprice using all variables to see what variables are selected as significant. Not surprisingly considering the correlation we discovered earlier, it uses carat for all of the splits and has seven terminal nodes.

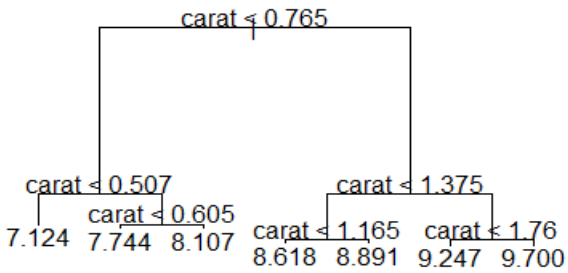

Figure 6 – Tree Diagram Using Logprice and All Variables

Modeling

Variable selection techniques

Model	# Variables	Formula	Train MSE	Train Error	Test MSE	Test Error
Variable Selected Models						
bkwd selection	28	price~	1607498	1267.87	3851158	1962.43
bkwd selection	10	logprice~.	5453486	2335.27	4813277	2193.92
fwd selection	10	logprice~.	5681092	2383.5	6946762	2635.67
stepwise selection	18	logprice~.	5975342	2444.451	3568662	1889.09
best subset selection	21	logprice~.	5626378	2371.999	5273303	2296.37
LASSO	11	logprice~.	4897930	2213.12	5994621	2448.39

Table 6 – Models Built by Variable Selection Techniques

First, naïve backward selection model is built on the response variable price using all variables. Our test mean test error is $1,962, which as we will see is pretty good, but this model uses 28 out of 37 variables, which doesn't make it very practical. The distribution of price is right skewed so we log transform it to "logprice" and create a more normal distribution to use in the model. This increases our test error to $2,194, but we are using almost a third of the variables (10) as before, making it much more practical. We try forward selection using all variables on logprice but the results are the worst of any variable selection method as error increases to $2,636 using 10 variables.

Stepwise is the best variable selection technique of the five techniques tested. It did select more variables than either backward, forward, or LASSO models but its average error was $1889, some ~$70 better than our naïve backward selection model using price as the response variable that used 28 variables.

Best subset selection, which uses an exhaustive search algorithm, selected the second most variables (21) and was about middle of the pack in terms of test error ($2,296). While LASSO produced its lowest test error using about the lowest number of variables (11), its test error was the second largest at $2,448.

Overall, from a practical standpoint, it is often times difficult to justify using almost twice as many variables (18 to 10) to obtain lower prediction error. However, in this case, it is worth it because the prediction error is nearly $300 lower than average test error from the model with only 10 variables. Furthermore, the goal of the study is to produce the model with the lowest average prediction error. Thus, of the variable selection models, stepwise selection method is the best.

Models

Linear model

We run four types of linear regression models to determine the effect of transformations and interactions. The first model regresses carat on price. Here there are no variable transformations and we achieve an average test error of $2,585. Next, still using carat as the independent variable, we transform the response variable price to logprice to create a more normally distributed response variable. Unfortunately, this actually increases our test error ~$150. However, taking from our EDA that the highest correlation was found by log transforming both price and carat, we log transform carat as well and now our average test error improves to the best we've achieved thus far with linear regression ($2,517).

To complete our linear regression model build, we apply the potential interactions we discovered in our EDA. Individually we apply each interaction and keep it in the model if the overall average model error improves. As such, we learn that the interactions between Blue Nile and Internet, Ashford and Internet, Blue Nile and color G, and Ashford and color I do not have an effect on the model. However, when applying the interactions Blue Nile and VS2, Ashford and VS2, Blue Nile and H, Blue Nile and I, Ashford and G, and Ashford and H we see a large improvement in our average test error from $2,517 down to $2,204. Clearly the addition of interactions has an important effect in the model.

Decision Tree

A decision tree is used with all the variables and fitted using price as the response and logprice as the response. Interestingly, when price is the response it achieves the lowest average test error of any model at $1,860. The next closest model is the less than practical backward selection model which uses 28 variables. This particular decision tree only uses the variable carat, ideal, and Blue Nile to predict the price of the diamond.

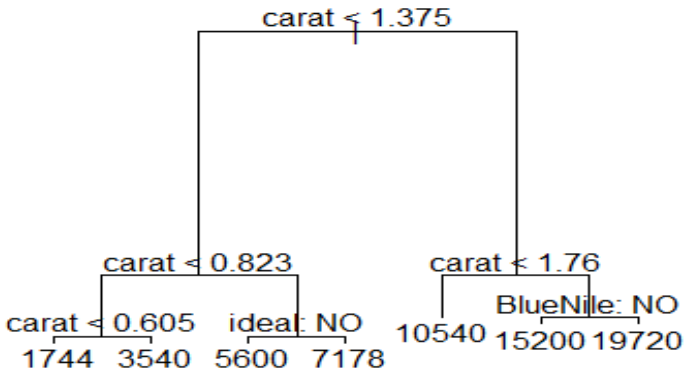

Figure 7 - Decision Tree Variables

All remaining decision trees that use logprice as the response raise the average test error. Interestingly, when we add more variables to the decision tree model the test error remains the same. This most likely due to the very high correlation of carat with the response price or logprice and forces the decision tree to continue to use it as the best split initially, which propagates through the remaining branches to reach essentially the same predictive accuracy regardless of what other variables are used in the model.

Random Forest

Random forest is run on all variables using price and logprice as the response variables. Again, similar to decision tree, the effects of log transforming the response variable and/or the log transformation of the variable carat seem to have more deleterious effects on the test error. This most likely shows how trees are less influenced by the assumptions of normal distribution of variables than more traditional and statistically reliant models such as linear regression. However, regardless of which random forest model we observe, the most important variable is always carat or logcarat.

Random forest and decision trees cannot handle interaction variables. But considering how well our interaction variables improve the linear regression model, for our final random forest model, we create a model that only includes the variables involved in those interactions. This dramatically improves our average test error by over $600 ($2,002).

Model	# Variables	Formula	Train MSE	Train Error	Test MSE	Test Error
Built models						
linear regression	1	price~carat	3537921	1880.94	6686648	2585.85
linear regression	1	logprice~ carat	8433771	2904.096	7506122	2739.73
linear regression	1	logprice~ logcarat	3435912	1853.62	6335009	2516.94
linear regression	14	interactions	3022919	1738.65	4856701	2203.79
decision tree	3	price~.	2501309	1581.553	3458866	**1859.8**
decision tree	1	logprice~.	2884486	1698.38	4982805	2232.22
decision tree	1	logprice~logcarat	2884486	1698.38	4982805	2232.22
decision tree	1	interaction vars	2884486	1698.38	4982805	2232.22
random forest		price~.	2927320	1710.941	5037517	2244.44
random forest		logprice~.	5294780	2301.039	7315419	2704.7
random forest		logprice~logcarat	1812793	1346.4	6958801	2637.95
random forest		interaction vars	1123558	1059.98	4011680	2002.91

Table 7 - Comparison of Model Performance

Conclusions

It is clear based on our exploratory data analysis, variable model selection, and exploration with building linear regression, decision trees, and random forests that the most important variable to consider when diamond shopping is the size of the diamond in carats. In addition, when shopping at either Blue Nile or Ashford jewelers, the clarity value of VS2 and color levels of G, H, or I can have a significantly different price relative to the other store. Furthermore, the best predictor of price is found by using three main variables: the carat size, if the cut ideal or not, and if the diamond is being purchased at Blue Nile or not.

Some recommendations to improve these models are the more extensive use of pruning on the decision trees and the use of boosting on the random forest models. Finally, it should be noted that by using our finding through our EDA we discovered interactions that, when applied to our models, greatly improved their predictive accuracy. For example, our linear regression model which used transformed variables with interaction variables performed much better than simply performing a "wood chipper" style approach with the much more lauded random forest algorithm on the entire data set. It wasn't until we used the variables found in our interactions within our random forest model were we finally able to achieve an average test error better than that which we had achieved with interactions in the linear regression model. This is a good example of how and why understanding our data and using sound statistical methods are important to creating effective models, regardless of the type of model being used.

Appendix

Hastie, T., James, G., Tibshirani, R., Witten, D. (2013). *An Introduction to Statistical Learning with Applications in R*. New York, NY: Springer.

Johnson, W. & Myatt, G. (2014). *Making Sense of Data I. A Practical Guide to Exploratory Data Analysis and Data Mining*. Hoboken, New Jersey: Wiley.

Mount, J. & Zumel, N. (2014). *Practical Data Science with R*. Shelter Island, NY: Springer.

R Code

```
### FIGURE 1####
par(mfrow=c(2,2))
hist(diamonds$price, xlab="$ US Dollars", main="Price")
hist(log(diamonds$price), xlab="$ US Dollars", main="LogPrice")
hist(diamonds$carat, xlab="Size in carats", main="Carat")
hist(log(diamonds$carat), xlab="Size in carats", main="LogCarat")


####FIGURE 2 ######
histogram(diamonds$store, xlab="Store Names", scales=list(x = list(rot = 45)),main="Jewelers")


#####FIGURE 3########
xyplot(log(diamonds$price) ~ log(diamonds$carat), data=diamonds,
     type = c("p","smooth"), xlab="Log of Carat", ylab="Log of Price", main="Log of Price and Carat")


#####FIGURE 4#######
bwplot(log(diamonds$price) ~ clarity , data=diamonds,xlab="Clarity Value", ylab="Log Price", scales=list(x =
list(rot = 45)),main="Clarity", type = c("smooth"))
#####FIGURE 5#######
bwplot(logprice ~ color, data=diamonds, groups = store,
       panel = "panel.superpose",
       panel.groups = "panel.linejoin",
       xlab = "channel - treatment",
       key = list(lines = Rows(trellis.par.get("superpose.line"),
                               c(1:10, 1)),
                  text = list(lab = as.character(unique(diamonds$store))),
                  columns = 3, title = "Store value"))


#######FIGURE 6########
tree.fit <- tree(logprice~., data=train.tr.p)
summary(tree.fit) # summarize findings

plot(tree.fit) # plot model
text(tree.fit, pretty=0)
# tree price~. selects carat, ideal:No, BlueNile:No
```

```
# tree logprice~. selects carat only
# tree logprice~ logcarat selects logcarat only


#######FIGURE 7########
tree.fit <- tree(price~., data=train)
summary(tree.fit) # summarize findings

plot(tree.fit) # plot model
text(tree.fit, pretty=0)
# tree price~. selects carat, ideal:No, BlueNile:No
# tree logprice~. selects carat only
# tree logprice~ logcarat selects logcarat only



######TABLE 1########
###### data description table ######
VAR.NAME = colnames(diamonds)
VAR.CLASS <- c("numeric","ord-factor","ord-factor","factor","factor","factor","integer")
LEVELS <- c("0", "10","11","2","3","12","0")
DESCRIPTION <- c("weight of diamond",
                 "clarity of diamond",
                 "color of diamond",
                 "type of cut",
                 "jewler location",
                 "name of jewler",
                 "cost in US dollars")
var.descrip= data.frame(VAR.NAME, VAR.CLASS, LEVELS, DESCRIPTION)
write.csv(var.descrip, file = "data_dictionary_diamonds.csv")



#########TABLE 2############
###### summarize numeric data ######
my.numeric.summary <- function(in.orig.df){

  # check and create df with only numeric columns for the summary function
  col.to.add <<- numeric()  # initialize column vector

  for (i in 1:ncol(in.orig.df)){
    if(is.numeric(in.orig.df[,i]) == 'TRUE'){
      col.to.add = append(col.to.add,i)  # add column number to vector
    }
  }

  in.df <-in.orig.df[col.to.add]  # using '<<' adds object to global env for use outside function
  VAR.NAME <- colnames(in.df);   # get variable names from df

  # histogram of distribution of numeric variables
  for (i in 1:ncol(in.df)){
    hist(in.df[,i])
  }

  # count not-NAs function
  not.na <- function(x){
    out <- sum(-1*(is.na(x)-1));
```

```
  return(out);
}
NOT.NA <- apply(X=in.df,MARGIN=2,FUN='not.na');

# count NAs function
count.na <- function(x){
  out <- sum(is.na(x));
  return(out);
}

#### OUTLIER FUNCTION ####
find.outliers <- function(x){

  # count outliers in each variable based on outside Q01 or Q99 (these limits are adjustable)
  # create quantile of input
  quantile.99=quantile(x,c(0.99), na.rm=T)
  quantile.01=quantile(x,c(0.01), na.rm=T)

  # create min and max of input
  min= min(x)
  max= max(x)

  # initial outlier value for this column loop
  outlier = 0

  for(j in 1:length(x)){  # loop through each value in each column

    # 1) find observation value
    obs=x[j]

    # 2) find observation and outlier barrier ratios
    obs.min.ratio = obs/min
    obs.max.ratio = max/obs
    outlier.min.ratio = quantile.01/min
    outlier.max.ratio = max/quantile.99

    # 3) compare if observation ratio is less than outlier ratios
    result.min = obs.min.ratio < outlier.min.ratio #if observation/min ratio less than outlier/min ratio = outlier
    result.max = obs.max.ratio < outlier.max.ratio #if max/obs ratio less than max/outlier ratio = outlier

    # result.min
    if(result.min == 'TRUE' || result.max == 'TRUE'){
      outlier = outlier + 1
    }
  }
  return(outlier)
}
#### END OUTLIER SECTION ####


#### combine functions and apply() to the numeric data frame #####
IS.NA <- apply(X=in.df,MARGIN=2,FUN='count.na');  # number of NAs
PCT.NA <- round(IS.NA/(NOT.NA+IS.NA),digits=3);   # percent that are NAs
# Note that we need to account for computations with NA values;

p <- c(0.01,0.05,0.25,0.50,0.75,0.95,0.99); # set quantiles we want to use
Q <- round(apply(X=in.df,MARGIN=2,FUN='quantile',p,na.rm=TRUE),digits=3); # quantiles
```

```
  tQ.df <- as.data.frame(t(Q));  # transpose quantile vector and turn into data frame to display properly later
  colnames(tQ.df) <- c('Q01','Q05','Q25','Q50','Q75','Q95','Q99');  # change col names in data frame to be more
readable

  MIN <- round(apply(X=in.df,MARGIN=2,FUN='min',na.rm=TRUE),digits=3); # min value
  MAX <- round(apply(X=in.df,MARGIN=2,FUN='max',na.rm=TRUE),digits=3); # max value
  MEAN <- round(apply(X=in.df,MARGIN=2,FUN='mean',na.rm=TRUE),digits=3); # mean value
  SD <- round(apply(X=in.df,MARGIN=2,FUN='sd',na.rm=TRUE),digits=3); # Standard deviation value
  OUT = apply(X=in.df,MARGIN=2,FUN='find.outliers')  # outlier values for each variable stored in vector


  # combine all results into one data frame for display
  wide.df <- cbind(VAR.NAME,NOT.NA,IS.NA,PCT.NA,MIN,tQ.df,MAX,MEAN,SD,OUT);
  rownames(wide.df) <- seq(1,dim(wide.df)[1],1);   # rename rownames to numeric values

  # Write CSV in R
  write.csv(wide.df, file = "diamonds_summary_numeric_data.csv")

  return(wide.df);
}

my.numeric.summary.df = my.numeric.summary(in.orig.df=diamonds)  # test my.summary() function with
diamonds data set

#####         end numeric summary                ######
```

```
#######TABLE 3 and 4 #########

#### Determine if factor means are different for given variable ####

# 1) run boxplot to visually estimate possible differences
par(mfrow=c(2,2))

bwplot(log(diamonds$price) ~ clarity , data=diamonds)
bwplot(log(diamonds$price) ~ cut , data=diamonds)
bwplot(log(diamonds$price) ~ color , data=diamonds)
bwplot(log(diamonds$price) ~ store , data=diamonds)

# 2) use ANOVA to compare differences btn 3 or more means
# checking means of factors on response
results = aov(logprice ~ clarity, data=diamonds)

# sig differences for mean differences of these factors on logprice
#     store (p=0.00000396)
#     channel (p=0.00495)
#     cut (p=0.00728)
#     clarity (p=0.000881)

# 3) if ANOVA significant: must determine which means are different
#    pairwise T-Test and/or TukeyHSD to determine where mean differences are
#    based on findngs from ANOVA test on mean differences
TukeyHSD(results, conf.level = 0.95)
pairwise.t.test(diamonds$logprice, diamonds$clarity, p.adjust="bonferroni")
# sig diff for stores:
#     Ashford- Blue Nile (p=0.0294)
```

```
#     Ashford- Riddles (p=0.00000023)
#     Blue Nile - Riddles (p=.00034)
#     Fred Meyer - Riddles (p=04821)
#     Goodmans - Riddles (p=0.00020)

# sig diff for channels:
#     Mall - Internet (p=0.0035)

# sig diff for cut:
#     Ideal - not-Ideal (p=0.0073)

# sig diff for clarity:
#     I1 - VVS2 (p=0.00056)
#     I1 - VS2 (p=0.01241)
#     I1 - SI1 (p=0.00615)
########################


########TABLE 5##########
# 2)  if an interation appears in the graph (examples: http://courses.washington.edu/smartpsy/interactions.htm)
#     we can check it by creating a data frame with only those values and levels
#     and check if their means are significant with a t.test
#
# want to test mean logprice of VS1 ideal vs mean price of VS1 not ideal
  internet.bluenile <- diamonds %>%
  filter(channel=="Internet" & store=="Blue Nile") %>%
    select(logprice)
#table(diamonds$store, diamonds$channel)

  internet.ashford <- diamonds %>%
  filter(channel=="Internet" & store=="Ashford") %>%
    select(logprice)
# dim(df.clarity.vvs2)
#
# 3) applying t.test to test for significant difference between means
#
  t.test(internet.bluenile $logprice, internet.ashford $logprice, p.adjust="bonferroni")
#
#



##### MODELING ##########

########################################

# NAIVE - simple linear model
lm.fit <- lm(price~ carat, data=train)
pred.lm <- predict(lm.fit, newdata=test)
lm.mse <- mean((pred.lm-test$price)^2)  # 6686648
lm.error <- sqrt(lm.mse) # 2585.85



# NAIVE - backward selection w/o any transformations
bkwd.subset.fit = regsubsets(price~.,train,nvmax=29,method="backward")
```

```
# test for best # variables to use based on model fit on test set
# create matrix to use in subset selection
test.mat = model.matrix(price~., data=test)
val.errors=rep(NA,29)
  for(i in 1:29){
    coefi=coef(bkwd.subset.fit, id=i)
    pred=test.mat[,names(coefi)]%*%coefi
    val.errors[i]=mean((test$price-pred)^2)
  }
min.error = which.min(val.errors)
val.errors[min.error] # MSE = 3851158 (28 variables)
bkwd.errors.10 = sqrt(val.errors[min.error])  # error $1962.43 (28 variables)

# show coefficents for model built with 28 vars on test set using backward var selection
bkwd.subset.best.27 = regsubsets(price~., data=test, nvmax=29, method="backward")
coef(bkwd.subset.best.27,27)
# (Intercept)         carat           D           E          FF           G           H
# -7699.1390    11586.0401     9433.3851     8498.1432     8096.0662     7141.7779     6498.6512
#     I           J         VS2         SI1         SI2 independent    internet
# 5574.0476     4042.6910    -1022.1923    -2296.7667    -2679.3013     -365.7019    -4700.4053
#   ideal      Chalmers      University         Kay       Zales      Danford     BlueNile
# 434.5296    -3215.9227    -3154.5830     -387.7911      196.4776    -3542.3603     809.6390
#     K           L           M          FL         I1          I2          I3
# 0.0000      0.0000      0.0000      0.0000    -3821.4102      0.0000      0.0000



# transforming response:price to logprice on train and test sets
train$logprice <- log(train$price)
train$price = NULL
test$logprice <- log(test$price)
test$price = NULL



# NAIVE - simple linear model
lm.fit <- lm(logprice~logcarat, data=train.tr.pc)
exp.pred.lm <- exp(predict(lm.fit, newdata=test.tr.pc))
lm.mse.logY <- mean((exp.pred.lm-exp(test.tr.pc$logprice))^2)  # 7506122
lm.error.logY <- sqrt(lm.mse.logY) # 2739.73



# backward selection w. transformations of response
bkwd.subset.fit = regsubsets(logprice~.,train,nvmax=37,method="backward")

# test for best # variables to use based on model fit on test set
test.mat = model.matrix(logprice~., data=test) # create matrix to use in subset selection
val.errors=rep(NA,37)
for(i in 1:29){
  coefi=coef(bkwd.subset.fit, id=i)
  exp.pred=exp(test.mat[,names(coefi)]%*%coefi)
  val.errors[i]=mean((exp(test$logprice)-exp.pred)^2)
}
min.error <- which.min(val.errors)
```

```
MSE <- val.errors[min.error] # MSE = 4813277 (10 variables) , NOTE: MSE (1 variable = same as
logprice~carat from above)
bkwd.errors.10 = sqrt(val.errors[min.error])  # error $2193.92 (10 variables)

# show coefficents for model built with 10 vars on test set using backward var selection
bkwd.subset.best.10 = regsubsets(logprice~., data=test, nvmax=29)
coef(bkwd.subset.best.10,7)

# BEST 10 BKWD VARIABLE SELECTION ON ENTIRE TEST SET w. response = log price
# (Intercept)     carat         I          J          IF        FredMeyer
# 7.00048091   1.54644085  -0.22348621  -0.38450214  -0.11908715  -0.09596698
#   Kay         Danford         M           I2          I3
# 0.46100698   0.03553032   0.00000000   0.00000000   0.00000000



# LM - using the 10 backward selected variables from training set
lm.bwd.fit <- lm(logprice ~ carat+D+I+J+K+IF+L+internet+ideal+Ausmans, data=train)
exp.pred.lm.bwd <- exp(predict(lm.bwd.fit, newdata=test))
lm.bwd.mse.logY <- mean((exp.pred.lm.bwd-exp(test$logprice))^2)  # 4282908
lm.bwd.error.logY <- sqrt(lm.bwd.mse.logY) # 2069.52



# forward selection w. log transformation of response
fwd.subset.fit = regsubsets(logprice~.,train,nvmax=37,method="forward")
#summary(fwd.subset.fit)

# test for best # variables to use based on model fit on test set
test.mat = model.matrix(logprice~., data=test) # create matrix to use in subset selection
val.errors=rep(NA,37)
for(i in 1:29){
  coefi=coef(fwd.subset.fit, id=i)
  exp.pred=exp(test.mat[,names(coefi)]%*%coefi)
  val.errors[i]=mean((exp(test$logprice)-exp.pred)^2)
}
min.error <- which.min(val.errors)
MSE <- val.errors[min.error]     # MSE = 6946762 (10 variables)
fwd.errors.10 = sqrt(val.errors[min.error])  # error $2635.67 (10 variables)

# show coefficents for model built with 10 vars on test set using forward var selection
fwd.subset.best.10 = regsubsets(logprice~., data=test, nvmax=29)
coef(fwd.subset.best.10,min.error)

# BEST 10 FWD VARIABLE SELECTION ON ENTIRE TEST SET w. response = log price
#   ---- same variables and coefficients as with backward selection ----
# (Intercept)      carat         I          J          IF        FredMeyer
# 7.00048091   1.54644085  -0.22348621  -0.38450214  -0.11908715  -0.09596698
#     Kay         Danford         M           I2          I3
# 0.46100698   0.03553032   0.00000000   0.00000000   0.00000000


# Backward and Forward subset selection methods both selected 10 variables as
# the optimum for the test set. Hence, if we select the best ten-variable model
# on the entire data set the same coefficients and variables will be selected
#
```

```
# BEST 10 VARIABLE SELECTION ON ENTIRE DATA SET w. response = log price
# (Intercept)      carat          D          I          J          K
# 7.08804403  1.58371926  0.17127766 -0.22981697 -0.31526895 -0.51745102
#     IF       internet      ideal     Ausmans         L
# 0.13320980 -0.20484511  0.06536706 -0.08875137 -0.74635540



# LM - using forward selected variables on train set
lm.fwd.fit <- lm(logprice ~ carat+D+I+J+E+SI2+L+I1+I2+internet, data=train)
exp.pred.lm.fwd <- exp(predict(lm.fwd.fit, newdata=test))
lm.fwd.mse.logY <- mean((exp.pred.lm.fwd-exp(test$logprice))^2)  # 8054493
lm.fwd.error.logY <- sqrt(lm.fwd.mse.logY) # 2838.04



# stepwise selection w. log transformation of response
stp.subset.fit = regsubsets(logprice~.,train,nvmax=24,method="seqrep")
summary(stp.subset.fit)

# test for best # variables to use based on model fit on test set
test.mat = model.matrix(logprice~., data=test) # create matrix to use in subset selection
val.errors=rep(NA,24)
for(i in 1:24){
  coefi=coef(stp.subset.fit, id=i)
  exp.pred=exp(test.mat[,names(coefi)]%*%coefi)
  val.errors[i]=mean((exp(test$logprice)-exp.pred)^2)
}
min.error <- which.min(val.errors)
MSE <- val.errors[min.error] # MSE = 3568662 (18 variables)
stp.errors.10 = sqrt(val.errors[min.error])
error <- sqrt(MSE)   # error $1889.09 (18 variables)

stp.subset.best.18 = regsubsets(logprice~., data=test, nvmax=24)
coef(stp.subset.best.18,min.error)

# BEST 18 STEPWISE VARIABLE SELECTION ON ENTIRE TEST SET w. response = log price
# (Intercept)        carat            D            H            I            J           IF
# 7.35022091    1.59216574    0.11284817   -0.01538028   -0.23939530   -0.27746507   -0.06265690
#     SI2       independent     internet     FredMeyer         Zales        Danford          M
# -0.02114809   -0.29363058   -0.38093959   -0.13444439    0.21298435   -0.05714899    0.00000000
#      FL            I1           I2           I3         Ashford
# 0.00000000   -0.64416205    0.00000000    0.00000000   -0.13680266



# LM - using stepwise selected variables on train set
lm.stp.fit <- lm(logprice ~
carat+D+H+I+J+IF+internet+FredMeyer+Zales+Danford+I1+Ashford+independent, data=train)
exp.pred.lm.stp <- exp(predict(lm.stp.fit, newdata=test))
lm.stp.mse.logY <- mean((exp.pred.lm.stp-exp(test$logprice))^2)  # 7528329
lm.stp.error.logY <- sqrt(lm.stp.mse.logY) # $2743.78
```

```
#### BEST SUBSET SELECTION ####
# run exhaustive algorithm over entire data set with response: logprice
best.subset.fit = regsubsets(logprice~.,train, nvmax=29)
best.subset.summary <- summary(best.subset.fit)
which.min(best.subset.summary$bic)

par(mfrow=c(2,2))
# 18 variables has min BIC value
plot(best.subset.summary$bic, xlab="Number of Variables", ylab="BIC", type='l')
points(18,best.subset.summary$bic[18], col="red",cex=2,pch=20)
# 21 variables for min cp value
plot(best.subset.summary$cp, xlab="Number of Variables", ylab="CP", type='l')
points(21,best.subset.summary$cp[21], col="blue",cex=2,pch=20)
# 22 variable for max Adjusted R^2
plot(best.subset.summary$adjr2, xlab="Number of Variables", ylab="ADJ R^2", type='l')
points(22,best.subset.summary$adjr2[22], col="green",cex=2,pch=20)

# coefficients and variables for best subset on full data set
coef(best.subset.fit,18)
# (Intercept)        carat         D             E         I             J         IF
# 7.0571490209  1.5686190566  0.2338614193  0.1612328755 -0.2061530766 -0.2689735827  0.3811967625
#     VS1           VS2          SI1       internet        ideal       Chalmers      R.Holland
# 0.0027593079  0.0255172889  0.0045684621 -0.2008481206  0.0539164031 -0.1631546256  0.1671012895
#     Ausmans        Kay         BlueNile          L          I2
# -0.1028465260  0.0366122712  0.0007605737 -0.7061582509 -0.6548241156

# test for best # variables to use based on model fit on test set
test.mat = model.matrix(logprice~., data=test) # create matrix to use in subset selection
val.errors=rep(NA,37)
for(i in 1:29){
  coefi=coef(best.subset.fit, id=i)
  exp.pred=exp(test.mat[,names(coefi)]%*%coefi)
  val.errors[i]=mean((exp(test$logprice)-exp.pred)^2)
}
min.error <- which.min(val.errors)
MSE <- val.errors[min.error]      # MSE = 5273303 (21 variables)
best.errors.21 = sqrt(val.errors[min.error])  # error $2296.37 (21 variables)

best.subset.best.21 = regsubsets(logprice~., data=test, nvmax=29)
coef(best.subset.best.21,min.error)
# (Intercept)         carat          D          H          I          J
# 7.32965857     1.59731321     0.08972663    -0.03040024    -0.24673242    -0.25625019
#     IF            VS2          SI2         independent    internet         ideal
# -0.04787783     0.01207307    -0.03640947    -0.28708583    -0.40580718     0.08475070
#   FredMeyer       Zales         Danford          L              M            FL
# -0.09555311     0.23103726    -0.03803485     0.00000000     0.00000000     0.00000000
#     I1            I2           I3           Ashford
# -0.65154035     0.00000000     0.00000000    -0.11377441



# LASSO - variable selection
x = model.matrix(logprice~.,train)[,-1]  # dependent variables training set
y = train$logprice  # response variable training set
x.test = model.matrix(logprice~.,test)[,-1]  # dependent variables test set
y.test = test$logprice  # response variable test set
grid = 10^seq(10,-2,length=100) # range of values for lambda
```

```
lasso.fit = glmnet(x,y,alpha=1,lambda=grid)
par(mfrow=c(1,1))
plot(lasso.fit)

set.seed(71)
cv.out=cv.glmnet(x,y,alpha=1)
plot(cv.out)
bestlam=cv.out$lambda.min  # best lambda = 0.002077423
exp.lasso.pred=exp(predict(lasso.fit,s=bestlam,newx=x.test))
MSE <- mean((exp.lasso.pred-exp(y.test))^2)
MSE  # 5994621
lasso.error <- sqrt(MSE)
lasso.error # 2448.391

# show model coefficients on test set using best lambda (0.002077423)
lasso.coef = predict(lasso.fit,type="coefficients", s=bestlam)[1:20,]
lasso.coef[lasso.coef!=0]
# (Intercept)       carat       D          E          FF         I          J          L          IF
# 7.17544650  1.50332188  0.25730257  0.17900644  0.02226282 -0.14468469 -0.18433991 -0.39735680
0.31716138
#   VVS1          VVS2        SI2
# 0.15859128  0.14897177 -0.07074255



# LM - using forward selected variables on train set
lm.las.fit <- lm(logprice ~ carat+D+E+FF+I+J+L+IF+VVS1+VVS2+SI2, data=train)
exp.pred.lm.las <- exp(predict(lm.las.fit, newdata=test))
lm.las.mse.logY <- mean((exp.pred.lm.las-exp(test$logprice))^2)  # 6681124
lm.las.error.logY <- sqrt(lm.las.mse.logY) # $2584.79




# select variables to use below based on best variable selection model results

# LM no interactions

# LM - simple (lm(logprice ~ carat))
# NAIVE - simple linear model
lm.fit <- lm(logprice~ carat, data=train)
exp.pred.lm <- exp(predict(lm.fit, newdata=test))
lm.mse.logY <- mean((exp.pred.lm-exp(test$logprice))^2)  # 7506122
lm.error.logY <- sqrt(lm.mse.logY) # 2739.73

train$logcarat<- log(train$carat)
train$carat = NULL
test$logcarat <- log(test$carat)
test$carat = NULL

# NAIVE - simple linear model
lm.fit <- lm(logprice~ logcarat, data=train)
exp.pred.lm <- exp(predict(lm.fit, newdata=test))
lm.mse.logY <- mean((exp.pred.lm-exp(test$logprice))^2)  # 6335009
```

```
lm.error.logY <- sqrt(lm.mse.logY) # 2516.944


# LM w. interactions
#   color-store - blue nile vs ashford, G-I - small sample sizes for all<--BN:G and Ashford:I no effect,
#   clarity-channel - none
#   clarity-store - blue nile vs ashford, VS2
#   channel-store - blue nile vs ashford, Internet <--- this interaction had no effect on MSE

lm.fit <- lm(logprice~ logcarat+BlueNile+Ashford+VS2+internet+G+H+I+
             BlueNile:I+BlueNile:H+Ashford:G+Ashford:H+
             BlueNile:VS2+Ashford:VS2, data=train)
exp.pred.lm <- exp(predict(lm.fit, newdata=test))
lm.mse.logY <- mean((exp.pred.lm-exp(test$logprice))^2)  # 4856701
lm.error.logY <- sqrt(lm.mse.logY) # 2203.792



par(mfrow=c(1,1))

############ DECISION TREE #############
tree.fit <- tree(logprice~., data=train.tr.p)
summary(tree.fit) # summarize findings

plot(tree.fit) # plot model
text(tree.fit, pretty=0)
# tree price~. selects carat, ideal:No, BlueNile:No
# tree logprice~. selects carat only
# tree logprice~ logcarat selects logcarat only
#
#
# check trained decision tree on test data and evaluate predictive accuracy
pred.tree.test <- predict(tree.fit, newdata=test)

# convert log(predicted) values back to original form
# for equal comparison of MSE and $error against
# a non-log transformed response variable
exp.pred.tree.test = exp(pred.tree.test)

# plot predicted response against test response
# must back transform test response for equal comparison against
# a non-log transformed response variable
plot(pred.tree.test, test$price)
abline(0,1)

# must back transform test response for equal comparison against
# a non-log transformed response variable
mse <- mean((pred.tree.test-test$price)^2)
# predictive accuracy (price ~. ) = 3458866
# predictive accuracy (logprice ~. ) = 4982805
# predictive accuracy (logprice ~ logcarat ) = 4982805
error <- sqrt(mse)
# $error (price~.) $1859.803 ----- less than logprice bc decision tree not affected by abnormally distributed
response vars
# $error (logprice~.) $2232.22  ----- same as logprice~logcarat bc tree only selects carat or logcarat in either
tree, no other variables selected in either
```

```
# $error (logprice~logcarat) $2232.22 ---- tree not affected by non-gaussian distributions
# pruning the tree at various points could improve our results as well



# Random Forest
set.seed(71)
rf1.fit <- randomForest(logprice~ logcarat+BlueNile+Ashford+VS2+internet+G+H+I,
                        data=train.tr.pc, mtry=5, importance=TRUE)
# using p/3 for mtry=12 for regression approach

exp.pred.rf1.fit <- exp(predict(rf1.fit, newdata=test.tr.pc)) # n.valid post probs
# validation predictions
plot(exp.pred.rf1.fit)
importance(rf1.fit)
# price~. important variables: carat, J, SI2, ideal, Riddles, idependent
# logprice~. important variables: carat, J, SI2, ideal, BlueNile
# logprice~logcarat important variables:
# logprice~variables used in interactions: important variables: logcarat, I, BlueNile, VS2. internet

MSE <- mean((exp.pred.rf1.fit-exp(test.tr.pc$logprice))^2) # mean prediction error
MSE  # price ~. 5037517---logprice~. 7315419---logprice~logcarat 6958801---w interaction variables
4011680
rf1.error <- sqrt(MSE)
rf1.error  # price~. $2244.44---logprice~. $2704.70---logprice~logcarat $2637.95---w interaction varaibles
2002.91
```