Crystal Mosley
MSDS 454 Sec 55

Individual Assignment #1

Introduction

The purpose of this report is to use Brian Pope's case study on diamond prices based on their physical characteristics, as well as where they are sold, by using the *two months salary* data set provided. I am going to perform Data Quality Checks and then go through Exploratory Data Analysis using multiple variable selection algorithms as required to fit regression models. I will fit four final models by the end of the report and provide an appendix of the R code used.

Data Quality Check

The data utilized for this analysis is the *two months salary* provided by Brian Pope's case study. The data set contains 425 observations and 7 variables to begin with. I added two extra variables by the end of this report: "channel" that takes in two binary indicators' ("YES", "NO"), and "logprice" since I transformed the price variable. No missing values existed within this dataset. **Table 1** displays the Variable, Type, and Descriptions of each.

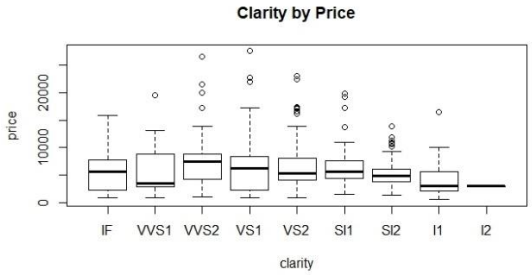
Table 1: Variables from Two Months Salary

Variable	Type	Description
carat	Numeric	Gemstone std unit of weight
color	Factor	GIA's color grading scale: D, E, F, G, H, I, J, K, L (only 1-9 used in this analysis)
clarity	Factor	GIA's scale of stone purity: IF, VVS1, VVS2, VS1, VS2, SI1, SI2, I1, I2 (only 2-10 used in this analysis)
cut	Factor	Cut of diamond: Ideal, Not_Ideal
channel	Factor	Sales channel: Independent, Mall, Internet
store	Factor	Stores included in research: Ashford, Ausmans, Blue_Nile, Chalmers, Danford, Fred_Meyer, Goodmans, Kay, R_Holland, Riddles, University, and Zales
price	Numeric	Price in USD

The response variable is price, the rest are the original predictor variables. All predictor variables are categorical except for carat. Color and clarity were integers until I converted both variables into factors, and I added the labels that were used for each with respect to what was in the data set. Another variable type that was changed was for price – it went from integer to numeric to go better with the analysis flow. I created histograms, boxplots, and scatterplots to get an understanding of any abnormalities and possible outliers in the data. The outliers resulting from this data set should be explored in greater detail due to the color and clarity scales possible throwing the plots off. For example, **Figure 1** shows that one clarity scale could be considered near the most valuable clarity, but it shows as an outlier within the boxplots. What's odd is that the best clarity grade, IF, doesn't show as very expensive in price while the other lower clarity grades show high in price. This would be deemed unusual, however, if color and cut are also used in the pricing of some of the diamonds, then this could

mean a lot of reasons as to why the lower grades are more expensive versus the higher grade (i.e.: IF could have the best clarity, but the worse cut and color, etc).

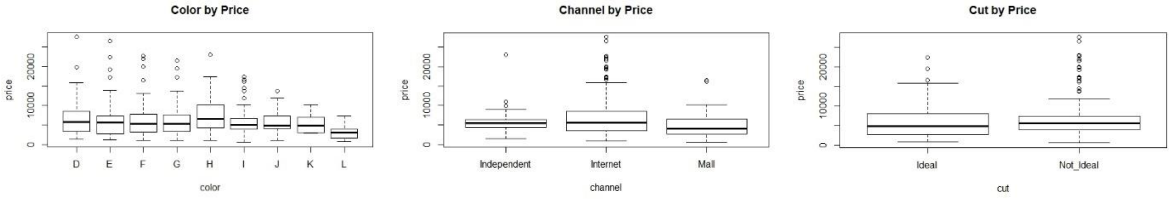
Figure 1: Boxplot of Price and Clarity Relationship



Exploratory Data Analysis

After initial data quality checks, I performed EDA on the data set. Since this is a regression problem, I needed to review the relationships between the predictor variables to better recognize them. **Figure 2** shows boxplots of the relationships between Price with Color, Cut, and Chanel. The channel pricing for Independent and Internet are more the same than the mall. Color is showing the most expensive as being D, E, and H with some outliers of course. Color grades E and H could be more expensive than the rest due to clarity and cut being better for some diamonds.

Figure 2: Boxplots of Color, Channel, and Cut



To get an overview of important relationships within the data, I used a tree plot. **Figure 3** outlines the variable carat as the most important predictor of price. The tree shows that the bottom five clarity grades are the lowest in pricing, and diamonds sold at Ashford, Fred_Meyer, and Kay sell for less than the other stores – this could be why only Kay currently exists in operation.

Figure 3: Tree Plot of Relationships

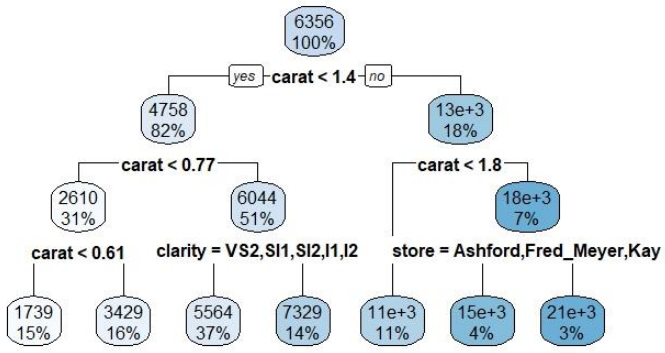
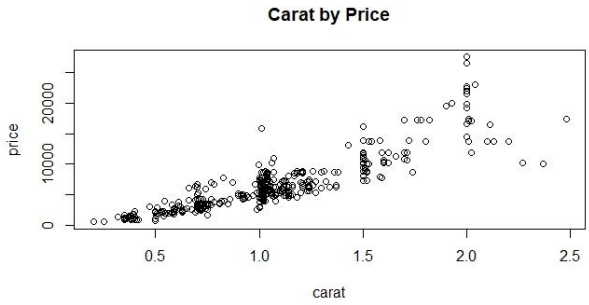


Figure 4 shows a correlation plot of the relationship between carat and price since the tree plot shows carat as the most important predict variable. The plot shows the higher the carat value, the more expensive the price becomes. Since the correlation between carat and price is about 0.88, they have a strong, positive relationship.

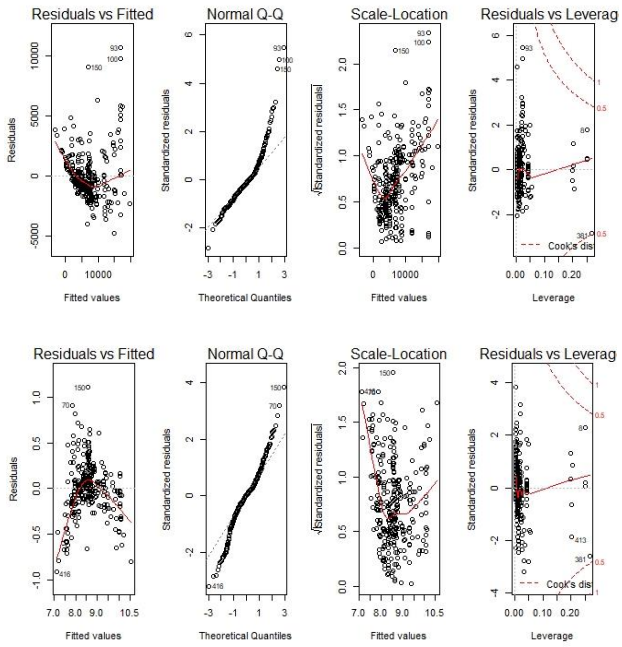
Figure 4: Corr Plot of Carat with Price



The Model Build

Now I move on to the building of my models. First, I split my data set into two sets: training and test; this way, I could use cross-validation to evaluate the accuracy of my models. The training set contained 70% of the data (298 observations) and the test set contained the rest of the observations. To classify which variables were the most important, I used backward variable selection. The top five variables were carat, color I, color J, color K, and color L. After creating a naïve model using these five variables as predictors and price as the response, there were noticeable issues with this model: non-normal distribution, heteroscedasticity, and nonlinearity. To try and fix this model, I used the log transform method of the same variables. **Figure 6** shows the *before* log transform (top) and the *after* log transform (bottom). It didn't help much.

Figure 6: Log Transform of Predictor Variables (before and after)



Since log(price) shows to be more normally distributed, I then used this variable as the response for the rest of the analysis. More variable selection methods were used for creating more models. **Table 2** show a summary of the techniques used.

Table 2: Variable Selection Methods Used

Variable Selection Method	Num of Pred	Adjusted R^2	Num of Pred	CP	Num of Pred	BIC
Backward	23	0.893539	19	14.55305	16	-576.225
Forward	23	0.893539	22	15.78015	10	-575.48
Stepwise	23	0.893539	19	14.55305	14	-578.462
All Subsets	23	0.893539	19	14.55305	14	-578.462

The LASSO technique was also used which automatically performs both variable selection and regularization to enhance the prediction accuracy and interpretability of the statistical model it produces. **Figure 8** shows the LASSO plot with cross-validation performed. The *bestlam* Lambda came out to be 0.000288169 which was used to estimate the coefficients and calculate the test errors in **Table 4**. LASSO chose a 21-variable model as the best which includes more clarity and colors variables than anything else.

Figure 8: LASSO Plot

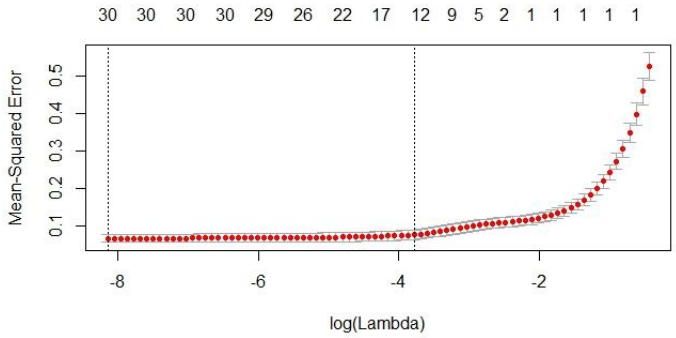


Table 4: LASSO Coefficient Estimates

Variable	Coeff. Est.	Variable	Coeff. Est.
(Int)	7.038796375	clarityVVS1	0
carat	1.516213564	clarityVVS2	0.11166203
colorE	0.104039691	clarityVS1	0
colorF	0	clarityVS2	0
colorG	-0.0136243	claritySI1	-0.000383586
colorH	0	claritySI2	-0.031110989
colorI	-0.16601489	clarityI1	-0.433471266

colorJ	-0.19519892	clarityI2	0
colorK	-0.69441599	cutNot_Ideal	-0.033580343
colorL	-0.44850348	channelInternet	-0.064853976

Model 1: Linear regression model with no interactions using the lm() function

Rather than using the 21-variable model chosen by LASSO, I reviewed all the predictors selected by all subset and stepwise and chose to use a 14-variable model because these same variables also appeared in the other two variable selection models but were the least complex of them all as far as modeling.

Table 5: Summary of Linear Model without Interaction Terms

Variable	Estimate	Std. Error	t-value	Pr	Significance
(Int)	7.07838	0.04572	154.821	< 2e-16	***
carat	1.55157	0.03391	45.756	< 2e-16	***
channel = "Mall" TRUE	0.40418	0.05183	7.798	1.22E-13	***
clarity == "I1"TRUE	-0.59335	0.08241	-7.2	5.44E-12	***
clarity == "SI2"TRUE	-0.10721	0.0423	-2.534	0.011811	*
clarity == "VVS2"TRUE	0.16525	0.0484	3.415	0.000732	***
color == "F"TRUE	-0.13457	0.04924	-2.733	0.006669	**
color == "G"TRUE	-0.19731	0.04705	-4.194	3.68E-05	***
color == "H"TRUE	-0.20202	0.04748	-4.255	2.84E-05	***
color == "I"TRUE	-0.35881	0.04545	-7.895	6.45E-14	***
color == "J"TRUE	-0.42459	0.06047	-7.021	1.64E-11	***
color == "K"TRUE	-1.0548	0.13203	-7.989	3.46E-14	***
color == "L"TRUE	-0.73275	0.12593	-5.819	1.60E-08	***
store == "Ausmans"TRUE	0.29572	0.10831	2.73	0.006727	**
store == "Goodmans"TRUE	0.66053	0.097	6.81	5.86E-11	***

A few things about this model: carat has the best estimate indicating a strong, positive relationship to log(price) which we noted earlier during the data exploration; on the other hand, color grade K had the strongest, negative relationship to log(price). This entire models' Adj R^2 value was about 0.89, which indicates the model fits the data well. VIF's were all under 2, confirming there were no multicollinearity in existence.

Model 2: Linear regression model including some interaction terms

For this model, the top 8 predictor variables were utilized from the stepwise model earlier. For this model, I found the best models using all interaction combinations available. Carat once again shows strong, positive relationship with log(price), and interacts with three other terms in **Table 6**, which is telling that carat is an important predictor variable.

Table 6: Summary of Linear Model with Interaction Terms

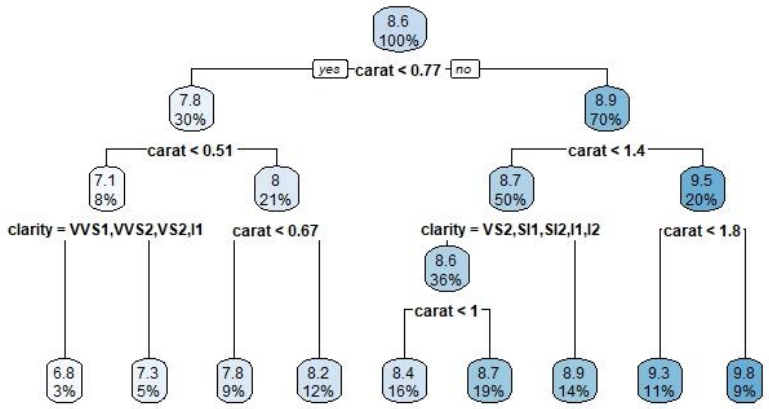
Variable	Estimate	Std. Error	t-value	Pr	Significance
(Int)	6.736022	0.078951	85.319	< 2e-16	***
carat	1.802736	0.057709	31.238	< 2e-16	***
clarity == "I1"	-0.247132	0.090908	-2.718	0.006968	**
store == "Riddles"	-0.036832	0.110202	-0.334	0.738463	
color == "I"	0.003795	0.070978	0.053	0.957403	
channel == "Internet"	0.043244	0.055344	0.781	0.435248	
color == "K"	-0.233967	0.321605	-0.727	0.46753	
color == "J"	0.148703	0.191955	0.775	0.439185	
store == "Goodmans"	0.578954	0.105687	5.478	9.58E-08	***
clarity == "VS2"	-0.062598	0.034785	-1.800	0.073003	.
channel == "Mall"	0.14909	0.073842	2.019	0.044437	*
cut == "Not_Ideal"	0.330023	0.079533	4.149	4.42E-05	***
clarity == "I1":store == "Riddles"	-0.771121	0.171741	-4.490	1.04E-05	***
color == "I":channel == "Internet"	-0.284824	0.081570	-3.492	0.000557	***
carat:color == "K"	-0.367609	0.221771	-1.658	0.098517	.
carat:color == "J"	-0.322304	0.151917	-2.122	0.034752	*
clarity == "VS2":channel == "Mall"	0.34959	0.104609	3.342	0.000945	***
carat:cut == "Not_Ideal"	-0.372166	0.071008	-5.241	3.15E-07	***

This entire models' Adj R^2 value was about 0.89, which indicates the model fits the data well; however, the VIF's were very high for some variables indicating an issue with multicollinearity.

Model 3: Tree model

For this model, I used the tree method with all predictor variables again, however, I used the log(price) to keep things cohesive across all the models. This model brings a little more complexity to it than the previous tree plot during EDA. **Figure 9** shows some noticeable differences within this model versus the original price tree model: the split on carat (top leaf node) is half (0.77) than the previous tree model; carats less than 0.51 and 1.4 utilize clarity as the decision maker for pricing; it's odd that cut didn't make it in the tree as a factor in deciding some of the pricing for the lower clarity grades.

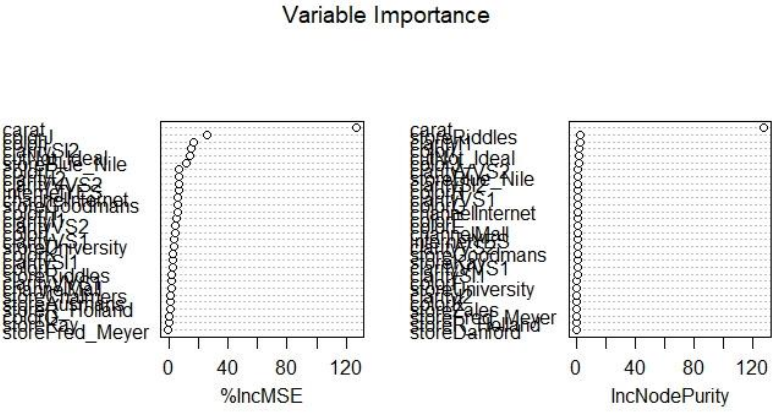
Figure 9: Tree Model



Model 4: Random Forest model

The last model is a Random Forest model. This type of model is an ensemble approach that can also be thought of as a form of nearest neighbor predictor. Ensembles are a divide-and-conquer approach used to improve performance. **Figure 10** shows the (ugly) results of a 10-fold repeated cross-validation RF model considering 18 variables that attained the lowest RMSE results. This model resulted in a 91.34% variance explained, using 500 trees.

Figure 10: Random Forest Model



Model Comparison

To identify which model performed best after fitting all models, I used the RSME (root mean square error) and MAE (mean absolute error) of the test set as shown in **Table 7**. Out of all the models, the Random Forest Model performed best. Focusing only on the RMSE, the tree model came second, linear interactions came third, and linear no interactions came last; however, when focusing only on the MAE, the tree model came second once again, but the linear no interaction came third, and the linear interactions came last.

Table 7: Model Performance

Model	dTrain: RMSE	dTrain: MAE	dTest: RMSE	dTest: MAE
Linear Model: No interactions	5627.34	3921.145	1790.202	940.274
Linear Model: Interactions	2353.941	1198.39	1383.583	955.6749
Tree Model	1129.989	716.347	780.8101	592.1088
Random Forest Model	1708.515	772.9977	634.1497	446.0044

Conclusion

After initial analysis, the Random Forest Model wins for best model to utilize in predicting the variables that will impact the price of diamonds. I believe more models should be vetted and more detailed analysis on the outliers found should occur before deploying the RF model into production. I would also want to try using different #-variable options to see how each of the current models would be affected because bias could potentially play a role within the current results if the user (myself) is allowed to choose however many variables to create the models.

Appendix A: R Code

```
# read in data from comma-delimited text file
diamonds <- read.csv("two_months_salary.csv")

# we can create a new channel factor called internet as a binary indicator
# the ifelse() function is a good way to do this type of variable definition
diamonds$internet <- ifelse((diamonds$channel == "Internet"),2,1)
diamonds$internet <- factor(diamonds$internet,levels=c(1,2),labels=c("NO","YES"))

#--------------------
# Data Quality Check
#--------------------

class(diamonds)
dim(diamonds) #2 new included: "internet" and "logprice"
str(diamonds) # two numeric (carat/logprice); three int (color/clarity/price); four factor
(cut/channel/store/internet)
summary(diamonds)

#Next, convert price to num and color/clarity to factor; keep price as int
diamonds$price <- as.numeric(diamonds$price)
diamonds$color <- as.factor(diamonds$color)
diamonds$clarity <- as.factor(diamonds$clarity)

summary(diamonds$color) # Includes GIA diamond colors from 1-9 only (D-L)
levels(diamonds$color) <- c("D", "E", "F", "G", "H", "I", "J", "K", "L")
```

```
summary(diamonds$clarity) # Includes GIA diamond clarity from 2-10 only (IF-I2)
levels(diamonds$clarity) <- c("IF", "VVS1", "VVS2", "VS1", "VS2", "SI1", "SI2", "I1", "I2")

# Fix data values so they have no spaces
summary(diamonds$cut)
levels(diamonds$cut) <- c("Ideal", "Not_Ideal")

summary(diamonds$store)
levels(diamonds$store) <- c("Ashford", "Ausmans", "Blue_Nile", "Chalmers",
                            "Danford", "Fred_Meyer", "Goodmans", "Kay",
                            "R_Holland", "Riddles", "University", "Zales")

#Re-review data
str(diamonds)

#Histograms for all predict vars: carat,color,clarity,cut,channel,store
for (i in 1:6){
  diaPlot = paste0("price ~ ", names(diamonds)[i]) #paste0 = concatenate vectors after converting to char
  p <- histogram(as.formula(diaPlot), data = diamonds, xlab = names(diamonds)[i])
  print(p)
}

#Boxplot for all predict vars to price, except carat
colnames(diamonds)
plot(diamonds$color, diamonds$price, main="Color by Price",
    xlab="color", ylab="price")
plot(diamonds$clarity, diamonds$price, main="Clarity by Price",
    xlab="clarity", ylab="price")
plot(diamonds$cut, diamonds$price, main="Cut by Price",
    xlab="cut", ylab="price")
plot(diamonds$channel, diamonds$price, main="Channel by Price",
    xlab="channel", ylab="price")
plot(diamonds$store, diamonds$price, main="Store by Price",
    xlab="store", ylab="price")

#--------------------------
# Exploratory Data Analysis
#--------------------------

#Tree plot with "rpart" package...not sure I understand this package purpose
rpartDia <- rpart(price ~ ., data = diamonds)
rpart.plot(rpartDia)

#Scatterplot for carat by price
plot(diamonds$carat, diamonds$price, main="Carat by Price",
    xlab="carat", ylab="price")

#Review corr between carat and price
```

```
cor(diamonds$price, diamonds$carat)
 #0.88

#Review quantile of carat at 0.99
quantile((diamonds$carat), probs = 0.99)
 #2.1252

#Plot only carat and arrange graphs in 3 col grid
caratHist <- histogram(~carat, data = diamonds)
caratHist

caratBP <- bwplot(~carat, data = diamonds)
caratBP

caratQuant <- qqmath(~carat, data = diamonds)
caratQuant

grid.arrange(caratHist, caratBP, caratQuant, ncol=3)

#---------------------------
# The Model Build - Initial
#---------------------------

set.seed(123)

#Create Train/Test data sets
dTrain <- sample_frac(diamonds, 0.70)
dTrainNum <- as.numeric(rownames(dTrain))
dTest <- diamonds[-dTrainNum,]

#Fit a naïve regression model using backwards variable selection
bvs_model1 <- regsubsets(price ~ ., data = dTrain, nvmax = 29, method="backward")
summary(bvs_model1)

# Create naïve model using top 5 predictors
naive_model1 <- lm(price ~ carat + (color=="I") + (color=="J") + (color=="K") + (color=="L"), data =
dTrain)

par(mfrow=c(1,1))
plot(density(resid(naive_model1)))
par(mfrow=c(1,4))
plot(naive_model1)

AIC(naive_model1) #5376.92
vif(naive_model1) # All results are close to 1, which is OK.

#Review price and carat to see if transformations are needed
par(mfrow=c(2,2))
```

```
hist(diamonds$price)
hist(dTrain$price)
hist(diamonds$carat)
hist(dTrain$carat)
 #Price distr looks exactly the same regardless of reg data and train data; carat distr look almost exact as well.
 #Gather log of price
  par(mfrow=c(1,1))
  logPrice <- log(diamonds$price)
  hist(logPrice)
  logPriceTrain <- log(dTrain$price)
  hist(logPriceTrain)
  #Looks more evenly distributed

  #Now plot logs of price data and train data, then put in grid
  priceHist <- histogram(~price, data = dTrain)
  logPriceHist <- histogram(~log(price), data = dTrain)
  grid.arrange(priceHist, logPriceHist, ncol=2)

  # Create log naïve model using top 5 predictors
  naive_model1_log <- lm(log(price) ~ carat + (color=="I") + (color=="J") + (color=="K") + (color=="L"), data = dTrain)

  plot(density(resid(naive_model1_log)))
  par(mfrow=c(1,4))
  plot(naive_model1_log)
  par(mfrow=c(1,1))

  AIC(naive_model1_log) #117.96 - much smaller than before
  vif(naive_model1_log) # same as non-log price

  #Show comparison of before and after
  par(mfrow=c(2,4))
  plot(naive_model1)
  plot(naive_model1_log)
  par(mfrow=c(1,1))

#Fit a naïve regression model using LOG backwards variable selection
bvs_model2 <- regsubsets(log(price) ~ ., data = dTrain, nvmax = 29, method="backward")
summary(bvs_model2)

#-------------------------
# The Model Build - Final
#-------------------------

rpartPriceLog <- rpart(price ~ ., data = dTrain)
rpart.plot(rpartPriceLog)
```

```
#Backward Selection
set.seed(123)

bvs_model3 <- regsubsets(log(price) ~ ., data = dTrain, nvmax = 29, method="backward")
summary(bvs_model3)

#Forward Selection
set.seed(123)

fs_model1 <- regsubsets(log(price) ~ ., data = dTrain, nvmax = 29, method="forward")
summary(fs_model1)

#Stepwise Selection
set.seed(123)

ss_model1 <- regsubsets(log(price) ~ ., data = dTrain, nvmax = 26, method="seqrep")
 #nvmax crashes with >26...
summary(ss_model1)

# All Subsets Selection
set.seed(123)

as_model1 <- regsubsets(log(price) ~ ., data = dTrain, nvmax = 29, method="exhaustive")
summary(as_model1)

#LASSO Model
grid <- 10^seq(10, -2, length=100)

set.seed(123)

dTrain_matrix  <- model.matrix(log(price) ~ ., data=dTrain)[,-1]

log_price <- log(dTrain$price)

set.seed(123)

lasso_model1 <- glmnet(dTrain_matrix, log_price, alpha=1, lambda=grid)
plot(lasso_model1)

 #Now use cross-val to select Lambda
  set.seed(123)

  cv.out.lasso <- cv.glmnet(dTrain_matrix, log_price, alpha=1)
  plot(cv.out.lasso)

  bestlamlasso <- cv.out.lasso$lambda.min
  bestlamlasso #0.000288169
```

```
lasso_dTrain_Predict <- predict(lasso_model1, newx = dTrain_matrix, s = bestlamlasso)
mean((lasso_dTrain_Predict - dTest)^2) #NA

out=glmnet(dTrain_matrix,log_price,alpha=1, lambda=grid)
lasso.coef=predict(out,type="coefficients",s=bestlamlasso) [1:20,]
lasso.coef

# Create best 21-variable stepwise model to compare to lasso model
set.seed(123)

m_stepwise21 <- lm(log(price) ~ carat + (channel=="Internet") + (channel=="Mall") +
                   (clarity=="I2") + (clarity=="SI1") + (clarity=="SI2") + (clarity=="VS1") +
                   (clarity=="VS2") +  (clarity=="VVS1") + (clarity=="VVS2") +
                   (color=="E") + (color=="F") + (color=="G") + (color=="H") +
                   (color=="I") + (color=="J") + (color=="K") + (color=="L") +
                   (cut=="Not_Ideal") + (store=="Ausmans") , data = dTrain)

coef(m_stepwise21)
plot(density(resid(m_stepwise21)))
par(mfrow=c(1,4))
plot(m_stepwise21)
par(mfrow=c(1,1))
AIC(m_stepwise21) #87.73165
vif(m_stepwise21) #All under 5

# Calculate stepwise21 train predictions
set.seed(123)

stepwise21.train <- predict(m_stepwise21, newdata = dTrain)
stepwise21.train.exp <- exp(stepwise21.train)
head(stepwise21.train.exp)


#------------------
# Model Build - Fit
#------------------

set.seed(123)

fit1 <- lm(log(price) ~ carat + (channel=="Mall") + (clarity=="I1") + (clarity=="SI2") +
             (clarity=="VVS2") + (color=="F") + (color=="G") + (color=="H") + (color=="I") +
             (color=="J") + (color=="K") + (color=="L") + (store=="Ausmans") + (store=="Goodmans"),
           data = dTrain)

summary(fit1) #ARsqr: 0.8869
AIC(fit1) #21.70701
vif(fit1) #All < 2
fit1Train_Exp <- exp(fit1$fitted.values)
```

```
par(mfrow=c(1,4))
plot(fit1)
par(mfrow=c(1,1))

#Linear regression model including some interaction terms
set.seed(123)

stepwise_int <- regsubsets(log(price) ~ .*., data = dTrain, method="seqrep")
summary(stepwise_int)

set.seed(123)

fit2 <- lm(log(price) ~ carat + (clarity=="I1")*(store=="Riddles") + (color=="I")*(channel=="Internet") +
carat*(color=="K") +
          carat*(color=="J") + (store=="Goodmans") + (clarity=="VS2")*(channel=="Mall") +
carat*(cut=="Not_Ideal"),
         data = dTrain)

summary(fit2) #ARsqr:  0.8939
AIC(fit2) #5.50042
vif(fit2) #range from 1.2 to 14.5
fit2Train_Exp <- exp(fit2$fitted.values)

par(mfrow=c(1,4))
plot(fit2)
par(mfrow=c(1,1))

#Create Tree Model
set.seed(123)

fit3 <- rpart(log(price) ~ ., data = dTrain)
fit3
fit3_plot <- rpart(log(price) ~ ., data = dTrain)
rpart.plot(fit3_plot)

#Create RandomForest model
set.seed(123)

fit4_baseline <- randomForest(dTrain_matrix, logPriceTrain, mtry=floor(sqrt(ncol(dTrain_matrix))),
importance=TRUE)
fit4_baseline

#https://machinelearningmastery.com/tune-machine-learning-algorithms-in-r/

control <- trainControl(method="repeatedcv", number=10, repeats=3, search="random")
set.seed(123)
mtry <- sqrt(ncol(dTrain_matrix))
mtry #5.656854
```

```
rf_random <- train(dTrain_matrix, logPriceTrain, method="rf", metric="RMSE", tuneLength=15,
trControl=control)
print(rf_random)

set.seed(123)

fit4 <- randomForest(log(dTrain$price) ~ ., data = dTrain_matrix, mtry=18, importance=TRUE)
fit4
fit4.train.exp <- exp(fit4$predicted)

plot(fit4, main = "Random Forest Plot")
varImpPlot(fit4, main = "Variable Importance")

#------------------
# Model Comparison
#------------------

#https://www.r-bloggers.com/calculate-rmse-and-mae-in-r-and-sas/
rmse <- function(error){
  sqrt(mean(error^2))
}

mae <- function(error){
  mean(abs(error))
}

#-----
#Fit 1
#-----

  # Calculate fit1 test predictions
  set.seed(123)
  fit1.pred <- predict(fit1, newdata = dTest)
  fit1.pred.exp <- exp(fit1.pred)

  # Calculate fit1 test errors
  actual <- dTest$price
  predicted <- fit1.pred.exp
  error <- actual - predicted
  fit1.test.rmse <- rmse(error)
  fit1.test.rmse #1790.202
  mae(error) #940.274

  # Calculate fit1 train predictions
  set.seed(123)
  fit1.train <- predict(fit1, newdata = dTest)
  fit1.train.exp <- exp(fit1.train)
```

```
  # Calculate fit1 train errors
  actual <- dTrain$price
  predicted <- fit1.train.exp
  error <- actual - predicted
  fit1.train.rmse <- rmse(error)
  fit1.train.rmse #5627.34
  mae(error) #3921.145

#-----
#Fit 2
#-----
  # Calculate fit2 test predictions
  set.seed(123)

  fit2.pred <- predict(fit2, newdata = dTest)
  fit2.pred.exp <- exp(fit2.pred)

  # Calculate fit2 test errors
  actual <- dTest$price
  predicted <- fit2.pred.exp
  error <- actual - predicted
  fit2.test.rmse <- rmse(error)
  fit2.test.rmse #1383.583
  mae(error) #955.6749

  # Calculate fit2 train predictions
  set.seed(123)

  fit2.train <- predict(fit2, newdata = dTrain)
  fit2.train.exp <- exp(fit2.train) # returns log(price) to price for interpretability

  # Calculate fit2 train errors
  actual <- dTrain$price
  predicted <- fit2.train.exp
  error <- actual - predicted
  fit2.train.rmse <- rmse(error)
  fit2.train.rmse #2353.941
  mae(error) #1198.39

#-----
#Fit 3
#-----

  # Calculate fit3 test predictions
  set.seed(123)

  fit3.pred <- predict(fit3, newdata = dTest)
  fit3.pred.exp <- exp(fit3.pred)
```

```
  # Calculate fit3 test errors
 actual <- dTest$price
 predicted <- fit3.pred.exp
 error <- actual - predicted
 fit3.pred.rmse <- rmse(error)
 fit3.pred.rmse #780.8101
 mae(error) #592.1088

 # Calculate fit3 train predictions
 set.seed(123)

 fit3.train <- predict(fit3, newdata = dTrain)
 fit3.train.exp <- exp(fit3.train)

 # Calculate fit3 train errors
 actual <- dTrain$price
 predicted <- fit3.train.exp
 error <- actual - predicted
 fit3.train.rmse <- rmse(error)
 fit3.train.rmse #1129.989
 mae(error) #716.347

#-----
#Fit 4
#-----

 set.seed(123)

 #Create test matrix
 test.matrix <- model.matrix(log(price) ~ ., data=dTest)[,-1]

 # Calculate fit4 test predictions
 set.seed(123)

 fit4.pred <- predict(fit4_baseline, newdata = test.matrix)
 fit4.pred.exp <- exp(fit4.pred)

 # Calculate fit4 test errors
 actual <- dTest$price
 predicted <- fit4.pred.exp
 error <- actual - predicted
 fit4.pred.rmse <- rmse(error)
 fit4.pred.rmse #634.1497
 mae(error) #446.0044

 # Calculate fit4 train predictions
 set.seed(123)
```

```
fit4.train <- predict(fit4_baseline, newdata = dTrain_matrix)
fit4.train.exp <- exp(fit4.train) # returns log(price) to price for interpretability

# Calculate fit4 train errors
actual <- dTrain$price
predicted <- fit4.train.exp
error <- actual - predicted
fit4.train.rmse <- rmse(error)
fit4.train.rmse #1708.515
mae(error) #772.9977
```