Andrea Bruckner
Predict 454 Sec 55

Individual Assignment #2

Introduction

In this report I am using the two months salary data set from Brian Pope's case study on diamond prices based on their physical characteristics, as well as where they are sold. After performing a Data Quality Check and Exploratory Data Analysis (EDA), I use several variable selection algorithms to fit regression models. I fit a few exploratory models before fitting four final models. An appendix of R code relevant to the tables and graphics presented is located at the end of this report.

Data Quality Check

The data used in this report is the two months salary data set from Brian Pope's case study. The data set contains 425 observations and 7 variables. There are no missing values. **Table 1** provides information about each variable after I performed data conversions.

Variable	Type	Description
carat	Numeric	Standard unit of weight used for gemstones
color	Factor	Standard color scale for grading diamonds, including grades D, E, F, G, H, I, J, K, and L
clarity	Factor	Standard scale for measuring the purity of the stone, including inclusion grades IF, VVS1, VVS2, VS1, VS2, SI1, SI2, I1, I2
cut	Factor	Cut of diamond, either Ideal or Not_Ideal
channel	Factor	Sales channel, including Independent, Internet, and Mall jewelers
store	Factor	Stores researched, including Ashford, Ausmans, Blue_Nile, Chalmers, Danford, Fred_Meyer, Goodmans, Kay, R_Holland, Riddles, University, and Zales
price	Numeric	Price in US dollars

Table 1: Variables in Two Months' Salary Data Set

With the exception of carat, all the predictor variables are categorical. The variables color and clarity were originally integers representing color grades and inclusion grades, respectively. I converted both of these to factors and renamed each level according to the grades described in Exhibit 1 in Brian Pope's case study. Some of the level names in the cut and store variables contained spaces, so I fixed these values to contain no spaces. I also decided to convert price from an integer to a numeric variable in order to facilitate analysis. Given that the case study focused on understanding which diamond characteristics and sale avenues influence price, I recognized this as a regression problem with price as the response variable.

After converting several variables to other data types, I created plots to try to get an idea of potential outliers or abnormalities in the data. Detecting outliers was difficult to do with the categorical variables in this data set. For instance, **Figure 1** contains box plots showing price given various color grades. Grade D is the most valuable color, so the light blue circle above $25,000 might be accurate. It also might be an outlier. Some of the other light blue circles might be outliers as well. The pricing of a diamond is not dependent on any one quality, so we cannot draw conclusions about potential outliers before examining the data further.

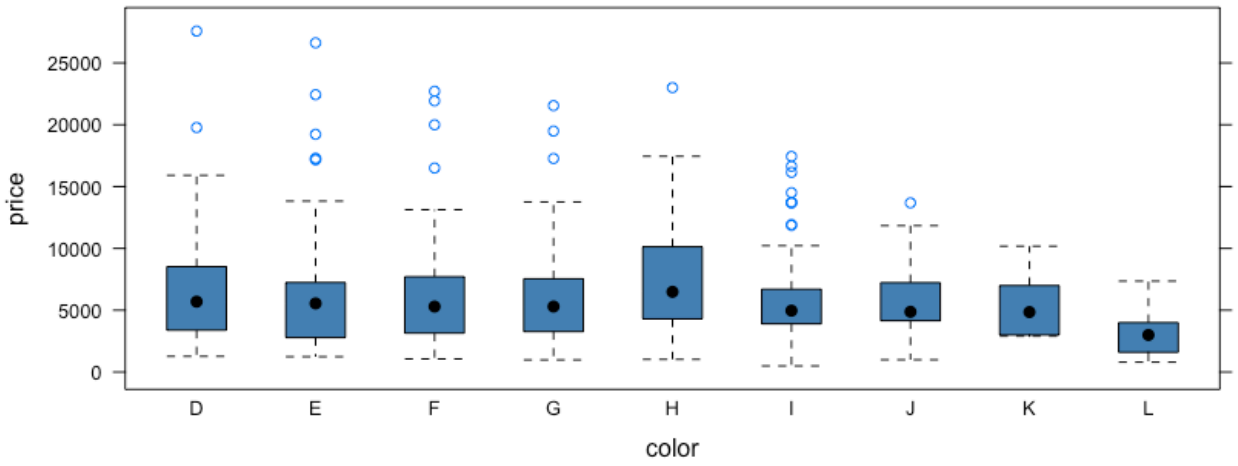

Figure 1: Box Plots Depicting Relationship Between Color and Price

Exploratory Data Analysis

After taking inventory of the data to make sure the data are correct and of good quality, I conducted a more in-depth analysis, known as EDA, to detect interesting relationships in the data. In the previous section, I mentioned that the type of statistical problem presented by the data is a regression problem. I decided to create box plots of each variable in order to better understand their relationships with price. **Figure 2** shows three predictors that I thought had some interesting relationships with price.

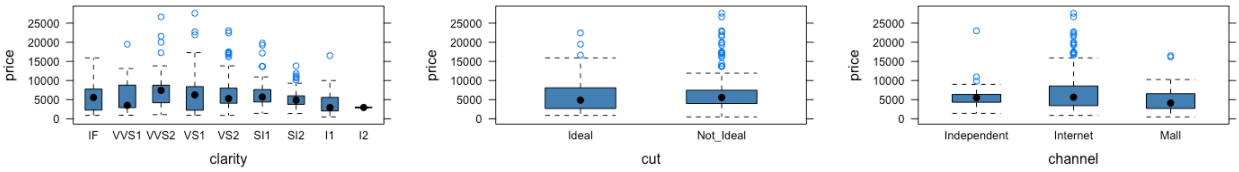

Figure 2: Box Plots of Select Predictors

Although some of the clarity factors had the kind of relationship with price that I expected (such as the lowest grade, I2, having the smallest price range), others did not. I thought that the highest grades of clarity, IF and VVS1, would fetch the highest prices, but some of the more middle grades, such as VVS2, VS1, and VS2, had higher price ranges and median prices than the higher grades. This could be due to some diamonds with higher clarity grades having lower color grades or not ideal cuts, but it is impossible to know at this point. Also, it is worth noting that the light blue circles might indicate potential outliers. However, a more thorough examination of the relationships in the data is necessary to draw conclusions.

I also thought it was interesting that diamonds classified as Not_Ideal had a higher median price as well as greater overall price range than Ideal diamonds. I expected the opposite. The fact that diamonds sold via the Internet had a larger and higher price range than diamonds sold in Independent stores or the Mall also surprised me. I expected Independent stores to have higher prices because I thought buyers would be willing to pay more money for diamonds they could inspect in person at a specialty store where they could consult with experts.

In addition to creating plots, I fit a tree model to get a quick overview of important relationships in the data. The tree model in **Figure 3** shows that carat is one of the most important predictors

of price. Generally, the higher the carat, the higher the price. The tree also shows that the lowest grades of clarity (VS2, SI1, SI2, I1, and I2) result in lower prices, and diamonds sold at Ashford, Fred_Meyer, or Kay often sell for less than diamonds at other stores.

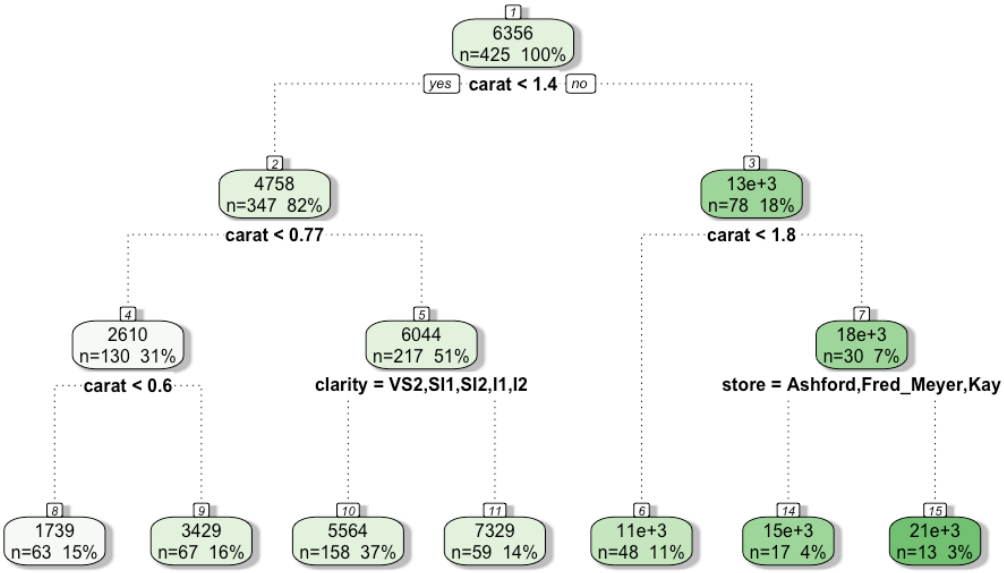

Figure 3: Tree Model of All Data

Because carat is an important predictor, I decided to plot its relationship with price. I determined a scatter plot was the most appropriate method for displaying their relationship. **Figure 4** shows that price generally increases with carat, although larger carat diamonds reveal more variance in price. I also calculated the correlation between price and carat: 0.8796315. This indicates these variables have a strong, positive relationship.

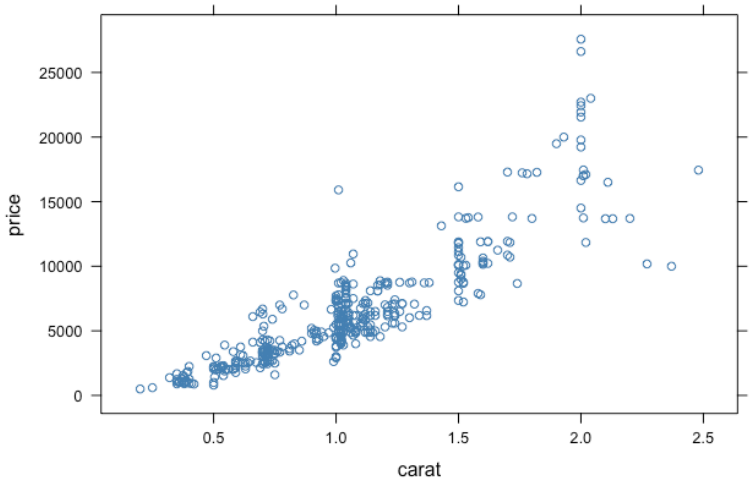

Figure 4: Correlation Plot of Carat and Price

In **Figure 5**, we can see that although carat does not have a normal distribution, it is not skewed enough to warrant any transformations. I tried a logarithmic transformation to confirm and noticed a remarkably worse distribution. The plots below show the variable untransformed.

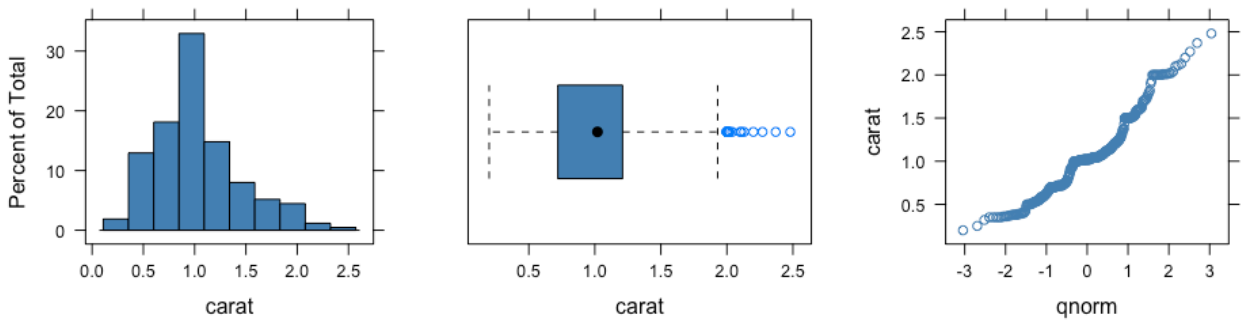

Figure 5: Carat Distribution Plots

Conducting EDA helped me identify interesting relationships and provided insights that I can use when building models. I expect to use carat and clarity in my models but will employ several variable selection algorithms in the next section to determine which other variables might prove valuable in predicting the response variable.

Model-Based Exploratory Data Analysis

After conducting a Data Quality Check and EDA, I split my data set in order to use cross-validation to evaluate the model. I used a random number generator to allocate 70 percent of the data to the training set and the remaining 30 percent to the test set. The training set contained 298 observations, and the test set contained 127 observations. I then fit some exploratory models. In addition to the tree model fit in the previous section, I decided to fit several models using variable selection algorithms. First I used backward selection to identify which are some of the most important variables in the data set. The top five variables were carat, color grade I, color grade J, color grade K, and color grade L. I fit a naïve model using these variables as predictors and price as the response variable, and then I plotted the model's residuals to evaluate its goodness-of-fit. I noticed that the residuals indicated some problems with the model. They exhibited nonlinearity, deviation from a normal distribution, heteroscedasticity, and high-leverage points. Because of this, I decided to logarithmically transform the response variable and re-run the model with the same predictors. **Figure 6** shows the diagnostic plots for the model using the untransformed response variable in the first row and the plots for the model using the transformed response variable in the second row. While transforming the response variable did not completely fix the issues, it did help improve the tails of the distribution and make the residuals more homoscedastic.

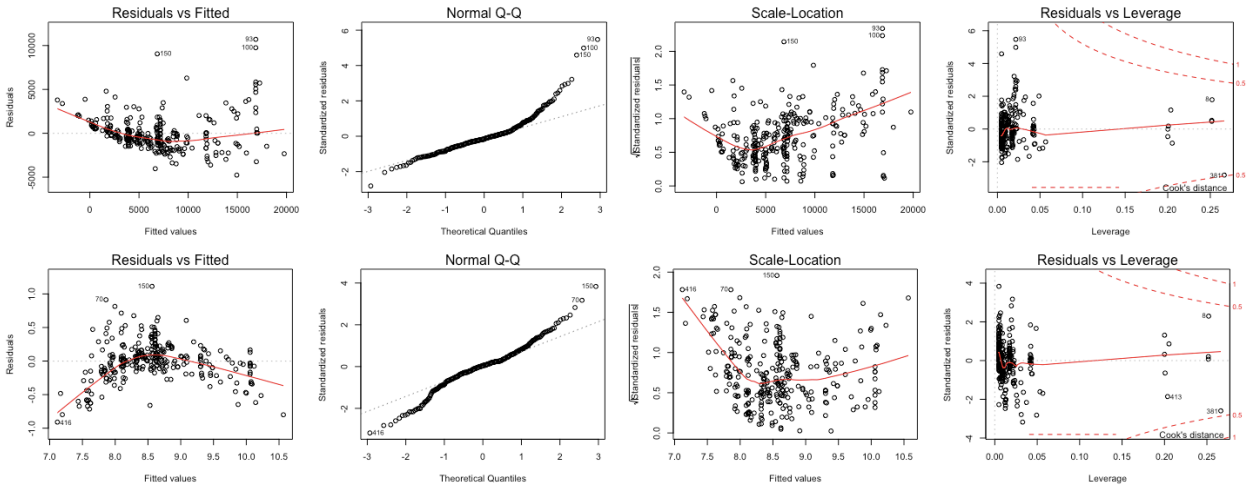

Figure 6: Diagnostic Plots Before and After Transformation of Response Variable

Although the diagnostic plots suggested log(price) was a better response variable to use, I decided to quickly compare the distributions of the response variable before and after the logarithmic transformation to confirm. The histograms in **Figure 7** show that the distribution for log(price) is more normal than the distribution for price.

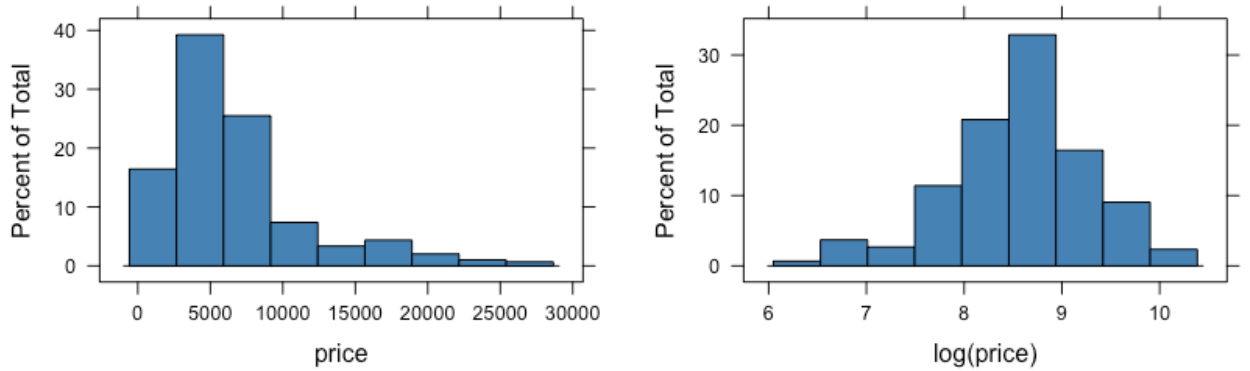

Figure 7: Distributions of price and log(price)

Using log(price) as the response variable, I employed additional variable selection methods for creating more exploratory models. **Table 2** below is a summary of several techniques I used.

Variable Selection Method	Maximized Adjusted R^2		Minimized Mallows' Cp		Minimized BIC	
	Number of Predictors	Adjusted R^2	Number of Predictors	Mallows' Cp	Number of Predictors	BIC
Backward	23	0.8935387	19	15.58155	16	-576.2245
Forward	23	0.8935387	22	16.79064	10	-575.4802
Stepwise	22	0.8934076	19	15.58155	14	-578.4617
All Subsets	23	0.8935387	19	15.58155	14	-578.4617

Table 2: Summary of Variable Selection Methods and Metrics

Three of the variable selection techniques chose the same 23-variable model; three chose the same 19-variable; and two chose the same 14-variable model. All of the above variable

selection techniques chose the same top eight predictors in the same order. **Table 3** lists the predictors in order of selection. I noticed that the variable selection techniques chose many of the same predictors in general but sometimes added them in a different order.

Top Eight Predictors
carat
clarityI1
storeGoodmans
colorK
channelMall
colorI
colorL
colorJ

Table 3: Top Eight Predictors

In addition to the variable selection methods detailed above, I also used the LASSO technique. **Figure 8** shows the cross-validation plot. The numbers at the top of the plot represent the number of predictors in each model, while the dotted vertical lines represent λ_{min} and λ_{1se}. Upper and lower standard deviations are shown along the red dotted cross-validation curve.

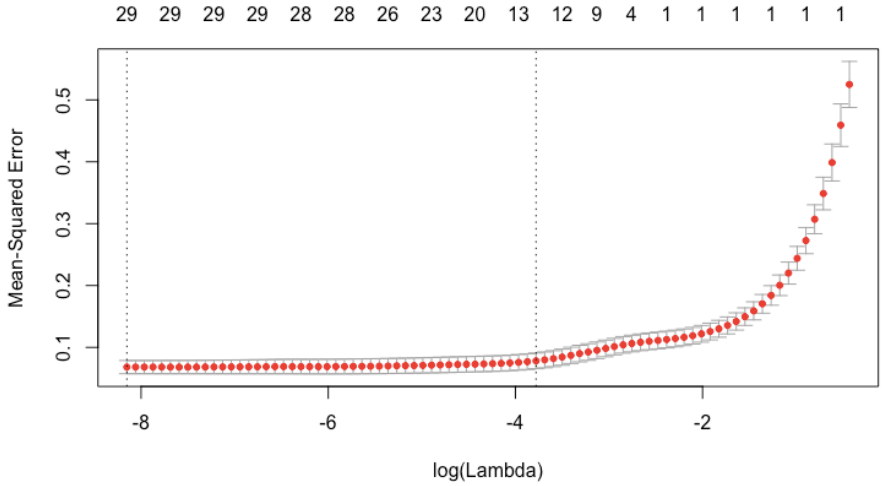

Figure 8: LASSO Cross-Validation Plot

To estimate the coefficients and calculate the test errors, I used lambda = 0.000288169 since it represented the minimum mean cross-validated error. **Table 4** shows the coefficient estimates for the LASSO model.

Variable	Coefficient Estimate	Variable	Coefficient Estimate
(Intercept)	7.0387781366	clarityI1	-0.4335002989
carat	1.5162323589	clarityI2	.
colorE	0.1040497083	cutNot_Ideal	-0.0335828452
colorF	.	channelInternet	-0.0672730671
colorG	-0.0136219643	channelMall	0.2073212723
colorH	.	storeAusmans	0.0752428898
colorI	-0.1660226304	storeBlue_Nile	0.0133888707
colorJ	-0.1952035510	storeChalmers	.
colorK	-0.6944737899	storeDanford	.
colorL	-0.4485544071	storeFred_Meyer	.
clarityVVS1	.	storeGoodmans	0.4775110035
clarityVVS2	0.1116691830	storeKay	0.0236379350
clarityVS1	.	storeR_Holland	0.0049069753
clarityVS2	.	storeRiddles	-0.0185529481
claritySI1	-0.0003815779	storeUniversity	-0.0400246955
claritySI2	-0.0311316946	storeZales	.

Table 4: LASSO Model Coefficient Estimates

This variable selection technique chose a 21-variable model as the best model. I noticed that the LASSO technique included more types of stores than any of the other techniques. For comparison, the 21-variable stepwise selection model included more grades of clarity and color instead of more stores. In fact, the only store the 21-variable stepwise model included was Ausmans. I decided to fit the 21-variable stepwise selection model to see how its Root Mean Squared Error (RMSE) compared to the RMSE of the LASSO model. Below is the linear model I fit to the training set.

```
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")
```

When I calculated the RMSE for each the LASSO and 21-variable stepwise selection model, I discovered that the LASSO model had a value of 2416.057 while the stepwise model had a value of 2515.701. In terms of RMSE, the LASSO model performed better.

Models and Analysis

For this next section, I will discuss the various models I created to predict the price of diamonds. Because this is a regression problem, not all modeling techniques are appropriate. For instance, principal components analysis is not an appropriate modeling technique for this data set. The models discussed below are all suited to regression data. I used log(price) as the response variable for each model.

Model 1: Linear Regression Model with No Interactions

After conducting model-based EDA, I decided to fit a 14-variable linear regression model based on the predictors selected by the stepwise and all subsets algorithms. I chose to fit the 14-variable model for several reasons: (1) it was one of the least complex best models chosen by any of the variable selection techniques; (2) the predictors consistently appeared in most of the more complex models of the various selection methods; and (3) at least two variable selection methods deemed it the best 14-variable model. **Table 5** below shows a summary of the model.

Variable	Estimate	Std. Error	t value	Pr(>|t|)	Significance
(Intercept)	7.07838	0.04572	154.8210	< 2e-16	***
carat	1.55157	0.03391	45.7560	< 2e-16	***
channel == "Mall"	0.40418	0.05183	7.7980	1.217E-13	***
clarity == "I1"	-0.59335	0.08241	-7.2000	5.4365E-12	***
clarity == "SI2"	-0.10721	0.04230	-2.5340	0.011811	*
clarity == "VVS2"	0.16525	0.04840	3.4150	0.000732	***
color == "F"	-0.13457	0.04924	-2.7330	0.006669	**
color == "G"	-0.19731	0.04705	-4.1940	3.67563E-05	***
color == "H"	-0.20202	0.04748	-4.2550	2.84381E-05	***
color == "I"	-0.35881	0.04545	-7.8950	6.45E-14	***
color == "J"	-0.42459	0.06047	-7.0210	1.63637E-11	***
color == "K"	-1.05480	0.13203	-7.9890	3.46E-14	***
color == "L"	-0.73275	0.12593	-5.8190	1.60245E-08	***
store == "Ausmans"	0.29572	0.10831	2.7300	0.006727	**
store == "Goodmans"	0.66053	0.09700	6.8100	5.85559E-11	***
Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1					

Table 5: Summary of Best 14-Variable Linear Model

The coefficient for carat, 1.55157, indicates that this variable has the strongest positive relationship with the response variable. On the other had, color grade K has the strongest negative relationship with log(price). It is interesting to see that all of the coefficients for color predictors are negative. I would have thought the higher grade colors would have a positive impact on log(price), not a negative one. This is something worth investigating more later.

Something else worth noting about this model is its adjusted R-squared value, 0.8869. This high value suggests the model fits the data well. I also calculated the variance inflation factor (VIF) for each predictor to see if multicollinearity was an issue in this model. All the VIFs are less than 2, indicating multicollinearity is not an issue in the model.

Model 2: Linear Regression Model with Interactions

The next model I fit was a linear model including interaction terms. I used stepwise variable selection to find the best models using all possible interaction combinations. Below is a linear model I fit using the top eight terms identified from the stepwise algorithm.

```
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")
```

The stepwise variable selection method had identified carat and storeGoodmans as the only non-interaction terms worthy of being included in an eight-variable model. However, in **Table 6** below, we can see other non-interaction terms with coefficient estimates for this model. These were included only because they were part of interaction terms. None of these variables are very significant alone, as indicated by the Significance column. Also, as in the linear model with no interaction terms, carat has the strongest positive relationship with log(price). It also pairs with three other terms in interactions, which reinforces the idea that it is an important predictor.

Variable	Estimate	Std. Error	t value	Pr(>|t|)	Significance
(Intercept)	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	0.467530	
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"	0.149090	0.073842	2.019	0.044437	*
cut == "Not_Ideal"	0.330023	0.079533	4.149	4.42456E-05	***
clarity == "I1":store == "Riddles"	-0.771121	0.171741	-4.490	0.000010419	***
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"	0.349590	0.104609	3.342	0.000945	***
carat:cut == "Not_Ideal"	-0.372166	0.071008	-5.241	3.146E-07	***
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1					

Table 6: Summary of Linear Model with Interaction Terms

I also calculated the adjusted R-squared value for this model. At 0.8939, it is slightly higher than the R-squared value for the linear model without interaction terms. However, when I calculated the VIFs for this model, I noticed that some of them were quite high. They ranged from about 1.25 to 14.5, which suggests that multicollinearity is an issue in this model.

Model 3: Tree Model

During my EDA, I used the entire data set to fit a tree model with price as the response variable to all the predictors. I decided to fit another tree model with all the predictors, but this time I used the training data and log(price) as the response. **Figure 9** shows the tree.

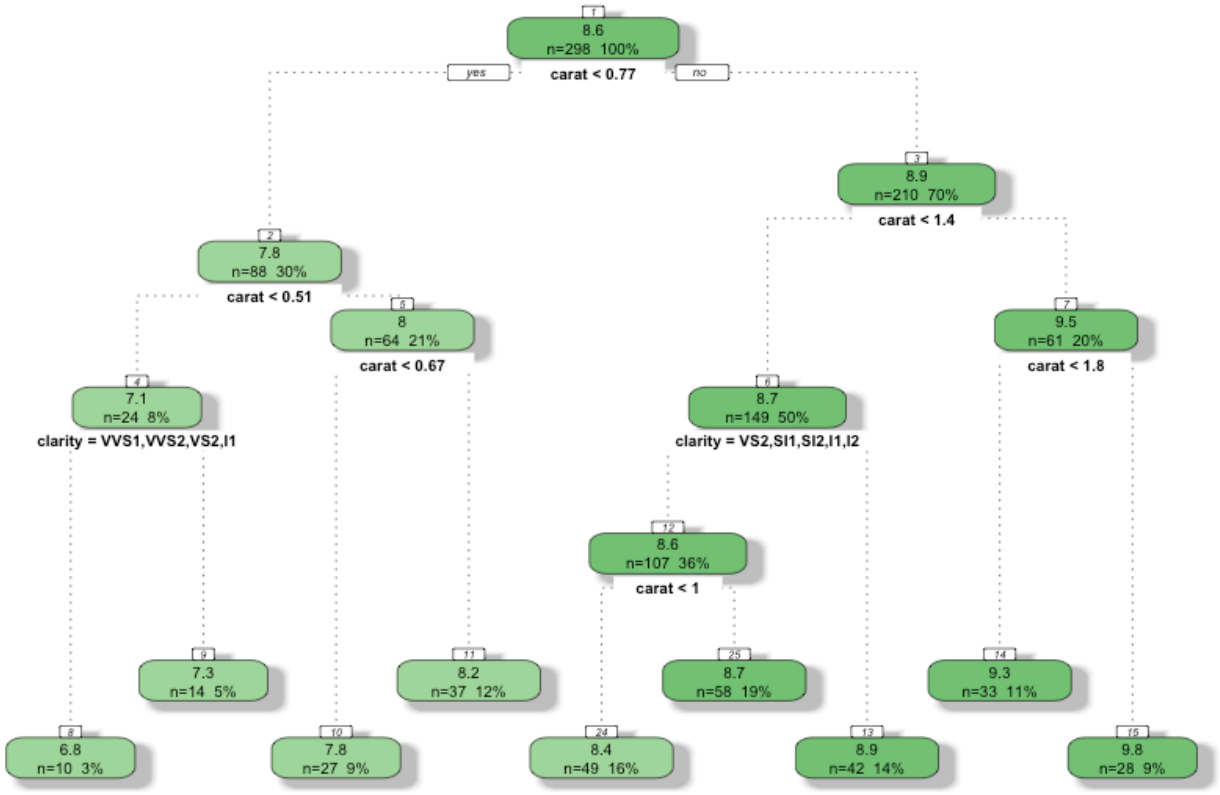

Figure 9: Tree Model of Training Data

Compared to the tree in **Figure 3**, this tree is more complex. However, it is similar in that carat is the most important predictor when it comes to the value of a diamond. It is interesting that the top node leaf splits on carat < 0.77; in the tree in **Figure 3**, the top node leaf split on approximately double the carat value. In this tree, clarity is a deciding factor in the diamond's price when they have greater than 0.77 carats but less than 1.4. As shown in **Figure 9**, diamonds with a higher clarity grade have a predicted log(price) of 8.9, but ones with a lower grade of VS2, SI1, SI2, I1, and I2 have a lower price that is further determined by the carat. It is also interesting to note that diamonds with less than 0.51 carats split on clarity = VVS1, VVS2, VS2, I1. Grade I1 is the second lowest grade of clarity in this data set, yet it is part of a decision rule involving some of the highest grades of clarity. Also, ironically, diamonds that are less than 0.51 carats and have a higher clarity grade of VVS1, VVS2, and VS2 have a lower predicted log(price) than some diamonds of a similar size and a worse grade. Perhaps a higher carat offsets the value of the lower grade diamonds, but this is impossible to know from looking at the tree alone. Examining carat and clarity relationships more in-depth might lead to more insight.

Model 4: Random Forest Model

The last model I fit on this data set was a Random Forest model. For this model, I used a 10-fold repeated cross-validation based on minimizing RMSE in order to determine the number of variables randomly sampled at each split. The cross-validation determined that considering 18 variables at each split was optimal for achieving the lowest RMSE. **Figure 10** below shows the RMSE values for each of the folds considered during the cross-validation.

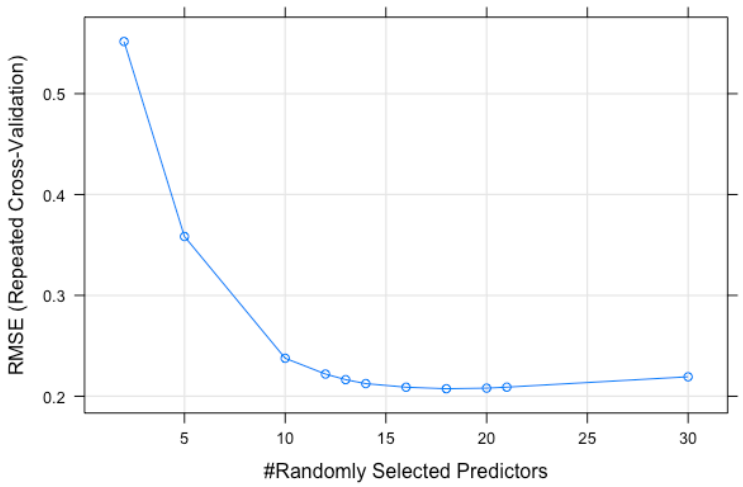

Figure 10: Random Forest Cross-Validation Plot

After learning the optimal number of variables to consider at each split, I fit my model. The model resulted in 91.46 percent of the variance explained using 500 trees. **Figure 11** shows two plots depicting variable importance. For the plot of the left, carat is the most important predictor since it has the highest mean decrease accuracy. Color grade J is next most important, but it has a significantly smaller mean decrease accuracy. For the plot of the right, carat again is the most important variable as it causes the largest decrease in node impurities when a node is split on carat. These plots are not surprising given the role carat has played throughout this analysis.

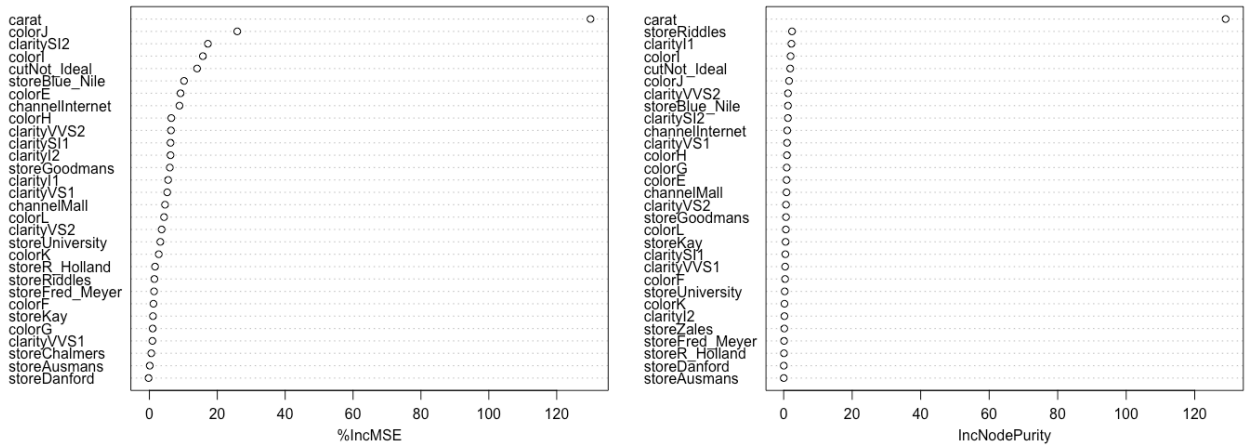

Figure 11: Random Forest Variable Importance Plots

Model Comparison

Once I fit all my models, I calculated a couple errors to evaluate their performance both in-sample and out-of-sample. I decided to use RMSE and Mean Absolute Error (MAE). For ease of interpretability, I computed the exponential functions of the log(price) predictions so we could view the errors in standard US dollars. **Table 7** provides the RMSE and MAE for each model.

Model	Train		Test	
	RMSE	MAE	RMSE	MAE
Linear Model with No Interaction Terms	2616.1390	1263.4960	1790.2020	940.2740
Linear Model with Interaction Terms	2353.9410	1198.3900	1383.5830	955.6749
Tree Model	1859.3170	1150.7920	1478.0380	956.3601
Random Forest Model	830.3552	431.6164	1001.2500	634.5892

Table 7: Summary of Model Performance

Regardless of which metric or data set was used, the Random Forest Model outperformed the other three models. It by far had the lowest RMSE and MAE scores. In terms of RMSE on the test set, the Linear Model with Interaction Terms performed second best, followed by the Tree Model and then the Linear Model with No Interaction Terms. However, in terms of MAE, the order of how the models performed on the test set completely changes. The order becomes the Linear Model with No Interaction Terms in second place, the Linear Model with Interaction Terms in third place, and the Tree Model in last place. For the training set, the order of performance following the Random Forest Model is the Tree Model, the Linear Model with Interaction Terms, and then the Linear Model with No Interaction Terms, regardless of RMSE or MAE. With the exception of the Random Forest Model, the models performed better out-of-sample than in-sample. I was expecting the opposite outcome, so reviewing my models and setting different seeds is a good way to make sure my models are built well. In the meantime, I would feel most confident deploying the Random Forest Model to predict diamond prices.

Conclusion

This report demonstrated various regression modeling techniques for predicting the price of diamonds based on their physical characteristics, as well as where they are sold. After assessing the quality of the data, I explored interesting relationships in the data by creating plots and fitting naïve models using several variable selection techniques. I then fit four final models: a linear model with no interaction terms, a linear model with interaction terms, a tree model, and a Random Forest model. The Random Forest model performed the best. Before deploying any model, I would like to check each model for robustness by setting different seeds when creating the training and test data sets.

Appendix A: Relevant R Code

```
############################################################
# Load Data
############################################################

# Read data
data <-
read.csv(file.path("/Users/annie/Desktop/Northwestern/PREDICT_454/Assignment_02","two_m
onths_salary.csv"), sep=",",header=TRUE)

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

# Explore the data -- how big is it, what types of variables included, distributions and missing
values.
class(data) # data.frame
dim(data) # 425   7
str(data) # all int or numeric
summary(data) # no missing data

# Convert to numeric
data$price <- as.numeric(data$price)

# Convert to factor
data$color <- as.factor(data$color)
data$clarity <- as.factor(data$clarity)

# Code factor levels (per two_months_salary.pdf)
levels(data$color) <- c("D", "E", "F", "G", "H", "I", "J", "K", "L")
levels(data$clarity) <- c("IF", "VVS1", "VVS2", "VS1", "VS2", "SI1", "SI2", "I1", "I2")

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

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

# Create box plots of specific predictors
colorb <- bwplot(price~color, data = data,
                 par.settings = list(
                   box.umbrella=list(col= "black"),
                   box.dot=list(col= "black"),
                   box.rectangle = list(col= "black", fill = "steelblue")),
                 strip = strip.custom(bg="lightgrey"),
```

```
                xlab = "color")
colorb

clarityb <- bwplot(price~clarity, data = data,
           par.settings = list(
             box.umbrella=list(col= "black"),
             box.dot=list(col= "black"),
             box.rectangle = list(col= "black", fill = "steelblue")),
           strip = strip.custom(bg="lightgrey"),
           xlab = "clarity")
clarityb

cutb <- bwplot(price~cut, data = data,
                par.settings = list(
                  box.umbrella=list(col= "black"),
                  box.dot=list(col= "black"),
                  box.rectangle = list(col= "black", fill = "steelblue")),
                strip = strip.custom(bg="lightgrey"),
             xlab = "cut")
cutb

channelb <- bwplot(price~channel, data = data,
                par.settings = list(
                  box.umbrella=list(col= "black"),
                  box.dot=list(col= "black"),
                  box.rectangle = list(col= "black", fill = "steelblue")),
                strip = strip.custom(bg="lightgrey"),
                xlab = "channel")
channelb

grid.arrange(clarityb, cutb, channelb, ncol=3)

############################################################
# EDA
############################################################

# Create tree plot -- How to plot without scientific notation?
fancyRpartPlot(rpart(price ~ ., data = data), sub = "")

# Create scatter plot for price~carat
xyplot(price~carat, data = data, col = "steelblue")

# Calculate correlation between price and carat
cor(data$price, data$carat) # 0.8796315

# Plot carat
ch <- histogram(~carat, data = data,
              col = "steelblue", strip = strip.custom(bg="lightgrey"))
ch
cb <- bwplot(~carat, data = data,
           par.settings = list(
```

```
            box.umbrella=list(col= "black"),
            box.dot=list(col= "black"),
            box.rectangle = list(col= "black", fill = "steelblue")),
          strip = strip.custom(bg="lightgrey"))
cb

cq <- qqmath(~carat, data = data, col = "steelblue")
cq

grid.arrange(ch, cb, cq, ncol=3)

############################################################
# Model Build -- Exploration Part 1
############################################################

# Create train/test sets (70/30)
dim(data) # 425: 0.70*425 = 297.5 (train should have 297 or 298 observations)

set.seed(123)
train <- sample_frac(data, 0.70)
train_id <- as.numeric(rownames(train))
test <- data[-train_id,]

head(train)
dim(train) # 298   7
head(test)
dim(test) # 127   7

# Fit a naïve regression model using backwards variable selection (nvmax = 29 to capture all
factor levels)
model.1.bwd <- regsubsets(price ~ ., data = train, nvmax = 29, method="backward")
summary(model.1.bwd) # the first level of each factorized variable doesn't appear in the output

# Create naïve model using top 5 predictors
m1 <- lm(price ~ carat + (color=="I") + (color=="J") + (color=="K") + (color=="L"), data = train)
par(mfrow=c(1,4))
plot(m1)
par(mfrow=c(1,1))
AIC(m1) # 5376.918
vif(m1) # all close to 1 - GOOD

# Should you make transformations to the response variable or some predictor variables?
par(mfrow=c(2,2))
hist(data$price) # right-skewed
hist(train$price) # right-skewed
hist(data$carat)
hist(train$carat)
# The distributions appear similar regardless of full or training data set

# Transform price
par(mfrow=c(1,1))
```

Andrea Bruckner
Predict 454 Sec 55

```
log_price_d <- log(data$price)
hist(log_price_d)
log_price <- log(train$price)
hist(log_price)

# Plot price and log_price
ph <- histogram(~price, data = train,
            col = "steelblue", strip = strip.custom(bg="lightgrey"))
ph_log <- histogram(~log(price), data = train,
            col = "steelblue", strip = strip.custom(bg="lightgrey"))

grid.arrange(ph, ph_log, ncol=2)

# Create log naïve model using top 5 predictors
m1_log <- lm(log(price) ~ carat + (color=="I") + (color=="J") + (color=="K") + (color=="L"), data =
train)
par(mfrow=c(1,4))
plot(m1_log)
par(mfrow=c(1,1))
AIC(m1_log) # 117.9585 -- can't compare AICs between log and nonlog models
vif(m1_log) # same as m1

# Compare residual plots of m1 and m1_log -- the log transformation makes it a little better?
par(mfrow=c(2,4))
plot(m1)
plot(m1_log)
par(mfrow=c(1,1))

# Log backward variable selection (nvmax = 29 to capture all factor levels)
model.2.bwd <- regsubsets(log(price) ~ ., data = train, nvmax = 29, method="backward")
summary(model.2.bwd) # the first level of each factorized variable doesn't appear in the output
# store still produces 2 linear dependencies found warning
# The top 5 variables change

reg.summary.bwd.2 <- summary(model.2.bwd)

which.max(reg.summary.bwd.2$adjr2) # 23
which.min(reg.summary.bwd.2$cp) # 19
which.min(reg.summary.bwd.2$bic) # 16
reg.summary.bwd.2$adjr2[23] # 0.8935387 # worse
reg.summary.bwd.2$cp[19] # 15.58155 # better
reg.summary.bwd.2$bic[16] # -576.2245 # worse

############################################################
# Model Build -- Exploration Part 2 Using log(price)
############################################################

# Create tree plot - top number in each leaf shows log(price)
fancyRpartPlot(rpart(log(price) ~ ., data = train), sub = "")
# Backward Selection (copied from above section)
set.seed(123)
```

```
model.2.bwd <- regsubsets(log(price) ~ ., data = train, nvmax = 29, method="backward")
summary(model.2.bwd) # the first level of each factorized variable doesn't appear in the output

reg.summary.bwd.2 <- summary(model.2.bwd)

which.max(reg.summary.bwd.2$adjr2) # 23
which.min(reg.summary.bwd.2$cp) # 19
which.min(reg.summary.bwd.2$bic) # 16
reg.summary.bwd.2$adjr2[23] # 0.8935387 # worse
reg.summary.bwd.2$cp[19] # 15.58155 # better
reg.summary.bwd.2$bic[16] # -576.2245 # worse

# Forward Selection
set.seed(123)
model.fwd <- regsubsets(log(price) ~ ., data = train, nvmax = 29, method="forward")
summary(model.fwd)

reg.summary.fwd <- summary(model.fwd)

which.max(reg.summary.fwd$adjr2) # 23
which.min(reg.summary.fwd$cp) # 22
which.min(reg.summary.fwd$bic) # 10
reg.summary.fwd$adjr2[23] # 0.8935387
reg.summary.fwd$cp[22] # 16.79064
reg.summary.fwd$bic[10] # -575.4802

# Stepwise Selection
set.seed(123)
model.stepwise <- regsubsets(log(price) ~ ., data = train, nvmax = 26, method="seqrep") #
nvmax > 26 causes R Studio session to crash
summary(model.stepwise)

reg.summary.stepwise <- summary(model.stepwise)

which.max(reg.summary.stepwise$adjr2) # 22
which.min(reg.summary.stepwise$cp) # 19
which.min(reg.summary.stepwise$bic) # 14
reg.summary.stepwise$adjr2[22] # 0.8934076
reg.summary.stepwise$cp[19] # 15.58155
reg.summary.stepwise$bic[14] # -578.4617

# All Subsets Selection
set.seed(123)
model.allsub <- regsubsets(log(price) ~ ., data = train, nvmax = 29, method="exhaustive")
summary(model.allsub

reg.summary.allsub <- summary(model.allsub)
names(reg.summary.allsub)

which.max(reg.summary.allsub$adjr2) # 23
which.min(reg.summary.allsub$cp) # 19
```

```
which.min(reg.summary.allsub$bic) # 14
reg.summary.allsub$adjr2[23] # 0.8935387
reg.summary.allsub$cp[19] # 15.58155
reg.summary.allsub$bic[14] # -578.4617

# LASSO Model
# Set up grid and data matrix for lasso model
grid <- 10^seq(10, -2, length=100)
set.seed(123)
train.matrix  <- model.matrix(log(price) ~ ., data=train)[,-1]

log_price <- log(train$price)
set.seed(123)
model.lasso <- glmnet(train.matrix, log_price, alpha=1, lambda=grid) # need to use log_price
instead of log(train$price) for this function

# Use cross-validation to select lambda.
set.seed(123)
cv.out.lasso <- cv.glmnet(train.matrix, log_price, alpha=1)
plot(cv.out.lasso) # 12 predictors ideal?

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

coef(model.lasso, s=bestlamlasso)

# Calculate lasso train predictions
set.seed(123)
lasso.train <- predict(model.lasso, newx = train.matrix, s = bestlamlasso)
lasso.train.exp <- exp(lasso.train) # returns log(price) to price for interpretability
head(lasso.train.exp)

# Calculate lasso train errors
actual <- train$price
predicted <- lasso.train.exp
error <- actual - predicted
lasso.train.rmse <- rmse(error)
lasso.train.rmse # 2416.057

# 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 = train)

coef(m_stepwise21)
AIC(m_stepwise21) # 87.73165
vif(m_stepwise21) # all under 5
```

```
# Calculate stepwise21 train predictions
set.seed(123)
stepwise21.train <- predict(m_stepwise21, newdata = train)
stepwise21.train.exp <- exp(stepwise21.train) # returns log(price) to price for interpretability

# Calculate stepwise21 train errors
actual <- train$price
predicted <- stepwise21.train.exp
error <- actual - predicted
stepwise21.train.rmse <- rmse(error)
stepwise21.train.rmse # 2515.701

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

# Although Backward, Stepwise, and All Subset Selection chose the same 19 variable model,
# I will use the 14 variable model that both Stepwise and All Subset chose since it is simpler.

# (1) Create a linear regression model with no interactions using the lm() function
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 = train)

summary(fit1) # Adjusted R-squared:  0.8869
vif(fit1) # all less than 2

# (2) Create a linear regression model including some interaction terms

# Stepwise Selection with Interaction (All Subset Selection has too long of processing time)
set.seed(123)
model.stepwise.I <- regsubsets(log(price) ~ .*., data = train, method="seqrep") # use all possible
interactions
summary(model.stepwise.I)

# In order of stepwise variable selection
# carat
# clarityI1:storeRiddles
# colorI:channelInternet
# carat:colorK
# carat:colorJ
# storeGoodmans
# clarityVS2:channelMall
# carat:cutNot_Ideal

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 = train)

summary(fit2) # Adjusted R-squared:  0.8939
vif(fit2) # range from 1.25 to 14.5

# (3) Create a tree model
set.seed(123)
fit3 <- rpart(log(price) ~ ., data = train)
fit3
fit3_plot <- fancyRpartPlot(rpart(log(price) ~ ., data = train), sub = "")
fit3_plot

# (4) Create a Random Forest model
set.seed(123)
# mtry can be 31 at most because of number of columns in train.matrix

#http://machinelearningmastery.com/tune-machine-learning-algorithms-in-r/
# Random Search
control <- trainControl(method="repeatedcv", number=10, repeats=3, search="random")
set.seed(123)
mtry <- sqrt(ncol(train.matrix)) # 5.567764
rf_random <- train(train.matrix, log_price, method="rf", metric="RMSE", tuneLength=15,
trControl=control)
print(rf_random)

plot(rf_random)

set.seed(123)
fit4 <- randomForest(log(train$price) ~ ., data = train.matrix, mtry=18, importance=TRUE)
fit4

varImpPlot(fit4, main = "Random Forest Model: \n Variable Importance") # How to do in Lattice?

#########################################################
# Model Comparison
#########################################################

# Prediction Functions
# https://heuristically.wordpress.com/2013/07/12/calculate-rmse-and-mae-in-r-and-sas/
rmse <- function(error){
  sqrt(mean(error^2))
}

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

Andrea Bruckner
Predict 454 Sec 55

```
#######
# fit1
#######

# Calculate fit1 test predictions
set.seed(123)
fit1.pred <- predict(fit1, newdata = test)
fit1.pred.exp <- exp(fit1.pred) # returns log(price) to price for interpretability
head(fit1.pred.exp)

# Calculate fit1 test errors
actual <- test$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 = train)
fit1.train.exp <- exp(fit1.train) # returns log(price) to price for interpretability
head(fit1.train.exp)

# Calculate fit1 train errors
actual <- train$price
predicted <- fit1.train.exp
error <- actual - predicted
fit1.train.rmse <- rmse(error)
fit1.train.rmse # 2616.139
mae(error) # 1263.496

#######
# fit2
#######

# Calculate fit2 test predictions
set.seed(123)
fit2.pred <- predict(fit2, newdata = test)
fit2.pred.exp <- exp(fit2.pred) # returns log(price) to price for interpretability

# Calculate fit2 test errors
actual <- test$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 = train)
fit2.train.exp <- exp(fit2.train) # returns log(price) to price for interpretability
head(fit2.train.exp)

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


#######
# fit3
#######

# Calculate fit3 test predictions
set.seed(123)
fit3.pred <- predict(fit3, newdata = test)
fit3.pred.exp <- exp(fit3.pred) # returns log(price) to price for interpretability

# Calculate fit3 test errors
actual <- test$price
predicted <- fit3.pred.exp
error <- actual - predicted
fit3.pred.rmse <- rmse(error)
fit3.pred.rmse # 1478.038
mae(error) # 956.3601

# Calculate fit3 train predictions
set.seed(123)
fit3.train <- predict(fit3, newdata = train)
fit3.train.exp <- exp(fit3.train) # returns log(price) to price for interpretability
head(fit3.train.exp)

# Calculate fit3 train errors
actual <- train$price
predicted <- fit3.train.exp
error <- actual - predicted
fit3.train.rmse <- rmse(error)
fit3.train.rmse # 1859.317
mae(error) # 1150.792

#######
# fit4
#######

# For fit4 test, must create test.matrix
set.seed(123)
test.matrix <- model.matrix(log(price) ~ ., data=test)[,-1]
```

```
# Calculate fit4 test predictions
set.seed(123)
fit4.pred <- predict(fit4, newdata = test.matrix)
fit4.pred.exp <- exp(fit4.pred) # returns log_price to price for interpretability
head(fit4.pred.exp)

# Calculate fit4 test errors
actual <- test$price
predicted <- fit4.pred.exp
error <- actual - predicted
fit4.pred.rmse <- rmse(error)
fit4.pred.rmse # 1001.25
mae(error) # 634.5892

# Calculate fit4 train predictions
set.seed(123)
fit4.train <- predict(fit4, newdata = train.matrix)
fit4.train.exp <- exp(fit4.train) # returns log(price) to price for interpretability
head(fit4.train.exp)

# Calculate fit4 train errors
actual <- train$price
predicted <- fit4.train.exp
error <- actual - predicted
fit4.train.rmse <- rmse(error)
fit4.train.rmse # 830.3552
mae(error) # 431.6164
```