Assignment 2

Predict 454, Sec. 55

Matt Hayden

4/15/2017

Introduction

In this assignment, Brian Pope is preparing to propose to his girlfriend and is investigating how to buy an engagement ring. In order to make an informed decision, he collected 7 facts about 425 ring cut diamonds. These facts include the carat, color, clarity, cut, store, store type, and price. We are going to use this information from Brian to model ring prices based on the 6 other facts. This modeling effort will help Brian choose which ring to buy and at which store so he gets the ring his girlfriend wants for the best price.

Additionally, we will consider how the pricing model could help stores determine at what price to buy and sell their diamonds. The stores that Brian researched are starting to show interest in the work and are curious to know how they can benefit from the models.

Data Quality Check

First, it is important to understand whether or not the data we have is trustworthy. We have two continuous variables, Price and Carat (size). There are 3 discrete variables for the Store, Type of Store, and Cut. Finally, Color and Clarity are technically discrete variables, but because they are a part of ordered scales, we will consider them as continuous variables in the quality check and later in the modeling.

An important fact to keep in mind is that several variables are subjective. The diamond cut is described as Ideal or Not Ideal, but that is up to the whims of one person. Also Color and Clarity can often be approximate. So for analysis this should be kept in mind.

Figure 1 - Description of Variables

Name	Type	Description
carat	Numeric	Size - 1 carat = 200milligrams
color	Numeric	Color on GIA scale (lower is better)
clarity	Numeric	Clarity on GIA visibility scale (lower is better)
cut	Factor	Cut - Ideal vs Not Ideal
channel	Factor	Type of Store - Mall, Independent, or Internet
store	Factor	Store Name
price	Numeric	Cost of the diamond

First, we will look at the continous variables. We can assert the following observations:

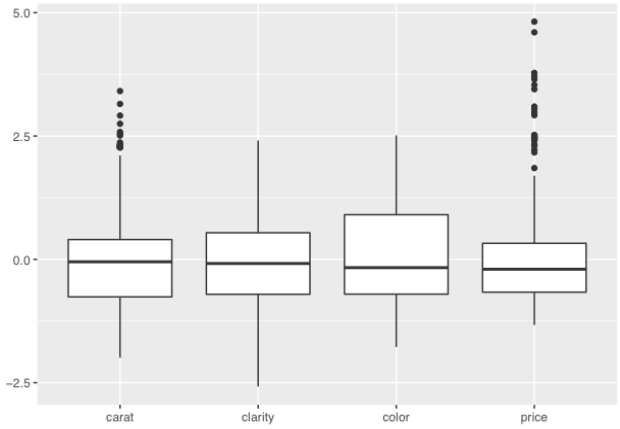
- There are no missing values.
- Color and Clarity are within the bounds of their respective scales. No abnormal values.
- There are some more extreme values for Carat and Price, but this is not unusual.

Figure 2 - Continuous Variables

	Mean	Min	Max	SD	Missing	Largest SD
carat	1.04	0.2	2.48	0.42	0	3.41
color	4.31	1.0	9.00	1.86	0	2.51
clarity	6.13	2.0	10.00	1.60	0	2.58
price	6355.99	497.0	27575.00	4404.24	0	4.82

We can get an overall look at the distributions of the variables by looking at the boxplots for each of the variables. The boxplots below show that the distributions are reasonable, with right skews for Carat and Price. These will be good candidates for transforms.

Figure 3 - Boxplots of Variables



Next, we take a look at the discrete variables. The important observation here is that the observations for the Internet Channel are 5x or 6x more frequent than the Mall and Independent Channels. This shows that the Internet Channel has an opportunity to bias the data since Mall and Independent have small sample sizes. This issue can be alleviated by turning to bagging techniques later for modeling.

Figure 3 - Discrete Variables

Ideal	NotIdeal
154	271

Independent	Internet	Mall
48	318	59

Ashford	Ausmans	BlueNile	Chalmers	Danford	FredMeyer
107	7	211	8	9	15

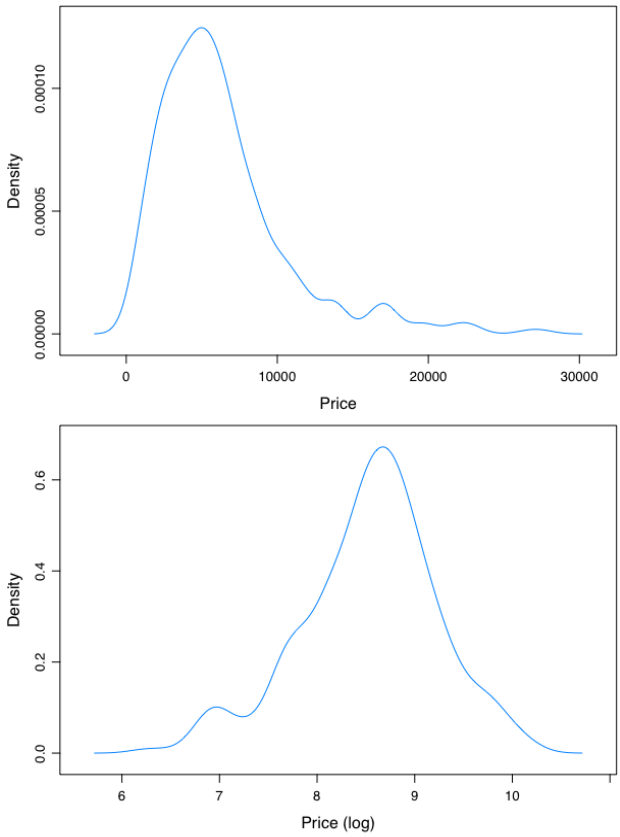
Goodmans	Kay	R.Holland	Riddles	University	Zales
11	14	7	16	13	7

Exploratory Data Analysis

Now that the data is validated, we turn to an exploratory data analysis to understand the nature of the information we are looking at. First, in Figure 4, we are looking to see the distribution of Price information.

These plots show that since Price has a right skew, it benefits from a transformation. A log transformation will likely produce better results in the modeling process.

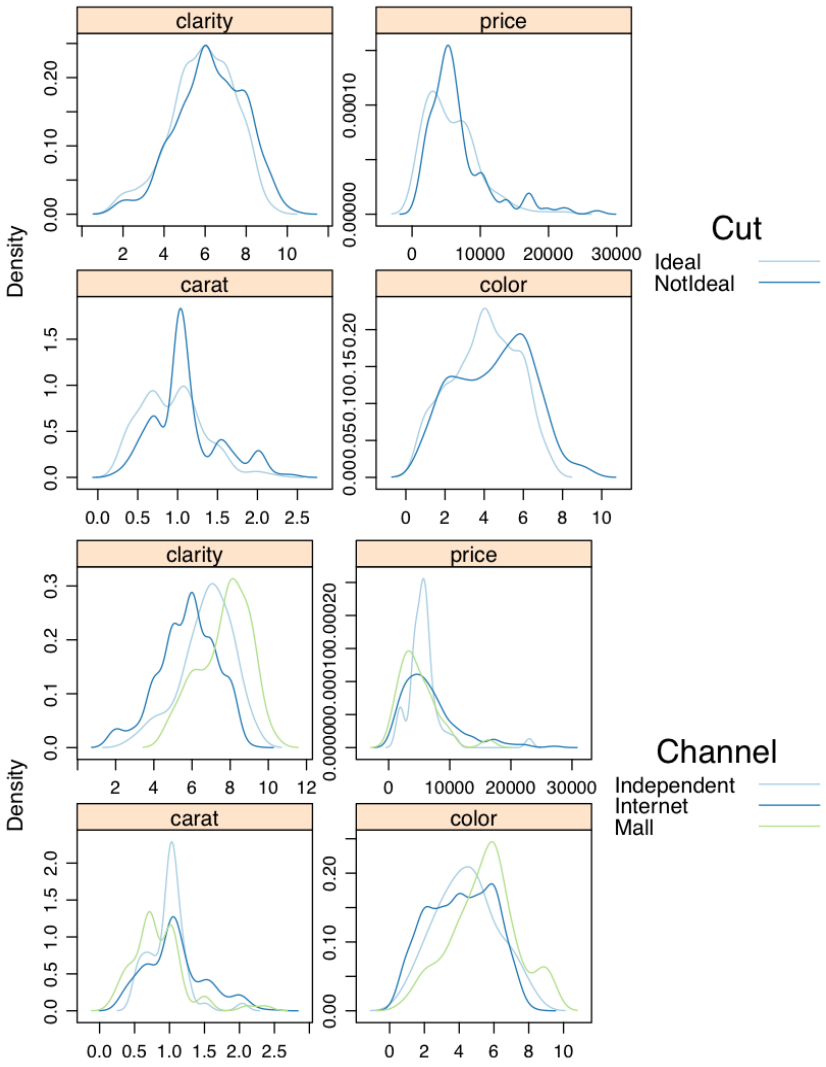
Figure 4 - Variable Density Plots for Price

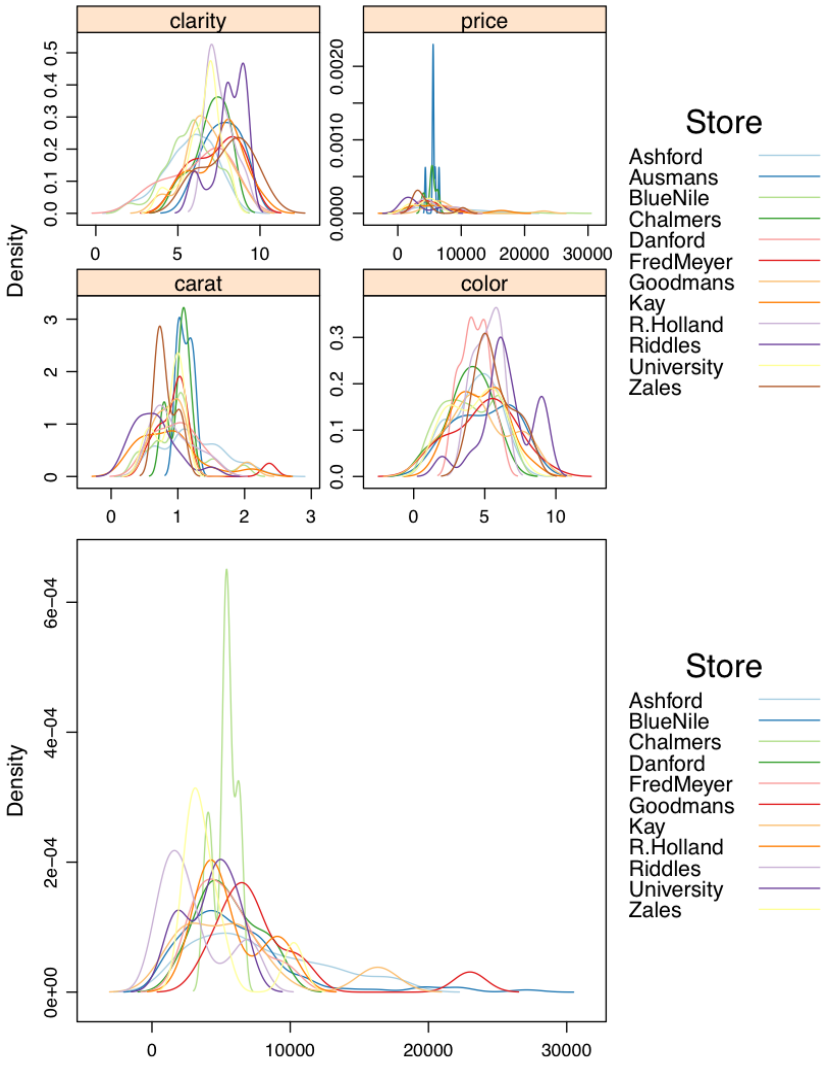


In Figure 5, we take a look at how the factors in discrete variables can impact the continuous varialbes. The important observation in these graphs include:

- Channel has an important impact on the Clarity distribution.
- Independent sellers are more consistent with pricing with a lower standard deviation.
- Ausmans store has an outlier that is worth considering to cull from the dataset.

Figure 5 - Variable Density Plots by Discrete Variables





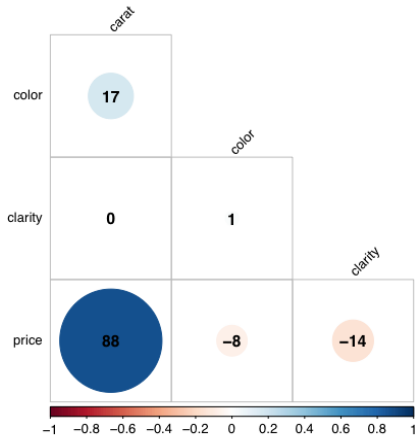
Next, we will look at the correlations between the variables to see what is related. This will help with addressing multi-collinearity issues in an analysis. We see that Carat and Price are highly correlated with a correlation coefficient of 0.88. Carat is definitively the most important predictor in the data set. This one fact does a decent job of predicting price.

It's also interesting to note that as the diamond size grows the color quality gets worse (a higher Color score means more color is in the diamond). This is either because as diamonds get bigger they're less likely to be pure, or because color is easier to observe in a bigger diamond.

We also see that as Price increases, the Clarity improves (a lower Clarity score means better clarity). What's surprising is that it's only mildly correlated with a correlation coefficient of -0.14. This confirms that size is

much more important than anything else when determining price.

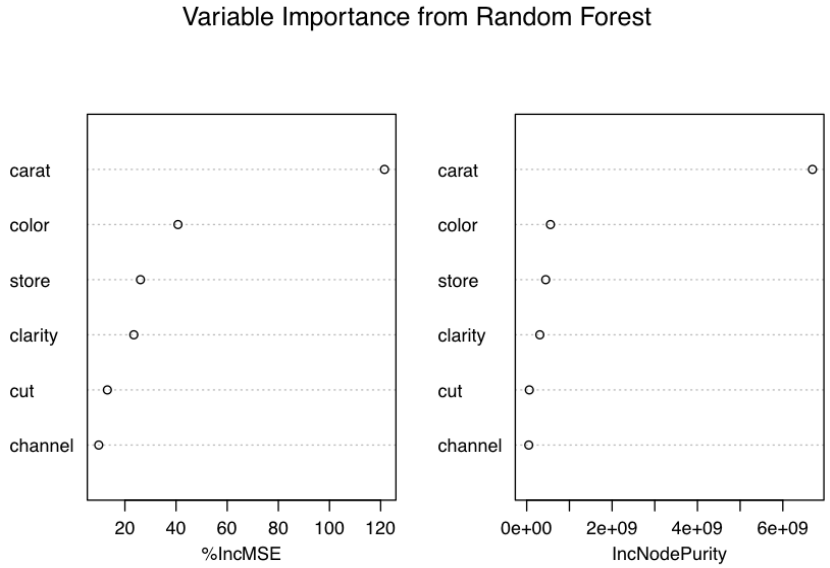
Figure 6 - Variable Correlation Matrix



Model-Based Exploratory Data Analysis

Doing basic model analysis of the diamond dataset can reveal some important truths within the data. By applying a random forest model, we are able to see which variables are more important in determining Price. We see again that Carat (size) is by far the most important predictor.

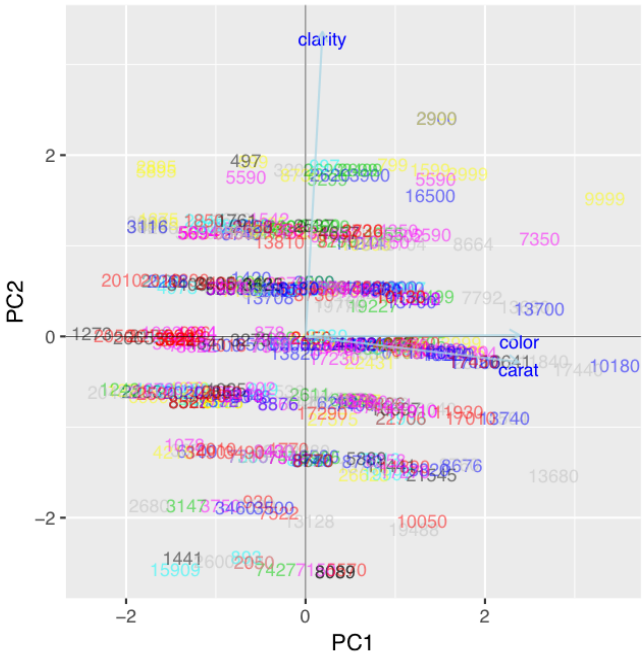
Figure 7



A look at a principal component analysis of the continuous variables shows that all of the continuous variables

are important in their predictive power, even though Carat is much more important. The model based analysis is in general showing the same important facts.

Figure 8 - Principal Component Analysis of PC1 vs PC2



At this point we can start to look at linear regressions to see what it has in store for the initial data analysis. At first we try a forward selection regression model. It identifies Carat, Color, Clarity, ChannelMall, and StoreGoodmans as the important variables, but with a relatively high AIC score of 7400. Thinking back to the distribution of the Price variable, we should run the regression again with a log transform on Price.

Linear Regression for EDA
2153.46 * (Intercept) + 10160.58 * carat + -698.79 * color + -625.13 * clarity + 2702.00 * channelMall + 3939.41 * storeGoodmans

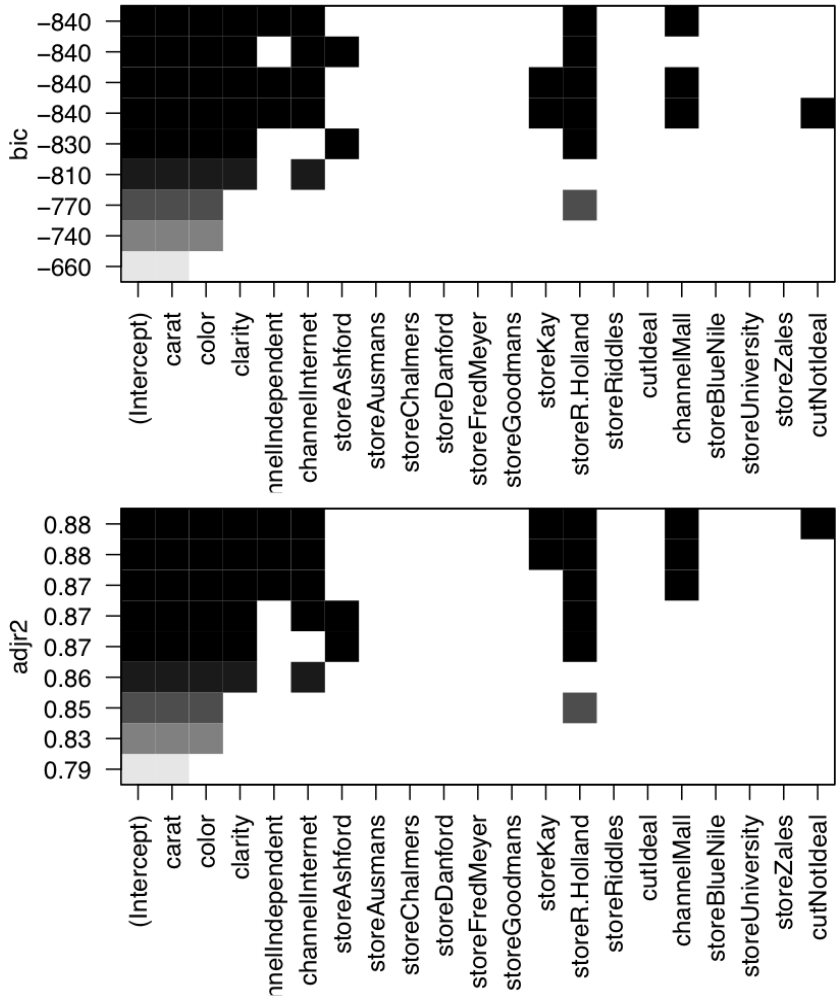
With a log transform on Price, we get a much better result with an AIC score of 46. This is comparison to the non-price-transformed version of the model that had an AIC of 7400. All models going forward will operate on LogPrice.

The linear regression again confirms that Carat is the most influential predictor with a much higher coefficient than the others. While the scales aren't exactly the same for the other variables, they are close enough to observe that once again Carat is the most influential.

Linear Regression for EDA with Price(log)
7.60 * (Intercept) + 1.60 * carat + -0.09 * color + -0.06 * clarity + 0.30 * channelMall + 0.64 * storeGoodmans

Finally for the exploratory data analysis, we will look at the results of best subset regression. With this analysis we look at BIC and Adjusted R Squared scored to determine how many variables to include in a linear regression. The BIC analysis suggests the top 6 variables, while the Adjusted R Squared analysis suggests the top 9 variables.

Figure 9 - Best Subset Selection Number of Variables



When we investigate the two options to optimize for BIC and Adjusted R Squared, we find that the 9 variable model outperforms the 6 variable model with an AIC score of 27 compared with 37. While the 9 variable model is more complex, it is marginally so. So the best model to this point includes these 9 variables seen below.

Linear Regression for EDA with 6 Variables
7.76 * (Intercept) + 1.61 * carat + -0.09 * color + -0.07 * clarity + -0.15 * channelInternet + 0.18 * channelMall + 0.51 * storeGoodmans

Linear Regression for EDA with 9 Variables
8.06 * (Intercept) + 1.62 * carat + -0.09 * color + -0.07 * clarity + -0.28 * channelIndependent + -0.44 * channelInternet + -0.17 * storeFredMeyer + 0.49 * storeGoodmans + -0.23 * storeRiddles + -0.05 * cutNotIdeal

Before we definitively prefer the 9 variable model, it is worth checking for multi-collinearity. We are starting to see multi-collinearity in the Channel variables. Though the highest variance inflation of 3.69 is within acceptable limits. This model should be acceptable for the modeling effort.

	VIF
carat	1.227985
color	1.184348
clarity	1.290050
channelIndependent	2.775882
channelInternet	3.691694
storeFredMeyer	1.499169
storeGoodmans	1.288591
storeRiddles	1.599336
cutNotIdeal	1.158949

Modeling

The exploratory data analysis has pointed us in the direction of the prefered variables to use in regression models. Given this insight we will attempt the following models to identify which ones produce the lowest Root Mean Squared Error (RMSE). This metric will guide us to choosing the best model.

- Linear Model - 5 Variable Model
- Linear Model - 9 Variable Model
- Linear Model - Forward Selection of Interactions
- Linear Model - Best Subset Selection of Interactions
- Boosted Linear Model
- Lasso Model
- Decision Tree
- Random Forest

Linear Model - 5 Variables
7.61 * (Intercept) + 1.56 * carat + -0.09 * color + -0.06 * clarity + 0.28 * channelMall + 0.47 * storeGoodmans

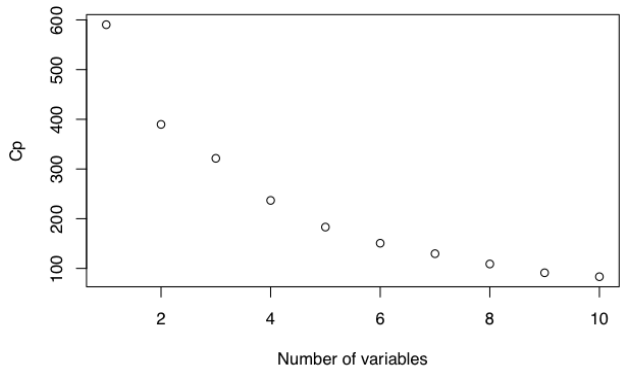
Linear Model - 9 Variables
8.03 * (Intercept) + 1.58 * carat + -0.09 * color + -0.06 * clarity + -0.28 * channelIndependent + -0.42 * channelInternet + -0.19 * storeFredMeyer + 0.35 * storeGoodmans + -0.22 * storeRiddles + -0.04 * cutNotIdeal

Linear Model with Interactions
6.991 * (Intercept) + 1.467 * carat + -0.035 * carat:storeAusmans

By doing a more thorough analysis of interactions with variables, we do find interactions that meaningfully improve the predictive power. The plot shows that the top 9 or 10 of the variables are significant in the model.

Figure 10 - Best Subset Selection Number of Variables with Interactions

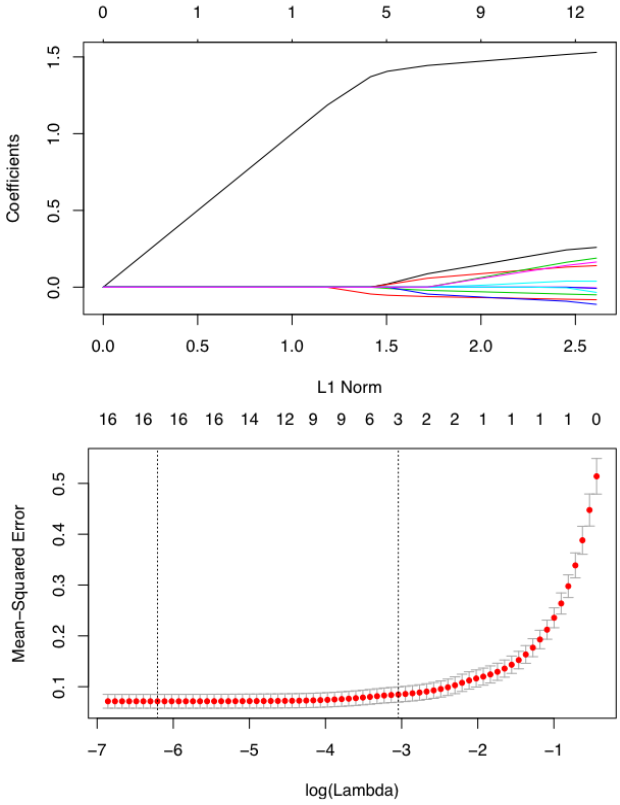
Linear Model with Interactions from Best Subset Selection
6.476 * (Intercept) + 3.018 * carat + 0.080 * clarity + -0.078 * carat:color + -0.165 * carat:clarity + -0.388 * carat:channelInternet + 1.394 * carat:storeRiddles + 0.519 * carat:cutIdeal + -0.167 * color:storeRiddles + 0.037 * clarity:channelInternet + -0.543 * channelInternet:cutIdeal



Boosted Linear Model
-0.95 * (Intercept) + 1.51 * carat + -0.08 * color + -0.04 * clarity + -0.09 * channelInternet + 0.04 * channelMall + 0.23 * storeGoodmans + 0.13 * storeKay + 0.14 * storeR.Holland + 0.13 * storeZales

In the lasso model, by looking at the graphs in Figure 11, we see that most of the value comes in by including 15 of the variables. Though an argument could be made for only 10 variables as well based on where the Cp score is minimized.

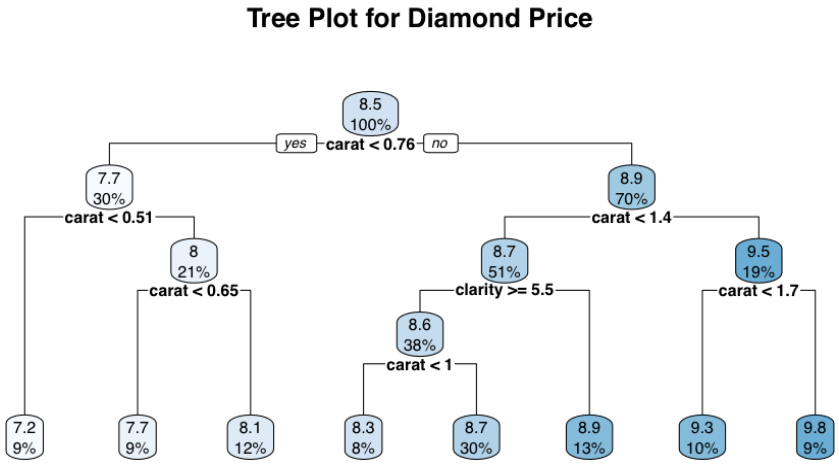
Figure 11 - Lasso Model



Lasso Model
7.652 * (Intercept) + 1.530 * carat + -0.082 * color + -0.050 * clarity + -0.112 * channelInternet + 0.039 * channelMall + -0.005 * storeAshford + 0.259 * storeGoodmans + 0.140 * storeKay + 0.189 * storeR.Holland

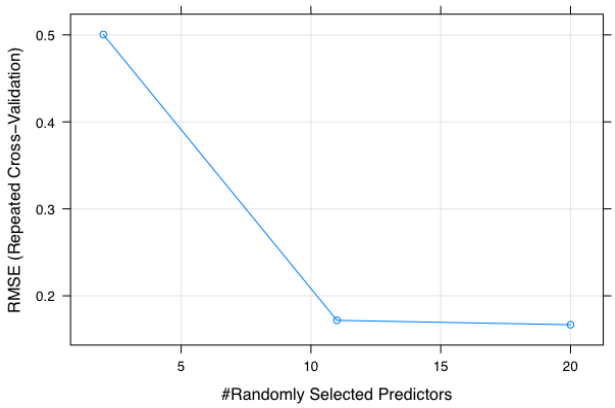
The decison tree plot below showcases again that Carat is the most important predictor, with Clarity helping in some instances.

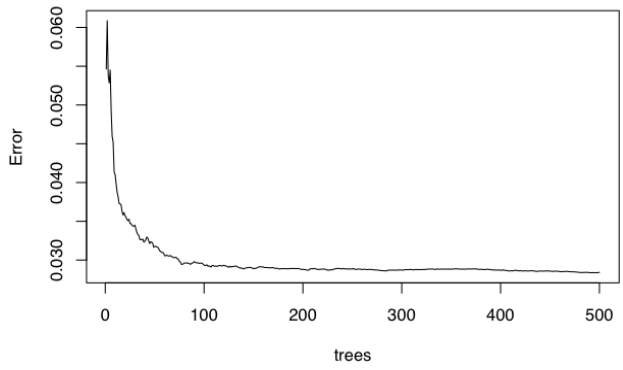
Figure 12 - Decison Tree



The random forest model shows that 11 predictors is the sweet spot for minimizing the RMSE. Also, most of the error is reduced by using around 100 trees in the forest.

Figure 13 - Random Forest





Model Selection and Conclusion

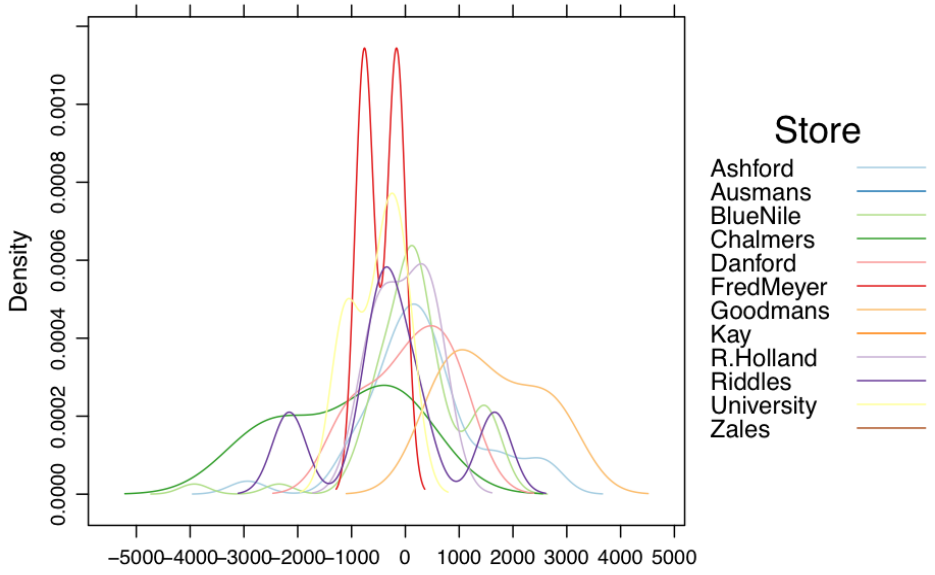
The metric for determining which model works the best in modeling diamond prices is the RMSE. Below we compare all of the models to see that the Random Forest model and Linear Model Best Subset Selection of Interactions both have a RMSE of 0.22. These two models are also the least performant though. The performance is acceptable for this dataset, but with larger datasets, the Linear Model with 9 Variables is a better option if performance ends up being a factor.

Figure 14 - Summary of Model Performance by RMSE of Price (Log)

Model	RMSE of Price(log)
Linear Model - 5 Variables	0.2541
Linear Model - 9 Variables	0.2431
Linear Model - Forward Selection of Interactions	0.3357
Linear Model - Best Subset Selection of Interactions	0.2215
Boosted Linear Model	0.2638
Lasso Model	0.2564
Decision Tree	0.2516
Random Forest	0.2234

Finally, we look at the distribution of how much store prices compare to predicted prices to help Brian Pope choose a store with better values. Based on the test data in the plot below, we see that Goodmans is overpriced. For overall value, Chalmers and Riddles are the stores to buy a ring at.

Figure 15 - Density Plot of Difference of Store Prices to Predicted Prices (Higher Values Mean Overpriced)



Appendix - R Code

```
# {r loaddata, echo=FALSE, warning=FALSE, error=FALSE,
# include=FALSE}
setwd("/Users/haydude/Google Drive/nu - 454 adv model/Assignment 2")
df = read.csv("two_months_salary.csv")

# remove spaces from factor values
df$store = sapply(df$store, function(x) return(gsub(" ", "",
    x)))
df$cut = sapply(df$cut, function(x) return(gsub(" ", "", x)))

# continuous variables
df$carat = as.numeric(df$carat)
df$price = as.numeric(df$price)

# discrete variables, but ranked, so treated like numeric
df$color = as.numeric(df$color)
df$clarity = as.numeric(df$clarity)

# discrete variables
df$cut = as.factor(df$cut)
df$channel = as.factor(df$channel)
df$store = as.factor(df$store)

library(lattice)
library(ggplot2)
library(randomForest)
library(RColorBrewer)
library(knitr)

###### Figure 1 - Description of Variables {r table of variables,
###### echo=FALSE, warning=FALSE}
df.descriptions = c("Size - 1 carat = 200milligrams", "Color on GIA scale (lower is better)",
    "Clarity on GIA visibility scale (lower is better)", "Cut - Ideal vs Not Ideal",
    "Type of Store - Mall, Independent, or Internet", "Store Name",
    "Cost of the diamond")
df.types = c("Numeric", "Numeric", "Numeric", "Factor", "Factor",
    "Factor", "Numeric")

df.summarytable = data.frame(colnames(df), df.types, df.descriptions)
colnames(df.summarytable) = c("Name", "Type", "Description")
library(pander)
temp = df.summarytable
row.names(temp) = seq_along(1:dim(temp)[1])
pander(temp, justify = c("left", "left", "left"))

###### Figure 2 - Continuous Variables {r data quality check
###### continuous, echo=FALSE} missing values
checkmissing = table(is.na(df))  # no missing values
checkmissing = which(is.na(df), arr.ind = TRUE)

idxs.numeric = grep("numeric", sapply(df, class))
```

```
df.means = sapply(df[, idxs.numeric], mean)
df.mins = sapply(df[, idxs.numeric], min)
df.maxs = sapply(df[, idxs.numeric], max)
df.missing = apply(is.na(df[, idxs.numeric]), 2, sum)
df.sds = sapply(df[, idxs.numeric], sd)
df.numbers = data.frame(df.means, df.mins, df.maxs, df.sds, df.missing)
maxs = (df.maxs - df.means)/df.sds
mins = (df.means - df.mins)/df.sds
bigger = apply(data.frame(maxs, mins), 1, max)
df.numbers = data.frame(df.means, df.mins, df.maxs, df.sds, df.missing,
    bigger)
df.numbers.names = c("Mean", "Min", "Max", "SD", "Missing", "Largest SD")
colnames(df.numbers) = df.numbers.names
df.numbers = round(df.numbers, digits = 2)


###### Figure 3 - Boxplots of Variables {r boxplots, echo=FALSE,
###### out.width = '300px', dpi=200}

df.scaled = scale(df[, idxs.numeric])

st = stack(as.data.frame(df.scaled))
ggplot(as.data.frame(st)) + geom_boxplot(aes(x = ind, y = values)) +
    theme(axis.text.x = element_text(angle = 0)) + scale_x_discrete(name = "") +
    scale_y_continuous(name = "")

#

# {r outliers, echo=FALSE, out.width = '500px'}

df.scaled.abs = abs(df.scaled)
df.variable.outliers.indices = which(df.scaled.abs[, -1] >= 3,
    arr.ind = TRUE)
df.variable.outliers.indices[, 2] = apply(df.variable.outliers.indices,
    1, function(x) x[2] = colnames(df.scaled.abs)[x[2] + 1])
df.variable.outliers.table = table(df.variable.outliers.indices[,
    2])
# barchart(df.variable.outliers.table, horizontal = FALSE,
# xlab='', ylab='Frequency', scales = list(x = list(rot =
# 45)))


###### Figure 3 - Discrete Variables

# {r data quality check discrete, echo=FALSE, out.width =
# '500px'}


idxs.factors = grep("factor", sapply(df, class))

tbl = table(df[, 6])
names = names(tbl)
values = tbl[1:6]
```

```
tbl = t(cbind(values))
colnames(tbl) = names[1:6]
rownames(tbl) = NULL

tbl = table(df[, 6])
names = names(tbl)
values = tbl[7:12]
tbl = t(cbind(values))
colnames(tbl) = names[7:12]
rownames(tbl) = NULL


#

### Exploratory Data Analysis

###### Figure 4 - Variable Density Plots for Price {r density
###### price, echo=FALSE, out.width = '300px'}

colors = brewer.pal(12, "Paired")
plot.settings = list(superpose.line = list(col = colors, border = "transparent"))

# log price gives a much more normal distribution
density.plots1 = densityplot(~price, data = df, plot.points = FALSE,
    auto.key = list(space = "right", title = "Cut"), par.settings = plot.settings,
    scales = list(x = "free", y = "free"), xlab = "Price")
density.plots2 = densityplot(~log(price), data = df, plot.points = FALSE,
    auto.key = list(space = "right", title = "Cut"), par.settings = plot.settings,
    scales = list(x = "free", y = "free"), xlab = "Price (log)")
par(mfrow = c(1, 2))
plot(density.plots1)
plot(density.plots2)
par(mfrow = c(1, 1))


###### Figure 5 - Variable Density Plots by Discrete Variables {r
###### density discrete, echo=FALSE, out.width='400px'}

colors = brewer.pal(12, "Paired")
plot.settings = list(superpose.line = list(col = colors, border = "transparent"))

density.plots = densityplot(~carat + color + clarity + price,
    data = df, groups = cut, plot.points = FALSE, auto.key = list(space = "right",
        title = "Cut"), par.settings = plot.settings, scales = list(x = "free",
        y = "free"), xlab = "")

plot(density.plots)

density.plots = densityplot(~carat + color + clarity + price,
    data = df, groups = channel, plot.points = FALSE, auto.key = list(space = "right",
        title = "Channel"), par.settings = plot.settings, scales = list(x = "free",
        y = "free"), xlab = "")
```

```r
plot(density.plots)

density.plots = densityplot(~carat + color + clarity + price,
    data = df, groups = store, plot.points = FALSE, auto.key = list(space = "right",
        title = "Store"), par.settings = plot.settings, scales = list(x = "free",
        y = "free"), xlab = "")

plot(density.plots)



temp = df[df$store != "Ausmans", ]
temp$store = as.character(temp$store)
temp$store = as.factor(temp$store)
density.plots = densityplot(~price, data = temp, groups = store,
    plot.points = FALSE, auto.key = list(space = "right", title = "Store"),
    par.settings = plot.settings, scales = list(x = "free", y = "free"),
    xlab = "")

plot(density.plots)


###### Figure 6 - Variable Correlation Matrix {r correlations v2,
###### echo=FALSE, out.width='200px'}
library(corrplot)
corrplot(cor(df[, idxs.numeric]), tl.col = "black", tl.cex = 0.8,
    tl.srt = 45, type = "lower", addCoefasPercent = TRUE, addCoef.col = TRUE,
    diag = FALSE)
#

### Model-Based Exploratory Data Analysis

###### Figure 7 {r random forest, echo=FALSE, out.width='400px'}
df.rf = randomForest(price ~ ., data = df, mtry = 4, importance = TRUE)
varImpPlot(df.rf, main = "Variable Importance from Random Forest",
    cex = 0.8)



###### Figure 8 - Principal Component Analysis of PC1 vs PC2 {r
###### pca pre, echo=FALSE, warning=FALSE,error=FALSE,
###### include=FALSE}
temp = df[, idxs.numeric]
idx = grep("price", colnames(temp))
temp = temp[, -idx]
df.pcr = prcomp(temp, scale = T)
# {r pca, echo=FALSE,
# warning=FALSE,error=FALSE,message=FALSE}
library(ggplot2)
PCbiplot <- function(dat, PC, x = "PC1", y = "PC2", colors = c("black",
    "black", "blue", "lightblue")) {
    # PC being a prcomp object
```

```
    data <- data.frame(obsnames = df$price, PC$x)
    plot <- ggplot(data, aes_string(x = x, y = y)) + geom_text(alpha = 0.5,
        size = 3, aes(label = obsnames), color = data$obsnames)
    plot <- plot + geom_hline(aes(0), size = 0.1, yintercept = 0) +
        geom_vline(aes(0), size = 0.1, color = colors[2], xintercept = 0)
    datapc <- data.frame(varnames = rownames(PC$rotation), PC$rotation)
    mult <- min((max(data[, y]) - min(data[, y])/(max(datapc[,
        y]) - min(datapc[, y]))), (max(data[, x]) - min(data[,
        x])/(max(datapc[, x]) - min(datapc[, x]))))
    datapc <- transform(datapc, v1 = 0.7 * mult * (get(x)), v2 = 0.7 *
        mult * (get(y)))
    plot <- plot + coord_equal() + geom_text(data = datapc, aes(x = v1,
        y = v2, label = varnames), size = 3, vjust = 1, color = colors[3])
    plot <- plot + geom_segment(data = datapc, aes(x = 0, y = 0,
        xend = v1, yend = v2), arrow = arrow(length = unit(0.2,
        "cm")), alpha = 0.75, color = colors[4])
    plot
}

PCbiplot(df, df.pcr)

#


# {r model setup, echo=FALSE, warning=FALSE, error=FALSE}

library(caret)
library(stats)
library(car)
library(leaps)
library(rpart)
library(rpart.plot)
library(glmnet)


set.seed(444)

engage = df
engage = cbind(engage, model.matrix(~channel - 1, engage))
engage$channel = NULL
engage = cbind(engage, model.matrix(~store - 1, engage))
engage$store = NULL
engage = cbind(engage, model.matrix(~cut - 1, engage))
engage$cut = NULL


engage.train.idx = sample(seq(1:nrow(df)), size = floor(0.7 *
    nrow(df)), replace = FALSE)
engage.train = engage[engage.train.idx, ]
engage.test = engage[which(!(1:nrow(df) %in% engage.train.idx)),
    ]

engage.test.logprice = log(engage.test$price)
```

```
modelControl = trainControl(method = "repeatedcv", number = 10,
    repeats = 5)

LinearModelToEquation = function(m, label = "Model Equation") {
    t = as.data.frame(coef(m))
    colnames(t) = c("coef")
    t$coef = as.numeric(round(t$coef, 2))
    t$name = row.names(t)
    t = apply(t, 1, function(x) {
        return(paste(x[1], "*", x[2], "  +  "))
    })
    t = trimws(paste(unlist(t), collapse = ""))
    t = as.data.frame(trimws(gsub("\\+$", "", t)))
    colnames(t) = c(label)
    return(t)
}

LassoModelToEquation = function(m, label = "Model Equation") {
    t = as.data.frame(m)
    colnames(t) = c("coef")
    t$coef = as.numeric(round(t$coef, 3))
    t$name = row.names(t)
    t = apply(t, 1, function(x) {
        return(paste(x[1], "*", x[2], "  +  "))
    })
    t = trimws(paste(unlist(t), collapse = ""))
    t = as.data.frame(trimws(gsub("\\+$", "", t)))
    colnames(t) = c(label)
    return(t)
}

RMSE = function(data) {
    return(sqrt(mean((data - engage.test.logprice)^2)))
}


# {r eda lm forward, echo=FALSE, cache=FALSE, results=FALSE,
# warning=FALSE, comment=FALSE, progress=FALSE}

garbage = capture.output(eda.model.lm1 = train(price ~ ., data = engage,
    method = "leapForward", trControl = modelControl))

form = as.formula("price ~ carat + color + clarity + channelMall + storeGoodmans")
eda.model.lm1 = lm(form, data = engage)
aic.eda.model.lm1 = AIC(eda.model.lm1)


# {r eda log price, echo=FALSE, warning=FALSE, message=FALSE,
# error=FALSE}

garbage = capture.output(eda.model.lm2 = train(log(price) ~ .,
    data = engage, method = "leapForward", trControl = modelControl))
```

```
form = as.formula("log(price) ~ carat + color + clarity + channelMall + storeGoodmans")
eda.model.lm2 = lm(form, data = engage)
aic.eda.model.lm2 = AIC(eda.model.lm2)


###### Figure 9 - Best Subset Selection Number of Variables {r eda
###### best subsets part 0, echo=FALSE, include=FALSE,
###### out.width='300px'}

eda.model.subsets = regsubsets(log(price) ~ ., data = engage,
    nbest = 1, method = "exhaustive")
# {r eda best subsets part 1, echo=FALSE,
# warning=FALSE,message=FALSE,trace=FALSE}
plot(eda.model.subsets, scale = "bic")  # bic savings occurs at 6 variables
plot(eda.model.subsets, scale = "adjr2")  # adjr2 savings occurs at 9 variables,
# summary(eda.model.subsets)


# {r eda best subsets part 2, echo=FALSE, warning=FALSE,
# out.width='300px'}

# 6 variables
form = as.formula("log(price) ~ carat + color + clarity + channelInternet + channelMall + storeGoodmans
eda.model.lm3 = lm(form, data = engage)
# eda.model.lm3
aic.eda.model.lm3 = AIC(eda.model.lm3)


# 9 variables
form = as.formula("log(price) ~ carat + color + clarity + channelIndependent + channelInternet + storeF
eda.model.lm4 = lm(form, data = engage)
# eda.model.lm4
aic.eda.model.lm4 = AIC(eda.model.lm4)


# {r vif, echo=FALSE, warning=FALSE}
tbl = as.data.frame(cbind(vif(eda.model.lm4)))
colnames(tbl) = c("VIF")

# {r modeling linear 1, echo=FALSE, warning=FALSE}

model.linear1 = train(log(price) ~ carat + color + clarity +
    channelMall + storeGoodmans, data = engage.train, method = "lm",
    trControl = modelControl)
# vif(model.linear1$finalModel)
model.linear1.predict = predict(model.linear1, newdata = engage.test,
    interval = "prediction")
rmse.linear1 = RMSE(model.linear1.predict)


#

# {r modeling linear 2, echo=FALSE, warning=FALSE}
```

```
model.linear2 = train(log(price) ~ carat + color + clarity +
    channelIndependent + channelInternet + storeFredMeyer + storeGoodmans +
    storeRiddles + cutNotIdeal, data = engage.train, method = "lm",
    trControl = modelControl)

model.linear2.predict = predict(model.linear2, newdata = engage.test,
    interval = "prediction")
rmse.linear2 = RMSE(model.linear2.predict)

#

# {r modeling linear interactions 0, echo=FALSE,
# include=FALSE}

model.linear3 = train(log(price) ~ . * ., data = engage.train,
    method = "leapForward", trControl = modelControl)

# {r modeling linear interactions 1, echo=FALSE,
# warning=FALSE, messages=FALSE}
model.linear3.predict = predict(model.linear3, newdata = engage.test,
    interval = "prediction")
rmse.linear3 = RMSE(model.linear3.predict)


###### Figure 10 - Best Subset Selection Number of Variables with
###### Interactions {r modeling linear interactions pre 2,
###### echo=FALSE, warning=FALSE}

model.linear4 = train(log(price) ~ . * ., data = engage.train,
    method = "lm", trControl = modelControl)
#

# {r modeling linear interactions 2, echo=FALSE,
# warning=FALSE, out.width='300px'}

model.linear4.coefficients = as.data.frame(model.linear4$finalModel$coefficients)
colnames(model.linear4.coefficients) = c("coefficients")
model.linear4.coefficients$variable = rownames(model.linear4.coefficients)
model.linear4.coefficients$variable = gsub("`", "", model.linear4.coefficients$variable)
model.linear4.coefficients = model.linear4.coefficients[!is.na(model.linear4.coefficients$coefficients)
    ]
form = apply(model.linear4.coefficients, 1, function(x) {
    return(paste(x[2], " + "))
})
form = trimws(paste(unlist(form), collapse = ""))
form = gsub("\\(Intercept\\)  \\+", "", form)
form = trimws(gsub("\\+$", "", form))
form = paste("log(price) ~ ", form)
form = as.formula(form)

model.linear4 = regsubsets(form, data = engage.train, nbest = 1,
    method = "exhaustive", nvmax = 10, really.big = T)
model.linear4.summary = summary(model.linear4)
```

```
predict.regsubsets = function(object, form, newdata, id, ...) {
    mat = model.matrix(form, newdata)
    coefi = coef(object, id = id)
    mat[, names(coefi)] %*% coefi
}
model.linear4.predict = predict.regsubsets(model.linear4, form = form,
    newdata = engage.test, id = 10)
rmse.linear4 = RMSE(model.linear4.predict)

plot(model.linear4.summary$cp, xlab = "Number of variables",
    ylab = "Cp")

#

# {r modeling glmboost pre, echo=FALSE, warning=FALSE,
# include=FALSE}

model.glmboost = train(log(price) ~ ., data = engage.train, method = "glmboost")
# {r modeling glmboost, echo=FALSE, warning=FALSE}
model.glmboost.predict = predict(model.glmboost$finalModel, newdata = engage.test,
    interval = "prediction")
rmse.glmboost = RMSE(model.glmboost.predict)


###### Figure 11 - Lasso Model {r modeling lasso 1, echo=FALSE,
###### warning=FALSE, error=FALSE, out.width='300px'}
x = model.matrix(log(price) ~ ., data = engage.train)[, -1]
y = log(engage.train$price)
grid = 10^seq(10, -2, length = 100)
model.lasso1 = glmnet(x, y, alpha = 1, lambda = grid, standardize = TRUE)
plot(model.lasso1)

set.seed(444)
model.lasso1.cv = cv.glmnet(x, y, alpha = 1, standardize = TRUE)
plot(model.lasso1.cv)

bestlam = model.lasso1.cv$lambda.min
# bestlam

model.lasso1.predict = predict(model.lasso1, type = "coefficients",
    s = bestlam)[1:16, ]
model.lasso1.coefficients = as.data.frame(model.lasso1.predict[model.lasso1.predict !=
    0])
kable(LassoModelToEquation(model.lasso1.coefficients, "Lasso Model"),
    format = "markdown", padding = 0)

newx = model.matrix(log(price) ~ ., data = engage.test)[, -1]
model.lasso1.predict = predict(model.lasso1, s = bestlam, newx = newx)

rmse.lasso = RMSE(model.lasso1.predict)


###### Figure 12 - Decison Tree {r modeling tree pre, echo=FALSE,
```

```
###### warning=FALSE, include=FALSE}

model.tree = train(log(price) ~ ., data = engage.train, method = "ctree",
    trControl = modelControl)
# {r modeling tree, echo=FALSE, warning=FALSE, error=FALSE}

rpart.plot(rpart(log(price) ~ ., data = engage.train), main = "Tree Plot for Diamond Price",
    sub = "", cex = 0.7)
model.tree.predict = predict(model.tree, newdata = engage.test)
rmse.tree = RMSE(model.tree.predict)

#

###### Figure 13 - Random Forest {r modeling rf, echo=FALSE,
###### warning=FALSE, out.width='300px'}

model.rf = train(log(price) ~ ., data = engage.train, method = "rf",
    trControl = modelControl)
plot(model.rf, main = "")
plot(model.rf$finalModel, main = "")
model.rf.predict = predict(model.rf, newdata = engage.test)
rmse.rf = RMSE(model.rf.predict)

#

### Model Selection and Conclusion

###### Figure 14 - Summary of Model Performance by RMSE of Price
###### (Log) {r model selection, echo=FALSE, warning=FALSE}

models = c("Linear Model - 5 Variables", "Linear Model - 9 Variables",
    "Linear Model - Forward Selection of Interactions", "Linear Model - Best Subset Selection of Intera
    "Boosted Linear Model", "Decision Tree", "Random Forest")
rmses = round(c(rmse.linear1, rmse.linear2, rmse.linear3, rmse.linear4,
    rmse.glmboost, rmse.tree, rmse.rf), digits = 4)
selection.summary = cbind(models, rmses)
colnames(selection.summary) = c("Model", "RMSE of Price(log)")

# ```{r best model, echo=FALSE, warning=FALSE}

idxs = grep("store|price", colnames(engage.test))
store.summary = engage.test[, idxs]
store.summary$offset = engage.test$price - exp(model.linear4.predict)
idxs = grep("store", colnames(store.summary))
# store.summary = store.summary[,idxs] * store.summary$offset
colnames(store.summary) = gsub("store", "", colnames(store.summary))
mat = as.matrix(store.summary[, idxs])
store.summary$store = as.factor(colnames(mat)[mat %*% 1:ncol(mat)])

colors = brewer.pal(12, "Paired")
plot.settings = list(superpose.line = list(col = colors, border = "transparent"))

# log price gives a much more normal distribution
```

```
density.plots = densityplot(~offset, data = store.summary[store.summary$offset >
    -5000, ], groups = store, plot.points = FALSE, auto.key = list(space = "right",
    title = "Store"), par.settings = plot.settings, scales = list(x = list(at = seq(-5000,
    5000, by = 1000)), y = "free"), xlab = "")

plot(density.plots)
```