Diamonds ML R Notebook

Diamonds ML R Notebook Robert M. Taylor, PhD

This notebook is to demonstrate Exploratory Data Analysis (EDA), visualizations, and machine learning in R on the diamonds dataset that is available in R. "Price" will be our target.s

I'll first import the libraries I'll use.

```r
library(ggplot2)
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```r
library(rpart)
library(randomForest)
```

```
## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
library(tidyr)
library(modelr)
```

Load the dataset

```r
data(diamonds)
```

I'll just look at/inspect the dataset first. This is a clean data set so data cleaning will not be needed or demonstrated in this notebook.

```r
head(diamonds)
```

```
## # A tibble: 6 x 10
##   carat cut       color clarity depth table price     x     y     z
##   <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
## 1 0.23  Ideal     E     SI2      61.5    55   326  3.95  3.98  2.43
## 2 0.21  Premium   E     SI1      59.8    61   326  3.89  3.84  2.31
## 3 0.23  Good      E     VS1      56.9    65   327  4.05  4.07  2.31
## 4 0.290 Premium   I     VS2      62.4    58   334  4.2   4.23  2.63
## 5 0.31  Good      J     SI2      63.3    58   335  4.34  4.35  2.75
## 6 0.24  Very Good J     VVS2     62.8    57   336  3.94  3.96  2.48
```

What are the dimensions of the dataset?

```r
dim(diamonds)
```

```
## [1] 53940    10
```

So there are 53,940 rows and 10 feature columns.

I'll now get 1) a summary and 2) the structure of the data...

```r
summary(diamonds)
```

```
##      carat               cut        color        clarity          depth      
##  Min.   :0.2000   Fair     : 1610   D: 6775   SI1    :13065   Min.   :43.00  
##  1st Qu.:0.4000   Good     : 4906   E: 9797   VS2    :12258   1st Qu.:61.00  
##  Median :0.7000   Very Good:12082   F: 9542   SI2    : 9194   Median :61.80  
##  Mean   :0.7979   Premium  :13791   G:11292   VS1    : 8171   Mean   :61.75  
##  3rd Qu.:1.0400   Ideal    :21551   H: 8304   VVS2   : 5066   3rd Qu.:62.50  
##  Max.   :5.0100                     I: 5422   VVS1   : 3655   Max.   :79.00  
##                                     J: 2808   (Other): 2531                  
##      table           price             x                y         
##  Min.   :43.00   Min.   :  326   Min.   : 0.000   Min.   : 0.000  
##  1st Qu.:56.00   1st Qu.:  950   1st Qu.: 4.710   1st Qu.: 4.720  
##  Median :57.00   Median : 2401   Median : 5.700   Median : 5.710  
##  Mean   :57.46   Mean   : 3933   Mean   : 5.731   Mean   : 5.735  
##  3rd Qu.:59.00   3rd Qu.: 5324   3rd Qu.: 6.540   3rd Qu.: 6.540  
##  Max.   :95.00   Max.   :18823   Max.   :10.740   Max.   :58.900  
##                                                                   
##        z         
##  Min.   : 0.000  
##  1st Qu.: 2.910  
##  Median : 3.530  
##  Mean   : 3.539  
##  3rd Qu.: 4.040  
##  Max.   :31.800  
## 
```

I see that there is an (Other) variable for Clarity. I want to look at that closer.

```r
unique(diamonds$clarity)
```

```
## [1] SI2  SI1  VS1  VS2  VVS2 VVS1 I1   IF  
## Levels: I1 < SI2 < SI1 < VS2 < VS1 < VVS2 < VVS1 < IF
```

So, everything appears good. The summary has just grouped the I1 and IF clarities in the count values in the summary table above.

We also see that the price ranges from a minimum price of $326 to a max of $18,823

I'll go ahead and look at the structure of the dataset...

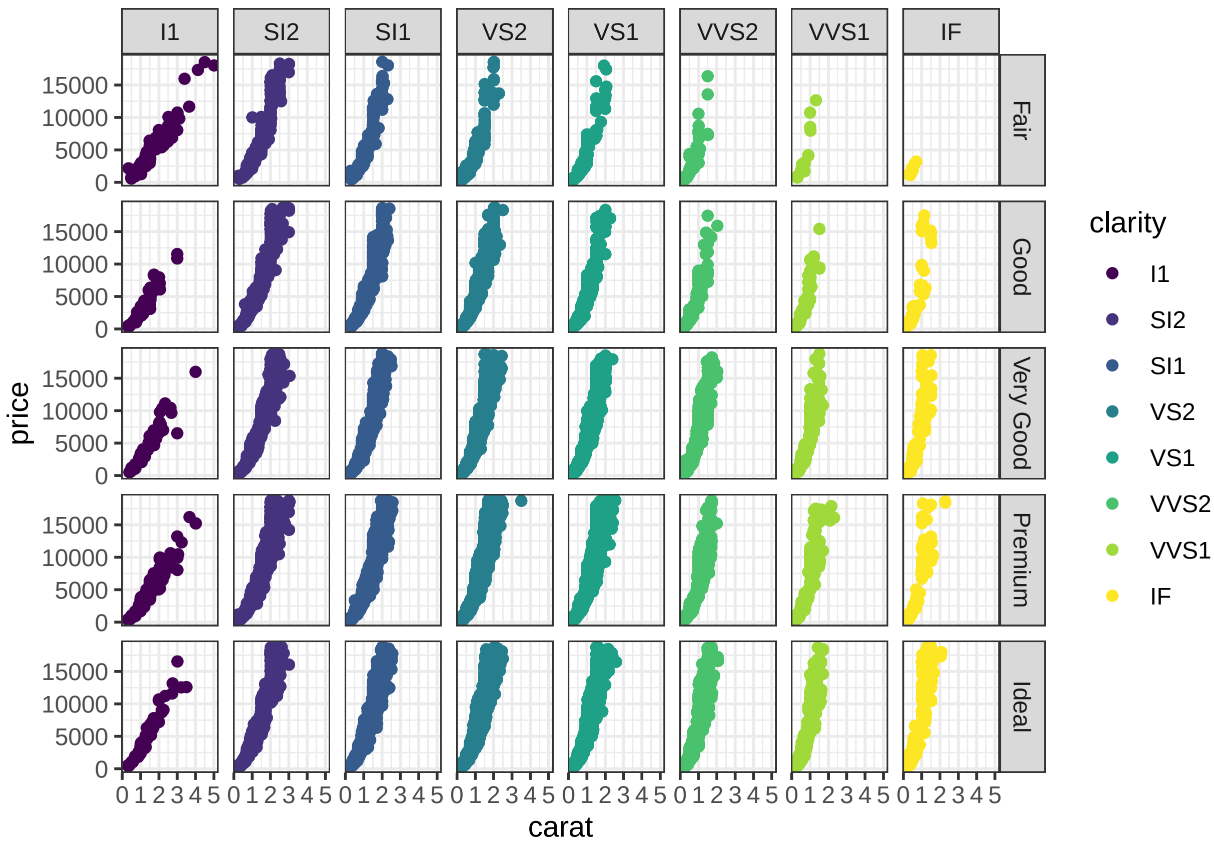
```r
str(diamonds)
```

```
## tibble [53,940 x 10] (S3: tbl_df/tbl/data.frame)
##  $ carat  : num [1:53940] 0.23 0.21 0.23 0.29 0.31 0.24 0.24 0.26 0.22 0.23 ...
##  $ cut    : Ord.factor w/ 5 levels "Fair"<"Good"<..: 5 4 2 4 2 3 3 3 1 3 ...
##  $ color  : Ord.factor w/ 7 levels "D"<"E"<"F"<"G"<..: 2 2 2 6 7 7 6 5 2 5 ...
##  $ clarity: Ord.factor w/ 8 levels "I1"<"SI2"<"SI1"<..: 2 3 5 4 2 6 7 3 4 5 ...
##  $ depth  : num [1:53940] 61.5 59.8 56.9 62.4 63.3 62.8 62.3 61.9 65.1 59.4 ...
##  $ table  : num [1:53940] 55 61 65 58 58 57 57 55 61 61 ...
##  $ price  : int [1:53940] 326 326 327 334 335 336 336 337 337 338 ...
##  $ x      : num [1:53940] 3.95 3.89 4.05 4.2 4.34 3.94 3.95 4.07 3.87 4 ...
##  $ y      : num [1:53940] 3.98 3.84 4.07 4.23 4.35 3.96 3.98 4.11 3.78 4.05 ...
```

```
##  $ z       : num [1:53940] 2.43 2.31 2.31 2.63 2.75 2.48 2.47 2.53 2.49 2.39 ...
```

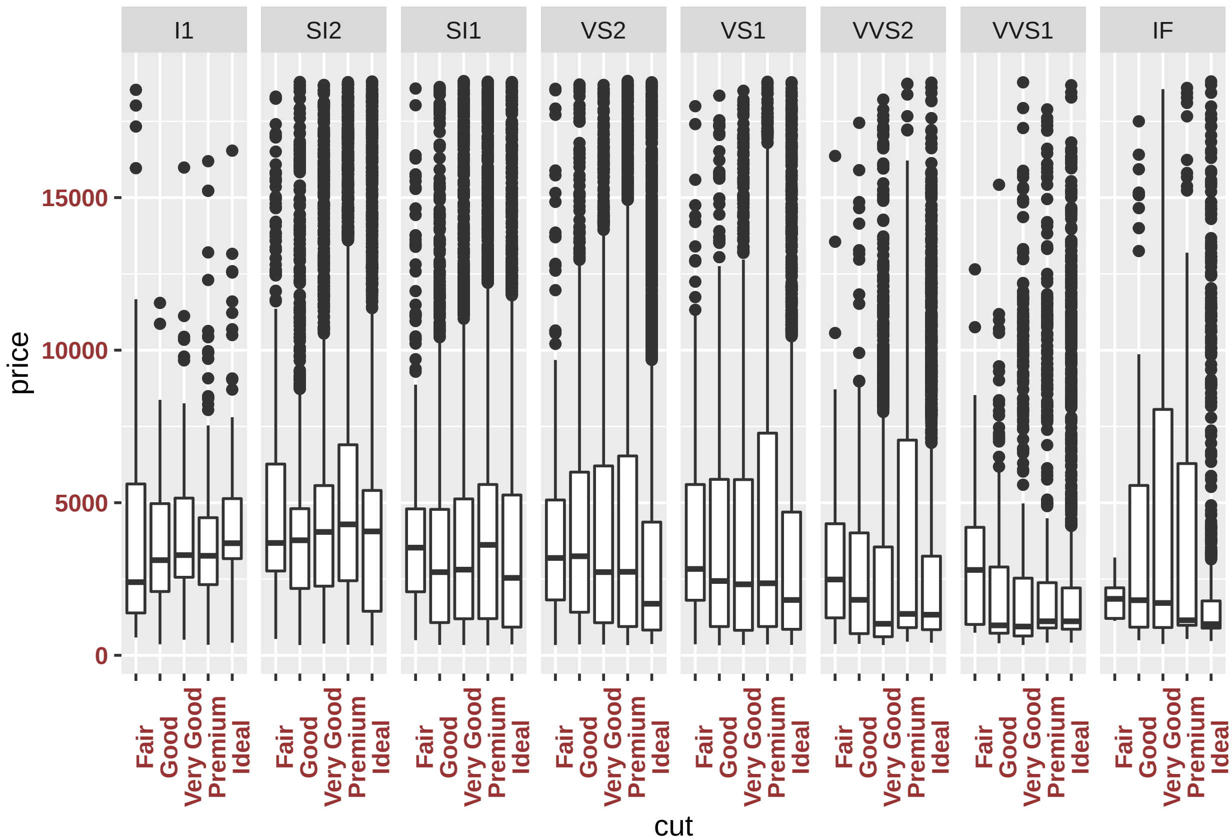
I'll now look at caret vs price using ggplot2

```r
g <- ggplot(diamonds, aes(x=carat, y=price))
g +
  geom_point(aes(color=clarity)) +
  facet_grid(cut ~ clarity)+
  theme_bw()
```



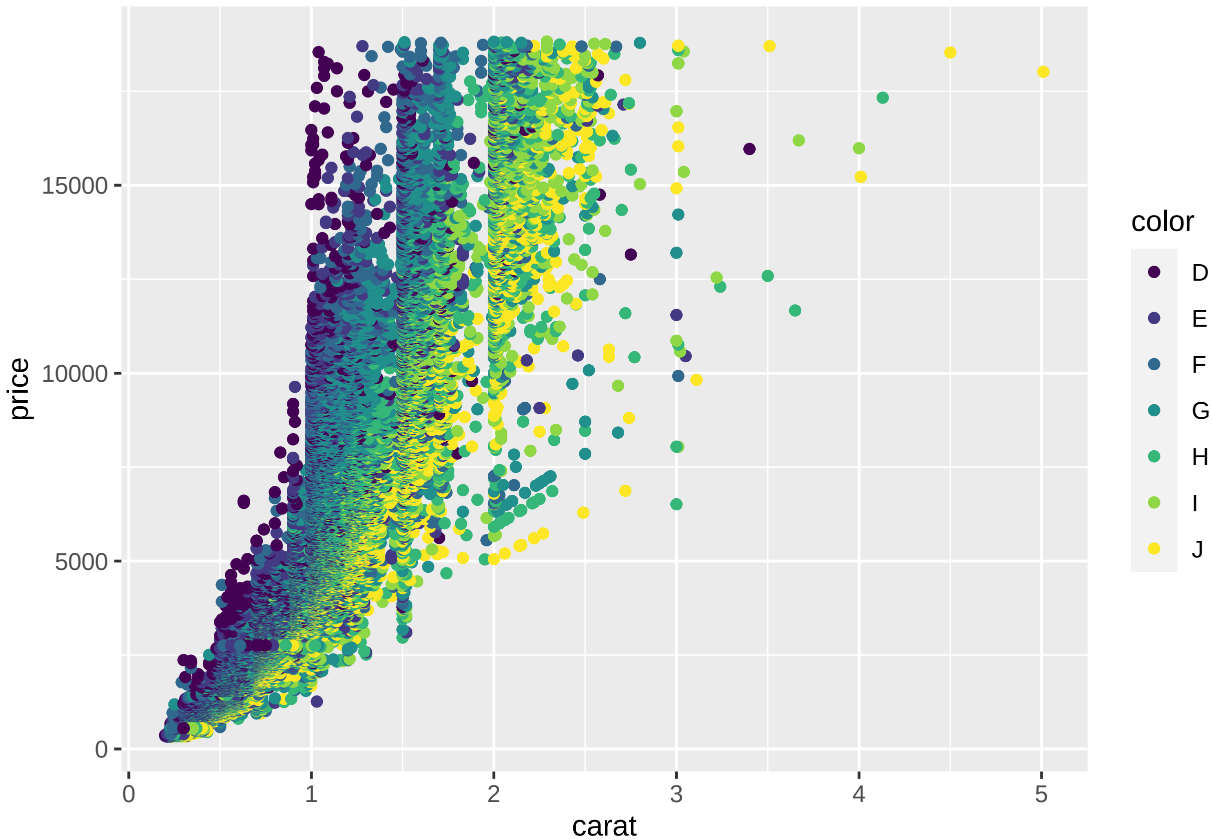
I'll use a boxplot to look at cut vs. price.

```r
diamonds$cut = as.factor(diamonds$cut)
g <- ggplot(diamonds, aes(x=cut, y=price))
g +
  geom_boxplot()+
  facet_grid(~clarity)+
  theme(axis.text.x = element_text(angle = 90, face = "bold", color = "#993333",
                                   size = 9))+
  theme(axis.text.y = element_text(face = "bold", color = "#993333"))
```



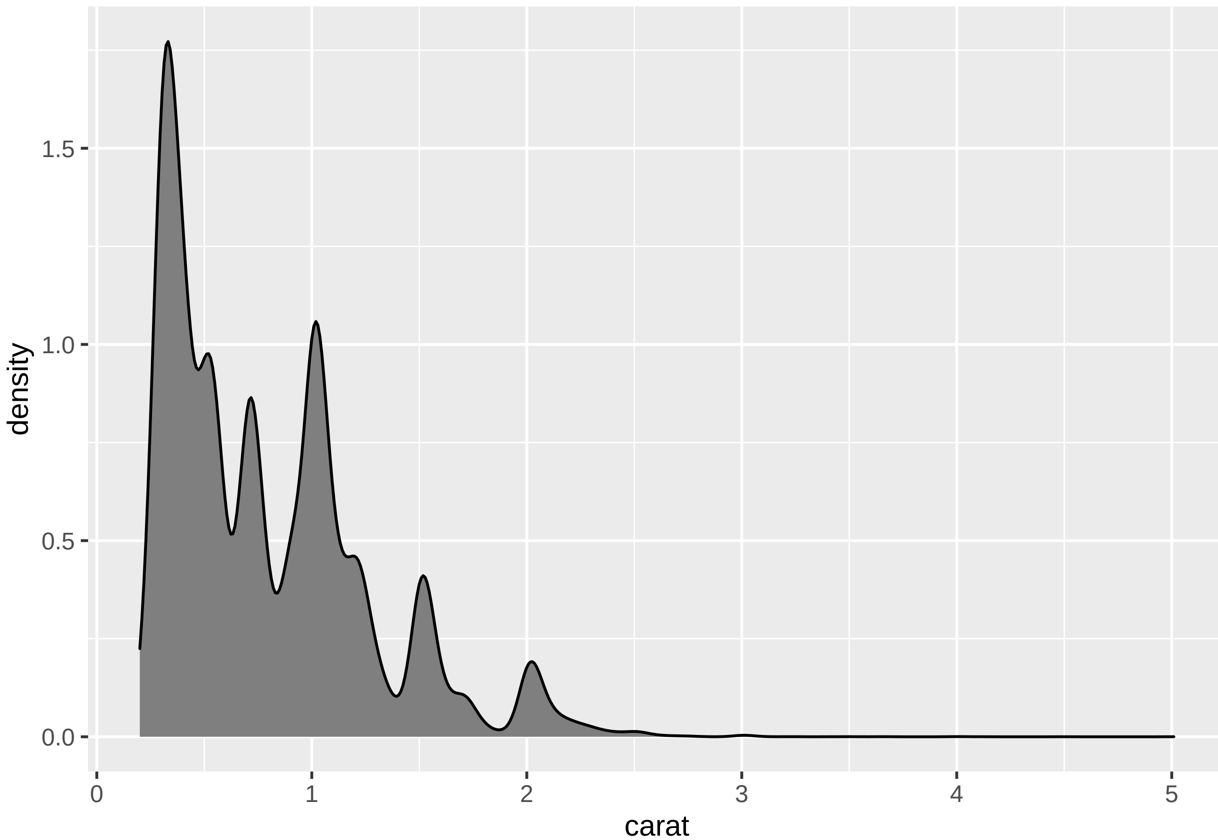
We can also look at the carat vs. color vs. price

```
g <- ggplot(diamonds, aes(x=carat, y=price))
g +
geom_point(aes(color = color))
```

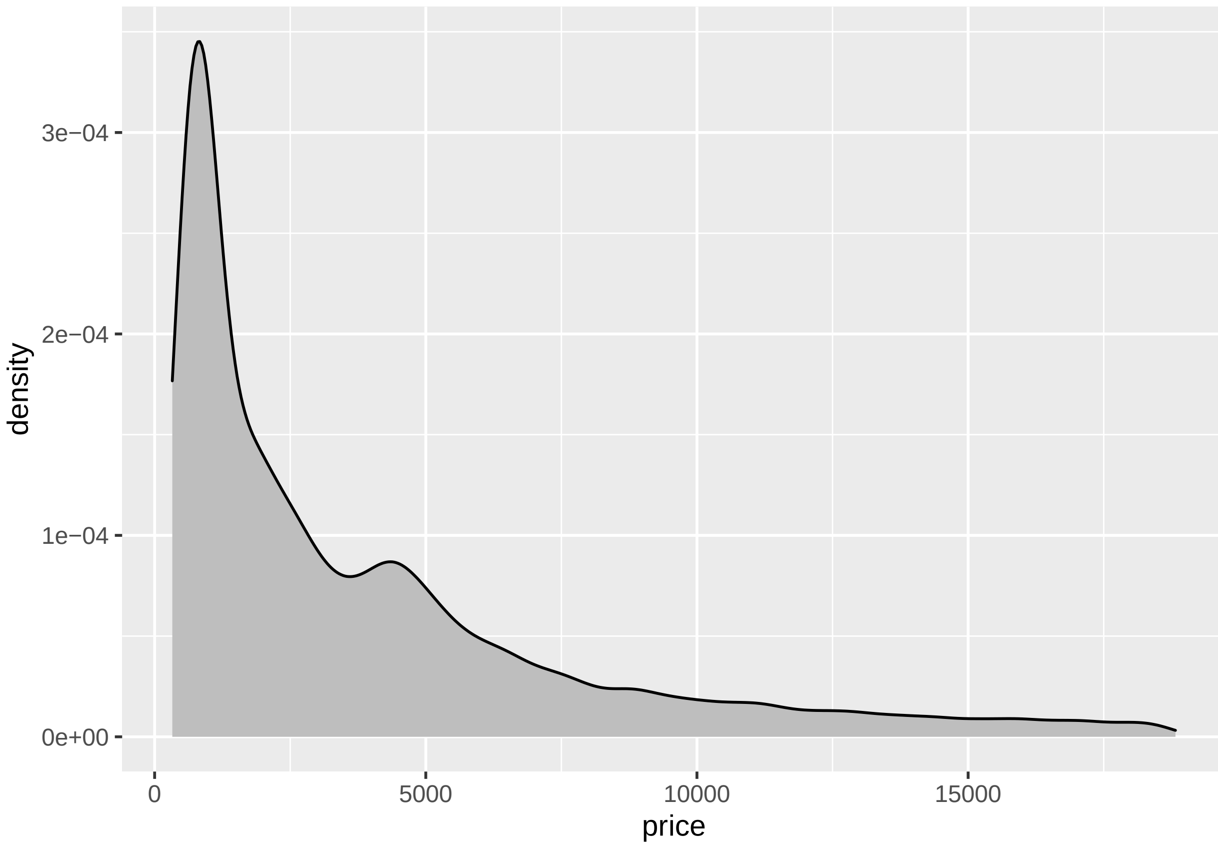


I'll now look at the distribution of carat and price.

```
g <- ggplot(diamonds)
g +
  geom_density(aes(x=carat), fill="gray50")
```




```
g <- ggplot(diamonds)
g +
  geom_density(aes(x=price), fill="gray")
```



So, thus far, we can now see that most diamonds are < 2 carats and < ~$2500 (although a second peak can be seen around $4000).

M.L.

DECISION TREE BUILD MODEL I'll first use a decision tree model to predict the diamond prices.

```
colnames(diamonds)
```

```
##  [1] "carat"   "cut"     "color"   "clarity" "depth"   "table"   "price"
##  [8] "x"       "y"       "z"
```

SPLIT THE DATA (70/30 split)

```
splitData <- resample_partition(diamonds, c(test=0.3, train=0.7))
```

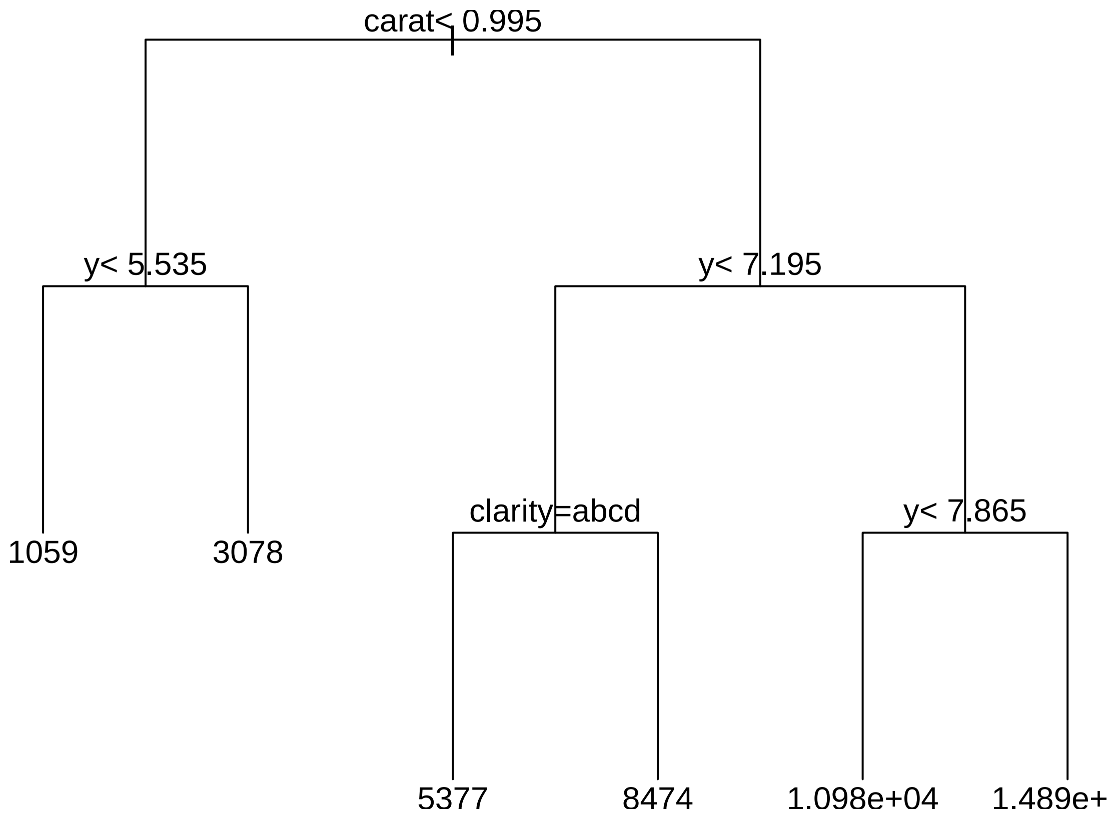
How many cases are in the test and training sets?

```
lapply(splitData, dim)
```

```
## $test
## [1] 16181    10
##
## $train
## [1] 37759    10
```

```
# train the tree
fit <- rpart(price ~ carat + cut + color + clarity + depth + table + x + y + z, data = splitData$train)
```

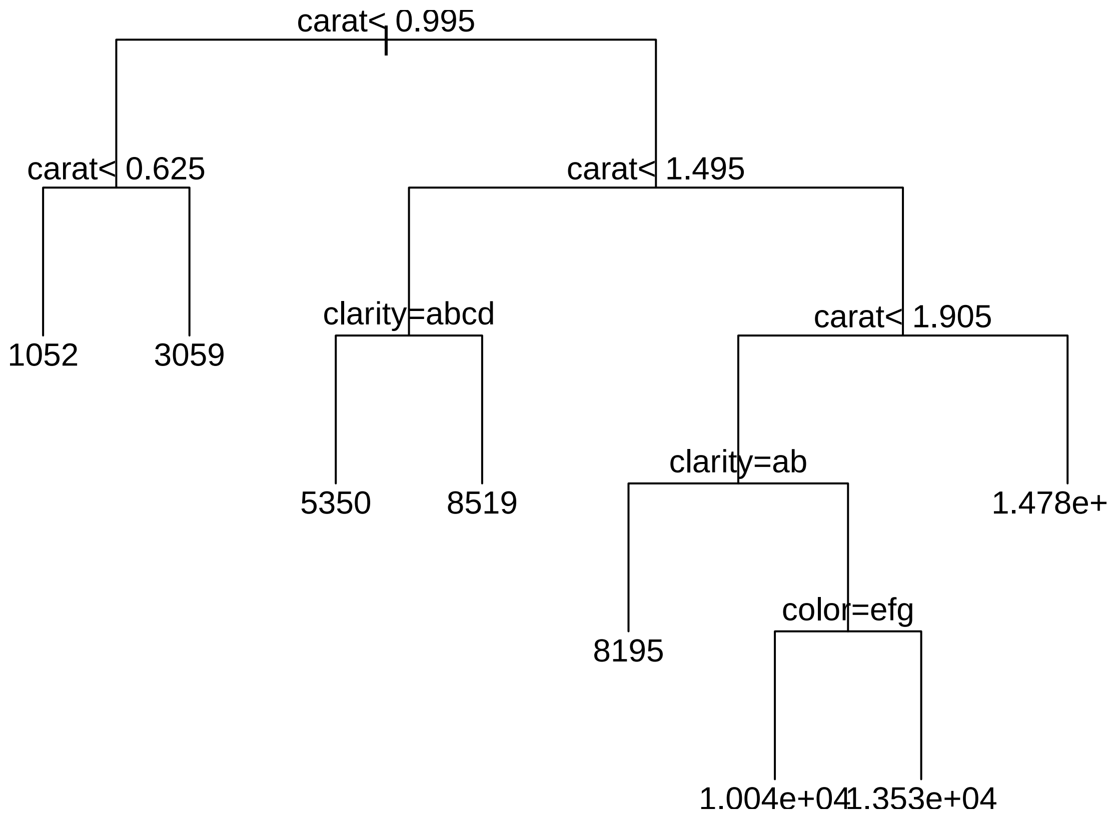
```
#plot the regression tree
plot(fit, uniform=TRUE)
text(fit, cex=1)
```



…What if we reomive the x, y, z, data, and table features? These above group are features that are not easily available to the common consumer, like me shopping for a new ring. Therefore, I'll see how the fit is with only the readily available features, carat, cut, color, and clarity.

```
# train the tree
fit2 <- rpart(price ~ carat + cut + color + clarity, data = splitData$train)
```

```
#plot the regression tree
plot(fit2, uniform=TRUE)
text(fit2, cex=1)
```



I'LL USE THE 'fit2' MODEL SINCE IT MAKES MORE SENSE FOR MY ENGAGEMENT RING SHOPPING SINCE CARAET, CLARITY, ETC ARE THINGS I CAN EASILY FIND OUT FROM A SELLAR, WHEREAS 'x, y, and z'ARE NOT.

DECISION TREE MODEL PREDICT DIAMOND PRICES Now, that I've generated a decision tree model, I'll now use it to predict prices.

DECISION TREE MODEL EVALUATION

MAE (Mean absolute Error)

```
maeTree <-mae(model = fit2, data=splitData$test)
maeTree
```

```
## [1] 835.146
```

I'll build a function to help compare MAE scores from different values for the tree depth (maxdepth)...

```
# A function to get the maximum average error for a given max depth. You should pass in
# the target as the name of the target colum and the predictors as vector where each
# item in the vector is the name of the column.

get_mae <- function(maxdepth, target, predictors, training_data, testing_data){
  #turn the predictors & target into a formula to pass to rpart
  predictors <- paste(predictors, collapse='+')
  formula <- as.formula(paste(target, "~", predictors, sep = ""))
  #build our model
  model <- rpart(formula, data=training_data, control = rpart.control(maxdepth = maxdepth))
  #get the mae
  mae <- mae(model, testing_data)
  return(mae)
}
```

```r
#Feed in the target and predictors
target <- "price"
predictors <- c("carat", "cut", "color", "clarity")
# get the MaE for the maxdepths between 1 and 10
for(i in 1:10){
  mae <- get_mae(maxdepth = i, target = target, predictors = predictors,
                 training_data = splitData$train, testing_data = splitData$test)
  print(glue::glue("Maxdepth: ",i, "\t MAE: ", mae))
}
```

```
## Maxdepth: 1   MAE: 1717.50055497977
## Maxdepth: 2   MAE: 1042.80304167472
## Maxdepth: 3   MAE: 891.329899980287
## Maxdepth: 4   MAE: 865.382423661785
## Maxdepth: 5   MAE: 835.146013826217
## Maxdepth: 6   MAE: 835.146013826217
## Maxdepth: 7   MAE: 835.146013826217
## Maxdepth: 8   MAE: 835.146013826217
## Maxdepth: 9   MAE: 835.146013826217
## Maxdepth: 10  MAE: 835.146013826217
```

RANDOM FOREST

```r
fitRandomForest <- randomForest(price ~ carat + cut + color + clarity, data=splitData$train)
maeForest <- mae(model = fitRandomForest, data=splitData$test)
maeForest
```

```
## [1] 899.502
```

```r
fitRandomForest
```

```
##
## Call:
##  randomForest(formula = price ~ carat + cut + color + clarity,      data = splitData$train)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 1
##
##           Mean of squared residuals: 1794744
##                     % Var explained: 88.66
```

LINEAR MODEL

```r
fitLinear <- lm(price ~ carat + cut + color + clarity, data=splitData$train)
maeLinear <- mae(model = fitLinear, data=splitData$test)
maeLinear
```

```
## [1] 808.8377
```

```r
summary(fitLinear)
```

```
##
## Call:
## lm(formula = price ~ carat + cut + color + clarity, data = splitData$train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
```

```
## -12621.1   -676.8   -195.9    461.5  10403.2
##
## Coefficients:
##              Estimate Std. Error  t value Pr(>|t|)
## (Intercept) -3707.097     16.598 -223.346  < 2e-16 ***
## carat        8877.777     14.342  619.006  < 2e-16 ***
## cut.L         704.988     24.039   29.327  < 2e-16 ***
## cut.Q        -327.297     21.178  -15.455  < 2e-16 ***
## cut.C         173.304     18.462    9.387  < 2e-16 ***
## cut^4           9.332     14.813    0.630    0.529
## color.L     -1895.714     21.060  -90.016  < 2e-16 ***
## color.Q      -614.831     19.177  -32.061  < 2e-16 ***
## color.C      -166.173     17.923   -9.271  < 2e-16 ***
## color^4        17.897     16.476    1.086    0.277
## color^5       -92.515     15.561   -5.945 2.78e-09 ***
## color^6       -58.799     14.159   -4.153 3.29e-05 ***
## clarity.L    4218.951     36.418  115.848  < 2e-16 ***
## clarity.Q   -1840.129     33.996  -54.128  < 2e-16 ***
## clarity.C     928.774     29.157   31.854  < 2e-16 ***
## clarity^4    -366.856     23.360  -15.704  < 2e-16 ***
## clarity^5     193.423     19.140   10.106  < 2e-16 ***
## clarity^6     -11.405     16.720   -0.682    0.495
## clarity^7     119.332     14.726    8.104 5.49e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1152 on 37740 degrees of freedom
## Multiple R-squared:  0.9162, Adjusted R-squared:  0.9161
## F-statistic: 2.291e+04 on 18 and 37740 DF,  p-value: < 2.2e-16
```

```r
coef(fitLinear)
```

```
##  (Intercept)        carat        cut.L        cut.Q        cut.C        cut^4
## -3707.096881  8877.777180   704.988263  -327.296838   173.303738     9.332101
##      color.L      color.Q      color.C      color^4      color^5      color^6
## -1895.713745  -614.831194  -166.173302    17.896622   -92.515227   -58.799452
##    clarity.L    clarity.Q    clarity.C    clarity^4    clarity^5    clarity^6
##  4218.950602 -1840.128996   928.774461  -366.856191   193.423086   -11.404789
##    clarity^7
##   119.332028
```

LOGISTIC REGRESSION

```r
fitLogistic <- glm(price ~ carat + cut + color + clarity, data=splitData$train)
maeLogistic <- mae(model = fitLogistic, data=splitData$test)
maeLogistic
```

```
## [1] 808.8377
```

```r
summary(fitLogistic)
```

```
##
## Call:
## glm(formula = price ~ carat + cut + color + clarity, data = splitData$train)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
```

```
## -12621.1    -676.8    -195.9     461.5   10403.2
##
## Coefficients:
##              Estimate Std. Error  t value Pr(>|t|)
## (Intercept) -3707.097     16.598 -223.346  < 2e-16 ***
## carat        8877.777     14.342  619.006  < 2e-16 ***
## cut.L         704.988     24.039   29.327  < 2e-16 ***
## cut.Q        -327.297     21.178  -15.455  < 2e-16 ***
## cut.C         173.304     18.462    9.387  < 2e-16 ***
## cut^4           9.332     14.813    0.630    0.529
## color.L     -1895.714     21.060  -90.016  < 2e-16 ***
## color.Q      -614.831     19.177  -32.061  < 2e-16 ***
## color.C      -166.173     17.923   -9.271  < 2e-16 ***
## color^4        17.897     16.476    1.086    0.277
## color^5       -92.515     15.561   -5.945 2.78e-09 ***
## color^6       -58.799     14.159   -4.153 3.29e-05 ***
## clarity.L    4218.951     36.418  115.848  < 2e-16 ***
## clarity.Q   -1840.129     33.996  -54.128  < 2e-16 ***
## clarity.C     928.774     29.157   31.854  < 2e-16 ***
## clarity^4    -366.856     23.360  -15.704  < 2e-16 ***
## clarity^5     193.423     19.140   10.106  < 2e-16 ***
## clarity^6     -11.405     16.720   -0.682    0.495
## clarity^7     119.332     14.726    8.104 5.49e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 1327262)
##
##     Null deviance: 5.9747e+11  on 37758  degrees of freedom
## Residual deviance: 5.0091e+10  on 37740  degrees of freedom
## AIC: 639527
##
## Number of Fisher Scoring iterations: 2
```

REPORT GENERATION

```r
print('REPORT')
```

```
## [1] "REPORT"
```

```r
print("Decision Tree")
```

```
## [1] "Decision Tree"
```

```r
print(maeTree)
```

```
## [1] 835.146
```

```r
print("Random Forest")
```

```
## [1] "Random Forest"
```

```r
print(maeForest)
```

```
## [1] 899.502
```

```r
print("Linear Regression")
```

```
## [1] "Linear Regression"
```

```r
print(maeLinear)
```

```
## [1] 808.8377
```

```r
print("Logistic Regression")
```

```
## [1] "Logistic Regression"
```

```r
print(maeLogistic)
```

```
## [1] 808.8377
```