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.

library(ggplot2)  
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

library(tidyr)  
library(modelr)

Load the dataset

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.

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?

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…

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.

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 minimium price of $326 to a max of $18,823

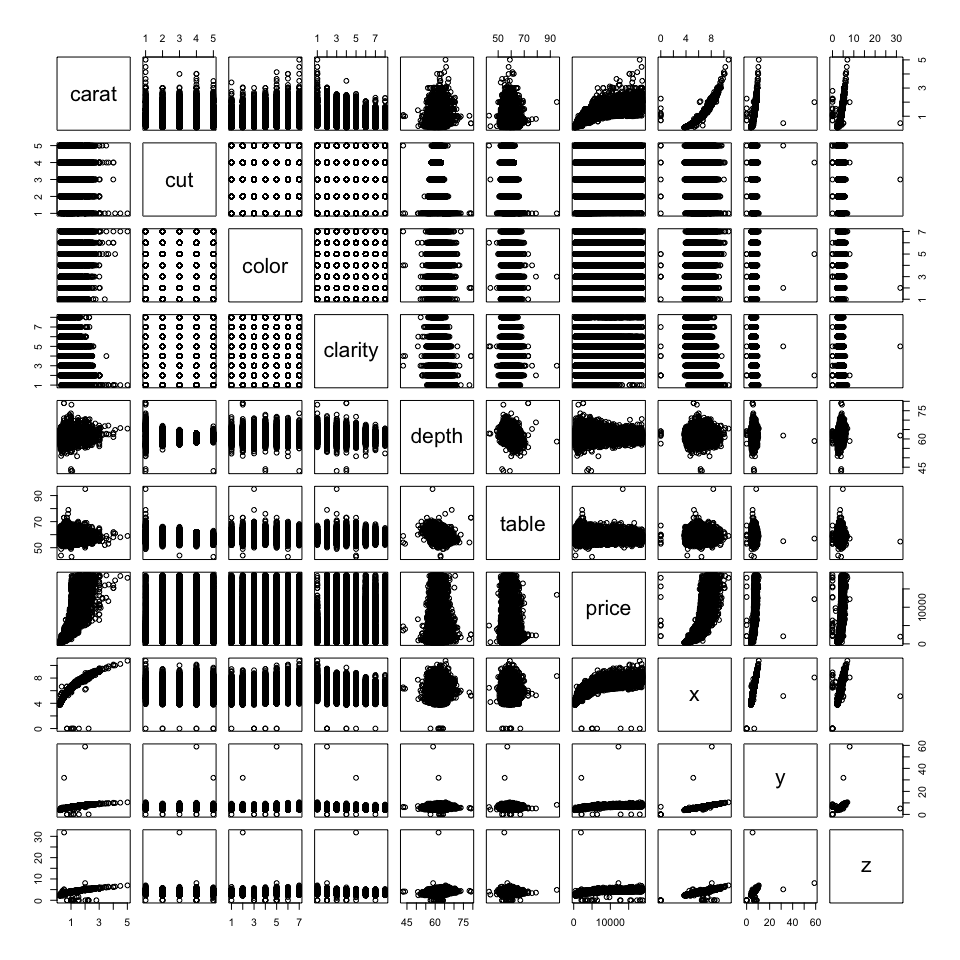
I’ll go ahead and look at the structure of the dataset…

str(diamonds)

## tibble [53,940 × 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 ...

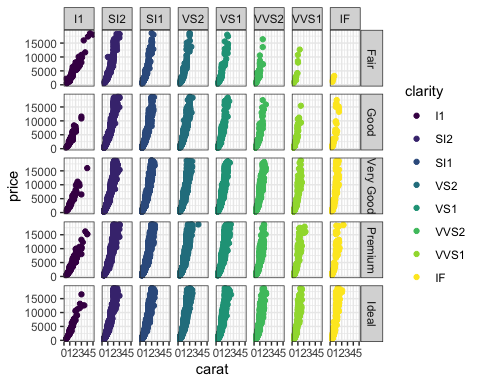
I’ll first do a quick and dirty look at the data…

plot(diamonds)



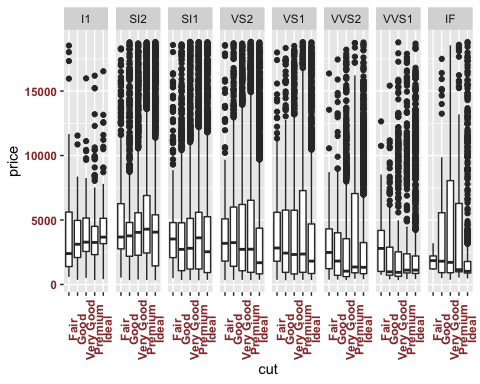
I’ll now look at caret vs price using ggplot2

g <- ggplot(diamonds, aes(x=carat, y=price))  
g +   
 geom\_point(aes(color=clarity)) +  
 facet\_grid(cut ~ clarity)+  
 theme\_bw()

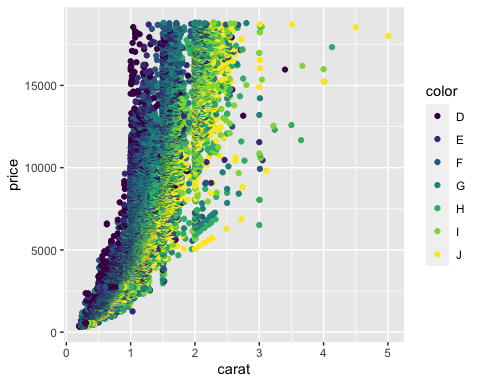


I’ll use a boxplot to look at cut vs. price.

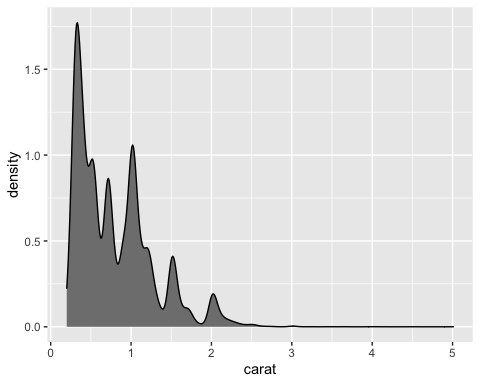
diamonds$cut = as.factor(diamonds$cut)  
g <- ggplot(diamonds, aes(x=cut, y=price))  
g +  
 geom\_boxplot()+  
 facet\_grid(~clarity)+  
 theme(axis.text.x = element\_text(angle = 90, face = "bold", color = "#993333",   
 size = 9))+  
 theme(axis.text.y = element\_text(face = "bold", color = "#993333"))

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

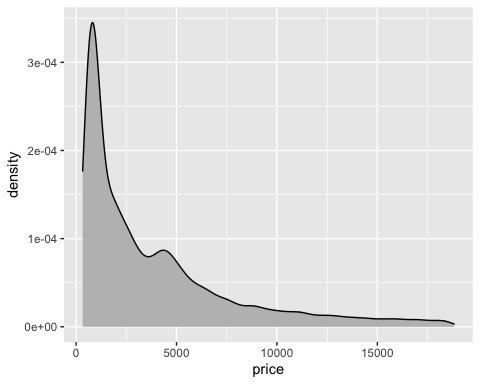
g <- ggplot(diamonds, aes(x=carat, y=price))  
g +  
geom\_point(aes(color = color))

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

g <- ggplot(diamonds)  
g +  
 geom\_density(aes(x=carat), fill="gray50")

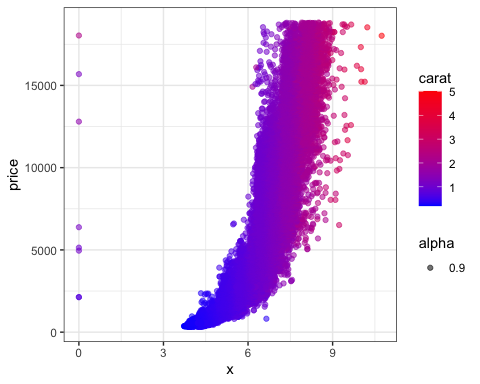


g <- ggplot(diamonds)  
g +  
 geom\_density(aes(x=price), fill="gray")

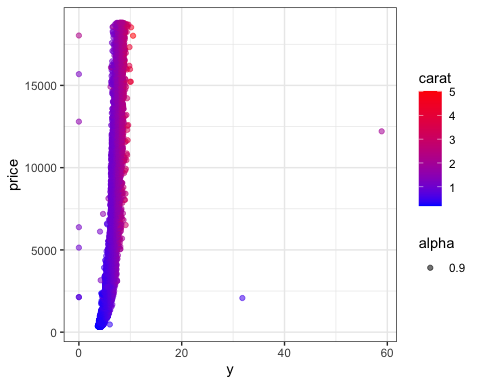
 So, thus far, we can now see that most diamonds are < 2 carats and < ~$2500 (although a second peak can be seen around $4000).

I’ll now look at the distribution of the ‘x’, ‘y’, and ‘z’ features and price.

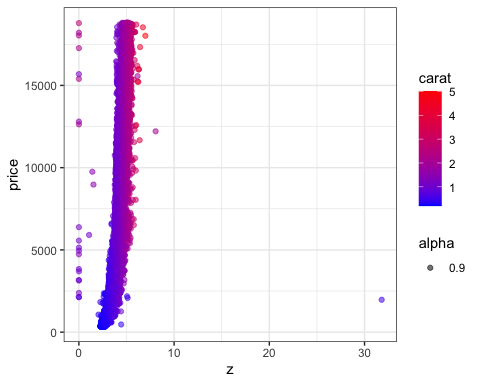
g <- ggplot(diamonds, aes(x=x, y=price))  
gg <- g + geom\_point(aes(color = carat, alpha=0.9))  
gg + scale\_color\_gradient(low="blue", high="red") + theme\_bw()



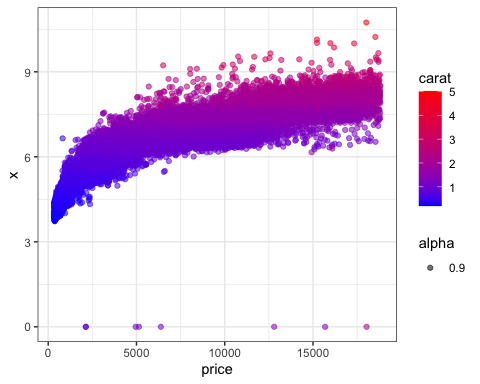
g <- ggplot(diamonds, aes(x=y, y=price))  
gg <- g + geom\_point(aes(color = carat, alpha=0.9))  
gg + scale\_color\_gradient(low="blue", high="red") + theme\_bw()



g <- ggplot(diamonds, aes(x=z, y=price))  
gg <- g + geom\_point(aes(color = carat, alpha=0.9))  
gg + scale\_color\_gradient(low="blue", high="red") + theme\_bw()

 So, generally, higher carat = higher x, y, and z, regardless of price…which we can see a little more clear by flipping the axes.

g <- ggplot(diamonds, aes(x=price, y=x))  
gg <- g + geom\_point(aes(color = carat, alpha=0.9))  
gg + scale\_color\_gradient(low="blue", high="red") + theme\_bw()



M.L.

I’ll first need to split the data for the machine learning

SPLIT THE DATA (70/30 split)

set.seed(27)  
diamond.indices <- sample(1:nrow(diamonds), 0.7\*nrow(diamonds), replace=F)  
diamond.train <- diamonds[diamond.indices,]  
print("Train/Test 70/30 Split Dimensions")

## [1] "Train/Test 70/30 Split Dimensions"

print("train set")

## [1] "train set"

dim(diamond.train)

## [1] 37758 10

diamond.test <- diamonds[-diamond.indices,]  
print("test set")

## [1] "test set"

dim(diamond.test)

## [1] 16182 10

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"

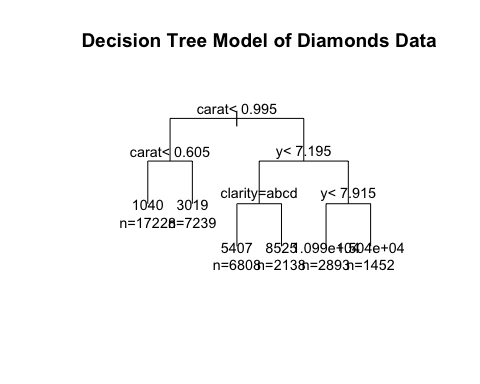
## fitting decision model on training set

diamond.tree.model <- rpart(price ~ ., data=diamond.train)  
summary(diamond.tree.model)

## Call:  
## rpart(formula = price ~ ., data = diamond.train)  
## n= 37758   
##   
## CP nsplit rel error xerror xstd  
## 1 0.61073090 0 1.0000000 1.0000473 0.010477443  
## 2 0.18531222 1 0.3892691 0.3906040 0.004601755  
## 3 0.03298210 2 0.2039569 0.2094537 0.002488080  
## 4 0.02624778 3 0.1709748 0.1731940 0.002450457  
## 5 0.02613996 4 0.1447270 0.1671793 0.002410000  
## 6 0.01000000 5 0.1185870 0.1225034 0.001906165  
##   
## Variable importance  
## carat y x z clarity color   
## 25 24 24 23 2 1   
##   
## Node number 1: 37758 observations, complexity param=0.6107309  
## mean=3931.256, MSE=1.602817e+07   
## left son=2 (24467 obs) right son=3 (13291 obs)  
## Primary splits:  
## carat < 0.995 to the left, improve=0.61073090, (0 missing)  
## y < 6.345 to the left, improve=0.61056270, (0 missing)  
## x < 6.335 to the left, improve=0.60650440, (0 missing)  
## z < 3.915 to the left, improve=0.60145420, (0 missing)  
## color splits as LLLLRRR, improve=0.02200518, (0 missing)  
## Surrogate splits:  
## x < 6.275 to the left, agree=0.983, adj=0.952, (0 split)  
## y < 6.265 to the left, agree=0.981, adj=0.945, (0 split)  
## z < 3.895 to the left, agree=0.978, adj=0.937, (0 split)  
## clarity splits as RRLLLLLL, agree=0.679, adj=0.089, (0 split)  
## color splits as LLLLLRR, agree=0.660, adj=0.035, (0 split)  
##   
## Node number 2: 24467 observations, complexity param=0.0329821  
## mean=1625.277, MSE=1234231   
## left son=4 (17228 obs) right son=5 (7239 obs)  
## Primary splits:  
## carat < 0.605 to the left, improve=0.66098860, (0 missing)  
## y < 5.535 to the left, improve=0.66065990, (0 missing)  
## x < 5.485 to the left, improve=0.65876000, (0 missing)  
## z < 3.375 to the left, improve=0.65761570, (0 missing)  
## clarity splits as RRRLLLLL, improve=0.01255531, (0 missing)  
## Surrogate splits:  
## x < 5.455 to the left, agree=0.992, adj=0.973, (0 split)  
## y < 5.465 to the left, agree=0.990, adj=0.967, (0 split)  
## z < 3.365 to the left, agree=0.989, adj=0.963, (0 split)  
## clarity splits as RRLLLLLL, agree=0.722, adj=0.061, (0 split)  
## cut splits as RLLLL, agree=0.713, adj=0.028, (0 split)  
##   
## Node number 3: 13291 observations, complexity param=0.1853122  
## mean=8176.263, MSE=1.545289e+07   
## left son=6 (8946 obs) right son=7 (4345 obs)  
## Primary splits:  
## y < 7.195 to the left, improve=0.54604620, (0 missing)  
## x < 7.195 to the left, improve=0.53867840, (0 missing)  
## carat < 1.495 to the left, improve=0.53777360, (0 missing)  
## z < 4.425 to the left, improve=0.52596500, (0 missing)  
## clarity splits as LLLRRRRR, improve=0.05469725, (0 missing)  
## Surrogate splits:  
## x < 7.185 to the left, agree=0.985, adj=0.954, (0 split)  
## carat < 1.445 to the left, agree=0.980, adj=0.939, (0 split)  
## z < 4.435 to the left, agree=0.964, adj=0.889, (0 split)  
## color splits as LLLLLRR, agree=0.676, adj=0.008, (0 split)  
## table < 67.5 to the left, agree=0.673, adj=0.001, (0 split)  
##   
## Node number 4: 17228 observations  
## mean=1039.79, MSE=254117.9   
##   
## Node number 5: 7239 observations  
## mean=3018.669, MSE=809435.1   
##   
## Node number 6: 8946 observations, complexity param=0.02613996  
## mean=6151.845, MSE=4746426   
## left son=12 (6808 obs) right son=13 (2138 obs)  
## Primary splits:  
## clarity splits as LLLLRRRR, improve=0.3725651, (0 missing)  
## y < 6.775 to the left, improve=0.1272920, (0 missing)  
## x < 6.775 to the left, improve=0.1119381, (0 missing)  
## carat < 1.165 to the left, improve=0.1110267, (0 missing)  
## color splits as RRRRLLL, improve=0.1101598, (0 missing)  
## Surrogate splits:  
## y < 4.175 to the right, agree=0.761, adj=0.001, (0 split)  
##   
## Node number 7: 4345 observations, complexity param=0.02624778  
## mean=12344.37, MSE=1.16855e+07   
## left son=14 (2893 obs) right son=15 (1452 obs)  
## Primary splits:  
## y < 7.915 to the left, improve=0.31285860, (0 missing)  
## x < 7.895 to the left, improve=0.30484200, (0 missing)  
## carat < 1.965 to the left, improve=0.29995570, (0 missing)  
## z < 4.805 to the left, improve=0.28459500, (0 missing)  
## clarity splits as LRRRRRRR, improve=0.07787712, (0 missing)  
## Surrogate splits:  
## x < 7.945 to the left, agree=0.979, adj=0.938, (0 split)  
## carat < 1.92 to the left, agree=0.966, adj=0.899, (0 split)  
## z < 4.845 to the left, agree=0.942, adj=0.827, (0 split)  
## clarity splits as RRLLLLLL, agree=0.681, adj=0.045, (0 split)  
## depth < 57.05 to the right, agree=0.667, adj=0.005, (0 split)  
##   
## Node number 12: 6808 observations  
## mean=5406.635, MSE=2097828   
##   
## Node number 13: 2138 observations  
## mean=8524.805, MSE=5781025   
##   
## Node number 14: 2893 observations  
## mean=10989.79, MSE=8745255   
##   
## Node number 15: 1452 observations  
## mean=15043.28, MSE=6603691

## VISUALIZE THE TRAINED MODEL

## plot the tree structure  
plot(diamond.tree.model, margin=c(.25), uniform=TRUE)  
title(main = "Decision Tree Model of Diamonds Data")  
text(diamond.tree.model, use.n = TRUE, cex=.9)

 ## make prediction using decision model

tree.predictions <- predict(diamond.tree.model, diamond.test)  
head(tree.predictions)

## 1 2 3 4 5 6   
## 1039.79 1039.79 1039.79 1039.79 1039.79 1039.79

## Comparison table

diamond.comparison <- diamond.test  
diamond.comparison$Predictions <- tree.predictions  
diamond.comparison[ , c("price", "Predictions")]

## # A tibble: 16,182 x 2  
## price Predictions  
## <int> <dbl>  
## 1 344 1040.  
## 2 348 1040.  
## 3 354 1040.  
## 4 357 1040.  
## 5 402 1040.  
## 6 402 1040.  
## 7 403 1040.  
## 8 403 1040.  
## 9 403 1040.  
## 10 404 1040.  
## # … with 16,172 more rows

## View misclassified rows

disagreement.index <- diamond.comparison$price != diamond.comparison$Predictions  
diamond.comparison[disagreement.index,]

## # A tibble: 16,182 x 11  
## carat cut color clarity depth table price x y z Predictions  
## <dbl> <ord> <ord> <ord> <dbl> <dbl> <int> <dbl> <dbl> <dbl> <dbl>  
## 1 0.31 Ideal J SI2 62.2 54 344 4.35 4.37 2.71 1040.  
## 2 0.3 Ideal I SI2 62 54 348 4.31 4.34 2.68 1040.  
## 3 0.23 Very Good G VVS2 60.4 58 354 3.97 4.01 2.41 1040.  
## 4 0.23 Very Good F VS1 60.9 57 357 3.96 3.99 2.42 1040.  
## 5 0.23 Very Good E VS1 59.5 58 402 4.01 4.06 2.4 1040.  
## 6 0.23 Good F VS1 58.2 59 402 4.06 4.08 2.37 1040.  
## 7 0.33 Ideal J SI1 61.1 56 403 4.49 4.55 2.76 1040.  
## 8 0.26 Good D VS2 65.2 56 403 3.99 4.02 2.61 1040.  
## 9 0.26 Good D VS1 58.4 63 403 4.19 4.24 2.46 1040.  
## 10 0.290 Very Good H SI2 60.7 60 404 4.33 4.37 2.64 1040.  
## # … with 16,172 more rows

## Extract the test data species to build the confusion matrix

tree.confusion <- table(tree.predictions, diamond.test$price)

MAE (Mean absolute Error) ## calculate Mean Absolute Error (MAE)

maeTree <-mae(model = diamond.tree.model, data=diamond.test)  
maeTree

## [1] 885.491

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)  
}

#Feed in the target and predictors  
target <- "price"  
predictors <- c("carat", "cut", "color", "clarity", "depth", "table", "x", "y", "z")  
# 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 = diamond.train, testing\_data = diamond.test)  
 print(glue::glue("Maxdepth: ",i, "\t MAE: ", mae))  
}

## Maxdepth: 1 MAE: 1730.90585726432  
## Maxdepth: 2 MAE: 1035.38950106321  
## Maxdepth: 3 MAE: 885.491040903657  
## Maxdepth: 4 MAE: 885.491040903657  
## Maxdepth: 5 MAE: 885.491040903657  
## Maxdepth: 6 MAE: 885.491040903657  
## Maxdepth: 7 MAE: 885.491040903657  
## Maxdepth: 8 MAE: 885.491040903657  
## Maxdepth: 9 MAE: 885.491040903657  
## Maxdepth: 10 MAE: 885.491040903657

RANDOM FOREST

diamond.RandomForest <- randomForest(price ~ ., data=diamond.train)  
maeForest <- mae(model = diamond.RandomForest, data=diamond.test)  
maeForest

## [1] 274.9425

diamond.RandomForest

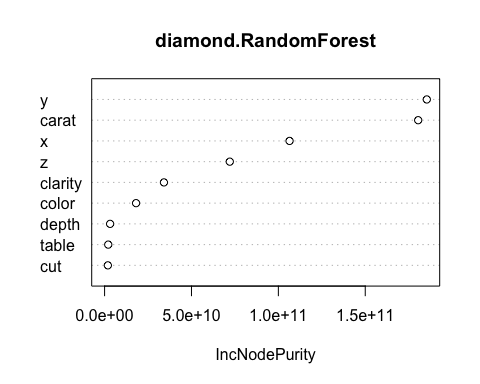
##   
## Call:  
## randomForest(formula = price ~ ., data = diamond.train)   
## Type of random forest: regression  
## Number of trees: 500  
## No. of variables tried at each split: 3  
##   
## Mean of squared residuals: 309318.8  
## % Var explained: 98.07

# What features does the RF think are important?

importance(diamond.RandomForest)

## IncNodePurity  
## carat 180459682587  
## cut 1882570899  
## color 18112198149  
## clarity 34160320948  
## depth 3168600730  
## table 2103473732  
## x 106442060336  
## y 185420070672  
## z 72041275748

varImpPlot(diamond.RandomForest)

 ## MODEL EVALUATION ## Predict test set outcomes, reporting class labels

diamond.rf.predictions <- predict(diamond.RandomForest, diamond.test, type="response")

## calculate the confusion matrix

diamond.rf.confusion <- table(diamond.rf.predictions, diamond.test$price)  
#print(diamond.rf.confusion)

## accuracy

diamond.rf.accuracy <- sum(diag(diamond.rf.confusion)) / sum(diamond.rf.confusion)  
print(diamond.rf.accuracy)

## [1] 0.0001235941

LINEAR MODEL

diamond.Linear <- lm(price ~ ., data=diamond.train)  
maeLinear <- mae(model = diamond.Linear, data=diamond.test)  
maeLinear

## [1] 739.8643

summary(diamond.Linear)

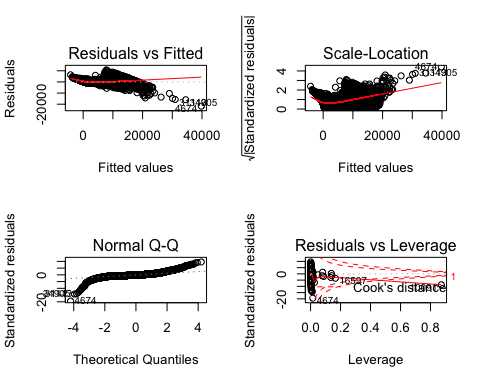
##   
## Call:  
## lm(formula = price ~ ., data = diamond.train)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -21748.4 -586.2 -181.0 372.7 10778.8   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 5786.088 544.769 10.621 < 2e-16 \*\*\*  
## carat 11411.050 60.415 188.877 < 2e-16 \*\*\*  
## cut.L 598.195 27.206 21.987 < 2e-16 \*\*\*  
## cut.Q -310.409 21.763 -14.263 < 2e-16 \*\*\*  
## cut.C 145.888 18.609 7.840 4.64e-15 \*\*\*  
## cut^4 -27.331 14.803 -1.846 0.06485 .   
## color.L -1932.450 20.636 -93.644 < 2e-16 \*\*\*  
## color.Q -684.164 18.763 -36.464 < 2e-16 \*\*\*  
## color.C -158.077 17.543 -9.011 < 2e-16 \*\*\*  
## color^4 34.263 16.157 2.121 0.03396 \*   
## color^5 -89.349 15.235 -5.865 4.54e-09 \*\*\*  
## color^6 -62.573 13.828 -4.525 6.06e-06 \*\*\*  
## clarity.L 4010.867 35.737 112.233 < 2e-16 \*\*\*  
## clarity.Q -1896.340 33.260 -57.015 < 2e-16 \*\*\*  
## clarity.C 926.119 28.497 32.499 < 2e-16 \*\*\*  
## clarity^4 -342.912 22.813 -15.031 < 2e-16 \*\*\*  
## clarity^5 213.096 18.707 11.391 < 2e-16 \*\*\*  
## clarity^6 14.698 16.345 0.899 0.36853   
## clarity^7 93.163 14.446 6.449 1.14e-10 \*\*\*  
## depth -60.478 6.897 -8.769 < 2e-16 \*\*\*  
## table -25.801 3.493 -7.387 1.53e-13 \*\*\*  
## x -1159.577 61.844 -18.750 < 2e-16 \*\*\*  
## y 126.668 40.372 3.137 0.00171 \*\*   
## z -105.675 81.289 -1.300 0.19361   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 1127 on 37734 degrees of freedom  
## Multiple R-squared: 0.9209, Adjusted R-squared: 0.9208   
## F-statistic: 1.909e+04 on 23 and 37734 DF, p-value: < 2.2e-16

coef(diamond.Linear)

## (Intercept) carat cut.L cut.Q cut.C cut^4   
## 5786.08785 11411.04971 598.19501 -310.40933 145.88834 -27.33105   
## color.L color.Q color.C color^4 color^5 color^6   
## -1932.45031 -684.16443 -158.07661 34.26324 -89.34931 -62.57294   
## clarity.L clarity.Q clarity.C clarity^4 clarity^5 clarity^6   
## 4010.86664 -1896.33982 926.11923 -342.91221 213.09596 14.69767   
## clarity^7 depth table x y z   
## 93.16267 -60.47815 -25.80074 -1159.57693 126.66779 -105.67507

## VISUALIZE THE TRAINED MODEL

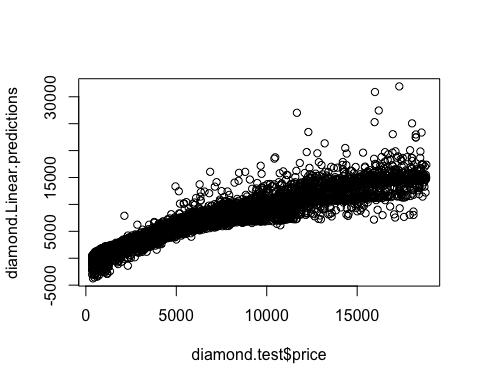
layout(matrix(c(1,2,3,4),2,2)) # set 4 graphs/page   
plot(diamond.Linear)



## MODEL EVALUATION

## make prediction using trained model

diamond.Linear.predictions <- predict(diamond.Linear, diamond.test)  
par(mfrow=c(1,1))  
plot(diamond.test$price, diamond.Linear.predictions)

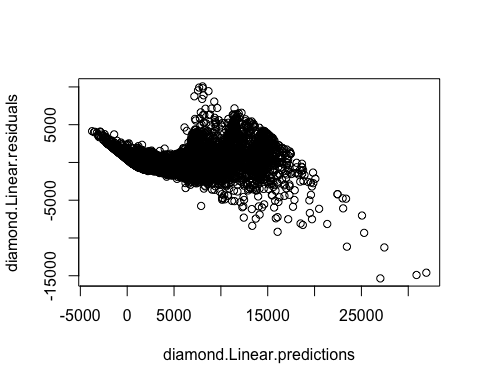


head(diamond.Linear.predictions)

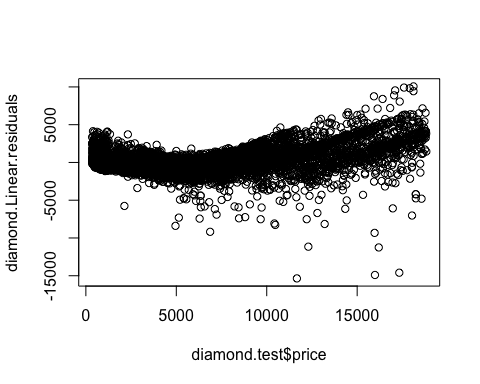
## 1 2 3 4 5 6   
## -3007.5280 -2160.7361 542.4617 360.0183 446.5429 225.2425

## calculate residuals

diamond.Linear.residuals <- diamond.test$price - diamond.Linear.predictions  
plot(diamond.Linear.predictions, diamond.Linear.residuals)



plot(diamond.test$price, diamond.Linear.residuals)



## calculate Root Mean Squared Error (RMSE)

diamond.Linear.rmse <- sqrt(mean(diamond.Linear.residuals^2))  
print(diamond.Linear.rmse)

## [1] 1140.287

## calculate Mean Absolute Error (MAE)

diamond.Linear.mae <- mean(abs(diamond.Linear.residuals))  
print(diamond.Linear.mae)

## [1] 739.8643

## R squared (coefficient of determination)

diamond.ss.tot <- sum((diamond.test$price - mean(diamond.test$price))^2)  
diamond.Linear.ss.res <- sum(diamond.Linear.residuals^2)  
diamond.Linear.r2 <- 1 - diamond.Linear.ss.res / diamond.ss.tot  
print(diamond.Linear.r2)

## [1] 0.9169275

REPORT GENERATION

print('REPORT')

## [1] "REPORT"

print("Decision Tree")

## [1] "Decision Tree"

print(maeTree)

## [1] 885.491

print("Random Forest")

## [1] "Random Forest"

print(maeForest)

## [1] 274.9425

print("Linear Regression")

## [1] "Linear Regression"

print(maeLinear)

## [1] 739.8643

The Random Forest Model clearly did this best as far as mean absolute error.