Linear Regression

Code

Khang Thai, David Park, Jonathan Ho, David Favela

Reading in the Dataset

Hide

```
DiamondDataset <- read.csv("diamonds.csv", header=TRUE)
str(DiamondDataset)
```

```
'data.frame':	53940 obs. of  11 variables:
 $ X      : int  1 2 3 4 5 6 7 8 9 10 ...
 $ carat  : num  0.23 0.21 0.23 0.29 0.31 0.24 0.24 0.26 0.22 0.23 ...
 $ cut    : chr  "Ideal" "Premium" "Good" "Premium" ...
 $ color  : chr  "E" "E" "E" "I" ...
 $ clarity: chr  "SI2" "SI1" "VS1" "VS2" ...
 $ depth  : num  61.5 59.8 56.9 62.4 63.3 62.8 62.3 61.9 65.1 59.4 ...
 $ table  : num  55 61 65 58 58 57 57 55 61 61 ...
 $ price  : int  326 326 327 334 335 336 336 337 337 338 ...
 $ x      : num  3.95 3.89 4.05 4.2 4.34 3.94 3.95 4.07 3.87 4 ...
 $ y      : num  3.98 3.84 4.07 4.23 4.35 3.96 3.98 4.11 3.78 4.05 ...
 $ z      : num  2.43 2.31 2.31 2.63 2.75 2.48 2.47 2.53 2.49 2.39 ...
```

Create train & test sets

Hide

```
library(caret)
set.seed(1234)
i <- sample(1:nrow(diamonds), .8*nrow(diamonds), replace=FALSE)
train <- diamonds[i,]
test <- diamonds[-i,]
```

Hide

```
library(tidyverse)
df <- select(train, c('carat', 'price', 'x', 'y', 'z', 'depth' , 'table'))
head(df)
```

	carat <dbl>	price <int>	x <dbl>	y <dbl>	z <dbl>	depth <dbl>	table <dbl>
40784	0.61	1168	5.37	5.43	3.42	63.4	57.1
40854	0.53	1173	5.21	5.19	3.16	60.8	58.0
41964	0.23	505	3.90	3.93	2.44	62.3	55.0

	carat	price	x	y	z	depth	table
	<dbl>	<int>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
15241	1.33	6118	7.11	7.08	4.35	61.3	57.0
33702	0.30	838	4.30	4.34	2.66	61.6	56.0
35716	0.30	911	4.34	4.31	2.63	60.8	57.0

6 rows

Hide

```
dim(train)
```

```
[1] 43152    11
```

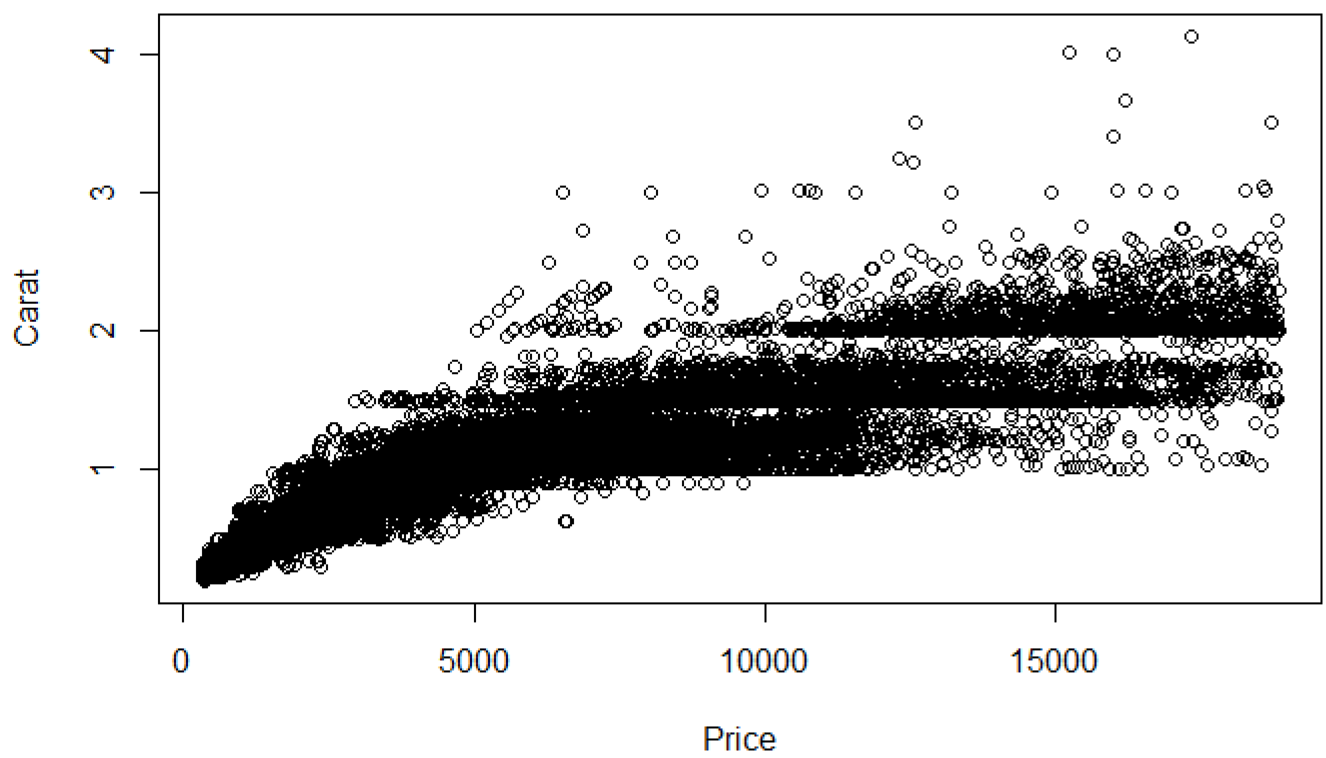
Hide

```
summary(df)
```

```
     carat             price              x                 y                 z               depth      
 Min.   :0.2000   Min.   :  326   Min.   : 0.000   Min.   : 0.000   Min.   : 0.000   Min.   :43.
00
 1st Qu.:0.4000   1st Qu.:  954   1st Qu.: 4.720   1st Qu.: 4.730   1st Qu.: 2.910   1st Qu.:61.
00
 Median :0.7000   Median : 2396   Median : 5.690   Median : 5.710   Median : 3.520   Median :61.
80
 Mean   :0.7978   Mean   : 3931   Mean   : 5.731   Mean   : 5.735   Mean   : 3.539   Mean   :61.
75
 3rd Qu.:1.0400   3rd Qu.: 5315   3rd Qu.: 6.540   3rd Qu.: 6.530   3rd Qu.: 4.030   3rd Qu.:62.
50
 Max.   :4.1300   Max.   :18823   Max.   :10.140   Max.   :58.900   Max.   :31.800   Max.   :79.
00
     table      
 Min.   :43.00  
 1st Qu.:56.00  
 Median :57.00  
 Mean   :57.46  
 3rd Qu.:59.00  
 Max.   :95.00  
```

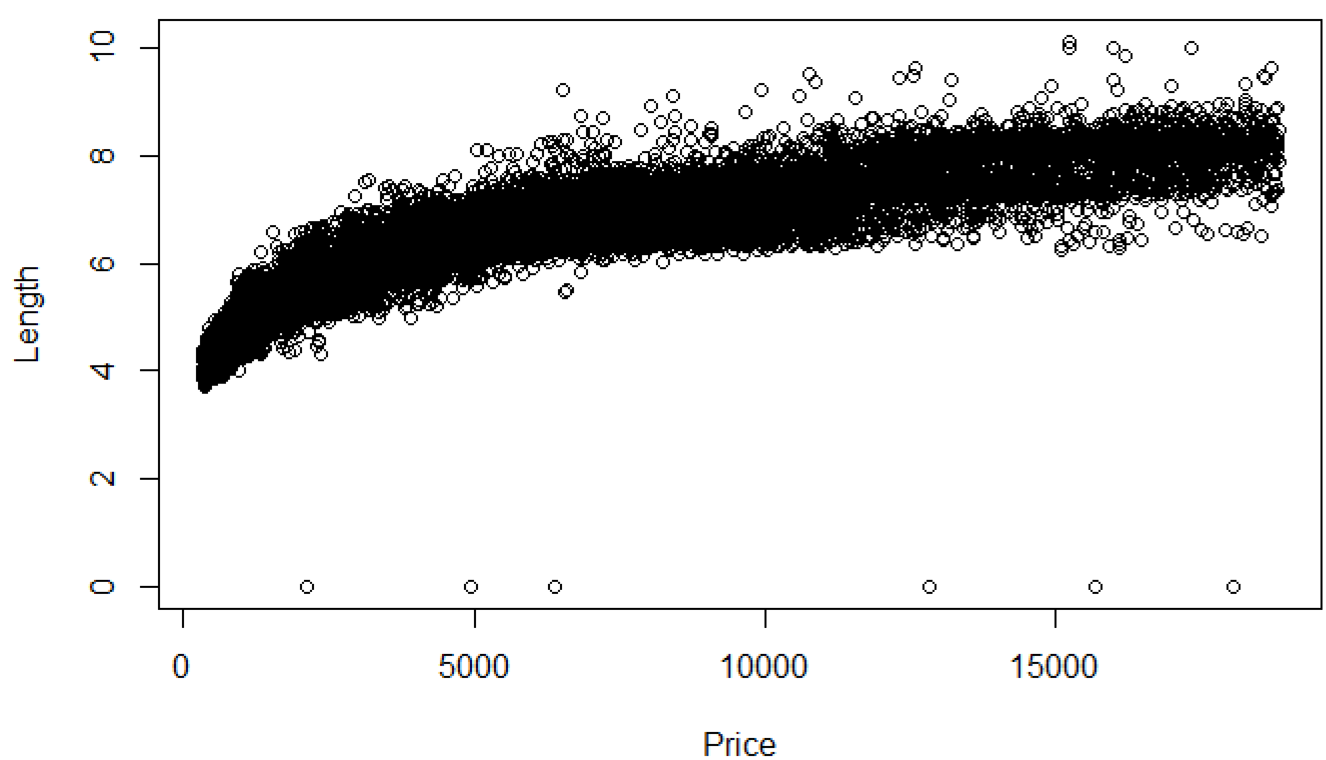
Hide

```
plot(train$price, train$carat, xlab = "Price", ylab = "Carat")
```



Hide

```
plot(train$price, train$x, xlab = "Price", ylab = "Length")
```



Plotting the data for Linear Regression

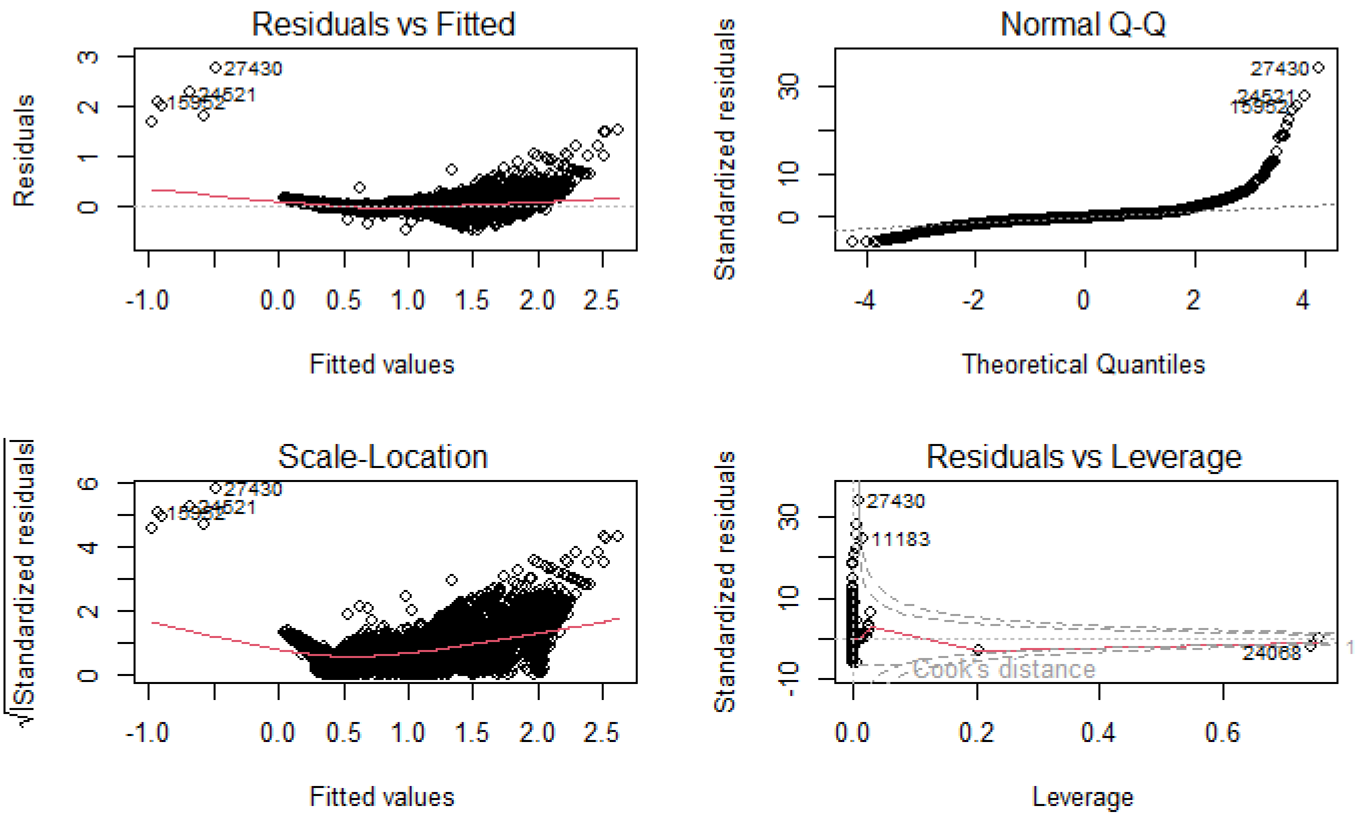
Hide

```
df <- select(train, c('carat', 'price', 'x', 'y', 'z', 'depth' , 'table'))
lm1 <- lm(carat ~., data = df)
summary(lm1)
```

```

Call:
lm(formula = carat ~ ., data = df)

Residuals:
     Min       1Q   Median       3Q      Max 
-0.48009 -0.03858 -0.00679  0.03540  2.73539 

Coefficients:
              Estimate Std. Error  t value Pr(>|t|)    
(Intercept) -2.496e+00  2.476e-02 -100.776  < 2e-16 ***
price        3.249e-05  2.102e-07  154.619  < 2e-16 ***
x            3.025e-01  2.034e-03  148.682  < 2e-16 ***
y            6.608e-03  1.387e-03    4.765 1.89e-06 ***
z           -1.053e-03  2.460e-03   -0.428    0.669    
depth        1.840e-02  3.205e-04   57.398  < 2e-16 ***
table        4.555e-03  1.871e-04   24.342  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.08098 on 43145 degrees of freedom
Multiple R-squared:  0.9707,    Adjusted R-squared:  0.9707 
F-statistic: 2.382e+05 on 6 and 43145 DF,  p-value: < 2.2e-16
```

Hide

```
par(mfrow=c(2,2))
plot(lm1)
```



Finding the correlation and MSE of Linear Regression

Hide

```
lm2 <- lm(carat~depth+table+price+x+y, data = train)
pred <- predict(lm2, newdata = test)
cor_lm <- cor(pred, test$carat)
mse_lm <- mean((pred - test$carat)^2)
print(paste("Cor = ", cor_lm))
```

```
[1] "Cor =  0.9828718138342"
```

Hide

```
print(paste("MSE = ", mse_lm))
```

```
[1] "MSE =  0.00776251483958306"
```

kNN regression and finding the Correlation and MSE

Hide

```
library(caret)
fit <- knnreg(train[,6:10], train[,2],k = 3)
knnPredict <- predict(fit, test[,6:10])
cor_knn1 <- cor(knnPredict, test$carat)
mse_knn1 <- mean((knnPredict - test$carat)^2)
print(paste("Cor = ", cor_knn1))
```

```
[1] "Cor =  0.943032710933997"
```

Hide

```
print(paste("MSE = ", mse_knn1))
```

```
[1] "MSE =  0.025373977899827"
```

Scaling the data for kNN and finding the Correlation and MSE

Hide

```
train_scaled <- train[,6:10]
means <- sapply(train_scaled, mean)
stdvs <- sapply(train_scaled, sd)
train_scaled <- scale(train_scaled, center = means, scale = stdvs)
test_scaled <- scale(test[,6:10], center = means, scale = stdvs)

fit <- knnreg(train_scaled, train$carat,k = 3)
knnPredict <- predict(fit, test_scaled)
cor_knn2 <- cor(knnPredict, test$carat)
mse_knn2 <- mean((knnPredict - test$carat)^2)
print(paste("Cor = ", cor_knn2))
```

```
[1] "Cor =  0.997076573839975"
```

Hide

```
print(paste("MSE = ", mse_knn2))
```

```
[1] "MSE =  0.00138048146345734"
```

Finding the best K value

Hide

```
cor_k <- rep(0,20)
mse_k <- rep(0,20)
i <- 1
for (k in seq(1,39,2)) {
  fit_k <- knnreg(train_scaled, train$carat, k = k)
  pred_k <- predict(fit_k, test_scaled)
  cor_k[i] <- cor(pred_k, test$carat)
  mse_k[i] <- mean((pred_k - test$carat)^2)
  print(paste("k= ", k, cor_k[i], mse_k[i]))
  i <- i+1
}
```

```
[1] "k=  1 0.996964190356735 0.00139515202076381"
[1] "k=  3 0.997076573839975 0.00138048146345734"
[1] "k=  5 0.996618950418704 0.0016122602583117"
[1] "k=  7 0.996379891095654 0.00173232413081334"
[1] "k=  9 0.996157277573308 0.00184397995372087"
[1] "k=  11 0.995937653837363 0.00195237696592679"
[1] "k=  13 0.995688514057629 0.00207543846520288"
[1] "k=  15 0.995603005064564 0.00212299435342282"
[1] "k=  17 0.995419125160753 0.00221422045914682"
[1] "k=  19 0.995174658059484 0.00233157229987158"
[1] "k=  21 0.994999188389399 0.00241968243948665"
[1] "k=  23 0.994821063995 0.00250873528165421"
[1] "k=  25 0.99468371918514 0.00257777393014404"
[1] "k=  27 0.99461992705823 0.00261118499060813"
[1] "k=  29 0.994523449714852 0.00266148413470744"
[1] "k=  31 0.994414660790092 0.00271638398131546"
[1] "k=  33 0.994339650273162 0.00275390881048127"
[1] "k=  35 0.994245986914347 0.0028003896936636"
[1] "k=  37 0.994138206074497 0.00285387264795795"
[1] "k=  39 0.994057126150063 0.00289398936935071"
```

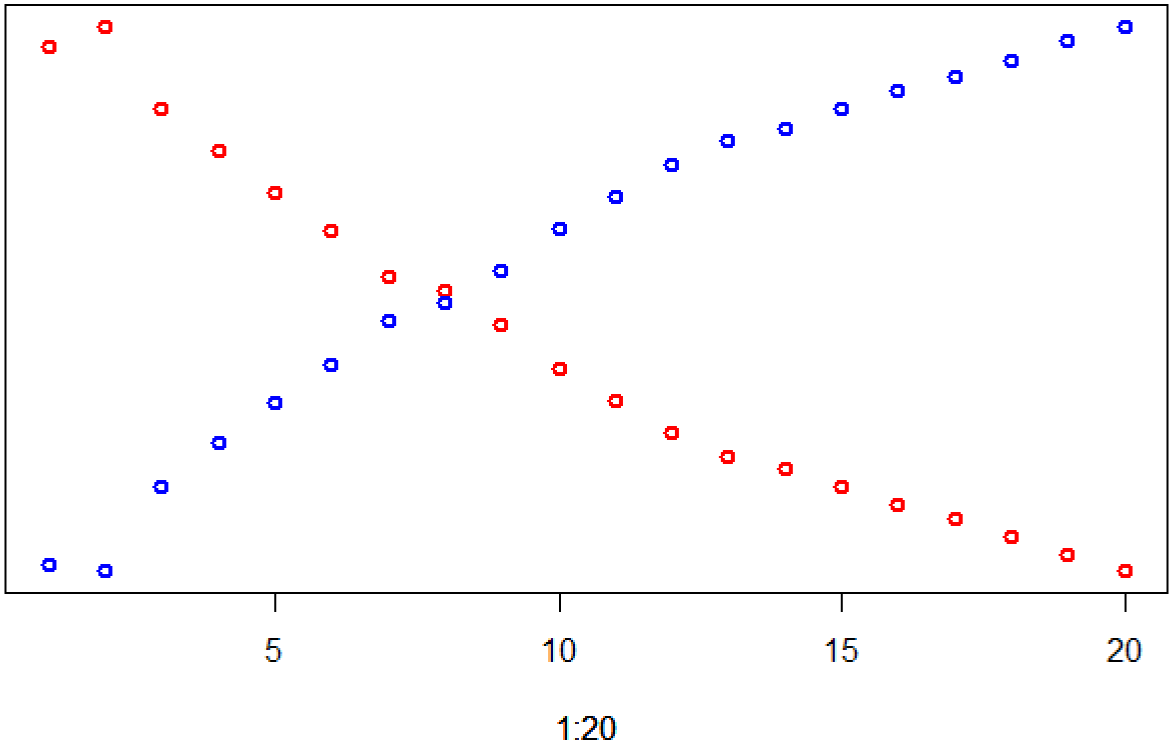
Plotting Knn

Hide

```
plot(1:20, cor_k, lwd = 2, col='red', ylab = "", yaxt='n')
par(new=TRUE)
```

Hide

```
plot(1:20, mse_k, lwd=2, col='blue', labels=FALSE, ylab="", yaxt='n')
```



Stating the best K

Hide

```
print(paste("Best K in MSE = ", which.min(mse_k)))
```

```
[1] "Best K in MSE =  2"
```

Hide

```
print(paste("Best K in Cor = ", which.max(cor_k)))
```

```
[1] "Best K in Cor =  2"
```

Decision Tree

Hide

```
library(tree)
tree1 <- tree(carat~depth+table+price+x+y, data = train)
summary(tree1)
```

```
Regression tree:
tree(formula = carat ~ depth + table + price + x + y, data = train)
Variables actually used in tree construction:
[1] "x"
Number of terminal nodes:  7
Residual mean deviance:  0.00609 = 262.8 / 43140
Distribution of residuals:
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-0.533300 -0.035720 -0.007861  0.000000  0.034280  2.027000
```

Finding the Correlation and RMSE of the Decision Tree

Hide

```
pred <- predict(tree1, newdata = test)
cor_tree <- cor(pred, test$carat)
rmse_tree <- sqrt(mean((pred-test$carat)^2))

print(paste('Correlation: ', cor_tree))
```

```
[1] "Correlation:  0.983105771647537"
```

Hide

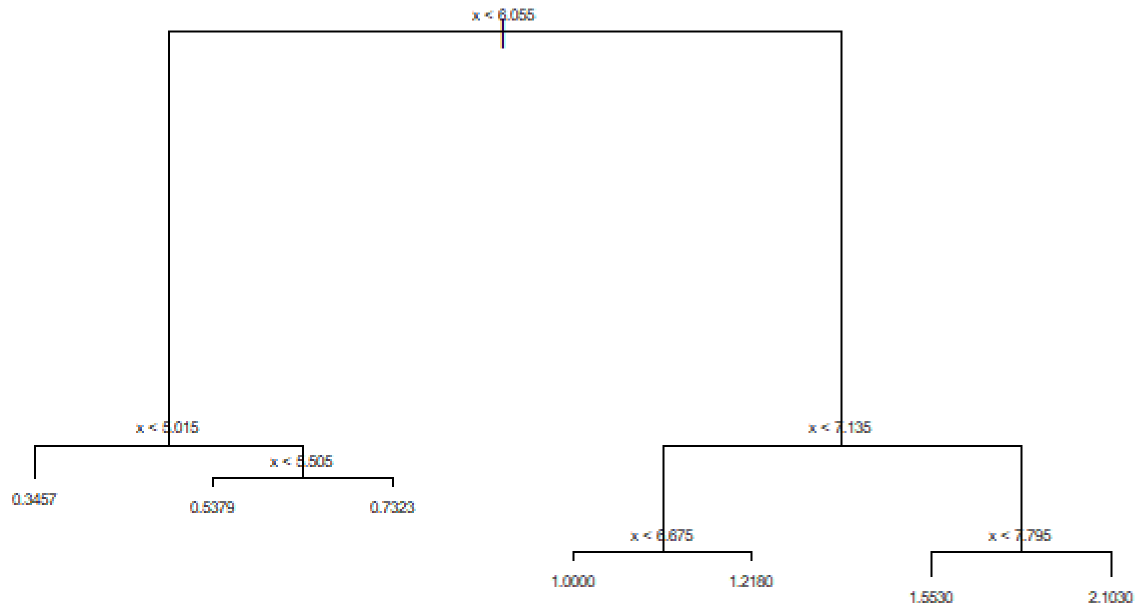
```
print(paste('RMSE: ', rmse_tree))
```

```
[1] "RMSE:  0.0876502208589375"
```

Plotting the Tree

Hide

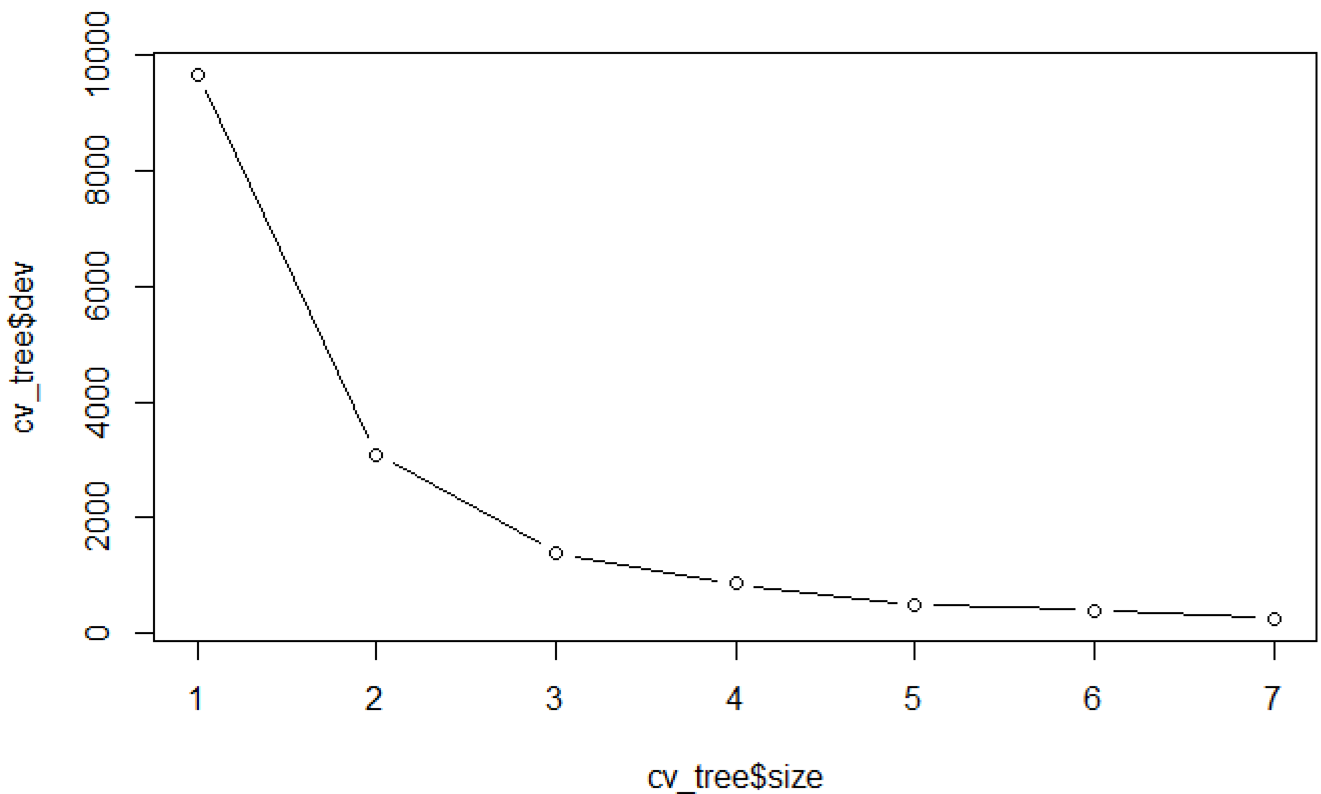
```
plot(tree1)
text(tree1, cex=0.5,pretty=0)
```



Plotting the Cross Validation

Hide

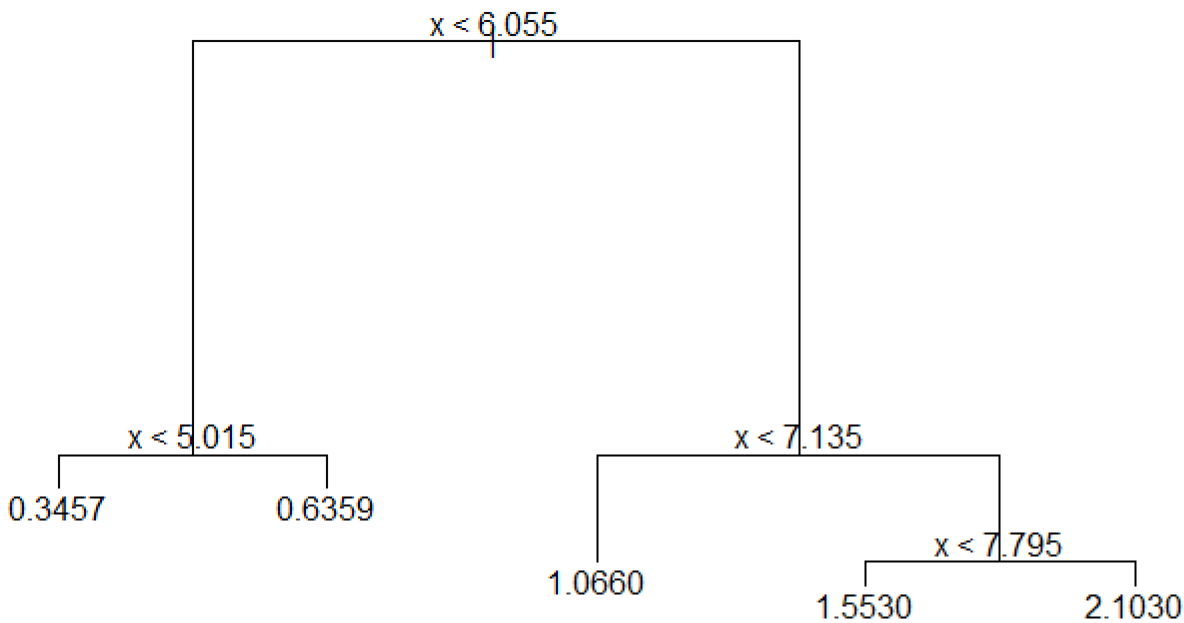
```
cv_tree <- cv.tree(tree1)
plot(cv_tree$size, cv_tree$dev, type ='b')
```



Plotting the Pruned tree

Hide

```
tree_pruned <- prune.tree(tree1, best = 5)
plot(tree_pruned)
text(tree_pruned, pretty=0)
```



Finding the Correlation and RMSE of pruned tree

Hide

```
pred_pruned <- predict(tree_pruned, newdata = test)
cor_pruned <- cor(pred_pruned, test$carat)
rmse_pruned <- sqrt(mean((pred_pruned-test$carat)^2))
print(paste("Cor of pruned tree = ", cor_pruned))
```

```
[1] "Cor of pruned tree =  0.971153870023846"
```

Hide

```
print(paste("RMSE of pruned tree = ", rmse_pruned))
```

```
[1] "RMSE of pruned tree =  0.11412531058094"
```

Comparing the Results

Using Linear Regression, we achieved a 98% correlation between Carat and Price, X, Y, Z, Table, and Depth of the diamond. Using kNN we were able to achieve a 94% correlation and after we scaled it, we got a 99% accuracy between the Carat and the same predictors. Using Decision Trees, we achieved a 98% correlation and after pruning the tree we got a 97% correlation. We used the same test and train data set for all 3 tests. In the end it seems that using kNN and scaling it is the best predictor for our diamonds data set.

How Results Were Achieved?

According to the first plot, we can see that for the most part aside from a few outliers, the price of a diamond will increase as the carat increases. Other factors might be the reason for the price and carat alone is not the main reason for price. In the second plot, we see that Length of the diamond also influence the price so if both of these were predictors, the accuracy of the price prediction will be better. In kNN we used carat, table, length(x), width(y), depth(z), and depth total percentage to determine the price. As we added more predictors, the correlation was lower, but after we scaled it, the correlation was higher than the linear regression correlation. We used decision trees as well but was unable to achieve similar results and it seems that decision trees lowered the correlation predictions even when pruned. I think that the reason the tree failed to come up with a better correlation is because there was not a categorical way for the data to be split without having a huge tree.