Regression

Ruben Mathew

2023-02-14

- What is Linear Regression?
- Set up
- Separate out Training and Test Data
- Data Exploration on Training Data
- Train Model and Predict
- Plotting Residuals
- Multiple Linear Regression
- Improved Linear Regression Model
- Results

What is Linear Regression?

Linear Regression is a technique used to find a relationship between two variables. It tries to find a line of best fit which properly describes the relationship. It tries to find a linear relationship which means it is highly biased and tends to underfit the data. However if the relationship is somewhat linear it can provide a good insight into the strength of the correlation between the two variables.

Set up

Here we reset the environment so that we have a clean slate to work with. We load in the diamonds.csv file. The data was found here (https://www.kaggle.com/datasets/shivam2503/diamonds) on Kaggle.

```
rm(list = ls()) # Reset Environment
df <- read.csv("diamonds.csv")
df$cut <- factor(df$cut, levels = c("Fair", "Good", "Very Good", "Premium", "Ideal"))
```

Separate out Training and Test Data

Here we partition the data into training and test data. We do this to more accurately assess the model and how well it fits to data it hasn't seen before.

```
set.seed(31415)
i <- sample(1:nrow(df), .8*nrow(df), replace=FALSE)
train <- df[i,]
test <- df[-i,]
```

Data Exploration on Training Data

Here we use a few of R's built-in functions to "explore" the data. The head and tail functions show us a few rows from the front and back of the training data (just to see what a few rows look like). The dim and str functions give us some more information on the structure of the data (dimensions and column names/types). Finally the summary function gives us a summary of the data by column (shows min/max, mean, median, etc.). We also use a couple of functions to graph out the relationships between a few of the different columns.

```
# 5 Functions
head(train)
```

	X	carat	cut	color	clarity	depth	table	price	x
	<int>	<dbl>	<fct>	<chr>	<chr>	<dbl>	<dbl>	<int>	<dbl>
22003	22003	1.31	Ideal	G	VS1	62.2	56.0	10071	7.05
32120	32120	0.34	Very Good	E	VS1	62.3	53.0	784	4.47
23459	23459	1.71	Ideal	I	VS2	60.5	56.0	11455	7.71
26220	26220	2.01	Very Good	E	SI2	63.3	61.0	15618	7.86
47523	47523	0.44	Ideal	E	VVS1	61.4	55.0	1868	4.92
5416	5416	1.02	Premium	G	SI2	62.3	58.1	3822	6.41

6 rows | 1-10 of 12 columns

```
tail(train)
```

	X	carat	cut	color	clarity	depth	table	price	x
	<int>	<dbl>	<fct>	<chr>	<chr>	<dbl>	<dbl>	<int>	<dbl>
12929	12929	1.04	Very Good	E	SI1	63.5	56	5381	6.41
5522	5522	0.60	Ideal	D	VVS1	62.3	55	3850	5.38
3704	3704	1.00	Ideal	E	SI2	62.9	56	3450	6.32
8725	8725	0.31	Good	H	VS2	63.8	56	586	4.33
37242	37242	0.50	Good	F	SI1	64.3	57	975	5.03
33830	33830	0.32	Ideal	E	VVS2	61.4	54	843	4.44

6 rows | 1-10 of 12 columns

```
dim(train)
```

```
## [1] 43152    11
```

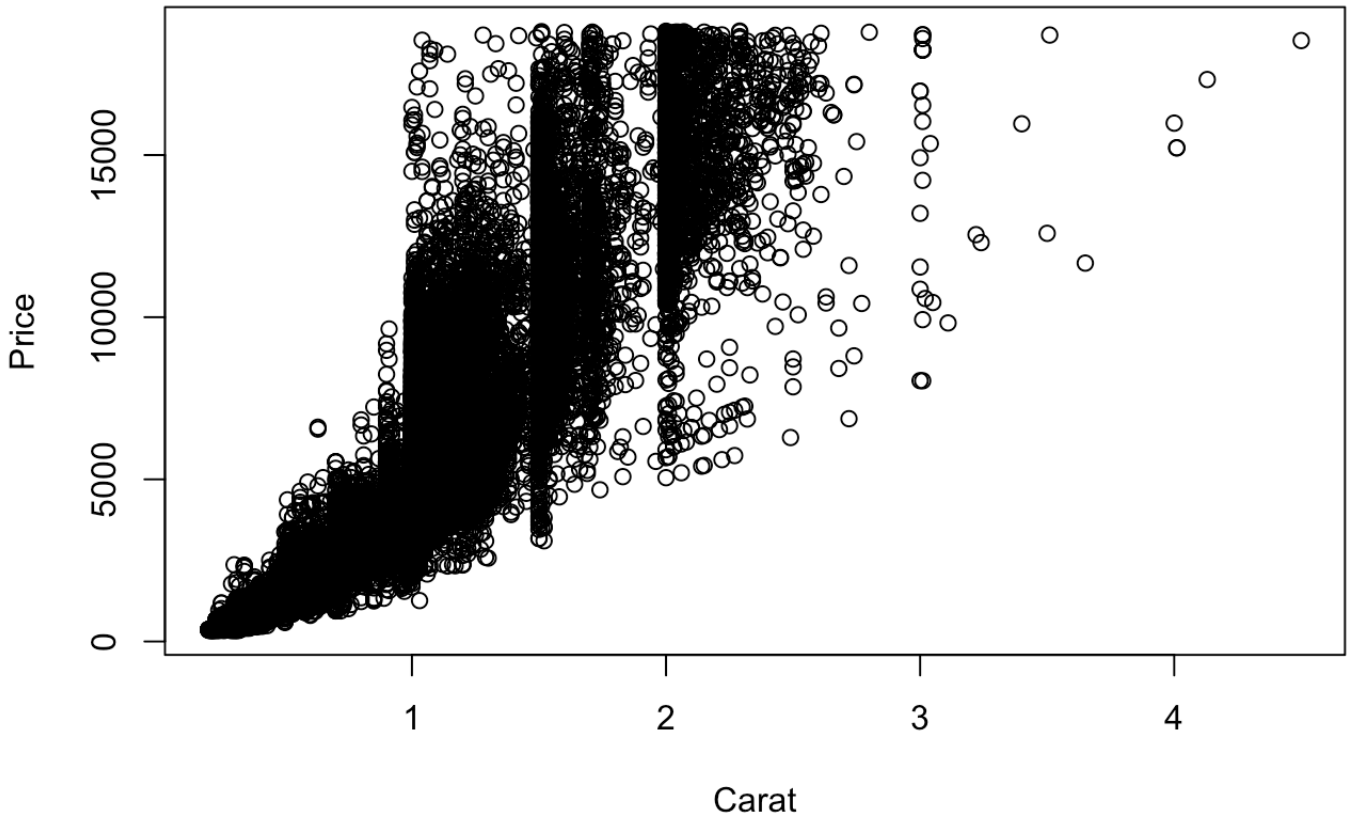
```
str(train)
```

```
## 'data.frame':    43152 obs. of  11 variables:
##  $ X      : int  22003 32120 23459 26220 47523 5416 19732 38153 23522 9498 ...
##  $ carat  : num  1.31 0.34 1.71 2.01 0.44 1.02 1.24 0.75 1.58 1.07 ...
##  $ cut    : Factor w/ 5 levels "Fair","Good",..: 5 3 5 3 5 4 5 2 5 3 ...
##  $ color  : chr  "G" "E" "I" "E" ...
##  $ clarity: chr  "VS1" "VS1" "VS2" "SI2" ...
##  $ depth  : num  62.2 62.3 60.5 63.3 61.4 62.3 61.6 64 61.7 60.5 ...
##  $ table  : num  56 53 56 61 55 58.1 56 59 53 61 ...
##  $ price  : int  10071 784 11455 15618 1868 3822 8299 1013 11526 4609 ...
##  $ x      : num  7.05 4.47 7.71 7.86 4.92 6.41 6.88 5.78 7.52 6.55 ...
##  $ y      : num  7.01 4.52 7.73 7.8 4.95 6.38 6.91 5.62 7.53 6.61 ...
##  $ z      : num  4.37 2.8 4.67 4.96 3.03 4 4.25 3.66 4.65 3.98 ...
```

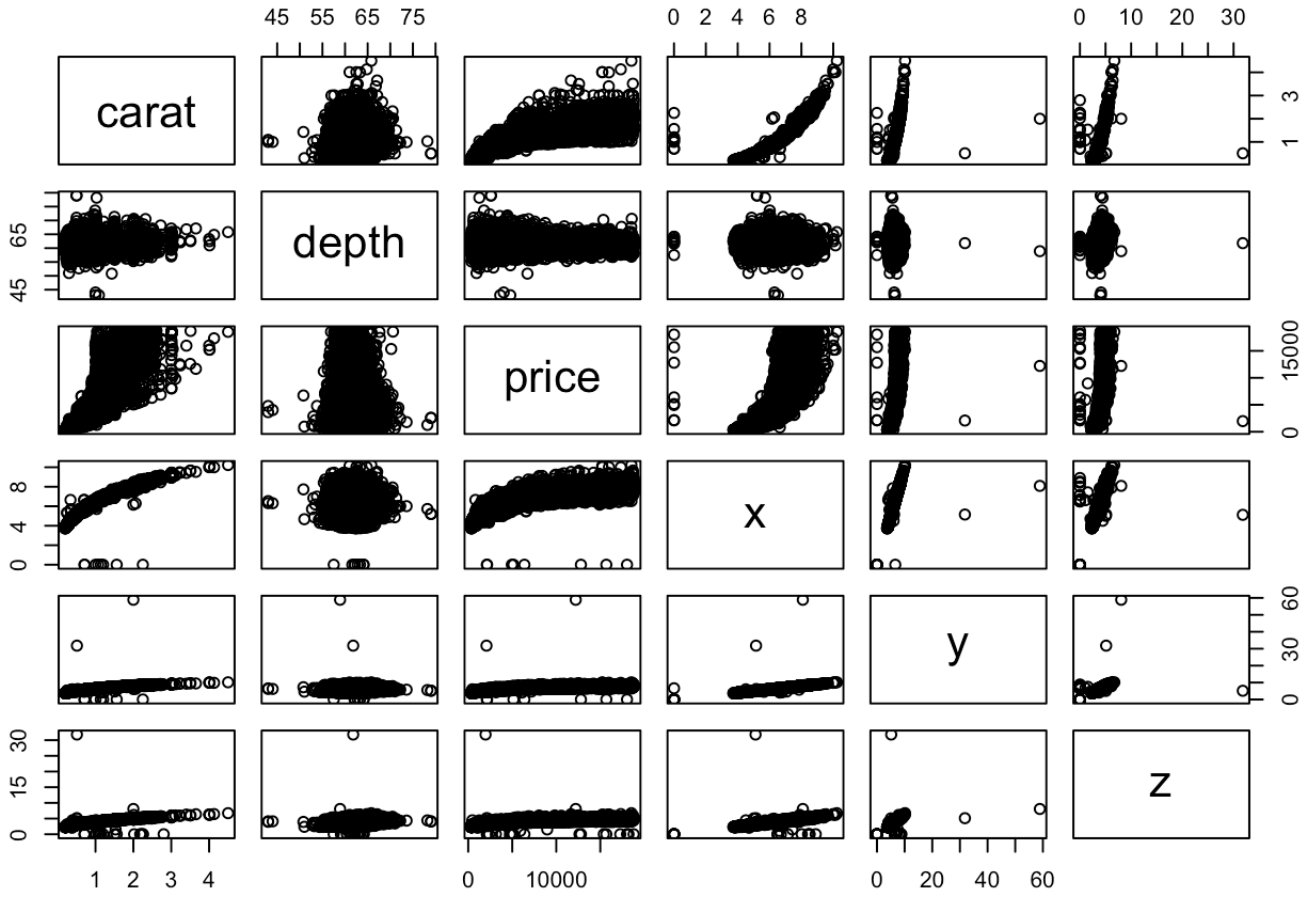
```
summary(train)
```

```
##        X             carat               cut           color
##  Min.   :    2   Min.   :0.2000   Fair     : 1304   Length:43152
##  1st Qu.:13461   1st Qu.:0.4000   Good     : 3949   Class :character
##  Median :26918   Median :0.7000   Very Good: 9607   Mode  :character
##  Mean   :26961   Mean   :0.7984   Premium  :11013
##  3rd Qu.:40452   3rd Qu.:1.0400   Ideal    :17279
##  Max.   :53940   Max.   :4.5000
##    clarity             depth           table           price
##  Length:43152       Min.   :43.00   Min.   :43.00   Min.   :  326
##  Class :character   1st Qu.:61.00   1st Qu.:56.00   1st Qu.:  949
##  Mode  :character   Median :61.80   Median :57.00   Median : 2418
##                     Mean   :61.75   Mean   :57.45   Mean   : 3937
##                     3rd Qu.:62.50   3rd Qu.:59.00   3rd Qu.: 5331
##                     Max.   :79.00   Max.   :95.00   Max.   :18823
##        x                y               z
##  Min.   : 0.000   Min.   : 0.000   Min.   : 0.00
##  1st Qu.: 4.710   1st Qu.: 4.720   1st Qu.: 2.91
##  Median : 5.700   Median : 5.710   Median : 3.53
##  Mean   : 5.732   Mean   : 5.736   Mean   : 3.54
##  3rd Qu.: 6.540   3rd Qu.: 6.540   3rd Qu.: 4.04
##  Max.   :10.230   Max.   :58.900   Max.   :31.80
```

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




```
pairs(train[,c(2,6,8,9,10,11)]) # carat, depth %, price, x(length), y(width), z(depth
)
```



Train Model and Predict

Here we train the model using the training data. We can make a summary of this model and see then use the built-in predict function to calculate other statistics such as correlation, mean squared error and root mean squared error.

```
lm1 <- lm(price~carat, data=train)

summary(lm1)
```

```
##
## Call:
## lm(formula = price ~ carat, data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -14366.4   -805.3    -21.5    534.3  12732.2
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2249.02      14.69  -153.1   <2e-16 ***
## carat        7748.88      15.83   489.5   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1557 on 43150 degrees of freedom
## Multiple R-squared:  0.8474, Adjusted R-squared:  0.8474
## F-statistic: 2.396e+05 on 1 and 43150 DF,  p-value: < 2.2e-16
```

```
pred1 <- predict(lm1, newdata=test)

correlation1 <- cor(pred1, test$carat)
mse1 <- mean((pred1 - test$carat)^2)
rmse1 <- sqrt(mse1)

print(paste("correlation: ", correlation1))
```

```
## [1] "correlation:  1"
```

```
print(paste("mse: ", mse1))
```

```
## [1] "mse:  28991593.9807407"
```

```
print(paste("rmse: ", rmse1))
```

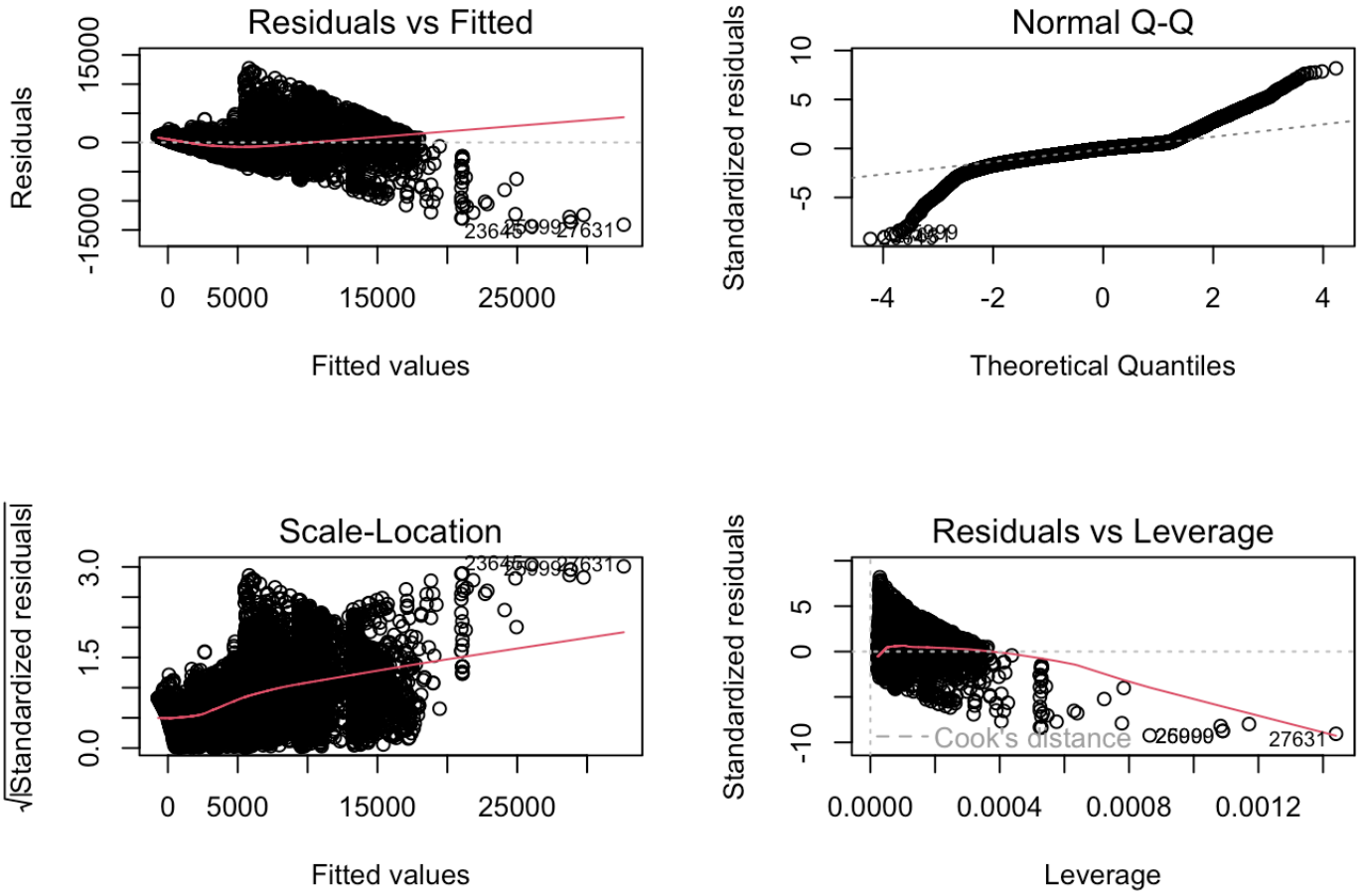
```
## [1] "rmse:  5384.3842712738"
```

This summary gives us good information on the correlation between carat and price. For instance since R-Square is about 0.847, we can assume that these are strongly positively correlated meaning if the carat value goes up, so does price (this does make sense logically). A more interesting statistic is the given slope, which we can assume is a good estimation due to the low p-value. This means that according to our model, for every 1 carat increase, the diamond's price goes up $7748.88.

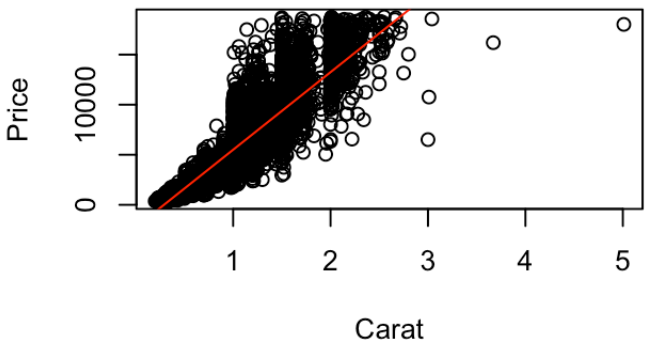
Plotting Residuals

By using R's built in plot function, we can get 4 different visual representations that tell us information on this data set.

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




```
plot(test$carat,test$price,xlab="Carat", ylab="Price")
abline(lm1, col='red')
```



Residuals vs Fitted

This plot shows us that our model for the most part does capture most of the variation in the data since the red line is almost horizontal. We can see however its a little shakier with the higher numbers. This makes sense due to the less data provided for those higher values.

Normal Q-Q

This plot shows us that our residuals are pretty normally distributed, due to the almost straight diagonal line plotted.

Scale-Location

This plot shows us that our data is not homoscedastic because the red line is more diagonal than horizontal

Residuals vs Leverage

This plot shows us there are definitely outliers that affects the model such as observation 27631

Multiple Linear Regression

Here is another regression model using an extra predictor of the type of cut. This is a factor of 5 different types (Fair, Good, Very Good, Premium, Ideal) describing the quality of the cut.

```
lm2 <- lm(price~carat+cut, data=train)

summary(lm2)
```

```
##
## Call:
## lm(formula = price ~ carat + cut, data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13873.6   -792.0    -42.7    518.4  12718.8
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -3879.16      45.11  -86.00   <2e-16 ***
## carat         7861.85      15.71  500.38   <2e-16 ***
## cutGood       1124.97      48.60   23.15   <2e-16 ***
## cutVery Good  1526.07      44.97   33.93   <2e-16 ***
## cutPremium    1449.74      44.54   32.55   <2e-16 ***
## cutIdeal      1816.22      43.93   41.35   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1519 on 43146 degrees of freedom
## Multiple R-squared:  0.8548, Adjusted R-squared:  0.8548
## F-statistic: 5.079e+04 on 5 and 43146 DF,  p-value: < 2.2e-16
```

```
pred2 <- predict(lm2, newdata=test)

correlation2 <- cor(pred2, test$carat)
mse2 <- mean((pred2 - test$carat)^2)
rmse2 <- sqrt(mse2)

print(paste("correlation: ", correlation2))
```

```
## [1] "correlation:  0.995928137207581"
```

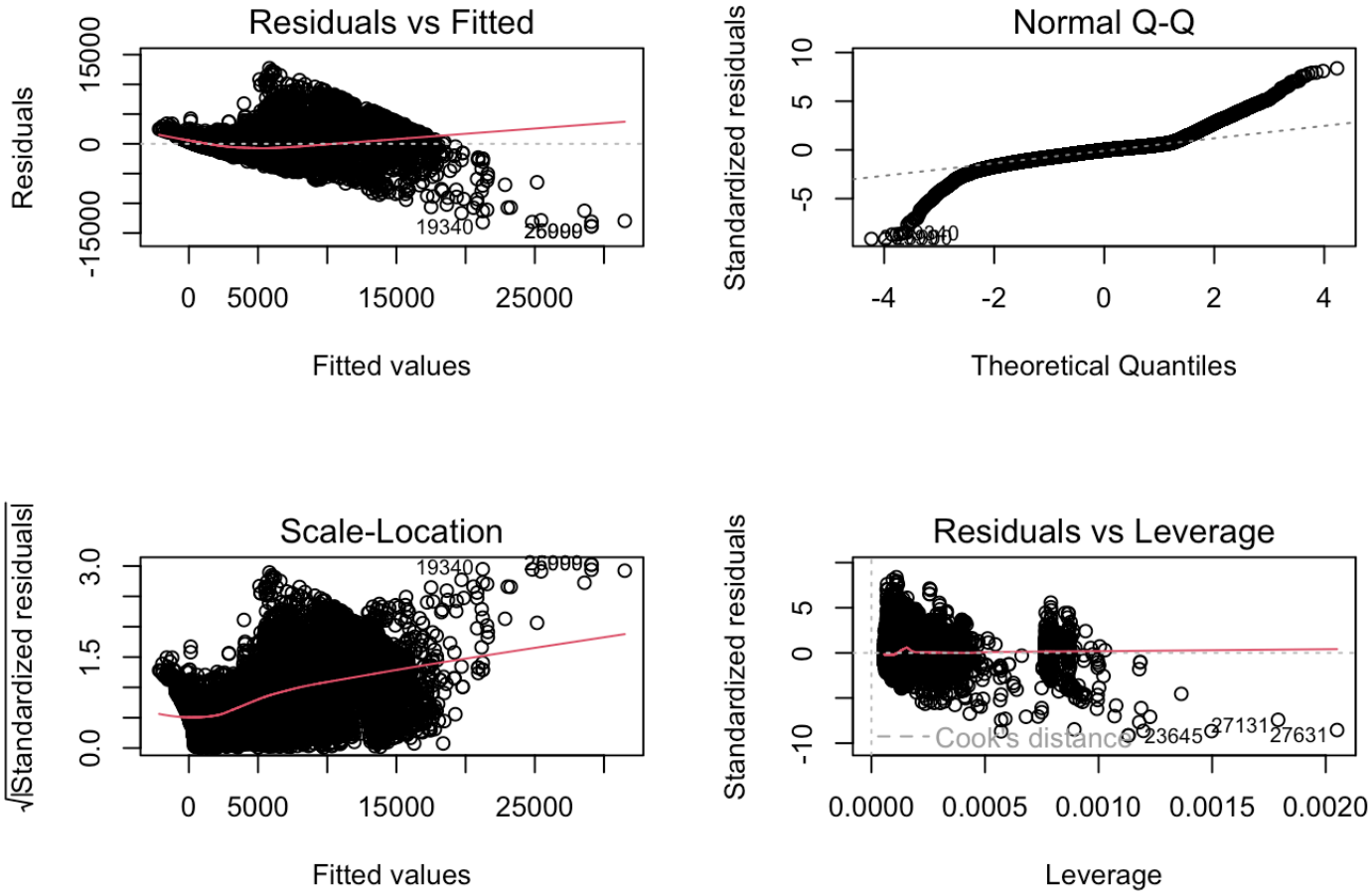
```
print(paste("mse: ", mse2))
```

```
## [1] "mse:  29070220.640737"
```

```
print(paste("rmse: ", rmse2))
```

```
## [1] "rmse:  5391.68068794295"
```

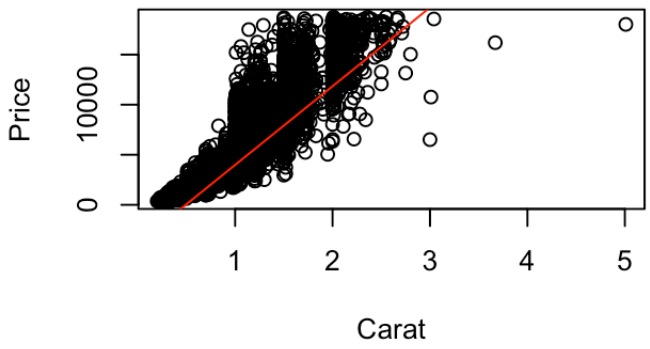
```
par(mfrow=c(2,2))
plot(lm2)
```

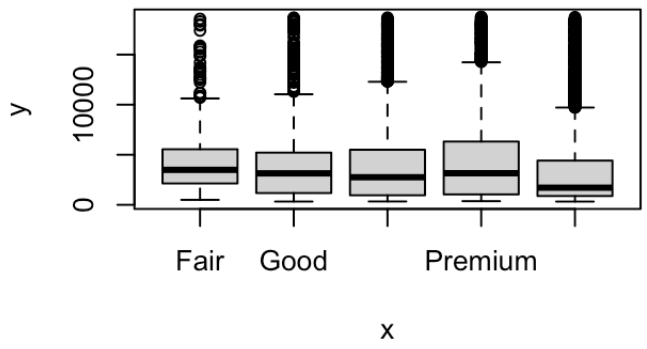



```
plot(test$carat,test$price,xlab="Carat", ylab="Price")
abline(lm2, col='red')
```

```
## Warning in abline(lm2, col = "red"): only using the first two of 6 regression
## coefficients
```

```
plot(test$cut, test$price)
```



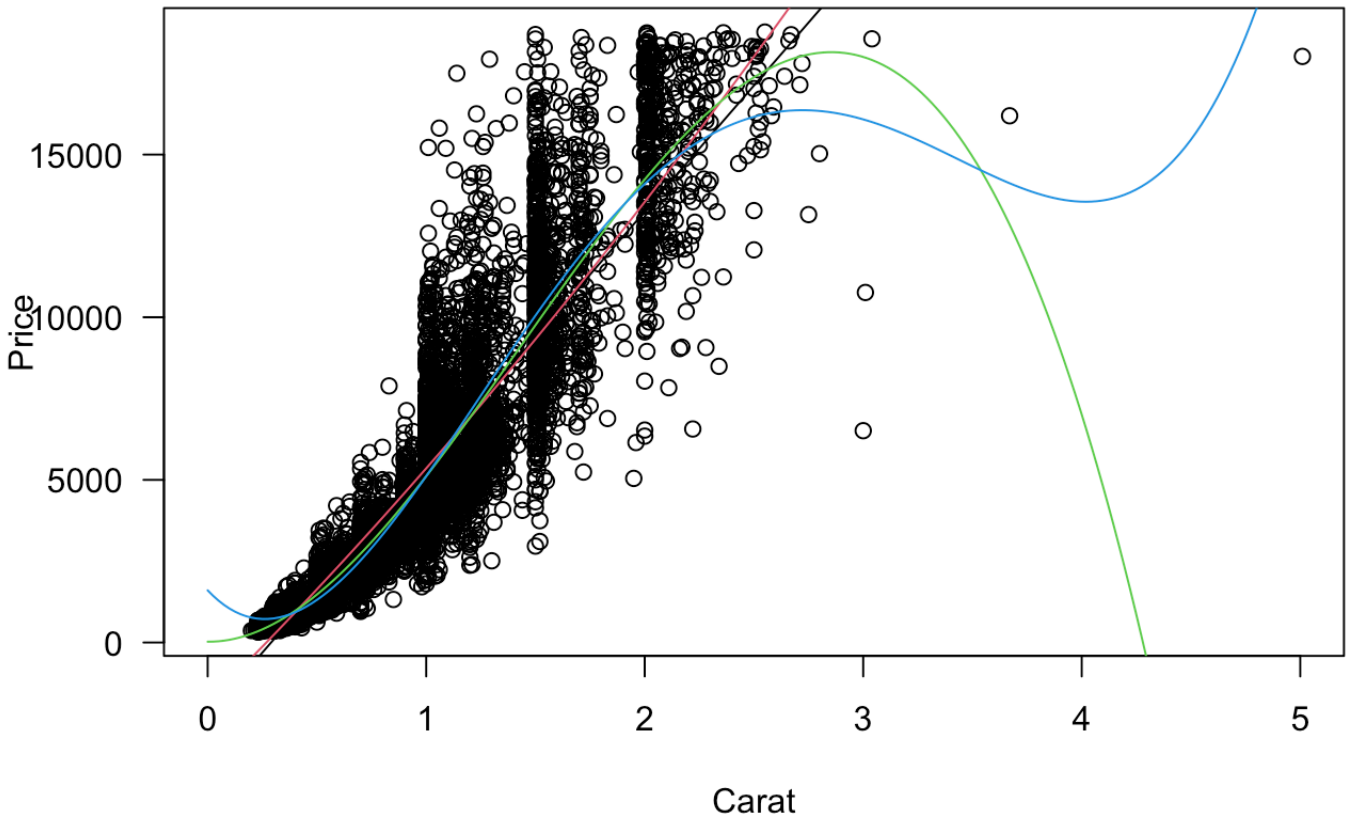


We can see that this is a slight improvement in the model because, R-squared increased to .855 while still maintaining a low p-value. We also get some insight to how the cut may affect price in consideration with carat.

Improved Linear Regression Model

In an attempt to improve the model, we can try using polynomial regression here. This tries to fit the relationship to a less straight line in order to account for other factors.

```
plot(test$carat, test$price, xlab="Carat", ylab="Price", las = 1, xlim = c(0, 5))
d <- seq(0, 5, length.out = 200)
for(degree in 1:4) {
  fm <- lm(price ~ poly(carat, degree), data = train)
  assign(paste("diamonds", degree, sep = "."), fm)
  lines(d, predict(fm, data.frame(carat = d)), col = degree)
}
```




```
anova(diamonds.1, diamonds.2, diamonds.3, diamonds.4)
```

	Res.Df <dbl>	RSS <dbl>	Df <dbl>	Sum of Sq <dbl>	F <dbl>	Pr(>F) <dbl>
1	43150	104584289997	*NA*	*NA*	*NA*	*NA*
2	43149	103500509512	1	1083780485	516.7571	1.003027e-113
3	43148	92830208367	1	10670301145	5087.7036	0.000000e+00
4	43147	90491019610	1	2339188757	1115.3480	1.845379e-241

4 rows

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

In this case we can see that the Polynomial model is not a better model for predicting the data. However we can see some interesting functions used for prediction, though with higher degrees, it seems to try to overfit the data with the outliers.

Results

From our models, it seems like the multiple linear regression model is the best for the dataset. It has a higher R-squared value and seems more accurate according to the residuals. It also has low mse/rmse and a close to 1 correlation. This makes sense because a simple linear regression does not take into account other factors like cut, but the polynomial regression overfits the data due to the higher amount of outliers and variance in values.

```
print(paste("model 1 correlation: ", correlation1))
```

```
## [1] "model 1 correlation:  1"
```

```
print(paste("model 1 mse: ", mse1))
```

```
## [1] "model 1 mse:  28991593.9807407"
```

```
print(paste("model 1 rmse: ", rmse1))
```

```
## [1] "model 1 rmse:  5384.3842712738"
```

```
print(paste("model 2 correlation: ", correlation2))
```

```
## [1] "model 2 correlation:  0.995928137207581"
```

```
print(paste("model 2 mse: ", mse2))
```

```
## [1] "model 2 mse:  29070220.640737"
```

```
print(paste("model 2 rmse: ", rmse2))
```

```
## [1] "model 2 rmse:  5391.68068794295"
```

```
anova(lm1, lm2)
```

	Res.Df	RSS	Df	Sum of Sq	F	Pr(>F)
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	43150	104584289997	*NA*	*NA*	*NA*	*NA*
2	43146	99525023042	4	5059266955	548.3222	0

2 rows

The correlation of model 1 is 1, and the correlation of model 2 is 0.998. The reason there is a slight dip in correlation for model 2 is because it is not completely based on just carat and price. But this does not lower the correlation enough to say the model is inaccurate.