END OF SPRING SEMESTER PROJECT.

Presented by

Stevye Tinchie F.

This project aims at building a predictive model based on some given data. The data considered in this project is the "Diamond" data which consist of data on the price of diamond and other attributes like color, cut, carat weight, depth, table, crown weight etc. However, we would try to build our "best" possible predictive model of the price of diamond by eliminating the categorical variable and considering only the quantitative variable.

We proceed is three stages. That exploratory stage, the model building proper and an exemple.

Exploratory Stage:

Here we try to understand or see how the variables interact between each other. We also provide some graphs of each predictor considered against the response variable to see how they each behave with the response variable.

The idea of doing so is to be able to have some sought of idea on which model would best describe the data.

setwd("C:/Users/steve/Desktop/work")

df = read.csv("diamond.csv")

head(df)

```
> head(df)
  carat depth table price crown_depth pav_depth girdle
1  0.23  61.5    55   326        3.95      3.98   2.43
2  0.21  59.8    61   326        3.89      3.84   2.31
3  0.23  56.9    65   327        4.05      4.07   2.31
4  0.29  62.4    58   334        4.20      4.23   2.63
5  0.31  63.3    58   335        4.34      4.35   2.75
6  0.24  62.8    57   336        3.94      3.96   2.48
```

summary(df)

```
> summary(df)
     carat             depth            table            price          crown_dept
h       pav_depth           girdle
```

```
 Min.   :0.2000   Min.   :43.00   Min.   :43.00   Min.   :  326   Min.   : 0.
000   Min.   : 0.000   Min.   : 0.000  
 1st Qu.:0.4000   1st Qu.:61.00   1st Qu.:56.00   1st Qu.:  950   1st Qu.: 4.
710   1st Qu.: 4.720   1st Qu.: 2.910  
 Median :0.7000   Median :61.80   Median :57.00   Median : 2401   Median : 5.
700   Median : 5.710   Median : 3.530  
 Mean   :0.7979   Mean   :61.75   Mean   :57.46   Mean   : 3933   Mean   : 5.
731   Mean   : 5.735   Mean   : 3.539  
 3rd Qu.:1.0400   3rd Qu.:62.50   3rd Qu.:59.00   3rd Qu.: 5324   3rd Qu.: 6.
540   3rd Qu.: 6.540   3rd Qu.: 4.040  
 Max.   :5.0100   Max.   :79.00   Max.   :95.00   Max.   :18823   Max.   :10.
740   Max.   :58.900   Max.   :31.800  
```

observations = nrow(df)

variables = ncol(df)

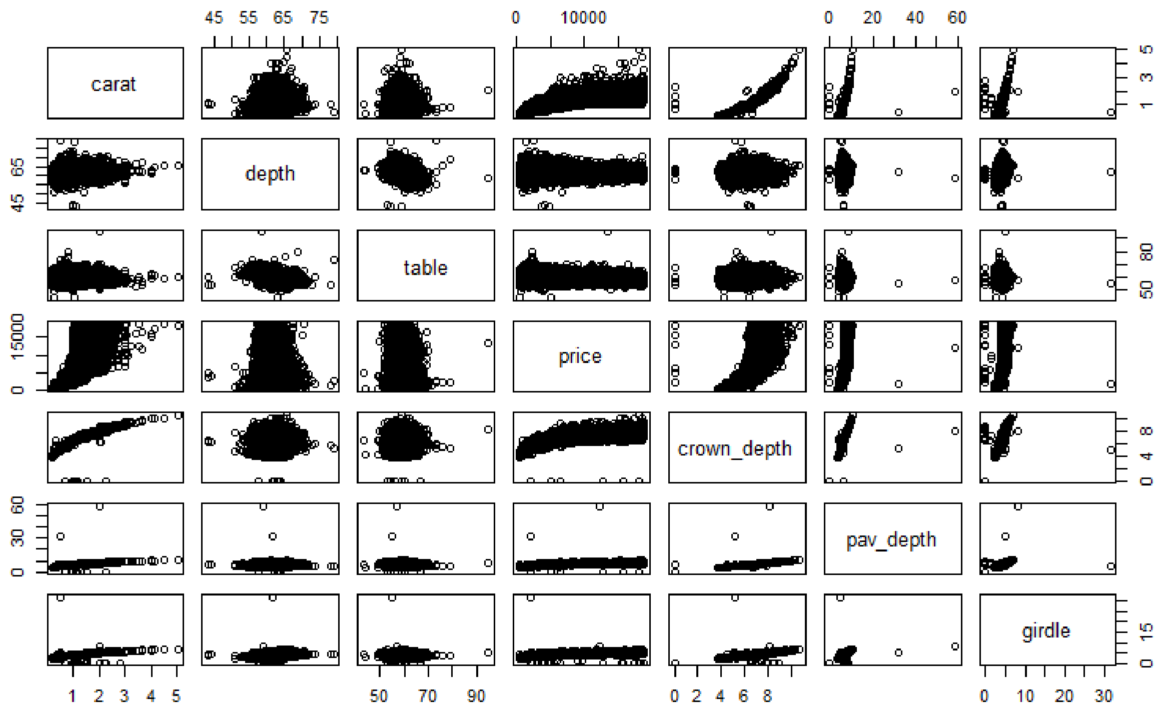
sprintf("observations: %s and variables: %s", observations, variables)

```
[1] "observations: 53940 and variables: 7"
```

cor(df)

```
> cor(df)
                 carat       depth      table      price crown_depth   pav_de
pth     girdle
carat       1.00000000  0.02822431  0.1816175  0.9215913  0.97509423  0.95172
220 0.95338738
depth       0.02822431  1.00000000 -0.2957785 -0.0106474 -0.02528925 -0.02934
067 0.09492388
table       0.18161755 -0.29577852  1.0000000  0.1271339  0.19534428  0.18376
015 0.15092869
price       0.92159130 -0.01064740  0.1271339  1.0000000  0.88443516  0.86542
090 0.86124944
crown_depth 0.97509423 -0.02528925  0.1953443  0.8844352  1.00000000  0.97470
148 0.97077180
pav_depth   0.95172220 -0.02934067  0.1837601  0.8654209  0.97470148  1.00000
000 0.95200572
girdle      0.95338738  0.09492388  0.1509287  0.8612494  0.97077180  0.95200
572 1.00000000
```

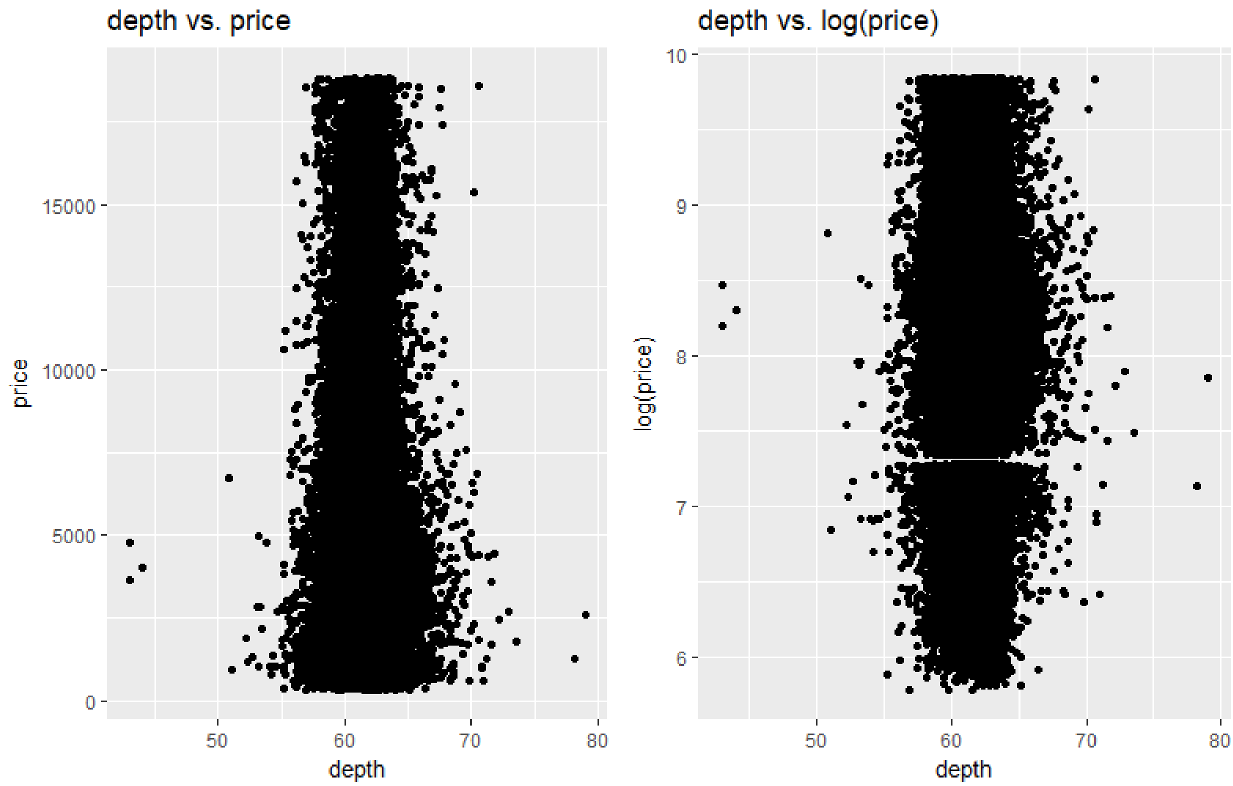
pairs(df)



p3 <- ggplot(df, aes(x=depth, y=price))+geom_point()+ ggtitle('depth vs. price')

p4 <- ggplot(df, aes(x=depth, y=log(price)))+geom_point() + ggtitle('depth vs. log(price)')

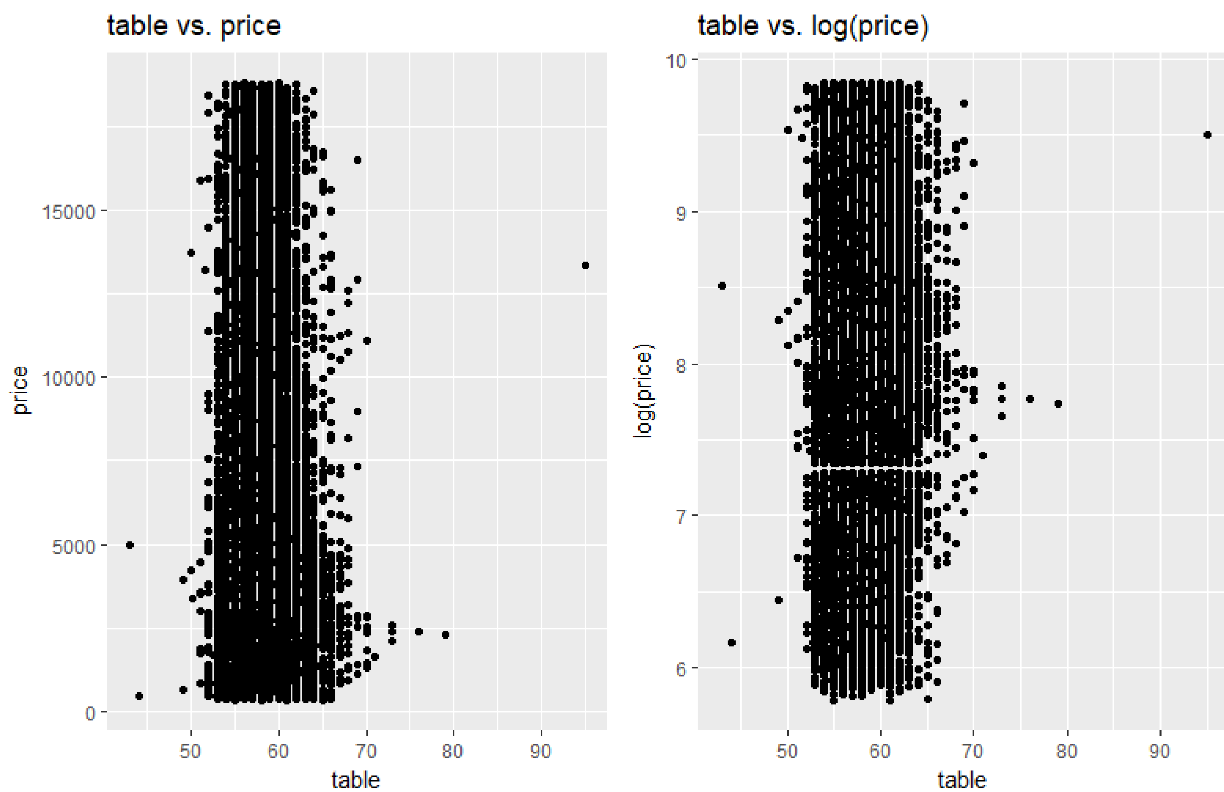
grid.arrange(p3, p4, ncol=2)



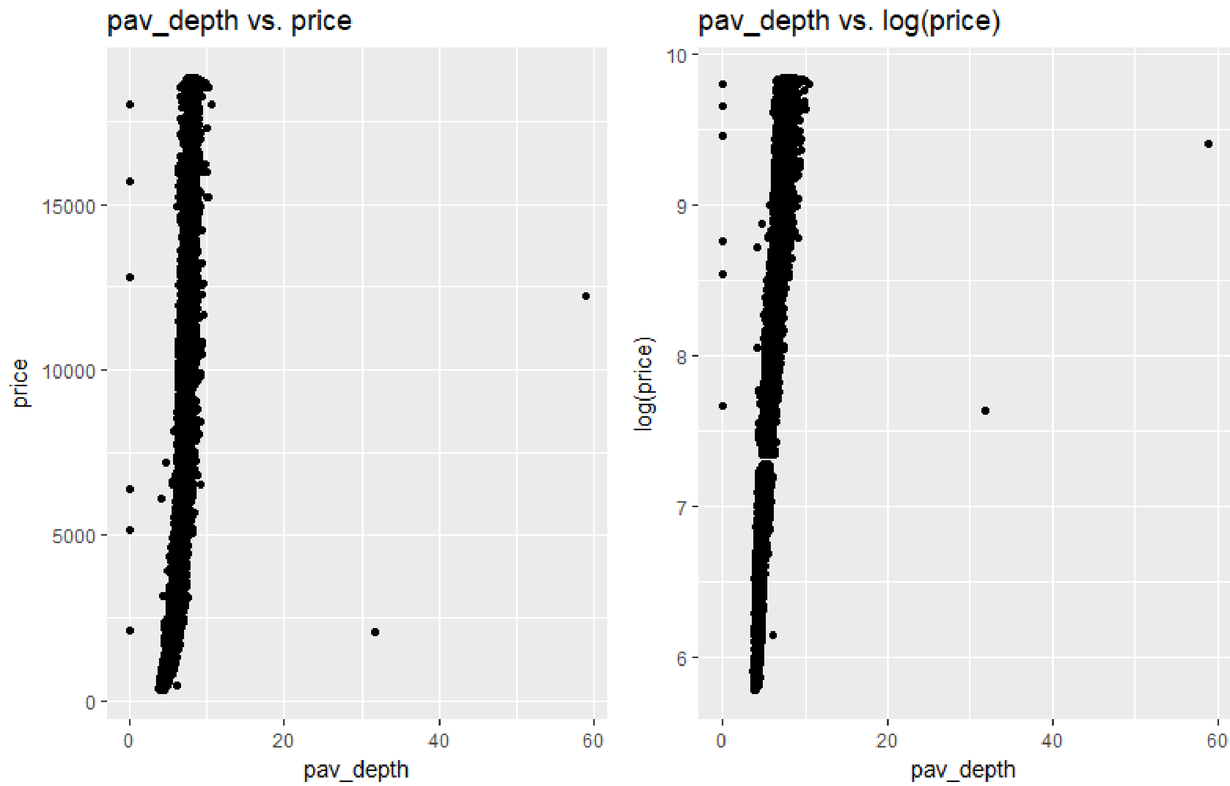
p5 <- ggplot(df, aes(x=table, y=price))+geom_point()+ ggtitle('table vs. price')

p6 <- ggplot(df, aes(x=table, y=log(price)))+geom_point() + ggtitle('table vs. log(price)')

grid.arrange(p5, p6, ncol=2)



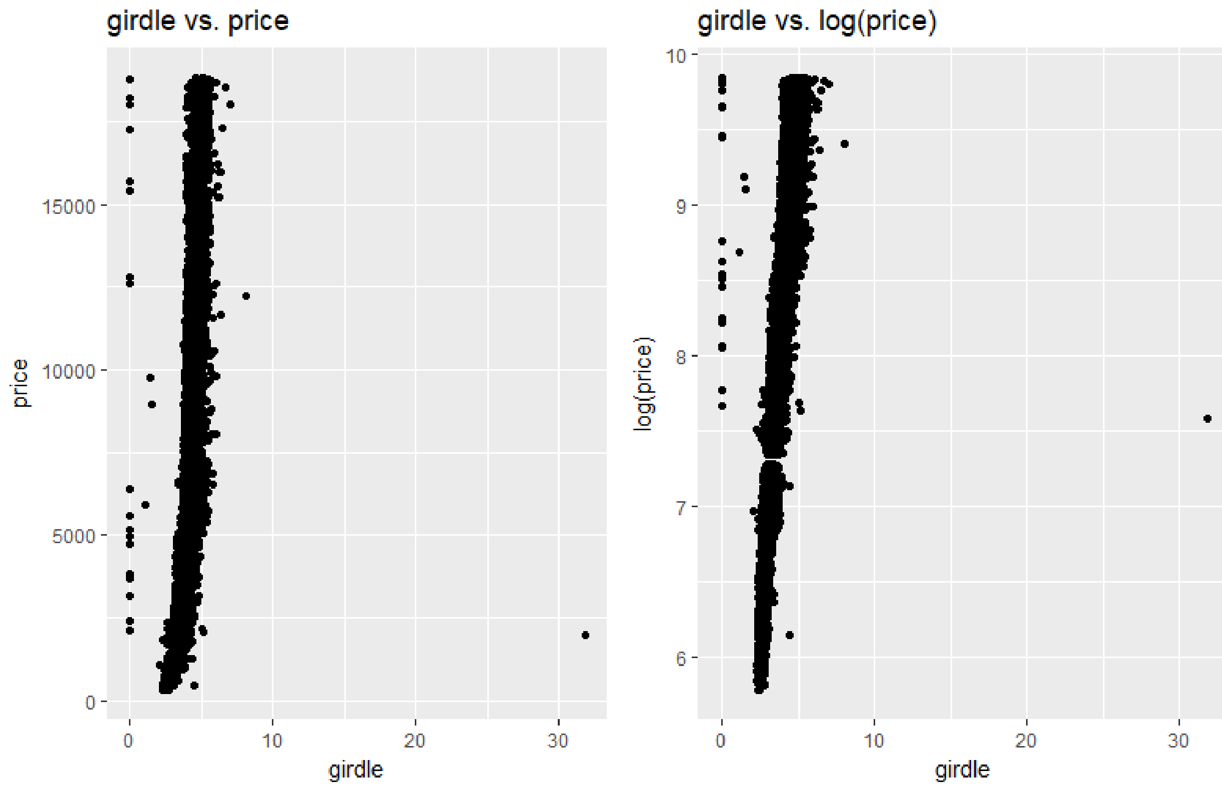

```r
p9 <- ggplot(df, aes(x=pav_depth, y=price))+geom_point()+ ggtitle('pav_depth vs. price')

p10 <- ggplot(df, aes(x=pav_depth, y=log(price)))+geom_point() + ggtitle('pav_depth vs. log(price)')

grid.arrange(p9, p10, ncol=2)
```



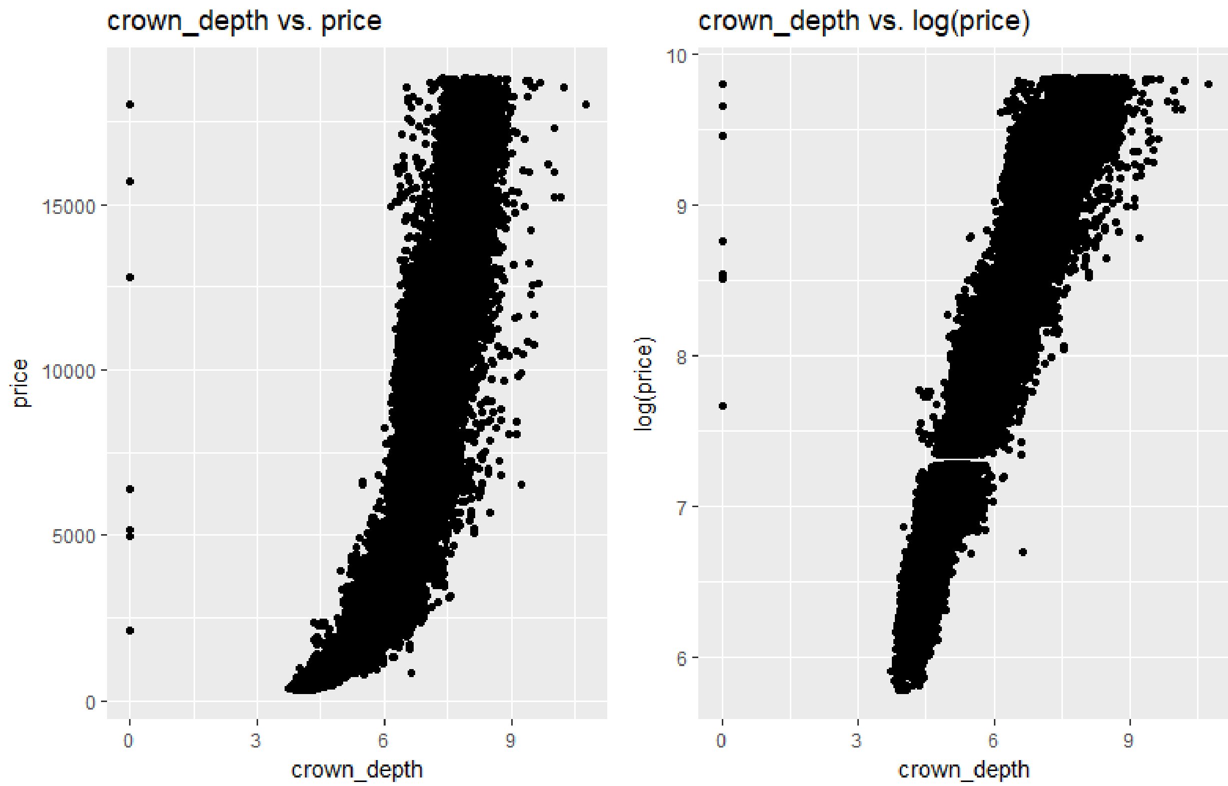
p11 <- ggplot(df, aes(x=girdle, y=price))+geom_point()+ ggtitle('girdle vs. price')

p12 <- ggplot(df, aes(x=girdle, y=log(price)))+geom_point() + ggtitle('girdle vs. log(price)')

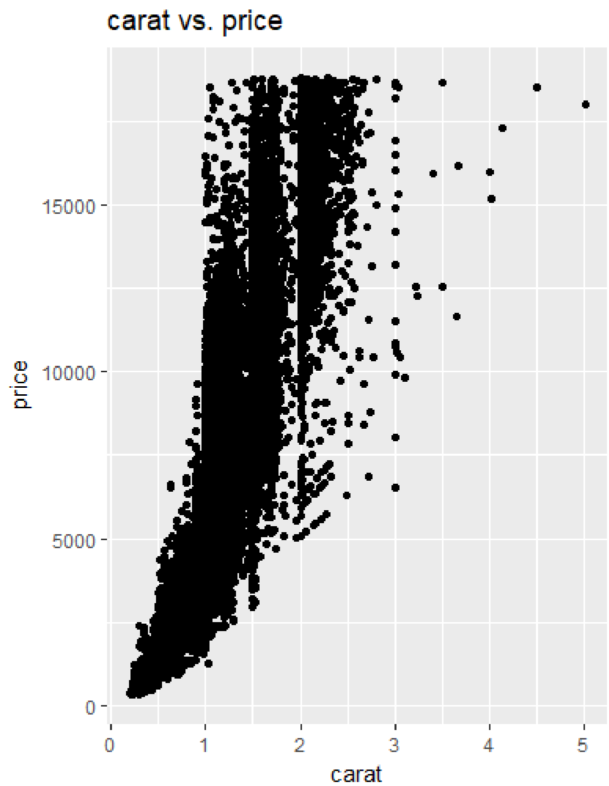
grid.arrange(p9, p10, ncol=2)

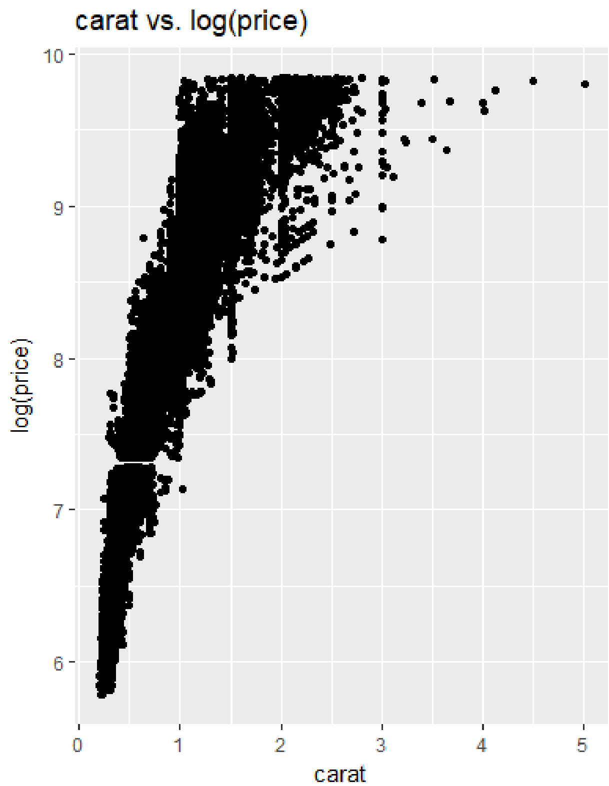



```
p7 <- ggplot(df, aes(x=crown_depth, y=price))+geom_point()+
ggtitle('crown_depth vs. price')

p8 <- ggplot(df, aes(x=crown_depth, y=log(price)))+geom_point() +
ggtitle('crown_depth vs. log(price)')

grid.arrange(p7, p8, ncol=2)
```




```
p1 <- ggplot(df, aes(x=carat, y=price))+geom_point()+ ggtitle('carat vs. price')

p2 <- ggplot(df, aes(x=carat, y=log(price)))+geom_point() + ggtitle('carat vs. log(price)')

grid.arrange(p1, p2, ncol=2)
```






```
library(leaps)                                    #method of best subset

best.subset <- regsubsets(price~., df, nvmax=6)

best.subset.summary <- summary(best.subset)

best.subset.summary$outmat
```

```
         carat depth table crown_depth pav_depth girdle
1  ( 1 ) "*"   " "   " "   " "         " "       " "
2  ( 1 ) "*"   " "   " "   "*"         " "       " "
3  ( 1 ) "*"   "*"   " "   "*"         " "       " "
4  ( 1 ) "*"   "*"   "*"   "*"         " "       " "
5  ( 1 ) "*"   "*"   "*"   "*"         "*"       " "
6  ( 1 ) "*"   "*"   "*"   "*"         "*"       "*"
```

```
best.subset.by.adjr2 <- which.max(best.subset.summary$adjr2)
best.subset.by.adjr2
```

[1] 5

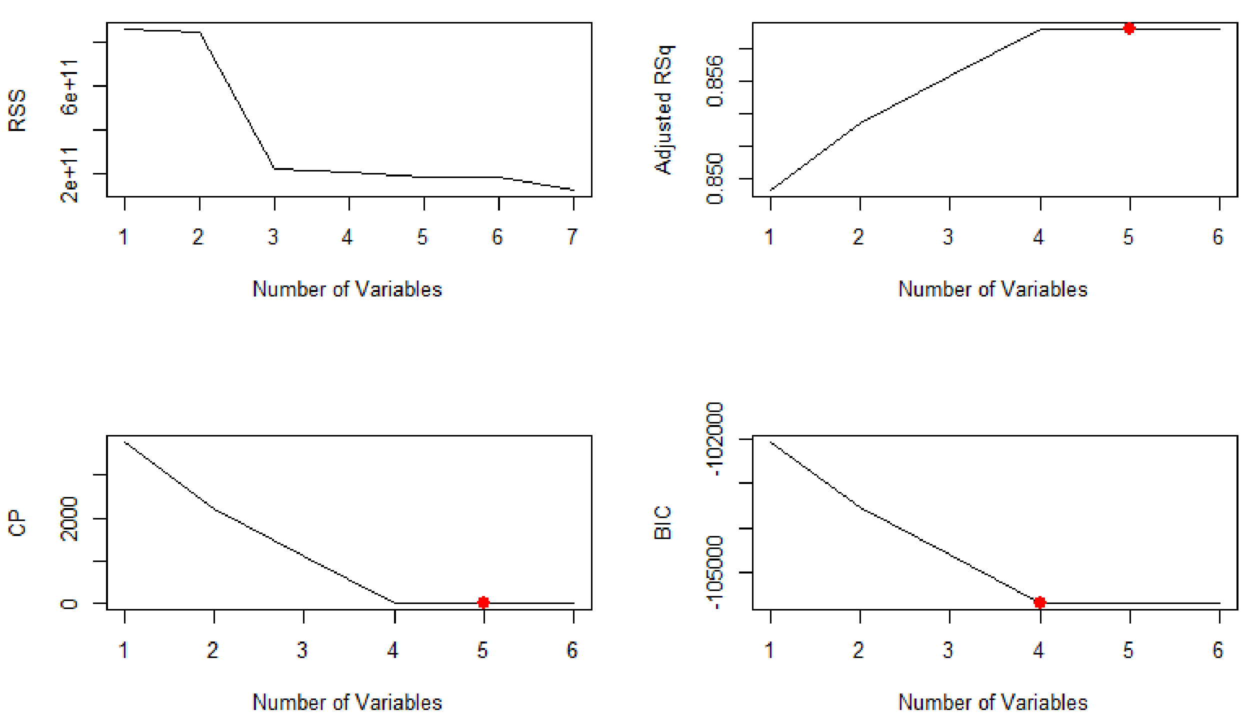
```
best.subset.by.cp <- which.min(best.subset.summary$cp)
best.subset.by.cp
```

```
[1] 5
```

```
best.subset.by.bic <- which.min(best.subset.summary$bic)
best.subset.by.bic
```

```
[1] 4
```

```
par(mfrow=c(2,2))
plot(best.subset$rss, xlab="Number of Variables", ylab="RSS", type="l")
plot(best.subset.summary$adjr2, xlab="Number of Variables", ylab="Adjusted RSq", type="l")
points(best.subset.by.adjr2, best.subset.summary$adjr2[best.subset.by.adjr2], col="red", cex =2, pch =20)
plot(best.subset.summary$cp, xlab="Number of Variables", ylab="CP", type="l")
points(best.subset.by.cp, best.subset.summary$cp[best.subset.by.cp], col="red", cex =2, pch =20)
plot(best.subset.summary$bic, xlab="Number of Variables", ylab="BIC", type="l")
points(best.subset.by.bic, best.subset.summary$bic[best.subset.by.bic], col="red", cex =2, pch =20)
```




```
coef(best.subset,4)
```

```
(Intercept)       carat       depth       table crown_depth
 20765.5208  10692.5100   -201.2311   -102.8239  -1226.7732
```

Model Building:

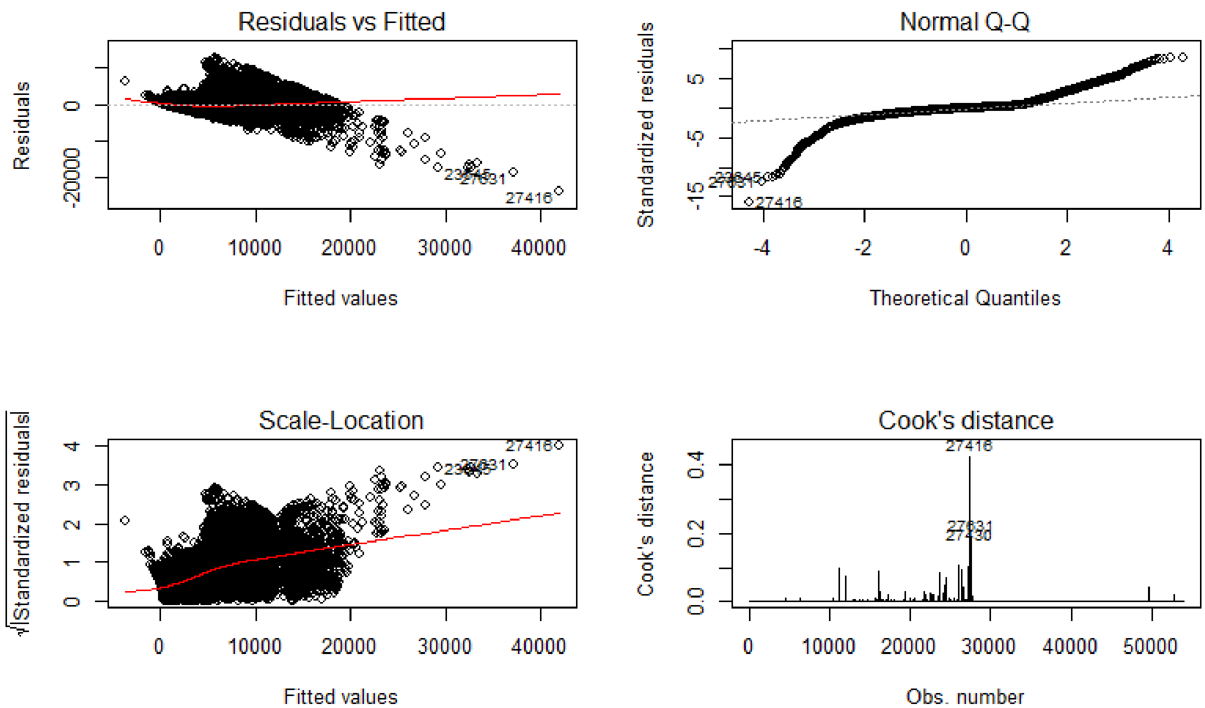
After decided on the number of predictors we want to include into our model, and after observing that the graphs of the response(price) against the most two correlated predictors seem some how like a curve, then we would proceed by trying to build a polynomial model.

We try try a couple of models before eventually arriving at what we believe could be a good model for the given data while making sure that the assumptions (LINE) for linear regression are met.

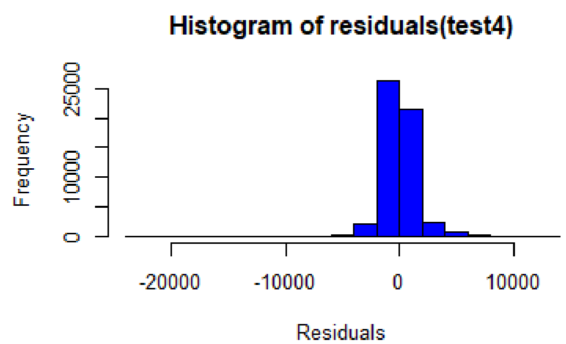
Model 1:

```
# Simple linear regression model.

test4 = lm(price~carat+depth+table+crown_depth, data=df)

test4

summary(test4)
```

```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) 20765.521    418.984   49.56   <2e-16 ***
carat       10692.510     63.168  169.27   <2e-16 ***
depth        -201.231      4.852  -41.48   <2e-16 ***
table        -102.824      3.082  -33.37   <2e-16 ***
crown_depth -1226.773     26.678  -45.98   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1497 on 53935 degrees of freedom
Multiple R-squared:  0.8592,	Adjusted R-squared:  0.8592 
F-statistic: 8.228e+04 on 4 and 53935 DF,  p-value: < 2.2e-16
```



Residuals vs Fitted
Residuals
Fitted values
23845
27631
27416
Normal Q-Q
Standardized residuals
Theoretical Quantiles
Scale-Location
√|Standardized residuals|
Fitted values
Cook's distance
Cook's distance
Obs. number
27416
27631
27430



Histogram of residuals(test4)
Frequency
Residuals

Model 2:

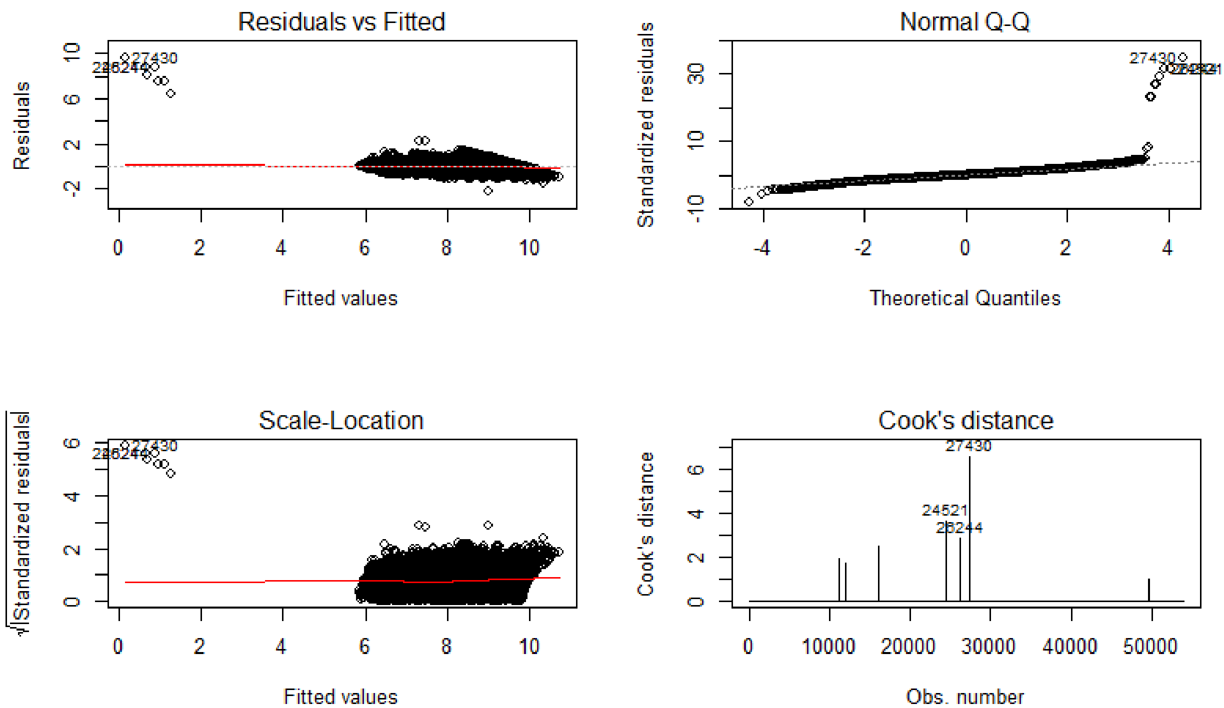
 # Simple linear regression model.

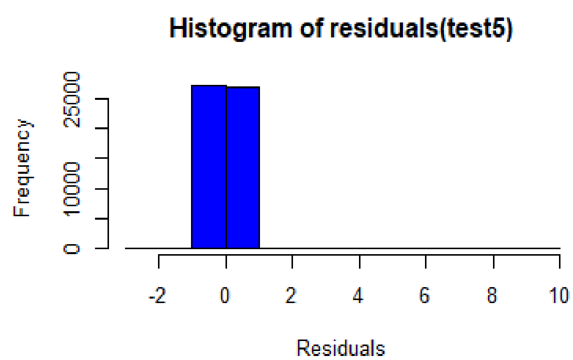
test5 = lm(log(price)~carat+depth+table+crown_depth, data=df)

test5

summary(test5)

```
> summary(test5)

Call:
lm(formula = log(price) ~ carat + depth + table + crown_depth,
    data = df)

Residuals:
    Min      1Q  Median      3Q     Max
-2.3103 -0.1708 -0.0022  0.1645  9.6570

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.7128179  0.0783546   9.097   <2e-16 ***
carat       -0.6979023  0.0118131 -59.079   <2e-16 ***
depth        0.0254160  0.0009073  28.013   <2e-16 ***
table       -0.0100964  0.0005763 -17.519   <2e-16 ***
crown_depth  1.1588446  0.0049891 232.276   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.28 on 53935 degrees of freedom
Multiple R-squared:  0.9239,	Adjusted R-squared:  0.9239
F-statistic: 1.636e+05 on 4 and 53935 DF,  p-value: < 2.2e-16
```


Residuals vs Fitted
Residuals
Fitted values
Normal Q-Q
Standardized residuals
Theoretical Quantiles
Scale-Location
√|Standardized residuals|
Fitted values
Cook's distance
27430
24521
26244
Cook's distance
Obs. number


Histogram of residuals(test5)
Frequency
Residuals

Model 3:

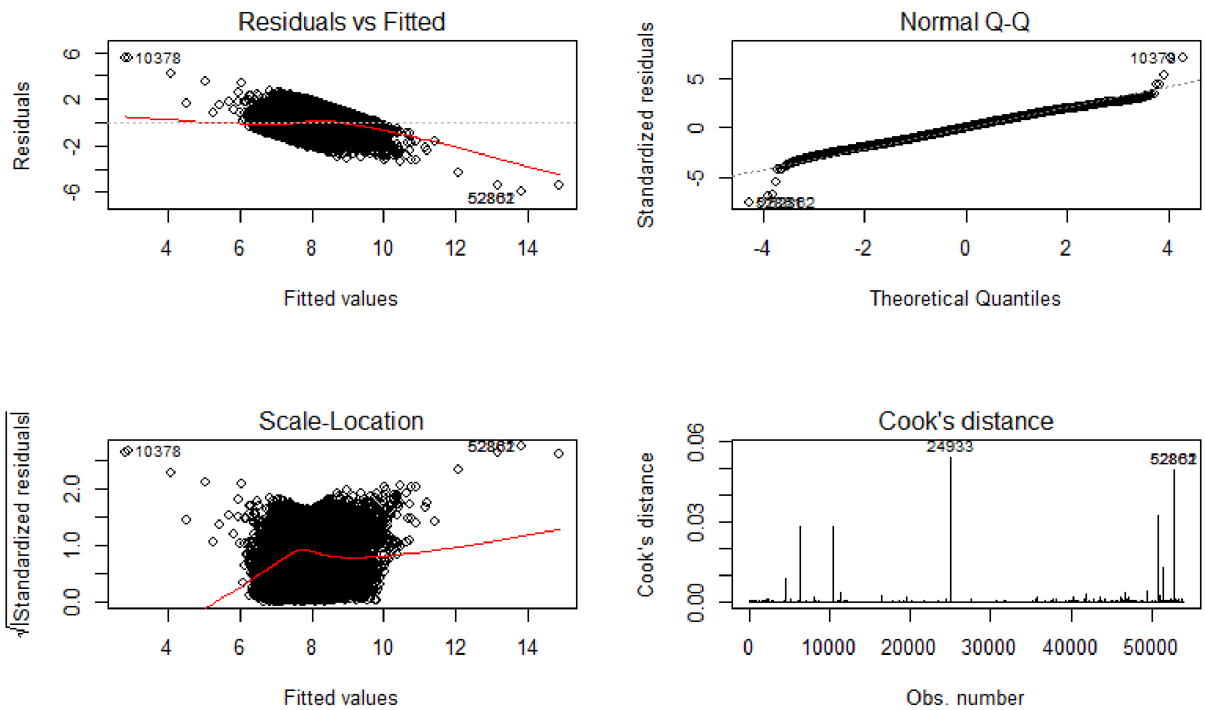
Simple linear regression model.

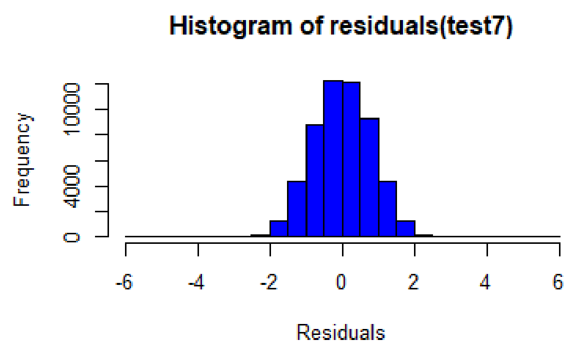
test7 = lm(log(price)~log(carat+depth+table+crown_depth), data=df)

test7

summary(test7)

```
> summary(test7)

Call:
lm(formula = log(price) ~ log(carat + depth + table + crown_depth), 
    data = df)

Residuals:
    Min      1Q  Median      3Q     Max 
-5.9939 -0.5529  0.0034  0.5654  5.5904 

Coefficients:
                                         Estimate Std. Error t value Pr(>|t|)    
(Intercept)                              -121.389      0.696  -174.4   <2e-16 
***
log(carat + depth + table + crown_depth)   26.723      0.144   185.6   <2e-16 
***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.7926 on 53938 degrees of freedom
Multiple R-squared:  0.3897,	Adjusted R-squared:  0.3897 
F-statistic: 3.445e+04 on 1 and 53938 DF,  p-value: < 2.2e-16
```


Residuals vs Fitted
Residuals
Fitted values
10378
52862
Normal Q-Q
Standardized residuals
Theoretical Quantiles
Scale-Location
√|Standardized residuals|
Fitted values
Cook's distance
Cook's distance
Obs. number
24933
52862


Histogram of residuals(test7)
Frequency
Residuals

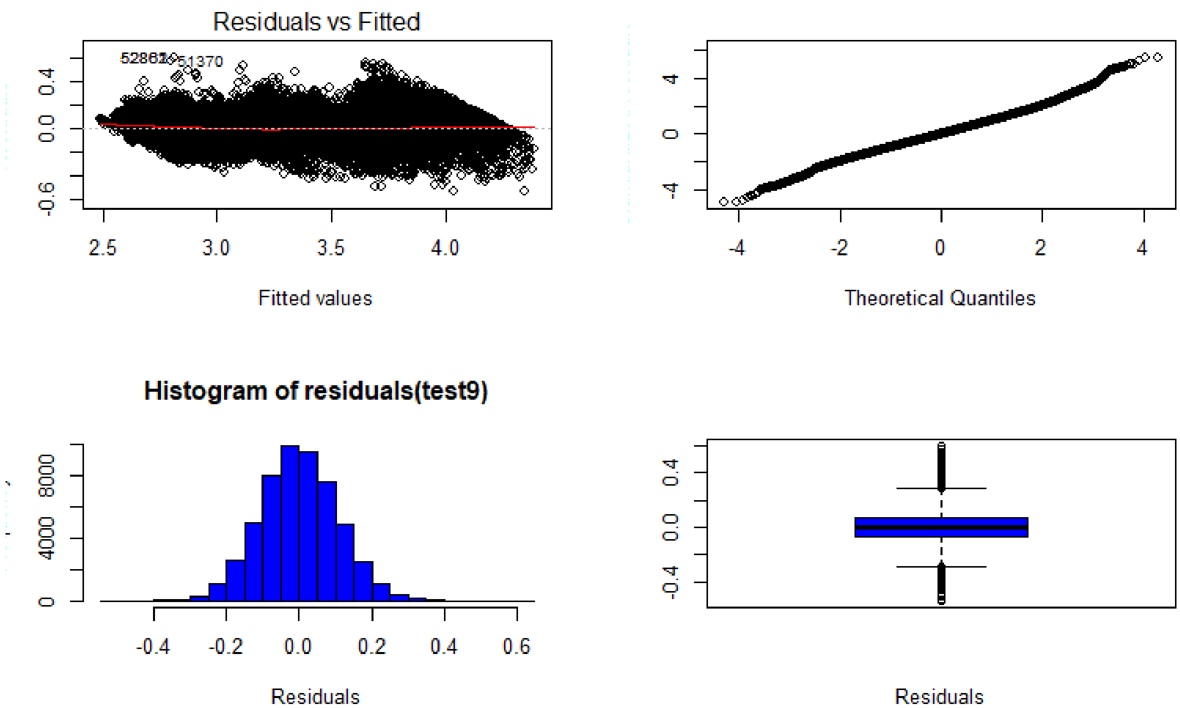
Model 4:

test9 = lm(formula = I(log10(price)) ~ I(carat^(1/3)) + carat + depth + table + crown_depth, data = df)

test9

summary(test9)

```
> summary(test9)

Call:
lm(formula = I(log10(price)) ~ I(carat^(1/3)) + carat + depth + 
    table + crown_depth, data = df)

Residuals:
     Min       1Q   Median       3Q      Max 
-0.54418 -0.07204 -0.00191  0.07131  0.60265 

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)    
(Intercept)     1.6766541  0.0334116  50.182   <2e-16 ***
I(carat^(1/3))  3.7585243  0.0350138 107.344   <2e-16 ***
carat          -0.5000636  0.0050055 -99.904   <2e-16 ***
depth          -0.0126550  0.0004203 -30.110   <2e-16 ***
table          -0.0082574  0.0002300 -35.896   <2e-16 ***
crown_depth     0.0003022  0.0050817   0.059    0.953    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.1104 on 53934 degrees of freedom
Multiple R-squared:  0.9373,	Adjusted R-squared:  0.9373 
F-statistic: 1.612e+05 on 5 and 53934 DF,  p-value: < 2.2e-16
```



library(nortest)

ad.test(residuals(test9))

```
ad.test(residuals(test9))

        Anderson-Darling normality test

data:  residuals(test9)
A = 18.475, p-value < 2.2e-16
```

Rainbow test for linearity.

raintest(test9)

```
> raintest(test9)

        Rainbow test

data:  test9
Rain = 0.88348, df1 = 26970, df2 = 26964, p-value = 1
```

Independence of residuals.

dwtest(test9, alternative="two.sided")

```
dwtest(test9, alternative="two.sided")

	Durbin-Watson test

data:  test9
DW = 1.2194, p-value < 2.2e-16
alternative hypothesis: true autocorrelation is not 0
```

ncvTest(test9)   # Test of non constant variance Null hypothesis: the variance is constant Alternative hypothesis: the variance is not constant

```
> ncvTest(test9)   # Test of non constant variance Null hypothesis: the variance is const
not constant
Non-constant Variance Score Test
Variance formula: ~ fitted.values
Chisquare = 291.1419    Df = 1     p = 2.80375e-65
```

x <- model.matrix(test9)[,-1]
e <- eigen(t(x) %*% x)   #EIGENVALUES
e$val

```
[1] 3.857309e+08 2.483299e+05 7.721494e+04 5.812422e+02 1.138366e+01
```

sqrt(e$val[1]/e$val)   #CONDITION NUMBER

```
[1]    1.00000   39.41193   70.67919  814.63610 5821.05039
```

require(faraway)

vif(x)

```
I(carat^(1/3))          carat          depth          table    crown_depth
    167.112092      24.927547       1.605414       1.169962     143.890662
```

confint(test9, level = 0.95)

```
> confint(test9, level = 0.95)
                      2.5 %       97.5 %
(Intercept)     1.611167052  1.742141154
I(carat^(1/3))  3.689897034  3.827151518
carat          -0.509874322 -0.490252830
depth          -0.013478761 -0.011831194
table          -0.008708306 -0.007806551
```

```
crown_depth    -0.009658006  0.010262343
```

the linear model for the diamond price is:

Log10(price) = 1.6766541 + 3.7585243*(carat^1/3) -0.5000636*carat - 0.0126550*depth -0.0082574*table + 0.0003022*crown_depth

Pick a diamond and predict the price by using our new model.

thisDiamond <- data.frame(carat=0.23, depth=56.9,

table=65,crown_depth=4.05)

modelEstimate <- predict(test9,newdata = thisDiamond,

interval = "prediction",level = .95)

10^modelEstimate

```
thisDiamond <- data.frame(carat=0.23, depth=56.9,
+                             table=65,crown_depth=4.05)
> modelEstimate <- predict(test9,newdata = thisDiamond,
+                            interval = "prediction",level = .95)
> 10^modelEstimate
      fit      lwr      upr
1 406.333 246.8909 668.7428
```

The predicted price is $406.33 vs. actual price $327.

Which overfits the actual price with a difference of $79.33

CONCLUSION:

Based on the results obtained from this model that we built (especially from the above example), we see that the this model over estimates the actual price of diamond instead of predicting(approximating) it. We believe that this behavior might be due to the existence of the many outliers and leverage points present in the dataset. Moreover, this might also be due to the categorical predictors that we eliminated at the very start of the work. Perhaps these variables actually have a very high impact on the price of diamond.

Hence, we conclude that the above model is a pretty good model but not the best because of the above reasons.

Therefore, as a future perspective of this project, we would want to consider all of the predictors (both categorical and quantitative) and maybe employ a categorical regression or logistic regression to get the best possible model.