Regression Analysis of 2022 Diamond Prices

Xiaofeng Cai Zitong Xi Bowen Yin

2024-06-14

Contents

1 PART 1: Data Description and Descriptive Statistics

1.1 1 Select a Random Sample

```
# Load the diamonds dataset
diamonds <- read.csv("Diamonds Prices2022.csv", header = T)

diamonds$cut <- as.factor(diamonds$cut)
diamonds$color <- as.factor(diamonds$color)
diamonds$clarity <- as.factor(diamonds$clarity)
```

1.2 2 Describe all the Variables

```
head(diamonds)
```

```
##   X carat       cut color clarity depth table price    x    y    z
## 1 1  0.23     Ideal     E     SI2  61.5    55   326 3.95 3.98 2.43
## 2 2  0.21   Premium     E     SI1  59.8    61   326 3.89 3.84 2.31
## 3 3  0.23      Good     E     VS1  56.9    65   327 4.05 4.07 2.31
## 4 4  0.29   Premium     I     VS2  62.4    58   334 4.20 4.23 2.63
## 5 5  0.31      Good     J     SI2  63.3    58   335 4.34 4.35 2.75
## 6 6  0.24 Very Good     J    VVS2  62.8    57   336 3.94 3.96 2.48
```

Observational Unit: Each row in the dataset represents a single diamond.

The Diamonds dataset contains information about diamond prices and various attributes, which are the following columns:

- `X`: The serial number of each diamond.
- `carat`: The weight of the diamond (independent quantities).
- `cut`: The quality of the cut including Fair, Good, Very Good, Premium, and Ideal (categorical variables).
- `color`: The diamond color including D, E, F, G, H, I, J (categorical variables).
- `clarity`: A measurement of how clear the diamond is, including I1, IF, SI1, SI2, VS1, VS2, VVS1, VVS2 (categorical variables).
- `depth`: Depth percentage is the diamond's depth divided by the width of the diamond (independent quantities).
- `table`: The width of the top of the diamond relative to the widest point (independent quantities).
- `price`: The price of the diamond in dollar (independent quantities).
- `x`: Length in mm.
- `y`: Width in mm.
- `z`: Depth in mm.

```
skim(diamonds)
```

Table 1: Data summary

Name	diamonds

Number of rows	53943
Number of columns	11
Column type frequency:	
factor	3
numeric	8
Group variables	None

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
cut	0	1	FALSE	5	Ide: 21551, Pre: 13793, Ver: 12083, Goo: 4906
color	0	1	FALSE	7	G: 11292, E: 9799, F: 9543, H: 8304
clarity	0	1	FALSE	8	SI1: 13067, VS2: 12259, SI2: 9194, VS1: 8171

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
X	0	1	26972.00	15572.15	1.0	13486.50	26972.00	40457.50	53943.00	
carat	0	1	0.80	0.47	0.2	0.40	0.70	1.04	5.01	
depth	0	1	61.75	1.43	43.0	61.00	61.80	62.50	79.00	
table	0	1	57.46	2.23	43.0	56.00	57.00	59.00	95.00	
price	0	1	3932.73	3989.34	326.0	950.00	2401.00	5324.00	18823.00	
x	0	1	5.73	1.12	0.0	4.71	5.70	6.54	10.74	
y	0	1	5.73	1.14	0.0	4.72	5.71	6.54	58.90	
z	0	1	3.54	0.71	0.0	2.91	3.53	4.04	31.80	

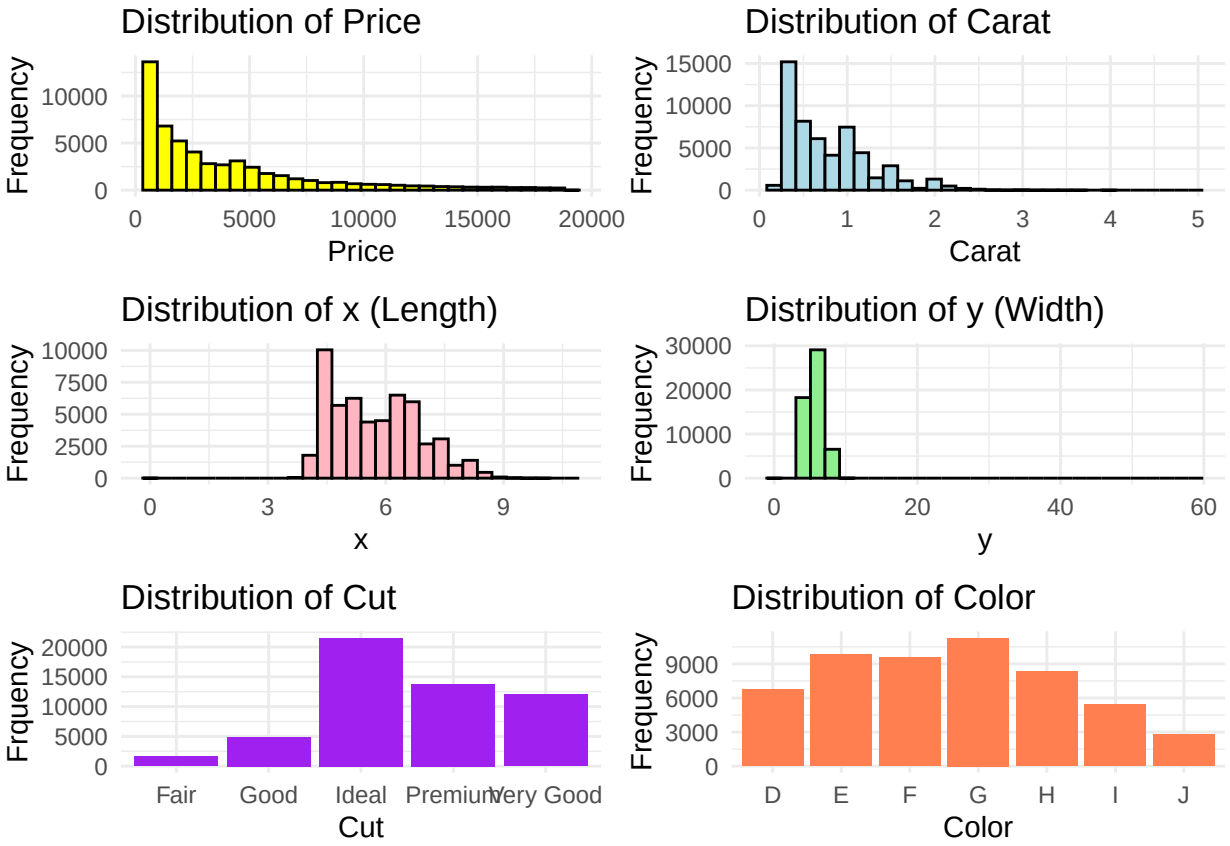
- Price: Mean is 3932.7997219, Standard Deviation is 3989.4397381, Minimum is 326, Maximum is 18823.00.
- Carat: Mean is 0.7979397, Standard Deviation is 0.4740112, Minimum 0.2, Maximum 5.01.
- Depth: Mean is 61.74932, Standard Deviation is 1.432626, Minimum 43.0, Maximum 79.00.
- Table: Mean is 57.45725, Standard Deviation is 2.234549, Minimum 43.0, Maximum 95.00.
- x (Length): Mean is 5.731158, Standard Deviation is 1.121730, Minimum is 0, Maximum is 10.74.
- y (Width): Mean is 5.734526, Standard Deviation is 1.142103, Minimum is 0, Maximum is 58.9.
- z (Depth): Mean is 3.538730, Standard Deviation is 0.7056795, Minimum is 0, Maximum is 31.80.

And to better see the distribution and frequency of some variables, we produced the following histograms and bar plots.

```
a <- ggplot(diamonds, aes(x = price)) +
  geom_histogram(fill = "yellow", color = "black") +
  ggtitle("Distribution of Price") +
  xlab("Price") +
  ylab("Frequency") +
  theme_minimal()
```

```
b <-ggplot(diamonds, aes(x = carat)) +
  geom_histogram(fill = "lightblue", color = "black") +
  ggtitle("Distribution of Carat") +
  xlab("Carat") +
  ylab("Frequency") +
  theme_minimal()

c <- ggplot(diamonds, aes(x = x)) +
  geom_histogram(fill = "lightpink", color = "black") +
  ggtitle("Distribution of x (Length)") +
  xlab("x") +
  ylab("Frequency") +
  theme_minimal()

d <- ggplot(diamonds, aes(x = y)) +
  geom_histogram(fill = "lightgreen", color = "black") +
  ggtitle("Distribution of y (Width)") +
  xlab("y") +
  ylab("Frequency") +
  theme_minimal()

e <- ggplot(diamonds, aes(x = cut)) +
  geom_bar(fill = 'purple') +
  ggtitle("Distribution of Cut") +
  xlab("Cut") +
  ylab("Frquency") +
  theme_minimal()

f <- ggplot(diamonds, aes(x = color)) +
  geom_bar(fill = 'coral') +
  ggtitle("Distribution of Color") +
  xlab("Color") +
  ylab("Frequency") +
  theme_minimal()

grid.arrange(a, b, c, d, e, f, ncol = 2)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



- Price Distribution: Right-skewed distribution. The histogram shows that most diamonds are priced below $5000, with the price distribution being right-skewed, meaning there are a few very high-priced diamonds.
- Carat Distribution: Right-skewed distribution. The histogram shows that most diamonds have a carat weight between 0.15 and 1.5.
- x Distribution: Bell-shaped distribution The histogram is concentrated, with most diamonds having a length between 4 and 7 mm. Since the distribution is bell-shaped, which indicate that most diamonds have a depth close to the median value, fitting a normal distribution.
- y Distribution: Bell-shaped distribution The histogram shows that most diamonds have a width between 4 and 9 mm, with a bell-shaped distribution, indicating that most diamonds have a width close to the median value, fitting a normal distribution.
- Cut Quality: The most common cut is the Ideal cut. The lest common cut is the Fair cut.
- Color: The most diamond colors is Color G. The least diamond color is Color J.

1.3 3 Determine Correlation

We chose three quantitative variables: carat, table, and depth. We also chose two categorical variables: cut and color. We want to see if there is any correlation between these variables and price, and among the variables themselves.

```
cor_matrix <- cor(diamonds[, c("price", "carat", "table", "depth")])
cor_matrix
```

```
##             price      carat      table       depth
## price  1.00000000 0.92159128  0.1271179 -0.01063007
## carat  0.92159128 1.00000000  0.1816017  0.02823442
## table  0.12711786 0.18160171  1.0000000 -0.29579799
## depth -0.01063007 0.02823442 -0.2957980  1.00000000
```

From the first row, we can see that carat, table, and depth all have a correlation with price. Carat has the strongest correlation with price, while depth has the weakest correlation with price. In addition, the correlations between carat and table, carat and depth, and table and depth are not very strong, particularly the correlation between price and depth.

1.4 4 Multiple Linear Regression Model

```
model_all <- lm(price ~ carat + table + depth + cut + color, diamonds)
summary(model_all)
```

```
##
## Call:
## lm(formula = price ~ carat + table + depth + cut + color, data = diamonds)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -17302.6   -751.0    -83.2    543.8  12163.4
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2228.841    456.687   4.880 1.06e-06 ***
## carat         8200.111     13.959 587.429  < 2e-16 ***
## table          -39.125      3.680 -10.631  < 2e-16 ***
## depth          -57.719      5.032 -11.471  < 2e-16 ***
## cutGood       1019.639     42.104  24.217  < 2e-16 ***
## cutIdeal      1558.095     41.775  37.297  < 2e-16 ***
## cutPremium    1272.963     40.353  31.546  < 2e-16 ***
## cutVery Good  1350.844     40.261  33.552  < 2e-16 ***
## colorE         -90.792     22.596  -4.018 5.88e-05 ***
## colorF         -74.861     22.744  -3.291 0.000997 ***
## colorG        -104.764     22.043  -4.753 2.01e-06 ***
## colorH        -728.628     23.678 -30.772  < 2e-16 ***
## colorI       -1070.294     26.549 -40.314  < 2e-16 ***
## colorJ       -1901.785     32.830 -57.928  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1430 on 53929 degrees of freedom
## Multiple R-squared:  0.8716, Adjusted R-squared:  0.8715
## F-statistic: 2.815e+04 on 13 and 53929 DF,  p-value: < 2.2e-16
```

1.5 5 Interesting Findings

- Carat being a major factor influencing diamond prices is an expected result because, in reality, diamonds that weigh more are more expensive. However, it is surprising that some diamonds weighing more than 5 carats still have a price below $20,000.

- It is surprising that diamonds depth percentage have negative correlation with price and table.

2 PART 2: Simple Linear Regression

2.1 1 & 2 Simple linear Regression Model

We are testing the following hypotheses:

- Null Hypothesis H_0: $\beta_1 = 0$ (The carat of a diamond does not affect its price)
- Alternative Hypothesis H_1: $\beta_1 > 0$ (The carat of a diamond positively affects its price)

The linear regression model is:

```
model <- lm(price ~ carat, diamonds)
summary(model)
```

```
##
## Call:
## lm(formula = price ~ carat, data = diamonds)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -18585.4   -804.7    -19.1    537.5  12731.7
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2256.40      13.05  -172.8   <2e-16 ***
## carat        7756.44      14.07   551.4   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1549 on 53941 degrees of freedom
## Multiple R-squared:  0.8493, Adjusted R-squared:  0.8493
## F-statistic: 3.041e+05 on 1 and 53941 DF,  p-value: < 2.2e-16
```

The coefficient for carat is 7756.44, which means that for every one unit increase in x (carat), the predicted value of Y (price) increases by 7756.44 units. The intercept coefficient of -2256.4 indicates that when x (carat) is zero, the predicted y (price) is -2256.4.

We obtained a p-value less than 0.05 for the coefficient of carat, indicating that carat has a statistically significant positive effect on the price of a diamond. And the p-value of F-test also support my interpretation that the model as a whole is statistically significant. And the Adjusted R-squared value of 0.8493 suggests that approximately 84.93% of the variation in diamond prices can be explained by carat and intercept term.

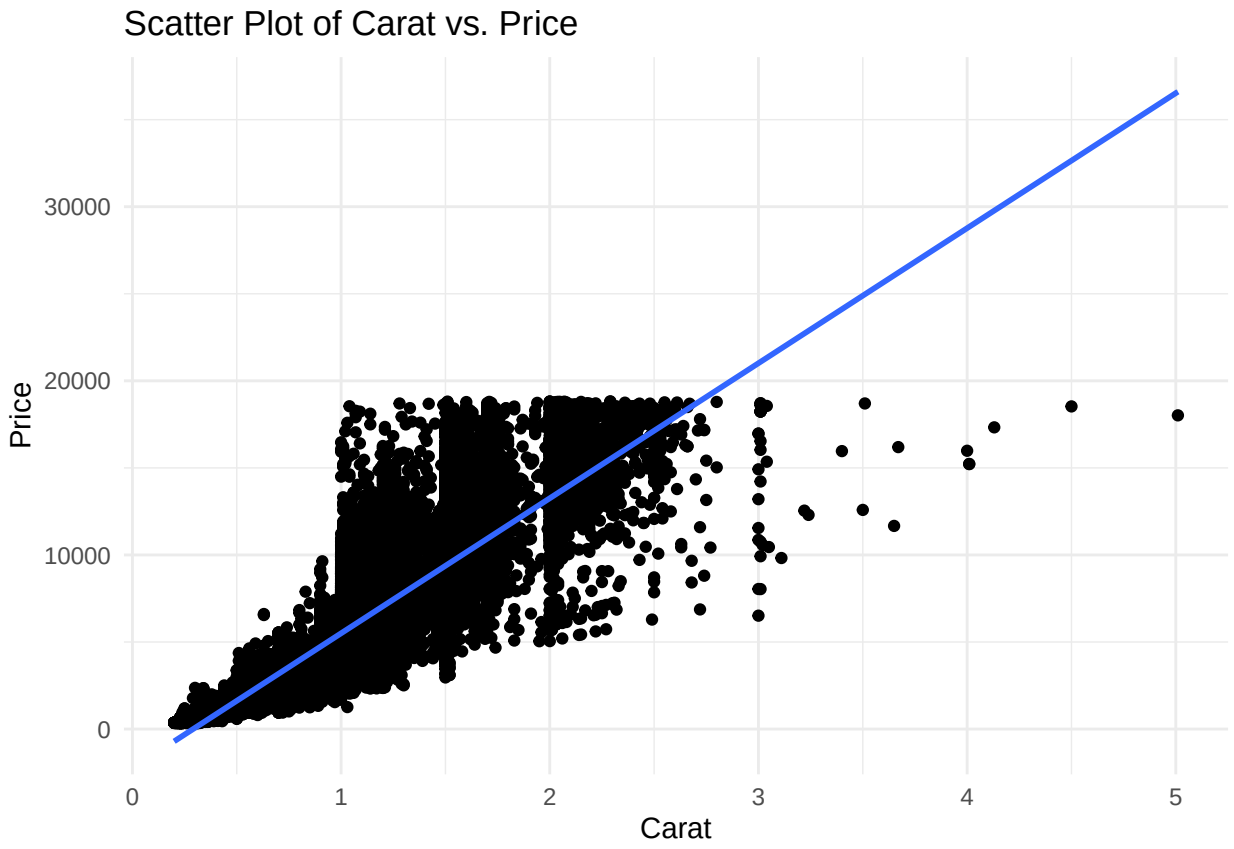
2.2 3 Test assumptions

2.2.1 1. Check scatter plot of Carat vs. Price to see if the relationship between them is linear

```
ggplot(diamonds, aes(x = carat, y = price)) +
  geom_point() +
  geom_smooth(method = "lm") +
  labs(title = "Scatter Plot of Carat vs. Price",
```

```
       x = "Carat",
       y = "Price") +
  theme_minimal()
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

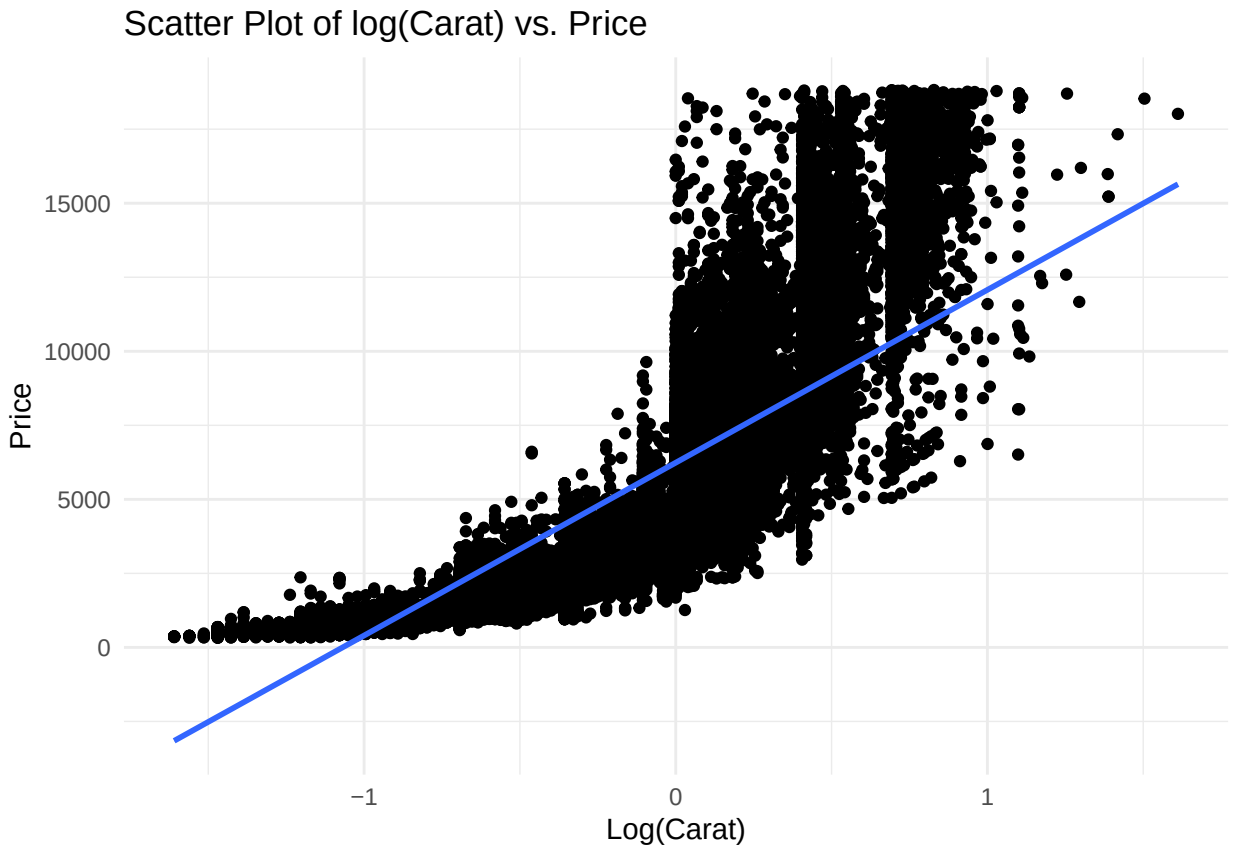


The relationship between price and carat is non-linear. Therefore, we will transform x first to linearize the relationship.

2.2.2 1.1 Log x to linearize the relationship

```
ggplot(diamonds, aes(x = log(carat), y = price)) +
  geom_point() +
  geom_smooth(method = "lm") +
  labs(title = "Scatter Plot of log(Carat) vs. Price",
       x = "Log(Carat)",
       y = "Price") +
  theme_minimal()
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



Now, the relationship between price and log(carat) is linear.

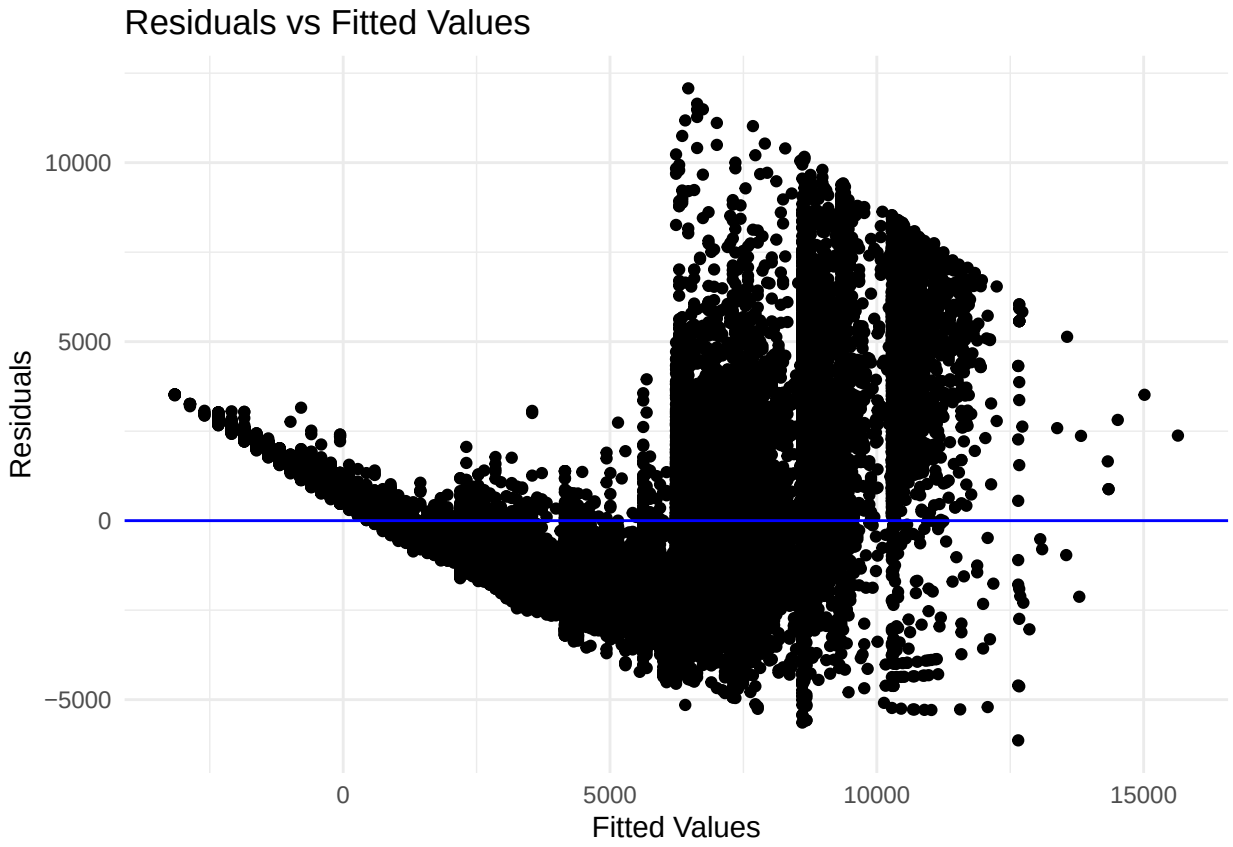
Our updated Log-Transformed Model:

```
log_model <- lm((price ~ log(carat)), data = diamonds)
summary(log_model)
```

```
##
## Call:
## lm(formula = (price ~ log(carat)), data = diamonds)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6137.3 -1494.9  -328.3  1142.0 12075.4
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  6237.75      10.73   581.2   <2e-16 ***
## log(carat)   5836.01      15.21   383.8   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2066 on 53941 degrees of freedom
## Multiple R-squared:  0.7319, Adjusted R-squared:  0.7319
## F-statistic: 1.473e+05 on 1 and 53941 DF,  p-value: < 2.2e-16
```

2.2.3 2. Check residuals vs. fitted values plot to test variance

```
ggplot(log_model, aes(.fitted, .resid)) +
  geom_point() +
  geom_hline(yintercept = 0, color = "blue") +
  labs(title = "Residuals vs Fitted Values",
       x = "Fitted Values",
       y = "Residuals") +
  theme_minimal()
```



The residuals vs. fitted values plot shows heteroscedasticity, we need to apply log transformation to price to make the spread of residuals more uniform.

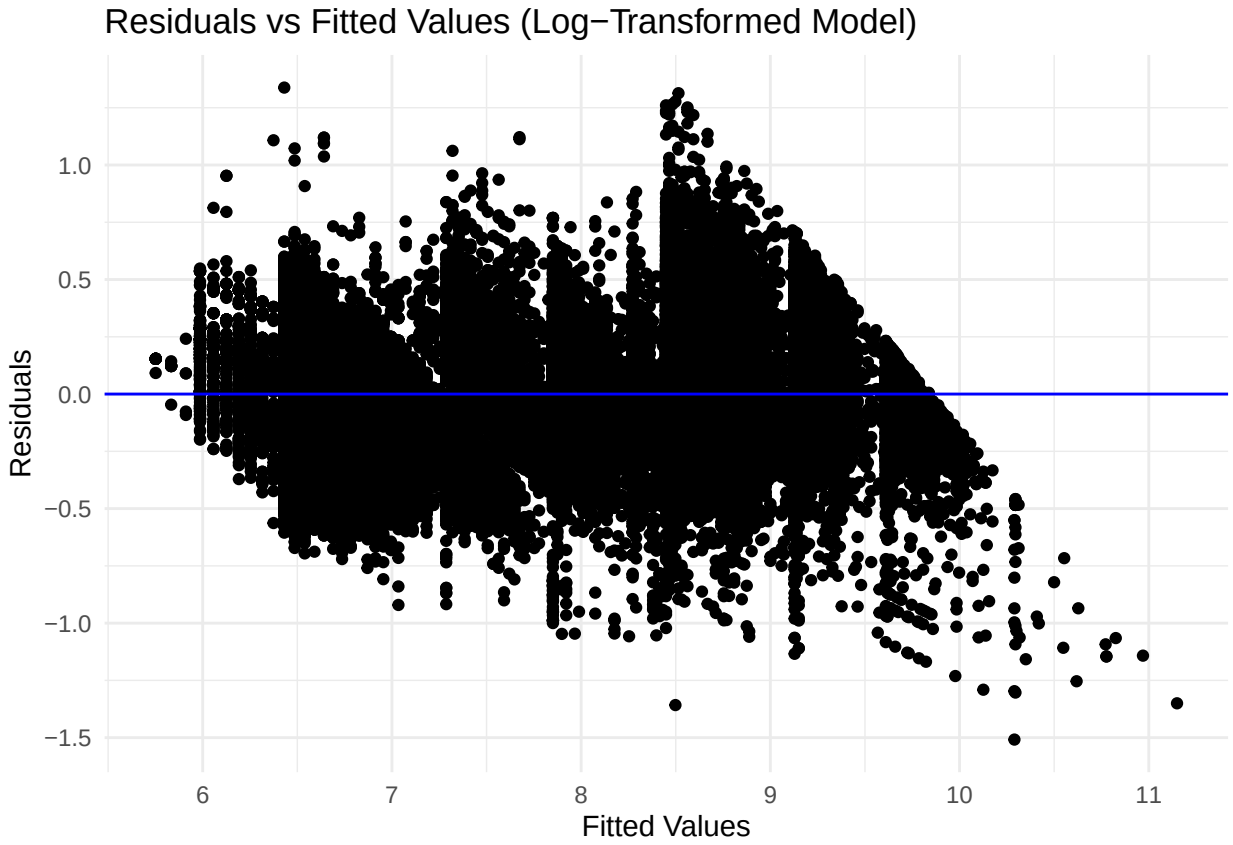
New model apply log-transformation to both price and carat:

```
new_log_model <- lm((log(price) ~ log(carat)), data = diamonds)
summary(new_log_model)
```

```
##
## Call:
## lm(formula = (log(price) ~ log(carat)), data = diamonds)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.5083 -0.1695 -0.0059  0.1664  1.3379
##
```

```
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 8.448664   0.001365  6191.2   <2e-16 ***
## log(carat)  1.675817   0.001934   866.6   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2627 on 53941 degrees of freedom
## Multiple R-squared:  0.933,  Adjusted R-squared:  0.933
## F-statistic: 7.51e+05 on 1 and 53941 DF,  p-value: < 2.2e-16
```

```
ggplot(new_log_model, aes(.fitted, .resid)) +
  geom_point() +
  geom_hline(yintercept = 0, color = "blue") +
  labs(title = "Residuals vs Fitted Values (Log-Transformed Model)",
       x = "Fitted Values",
       y = "Residuals") +
  theme_minimal()
```

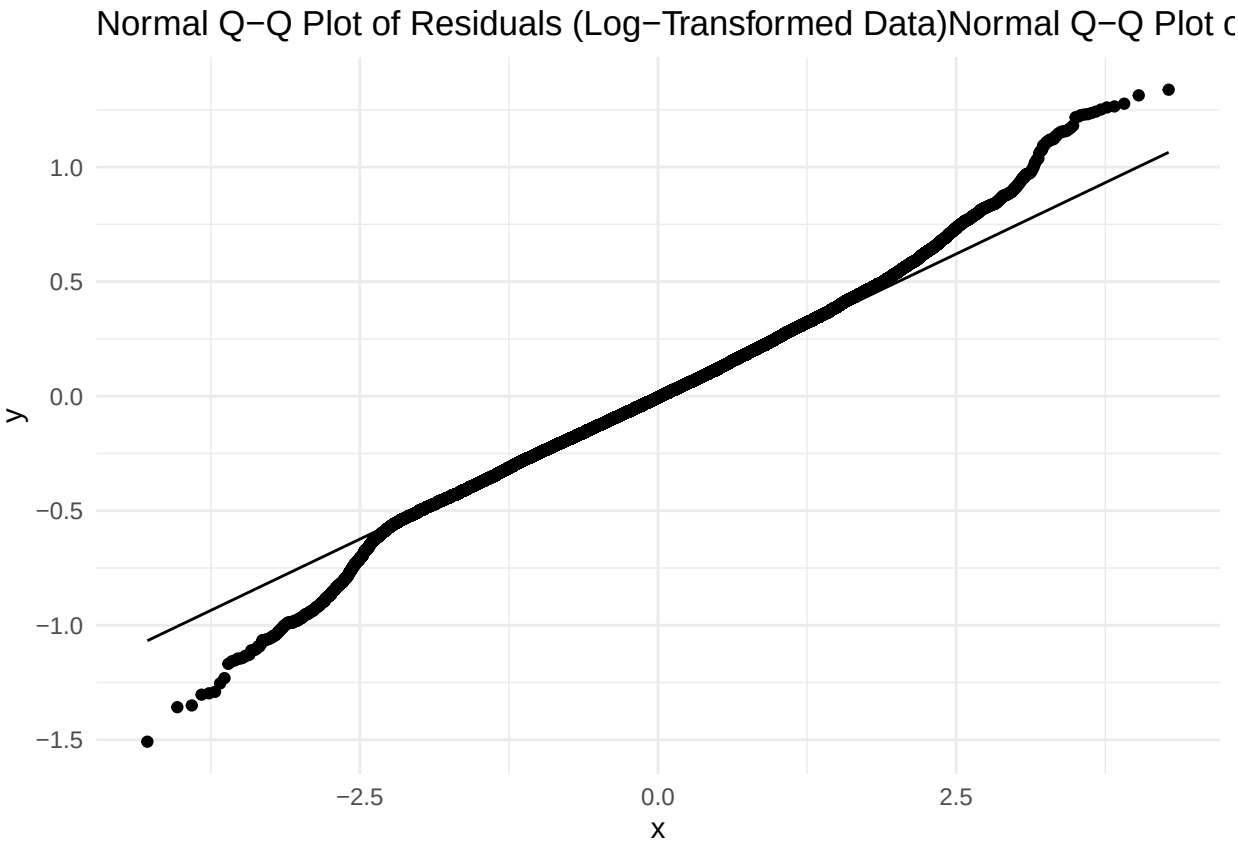


Now, we can say that the residuals have constant variance.

2.2.4 3. Check Q-Q plot to see if residuals are normally distributed for Log-transformed Model

```
ggplot(new_log_model, aes(sample = .resid)) +
  stat_qq() +
```

```
  stat_qq_line() +
  labs(title = "Normal Q-Q Plot of Residuals (Log-Transformed Data)Normal Q-Q Plot of Residuals (Log-Tra
  theme_minimal()
```



Most points closely aligning with the diagonal line on the Q-Q plot indicates that the residuals are normally distributed.

2.3 4 Run Model

```
summary(new_log_model)
```

```
##
## Call:
## lm(formula = (log(price) ~ log(carat)), data = diamonds)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.5083 -0.1695 -0.0059  0.1664  1.3379
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 8.448664   0.001365  6191.2   <2e-16 ***
## log(carat)  1.675817   0.001934   866.6   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 0.2627 on 53941 degrees of freedom
## Multiple R-squared:  0.933,  Adjusted R-squared:  0.933
## F-statistic: 7.51e+05 on 1 and 53941 DF,  p-value: < 2.2e-16
```

Now, the coefficient for log(carat) is 1.675817, which means that for every one unit increase in x (log(carat)), the predicted value of y (log(price)) increases by 1.675817 units. The intercept coefficient of 8.448664 indicates that when x (log(carat)) is zero, the predicted y (log(price)) is 8.448664.

And we have an Adjusted R-squared value of 0.933, it suggests that approximately 93.3% of the variation in log-transformed prices can be explained by the log-transformed carat variable and the intercept term.

2.4 5 Find Variables We Want

Adding cut in the simple linear regression model increased the adjusted R-squared from 0.933 to 0.9371, so we keep it. Adding color in the linear regression model increased the adjusted R-squared from 0.9371 to 0.9495, so we keep it. Adding clarity in the linear regression model increased the adjusted R-squared from 0.9495 to 0.9826, so we keep it. Adding depth did not change the adjusted R-squared value, so we do not need it. Adding table also did not change the adjusted R-squared value, so we do not need it. Adding x in the linear regression model increased the adjusted R-squared from 0.9826 to 0.9827, so we keep it. Adding y did not change the adjusted R-squared value, so we do not need it. Adding z did not change the adjusted R-squared value, so we do not need it.

2.5 6 Multicollinearity

```
model_5 <- lm(log(price) ~ log(carat) + cut + color + clarity + x, data = diamonds)
vif_values <- vif(model_5)
vif_values
```

```
##                 GVIF Df GVIF^(1/(2*Df))
## log(carat) 55.241887  1        7.432489
## cut         1.164613  4        1.019231
## color       1.154056  6        1.012012
## clarity     1.337232  7        1.020974
## x          55.016850  1        7.417334
```

Variables with high GVIF values greater than 5 indicate potential multicollinearity issues. We can see that there is multicollinearity between log(carat) and x. Therefore, we decided to exclude x in the model.

```
model_part2_final <- lm(log(price) ~ log(carat) + cut + color + clarity, data = diamonds)
vif_values <- vif(model_part2_final)
vif_values
```

```
##                GVIF Df GVIF^(1/(2*Df))
## log(carat) 1.313054  1        1.145885
## cut        1.105627  4        1.012631
## color      1.139971  6        1.010977
## clarity    1.323330  7        1.020212
```

Now, there is no such multicollinearity.

2.6 7 Interesting

The simple linear regression model with only the predictors 'carat' and 'price' achieved a good adjusted R-squared value of 0.933 after log-transformation. However, upon adding other variables, multicollinearity emerged between 'carat' and 'x'. Thus we need to exclude one of them from the model, which we decided to exclude 'x'.

3 Part 3: Find the Best Model

We start with a multiple linear regression model we got from part 2.

3.1 1 Best Model

```
model_part2_final <- lm(log(price) ~ log(carat) + cut + color + clarity, data = diamonds)
summary(model_part2_final)
```

```
##
## Call:
## lm(formula = log(price) ~ log(carat) + cut + color + clarity,
##     data = diamonds)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.01106 -0.08635 -0.00022  0.08340  1.94777
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   7.856853   0.005758 1364.46   <2e-16 ***
## log(carat)    1.883716   0.001129 1668.79   <2e-16 ***
## cutGood       0.080048   0.003890   20.58   <2e-16 ***
## cutIdeal      0.161218   0.003548   45.44   <2e-16 ***
## cutPremium    0.139353   0.003579   38.94   <2e-16 ***
## cutVery Good  0.117209   0.003619   32.39   <2e-16 ***
## colorE       -0.054280   0.002118  -25.62   <2e-16 ***
## colorF       -0.094587   0.002142  -44.16   <2e-16 ***
## colorG       -0.160377   0.002097  -76.49   <2e-16 ***
## colorH       -0.251071   0.002225 -112.85   <2e-16 ***
## colorI       -0.372573   0.002492 -149.51   <2e-16 ***
## colorJ       -0.510982   0.003074 -166.24   <2e-16 ***
## clarityIF     1.113731   0.006030  184.70   <2e-16 ***
## claritySI1    0.592963   0.005149  115.17   <2e-16 ***
## claritySI2    0.427879   0.005178   82.64   <2e-16 ***
## clarityVS1    0.812277   0.005257  154.53   <2e-16 ***
## clarityVS2    0.742158   0.005177  143.34   <2e-16 ***
## clarityVVS1   1.018743   0.005575  182.74   <2e-16 ***
## clarityVVS2   0.947272   0.005418  174.83   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1338 on 53924 degrees of freedom
## Multiple R-squared:  0.9826, Adjusted R-squared:  0.9826
## F-statistic: 1.693e+05 on 18 and 53924 DF,  p-value: < 2.2e-16
```

Our linear regression model has an adjusted R-squared value of 0.9826, which is a good model. This suggests that 98.26% of the variation in price can be explained all of the independent variables we included. However, we still want to see if we can obtain a better model.

Therefore, we apply Akaike Information Criterion backward elimination to see if we can identify an multiple linear regression model that has stronger predictive power.

```r
step(model_part2_final, direction = "backward")
```

```
## Start:  AIC=-216984.9
## log(price) ~ log(carat) + cut + color + clarity
##
##              Df Sum of Sq   RSS     AIC
## <none>                       965 -216985
## - cut         4        60  1025 -213762
## - color       6       897  1862 -181559
## - clarity     7      1838  2803 -159500
## - log(carat)  1     49855 50820   -3181
```

```
##
## Call:
## lm(formula = log(price) ~ log(carat) + cut + color + clarity,
##     data = diamonds)
##
## Coefficients:
##  (Intercept)    log(carat)       cutGood      cutIdeal    cutPremium
##      7.85685       1.88372       0.08005       0.16122       0.13935
## cutVery Good        colorE        colorF        colorG        colorH
##      0.11721      -0.05428      -0.09459      -0.16038      -0.25107
##       colorI        colorJ     clarityIF    claritySI1    claritySI2
##     -0.37257      -0.51098       1.11373       0.59296       0.42788
##   clarityVS1    clarityVS2   clarityVVS1   clarityVVS2
##      0.81228       0.74216       1.01874       0.94727
```

After running the Akaike Information Criterion backward elimination, it returns us with the same model, so our model is the best model.

3.2 2 Confidence Interval and Prediction Interval

```r
# CI for all rows
CI_log <- predict(model_part2_final, interval = "confidence", level = 0.95)
CI_original <- exp(CI_log)
head(CI_original,10) # first 10 rows
```

```
##         fit      lwr      upr
## 1  276.7784 275.4582 278.1049
## 2  269.0965 267.8606 270.3381
## 3  374.8196 372.7340 376.9170
## 4  417.3753 415.2250 419.5368
## 5  283.6160 281.5460 285.7012
## 6  305.5365 303.4095 307.6784
## 7  376.8884 374.5454 379.2460
## 8  323.2575 321.7156 324.8068
## 9  296.6471 294.3341 298.9783
## 10 319.5187 317.8900 321.1558
```

```r
# creating a new data X
new_data <- data.frame(
  carat = 0.78,
  color = factor("F", levels = levels(diamonds$color)),
  clarity = factor("SI1", levels = levels(diamonds$clarity)),
  cut = factor("Ideal", levels = levels(diamonds$cut))
)

# Calculate the CI for the mean predicted value
mean_prediction <- predict(model_part2_final, new_data, interval = "confidence", level = 0.95)
mean_prediction_original <- exp(mean_prediction)
mean_prediction_original
```

```
##        fit      lwr     upr
## 1 3128.835 3117.094 3140.62
```

```r
# PI for all Rows
PI_log <- predict(model_part2_final, interval = "prediction", level = 0.95)
```

```
## Warning in predict.lm(model_part2_final, interval = "prediction", level = 0.95): predictions on curr
```

```r
PI_original <- exp(PI_log)
head(PI_original,10) # first 10 rows
```

```
##         fit      lwr      upr
## 1  276.7784 212.9222 359.7852
## 2  269.0965 207.0133 349.7984
## 3  374.8196 288.3396 487.2371
## 4  417.3753 321.0794 542.5517
## 5  283.6160 218.1695 368.6951
## 6  305.5365 235.0338 397.1876
## 7  376.8884 289.9268 489.9335
## 8  323.2575 248.6781 420.2036
## 9  296.6471 228.1903 385.6409
## 10 319.5187 245.8003 415.3461
```

```r
# Calculate the PI for the future predicted value
future_prediction <- predict(model_part2_final, new_data, interval = "prediction", level = 0.95)
future_prediction_original <- exp(future_prediction)
future_prediction_original
```

```
##        fit      lwr      upr
## 1 3128.835 2407.014 4067.116
```

3.3 3 Summary

We initially constructed a simple linear regression model with 'carat' predicting 'price'. However, we applied a log transformation to both 'price' and 'carat' to address heteroskedasticity and ensure a linear relationship. Later, by simply adding variables and observing changes in the adjusted R-squared, we identified the variables we wanted to include. Additionally, we applied the Variance Inflation Factor (VIF) to detect multicollinearity.

Finally, we employed Akaike Information Criterion backward elimination, which confirmed that our original model was the best fit. The final fitted model can be represented by the following equation:

$$
\begin{aligned}
\text{price} \approx\ & 7.856853 + 1.883716 \cdot \log(\text{carat}) + 0.080048 \cdot \mathrm{I}_{\text{cut=Good}} + 0.161218 \cdot \mathrm{I}_{\text{cut=Ideal}} \\
& + 0.139353 \cdot \mathrm{I}_{\text{cut=Premium}} + 0.117209 \cdot \mathrm{I}_{\text{cut=Very Good}} - 0.054280 \cdot \mathrm{I}_{\text{color=E}} - 0.094587 \cdot \mathrm{I}_{\text{color=F}} \\
& - 0.160377 \cdot \mathrm{I}_{\text{color=G}} - 0.251071 \cdot \mathrm{I}_{\text{color=H}} - 0.372573 \cdot \mathrm{I}_{\text{color=I}} - 0.510982 \cdot \mathrm{I}_{\text{color=J}} \\
& + 1.113731 \cdot \mathrm{I}_{\text{clarity=IF}} + 0.592963 \cdot \mathrm{I}_{\text{clarity=SI1}} + 0.427879 \cdot \mathrm{I}_{\text{clarity=SI2}} + 0.812277 \cdot \mathrm{I}_{\text{clarity=VS1}} \\
& + 0.742158 \cdot \mathrm{I}_{\text{clarity=VS2}} + 1.018743 \cdot \mathrm{I}_{\text{clarity=VVS1}} + 0.947272 \cdot \mathrm{I}_{\text{clarity=VVS2}}
\end{aligned}
$$