Data science project of starting new investment in jewelry company by reselling diamonds

Prepared by:

Maha Fahad Modawi 441814106

Sarah Saad Alhumaidi 441800134

Data science project of starting new investment in jewelry company by reselling diamonds

1. Idea of the project

A famous jewelry company wants to to expand its business by starting a new investment in reselling diamonds. The company wants to study the status of the diamond market, so they want to use predictive modeling to estimate how much the market will pay for diamonds.

Of course, to sell diamonds in the market, first they must buy them from the producers. This is where predictive modeling becomes useful.

Let's say we know ahead of time that we will be able to sell a specific diamond in the market for a certain price. With that information, we know how much to pay when buying this diamond. If someone tries to sell that diamond to us for less than the price we well sell it by, then that would be a very good deal; on the other hand, it would be a bad deal to pay more than the certain price for such a diamond. Therefore, it would be very important to the company be able to predict the price the market will pay for diamonds accurately.

The end goal is to build a predictive model of diamonds that is going to help the company to determine whether a given diamond is a good deal or a theft deal.

1.1. Dataset Selection

We been able to get a dataset containing the prices and key characteristics of about 53,940 diamonds

	carat	cut	color	clarity	depth	table	price	x	y	z
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	0.29	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
7	0.24	Very Good	I	VVS1	62.3	57	336	3.95	3.98	2.47
8	0.26	Very Good	H	SI1	61.9	55	337	4.07	4.11	2.53
9	0.22	Fair	E	VS2	65.1	61	337	3.87	3.78	2.49
10	0.23	Very Good	H	VS1	59.4	61	338	4	4.05	2.39

The Features contained in the dataset:

Variable	**Description**
Price	Price (326 - 18,823), there no information about the currency so we assumed it to be in US Dollars
Carat	Weight of the diamond (0.2 - 5.01)
Cut	Quality of the cut (Fair, Good, Very Good, Premium, Ideal)
Color	Diamond color, from J (worst) to D (best). D,E,F mean colorless diamonds and F,G,H,I,J mean near colorless.
Clarity	Measurement of how clear the diamond is (I1 (worst), SI2, SI1, VS2, VS1, VVS2, VVS1, IF (best))
X	Length in mm (0 - 10.74)
Y	Width in mm (0 - 58.9)
Z	Depth in mm (0 - 31.8)
Depth	Total depth percentage = z
Table	Width of top of diamond relative to widest point (43 - 95)

1.2.Goals of our project:

- To use the features contained in the dataset
- To build a predictive model that predicts the price of diamonds, as accurately as possible, based on those features
- To predict the prices of diamonds offered to company by the producers, so company can decide how much to pay for those diamonds

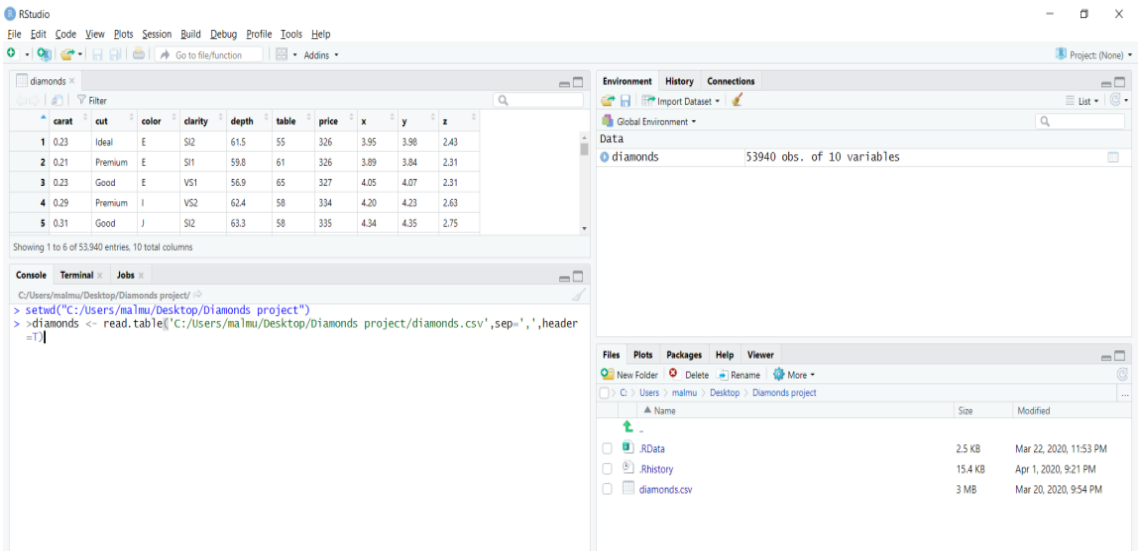
2. Explore Dataset & Examine what Features affect the Price of Diamonds

2.1.Extract Dataset

We will work on dataset diamonds, download the data set into our working directory, set directory and load it into our workspace.

While our data in well-structured we pre-prepared dataset use command read.table() to load the well-structured data and store it in a new R data frame object (Diamonds)

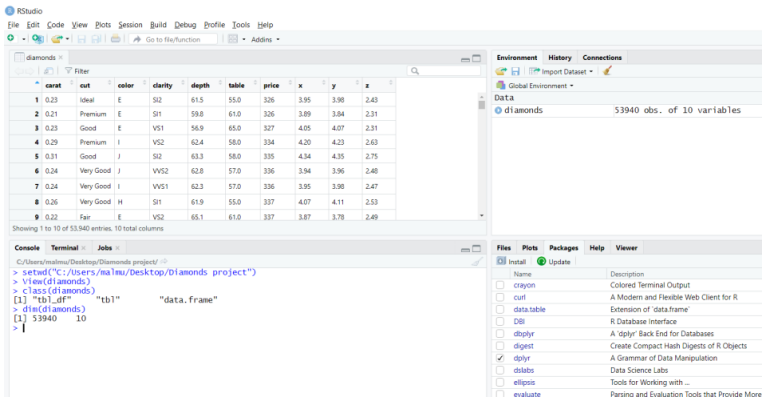
```
> setwd("C:/Users/malmu/Desktop/Diamonds project")
> diamonds <- read.table('C:/Users/malmu/Desktop/Diamonds
project/diamonds.csv',sep=',',header=T)
 > View(diamonds)
```



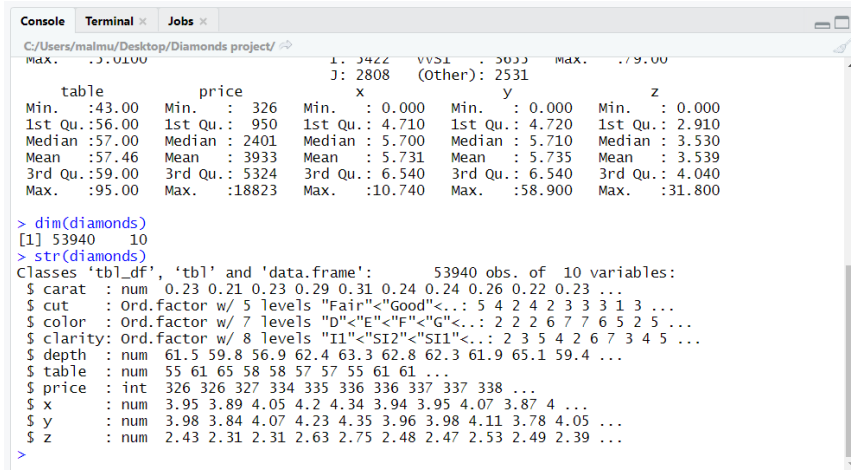
Once we've loaded the data into R, we want to explore and visualize it by using R commands to get know more about our dataset and what Features affect the Price of Diamonds:

2.2. Descriptive statistic to Explore dataset

- `> dim(diamonds)` -- for data frames to show as how many rows and columns are in our data.



- `> str(diamonds)` -- To reviews basic internal structure of the dataset



```
> dim(diamonds)
[1] 53940    10
> str(diamonds)
Classes 'tbl_df', 'tbl' and 'data.frame':       53940 obs. of  10 variables:
 $ carat  : num  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  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 ...
>
```

The diamonds dataset contains 53940 rows and 10 columns (variables). Each rows correspond to a sample diamond. The dataset consists of 3 categorical variables, 1 integer variable, and 6 numeric variables.

- The categorical variables are:
  - Cut
  - Color
  - Clarity
- The Integer variables is:
  - Price
- Numeric variables are:
  - Carat
  - Depth
  - Table
  - X
  - Y
  - Z

- `> summary(diamonds)` -- used to display several basic descriptive statistic summaries of either one variable or an entire data frame

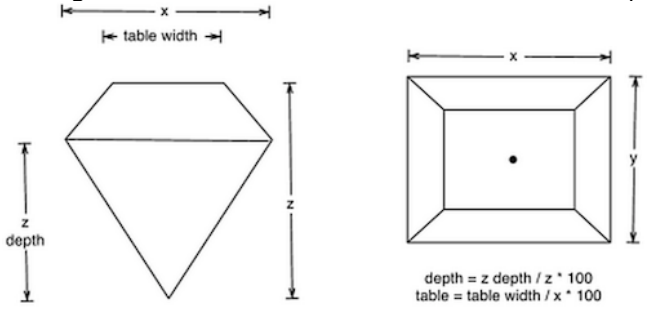
```
Console  Terminal  Jobs
C:/Users/malmu/Desktop/Diamonds project/
 $ clarity: Ord.factor w/ 8 levels "I1" < "SI2" < "SI1" <...: 2 3 5 4 2 6 7 3 4 5 ...
 $ depth  : num  61.5 59.8 56.9 62.4 63.3 62.8 62.3 61.9 65.1 59.4 ...
 $ table  : num  55 61 65 58 58 57 57 55 61 61 ...
 $ price  : int  326 326 327 334 335 336 336 337 337 338 ...
 $ 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 ...
> 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                z
 Min.   :43.00   Min.   :  326   Min.   : 0.000   Min.   : 0.000   Min.   : 0.000
 1st Qu.:56.00   1st Qu.:  950   1st Qu.: 4.710   1st Qu.: 4.720   1st Qu.: 2.910
 Median :57.00   Median : 2401   Median : 5.700   Median : 5.710   Median : 3.530
 Mean   :57.46   Mean   : 3933   Mean   : 5.731   Mean   : 5.735   Mean   : 3.539
 3rd Qu.:59.00   3rd Qu.: 5324   3rd Qu.: 6.540   3rd Qu.: 6.540   3rd Qu.: 4.040
 Max.   :95.00   Max.   :18823   Max.   :10.740   Max.   :58.900   Max.   :31.800

> I
```

- ❖ The carat variable represents Wight of the diamond, we can see the minimum= 0.2000 and maximum= 5.0100 and the median=0.7000
- ❖ The Cut variable represent Quality of diamond cut, we can see about quarter of data is premium and about other quarter is very good, and 21551 of data is good
- ❖ The color represents diamond color, we can see it's start from D=6775 until J=2808
- ❖ The Clarity variable represent Measurement of how clear the diamond is, we can see there's 2531 values with no letter and it contains letters SI1, VS2, SI2, VSI, VVS2, VVS1. This means that clarity needs more investigation
- ❖ The depth variable represents Total depth of diamond, we can see the minimum= 0.2000 and maximum= 5.0100 and the median=0.7000
- ❖ The X variable represent Length in mm of diamond, we can see the minimum= 0.000 and maximum= 10.740 and the median=5.731
- ❖ The Y, Z variables represent width and depth of diamond in mm

Before start preparation our data we need to know more about the dataset and their features:

The dataset contains information on prices of diamonds, as well as various attributes of diamonds, the 4 Cs (carat, cut, color, and clarity) , as well as some physical measurements (depth, table, price, x, y, and z).

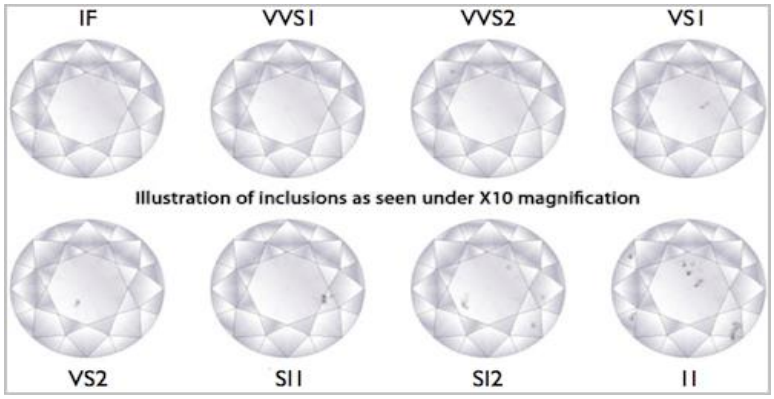
The figure below shows what these measurements represent.



Carat is a unit of mass equal to 200 mg and is used for measuring gemstones and pearls. Cut grade is an objective measure of a diamond's light performance, or, what we generally think of as sparkle.

The figures below shows color grading of diamonds:

Lastly, the figure below shows clarity grading of diamonds:



For the problem we have defined, the target is the price of the diamond, and our features will be the nine remaining columns in our dataset: carat, cut, color, clarity, x, y, z, depth, and table.

- `> summary(diamonds$price)` -- To know summary of the minimum, maximum , and average prices in the market
-

```
 $ 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 ...
> summary(diamonds$price)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    326     950    2401    3933    5324   18823
>
```

We can see the Price variable represent Price of the diamond in market, the minimum= 326, maximum= 18823 and mean= 3933

- Check missing values for each column in data frame

```
> count_missing = function(df) {
+     sapply(df, FUN=function(col) sum(is.na(col)) )
+ }
> nacounts <- count_missing(diamonds)
> hasNA = which(nacounts > 0)
> nacounts[hasNA]
```

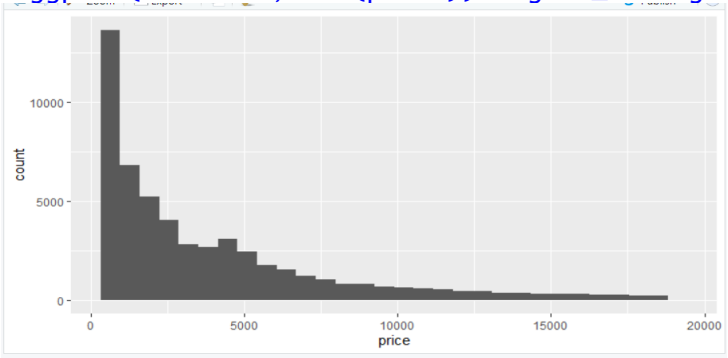
```
> count_missing = function(df) {
+     sapply(df, FUN=function(col) sum(is.na(col)) )
+ }
> nacounts <- count_missing(diamonds)
> hasNA = which(nacounts > 0)
> nacounts[hasNA]
named integer(0)
> 
```

After count number of missing value in each column of dataset we can see its zero missing value which is great !

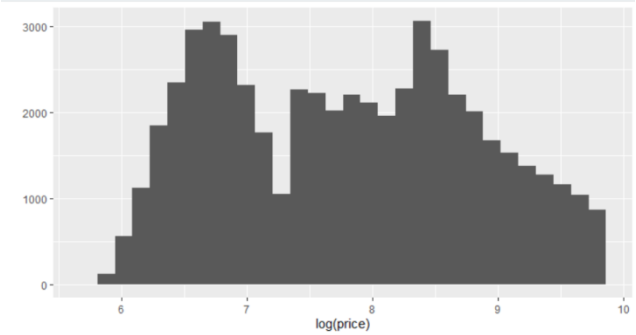
2.3.Visualization to Explore dataset

- First we use frequency distribution for a single variable(price), which tells us how often particular values of a variable occur in a dataset. we want to know about the price breakdown of the diamonds in the dataset.

```
> ggplot(diamonds, aes(price)) + geom_histogram()
```



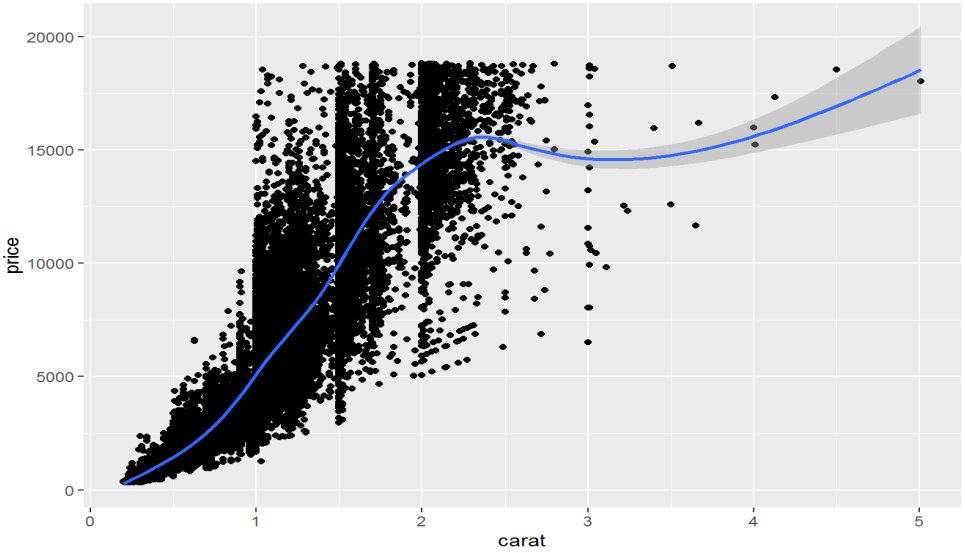
The distribution is highly skewed, with lots of cheaper diamonds and fewer expensive ones.
We take the log for price to see the normal distribution



The distribution look less long-tail and more normal.

- We use ggplot2 to create a scatter plot where we put carat, or weight, on the x axis and price, in dollars, on the y axis

```
> ggplot(diamonds, aes(x=carat, y=price)) + geom_point()
```

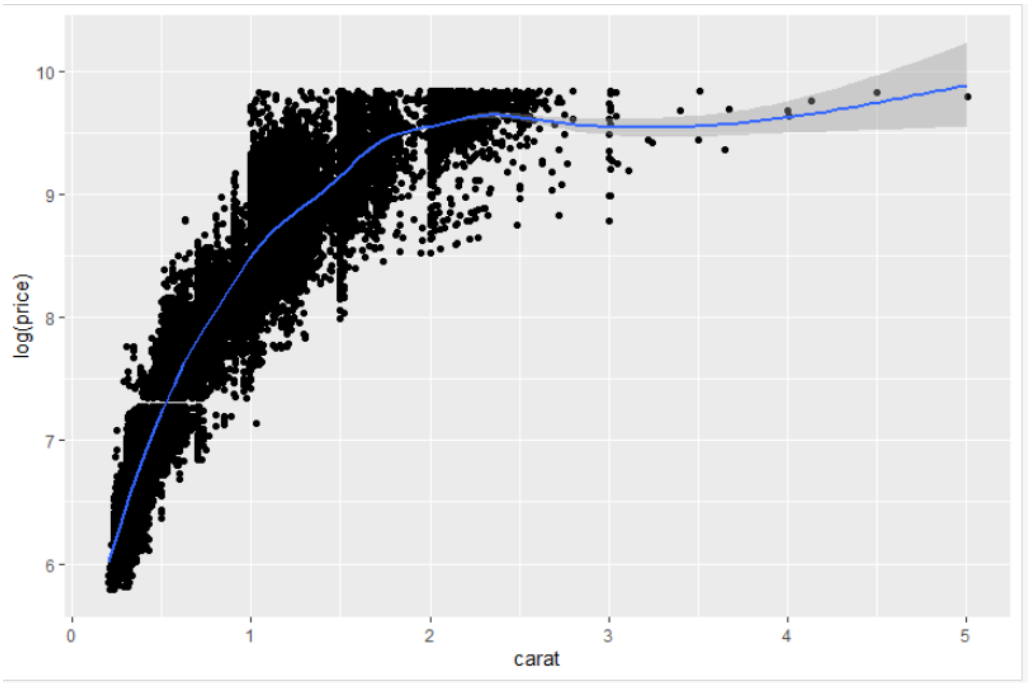


We see the relationship between price and carat clearly, it increases as carat size increases. The shape of the curve provides additional evidence that the relationship is exponential but after the 2-carat mark; the relationship flattens out. This raises a question because we have no prior reason to expect that the relationship between carat and price would fall off after two carats.

An explanation could be that there seems to be a ceiling in price. It turns out, this dataset only includes diamonds that are below a certain price(18000$) This means for very large diamonds, there is a selection bias for cheaper diamonds relative to the size.

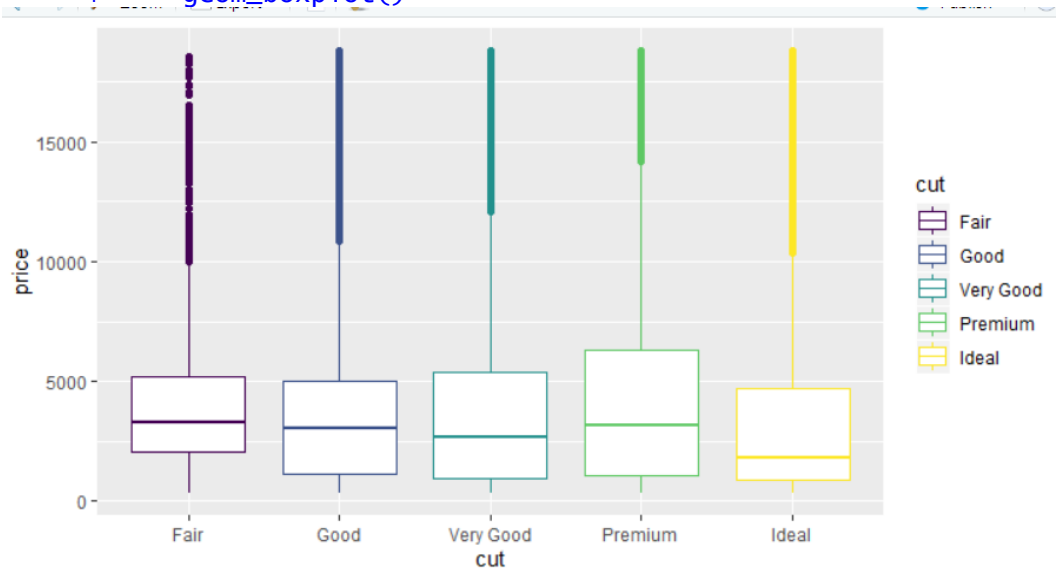
Taking the log10(price) will show the relationship more clearly, this proves the exponential coloration between price and carat

```
> ggplot(diamonds, aes(x=carat, y=log10(price))) +
geom_point()
```



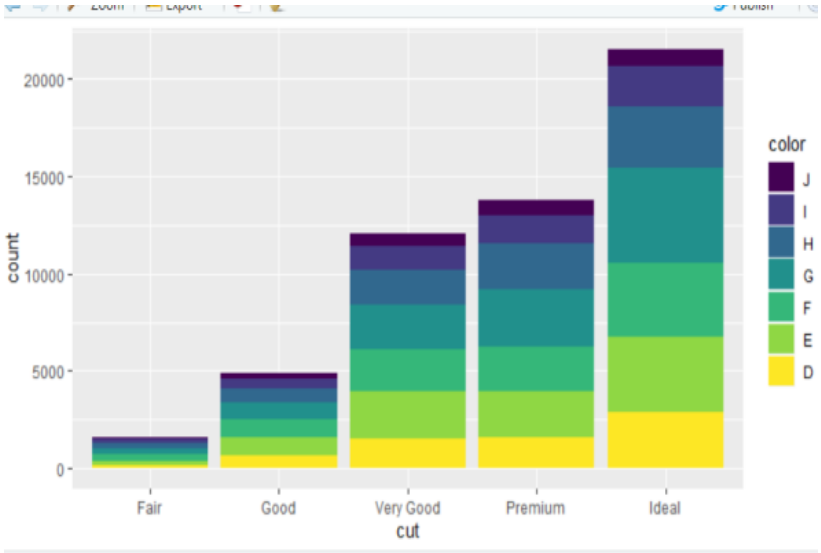
- Now, we explore the relationship between cut and price by using box plot

```
> diamonds %>%
+     ggplot(aes(x=cut,y=price, color=cut)) +
+     geom_boxplot()
```



We observe that the highest price diamonds are premium cut, lowest price diamonds are ideal cut.

- Now, we explore the relationship between cut and color by using bar chart
- `> ggplot(diamonds,aes(x=cut, fill = color))+geom_bar()`



We observe that the highest cut diamonds are ideal cut with above 20000 diamond, lowest cut diamonds are fair cut with less than 2500 diamond.

3.Data preparation

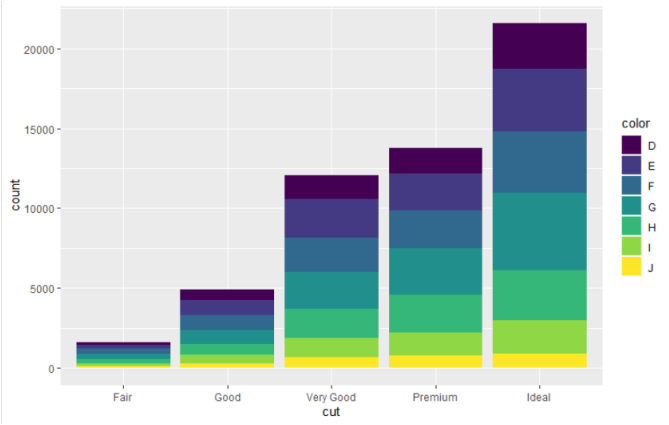
Re-ordering factor:

we will use the levels() function to check the ordering of the factor levels and we then applied the ordered() function to manually re-order the factor levels to the order that would make our downstream analysis more consistent. The benefit will be more prominent when we use layers in the graph plotting , we will see the graphs display in the worst-to-best order, in a consistent manner.

- We will First, check levels of color on color variables before reorder :

```
C:/Users/malmu/Desktop/Diamonds project/
> levels(diamonds$color)
[1] "D" "E" "F" "G" "H" "I" "J"
>
```

Contains the 7 factors (D, E, F, G, H, I, J) in which D being best and J being worst, for consistency it may be better to reverse the order so the order is always from worse (J) to best (D). we apply factor re-ordering to ensure consistent ordering of scale from worst to best.



The ordered factor for color is now ordered from leftmost (worst) to rightmost (best). As we see it also on graph plot .

- The same thing to re-order the factor levels for the variable clarity, so that the order is shown consistent from worst to best, the order to be: I1, SI1, SI2, VS1, VS2, VVS1, VVS2, IF.

```
> levels(diamonds$clarity)
[1] "I1"   "SI2"  "SI1"  "VS2"  "VS1"  "VVS2" "VVS1" "IF"
> diamonds$clarity <- ordered(diamonds$clarity, levels = c("I1", "SI1", "SI2", "VS1", "VVS1","VS
2", "VVS2", "IF"))
> levels(diamonds$clarity)
[1] "I1"   "SI1"  "SI2"  "VS1"  "VVS1" "VS2"  "VVS2" "IF"
>
```

In the first levels() function the order requires fixing. So that leftmost scale is worst and rightmost scale is best. For example:

- The order of SI2 and SI1 need switching, because SI2 is "better than" SI1.

- The order of VS2 and VS1 need switching, because VS2 is " better than " SI1.
- The order of VVS2 and VVS1 need switching, because VVS2 is " better than" VVS1.

At the end the ordered factor for clarity is now ordered from leftmost (worst) to rightmost (best).

The ordering helps to know the reference level for categorical variable , by default R takes the first one.

```
> summary(diamonds)
     carat               cut        color        clarity          depth      
 Min.   :0.2000   Fair      : 1610   J: 2808   SI1     :13065   Min.   :43.00  
 1st Qu.:0.4000   Good      : 4906   I: 5422   VS2     :12258   1st Qu.:61.00  
 Median :0.7000   Very Good:12082   H: 8304   SI2     : 9194   Median :61.80  
 Mean   :0.7979   Premium   :13791   G:11292   VS1     : 8171   Mean   :61.75  
 3rd Qu.:1.0400   Ideal     :21551   F: 9542   VVS2    : 5066   3rd Qu.:62.50  
 Max.   :5.0100                     E: 9797   VVS1    : 3655   Max.   :79.00  
                                    D: 6775   (Other): 2531                   
     table           price             x                y                z          
 Min.   :43.00   Min.   :  326   Min.   : 0.000   Min.   : 0.000   Min.   : 0.000  
 1st Qu.:56.00   1st Qu.:  950   1st Qu.: 4.710   1st Qu.: 4.720   1st Qu.: 2.910  
 Median :57.00   Median : 2401   Median : 5.700   Median : 5.710   Median : 3.530  
 Mean   :57.46   Mean   : 3933   Mean   : 5.731   Mean   : 5.735   Mean   : 3.539  
 3rd Qu.:59.00   3rd Qu.: 5324   3rd Qu.: 6.540   3rd Qu.: 6.540   3rd Qu.: 4.040  
 Max.   :95.00   Max.   :18823   Max.   :10.740   Max.   :58.900   Max.   :31.800  

> summary(diamonds$clarity)
   I1   SI1   SI2   VS1  VVS1   VS2  VVS2    IF 
  741 13065  9194  8171  3655 12258  5066  1790 
> 
```

- For skewed and wide distribution in price factor we use log transformation for a more normal distribution as done in (Visualization to Explore dataset)

- For variables regarding the dimension of the diamonds: x, y, and z :
  The first thing we notice is that the minimum values for these features are zero. From what these variables represent, we know this can't be possible.

  Let's examine the values of x that are equal to zero:

```
> head(subset(diamonds, x == 0))
# 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  1.07 Ideal     F     SI2      61.6    56  4954     0  6.62     0
2  1    Very Good H     VS2      63.3    53  5139     0  0        0
3  1.14 Fair      G     VS1      57.5    67  6381     0  0        0
4  1.56 Ideal     G     VS2      62.2    54 12800     0  0        0
5  1.2  Premium   D     VVS1     62.1    59 15686     0  0        0
6  2.25 Premium   H     SI2      62.8    59 18034     0  0        0
> 
```

So, it turns out that we have diamonds with x = 0 or y = 0 or z = 0 and that doesn't seem to make sense to have a diamond with zero values for any of these variables. It is logical to remove these observations from the dataset entirely. i.e. any observations where x or y or z is less than or equal to zero.

```
>  subset(diamonds, !(x <= 0 | y <= 0 | z <= 0))
# A tibble: 53,920 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
 7 0.24  Very Good I     VVS1     62.3    57   336  3.95  3.98  2.47
 8 0.26  Very Good H     SI1      61.9    55   337  4.07  4.11  2.53
 9 0.22  Fair      E     VS2      65.1    61   337  3.87  3.78  2.49
10 0.23  Very Good H     VS1      59.4    61   338  4     4.05  2.39
# ... with 53,910 more rows
> summary(diamonds$x)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  3.730   4.710   5.700   5.732   6.540  10.740 
> summary(diamonds$y)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  3.680   4.720   5.710   5.735   6.540  58.900 
> summary(diamonds$z)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
   1.07    2.91    3.53    3.54    4.04   31.80 
> 
```

Minimum value is no longer zero.

4.Model Building

Our objective will be to use R to build a model that predicts the price of a diamond. The most relevant features of diamonds are carat, cut, color, and clarity (the 4 C's). Since we have this data available to use, we should be able to build a highly accurate model.
Since we are predicting a continuous variable (price), we are trying to solve a regression problem; in predictive analytics, when the target is a numerical variable, we are within a category of problems known as regression tasks. For that reason, our methodology will be building a linear regression model.
The dataset is already cleaned and almost ready for analysis. We will split the data twice; first split would be 50-50 and the second will be 30-70 and compare both results which would initially help confirming that our model generated works well.
For input , we will take the most important features to determine the price in diamond world which are carat, cut, color, and clarity.

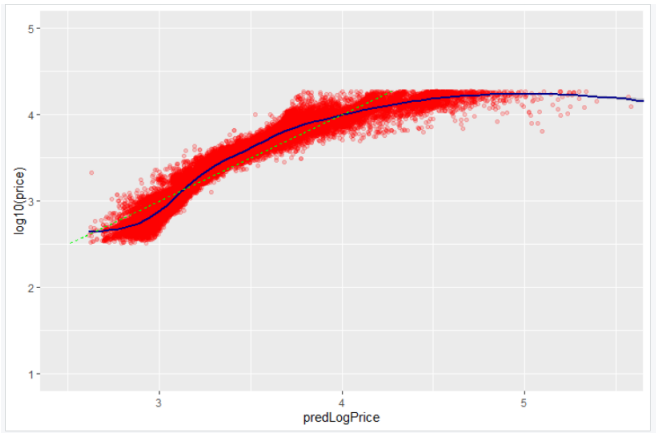
Building first model with 50/50 split:

```
> set.seed(235235)
>  p <- runif(nrow(diamond))
>   dtrain <- subset(diamond, p >= 0.5)
>   dtest <- subset(diamond, p = 0.5)
> model <- lm(log10(price) ~ carat + clarity +color + cut,
data = dtrain)
>   dtest$predLogPrice <- predict(model, newdata = dtest)
>   dtrain$predLogPrice <- predict(model, newdata = dtrain)
```

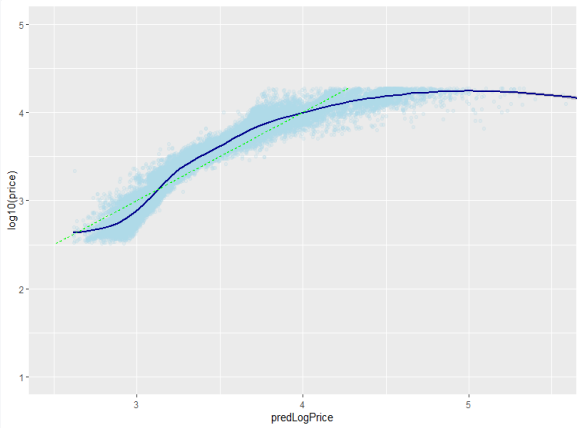
Building the same model with 30/70 split:

```
> set.seed(235235)
>  p <- runif(nrow(diamond))
>   dtrain <- subset(diamond, p >= 0.3)
>   dtest <- subset(diamond, p = 0.7)
> model <- lm(log10(price) ~ carat + clarity +color + cut,
data = dtrain)
>   dtest$predLogPrice <- predict(model, newdata = dtest)
>   dtrain$predLogPrice <- predict(model, newdata = dtrain)
```

From now on we will present the results of each split together, so it would be easier to compare them. Starting with plotting log price as a function of predicted log price



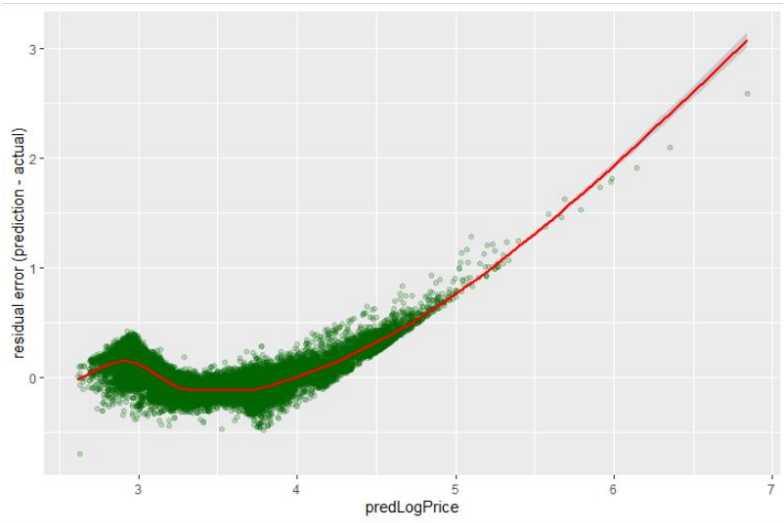
First split plot



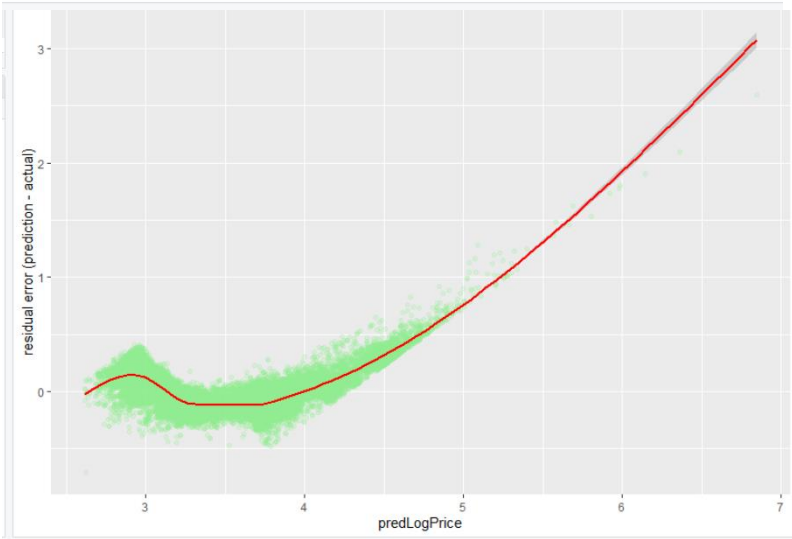
second split plot

We can see from the plot that The prediction is relatively good because most of the dots are arranged near the line x=y

Statisticians prefer the residual plot which eill be our next comparison. The residual plot shows where the residual errors (predLogPrice - log10(price)) are plotted as a function of predLogPrice.
In this case, the line of perfect prediction is the line y=0.



First split plot for residual



second split plot for residual

Notice that the points are scattered along the line of prediction and thy appear almost identical.

The previous ways are for inspecting the model using visualization methods, but there are other mathematical ways to evaluate our model's quality:

- Computing R-Squared
- Computing RMSE (Root Mean Square Error)
- Computing MAE (Mean Absolute Error)
- Using Summary function in R
- Coefficients function in R

R-Squared:

```
>   rsq <- function(y, f) { 1 - sum((y - f)^2)/sum((y - mean(y))^2) }
>   rsq(log10(dtrain$price), dtrain$predLogPrice)
[1] 0.8864234
>   rsq(log10(dtest$price), dtest$predLogPrice)
[1] 0.8884509
>
```

R-Squared for first split

```
>    rsq(log10(dtrain$price), dtrain$predLogPrice)
[1] 0.8872595
>
>    rsq(log10(dtest$price), dtest$predLogPrice)
[1] 0.8884676
>
```

R-Squared for second split

It is preferable to see R-Squared in between (0.7–1.0). The shown result are within the preferred range which means that the model is of a high quality, but could indicate the presence of overfit.

RMSE:

We can think of the RMSE as a measure of the width of the data cloud around the line of perfect prediction. We'd like RMSE to be small, and one way to achieve this is to introduce more useful, explanatory variables. That means if our variable selection is good the RMSE would be near 0.

```
>
>
> rmse (log10(dtrain$price), dtrain$predLogPrice)
[1] 0.1482075
>   rmse (log10(dtest$price), dtest$predLogPrice)
[1] 0.1471734
>
```

RMSE for first split

```
>
>    rmse (log10(dtrain$price), dtrain$predLogPrice)
[1] 0.1478303
>
>    rmse (log10(dtest$price), dtest$predLogPrice)
[1] 0.1471624
>
```

RMSE for second split

We can see that the result in each split is close to 0 .

MAE:

In statistics, mean absolute error (MAE) is a measure of errors between paired observations expressing the same phenomenon. The smaller the value of MAE the smaller the errors.

```
>  Mean_Absolute_Error(log10(dtrain$price), dtrain$predLogPrice)
[1] 0.1170501
>
>
> Mean_Absolute_Error(log10(dtest$price), dtest$predLogPrice)
[1] 0.1165416
```

MAE of first split

```
>
>    Mean_Absolute_Error(log10(dtrain$price), dtrain$predLogPrice)
[1] 0.1168079
>
>    Mean_Absolute_Error(log10(dtest$price), dtest$predLogPrice)
[1] 0.116425
>
```

MAE of second split

The value of MAE in each split is small which is good.

Summary and coefficients:
The summary function view and explain the quality of the model.
The first split summary :

```
Call:
lm(formula = log10(price) ~ carat + clarity + color + cut, data = dtrain)

Residuals:
     Min       1Q   Median       3Q      Max
-2.08890 -0.09542  0.02640  0.10827  0.47544

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   2.647239   0.006072 435.984  < 2e-16 ***
carat         0.952082   0.002187 435.342  < 2e-16 ***
clarity.L     0.301460   0.005336  56.498  < 2e-16 ***
clarity.Q    -0.147082   0.005199 -28.292  < 2e-16 ***
clarity.C     0.106981   0.004295  24.910  < 2e-16 ***
clarity^4    -0.034360   0.003515  -9.776  < 2e-16 ***
clarity^5     0.075808   0.002804  27.033  < 2e-16 ***
clarity^6    -0.111979   0.002498 -44.821  < 2e-16 ***
clarity^7     0.024292   0.002730   8.899  < 2e-16 ***
colorE       -0.026513   0.003321  -7.983 1.48e-15 ***
colorF       -0.023585   0.003356  -7.028 2.15e-12 ***
colorG       -0.058758   0.003285 -17.887  < 2e-16 ***
colorH       -0.116286   0.003512 -33.109  < 2e-16 ***
colorI       -0.183986   0.003920 -46.938  < 2e-16 ***
colorJ       -0.258130   0.004765 -54.168  < 2e-16 ***
cutGood       0.022329   0.006076   3.675 0.000238 ***
cutIdeal      0.036970   0.005537   6.677 2.49e-11 ***
cutPremium    0.025391   0.005586   4.546 5.50e-06 ***
cutVery Good  0.025556   0.005645   4.527 5.99e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.1483 on 26957 degrees of freedom
Multiple R-squared:  0.8864,    Adjusted R-squared:  0.8863
F-statistic: 1.169e+04 on 18 and 26957 DF,  p-value: < 2.2e-16
```

The second split summary:

```
Call:
lm(formula = log10(price) ~ carat + clarity + color + cut, data = dtra

Residuals:
     Min       1Q   Median       3Q      Max
-2.09404 -0.09513  0.02575  0.10801  0.48084

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   2.646786   0.005163 512.678  < 2e-16 ***
carat         0.953702   0.001841 518.153  < 2e-16 ***
clarity.L     0.305973   0.004552  67.222  < 2e-16 ***
clarity.Q    -0.151301   0.004440 -34.073  < 2e-16 ***
clarity.C     0.108596   0.003663  29.645  < 2e-16 ***
clarity^4    -0.036375   0.002984 -12.189  < 2e-16 ***
clarity^5     0.076214   0.002370  32.153  < 2e-16 ***
clarity^6    -0.110819   0.002104 -52.680  < 2e-16 ***
clarity^7     0.024278   0.002303  10.540  < 2e-16 ***
colorE       -0.027247   0.002805  -9.714  < 2e-16 ***
colorF       -0.024742   0.002832  -8.737  < 2e-16 ***
colorG       -0.058192   0.002769 -21.014  < 2e-16 ***
colorH       -0.116463   0.002954 -39.426  < 2e-16 ***
colorI       -0.183956   0.003302 -55.716  < 2e-16 ***
colorJ       -0.253855   0.004050 -62.676  < 2e-16 ***
cutGood       0.020300   0.005149   3.943 8.08e-05 ***
cutIdeal      0.034800   0.004703   7.399 1.40e-13 ***
cutPremium    0.023685   0.004746   4.991 6.04e-07 ***
cutVery Good  0.023608   0.004796   4.923 8.56e-07 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.1479 on 37709 degrees of freedom
Multiple R-squared:  0.8873,    Adjusted R-squared:  0.8872
F-statistic: 1.649e+04 on 18 and 37709 DF,  p-value: < 2.2e-16
```

The summary explains the model in general, it has four sections:

- First one shows the model's formula
- Second one shows the residuals
- Third one is the coefficients of the model
- The last one shows more quantitative measures for the model

Both summaries show significant results indicating that our model performs quite well.

We expect the residuals to be small, which appears to be the situation in this model, and the median is near 0.

In the coefficients part:

All our variables are related to the prediction, for example, let us discuss the color variable. we know that it is a categorical variable this means that it will be divided into levels from D to J and because we rearranged the level previously the D level became our reference level that is why the estimation has negative sign; they are compared to D the highest level so they would lower the price. On the other hand, the cut variable levels are compared to the fair level (the lowest) so they would increase the price that's why they have positive values.

The p-value estimates the probability of seeing a coefficient with a magnitude as large as we observed if the true coefficient is really zero indicating that the variable has no effect on the outcome(the price), in our model all variables has effect on the price. The common threshold is $p < 0.05$ and all variables are less than the threshold. Multiple R-squared is just the R-squared of the model on the training data. The adjusted R-squared is the multiple R-squared penalized for the number of input variables. They both have values near 1 which is required for a good model.

Problems with the model:

1. There may be a presence of overfitting.
2. The model will predict well within the range of variables, but we cannot judge it performance beyond that
3. Global economy effects diamond prices and we did not include that in our model which means that the company should be aware of that.