Project Proposal - Predicting Diamond Prices Using Regression Analysis on Fair-Cut Diamonds

Ajay Aryan, Deepak Dulal, Rahul Sen

Section 1: Introduction

Objective

The primary aim of this project is to develop a predictive model for diamond prices, leveraging various physical and quality attributes of the diamonds. This model is intended to assist both buyers and sellers by establishing a transparent pricing mechanism that reflects the intrinsic qualities of the diamonds. The data for this analysis has been compiled by Karnika Kapoor and is publicly available on Kaggle.

Data Description

Our dataset contains detailed records of approximately 53,000 diamonds, encompassing attributes such as carat weight, color, clarity, and physical dimensions (length x, width y, and depth z). For the purpose of focused analysis, we have narrowed our study to a subset of 1,610 diamonds classified with a "Fair" cut quality. This subset allows us to scrutinize the impact of specified attributes on the pricing within this particular cut category, providing insights that are applicable to a significant segment of the diamond market.

Data dictionary:

The data set contains information about diamonds, with the following columns:

1. price: The target variable, representing the price of the diamond in US dollars.
2. carat: The weight of the diamond, measured in carats.
3. color: Diamond color, ranging from J (worst) to D (best).
4. clarity: A measurement of how clear the diamond is, with categories ranging from I1 (worst) to IF (best).
5. x: Length of the diamond in mm.
6. y: Width of the diamond in mm.
7. z: Depth of the diamond in mm.
8. depth: Total depth percentage, calculated as the ratio of z to the mean of x and y.
9. table: The table percentage of the diamond (Width of the top of the diamond relative to the widest point.)

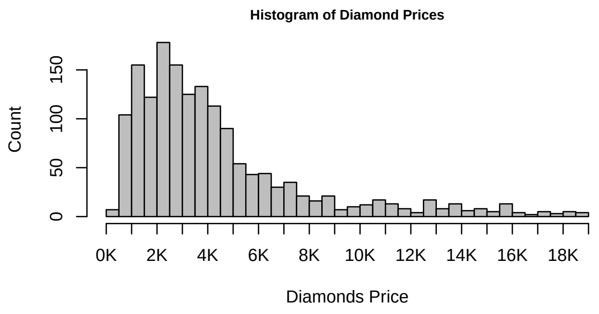
Exploratory Data Analysis (EDA)

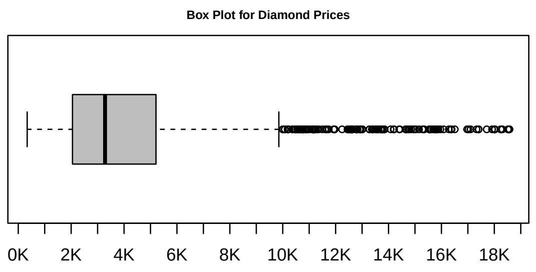
The overarching question guiding this project is: How well can we predict the price of a diamond based on its attributes? This question reflects the broader goal of the model, which is to provide a reliable tool for estimating diamond prices. By developing a robust predictive model, we aim to demystify diamond pricing and offer a valuable resource for market participants. The success of our model will be measured by its accuracy in predicting prices and its ability to handle the complex interplay of factors that determine a diamond's value.

Our initial exploratory data analysis (EDA) has been instrumental in identifying key trends and relationships within the data. Notably, the price distribution of diamonds is right-skewed, which suggests that while most diamonds are priced within a lower range, the higher-priced diamonds are comparatively rare. This skewness in price distribution warrants the application of models that can adeptly handle outliers and non-linearity. Additionally, our analysis has begun to uncover the nuanced effects of color and clarity on diamond pricing. These attributes, while categorical, exhibit a varied influence on price, suggesting that different grades significantly affect the market value of diamonds. The EDA includes detailed visualizations such as histograms and box plots that illustrate these relationships, highlighting both the typical values and the outliers.

Loading our data set and displaying the top 10 records:

Histogram and Box Plot for diamond's price

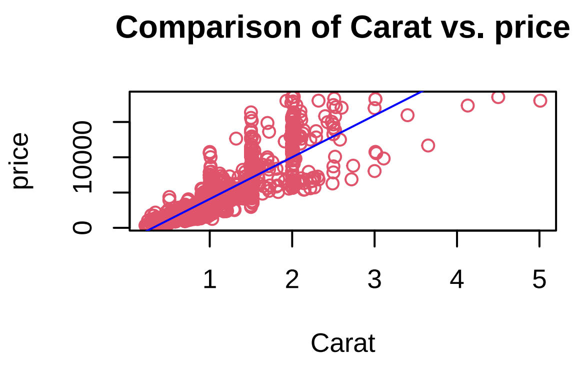


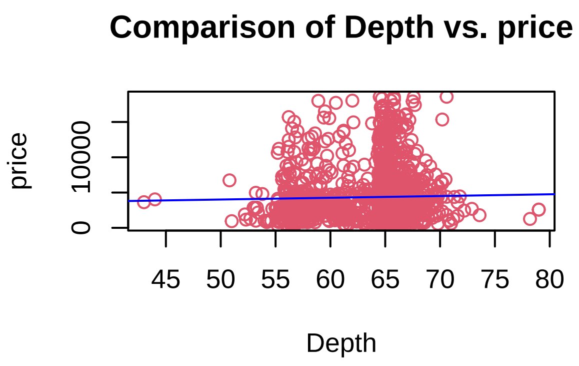


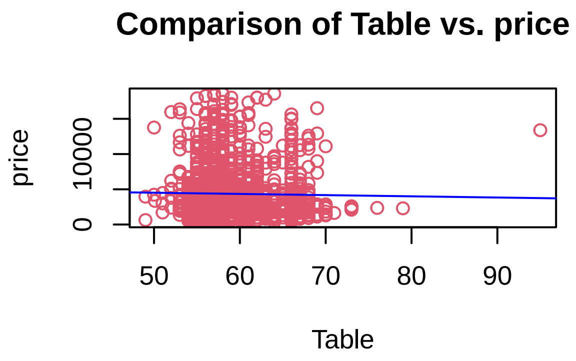
Histogram Explanation: The histogram displays a right-skewed distribution of diamond's prices, with the bulk of diamonds falling within a lower diamond's price range. As the diamond's price increases, the frequency of diamonds decreases, indicating that high-diamonds priced diamonds are comparatively rare.

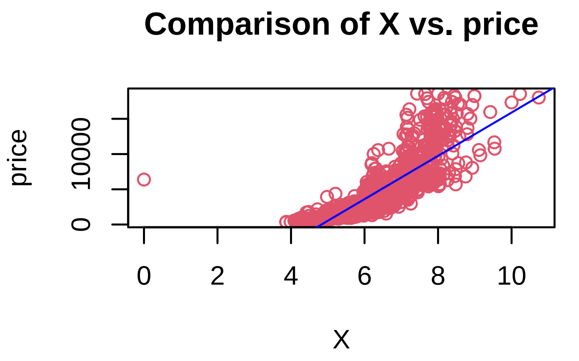
Box Plot Explanation: The box plot illustrates the diamonds price distribution of diamonds, with a central box representing the interquartile range where the middle 50% of the diamonds prices lie. The median diamonds price is indicated by a line within the box, slightly above the $3,000 mark, which divides the diamond set into two equal halves in terms of diamonds price. Whiskers extend from the box to the highest and lowest diamonds prices that are still within 1.5 times the interquartile range, while individual points beyond these whiskers are labeled as outliers, signifying diamonds$prices that are unusually high or low. This plot reveals a right-skewed distribution, evidenced by a longer right whisker and several high-diamonds priced outliers, indicating that a selection of diamonds has diamonds prices considerably higher than the general collection.

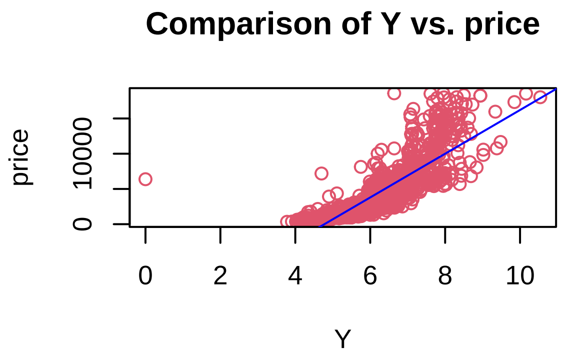
Scatter Plot for Response Vs Each Predictor

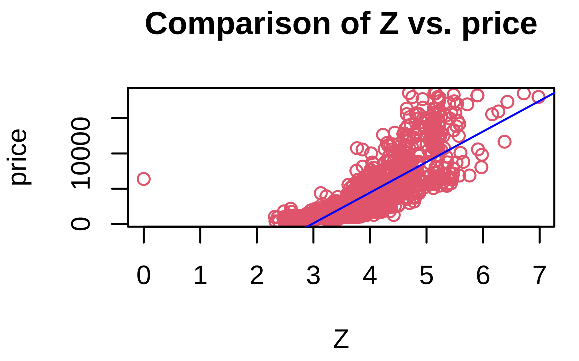




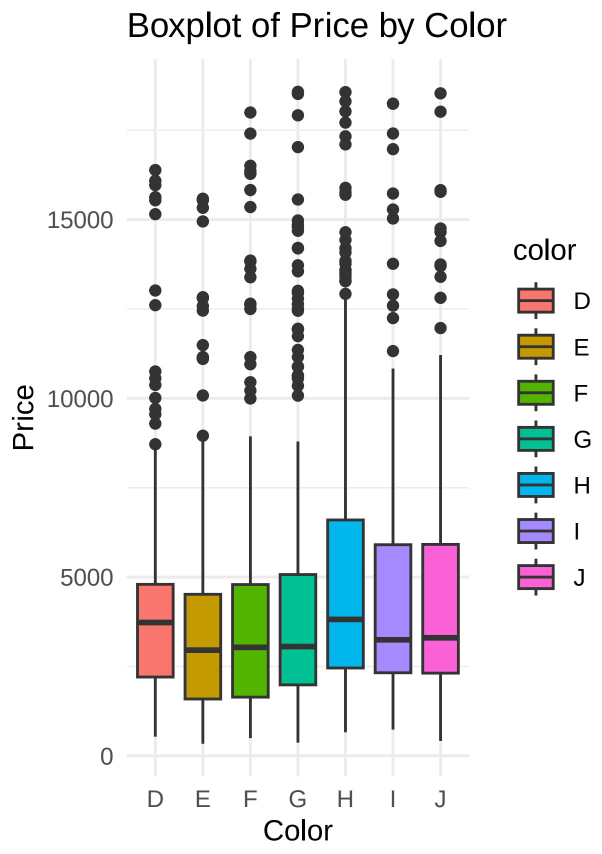


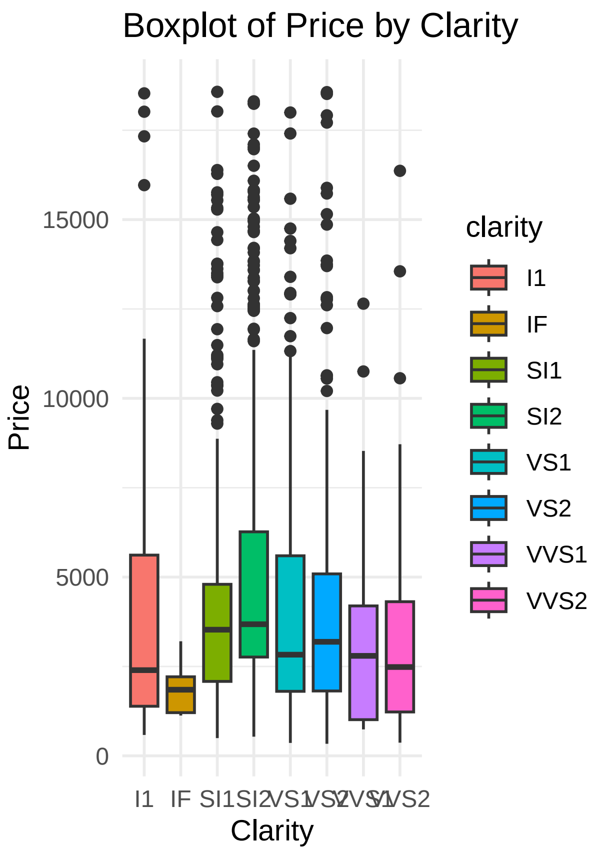






Box Plot for Categorical Predictors





The relationship between diamond predcitors and price varies across different attributes. Carat exhibits a strong positive correlation with price, indicating that as carat size increases, so does the price. Depth and table show relatively weak relationships with price, with prices showing minimal variation with changes in these attributes. X and Y dimensions demonstrate strong positive correlations, suggesting that larger lengths and widths correspond to higher prices. Z dimension also shows a positive relationship with price, though weaker than X and Y. When considering clarity and color, higher numeric values in clarity tend to correlate with higher prices, indicating better clarity. Conversely, lower numeric values in color, closer to colorlessness, are associated with higher prices, implying a potentially inverse relationship between color and price when numerically coded from best to worst.

Section 2: Regression Analysis

Data Preparation

The data required cleaning to handle missing values and errors. Categorical variables such as color and clarity were transformed into numerical factors suitable for regression analysis.

In navigating the challenge of dataset selection for our project, we opted to refine our focus from a vast pool of around 53,000 entries to a more manageable subset of approximately 1,600 observations. This subset, characterized by the "Fair" designation within the "Quality of Cut" category, strikes a balance between data set size and analytical depth. Despite the reduction in data set scale, we retained a robust set of nine predictors, excluding the serial number and response variable. This targeted approach ensures a solid foundation for our analysis, allowing for focused exploration of the relationship between predictor variables and diamond price. We remain receptive to feedback and open to refining our approach to ensure the efficacy and relevance of our analysis. As we have no missing value, so we are considering color and clarity as the categorical values.

Model Selection Process

The selection of the final model was guided by both AIC and BIC criteria, which helped in identifying the most significant predictors and eliminating overfitting. Multicollinearity was assessed using Variance Inflation Factor (VIF) scores, leading to the removal of highly correlated predictors to improve model stability.

Multi-linear Regression (MLR)

For ease, It is always better to perform MLR to know significant predictor. Thus, we are doing it in our next step.

With the help of is.factor() function color and clarity are factors. From the MLR Summary, Five of the predictors (depth, table, x, y and z) doesn't have significant relation with the response "price" when carat, color and clarity are present in the model. We'll further verify it by doing criterion process.

Model Selection using search algorithm for different Quality Criterion

For model selection, we utilized search algorithms such as forward, backward, and stepwise selection, along-side calculation of quality selection criteria(AIC, BIC, Adjusted R 2 , LOO-CV RMSE) to determine the most suitable model from a range of possibilities. Additionally, we calculated the RMSE LOOCV value for each model, providing a robust assessment of predictive performance. Below are the RMSE LOOCV values associated with each model, aiding in the selection of the optimal model for our analysis. (For detailed View, Look at Additional Work Section)

Criterion & Search method	**RMSE**	**Variables Included**
Forward AIC model.	1365.982	carat, clarityIF, claritySI1, claritySI2, clarityVS1, clarityVS2, clarityVVS1, clarityVVS2, colorE, colorF, colorG, colorH, colorI, colorJ, x
Forward BIC model	1365.982	carat, clarityIF, claritySI1, claritySI2, clarityVS1, clarityVS2, clarityVVS1, clarityVVS2, colorE, colorF, colorG, colorH, colorI, colorJ, x
Backward AIC model	1365.982	carat, colorE, colorF, colorG, colorH, colorI, colorJ, clarityIF, claritySI1, claritySI2, clarityVS1, clarityVS2, clarityVVS1, clarityVVS2, x
Backward BIC model	1365.982	carat, colorE, colorF, colorG, colorH, colorI, colorJ, clarityIF, claritySI1, claritySI2, clarityVS1, clarityVS2, clarityVVS1, clarityVVS2, x
Stepwise AIC model	1365.982	carat, clarityIF, claritySI1, claritySI2, clarityVS1, clarityVS2, clarityVVS1, clarityVVS2, colorE, colorF, colorG, colorH, colorI, colorJ, x
Stepwise BIC model	1365.982	carat, clarityIF, claritySI1, claritySI2, clarityVS1, clarityVS2, clarityVVS1, clarityVVS2, colorE, colorF, colorG, colorH, colorI, colorJ, x
Best model using AIC	1371.926	carat, colorJ, claritySI1, claritySI2, clarityVS1, clarityVS2, clarityVVS1, clarityVVS2
Best model using BIC	1371.926	carat, colorJ, claritySI1, claritySI2, clarityVS1, clarityVS2, clarityVVS1, clarityVVS2

Criterion & Search method	**RMSE**	**Variables Included**
Best Model using Adjusted R2	1371.926	carat, colorJ, claritySI1, claritySI2, clarityVS1, clarityVS2, clarityVVS1, clarityVVS2

Considering the outcomes above, our preference leans towards the model with the lowest Root Mean Squared Error (RMSE). This choice is motivated by the desire to have a model that exhibits better overall accuracy in predicting price, as indicated by the minimized RMSE.

Refined Final Model

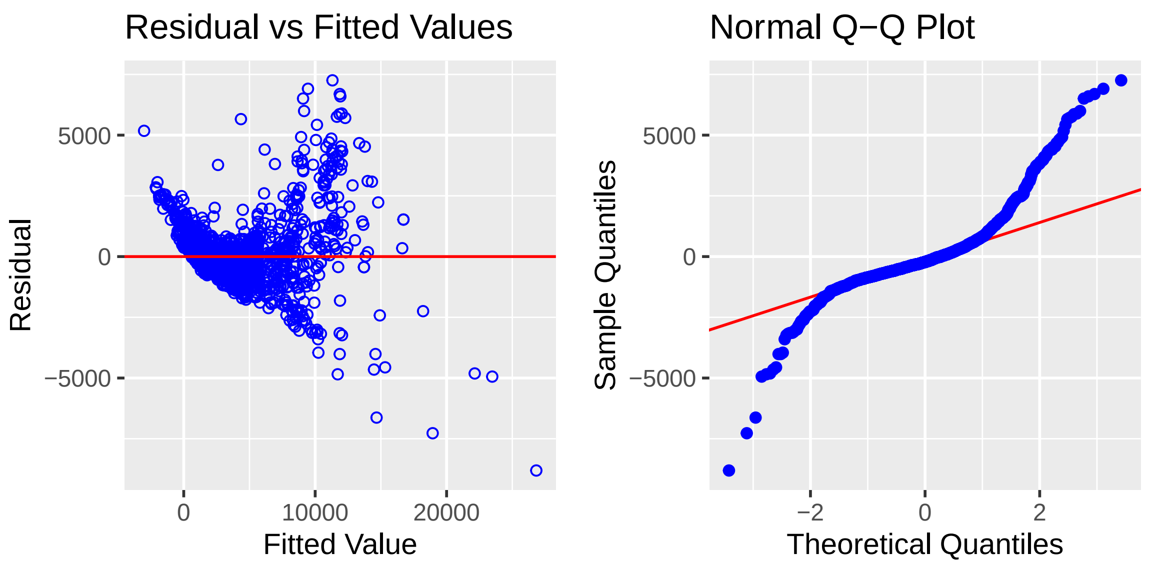
Based on the RMSE value, we have selected the below as our model: (Selected Model): price ~ carat + color + clarity + x

The chosen model includes carat, color, clarity, and x as predictors. This model was selected based on its LOOCV and RMSE Value and it doesn't much differ with the other model as this model only includes a extra predictor "x" which was not included by best subset quality criterion.

Model Diagnostics

When the LINE assumptions (Linearity, Independence, Normality, and Equal Variance) are violated, it can lead to incorrect inferences from statistical tests. Although predictions may still be reliable, the validity of our tests may be compromised. Now, we discuss the methods used to check these assumptions:

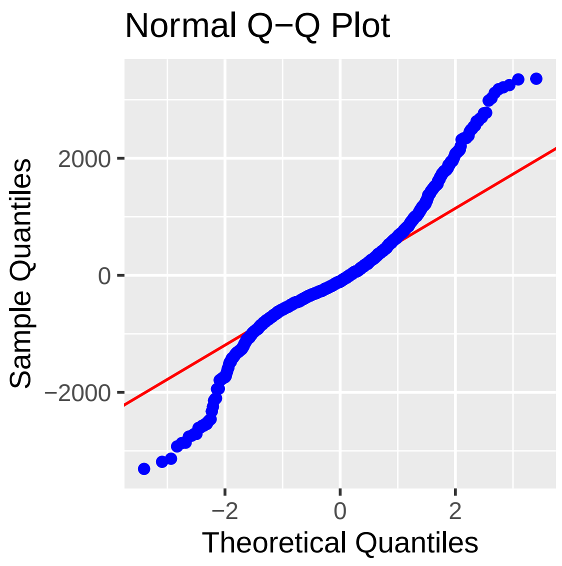
Fitted Vs Residuals and Normality Test: Q-Q Plot of the model:

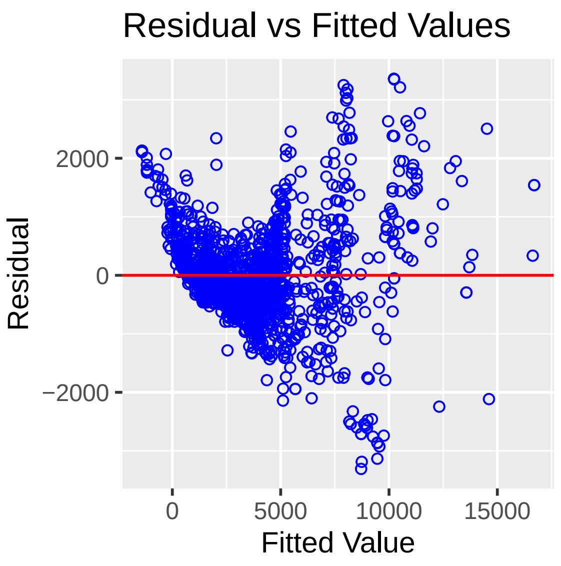


From the above Fitted VS Residual Plot, we can say that, Data Points are not centered at zero and doesn't spread about non-zero. So we can conclude that It violates both Linearity and Constant Variance assumptions.

In this model, Q-Q plot does not look good. It can be observed that at the small and large quantiles, the points do not follow the line closely at all. So, we can conclude that the errors are not normally distributed. It can be caused by High Influential Points and sometimes models can be fixed by removing such points.

Fitted Vs Residuals Plot and Normality Test: Q-Q Plot of the model_fix:

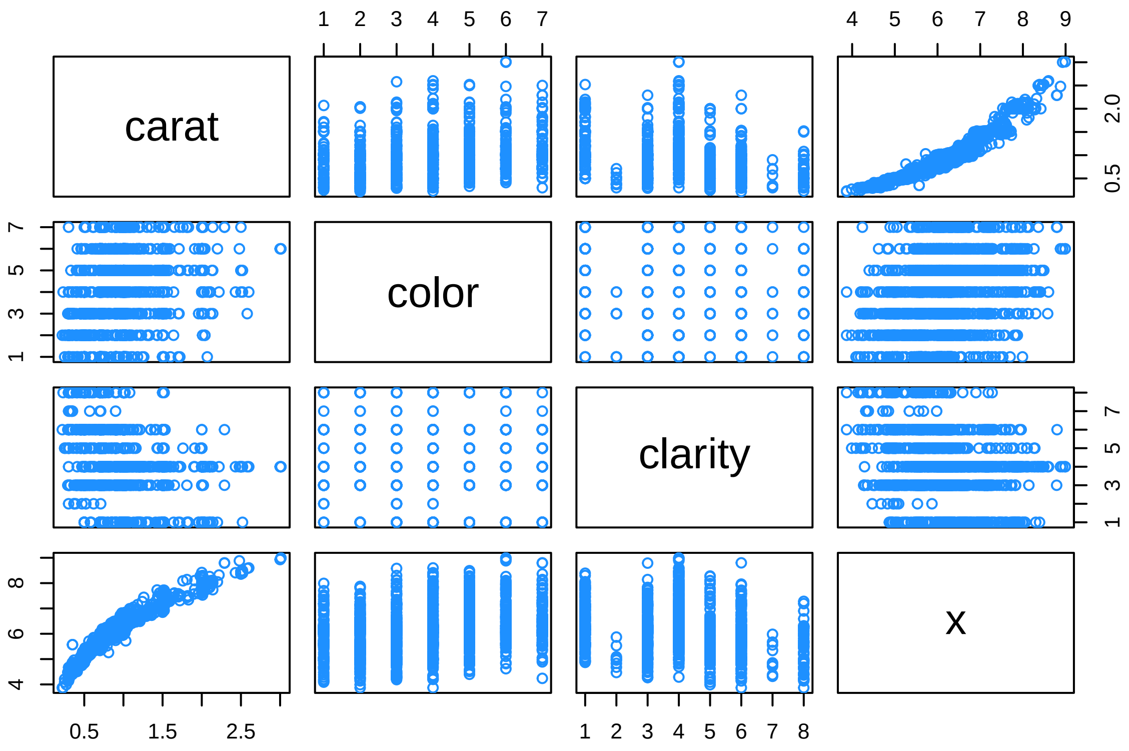




From Both test even after removing High Influential Points, All the assumptions violations still exists. In a nutshell, Removing the influential observations doesn't fix the issues of Normality Assumptions Violation and other violations.

Collinearity

Collinearity decreases the power of the hypothesis test – the probability of correctly detecting a non-zero coefficient. We already removed the few predictors before while selecting the best model for prediction and we're mainly focused on prediction.



After performing ols_eigen_cindex() on our refined_model, observed that 75.28 is the condition number and is the only greater than 30. So, we can conclude that only one linear dependence is likely causing most of the problem. It can also be observed that **carat** and **x** dominated the linear combination.

Solutions to Collinearity based on our Goal We are mainly focused on the prediction of diamonds price and collinearity is not the problem for prediction. So, we can ignore it. But, Explanation is always a good for regression analysis. So, to find the best model we calculated R2 and RMSE Value of models with x and without carat as predictor and vice versa

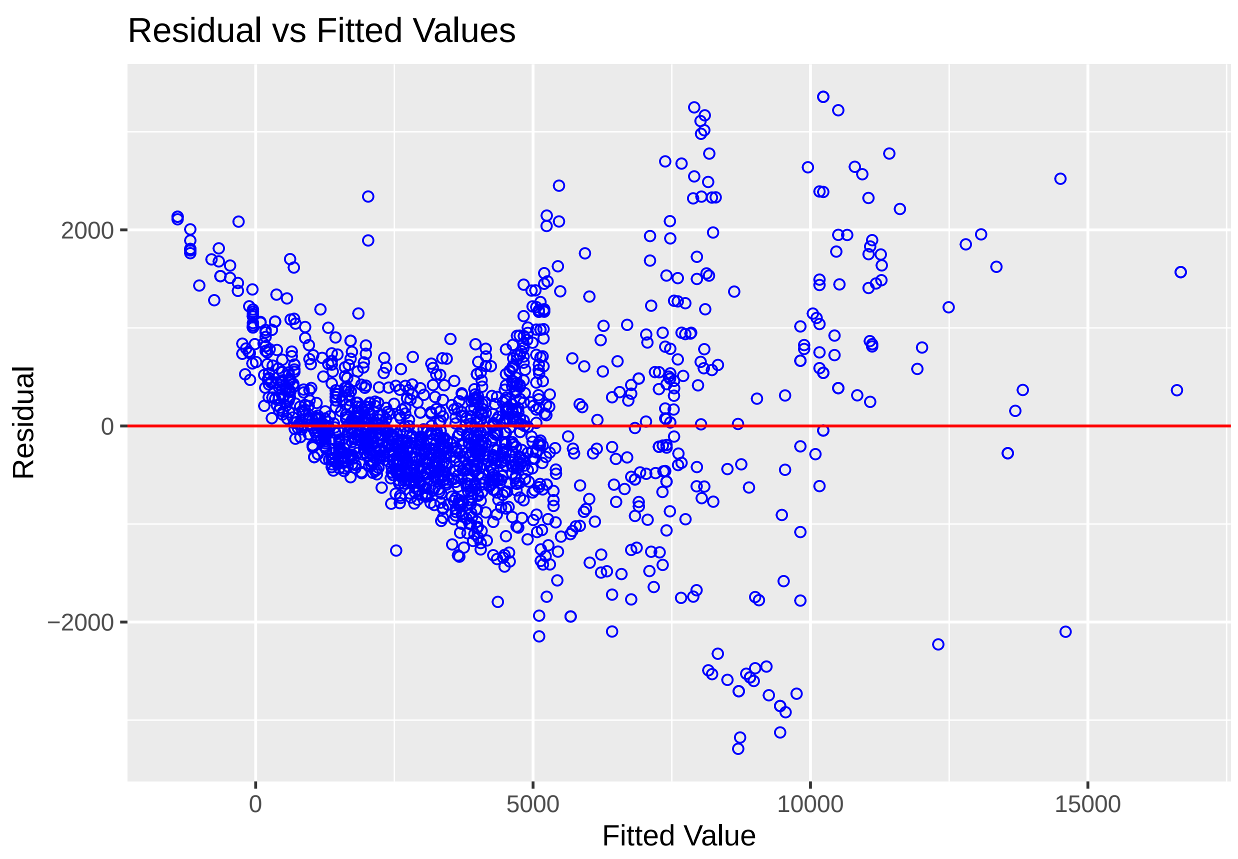
Model	R2	RMSE
model with x and without carat	0.836	1139.241
model with carat and without x	0.916	816.191

To Sum Up, Based on the R2 Value and RMSE, Model without "x" predictor performs better for prediction. Therefore, Model after collinearity now included: carat, color, and clarity as the predictors. Therefore, Our Refined Model now is given by:

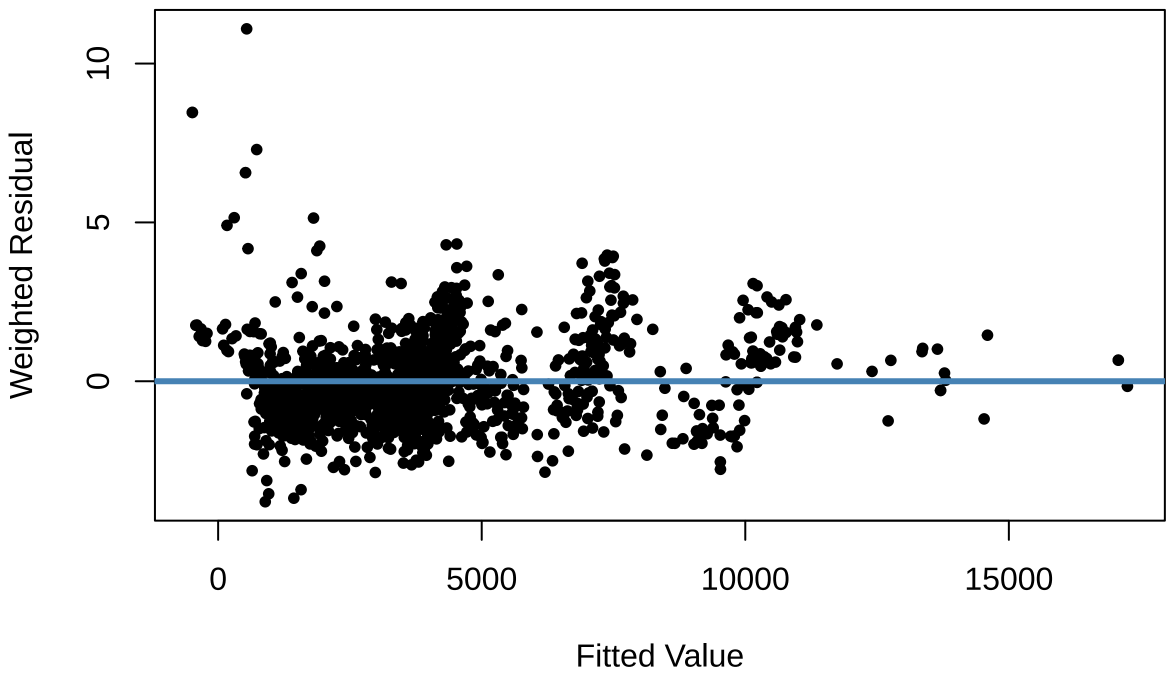
After checking the Fitted Vs Residual Plot and Q-Q Plot, Violation still persists. so we are moving through WLS.

Weighed least Square (WLS)

Now, we'll create a weighted least squares model. We assume errors are independent since data aren't aren't taken over time.



The Value of BP $= 416.94$ and P-value is 2.2e-16



From this plot, we see that the WLS looks much better. The difference in the spread between small and large fitted values is smaller.

Diagnostics for WLS Model

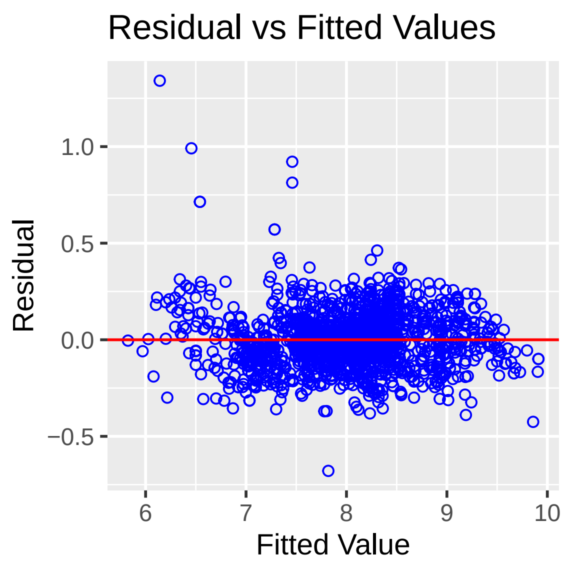
Result of the BP Test shows that the p-value of the WLS Model is 1 which means that the constant variance assumption is not violated. Getting the p-value =1 is rare, so we decided to visually check the Constant Variance assumptions whether it is violated or not and on checking, we can easily say that, the constant variance assumptions is violated. So, we can say that if there are categorical predictors are in the model, it is better to check constant variance assumptions by Fitted Vs Residual Plot. Similarly, For Normality test, it is always better to check by Q-Q Plot if the model have 1 or more categorical predictors. To Sum Up, There's still Linearity, Normality and Constant Variance Violations. So, We may need to use transformation to fix the issues.

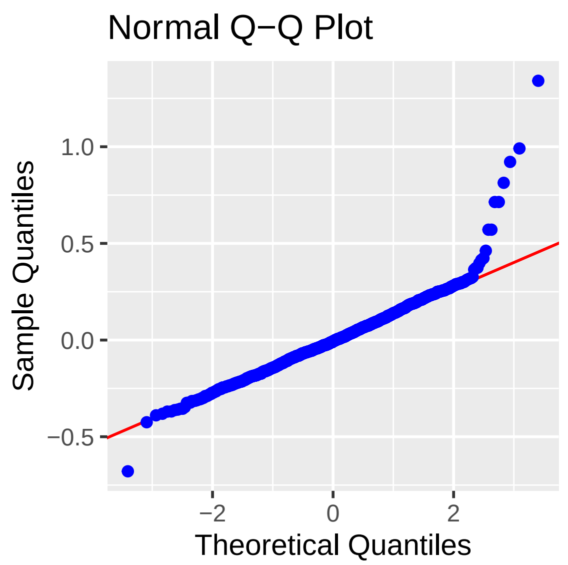
Transformation

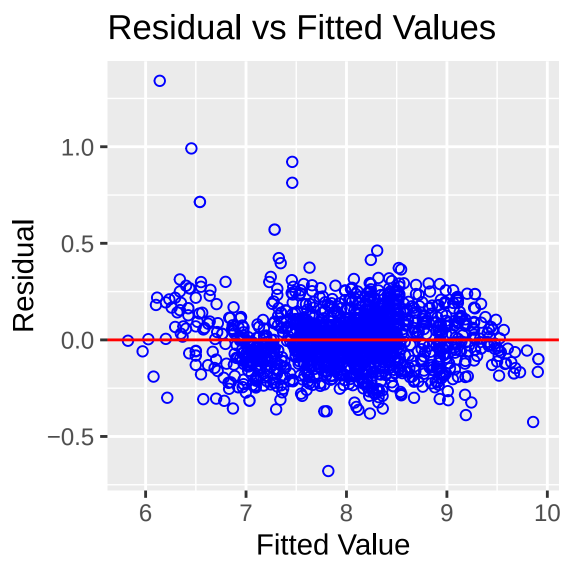
As seen before, we have a model with the data where errors violates constant variance and normality assumptions. we then used The Box-Cox Transformation and the violation still persists. so now we are going for Log Transformation.

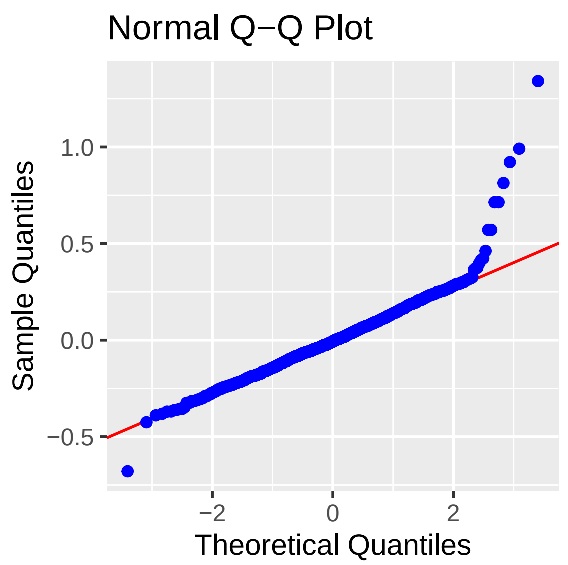
Predictor Transformations

Log Transformation of predictor

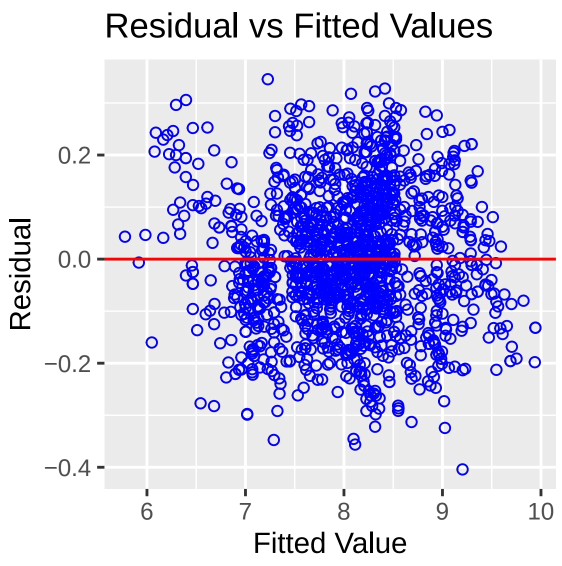


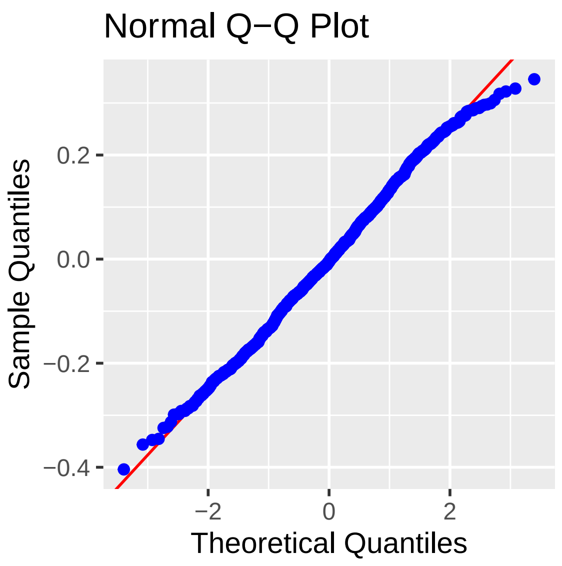






From above plots, we can sat that errors nearly follows constant variation assumptions. The deviation can be seen in the Q-Q Plot, at the end of the points i.e Normality violations and High Influential points may cause such type of violations. So, Checking for it.





From the above Fitted Vs Residual Plot and Q-Q Plot, we can say that Line Assumptions are not violated. We are here getting little less p-value than required to accept the null hypothesis for Shapiro-Wilk test and BP test, because these tests aren't ideal to check Normality and Equal Variance violation when there is any Categorical predictor present in the model.

Final Model

```
##
## Call:
## lm(formula = log(price) ~ log(carat) + color + clarity, data = diamonds)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.67878 -0.09249 -0.00661  0.08877  1.34114
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   7.80226    0.01732 450.353  < 2e-16 ***
## log(carat)    1.77796    0.01071 165.968  < 2e-16 ***
## colorE       -0.03558    0.01623  -2.191   0.0286 *
## colorF       -0.07037    0.01534  -4.589 4.83e-06 ***
## colorG       -0.14489    0.01552  -9.336  < 2e-16 ***
## colorH       -0.20546    0.01580 -13.007  < 2e-16 ***
## colorI       -0.29215    0.01763 -16.572  < 2e-16 ***
## colorJ       -0.38444    0.01994 -19.281  < 2e-16 ***
## clarityIF     1.13455    0.05281  21.485  < 2e-16 ***
## claritySI1    0.60998    0.01397  43.662  < 2e-16 ***
## claritySI2    0.44156    0.01355  32.594  < 2e-16 ***
## clarityVS1    0.77703    0.01690  45.986  < 2e-16 ***
## clarityVS2    0.74989    0.01530  49.028  < 2e-16 ***
## clarityVVS1   1.01861    0.05023  20.278  < 2e-16 ***
## clarityVVS2   0.92570    0.02282  40.563  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 0.1521 on 1495 degrees of freedom
## Multiple R-squared:  0.9543, Adjusted R-squared:  0.9539
## F-statistic:  2230 on 14 and 1495 DF,  p-value: < 2.2e-16
```

The LOOCV-RMSE Value of this model is 0.1344318.

Therefore, There is no collinearity issues. Note, We already addressed Collinearity issues before.

Section 3: Discussion

Model Utility : This model can serve as a valuable tool for pricing diamonds in real-time market conditions, allowing sellers to set prices competitively and buyers to evaluate the fairness of prices.

Model Interpretation : The coefficients of carat, clarity, and color were found to have significant impacts on price, with larger carat sizes and higher clarity and color ratings corresponding to higher prices. These findings underscore the critical factors that potential buyers and sellers should consider.

Other Predictors (x,y,z, depth, and table) seems to not have any significant relationship with the diamonds price while carat, color, and clarity are in the model.

Also, the final model is composed of Log transformation. So, We can't predict the price of the diamond directly without log calculation, but it can be done with very ease with R, so it is not a big deal if we are using R.

Practical Conclusions : The model's insights can help inform business strategies in the diamond industry, particularly in pricing and inventory management (Investment Required), ensuring that stakeholders can make data-driven decisions.

Section 4: Limitations

Data Reliability and Validity: While the dataset is comprehensive, the lack of detailed provenance might affect the generalizability of the findings. The data's scope might not capture all nuances of the global diamond market.

Model Limitations : Some deviation at the end of the data points on the Q-Q Plots were seen, even after transformations (Used Box-Cox Transformation & Polynomial Transformation) and adjustments. If given more time, further refinement of the model or alternative modeling approaches could be explored.

Future Work: Future research could integrate additional variables such as market trends or economic factors, and apply machine learning techniques to enhance predictive performance.

Section 5: Conclusions

The analysis successfully developed a predictive model that explains a significant portion of the variance in diamond prices based on key attributes. The model performs well in terms of predictive accuracy and provides actionable insights into important pricing factors. This project highlights the potential of statistical modeling in understanding and predicting diamond prices, offering valuable insights that can be leveraged by stakeholders across the diamond industry.

We got the R2 value of 0.9531 i.e. (95.31%) which means that 96.31% variablity in the diamonds price is explained by the final model predictors i.e. (carat+color+clarity).

Section 6: *Additional Work*

Additional Models: Other models, including non-linear and interaction effect models, were also considered and are detailed in the supplementary materials. These models were tested for their ability to capture more complex patterns in the data. Please refer to the RMD file for additional information in the Additional Work section.

Supplementary Analysis: Additional analyses such as geographic or temporal effects on diamond prices could further enrich the understanding of the market dynamics and are recommended for further study. Please refer to the RMD file for additional information in the Additional Work section.

Section 7: Code Appendix

```r
# Loading our data set
diamonds = read.csv("final_diamonds.csv")

#printing the top 10 records
#head(diamonds, 10)
# Set up the plotting area to display two plots side by side
par(mfrow=c(2, 2), mar=c(4, 4, 2, 1))

# Plot histogram of diamond prices
hist(diamonds$price,
     xlab='Diamonds Price',
     ylab='Count',
     main='Histogram of Diamond Prices',
     breaks=45,
     col='gray',
     cex.main=0.8,
     xaxt='n')
# Customize x-axis labels for the histogram
axis(side=1, at=seq(0, max(diamonds$price) + 1000, by=1000),
     labels=paste0(seq(0, max(diamonds$price) + 1000, by=1000) / 1000, "K"))

# Converting Categorical Values as factor for R
diamonds$clarity = as.factor(diamonds$clarity)
diamonds$color = as.factor(diamonds$color)

# Create a horizontal box plot of diamond prices
boxplot(diamonds$price,
        col='grey',
        main='Box Plot for Diamond Prices',
        horizontal=TRUE,
        cex.main=0.75,
        xaxt='n')
# Customize x-axis labels for the box plot
axis(side=1, at=seq(0, max(diamonds$price) + 1000, by=1000),
     labels=paste0(seq(0, max(diamonds$price) + 1000, by=1000) / 1000, "K"))

# Reset plotting parameters
par(mfrow=c(1, 1), mar=c(5, 4, 4, 2) + 0.1)
```

```r
# Reducing the size of the plot
par(mfrow = c(2, 2))
# Plotting relationship between each predictor and the response variable 'diamonds$price', including re

# Plotting 'carat' vs. 'price' with regression line
plot(diamonds$carat, diamonds$price, xlab = 'Carat', ylab = 'price', main = 'Comparison of Carat vs. pr
abline(lm(diamonds$price ~ diamonds$carat), col = "blue")

# Plotting 'depth' vs. 'price' with regression line
plot(diamonds$depth, diamonds$price, xlab = 'Depth', ylab = 'price', main = 'Comparison of Depth vs. pr
abline(lm(diamonds$price ~ diamonds$depth), col = "blue")

# Plotting 'table' vs. 'price' with regression line
plot(diamonds$table, diamonds$price, xlab = 'Table', ylab = 'price', main = 'Comparison of Table vs. pr
abline(lm(diamonds$price ~ diamonds$table), col = "blue")

# Plotting 'x' vs. 'price' with regression line
plot(diamonds$x, diamonds$price, xlab = 'X', ylab = 'price', main = 'Comparison of X vs. price', col = 
abline(lm(diamonds$price ~ diamonds$x), col = "blue")

# Plotting 'y' vs. 'price' with regression line
plot(diamonds$y, diamonds$price, xlab = 'Y', ylab = 'price', main = 'Comparison of Y vs. price', col = 
abline(lm(diamonds$price ~ diamonds$y), col = "blue")

# Plotting 'z' vs. 'price' with regression line
plot(diamonds$z, diamonds$price, xlab = 'Z', ylab = 'price', main = 'Comparison of Z vs. price', col = 
abline(lm(diamonds$price ~ diamonds$z), col = "blue")
suppressPackageStartupMessages(library(ggplot2))
library(gridExtra)

# If there's a custom dataset you're using, make sure it's loaded correctly
# If you're using ggplot2's built-in diamonds dataset, the next line is not necessary
# data("diamonds", package = "ggplot2")

# Boxplot of price by color
p1 <- ggplot(diamonds, aes(x = color, y = price, fill = color)) +
  geom_boxplot() +
  xlab("Color") +
  ylab("Price") +
  ggtitle("Boxplot of Price by Color") +
  theme_minimal()

# Boxplot of price by clarity
p2 <- ggplot(diamonds, aes(x = clarity, y = price, fill = clarity)) +
  geom_boxplot() +
  xlab("Clarity") +
  ylab("Price") +
  ggtitle("Boxplot of Price by Clarity") +
  theme_minimal()

# Arrange plots side by side
grid.arrange(p1, p2, ncol = 2)
```

```r
#is.factor(diamonds$color)
#is.factor(diamonds$clarity)

# Fit a multiple linear regression model
model = lm(price ~ ., data = diamonds)

# Print the summary of the model
#summary(model)
# Creating a model
model=lm(price~ carat+color+clarity+x, data=diamonds)
suppressPackageStartupMessages(library(olsrr))

#Ploting the residual vs fitted graph
ols_plot_resid_fit(model)

#Plotting the Q-Q plot
ols_plot_resid_qq(model)

# Influential Observations
n = length(resid(model))

#Getting the influential points
influential_points=which(cooks.distance(model) > 4/n)

# Printing out the list of influential points
#influential_points
# Normality, Linearity and Constant Variances Test after Removing High Influential Points
# We need to remove these influential points to see if the assumptions violations might have been cause
noninfluential_ids = which(
    cooks.distance(model) <= 4 / length(cooks.distance(model)))

# fit the model on non-influential subset
model_fix = lm(price ~ carat+color+clarity+x ,
               data = diamonds,
               subset = noninfluential_ids)
refined_model=model_fix
#refined_model
library(olsrr)

# Check if model_fix is correctly defined and loaded
# Assuming model_fix is a linear model object

# Set up the plotting area to display two plots side by side
par(mfrow=c(2, 2), mar=c(4, 4, 2, 1))

# Normality Test: Q-Q Plot
ols_plot_resid_qq(model_fix)

# Fitted Vs Residual Plot
ols_plot_resid_fit(model_fix)

# Reset to default plotting parameters
par(mfrow=c(1, 1), mar=c(5, 4, 4, 2) + 0.1)
```

```r
#Before Moving on, now we are deleting high influential points as they may create problem in the future

refined_data = diamonds[noninfluential_ids, ]

#refined_data or we are always using diamonds as the data name so we are going to rename it as diamonds

diamonds = refined_data
#diamonds

suppressPackageStartupMessages(library(dplyr))

# Create a data frame with the response `price` and other specified columns removed
#price_pred = select(diamonds, -price, -y, -z, -depth, -table)
price_pred = dplyr::select(diamonds, -price, -y, -z, -depth, -table)

# Create a pairwise scatter plot of the columns in price_pred
pairs(price_pred, col = 'dodgerblue')
suppressPackageStartupMessages(library(faraway))

#Getting VIF value
#vif(refined_model)
library(olsrr)

#round(ols_eigen_cindex(refined_model)[, 1:2], 4)

# Getting the conditional number
#ols_eigen_cindex(refined_model)

# Creating the refined model
refined_model=lm(price~carat+color+clarity, data=diamonds)

# Creating the Model OLS
model_ols = lm(price ~ . -x-y-z-table-depth, data = diamonds)

# Plotting the REsidual vs Fitted Values
ols_plot_resid_fit(model_ols)
suppressPackageStartupMessages(library(lmtest))

# Performing the BP test
#bptest(model_ols)

# NOTE: Remember to remove the response!
model_wts = lm(abs(resid(model_ols)) ~ carat+color+clarity, data = diamonds)

#coef(model_wts)
weights = 1 / fitted(model_wts)^2

# run WLS
model_wls = lm(price ~ ., data = diamonds, weights = weights)

# Plotting the value
plot(fitted(model_wls), weighted.residuals(model_wls),
     pch = 20, xlab = 'Fitted Value', ylab = 'Weighted Residual')

abline(h=0, lwd=3, col='steelblue')

library(lmtest)
# Fit a linear regression model with logged response and predictor variables
model_pt = lm(log(price) ~ log(carat) + color + clarity, data = diamonds)

# Print the summary of the model
#summary(model_pt)

# Shapiro-Wilk test for normality of residuals
# Note: This might not work directly with models including categorical variables
#shapiro.test(resid(model_pt))
#so we are ignoring it


# Breusch-Pagan test for heteroscedasticity (if applicable)
# Note: This might not work directly with models including categorical variables

#bptest(model_pt)
#so we are ignoring it

ols_plot_resid_fit(model_pt)

ols_plot_resid_qq(model_pt)
# Adjust these values as needed
library(olsrr)
library(lmtest)

# Assuming model_pt is already defined and is a valid lm object
# If not defined in this chunk, make sure it's defined in your document before this chunk

# Set up the plotting area to display two plots side by side with smaller margins
par(mfrow=c(2, 2), mar=c(4, 4, 2, 1))



# Plot the Residuals vs Fitted plot
ols_plot_resid_fit(model_pt)

# Plot the Q-Q plot of residuals
ols_plot_resid_qq(model_pt)

# Reset to default plotting parameters after plotting
par(mfrow=c(1, 1), mar=c(5, 4, 4, 2) + 0.1)


# Getting High Influential points
high_inf_ids = which(cooks.distance(model_pt) > 4 /length(resid(model_pt)))
#high_inf_ids
library(lmtest)
```

```r
non_inf_ids = which(cooks.distance(model_pt) <= 4/length(resid(model_pt)))

#Now, Creating a model model_log which, is the model with all the high influential points removed, plot
model_log = lm(log(price) ~ log(carat) + color + clarity, data = diamonds, subset = non_inf_ids)

#summary(model_log)
#bptest(model_log)
#shapiro.test(resid(model_log))


ols_plot_resid_fit(model_log)
ols_plot_resid_qq(model_log)

#nobs(model_log)
model_log = lm(log(price) ~ log(carat) + color + clarity, data = diamonds)
summary(model_log)
library(faraway)

#vif(model_log)

summarized_diamonds = aggregate(price ~ color, data = diamonds, sum)

percentage = (summarized_diamonds$price / sum(summarized_diamonds$price)) * 100
par(mar = c(0.25, 0.25, 0.25, 0.25))  # Adjust the margins to give more space for the chart

# Plotting
pie(
  summarized_diamonds$price,
  labels = paste0(summarized_diamonds$color, "\n", round(percentage, 1), "%"),
  col = rainbow(length(summarized_diamonds$color)),
  cex = 1 # Adjust the size of the text labels
)

# Giving the legends of the plotted graph
legend("topright", legend = summarized_diamonds$color, fill = rainbow(length(summarized_diamonds$color)


summarized_diamonds = aggregate(price ~ clarity, data = diamonds, sum)

percentage = (summarized_diamonds$price / sum(summarized_diamonds$price)) * 100

# Increase the size of the pie chart
par(mar = c(0.25, 0.25, 0.25, 0.25))  # Adjust the margins to give more space for the chart

# Plotting graph
pie(
  summarized_diamonds$price,
  labels = paste0(summarized_diamonds$clarity, "\n", round(percentage, 2), "%"),
  col = rainbow(length(summarized_diamonds$clarity)),
  cex = 0.7 # Adjust the size of the text labels
)

# Legends of the plotting graph
```

```
legend("topright", legend = summarized_diamonds$clarity, fill = rainbow(length(summarized_diamonds$clar



## Model Selection
calc_loocv_rmse = function(model) {
  sqrt(mean((resid(model) / (1 - hatvalues(model))) ^ 2))
}

# AIC Forward Selection
#data(diamonds)

# forward selection with AIC
mod_start = lm(price ~ 1, data = diamonds)
mod_forwd_aic = step(mod_start, scope = price ~ carat + color + clarity + depth + table + x + y + z,
direction = 'forward', trace = 0)
coef(mod_forwd_aic)
calc_loocv_rmse(mod_forwd_aic)

# BIC Forward Selection
n = nrow(diamonds)
mod_forwd_bic = step(mod_start, scope = price ~ carat + color + clarity + depth + table + x + y + z,
direction = 'forward', k = log(n), trace = 0)
coef(mod_forwd_bic)
calc_loocv_rmse(mod_forwd_bic)

# AIC backward selection
mod_start = lm(price ~ ., data = diamonds)
mod_backwd_aic = step(mod_start, direction = 'backward', trace = 0)
coef(mod_backwd_aic)
calc_loocv_rmse(mod_backwd_aic)

# BIC backward selection
mod_backwd_bic = step(mod_start, direction = 'backward', k = log(n), trace = 0)
coef(mod_backwd_bic)
calc_loocv_rmse(mod_backwd_bic)

## AIC stepwise selection
mod_start = lm(price ~ 1, data = diamonds)
mod_step_aic = step(mod_start,
scope = price ~ carat + color + clarity + depth + table + x + y + z,
direction = 'both', trace = 0)
coef(mod_step_aic)
 calc_loocv_rmse(mod_step_aic)

## BIC stepwise selection
mod_step_bic = step(mod_start,
scope = price ~ carat + color + clarity + depth + table + x + y + z,
direction = 'both', k = log(n), trace = 0)
coef(mod_step_bic)
calc_loocv_rmse(mod_step_bic)

library(leaps)
```

```r
### Best Subset Selection
mod_subsets = summary(regsubsets(price ~ ., data = diamonds))
coef_names = colnames(mod_subsets$which)

## coeficients of the best model using adjusted R2
best_r2_ind = which.max(mod_subsets$adjr2)
coef_names[mod_subsets$which[best_r2_ind, ]]
mod_exh_r2=lm(price ~ carat + color + clarity, data=diamonds)
calc_loocv_rmse(mod_exh_r2)
library(leaps)  # Ensuring the 'leaps' library is loaded

# Assuming 'mod_subsets' is a result from 'regsubsets'
p = ncol(mod_subsets$which)  # Total number of predictors considered

# Correct AIC calculation:
# 'n' should be the number of observations used in the fitting process
n = nrow(diamonds)  # Replace 'diamonds' with your dataset used in regsubsets
mod_aic = n * log(mod_subsets$rss / n) + 2 * seq_along(mod_subsets$rss)  # Applying AIC formula correct

best_aic_ind = which.min(mod_aic)  # Finding the model with the minimum AIC
coef_names_at_best_aic = mod_subsets$which[best_aic_ind, ]  # Extracting the best model coefficients

# Optionally, print or return the best coefficients
print(coef_names_at_best_aic)

mod_exh_aic=lm(price ~ carat + color + clarity, data=diamonds)
calc_loocv_rmse(mod_exh_aic)

## best model using BIC
suppressWarnings({mod_bic = n * log(mod_subsets$rss / n) + log(n) * (2:p)
best_bic_ind = which.min(mod_bic)})
coef_names[mod_subsets$which[best_bic_ind, ]]

mod_exh_bic=lm(price ~ carat + color + clarity, data=diamonds)
calc_loocv_rmse(mod_exh_bic)

library(olsrr)
ols_plot_added_variable(model)

library(lmtest)

# run the breush-pagan test
bptest(model)
# Normality: Visual Test of Normality Assumptions

# Histogram
plot_resid_histogram = function(model, title = 'Histogram of Residuals',
lwd = 3, breaks = 20) {
# extract residuals
res = resid(model)
# histogram of residuals
h = hist(res, xlab = "Residuals", main = title, col = "#ADD8E6",
border = "black", breaks = breaks)
# fitted normal density
xgrid = seq(min(res) - 1, max(res) + 1, length = 100)
yden = dnorm(xgrid, mean = mean(res), sd = sd(res))
yden = yden * length(res) * diff(h$mids[1:2])
lines(xgrid, yden, col = 'darkorange', lwd = lwd)
}
plot_resid_histogram(model)
# UNUSUAL Observation

#High Leverage Points
leverage_points=which(hatvalues(model) > 2 * mean(hatvalues(model)))
leverage_points
#### Outlier Test
# function that returns the Bonferroni corrected critical value
outlier_test_cutoff = function(model, alpha = 0.05) {
n = length(resid(model))
qt(alpha/(2 * n), df = df.residual(model) - 1, lower.tail = FALSE)
}
# vector of indices for observations deemed outliers.
cutoff = outlier_test_cutoff(model, alpha = 0.05)
outlierss_points=which(abs(rstudent(model)) > cutoff)
outlierss_points

###Collinearity
library(dplyr)

# Create a data frame with the response `price` and other specified columns removed
#price_pred = select(diamonds, -price, -y, -z, -depth, -table)
price_pred = dplyr::select(diamonds, -price, -y, -z, -depth, -table)

# Create a pairwise scatter plot of the columns in price_pred
pairs(price_pred, col = 'dodgerblue')


####Solutions to Collinearity
#R2 value calculation
print('R2 value For Model without carat as predictor:')
#Model without carat as predictor
model_with_carat_removed = lm(price ~ color+clarity+x,
               data = diamonds)
summary(model_with_carat_removed)$adj.r.squared
print('RMSE value For Model without carat as predictor:')
calc_loocv_rmse(model_with_carat_removed)

print('R2 value For Model without x as predictor:')
#Model without x as predictor
model_with_x_removed = lm(price ~ carat+color+clarity,
               data = diamonds)

summary(model_with_x_removed)$adj.r.squared
print('RMSE value For Model without x as predictor:')
calc_loocv_rmse(model_with_x_removed)
```

```r
# VIF
library(faraway)
vif(refined_model)

# Normality: Shapiro Will test
shapiro.test(resid(model_fix))

# Normality: (Calculation) Testing for Normality
shapiro.test(resid(model))

# The Condition Number and the condition indices
library(olsrr)

round(ols_eigen_cindex(refined_model)[, 1:2], 4)

#WLS
# Diagnostics for WLS Model*
library(lmtest)
bptest(model_wls)
shapiro.test(resid(model_wls))

ols_plot_resid_fit(model_wls)
#OLS
library(olsrr)

# fit the model using OLS
summary(model_ols)

# fitted-vs-residuals plot
ols_plot_resid_fit(model_ols)

library(lmtest)
bptest(model_ols)

## Transformation
library(faraway)

#let's take this model decided after removing x because of collinearity
model = lm(price ~ carat+color+clarity, data = diamonds)

library(MASS)
bc = boxcox(model, lambda = seq(-0.25, 0.75, by = 0.05), plotit = TRUE)
bc$x[which.max(bc$y)]
get_lambda_ci = function(bc, level = 0.95) {
    # lambda such that
    # L(lambda) > L(hat(lambda)) - 0.5 chisq_{1, alpha}
    CI_values = bc$x[bc$y > max(bc$y) - qchisq(level, 1)/2]

    # 95 % CI
    CI = range(CI_values)

    # label the columns of the CI
    names(CI) = c("lower bound","upper bound")
    CI
}

# extract the 95% CI from the box cox object
get_lambda_ci(bc)
model_bc = lm(price ^ 0.54 ~ color+clarity+carat, data = diamonds )
summary(model_bc)
library(lmtest)

# Performing Test
shapiro.test(resid(model_bc))
bptest(model_bc)

# Plotting graph
ols_plot_resid_fit(model_bc)
ols_plot_resid_qq(model_fix)
# Polynomial Transformation

# Fit a linear regression model with logged response and predictor variables
model_poly1 = lm(price ~ carat + color + clarity, data = diamonds)

model_poly2 = lm(price ~ carat + I(carat^2) + color + clarity, data = diamonds)

model_poly3 = lm(price ~ carat + I(carat^3) + color + clarity, data = diamonds)

model_poly4 = lm(price ~ carat + I(carat^4) + color + clarity, data = diamonds)

model_poly5 = lm(price ~ carat + I(carat^5) + color + clarity, data = diamonds)

model_poly6 = lm(price ~ carat + I(carat^6) + color + clarity, data = diamonds)

model_poly7 = lm(price ~ carat + I(carat^7) + color + clarity, data = diamonds)

model_polyyy = lm(price ~ poly(carat, 15, raw = TRUE) + color + clarity, data = diamonds)


# Print the summary of the model
summary(model_poly1)
summary(model_poly2)
summary(model_poly3)
summary(model_poly4)
summary(model_poly5)
summary(model_poly6)
summary(model_poly7)
summary(model_polyyy)


ols_plot_resid_fit(model_polyyy)

shapiro.test(resid(model_polyyy))

library(lmtest)
bptest(model_polyyy)
```