# Diamond Price Regression Analysis

## Final Project Report

Samay Jain (s3963844)

2024-06-13

## Introduction

This report focuses on conducting regression analysis on the diamond data. The dataset includes details affecting diamond prices, such as carat weight, cut, color, clarity, and dimensions. The data is divided into training and testing sets. Various methods, including all possible regression subsets and stepwise regression, are employed to identify the optimal regression model for predicting diamond prices.

## Methodology

**Data Cleaning:** - Checked for missing values - Checked for duplicates - Checked for outliers

**Data Transformation:** - Encoded categorical variables: Converted categorical variables into numerical format using label encoding - Transformed skewed data: Apply suitable transformations to reduce skewness in the dependant variable

**Data Exploration:** - Descriptive statistics - Visualization - Correlation Analysis:

**Model Building:** - Split the dataset: Split the dataset into training(0.8) and testing sets(0.2), to evaluate the model's performance. - Built regression models: Built multiple linear regression models and polynomial regression models with different degrees to predict the diamond prices based on other attributes.

**Model Adequacy:** - Evaluated model performance: Assessed the models' performance based on MSE, RMSE, MAE, and R-squared on the testing set. - Diagnostics: Checked for model assumptions such as linearity, homoscedasticity, normality of residuals, and independence of errors using diagnostic plots. - Autocorrelation: Checked for autocorrelation in the residuals using the Durbin-Watson test or autocorrelation plots. - Multicollinearity: Checked for multicollinearity among predictors using variance inflation factor (VIF) or correlation matrix.

**Model Selection:** - Subset selection: Used methods like all possible regression subset, and stepwise regression to identify the subset of predictors that best explain the variation in the response variable (price). - Selected the best model: Selected the final model based on criteria such as adjusted R-squared, PRESS residuals, adn least error values.

**Model Validation:** - Validated the model: Validated the final model using cross-validation techniques to ensure its generalizability.

## Data set

The diamond dataset contains information on various attributes of diamonds, such as carat weight, cut, color, clarity, dimensions, and price. it contains the information of about 54000 diamonds. The dataset includes the following attributes:

- Carat: The weight of the diamond (numeric).
- Cut: The quality of the cut (categorical, Fair, Good, Very Good, Premium, Ideal).
- Color: The diamond color, ranging from J (worst) to D (best).
- Clarity: A measure of how clear the diamond is (categorical, e.g., I1, SI2, SI1, VS2, VS1, VVS2, VVS1, IF).
- Depth: The total depth percentage, calculated as z / mean(x, y) = 2 * z / (x + y) (numeric).
- Table: The width of the top of the diamond relative to the widest point (numeric).
- Price: The price of the diamond (numeric).
- X, Y, Z: The length, width, and depth of the diamond, respectively (numeric).

```
Diamond <- read_csv("diamonds.csv")
head(Diamond)
```

```
## # A tibble: 6 × 11
##    ...1 carat cut       color clarity depth table price     x     y     z
##   <dbl> <dbl> <chr>     <chr> <chr>   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 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  4.2   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
```

## Data preprocessing

### Removing redundant columns and Renaming columns

```
# Removing redundant columns
Diamond <- Diamond[, -c(1)]
# Renaming column names
colnames(Diamond) <- c("Carat", "Cut", "Color", "Clarity", "Depth",
"Top_width_ratio", "Price_USD", "Length", "Width", "Height")
```

### Descriptive Statistics

```
# Descriptive statistics of data
summary(Diamond)
```

(Shown in the appendix). This table presents key statistics for various attributes of diamonds in the dataset. Carat weight ranges from 0.20 to 5.01, with a mean of around 0.80. Diamond dimensions, including length, width, and height, vary widely, reflecting diverse shapes and sizes. Diamond price ranges from $326 to $18,823, with a mean of roughly $3,933.

## Inconsistencies in data

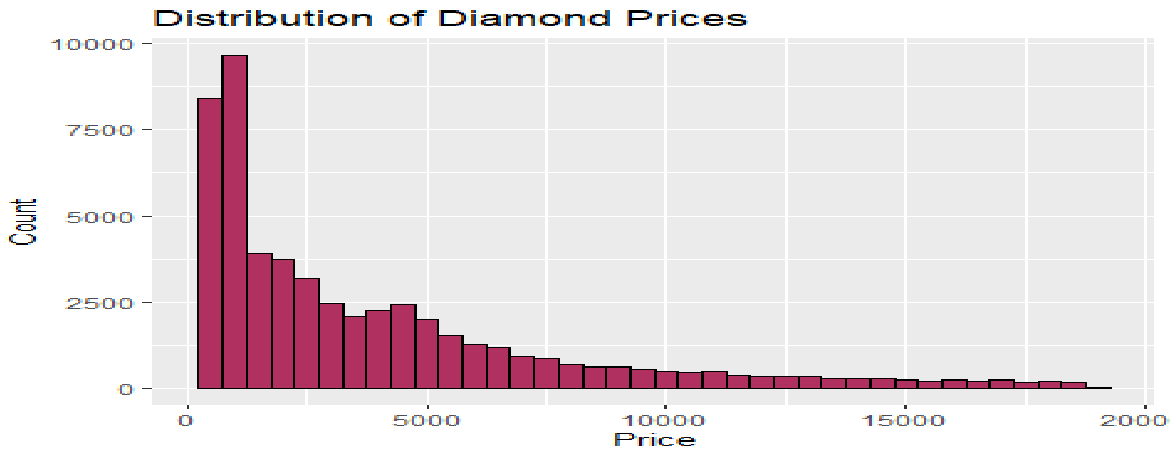
```r
# Missing values in the data
sum(is.na(Diamond))

## [1] 0

# Duplicated values
sum(duplicated(Diamond))

## [1] 146

# Removing the duplicate values
Diamond <- Diamond[!duplicated(Diamond), ]

# Removing any invalid values
if (any(Diamond$Length == 0 | Diamond$Width == 0 | Diamond$Height == 0)) {
  Diamond <- Diamond[!(Diamond$Length == 0 | Diamond$Width == 0 |
Diamond$Height == 0), ]
}

# Dropping Length, Width, and Height columns
Diamond <- Diamond[, -c(8:10)]

# Data set dimension
dim(Diamond)

## [1] 53775     7
```

Missing and duplicate values are eliminated from the dataset to ensure accuracy and precise analysis. Additionally, since depth is derived from the dimensions of length, width, and height, these columns, along with any invalid values within them, will be removed.

## Exploratory Data Analysis

### Diamond price distribution

```r
ggplot(Diamond, aes(x = Price_USD)) +
  geom_histogram(binwidth = 500, fill = "maroon", color = "black") +
  labs(title = "Distribution of Diamond Prices", x = "Price", y = "Count")
```



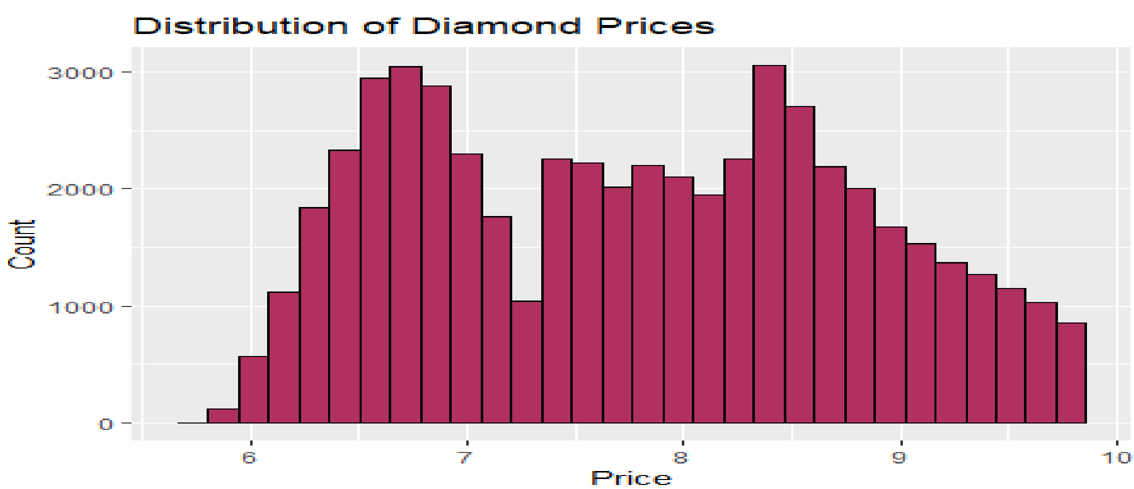

The diamond price is right-skewed (positively skewed), indicating that most diamonds are on the lower end of the price scale and it indicates that transformation is required on the Price variable.

## Transformation

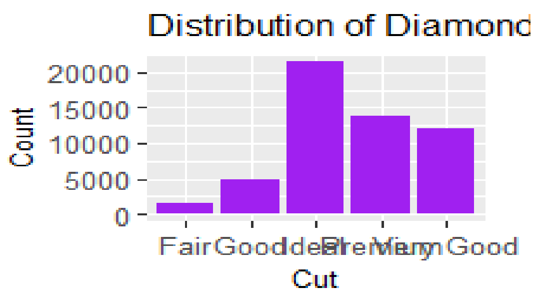
```
# Optimal lambda
lambda <- boxcox_result$x[which.max(boxcox_result$y)]
lambda

## [1] -0.06060606
```

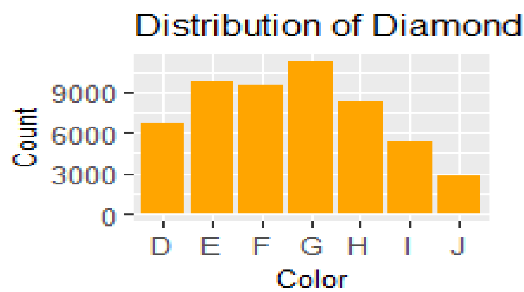
The BoxCox transformation lambda value is close to 0, so log transformation is applied on the price variable.

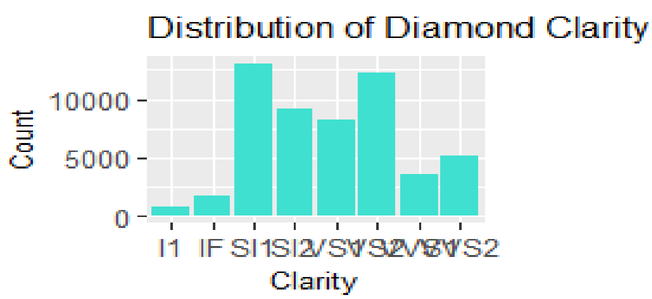
```
Diamond$Log_price <- log(Diamond$Price_USD)

ggplot(Diamond, aes(x = Log_price)) +
  geom_histogram( fill = "maroon", color = "black") +
  labs(title = "Distribution of Diamond Prices", x = "Price", y = "Count")
```



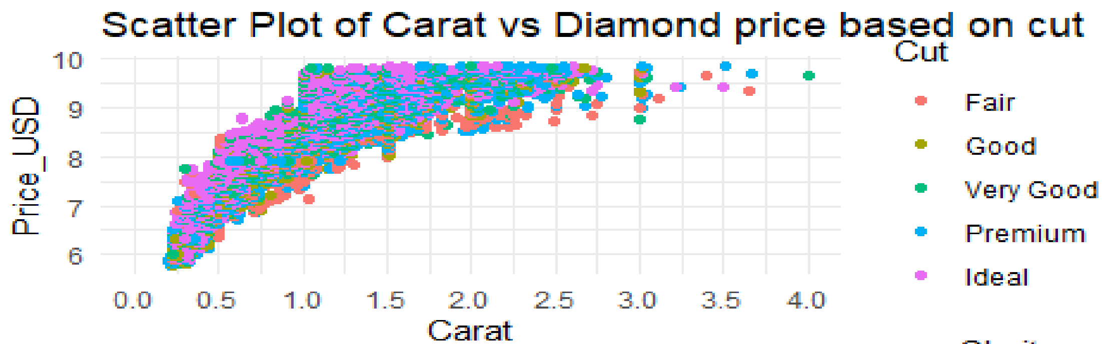
The log transformed price has less skewness making the distribution more normalized. While there is still a higher concentration in lower range prices, the transformation has made the spread more uniform which brings out the subtle differences within the lower price ranges.

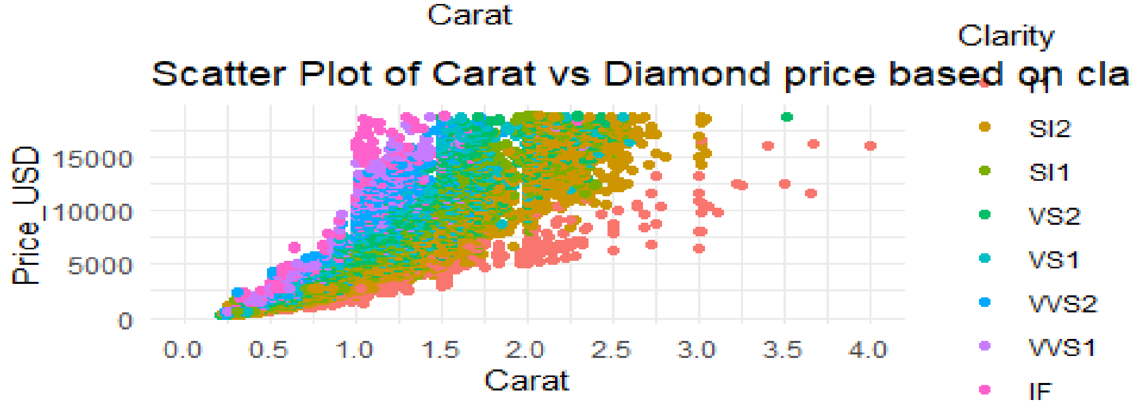



(Code in the appendix) - In the distribution of diamond cuts, the ideal cut dominates the market, followed by premium cuts very good cuts. Good cuts are comparatively fewer while fair cuts make up the smallest portion. - The 2nd bar chart represents the distribution of diamond colour. It reveals that G is the most common color grade for diamonds, followed closely by F. Grades H and I are also frequent. D and J are the least common among all the colors. - The 3rd bar chart displays the distribution of diamond clarity. The chart reveals that diamonds with SI2 (Slightly Included 2) and SI1 (Slightly Included 1) clarity grades are the most common, followed by VS1 (Very Slightly Included 1) and VS2 (Very Slightly Included 2).




The first scatter plot illustrates the relationship between carat and diamond price based on cut. The plot shows a positive correlation between carat weight and price, meaning that larger diamonds generally command higher prices. the scatter plot also shows that the Price of diamond is positively influenced by the quality of cut (the better the cut the more the price).

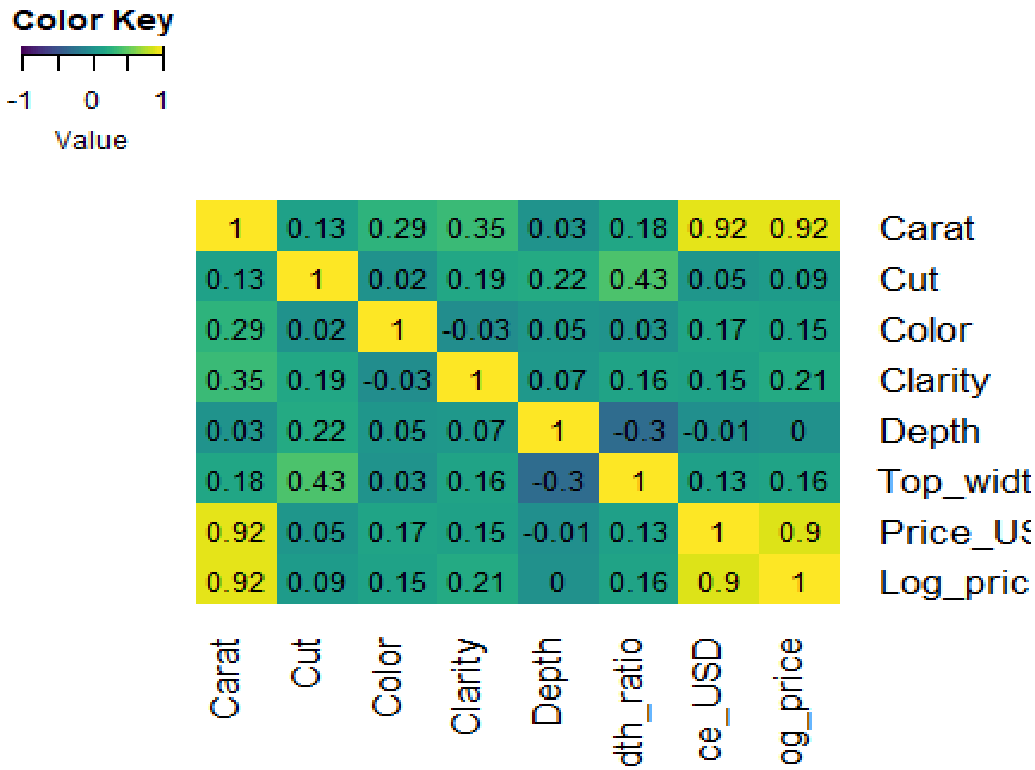
The second scatter plot depicts the relationship between carat weight and diamond price, by the clarity grade of the diamond. Similar to the previous plot, there is a positive correlation between carat weight and price, but with significant variation in prices for diamonds with similar carat weights. It can be seen from the graph that higher clarity grades have the higher price range for a given carat weight, while lower clarity grades tend to have lower prices.

```
# Encoding for Cut, Color, and Clarity
Diamond <- Diamond %>%
  mutate(
    Cut = as.integer(factor(Cut, levels = c("Ideal", "Premium", "Very Good",
"Good", "Fair"))),
    Color = as.integer(factor(Color, levels = c("D", "E", "F", "G", "H", "I",
"J"))),
    Clarity = as.integer(factor(Clarity, levels = c("IF", "VVS1", "VVS2",
"VS1", "VS2", "SI1", "SI2", "I1")))
  )
```

## Correlation plot

```
cor_matrix <- cor(Diamond)
heatmap.2(cor_matrix, trace = "none", col = viridis(256), cellnote =
```

```r
round(cor_matrix, 2), notecol = "black", density.info = "none", dendrogram =
"none", Rowv = FALSE, Colv = FALSE)
```




This correlation matrix reveals the relationships between various attributes of diamonds. - Carat exhibits a strong positive correlation with price, indicating that heavier diamonds tend to be more expensive. - Cut quality demonstrates a moderate positive correlation with price, suggesting that diamonds with superior cuts may have higher prices. - Color and clarity show weak positive correlations with price, implying that higher grades in these attributes might lead to slightly more prices. - Depth and top width ratio exhibit weak correlations, indicating a subtle relationship between these characteristics

## Train and test split data

```r
set.seed(100)
TrainIndex <- createDataPartition(Diamond$Price_USD, p = 0.8, list = FALSE,
times = 1)
Diamond_Train <- Diamond[TrainIndex, ]
Diamond_Test <-  Diamond[-TrainIndex, ]
```

To conduct model validation, the dataset will be split into 80% for the training set and 20% for the test set, based on the price.

## Multiple linear regression

```
Diamond_model <- lm(Diamond_Train$Log_price ~ Carat + factor(Cut) +
factor(Color) + factor(Clarity) + Depth + Top_width_ratio , data =
Diamond_Train)
```

The initial linear model was fitted using all the predictor variables. Upon testing (as detailed in the appendix), it was found that the following assumptions were violated:

- The relationship between variables is not linear.
- The residuals are not normally distributed.
- Homoscedasticity is not satisfied.

Therefore, polynomial regression model is used to address the above mentioned issues.

```
# Fitting a polynomial regression
poly_model <- lm(Log_price ~ Carat + I(Carat^2) + factor(Cut) + factor(Color)
+ factor(Clarity) + Depth + Top_width_ratio, data = Diamond_Train)
```

"Carat" has the highest correlation with the diamond price, its squared term is included to better capture the potential non-linear relationship between carat size and price.

The polynomial fitted model violates the assumption of linearity, normality, and constant variance(shown in the appendix).

```
# Removing influential values
Diamond_Train_pre <- Diamond_Train[-c(21882, 22048, 21642), ]
```

The following influential values were identified in the polynomial model and were removed to normalise the data to reduce bias in the analysis.

```
# Fitting a polynomial regression on processed data
poly_model2 <- lm(Log_price ~ Carat + I(Carat^2) + factor(Cut) +
factor(Color) + factor(Clarity) + Depth + Top_width_ratio, data =
Diamond_Train_pre)
summary(poly_model2)

##
## Call:
## lm(formula = Log_price ~ Carat + I(Carat^2) + factor(Cut) + factor(Color)
+
##     factor(Clarity) + Depth + Top_width_ratio, data = Diamond_Train_pre)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.7838 -0.1012  0.0085  0.1009  4.0490
##
## Coefficients:
##                     Estimate Std. Error  t value Pr(>|t|)
## (Intercept)        5.9641738  0.0529502  112.637  < 2e-16 ***
## Carat              4.3358824  0.0056289  770.284  < 2e-16 ***
## I(Carat^2)        -1.0014849  0.0024971 -401.061  < 2e-16 ***
```

```
## factor(Cut)2     -0.0339889  0.0022306  -15.237  < 2e-16 ***
## factor(Cut)3     -0.0539062  0.0021759  -24.774  < 2e-16 ***
## factor(Cut)4     -0.0774957  0.0030843  -25.126  < 2e-16 ***
## factor(Cut)5     -0.1380661  0.0051239  -26.945  < 2e-16 ***
## factor(Color)2   -0.0573578  0.0027426  -20.914  < 2e-16 ***
## factor(Color)3   -0.0966210  0.0027731  -34.842  < 2e-16 ***
## factor(Color)4   -0.1664669  0.0027163  -61.284  < 2e-16 ***
## factor(Color)5   -0.2711803  0.0028896  -93.848  < 2e-16 ***
## factor(Color)6   -0.3862120  0.0032440 -119.056  < 2e-16 ***
## factor(Color)7   -0.5192328  0.0039925 -130.050  < 2e-16 ***
## factor(Clarity)2 -0.0947032  0.0050059  -18.918  < 2e-16 ***
## factor(Clarity)3 -0.1583678  0.0047863  -33.088  < 2e-16 ***
## factor(Clarity)4 -0.2672003  0.0045724  -58.438  < 2e-16 ***
## factor(Clarity)5 -0.3354021  0.0044625  -75.159  < 2e-16 ***
## factor(Clarity)6 -0.4698115  0.0044942 -104.537  < 2e-16 ***
## factor(Clarity)7 -0.6326340  0.0046732 -135.376  < 2e-16 ***
## factor(Clarity)8 -1.0094006  0.0078064 -129.304  < 2e-16 ***
## Depth            -0.0025212  0.0006111   -4.126 3.71e-05 ***
## Top_width_ratio  -0.0007664  0.0004453   -1.721   0.0852 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1545 on 42997 degrees of freedom
## Multiple R-squared:  0.9768, Adjusted R-squared:  0.9768
## F-statistic: 8.632e+04 on 21 and 42997 DF,  p-value: < 2.2e-16
```

R-squared: Indicates that approximately 97.68% of the variance in log price is explained by the predictors in the model, indicating a very strong fit of the model to the data.

Adjusted R-squared:The value is the same as the R-squared value, suggesting that the model is well-fitted without overfitting.

F-statistic: High F-statistic and very low p-value of less than 0.05 indicate that the overall model is highly significant and fits the data well.

The coefficients from the polynomial regression model highlight several significant predictors of diamond prices. Significant predictors include Carat and its quadratic term I(Carat^2), indicating a strong, non-linear relationship with price. Each unit increase in Carat is associated with a substantial increase in log price (4.34 units), highlighting Carat as a significant predictor of diamond price. The quadratic term for Carat shows a negative coefficient and is highly significant with a p value of less than 0.05. Categorical factors such as Cut, Color, and Clarity also show significant effects across their respective levels with p-values of less than 0.05, illustrating their substantial impact on price variation. Depth has a negative coefficient and is statistically significant with a p value of less than 0.05, suggesting that deeper diamonds tend to have slightly lower prices. Top_width_ratio shows a marginally insignificant negative coefficient with a p value of 0.0852, indicating that this predictor may not strongly influence diamond prices in this model.

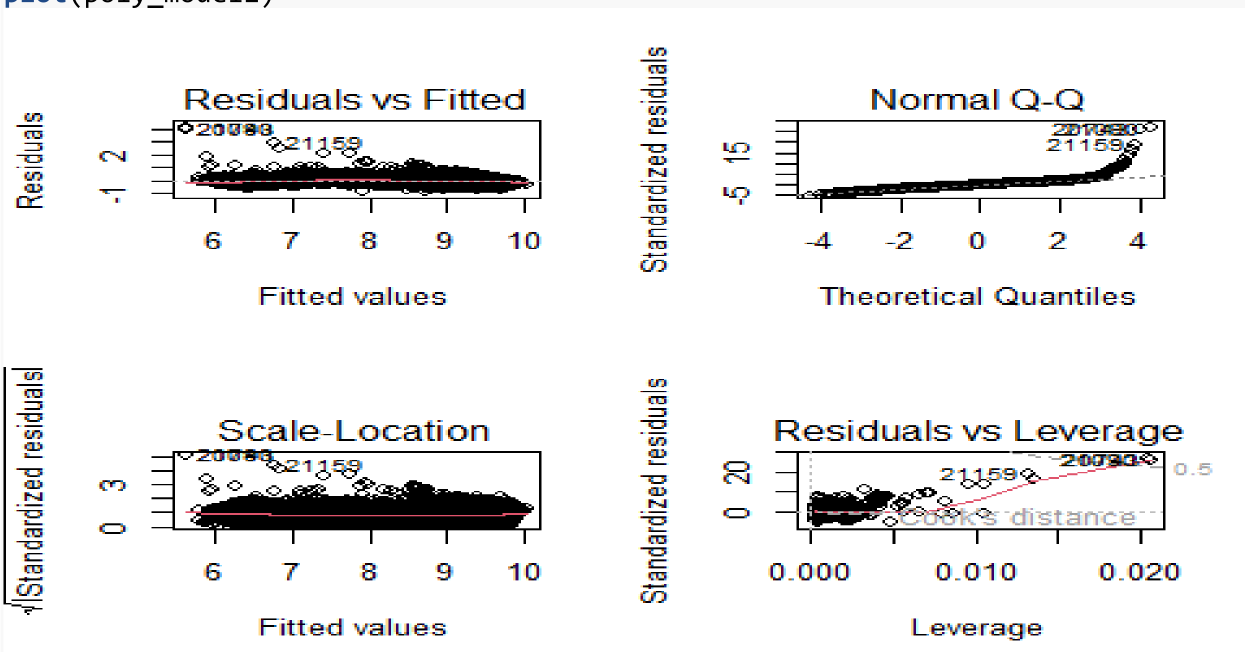
## ANOVA test

```
anova(poly_model2)

## Analysis of Variance Table
##
## Response: Log_price
##                    Df Sum Sq Mean Sq     F value    Pr(>F)
## Carat               1  37615   37615 1.5754e+06 < 2.2e-16 ***
## I(Carat^2)          1   3679    3679 1.5407e+05 < 2.2e-16 ***
## factor(Cut)         4    171      43 1.7859e+03 < 2.2e-16 ***
## factor(Color)       6    585      97 4.0806e+03 < 2.2e-16 ***
## factor(Clarity)     7   1233     176 7.3749e+03 < 2.2e-16 ***
## Depth               1      0       0 1.4058e+01 0.0001774 ***
## Top_width_ratio     1      0       0 2.9626e+00 0.0852173 .
## Residuals       42997   1027       0
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The ANOVA table confirms that Carat, its quadratic term, and categorical factors (Cut, Color, Clarity) are highly significant predictors of Log_price in the polynomial regression model. Depth also contributes significantly, to a lesser extent. Top_width_ratio is not significant in predicting diamond prices based on the given attributes. Overall, the model fits the data well with a p-value < 0.05, indicating its reliability in explaining variations in diamond prices.

## Model adequacy

```
par(mfrow = c(2,2))
# Test 1: Residual plots
plot(poly_model2)
```



- Residuals vs Fitted plot: The residuals are more evenly scattered around zero across the range of fitted values, indicating linearity and a better fit to the data.

- Q-Q plot: The Q-Q plot shows that the residuals follow the theoretical quantiles more closely, except in the tails. This suggests that the residuals are closer to being normally distributed.

- Scale-Location plot: The plot shows a more consistent sqrt(standardized residuals), indicating homoscedasticity and a better fit.

- Residuals vs Leverage plot: The residuals are more evenly scattered around the horizontal band, suggesting no significant influence of leverage on the residuals.

```
# Test 2: Test of constant variance
ncv_test = ncvTest(poly_model2)
print(ncv_test)

## Non-constant Variance Score Test
## Variance formula: ~ fitted.values
## Chisquare = 9.092131, Df = 1, p = 0.0025671
```

H0: There is constant variance(homoscedasticity). H1: There is non constant variance(heteroscedasticity).

With a p-value of 0.0025, which is less than 0.05, indicates that the variance of the residuals is not constant across the range of fitted values.

```
# Test 3: Test of Autocorrelation
DBtest = durbinWatsonTest(poly_model2)
print(DBtest)

##  lag Autocorrelation D-W Statistic p-value
##    1       0.3891048      1.221727       0
##  Alternative hypothesis: rho != 0
```

H0: There is no autocorrelation in the residuals. H1: There is autocorrelation in the residuals.

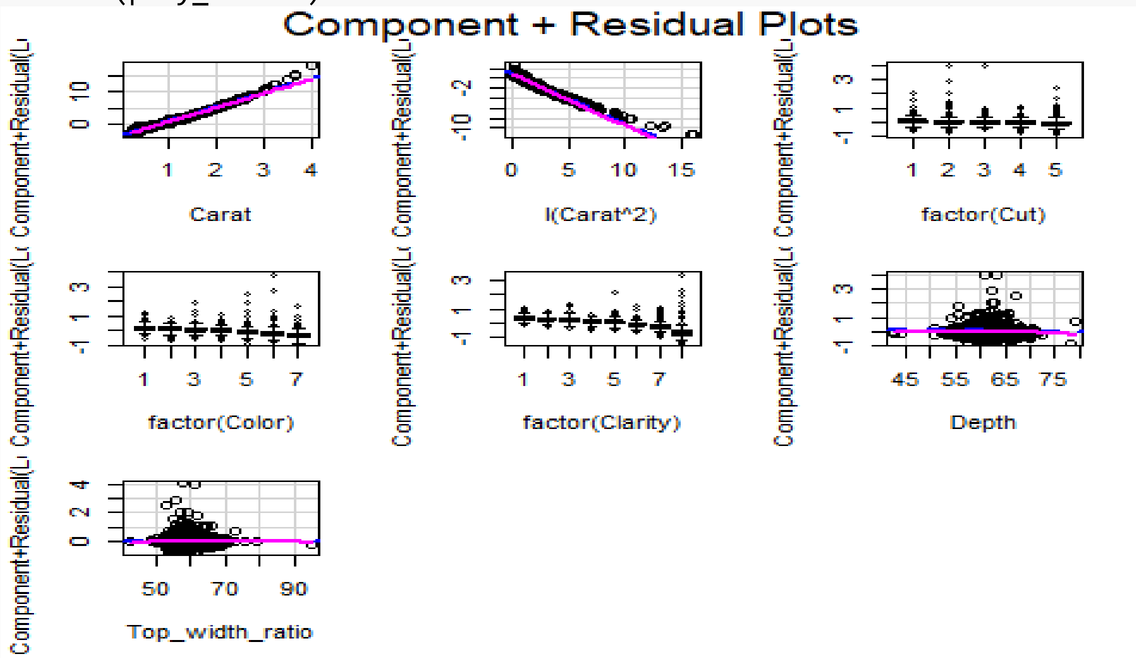
P-value of 0 suggest that there is a significant positive autocorrelation at lag 1 in the residuals of the model.

```
# Test 4: Testv of normality
model_residuals <- residuals(poly_model2)
ks.test(model_residuals, "pnorm", mean(model_residuals),
sd(model_residuals))

##
##  Asymptotic one-sample Kolmogorov-Smirnov test
##
## data:  model_residuals
## D = 0.033759, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

H0: Residuals are normally distributed. H1: Residuals are normally distributed.

With a p-value significantly less than any reasonable significance level (such as 0.05), the null hypothesis is rejected in favor of the alternative hypothesis that the distribution of residuals deviates from normality.

```r
# Test 5: Components residual plots
crPlots(poly_model2)
```




In the above plots, the residuals appear to be more randomly distributed with no obvious patterns, suggesting a better fit to the data.

The residual plots for the “Cut” and “Color” factors show relatively random scatter around zero, suggesting that the model adequately accounts for the effects of these variables.

However, the residual plot for “Clarity” exhibits a somewhat structured pattern, with higher clarity levels (lower factor levels) tending to have positive residuals.

The residuals for Depth and Top-Width-ratio show no clear pattern, suggesting an adequate fit.

```r
# Test 6: Test of multicollinearity
Multicoll_test = vif(poly_model2)
print(Multicoll_test)
```

```
##                       GVIF Df GVIF^(1/(2*Df))
## Carat            12.799456  1        3.577633
## I(Carat^2)       12.289688  1        3.505665
## factor(Cut)       1.932179  4        1.085815
## factor(Color)     1.177285  6        1.013694
## factor(Clarity)   1.340600  7        1.021158
```

```
## Depth              1.383628  1        1.176277
## Top_width_ratio  1.787361  1        1.336922
```

All variables have GVIF values less than 5, suggesting minimal multicollinearity concerns. Therefore, there is no strong evidence of multi-collinearity among the predictor variables in the model.

```
# Test 7: Outliers test
outlier_test = outlierTest(poly_model2)
print(outlier_test)

##         rstudent unadjusted p-value Bonferroni p
## 21090 26.693581        1.0446e-155  4.4936e-151
## 20743 26.458668        4.9002e-153  2.1080e-148
## 21159 19.197026         8.6304e-82   3.7127e-77
## 18874 16.637751         5.8049e-62   2.4972e-57
## 22082 13.574276         6.9431e-42   2.9868e-37
## 19418 13.374640         1.0262e-40   4.4146e-36
## 39687 11.605605         4.2946e-31   1.8475e-26
## 19389  9.692303         3.4258e-22   1.4737e-17
## 19262  9.321702         1.1974e-20   5.1511e-16
## 17358  8.491357         2.1067e-17   9.0630e-13

# Test 8: Influential measures
# In the appendix
```

## Corrective measure

### Fixing Autocorrelation

```
cochrane_orcutt_model_poly2 <- cochrane.orcutt(poly_model2)
summary(cochrane_orcutt_model_poly2)

## Call:
## lm(formula = Log_price ~ Carat + I(Carat^2) + factor(Cut) + factor(Color)
+
##     factor(Clarity) + Depth + Top_width_ratio, data = Diamond_Train_pre)
##
##                      Estimate  Std. Error  t value  Pr(>|t|)
## (Intercept)       7.02653056  0.03817462  184.063 < 2.2e-16 ***
## Carat             2.06891930  0.01149161  180.037 < 2.2e-16 ***
## I(Carat^2)       -0.46337061  0.00314840 -147.176 < 2.2e-16 ***
## factor(Cut)2     -0.01771984  0.00131164  -13.510 < 2.2e-16 ***
## factor(Cut)3     -0.01649556  0.00133882  -12.321 < 2.2e-16 ***
## factor(Cut)4     -0.02791829  0.00179440  -15.559 < 2.2e-16 ***
## factor(Cut)5     -0.06713524  0.00298270  -22.508 < 2.2e-16 ***
## factor(Color)2   -0.02384237  0.00180754  -13.191 < 2.2e-16 ***
## factor(Color)3   -0.04136982  0.00181133  -22.839 < 2.2e-16 ***
## factor(Color)4   -0.07824115  0.00183106  -42.730 < 2.2e-16 ***
## factor(Color)5   -0.13017380  0.00200398  -64.958 < 2.2e-16 ***
```

```
## factor(Color)6   -0.18537184  0.00228461  -81.139 < 2.2e-16 ***
## factor(Color)7   -0.24605959  0.00281522  -87.403 < 2.2e-16 ***
## factor(Clarity)2 -0.04581337  0.00338948  -13.516 < 2.2e-16 ***
## factor(Clarity)3 -0.07986978  0.00325594  -24.530 < 2.2e-16 ***
## factor(Clarity)4 -0.14194771  0.00317638  -44.688 < 2.2e-16 ***
## factor(Clarity)5 -0.18087442  0.00319219  -56.662 < 2.2e-16 ***
## factor(Clarity)6 -0.24571977  0.00330716  -74.299 < 2.2e-16 ***
## factor(Clarity)7 -0.32838878  0.00358247  -91.665 < 2.2e-16 ***
## factor(Clarity)8 -0.52148232  0.00559946  -93.131 < 2.2e-16 ***
## Depth            -0.00195138  0.00035046   -5.568 2.591e-08 ***
## Top_width_ratio  -0.00141969  0.00025265   -5.619 1.929e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1205 on 43010 degrees of freedom
## Multiple R-squared:  0.45 ,  Adjusted R-squared:  0.4499
## F-statistic: 1674.9 on 7 and 43010 DF,  p-value: < 0e+00
##
## Durbin-Watson statistic
## (original):    1.22173 , p-value: 0e+00
## (transformed): 2.38308 , p-value: 1e+00
```

The Cochrane-Orcutt method is used to correct autocorrelation in the model.

Based on the results all predictors are highly significant: - Carat has a positive significant effect on Log_price - I(Carat^2) has a negative significant effect, indicating a non-linear relationship - factor(Cut), factor(Color), factor(Clarity) are categorical levels with positive significant coefficients - Depth, Top_width_ratio both have small but significant negative effects

Multiple R-squared value is 0.45 and adjusted R-squared value is 0.4499 suggesting that about 45% of the variance in Log_price is explained by the model.

F-statistic is 1674.9, indicating the model is significantly better than a model with no predictors.

Durbin-Watson Statistic: Original value was 1.22173, indicating the presence of positive autocorrelation. Transformed value is 2.38308, close to 2, suggesting that the Cochrane-Orcutt procedure has successfully reduced autocorrelation.

## Variable selection

### All possible regression subsets

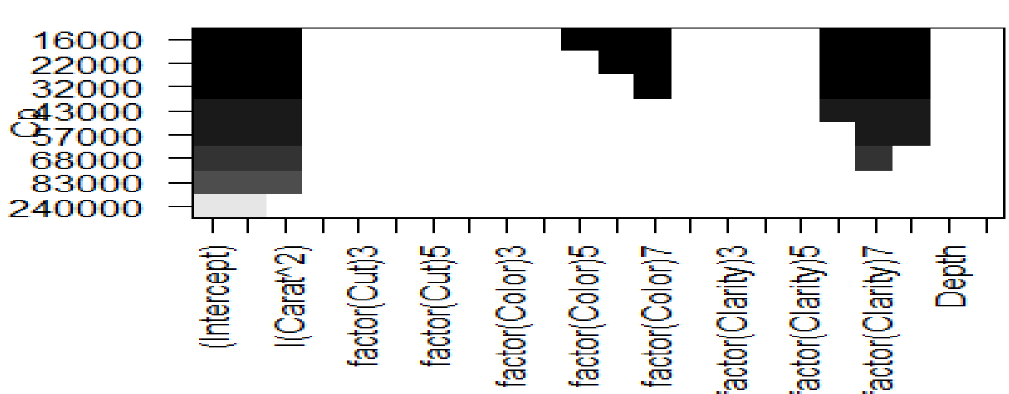
Subset selection is done to identify the most relevant predictors for the model and removing irrelevant variables. An exhaustive search was performed, evaluating all possible subsets of predictors.

```r
Diamond_model_r <- leaps::regsubsets(Log_price ~ Carat + I(Carat^2) +
factor(Cut) + factor(Color) + factor(Clarity) + Depth + Top_width_ratio, data
= Diamond_Train_pre)
subset_result <- summary(Diamond_model_r)
subset_result
```

```
## Subset selection object
## Call: regsubsets.formula(Log_price ~ Carat + I(Carat^2) + factor(Cut) +
##     factor(Color) + factor(Clarity) + Depth + Top_width_ratio,
##     data = Diamond_Train_pre)
## 21 Variables  (and intercept)
##                  Forced in Forced out
## Carat                FALSE      FALSE
## I(Carat^2)           FALSE      FALSE
## factor(Cut)2         FALSE      FALSE
## factor(Cut)3         FALSE      FALSE
## factor(Cut)4         FALSE      FALSE
## factor(Cut)5         FALSE      FALSE
## factor(Color)2       FALSE      FALSE
## factor(Color)3       FALSE      FALSE
## factor(Color)4       FALSE      FALSE
## factor(Color)5       FALSE      FALSE
## factor(Color)6       FALSE      FALSE
## factor(Color)7       FALSE      FALSE
## factor(Clarity)2     FALSE      FALSE
## factor(Clarity)3     FALSE      FALSE
## factor(Clarity)4     FALSE      FALSE
## factor(Clarity)5     FALSE      FALSE
## factor(Clarity)6     FALSE      FALSE
## factor(Clarity)7     FALSE      FALSE
## factor(Clarity)8     FALSE      FALSE
## Depth                FALSE      FALSE
## Top_width_ratio      FALSE      FALSE
## 1 subsets of each size up to 8
## Selection Algorithm: exhaustive
##          Carat I(Carat^2) factor(Cut)2 factor(Cut)3 factor(Cut)4
factor(Cut)5
## 1  ( 1 ) "*"   " "        " "          " "          " "          " "
## 2  ( 1 ) "*"   "*"        " "          " "          " "          " "
## 3  ( 1 ) "*"   "*"        " "          " "          " "          " "
## 4  ( 1 ) "*"   "*"        " "          " "          " "          " "
## 5  ( 1 ) "*"   "*"        " "          " "          " "          " "
## 6  ( 1 ) "*"   "*"        " "          " "          " "          " "
## 7  ( 1 ) "*"   "*"        " "          " "          " "          " "
## 8  ( 1 ) "*"   "*"        " "          " "          " "          " "
##          factor(Color)2 factor(Color)3 factor(Color)4 factor(Color)5
## 1  ( 1 ) " "            " "            " "            " "
## 2  ( 1 ) " "            " "            " "            " "
## 3  ( 1 ) " "            " "            " "            " "
## 4  ( 1 ) " "            " "            " "            " "
```

```
## 5  ( 1 ) " "            " "            " "            " "
## 6  ( 1 ) " "            " "            " "            " "
## 7  ( 1 ) " "            " "            " "            " "
## 8  ( 1 ) " "            " "            " "            "*"
##          factor(Color)6 factor(Color)7 factor(Clarity)2 factor(Clarity)3
## 1  ( 1 ) " "            " "            " "              " "
## 2  ( 1 ) " "            " "            " "              " "
## 3  ( 1 ) " "            " "            " "              " "
## 4  ( 1 ) " "            " "            " "              " "
## 5  ( 1 ) " "            " "            " "              " "
## 6  ( 1 ) " "            "*"            " "              " "
## 7  ( 1 ) "*"            "*"            " "              " "
## 8  ( 1 ) "*"            "*"            " "              " "
##          factor(Clarity)4 factor(Clarity)5 factor(Clarity)6
factor(Clarity)7
## 1  ( 1 ) " "              " "              " "              " "
## 2  ( 1 ) " "              " "              " "              " "
## 3  ( 1 ) " "              " "              " "              "*"
## 4  ( 1 ) " "              " "              " "              "*"
## 5  ( 1 ) " "              " "              "*"              "*"
## 6  ( 1 ) " "              " "              "*"              "*"
## 7  ( 1 ) " "              " "              "*"              "*"
## 8  ( 1 ) " "              " "              "*"              "*"
##          factor(Clarity)8 Depth Top_width_ratio
## 1  ( 1 ) " "              " "   " "
## 2  ( 1 ) " "              " "   " "
## 3  ( 1 ) " "              " "   " "
## 4  ( 1 ) "*"              " "   " "
## 5  ( 1 ) "*"              " "   " "
## 6  ( 1 ) "*"              " "   " "
## 7  ( 1 ) "*"              " "   " "
## 8  ( 1 ) "*"              " "   " "
```

```
plot(Diamond_model_r, scale = "Cp")
```



CP, adjusted R square and BIC plots were plotted. Since the plots were identical, only CP plot is displayed. The BIC and adjusted R square plot are displayed in the appendix. The lowest value is observed in the model with the regressors: Carat, I(Carat^2), factor(Color)5, factor(Color)6, factor(Color)7, factor(Clarity)6, factor(Clarity)7, and factor(Clarity)8.

```
##      Diamond_model_RSS Diamond_model_r2 Diamond_model_Cp Diamond_model_BIC
## [1,]          6693.328        0.8489382        237317.37         -81287.42
## [2,]          3014.769        0.9319596         83252.65        -115588.18
## [3,]          2658.179        0.9400075         68319.82        -120992.83
## [4,]          2388.647        0.9460906         57033.18        -125581.50
## [5,]          2058.754        0.9535359         43218.49        -131964.57
## [6,]          1800.204        0.9593712         32391.78        -137727.11
## [7,]          1563.272        0.9647185         22470.50        -143787.24
## [8,]          1404.654        0.9682983         15829.23        -148379.16
##      Diamond_model_Adj_r2
## [1,]            0.8489347
## [2,]            0.9319565
## [3,]            0.9400033
## [4,]            0.9460856
## [5,]            0.9535305
## [6,]            0.9593655
## [7,]            0.9647128
## [8,]            0.9682924
```

```
which.min(Diamond_model_Cp)
```

```
## [1] 8
```

```
which.min(Diamond_model_BIC)
```

```
## [1] 8
```

```
which.max(Diamond_model_Adj_r2)
```

```
## [1] 8
```

Based on the minimum value of CP and BIC, and the maximum Adjusted R2 value, the model subset model with 8 predictors is the best model.

## Regression subsets

The regressors identified are Carat, Carat^2, Color, and Cut.

```
reg_subset_model <- lm(Diamond_Train_pre$Log_price ~ Carat + I(Carat^2) +
factor(Color) + factor(Clarity), data = Diamond_Train_pre)
reg_subset_result <- summary(reg_subset_model)
reg_subset_result
```

```
##
## Call:
```

```
## lm(formula = Diamond_Train_pre$Log_price ~ Carat + I(Carat^2) + 
##     factor(Color) + factor(Clarity), data = Diamond_Train_pre)
## 
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -0.8901 -0.1016  0.0071  0.1021  4.0246 
## 
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)    
## (Intercept)        5.758592   0.005121 1124.53   <2e-16 ***
## Carat              4.316033   0.005723  754.17   <2e-16 ***
## I(Carat^2)        -0.995115   0.002546 -390.80   <2e-16 ***
## factor(Color)2    -0.058985   0.002803  -21.04   <2e-16 ***
## factor(Color)3    -0.098744   0.002834  -34.84   <2e-16 ***
## factor(Color)4    -0.167051   0.002775  -60.20   <2e-16 ***
## factor(Color)5    -0.272689   0.002952  -92.38   <2e-16 ***
## factor(Color)6    -0.387234   0.003315 -116.83   <2e-16 ***
## factor(Color)7    -0.523043   0.004080 -128.21   <2e-16 ***
## factor(Clarity)2 -0.101102   0.005113  -19.77   <2e-16 ***
## factor(Clarity)3 -0.167551   0.004885  -34.30   <2e-16 ***
## factor(Clarity)4 -0.278741   0.004661  -59.80   <2e-16 ***
## factor(Clarity)5 -0.347735   0.004543  -76.54   <2e-16 ***
## factor(Clarity)6 -0.488286   0.004563 -107.01   <2e-16 ***
## factor(Clarity)7 -0.652977   0.004744 -137.63   <2e-16 ***
## factor(Clarity)8 -1.058863   0.007846 -134.96   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 0.158 on 43003 degrees of freedom
## Multiple R-squared:  0.9758, Adjusted R-squared:  0.9758 
## F-statistic: 1.155e+05 on 15 and 43003 DF,  p-value: < 2.2e-16
```

Residual standard error is 0.158. This is used to measure the average amount of deviation of observed values from the predicted values

R-squared and Adjusted R-squared values are 0.9758. Thus about 97.58% of the variability in Log_price can be explained by the model.

H0: p-value > 0.05 indicating model is insignificant. H1: p-value < 0.05 indicating model is significant

p-value < 2.2e-16 indicates that the model is highly significant overall.

### ANOVA test

```
anova(reg_subset_model)
```

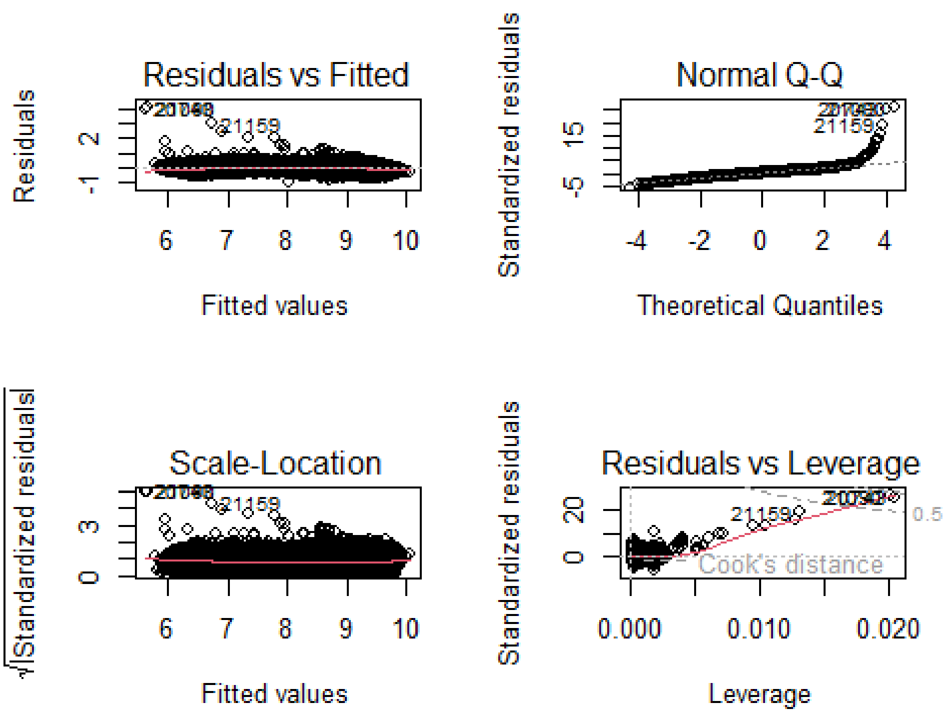
```
## Analysis of Variance Table
## 
## Response: Diamond_Train_pre$Log_price
##                     Df Sum Sq Mean Sq   F value    Pr(>F)
```

```
## Carat                  1  37615   37615 1507121.6 < 2.2e-16 ***
## I(Carat^2)             1   3679    3679  147388.1 < 2.2e-16 ***
## factor(Color)          6    582      97    3885.8 < 2.2e-16 ***
## factor(Clarity)        7   1360     194    7782.1 < 2.2e-16 ***
## Residuals          43003   1073       0
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Carat and I(Carat^2) have very high F values and extremely low p-values, indicating they are highly significant predictors of Log_price. factor(Color) and factor(Clarity) are also highly significant, although their F values are lower than for Carat and I(Carat^2), which means that they significantly contribute to the model but their contribution is relatively smaller.

## Model adequacy

```r
par(mfrow = c(2,2))
# Test 1: Residual plots
plot(reg_subset_model)
```




- Residuals vs Fitted plot: The residuals are more evenly scattered around zero across the range of fitted values, indicating linearity and a better fit to the data. There are some outliers in this such as observation 21159.

- Q-Q plot: The Q-Q plot shows that the residuals follow the theoretical quantiles more closely, except in the tails. This suggests that the residuals are closer to being normally distributed.

- Scale-Location plot: The plot shows a more consistent sqrt(standardized residuals), indicating homoscedasticity and a better fit.

- Residuals vs Leverage plot: The residuals are more evenly scattered around the horizontal band, suggesting no significant influence of leverage on the residuals.

```r
# Test 2: Test of constant variance
ncv_test = ncvTest(reg_subset_model)
print(ncv_test)

## Non-constant Variance Score Test
## Variance formula: ~ fitted.values
## Chisquare = 5.731582, Df = 1, p = 0.016662
```

H0: There is homoscedasticity in the residuals H1: There is heteroscedasticity in the residuals

The p-value is less than 0.05, indicating that the null hypothesis of constant variance can be rejected at the 5% significance level. This suggests that there is significant evidence of heteroscedasticity in the residuals.

```r
# Test 3: Test of Autocorrelation
DBtest = durbinWatsonTest(reg_subset_model)
print(DBtest)

##  lag Autocorrelation D-W Statistic p-value
##    1       0.3884209      1.223103       0
##  Alternative hypothesis: rho != 0
```

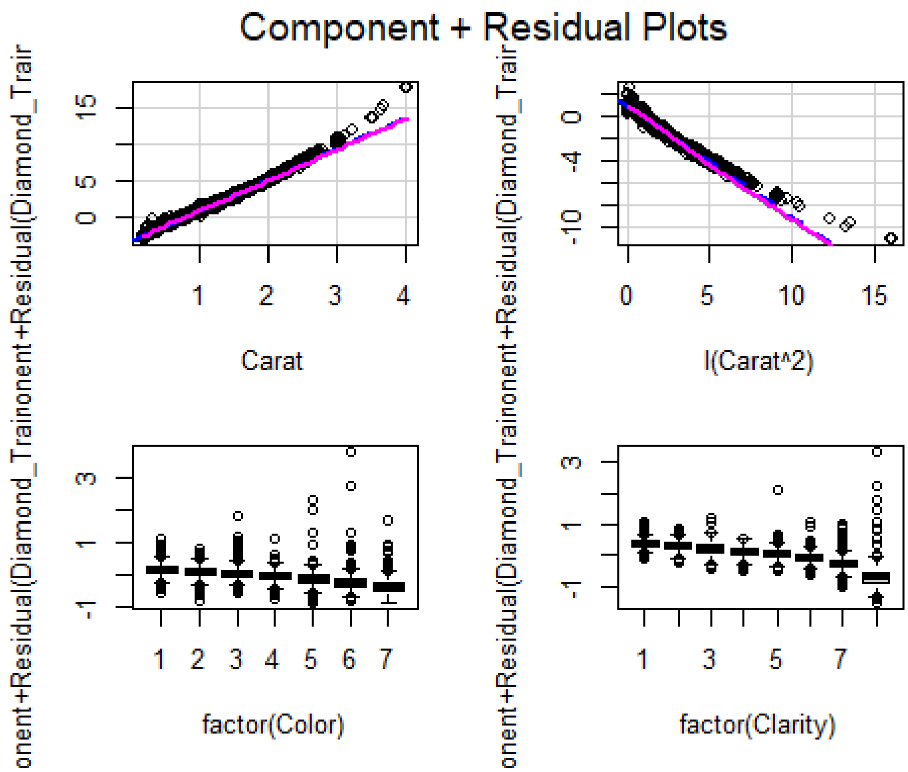
H0: There is no significant autocorrelation in residuals H1: there is significant autocorrelation in residuals the p-value is 0, which indicates strong evidence against the null hypothesis. Therefore, we reject the null hypothesis in favor of the alternative hypothesis, concluding that there is first-order autocorrelation in the residuals

```r
model_residuals <- residuals(reg_subset_model)
# Test 4: Test of normality
ks.test(model_residuals, "pnorm", mean(model_residuals),
sd(model_residuals))

##
##  Asymptotic one-sample Kolmogorov-Smirnov test
##
## data:  model_residuals
## D = 0.032432, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

H0: residuals follow normal distribution H1: residuals do not follow normal distribution

A very small p-value of < 2.2e-16 indicates strong evidence against the null hypothesis, suggesting that the residuals do not follow a normal distribution.

```
# Test 5: Components residual plots
crPlots(reg_subset_model)
```




In the CR plot for Carat, The pink line follows the blue dotted line of linearity and most of the points are distributed along the line. This suggests that the relationship between the response variable and carat is linear. as the line goes up, the relationship is positive.

In the CR plot for I(Carat^2) the line of linearity is followed, but the relation is negative.

In the CR plot of Color and Clarity the BoxPlot of each color and clarity is seen.

```
# Test 6: Test of multicollinearity
Multicoll_test = vif(reg_subset_model)
print(Multicoll_test)

##                       GVIF Df GVIF^(1/(2*Df))
## Carat            12.656819  1        3.557642
## I(Carat^2)       12.225294  1        3.496469
## factor(Color)     1.169173  6        1.013110
## factor(Clarity)   1.252158  7        1.016192
```

The VIF value is less than 5 for all the variables Based on the output, the variables Carat and I(Carat^2) are highly correlated, as they are the same variable have high VIF values (indicating multicollinearity), while factor(Color) and factor(Clarity) have lower VIF values, suggesting less multicollinearity.

```
# Test 7: Outliers test
outlier_test = outlierTest(reg_subset_model)
print(outlier_test)

##        rstudent unadjusted p-value Bonferroni p
## 21090 25.936127        3.5506e-147  1.5274e-142
## 20743 25.859526        2.5112e-146  1.0803e-141
## 21159 18.829202         9.0402e-79   3.8890e-74
## 18874 15.598122         1.0592e-54   4.5568e-50
## 19418 13.373789         1.0379e-40   4.4651e-36
## 22082 13.122525         2.9112e-39   1.2524e-34
## 39687 10.940972         7.9884e-28   3.4365e-23
## 19389  9.822716         9.4986e-23   4.0862e-18
## 19262  9.237670         2.6306e-20   1.1317e-15
## 22085  8.117050         4.9022e-16   2.1089e-11
```

## Stepwise regression subset

### Forward stepwise regression

```
Diamond_forward <- regsubsets(Diamond_Train_pre$Log_price ~ Carat + +
I(Carat^2) + factor(Cut) + factor(Color) + factor(Clarity) + Depth +
Top_width_ratio, data = Diamond_Train_pre, method = "forward")
summary_Diamond_forward <- summary(Diamond_forward)

Diamond_forward_model_Cp <- summary_Diamond_forward$cp
Diamond_forward_model_BIC <- summary_Diamond_forward$bic
Diamond_forward_model_Adj_r2 <- summary_Diamond_forward$adjr2

which.min(Diamond_forward_model_Cp)

## [1] 8

which.min(Diamond_forward_model_BIC)

## [1] 8

which.max(Diamond_forward_model_Adj_r2)

## [1] 8
```

Same model was observed as in all possible subsets so we did not fit the model again and did not do model adequacy check.

### Backward stepwise

```
Diamond_backward <- regsubsets(Diamond_Train_pre$Log_price ~ Carat + +
I(Carat^2) + factor(Cut) + factor(Color) + factor(Clarity) + Depth +
```

```r
Top_width_ratio, data = Diamond_Train_pre, method = "backward")
summary_Diamond_backward <- summary(Diamond_backward)

Diamond_backward_model_Cp <- summary_Diamond_backward$cp
Diamond_backward_model_BIC <- summary_Diamond_backward$bic
Diamond_backward_model_Adj_r2 <- summary_Diamond_backward$adjr2

which.min(Diamond_backward_model_Cp)

## [1] 8

which.min(Diamond_backward_model_BIC)

## [1] 8

which.max(Diamond_backward_model_Adj_r2)

## [1] 8
```

We got the same model as all possible subsets and forward model, so we did not repeat the process.

### Seqrep stepwise

```r
Diamond_seqrep <- regsubsets(Diamond_Train_pre$Log_price ~ Carat +
I(Carat^2) + factor(Cut) + factor(Color) + factor(Clarity) + Depth +
Top_width_ratio, data = Diamond_Train_pre, method = "seqrep")
summary_Diamond_seqrep <- summary(Diamond_seqrep)

Diamond_seqrep_model_Cp <- summary_Diamond_seqrep$cp
Diamond_seqrep_model_BIC <- summary_Diamond_seqrep$bic
Diamond_seqrep_model_Adj_r2 <- summary_Diamond_seqrep$adjr2

which.min(Diamond_seqrep_model_Cp)

## [1] 8

which.min(Diamond_seqrep_model_BIC)

## [1] 8

which.max(Diamond_seqrep_model_Adj_r2)

## [1] 8
```

Same model was observed as as all possible subsets, forward and backward so we didn’t fit the model.

Based on Forward, backward, and seqrep stepwise regression, the best model identified(in the appendix) is same as the one identified in all possible subset regression. So, the model was not fitted again.

## Press residuals

```r
DAAG::press(Diamond_model)

## [1] 4953.37

DAAG::press(poly_model2)

## [1] 1030.004

DAAG::press(reg_subset_model)

## [1] 1076.324
```

According to PRESS residuals, the poly model 2 has the least press value, indicating that the predictive accuracy for polymodel 2 is the best fit for the data.

```r
# Predict on the testing set
predictions_full_model <- predict(Diamond_model, Diamond_Test)

# Calculate performance metrics
mse <- mean((predictions_full_model - Diamond_Test$Log_price)^2)
rmse <- sqrt(mse)
mae <- mean(abs(predictions_full_model - Diamond_Test$Log_price))

## Diamond_model

## MSE: 0.1106793

## RMSE: 0.332685

## MAE: 0.2661608

# Predict on the testing set
predictions_poly_model2 <- predict(poly_model2, Diamond_Test)

# Calculate performance metrics
mse_poly <- mean((predictions_poly_model2 - Diamond_Test$Log_price)^2)
rmse_poly <- sqrt(mse_poly)
mae_poly <- mean(abs(predictions_poly_model2 - Diamond_Test$Log_price))

## Poly_model2)

## MSE: 0.0244774

## RMSE: 0.1564526

## MAE: 0.1197905

# Predict on the testing set
predictions_subset_model <- predict(reg_subset_model, Diamond_Test)

# Calculate performance metrics
mse_reg_subset <- mean((predictions_subset_model - Diamond_Test$Log_price)^2)
```

```
rmse_reg_subset <- sqrt(mse_reg_subset)
mae_reg_subset <- mean(abs(predictions_subset_model -
Diamond_Test$Log_price))

## Regression subset model

## MSE: 0.02553255

## RMSE: 0.1597891

## MAE: 0.1221663
```

The best fit of the poly model 2 is also confirmed by the metrics. The poly_model2 performs the best on the testing set, as it has the lowest MSE, RMSE, and MAE values, indicating better predictive performance compared to the other two models.

## Conclusion

This project aimed to develop a regression model for predicting diamond prices using various attributes such as carat weight, cut, color, clarity, and dimensions. After extensive data preprocessing, transformation, exploratory analysis, and regression modeling, the polynomial regression model was identified as the best fit for predicting log-transformed diamond prices. Despite initial violations of regression assumptions, corrective measures like removing influential values and employing the Cochrane-Orcutt procedure improved model adequacy. Subset selection further optimized the model by identifying the most relevant predictors.

**Key Findings** Carat: The most significant predictor, with a strong, non-linear relationship with diamond price. Cut, Color, Clarity: Significant categorical predictors affecting diamond price. Depth and Top-Width Ratio: Although significant, their impact on price is relatively minimal compared to other factors.

## Reference


1. Montgomery, D. C., Peck, E. A., & Vining, G. G. (2012). Introduction to Linear Regression (5th ed.). John Wiley & Sons.

2. Yan Wang (2024). Lecture notes and Lab recordings. RMIT Canvas. https://rmit.instructure.com/courses/124443/modules

3. Agrawal, S “Diamonds,” Kaggle, viewed 13 June 2024, https://www.kaggle.com/datasets/shivam2503/diamonds.

# Appendix

### Summary of data

```
str(Diamond)

## tibble [53,775 × 8] (S3: tbl_df/tbl/data.frame)
##  $ Carat          : num [1:53775] 0.23 0.21 0.23 0.29 0.31 0.24 0.24 0.26
0.22 0.23 ...
##  $ Cut            : int [1:53775] 1 2 4 2 4 3 3 3 5 3 ...
##  $ Color          : int [1:53775] 2 2 2 6 7 7 6 5 2 5 ...
##  $ Clarity        : int [1:53775] 7 6 4 5 7 3 2 6 5 4 ...
##  $ Depth          : num [1:53775] 61.5 59.8 56.9 62.4 63.3 62.8 62.3 61.9
65.1 59.4 ...
##  $ Top_width_ratio: num [1:53775] 55 61 65 58 58 57 57 55 61 61 ...
##  $ Price_USD      : num [1:53775] 326 326 327 334 335 336 336 337 337 338
...
##  $ Log_price      : num [1:53775] 5.79 5.79 5.79 5.81 5.81 ...

# Bar plot for cut
plot1 = ggplot(Diamond, aes(x = Cut)) +
  geom_bar(fill = "purple") + labs(title = "Distribution of Diamond Cut", x =
"Cut", y = "Count") +
  theme(plot.margin = unit(c(0.5, 0.5, 0.5, 0.5), "cm")) + theme(plot.title =
element_text(size = 12)) +
  theme(axis.text = element_text(size = 10)) + theme(axis.title =
element_text(size = 10))

# Bar plot for color
plot2 = ggplot(Diamond, aes(x = Color)) +
  geom_bar(fill = "orange") +
  labs(title = "Distribution of Diamond Color", x = "Color", y = "Count") +
theme(plot.margin = unit(c(0.5, 0.5, 0.5, 0.5), "cm")) +   theme(plot.title =
element_text(size = 12)) + theme(axis.text = element_text(size = 10)) +
  theme(axis.title = element_text(size = 10))

# Bar plot for clarity
plot3 = ggplot(Diamond, aes(x = Clarity)) +
  geom_bar(fill = "turquoise") +
  labs(title = "Distribution of Diamond Clarity", x = "Clarity", y = "Count")
+
  theme(plot.margin = unit(c(0.5, 0.5, 0.5, 0.5), "cm")) + theme(plot.title =
element_text(size = 12)) +
  theme(axis.text = element_text(size = 10)) + theme(axis.title =
element_text(size = 10))

grid.arrange(plot1, plot2, plot3, ncol = 2)
```

```r
Diamond$Cut <- factor(Diamond$Cut, levels = c("Fair", "Good", "Very Good",
"Premium", "Idea
 l"))
 plot5 <- ggplot(Diamond, aes(x = Carat, y = Log_price, color = Cut)) +
geom_point() +
labs(title = "Scatter Plot of Carat vs Diamond price based on cut", x =
"Carat", y = "Price
 _USD") +
scale_x_continuous(limits = c(0, 4), breaks = seq(0, 4, by = 0.50)) +
theme_minimal()
### Scatter plot between Carat, Clarity and Diamond price
Diamond$Clarity <- factor(Diamond$Clarity, levels = c("I1", "SI2", "SI1",
"VS2", "VS1", "VVS
 2", "VVS1", "IF"))
 plot6 <- ggplot(Diamond, aes(x = Carat, y = Price_USD, color = Clarity)) +
geom_point() +
labs(title = "Scatter Plot of Carat vs Diamond price based on clarity", x =
"Carat", y = "P
 rice_USD") +
scale_x_continuous(limits = c(0, 4), breaks = seq(0, 4, by = 0.50)) +
theme_minimal()
```

### Diamond model summary

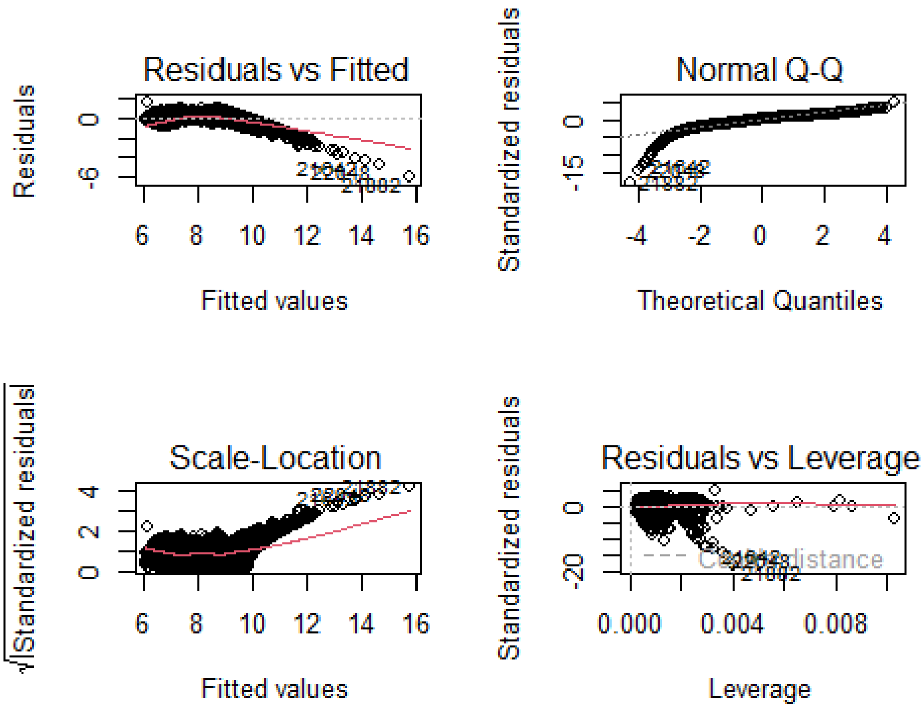
```r
summary(Diamond_model)

##
## Call:
## lm(formula = Diamond_Train$Log_price ~ Carat + factor(Cut) +
##     factor(Color) + factor(Clarity) + Depth + Top_width_ratio,
##     data = Diamond_Train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.9749 -0.2202  0.0582  0.2482  1.5919
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)       6.104e+00  1.162e-01  52.519  < 2e-16 ***
## Carat             2.192e+00  3.965e-03 552.863  < 2e-16 ***
## factor(Cut)2     -4.379e-02  4.896e-03  -8.945  < 2e-16 ***
## factor(Cut)3     -3.939e-02  4.775e-03  -8.250  < 2e-16 ***
## factor(Cut)4     -5.033e-02  6.768e-03  -7.437 1.05e-13 ***
## factor(Cut)5     -1.046e-01  1.124e-02  -9.302  < 2e-16 ***
## factor(Color)2   -5.781e-02  6.020e-03  -9.602  < 2e-16 ***
## factor(Color)3   -5.391e-02  6.082e-03  -8.863  < 2e-16 ***
## factor(Color)4   -1.291e-01  5.959e-03 -21.673  < 2e-16 ***
## factor(Color)5   -2.599e-01  6.342e-03 -40.975  < 2e-16 ***
## factor(Color)6   -4.224e-01  7.116e-03 -59.352  < 2e-16 ***
## factor(Color)7   -5.814e-01  8.756e-03 -66.402  < 2e-16 ***
```

```
## factor(Clarity)2 -9.206e-02  1.099e-02  -8.378  < 2e-16 ***
## factor(Clarity)3 -1.014e-01  1.050e-02  -9.660  < 2e-16 ***
## factor(Clarity)4 -1.490e-01  1.002e-02 -14.874  < 2e-16 ***
## factor(Clarity)5 -2.125e-01  9.772e-03 -21.749  < 2e-16 ***
## factor(Clarity)6 -3.072e-01  9.825e-03 -31.267  < 2e-16 ***
## factor(Clarity)7 -4.874e-01  1.023e-02 -47.657  < 2e-16 ***
## factor(Clarity)8 -1.014e+00  1.712e-02 -59.252  < 2e-16 ***
## Depth            -4.376e-05  1.341e-03  -0.033    0.974
## Top_width_ratio   6.649e-03  9.765e-04   6.809 9.93e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3392 on 43001 degrees of freedom
## Multiple R-squared:  0.8884, Adjusted R-squared:  0.8883
## F-statistic: 1.711e+04 on 20 and 43001 DF,  p-value: < 2.2e-16
```

```
# Descriptive statistics of data
summary(Diamond)
```

```
##      Carat                    Cut             Color          Clarity
##  Min.   :0.2000   Fair     :    0   Min.   :1.000   I1     :    0
##  1st Qu.:0.4000   Good     :    0   1st Qu.:2.000   SI2    :    0
##  Median :0.7000   Very Good:    0   Median :4.000   SI1    :    0
##  Mean   :0.7975   Premium  :    0   Mean   :3.594   VS2    :    0
##  3rd Qu.:1.0400   Idea\n l :    0   3rd Qu.:5.000   VS1    :    0
##  Max.   :5.0100   NA's     :53775   Max.   :7.000   (Other):    0
##                                                     NA's   :53775
##      Depth       Top_width_ratio   Price_USD       Log_price
##  Min.   :43.00   Min.   :43.00   Min.   :  326   Min.   :5.787
##  1st Qu.:61.00   1st Qu.:56.00   1st Qu.:  951   1st Qu.:6.858
##  Median :61.80   Median :57.00   Median : 2401   Median :7.784
##  Mean   :61.75   Mean   :57.46   Mean   : 3931   Mean   :7.787
##  3rd Qu.:62.50   3rd Qu.:59.00   3rd Qu.: 5324   3rd Qu.:8.580
##  Max.   :79.00   Max.   :95.00   Max.   :18823   Max.   :9.843
##
```

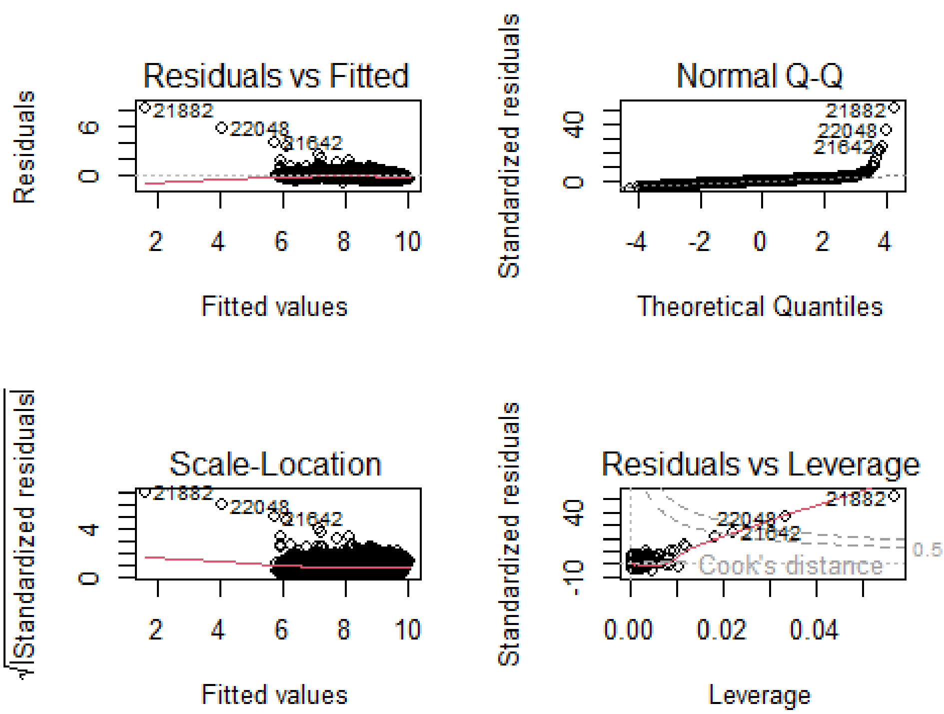
```
par(mfrow = c(2,2))
plot(Diamond_model)
```




**Polynomial fitted model**

```
summary(poly_model)

##
## Call:
## lm(formula = Log_price ~ Carat + I(Carat^2) + factor(Cut) + factor(Color)
+
##     factor(Clarity) + Depth + Top_width_ratio, data = Diamond_Train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.8027 -0.1046  0.0112  0.1032  8.1660
##
## Coefficients:
##                   Estimate Std. Error  t value Pr(>|t|)
## (Intercept)       5.9653461  0.0562139  106.119  < 2e-16 ***
## Carat             4.2177169  0.0057292  736.179  < 2e-16 ***
## I(Carat^2)       -0.9425830  0.0025119 -375.247  < 2e-16 ***
## factor(Cut)2     -0.0353032  0.0023680  -14.908  < 2e-16 ***
## factor(Cut)3     -0.0533630  0.0023100  -23.100  < 2e-16 ***
## factor(Cut)4     -0.0765561  0.0032744  -23.380  < 2e-16 ***
## factor(Cut)5     -0.1265067  0.0054371  -23.267  < 2e-16 ***
## factor(Color)2   -0.0576654  0.0029117  -19.805  < 2e-16 ***
## factor(Color)3   -0.0953456  0.0029440  -32.387  < 2e-16 ***
## factor(Color)4   -0.1659434  0.0028838  -57.544  < 2e-16 ***
## factor(Color)5   -0.2724794  0.0030676  -88.825  < 2e-16 ***
```

```
## factor(Color)6    -0.3917779  0.0034430 -113.791  < 2e-16 ***
## factor(Color)7    -0.5208814  0.0042382 -122.900  < 2e-16 ***
## factor(Clarity)2 -0.0944842  0.0053145  -17.779  < 2e-16 ***
## factor(Clarity)3 -0.1559534  0.0050812  -30.692  < 2e-16 ***
## factor(Clarity)4 -0.2621214  0.0048537  -54.005  < 2e-16 ***
## factor(Clarity)5 -0.3304871  0.0047371  -69.766  < 2e-16 ***
## factor(Clarity)6 -0.4632762  0.0047703  -97.116  < 2e-16 ***
## factor(Clarity)7 -0.6290727  0.0049610 -126.805  < 2e-16 ***
## factor(Clarity)8 -0.9866087  0.0082805 -119.148  < 2e-16 ***
## Depth            -0.0021933  0.0006488   -3.381 0.000724 ***
## Top_width_ratio  -0.0004479  0.0004727   -0.948 0.343373
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.164 on 43000 degrees of freedom
## Multiple R-squared:  0.9739, Adjusted R-squared:  0.9739
## F-statistic: 7.638e+04 on 21 and 43000 DF,  p-value: < 2.2e-16

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

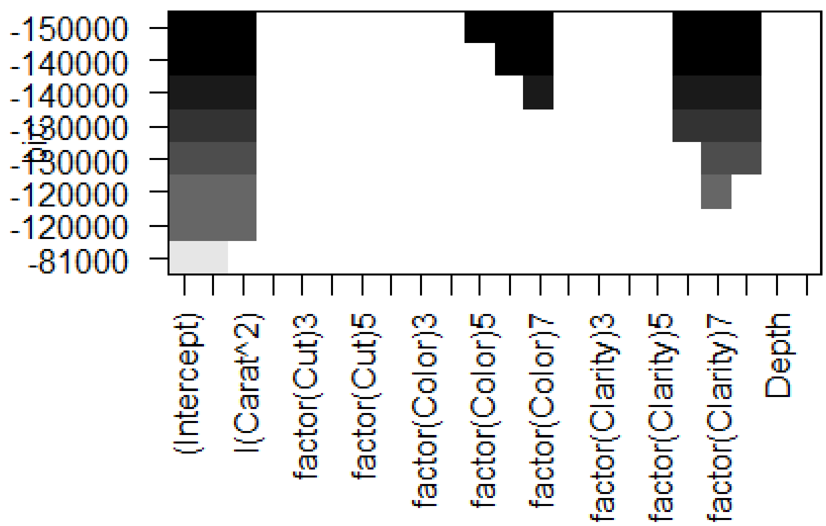



# Influential measures

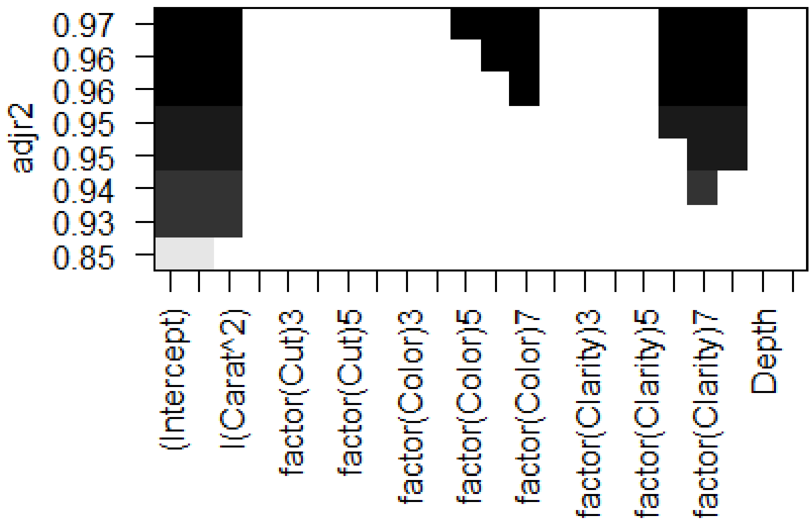
```
#influence_measure_poly <- influence.measures(poly_model2)
#influence_measure_poly
```

**All possible subset BIC and adjusted r2 plot**

```r
plot(Diamond_model_r, scale = "bic")
```




```r
plot(Diamond_model_r, scale = "adjr2")
```




# Test values for all possible subsets

```
Diamond_model_RSS <- subset_result$rss
Diamond_model_r2 <- subset_result$rsq
Diamond_model_Cp <- subset_result$cp
Diamond_model_BIC <- subset_result$bic
Diamond_model_Adj_r2 <- subset_result$adjr2
cbind(Diamond_model_RSS, Diamond_model_r2 , Diamond_model_Cp ,
Diamond_model_BIC, Diamond_model_Adj_r2)

##      Diamond_model_RSS Diamond_model_r2 Diamond_model_Cp Diamond_model_BIC
## [1,]          6693.328        0.8489382        237317.37         -81287.42
## [2,]          3014.769        0.9319596         83252.65        -115588.18
## [3,]          2658.179        0.9400075         68319.82        -120992.83
## [4,]          2388.647        0.9460906         57033.18        -125581.50
## [5,]          2058.754        0.9535359         43218.49        -131964.57
## [6,]          1800.204        0.9593712         32391.78        -137727.11
## [7,]          1563.272        0.9647185         22470.50        -143787.24
## [8,]          1404.654        0.9682983         15829.23        -148379.16
##      Diamond_model_Adj_r2
## [1,]            0.8489347
## [2,]            0.9319565
## [3,]            0.9400033
## [4,]            0.9460856
## [5,]            0.9535305
## [6,]            0.9593655
```

```
## [7,]           0.9647128
## [8,]           0.9682924
```

The results are also displayed in the form of a table to compare all the values for the subsets of all sizes.

**Stepwise regression summary**

```
summary_Diamond_forward

## Subset selection object
## Call: regsubsets.formula(Diamond_Train_pre$Log_price ~ Carat + +I(Carat^2)
+
##     factor(Cut) + factor(Color) + factor(Clarity) + Depth +
Top_width_ratio,
##     data = Diamond_Train_pre, method = "forward")
## 21 Variables  (and intercept)
##                   Forced in Forced out
## Carat                 FALSE      FALSE
## I(Carat^2)            FALSE      FALSE
## factor(Cut)2          FALSE      FALSE
## factor(Cut)3          FALSE      FALSE
## factor(Cut)4          FALSE      FALSE
## factor(Cut)5          FALSE      FALSE
## factor(Color)2        FALSE      FALSE
## factor(Color)3        FALSE      FALSE
## factor(Color)4        FALSE      FALSE
## factor(Color)5        FALSE      FALSE
## factor(Color)6        FALSE      FALSE
## factor(Color)7        FALSE      FALSE
## factor(Clarity)2      FALSE      FALSE
## factor(Clarity)3      FALSE      FALSE
## factor(Clarity)4      FALSE      FALSE
## factor(Clarity)5      FALSE      FALSE
## factor(Clarity)6      FALSE      FALSE
## factor(Clarity)7      FALSE      FALSE
## factor(Clarity)8      FALSE      FALSE
## Depth                 FALSE      FALSE
## Top_width_ratio       FALSE      FALSE
## 1 subsets of each size up to 8
## Selection Algorithm: forward
##          Carat I(Carat^2) factor(Cut)2 factor(Cut)3 factor(Cut)4
factor(Cut)5
## 1  ( 1 ) "*"   " "        " "          " "          " "          " "
## 2  ( 1 ) "*"   "*"        " "          " "          " "          " "
## 3  ( 1 ) "*"   "*"        " "          " "          " "          " "
## 4  ( 1 ) "*"   "*"        " "          " "          " "          " "
## 5  ( 1 ) "*"   "*"        " "          " "          " "          " "
## 6  ( 1 ) "*"   "*"        " "          " "          " "          " "
## 7  ( 1 ) "*"   "*"        " "          " "          " "          " "
## 8  ( 1 ) "*"   "*"        " "          " "          " "          " "
```

```
##          factor(Color)2 factor(Color)3 factor(Color)4 factor(Color)5
## 1  ( 1 ) " "            " "            " "            " "
## 2  ( 1 ) " "            " "            " "            " "
## 3  ( 1 ) " "            " "            " "            " "
## 4  ( 1 ) " "            " "            " "            " "
## 5  ( 1 ) " "            " "            " "            " "
## 6  ( 1 ) " "            " "            " "            " "
## 7  ( 1 ) " "            " "            " "            " "
## 8  ( 1 ) " "            " "            " "            "*"
##          factor(Color)6 factor(Color)7 factor(Clarity)2 factor(Clarity)3
## 1  ( 1 ) " "            " "            " "              " "
## 2  ( 1 ) " "            " "            " "              " "
## 3  ( 1 ) " "            " "            " "              " "
## 4  ( 1 ) " "            " "            " "              " "
## 5  ( 1 ) " "            " "            " "              " "
## 6  ( 1 ) " "            "*"            " "              " "
## 7  ( 1 ) "*"            "*"            " "              " "
## 8  ( 1 ) "*"            "*"            " "              " "
##          factor(Clarity)4 factor(Clarity)5 factor(Clarity)6
factor(Clarity)7
## 1  ( 1 ) " "              " "              " "              " "
## 2  ( 1 ) " "              " "              " "              " "
## 3  ( 1 ) " "              " "              " "              "*"
## 4  ( 1 ) " "              " "              " "              "*"
## 5  ( 1 ) " "              " "              "*"              "*"
## 6  ( 1 ) " "              " "              "*"              "*"
## 7  ( 1 ) " "              " "              "*"              "*"
## 8  ( 1 ) " "              " "              "*"              "*"
##          factor(Clarity)8 Depth Top_width_ratio
## 1  ( 1 ) " "              " "   " "
## 2  ( 1 ) " "              " "   " "
## 3  ( 1 ) " "              " "   " "
## 4  ( 1 ) "*"              " "   " "
## 5  ( 1 ) "*"              " "   " "
## 6  ( 1 ) "*"              " "   " "
## 7  ( 1 ) "*"              " "   " "
## 8  ( 1 ) "*"              " "   " "
```

```
summary_Diamond_backward
```

```
## Subset selection object
## Call: regsubsets.formula(Diamond_Train_pre$Log_price ~ Carat + +I(Carat^2)
+
##     factor(Cut) + factor(Color) + factor(Clarity) + Depth +
Top_width_ratio,
##     data = Diamond_Train_pre, method = "backward")
## 21 Variables  (and intercept)
##                   Forced in Forced out
## Carat                 FALSE      FALSE
## I(Carat^2)            FALSE      FALSE
```

```
## factor(Cut)2         FALSE      FALSE
## factor(Cut)3         FALSE      FALSE
## factor(Cut)4         FALSE      FALSE
## factor(Cut)5         FALSE      FALSE
## factor(Color)2       FALSE      FALSE
## factor(Color)3       FALSE      FALSE
## factor(Color)4       FALSE      FALSE
## factor(Color)5       FALSE      FALSE
## factor(Color)6       FALSE      FALSE
## factor(Color)7       FALSE      FALSE
## factor(Clarity)2     FALSE      FALSE
## factor(Clarity)3     FALSE      FALSE
## factor(Clarity)4     FALSE      FALSE
## factor(Clarity)5     FALSE      FALSE
## factor(Clarity)6     FALSE      FALSE
## factor(Clarity)7     FALSE      FALSE
## factor(Clarity)8     FALSE      FALSE
## Depth                FALSE      FALSE
## Top_width_ratio      FALSE      FALSE
## 1 subsets of each size up to 8
## Selection Algorithm: backward
##          Carat I(Carat^2) factor(Cut)2 factor(Cut)3 factor(Cut)4
factor(Cut)5
## 1  ( 1 ) "*"   " "        " "          " "          " "          " "
## 2  ( 1 ) "*"   "*"        " "          " "          " "          " "
## 3  ( 1 ) "*"   "*"        " "          " "          " "          " "
## 4  ( 1 ) "*"   "*"        " "          " "          " "          " "
## 5  ( 1 ) "*"   "*"        " "          " "          " "          " "
## 6  ( 1 ) "*"   "*"        " "          " "          " "          " "
## 7  ( 1 ) "*"   "*"        " "          " "          " "          " "
## 8  ( 1 ) "*"   "*"        " "          " "          " "          " "
##          factor(Color)2 factor(Color)3 factor(Color)4 factor(Color)5
## 1  ( 1 ) " "            " "            " "            " "
## 2  ( 1 ) " "            " "            " "            " "
## 3  ( 1 ) " "            " "            " "            " "
## 4  ( 1 ) " "            " "            " "            " "
## 5  ( 1 ) " "            " "            " "            " "
## 6  ( 1 ) " "            " "            " "            " "
## 7  ( 1 ) " "            " "            " "            " "
## 8  ( 1 ) " "            " "            " "            "*"
##          factor(Color)6 factor(Color)7 factor(Clarity)2 factor(Clarity)3
## 1  ( 1 ) " "            " "            " "              " "
## 2  ( 1 ) " "            " "            " "              " "
## 3  ( 1 ) " "            " "            " "              " "
## 4  ( 1 ) " "            " "            " "              " "
## 5  ( 1 ) " "            " "            " "              " "
## 6  ( 1 ) " "            "*"            " "              " "
## 7  ( 1 ) "*"            "*"            " "              " "
## 8  ( 1 ) "*"            "*"            " "              " "
##          factor(Clarity)4 factor(Clarity)5 factor(Clarity)6
```

```
factor(Clarity)7
## 1  ( 1 ) " "              " "              " "              " "
## 2  ( 1 ) " "              " "              " "              " "
## 3  ( 1 ) " "              " "              " "              "*"
## 4  ( 1 ) " "              " "              " "              "*"
## 5  ( 1 ) " "              " "              "*"              "*"
## 6  ( 1 ) " "              " "              "*"              "*"
## 7  ( 1 ) " "              " "              "*"              "*"
## 8  ( 1 ) " "              " "              "*"              "*"
##          factor(Clarity)8 Depth Top_width_ratio
## 1  ( 1 ) " "              " "   " "
## 2  ( 1 ) " "              " "   " "
## 3  ( 1 ) " "              " "   " "
## 4  ( 1 ) "*"              " "   " "
## 5  ( 1 ) "*"              " "   " "
## 6  ( 1 ) "*"              " "   " "
## 7  ( 1 ) "*"              " "   " "
## 8  ( 1 ) "*"              " "   " "
```

```
summary_Diamond_seqrep
```

```
## Subset selection object
## Call: regsubsets.formula(Diamond_Train_pre$Log_price ~ Carat + I(Carat^2)
+
##     factor(Cut) + factor(Color) + factor(Clarity) + Depth +
Top_width_ratio,
##     data = Diamond_Train_pre, method = "seqrep")
## 21 Variables  (and intercept)
##                   Forced in Forced out
## Carat                 FALSE      FALSE
## I(Carat^2)            FALSE      FALSE
## factor(Cut)2          FALSE      FALSE
## factor(Cut)3          FALSE      FALSE
## factor(Cut)4          FALSE      FALSE
## factor(Cut)5          FALSE      FALSE
## factor(Color)2        FALSE      FALSE
## factor(Color)3        FALSE      FALSE
## factor(Color)4        FALSE      FALSE
## factor(Color)5        FALSE      FALSE
## factor(Color)6        FALSE      FALSE
## factor(Color)7        FALSE      FALSE
## factor(Clarity)2      FALSE      FALSE
## factor(Clarity)3      FALSE      FALSE
## factor(Clarity)4      FALSE      FALSE
## factor(Clarity)5      FALSE      FALSE
## factor(Clarity)6      FALSE      FALSE
## factor(Clarity)7      FALSE      FALSE
## factor(Clarity)8      FALSE      FALSE
## Depth                 FALSE      FALSE
## Top_width_ratio       FALSE      FALSE
```

```
## 1 subsets of each size up to 8
## Selection Algorithm: 'sequential replacement'
##          Carat I(Carat^2) factor(Cut)2 factor(Cut)3 factor(Cut)4
factor(Cut)5
## 1  ( 1 ) "*"   " "        " "          " "          " "          " "
## 2  ( 1 ) "*"   "*"        " "          " "          " "          " "
## 3  ( 1 ) "*"   "*"        " "          " "          " "          " "
## 4  ( 1 ) "*"   "*"        " "          " "          " "          " "
## 5  ( 1 ) "*"   "*"        " "          " "          " "          " "
## 6  ( 1 ) "*"   "*"        " "          " "          " "          " "
## 7  ( 1 ) "*"   "*"        " "          " "          " "          " "
## 8  ( 1 ) "*"   "*"        " "          " "          " "          " "
##          factor(Color)2 factor(Color)3 factor(Color)4 factor(Color)5
## 1  ( 1 ) " "            " "            " "            " "
## 2  ( 1 ) " "            " "            " "            " "
## 3  ( 1 ) " "            " "            " "            " "
## 4  ( 1 ) " "            " "            " "            " "
## 5  ( 1 ) " "            " "            " "            " "
## 6  ( 1 ) " "            " "            " "            " "
## 7  ( 1 ) " "            " "            " "            " "
## 8  ( 1 ) " "            " "            " "            "*"
##          factor(Color)6 factor(Color)7 factor(Clarity)2 factor(Clarity)3
## 1  ( 1 ) " "            " "            " "              " "
## 2  ( 1 ) " "            " "            " "              " "
## 3  ( 1 ) " "            " "            " "              " "
## 4  ( 1 ) " "            " "            " "              " "
## 5  ( 1 ) " "            " "            " "              " "
## 6  ( 1 ) " "            "*"            " "              " "
## 7  ( 1 ) "*"            "*"            " "              " "
## 8  ( 1 ) "*"            "*"            " "              " "
##          factor(Clarity)4 factor(Clarity)5 factor(Clarity)6
factor(Clarity)7
## 1  ( 1 ) " "              " "              " "              " "
## 2  ( 1 ) " "              " "              " "              " "
## 3  ( 1 ) " "              " "              " "              "*"
## 4  ( 1 ) " "              " "              " "              "*"
## 5  ( 1 ) " "              " "              "*"              "*"
## 6  ( 1 ) " "              " "              "*"              "*"
## 7  ( 1 ) " "              " "              "*"              "*"
## 8  ( 1 ) " "              " "              "*"              "*"
##          factor(Clarity)8 Depth Top_width_ratio
## 1  ( 1 ) " "              " "   " "
## 2  ( 1 ) " "              " "   " "
## 3  ( 1 ) " "              " "   " "
## 4  ( 1 ) "*"              " "   " "
## 5  ( 1 ) "*"              " "   " "
## 6  ( 1 ) "*"              " "   " "
## 7  ( 1 ) "*"              " "   " "
## 8  ( 1 ) "*"              " "   " "
```