Total = 50/50 (Every part has been handled well as the Prof said in class. Even tried different transformations to try and modify the results)

Homework-3

Dancun Juma

2025-01-29

Table of Contents

```r
# Load necessary libraries
library(tidymodels)
library(dplyr)
library(ggplot2)
library(broom)
library(caret)
library(flextable)
library(GGally)
```

Loading the dataset

```r
# Load the Diamonds dataset
data("diamonds")
set.seed(1995)
diamond_subset <- diamonds %>%
  sample_n(1000)

head(diamond_subset)

## # A tibble: 6 × 10
##   carat cut     color clarity depth table price     x     y     z
##   <dbl> <ord>   <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
## 1  0.7  Ideal   F     SI1      62.2    55  2697  5.73  5.68  3.55
## 2  1.35 Premium H     VS2      60.5    62  6093  7.16  7.11  4.32
## 3  0.81 Premium J     SI1      61.4    60  1917  5.98  5.94  3.66
## 4  0.43 Ideal   D     VS2      62      55  1056  4.81  4.86  3
## 5  0.3  Premium G     VS1      61.9    59   776  4.32  4.28  2.66
## 6  0.38 Premium E     VS2      61.8    58   866  4.61  4.64  2.86
```

Variables descriptions

```r
# Create a data frame with variable descriptions
variable_description <- data.frame(
  Variable = c("carat", "cut", "color", "clarity", "depth", "table", "price",
"x", "y", "z"),
  Type = c(
    "Numeric (dbl)",
    "Ordinal",
    "Ordinal",
    "Ordinal",
    "Numeric (dbl)",
    "Numeric (dbl)",
    "Integer",
    "Numeric (dbl)",
    "Numeric (dbl)",
    "Numeric (dbl)"
  ),
  Description = c(
    "Weight of the diamond in carats",
    "Quality of the cut (Fair, Good, Very Good, Premium, Ideal)",
    "Diamond color, from J (worst) to D (best)",
    "Clarity of the diamond (IF, VVS1, VVS2, VS1, VS2, SI1, SI2, I1)",
    "Total depth percentage = (z / mean(x, y)) * 100",
    "Width of the top of the diamond's table as a percentage of the
diameter",
    "Price of the diamond in US dollars",
    "Length of the diamond (in mm)",
    "Width of the diamond (in mm)",
    "Depth of the diamond (in mm)"
  )
)

# Create a flextable for the variable description
flextable_description <- flextable(variable_description) %>%
  set_header_labels(
    Variable = "Variable Name",
    Type = "Data Type",
    Description = "Description"
  ) %>%
  autofit() %>%
  theme_vanilla()

# Print the flextable
flextable_description
```

Variable Name	Data Type	Description
carat	Numeric (dbl)	Weight of the diamond in carats

Variable Name	Data Type	Description
cut	Ordinal	Quality of the cut (Fair, Good, Very Good, Premium, Ideal)
color	Ordinal	Diamond color, from J (worst) to D (best)
clarity	Ordinal	Clarity of the diamond (IF, VVS1, VVS2, VS1, VS2, SI1, SI2, I1)
depth	Numeric (dbl)	Total depth percentage = (z / mean(x, y)) * 100
table	Numeric (dbl)	Width of the top of the diamond's table as a percentage of the diameter
price	Integer	Price of the diamond in US dollars
x	Numeric (dbl)	Length of the diamond (in mm)
y	Numeric (dbl)	Width of the diamond (in mm)
z	Numeric (dbl)	Depth of the diamond (in mm)

1) Exploration and Model Identification (Good in clarification, Exxecuting, 10/10)

Exploration of the Data Structure

```
# Explore the data structure
glimpse(diamond_subset)

## Rows: 1,000
## Columns: 10
## $ carat   <dbl> 0.70, 1.35, 0.81, 0.43, 0.30, 0.38, 1.00, 0.90, 0.70,
0.50, 1.…
## $ cut     <ord> Ideal, Premium, Premium, Ideal, Premium, Premium, Premium,
Ver…
## $ color   <ord> F, H, J, D, G, E, G, H, G, E, I, G, F, H, G, D, H, H, G,
F, E,…
## $ clarity <ord> SI1, VS2, SI1, VS2, VS1, VS2, SI2, SI1, SI2, SI2, VS2,
VS2, VS…
## $ depth   <dbl> 62.2, 60.5, 61.4, 62.0, 61.9, 61.8, 62.2, 58.7, 60.7,
63.2, 62…
## $ table   <dbl> 55, 62, 60, 55, 59, 58, 59, 59, 58, 61, 56, 54, 55, 59,
56, 57…
## $ price   <int> 2697, 6093, 1917, 1056, 776, 866, 4480, 3447, 2141, 1097,
5120…
## $ x       <dbl> 5.73, 7.16, 5.98, 4.81, 4.32, 4.61, 6.39, 6.30, 5.74,
5.05, 6.…
## $ y       <dbl> 5.68, 7.11, 5.94, 4.86, 4.28, 4.64, 6.35, 6.35, 5.80,
5.02, 6.…
```

```
## $ z       <dbl> 3.55, 4.32, 3.66, 3.00, 2.66, 2.86, 3.96, 3.71, 3.50,
3.18, 4.…
```

As can be observed above, the data contains 1000 rows and 10 columns. Key variables include `carat` (numeric, weight of the diamond), `cut` (ordinal, quality rating), `color` (ordinal, graded from J to D), and `clarity` (ordinal, describing diamond inclusions). Other quantitative attributes include `depth` (numeric, percentage), `table` (numeric, table width percentage), and `price` (integer, USD). Dimensions are captured by `x`, `y`, and `z`, representing length, width, and depth in millimeters. The data provides a rich mix of numeric and ordinal variables making it well-suited for statistical modeling, including regression analysis and exploring relationships between diamond features and price.

Summary statistics

```
# Summary statistics of the dataset
summary(diamond_subset)
```

```
##      carat               cut       color      clarity       depth      
##  Min.   :0.230   Fair     : 36   D:140   VS2    :250   Min.   :55.9  
##  1st Qu.:0.380   Good     :101   E:199   SI1    :223   1st Qu.:61.1  
##  Median :0.700   Very Good:213   F:167   SI2    :182   Median :61.9  
##  Mean   :0.785   Premium  :249   G:205   VS1    :141   Mean   :61.8  
##  3rd Qu.:1.030   Ideal    :401   H:144   VVS2   : 81   3rd Qu.:62.5  
##  Max.   :3.670                   I:101   VVS1   : 75   Max.   :67.8  
##                                  J: 44   (Other): 48                 
##      table           price               x               y        
##  Min.   :52.00   Min.   :  357.0   Min.   :3.870   Min.   :3.900  
##  1st Qu.:56.00   1st Qu.:  898.8   1st Qu.:4.650   1st Qu.:4.678  
##  Median :57.00   Median : 2191.5   Median :5.635   Median :5.635  
##  Mean   :57.45   Mean   : 3810.5   Mean   :5.686   Mean   :5.689  
##  3rd Qu.:59.00   3rd Qu.: 5306.0   3rd Qu.:6.532   3rd Qu.:6.530  
##  Max.   :66.00   Max.   :18532.0   Max.   :9.860   Max.   :9.810  
##                                                                   
##        z        
##  Min.   :2.420  
##  1st Qu.:2.870  
##  Median :3.465  
##  Mean   :3.515  
##  3rd Qu.:4.030  
##  Max.   :6.130  
## 
```

carat: Diamond weight, ranging from 0.23 to 3.67 carats (Mean: 0.785). Larger carats indicate heavier diamonds.

cut: Quality of cut, most common is “Ideal” (401), with fewer “Fair” (36).

color: Color grade, ranging from best (D) to worst (J), with E being most frequent.

clarity: Diamond clarity; common grades are VS2 (250) and SI1 (223), showing inclusion levels.

depth: Depth percentage (55.9–67.8, Mean: 61.8), indicating diamond proportion.

table: Table width percentage (52–66, Mean: 57.45), reflecting diamond's flat top area.

price: Diamond cost in USD, varying widely ($357–$18,532, Mean: $3,810.5).

x: Diamond length (3.87–9.86 mm, Mean: 5.69).

y: Diamond width (3.90–9.81 mm, Mean: 5.69).

z: Diamond depth (2.42–6.13 mm, Mean: 3.51).

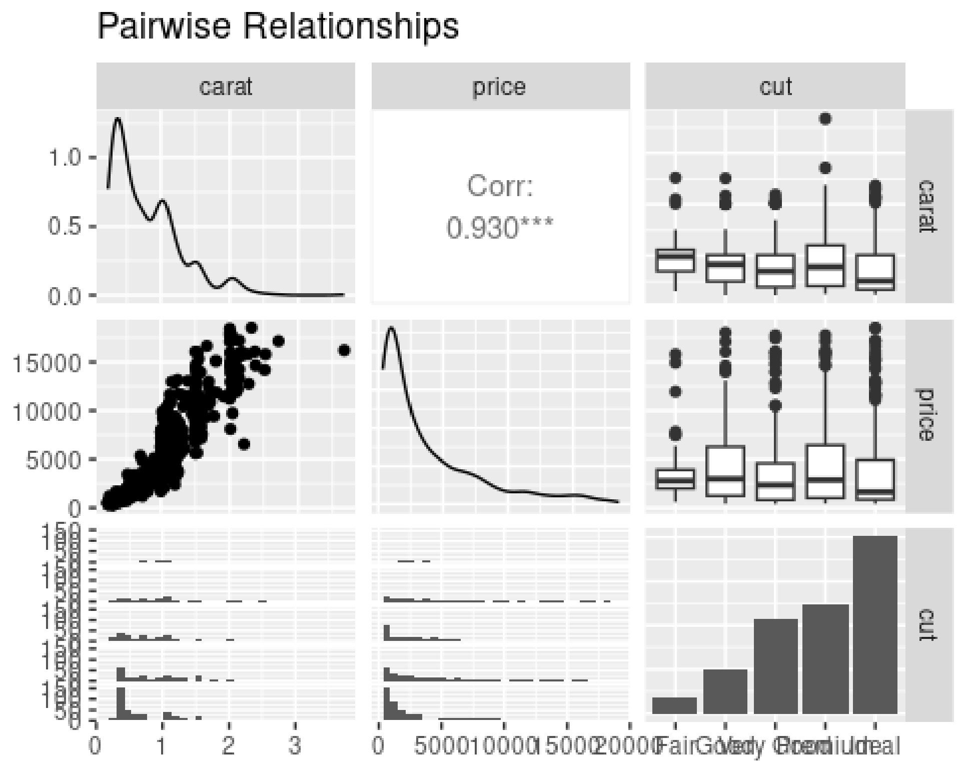
Checking missing values

```r
# Check for missing values
missing_values <- sum(is.na(diamond_subset))
```

There are 0 missing values in the dataset.

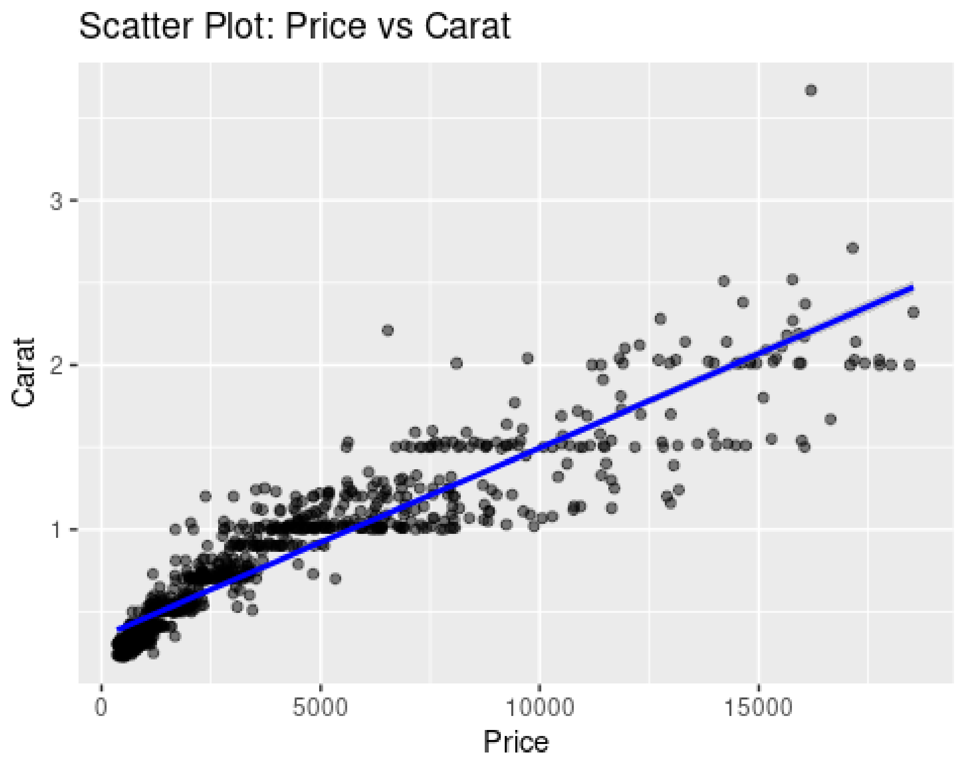
Visualize relationships

```r
# Now we create a ggpairs plot with 'carat' as the first column
ggpairs_plot <- ggpairs(
  diamond_subset,
  columns = c("carat", "price", "cut"),
  title = "Pairwise Relationships"
)

# show the plot
print(ggpairs_plot)
```



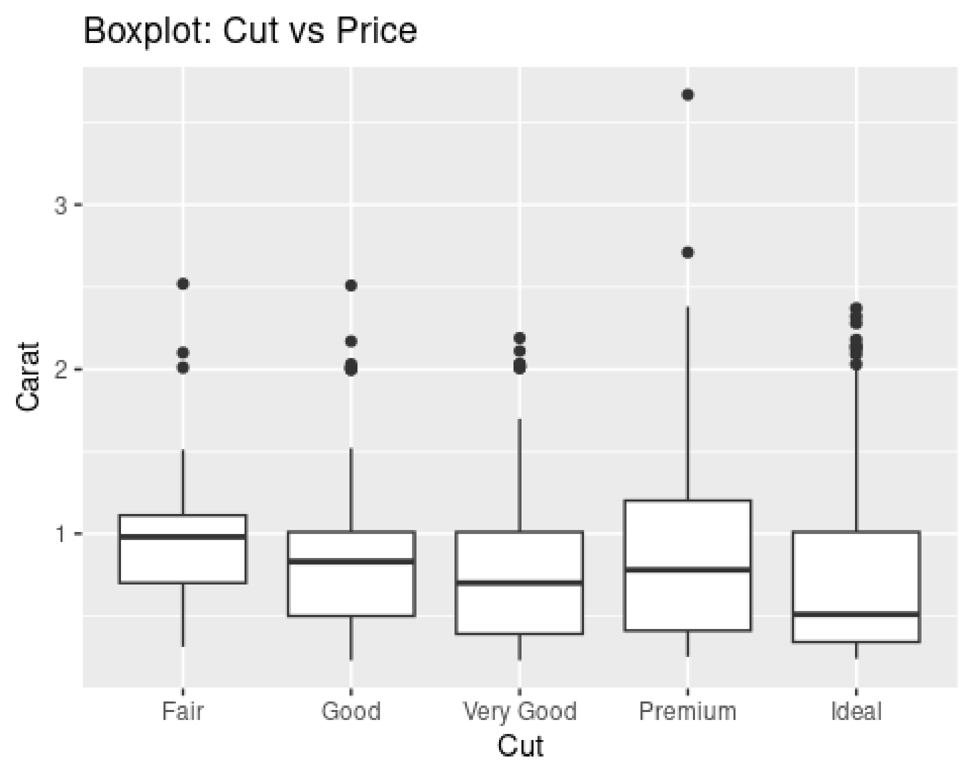
The scatter plot above between price and carat indicates a strong positive correlation (0.930), suggesting that as carat increases, price also rises. The density plots on the diagonal provide insights into data distribution, with price showing a right-skewed distribution and carat having a similar skew. The box plots illustrate how price and carat vary across different cut categories, with "Ideal" being the most frequent cut type. The spread in prices is highest for lower-quality cuts, while carat remains relatively stable across cuts. This visualization helps identify trends and potential non-linear relationships in the data, emphasizing that carat is a dominant factor influencing price, while cut has a less pronounced effect.

```r
# Scatter plot for carat (dependent) vs price (quantitative predictor)
ggplot(diamond_subset, aes(x = price, y = carat)) +
  geom_point(alpha = 0.5) +
  geom_smooth(method = "lm", color = "blue") +
  labs(title = "Scatter Plot: Price vs Carat",
       x = "Price",
       y = "Carat")
```



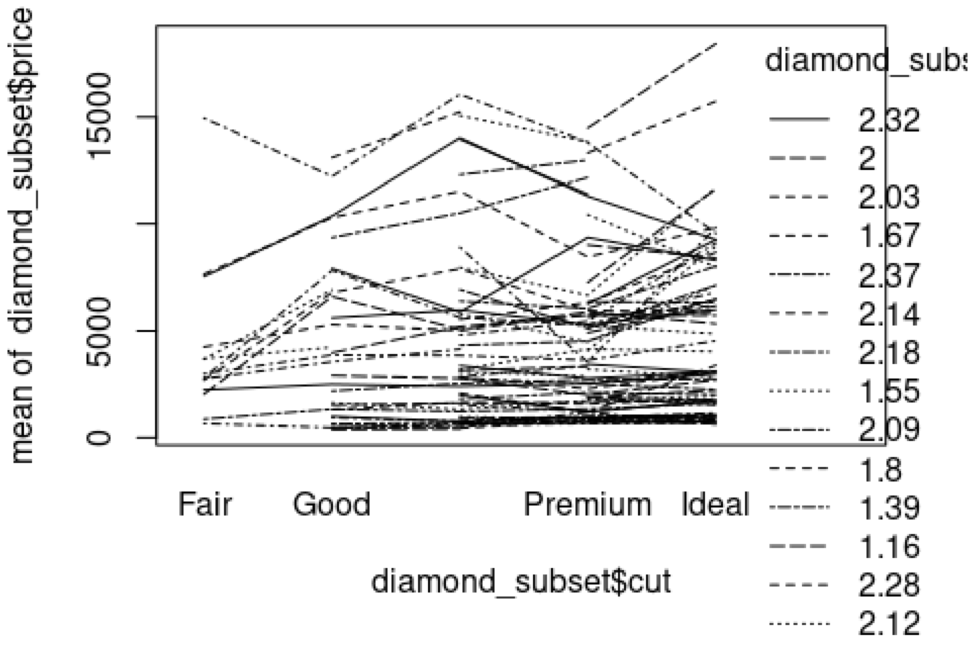
The scatter plot visualizes the relationship between `carat` (diamond weight) and `price`. The positive trend suggests that as carat increases, price also increases. The relationship appears nonlinear, with price rising more steeply for larger carats. There's greater price variability among larger diamonds, likely influenced by other factors like `cut`, `clarity`, and `color`. The clustering of points at lower carat values suggests that smaller diamonds are more common. Some high-priced outliers indicate premium diamonds with exceptional quality attributes.

```
# Boxplot for carat (dependent) vs cut (qualitative predictor)
ggplot(diamond_subset, aes(x = cut, y = carat)) +
  geom_boxplot() +
  labs(title = "Boxplot: Cut vs Price", x = "Cut", y = "Carat")
```



It can be observed that Premium and Ideal cuts have higher variability, while Fair, Good, and Very Good cuts are more consistent. Outliers, especially in Ideal and Premium cuts, indicate larger diamonds. Medians are similar across categories, but higher-quality cuts exhibit a broader range of carat sizes.

```
# Interaction plot: Cut and carat
interaction.plot(diamond_subset$cut, diamond_subset$carat,
diamond_subset$price, fun = mean)
```



Based on the plot, each line represents a unique carat value and we can observe that price varies significantly across cuts, with higher-quality cuts (for example "Ideal") generally commanding higher prices. However, the slopes and overlaps suggest the effect of carat weight on price differs by cut quality. Some lines intersect, indicating non-linear interactions. This highlights the importance of considering both predictors and their interaction to accurately model diamond prices.

2) Model Assessment(10/10)

```r
# Define the multiple linear regression model specification
lm_spec <- linear_reg() %>%
  set_mode("regression") %>%
  set_engine("lm")

# Fit the model with both quantitative (carat) and qualitative (cut)
predictors, including their interaction
full_model <- lm_spec %>%
  fit(carat ~ price + cut + price:cut, data = diamond_subset)

# Extract and format the tidy output of the model
tidy_full <- tidy(full_model) %>%
  mutate(p.value = formatC(p.value, format = "f", digits = 4))

# Print the model output
print(tidy_full)
```

```
## # A tibble: 10 × 5
##    term              estimate  std.error statistic p.value
##    <chr>                <dbl>      <dbl>     <dbl> <chr>
##  1 (Intercept)   0.390       0.0110        35.6   0.0000
##  2 price         0.000115    0.00000200    57.7   0.0000
##  3 cut.L        -0.143       0.0290        -4.91  0.0000
##  4 cut.Q         0.0625      0.0260         2.40  0.0164
##  5 cut.C        -0.0553      0.0230        -2.40  0.0164
##  6 cut^4         0.0124      0.0188         0.657 0.5113
##  7 price:cut.L -0.00000557  0.00000538    -1.04  0.3004
##  8 price:cut.Q  0.000000584 0.00000484     0.121 0.9040
##  9 price:cut.C -0.00000699  0.00000403    -1.74  0.0829
## 10 price:cut^4 -0.00000626  0.00000337    -1.86  0.0632
```

$$
\begin{aligned}
\widehat{carat} \\
= 0.3895 + 0.0001153 \times price - 0.1425 \times cut.L + 0.06246 \times cut.Q - 0.05533 \times cut.C \\
+ 0.01236 \times cut^4 - 0.000005575 \times price \times cut.L + 0.0000005836 \times price \times cut.Q \\
- 0.000006992 \times price \times cut.C - 0.000006263 \times price \times cut^4
\end{aligned}
$$

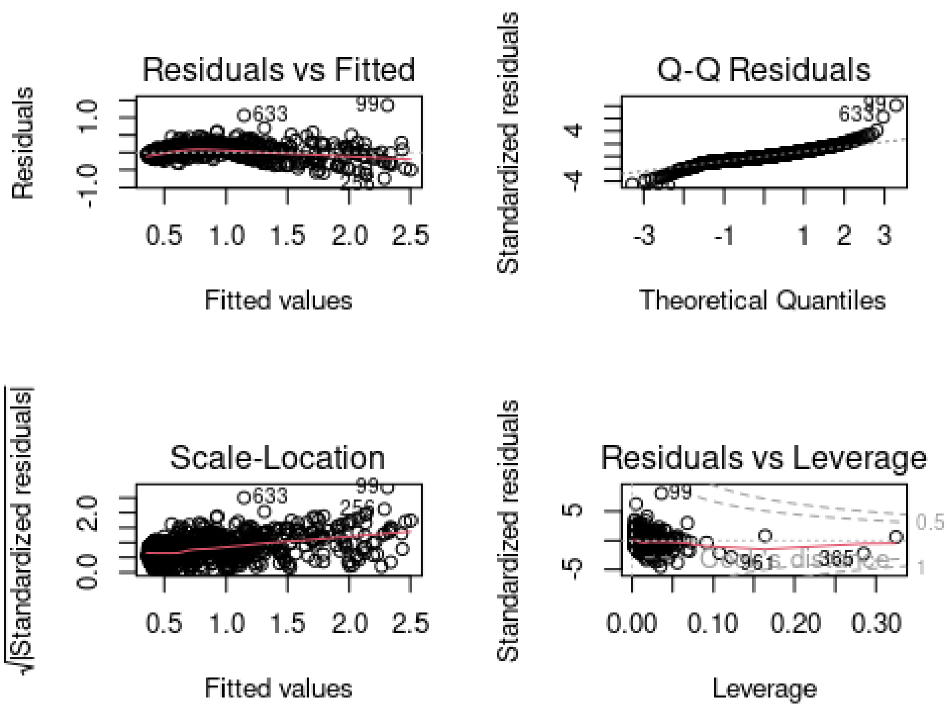
```r
glance(full_model)
```

```
## # A tibble: 1 × 12
##   r.squared adj.r.squared sigma statistic p.value    df logLik   AIC   BIC
##       <dbl>         <dbl> <dbl>     <dbl>   <dbl> <dbl>  <dbl> <dbl> <dbl>
## 1     0.880         0.879 0.170      804.       0     9   356. -690. -636.
## # ℹ 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>
```

88% of the variations (in response variable) could be explained by the predictors

Residual plot

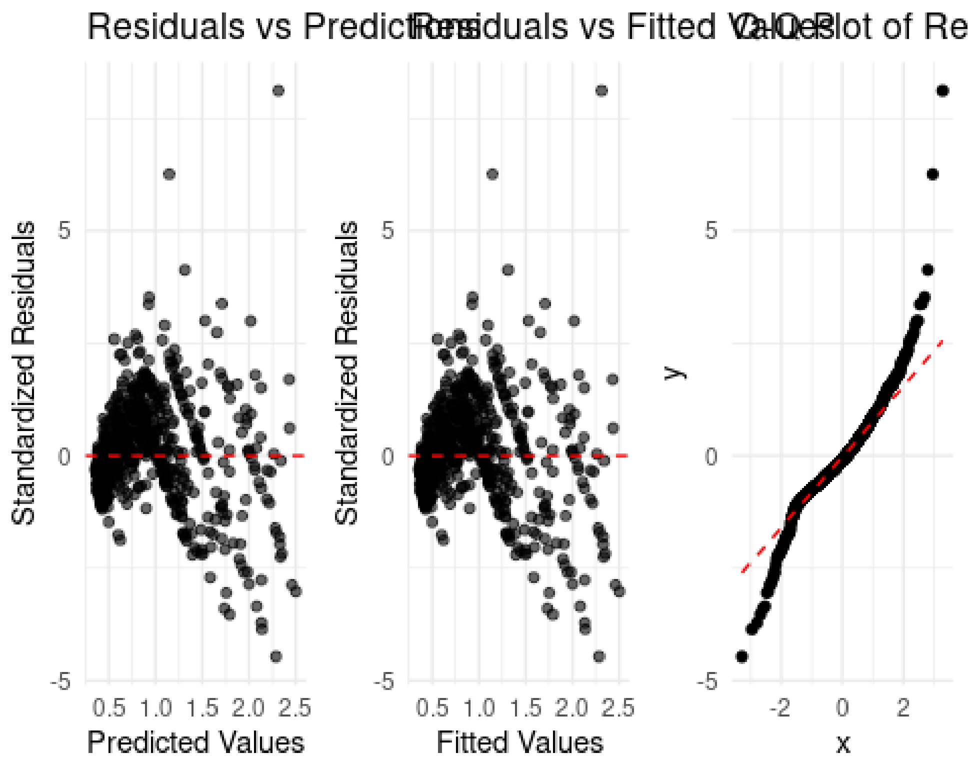
```r
# Extract the lm object from the tidymodels fitted model
lm_model <- full_model$fit

# Base R diagnostic plots for the extracted lm model
par(mfrow = c(2, 2))
plot(lm_model)
```



Based on the plot above on, the Residuals vs. Fitted plot shows non-random patterns, indicating potential non-linearity. The Q-Q plot reveals deviations from normality, especially in the tails. The Scale-Location plot suggests heteroscedasticity, as variance increases with fitted values. The Residuals vs. Leverage plot highlights influential points, with some exceeding Cook's distance, suggesting they significantly impact the model. Model adjustments may be needed for better performance.

Residual Analysis

```r
# Define the residual analysis function
residual_analysis <- function(model = NULL) {
  # Ensure the input is a valid model
  if ("model_fit" %in% class(model)) {
    model <- model$fit
  }
  if (!inherits(model, "lm")) {
    stop("The provided model is not an lm object.")
  }

  # Create a data frame for residual analysis
  df <- data.frame(
    Prediction = predict(model),
    Residual = rstandard(model),
    Fitted = fitted(model),
    Observed = model$model[[1]] # Extract observed values
  )
  # Create diagnostic plots
  p1 <- ggplot(df, aes(x = Prediction, y = Residual)) +
    geom_point(alpha = 0.6) +
    geom_hline(yintercept = 0, color = "red", linetype = "dashed") +
    labs(title = "Residuals vs Predictions", x = "Predicted Values", y =
"Standardized Residuals") +
    theme_minimal()

  p2 <- ggplot(df, aes(x = Fitted, y = Residual)) +
    geom_point(alpha = 0.6) +
    geom_hline(yintercept = 0, color = "red", linetype = "dashed") +
    labs(title = "Residuals vs Fitted Values", x = "Fitted Values", y =
"Standardized Residuals") +
    theme_minimal()

  p3 <- ggplot(df, aes(sample = Residual)) +
    stat_qq() +
    stat_qq_line(color = "red", linetype = "dashed") +
    labs(title = "Q-Q Plot of Residuals") +
    theme_minimal()

  # Combine plots using gridExtra
  gridExtra::grid.arrange(p1, p2, p3, ncol = 3)
}

residual_analysis(full_model)
```



Based on the plot above, the "Residuals vs. Predictions" and "Residuals vs. Fitted Values" plots reveal a funnel shape, suggesting heteroscedasticity, where variance increases with predicted values. This indicates the model may not adequately capture variability. The Q-Q plot of residuals shows deviations from normality, particularly in the tails, implying potential issues with normality assumptions. The presence of outliers and non-random residual dispersion suggests the model may benefit from transformation or additional predictor variables. Addressing these issues could improve model performance and predictive accuracy, possibly through transformations or alternative modeling approaches like generalized linear models.

3) Accuracy of the Model(10/10)

```
# Split data into training and test sets
set.seed(12345)
data_split <- initial_split(diamond_subset, prop = 0.8)
train_data <- training(data_split)
test_data <- testing(data_split)

# Fit the model on training data
train_model <- lm_spec %>%
  fit(carat ~ price + cut + price:cut, data = train_data)

# Predict on test data
test_predictions <- predict(train_model, new_data = test_data) %>%
  bind_cols(test_data)

# Compute accuracy metrics
metrics <- metric_set(rmse, rsq)
accuracy_results <- metrics(test_predictions, truth = carat, estimate =
.pred)

# Print accuracy results
print(accuracy_results)

## # A tibble: 2 × 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard       0.163
## 2 rsq     standard       0.886
```

The model's RMSE of 0.1633 indicates the better prediction error, with lower values indicating better fit. The R-squared value of 0.8863 suggests the model explains 89% of the variance in carat, demonstrating a strong relationship between the predictors and the target variable. Overall, the model shows good performance with minimal error and high explanatory power.

4) Predictions of New Observations(10/10)

```
# Create new observation(s)
new_data <- data.frame(
  price = c(10010, 7001),
  cut = c("Ideal", "Fair")
)

# Predict price for the new data
new_predictions <- predict(full_model, new_data = new_data, type = "numeric")

# Add predictions to the new_data dataframe
new_data <- new_data %>%
  mutate(predicted_carat = new_predictions$.pred)

# Print predictions
print(new_data)

##   price   cut predicted_carat
## 1 10010 Ideal        1.409073
## 2  7001  Fair        1.376328
```

A diamond worth $10010 and Ideal cut is predicted to have a carat of 1.4091.

A diamond with $7001 and Fair cut is predicted to have a carat of 1.3763.

CI and Predictions

```
result <- cbind(
  predict(full_model, new_data = diamond_subset),
  predict(full_model, new_data = diamond_subset, type = "conf_int"),
  predict(full_model, new_data = diamond_subset, type = "pred_int")
)

head(result, 5)

##       .pred .pred_lower .pred_upper .pred_lower .pred_upper
## 1 0.6110695   0.5940823   0.6280566  0.27632891   0.9458100
## 2 1.0928475   1.0700271   1.1156678  0.75776022   1.4279347
## 3 0.5884208   0.5636472   0.6131944  0.25319486   0.9236467
## 4 0.4320015   0.4124410   0.4515619  0.09712048   0.7668825
## 5 0.4505973   0.4223611   0.4788335  0.11509772   0.7860969
```

Narrower intervals indicate higher precision, while wider intervals suggest greater uncertainty. In general, the model provides reasonable estimates, but variability in prediction intervals suggests room for refinement.

5) Interpretation of Coefficients(10/10)

```
# Interpret coefficients of the model
tidy_full %>%
```

```r
  mutate(
    significance = case_when(
      p.value < 0.001 ~ "***",
      p.value < 0.01 ~ "**",
      p.value < 0.05 ~ "*",
      TRUE ~ ""
    )
  ) %>%
  select(term, estimate, std.error, p.value, significance) %>%
  print()
```

```
## # A tibble: 10 × 5
##    term            estimate  std.error p.value significance
##    <chr>              <dbl>      <dbl> <chr>   <chr>
##  1 (Intercept)  0.390       0.0110     0.0000  "***"
##  2 price        0.000115    0.00000200 0.0000  "***"
##  3 cut.L       -0.143       0.0290     0.0000  "***"
##  4 cut.Q        0.0625      0.0260     0.0164  "*"
##  5 cut.C       -0.0553      0.0230     0.0164  "*"
##  6 cut^4        0.0124      0.0188     0.5113  ""
##  7 price:cut.L -0.00000557  0.00000538 0.3004  ""
##  8 price:cut.Q  0.000000584 0.00000484 0.9040  ""
##  9 price:cut.C -0.00000699  0.00000403 0.0829  ""
## 10 price:cut^4 -0.00000626  0.00000337 0.0632  ""
```

Intercept (0.3895, $p < 0.001$): This represents the estimated carat weight when all predictor variables (price and cut contrasts) are zero. While not directly meaningful due to categorical variables, it serves as a baseline reference.

Price (0.0001153, $p < 0.001$): A one-unit increase in price increases the predicted carat by 0.0001153, indicating a strong positive relationship between price and carat size, as expected.

Cut.L (-0.1425, $p < 0.001$): The linear contrast for cut shows that higher-quality cuts tend to have slightly smaller carat weights, likely due to better proportions optimizing brilliance rather than raw size.

Cut.Q (0.06246, $p = 0.0164$): The quadratic contrast suggests a non-linear effect of cut on carat, indicating that mid-tier cuts may have slightly higher carat sizes compared to extremes (best or worst cuts).

Cut.C (-0.05533, $p = 0.0164$): This cubic contrast captures complex patterns in how cut affects carat. A negative value suggests that certain intermediate cut grades may have slightly lower carat sizes.

Cut^4 (0.01236, $p = 0.5113$, ns): The fourth-degree contrast is not statistically significant, indicating that additional complexity in cut's effect on carat is unnecessary.

Price:Cut.L (-0.000005575, p = 0.3004, ns): This interaction term suggests that the effect of price on carat differs across cut qualities, but the lack of significance means the effect is weak.

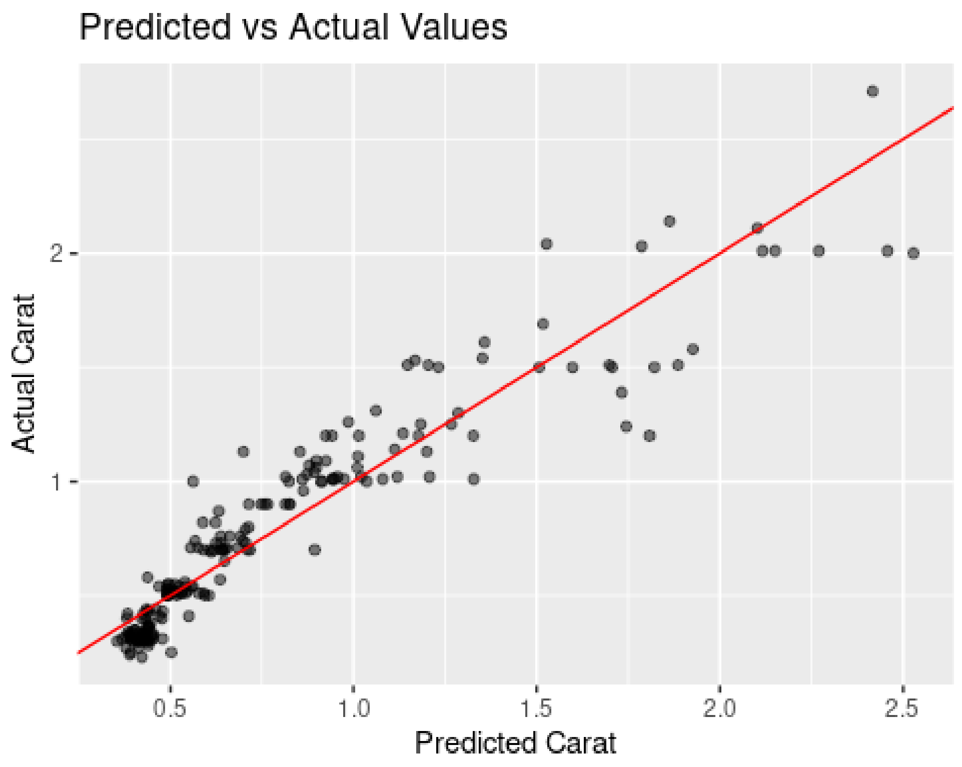
Price:Cut.Q (0.0000005836, p = 0.9040, ns): The interaction between price and quadratic cut contrast is also non-significant, indicating that price's effect on carat doesn't meaningfully vary across mid-tier cut levels.

Price:Cut.C (-0.000006992, p = 0.0829): The cubic interaction term is marginally significant, suggesting price influences carat differently at specific cut levels, but the effect is small.

Price:Cut^4 (-0.000006263, p = 0.0632): The fourth-degree interaction term is nearly significant, implying minor non-linear effects of price across cut categories.

BONUS - Visualization of Predictions

```
# Scatter plot of predicted vs actual values for test data
ggplot(test_predictions, aes(x = .pred, y = carat)) +
  geom_point(alpha = 0.5) +
  geom_abline(slope = 1, intercept = 0, color = "red") +
  labs(title = "Predicted vs Actual Values", x = "Predicted Carat", y =
"Actual Carat")
```



The plot shows predictions perfectly match actual values. The clustering of points around this line indicates that the model performs well in predicting carat weight.

However, there are some deviations, particularly at higher carat values, suggesting potential underestimation or overestimation in some cases. The spread of points around the line reflects variance in prediction accuracy, implying that while the model captures general trends well, further refinements may improve precision.