Predicting Prices of Diamonds

Syed Huzaifa and William Redmond and Maaz Saad and Samson Tse

Faculty of Business & IT, Ontario Tech University

MBAI 5100 - Business Analytics

Dr. Amir Rastpour

December 10, 2022

Introduction

Diamonds are valuable gemstones that are in high demand due to their beauty and durability. As a result, the prices of diamonds can vary greatly depending on a variety of factors. In recent years, there has been an increase in using statistical methods to predict the prices of diamonds. In this paper, we will use regression analysis and neural networks to predict the prices of diamonds. The fictional narrative that we have established is that our group has been hired by a jewelry store and our objective is to create pricing models that can help the jewelry store decide on price points for their various offerings. As we will see in the next section, the price of a diamond is a function of numerous factors.

We will first review the data that was used for this project. Then, we will go over some summary statistics to get a better understanding of the variables and finally, we will present the results of our analysis and discuss our findings. Additionally, we use RStudio to build our models.

Data

The data that we used for our project was downloaded from Kaggle. In its entirety, the data consists of 10 variables and 53,940 rows. The data did not require much cleaning and was in a format that was ready to be fed into R.

This table provides a snapshot of the variables that were in the data set and their description.

Variable	Description
carat	weight of the diamond
cut	quality of the cut (Fair, Good, Very Good, Premium, Ideal)
color	diamond color, from J (worst) to D (best)
clarity	a measurement of how clear the diamond is (I1 (worst), SI2, SI1, VS2, VS1, VVS2, VVS1, IF (best))
depth	total depth percentage
table	width of top of diamond relative to widest point
price	price in US dollars
x	length in mm
y	width in mm
z	depth in mm

The majority of these variables are straightforward and simple to understand. We have the price of a diamond and its carat, color, and clarity. However, it is pertinent to explain the variables depth and table. The depth is calculated using the following formula: z / mean(x, y). And it is essentially a percentage. Table is the width from the top of the diamond to its widest point.

Results and Analysis

Part 1 - Descriptive Statistics

These two tables allow us to review and understand some of the basic statistical measures for all the variables in our data set. We can see the count, mean, median, standard deviation, minimum, maximum, 25th percentile, 50th percentile, and 75th percentile for all our variables. Moreover, we learn that three of our variables (Cut, Color, Clarity) are indeed factors and need to be treated as such.

Descriptives

	Carat	Cut	Color	Clarity	Price
N	53940	53940	53940	53940	53940
Mean	0.7979397				3932.800
Median	0.7000000				2401.000
Standard deviation	0.4740112				3989.440
Minimum	0.2000000				326
Maximum	5.010000				18823
25th percentile	0.4000000				950.0000
50th percentile	0.7000000				2401.000
75th percentile	1.040000				5324.250

Descriptives

	Depth	Table	x	y	z
N	53940	53940	53940	53940	53940
Mean	61.74940	57.45718	5.731157	5.734526	3.538734
Median	61.80000	57.00000	5.700000	5.710000	3.530000
Standard deviation	1.432621	2.234491	1.121761	1.142135	0.7056988
Minimum	43.00000	43.00000	0.000000	0.000000	0.000000
Maximum	79.00000	95.00000	10.74000	58.90000	31.80000
25th percentile	61.00000	56.00000	4.710000	4.720000	2.910000
50th percentile	61.80000	57.00000	5.700000	5.710000	3.530000
75th percentile	62.50000	59.00000	6.540000	6.540000	4.040000

We will now look at the frequency tables for our variables (Color, Clarity, Cut) that are factors.

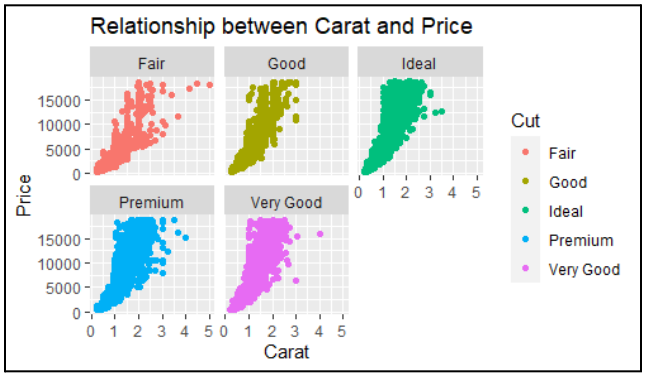
Frequencies of Color

Levels	Counts	% of Total	Cumulative %
D	6775	12.56025	12.56025
E	9797	18.16277	30.72303
F	9542	17.69003	48.41305
G	11292	20.93437	69.34742
H	8304	15.39488	84.74231
I	5422	10.05191	94.79422
J	2808	5.20578	100.00000

Frequencies of Clarity

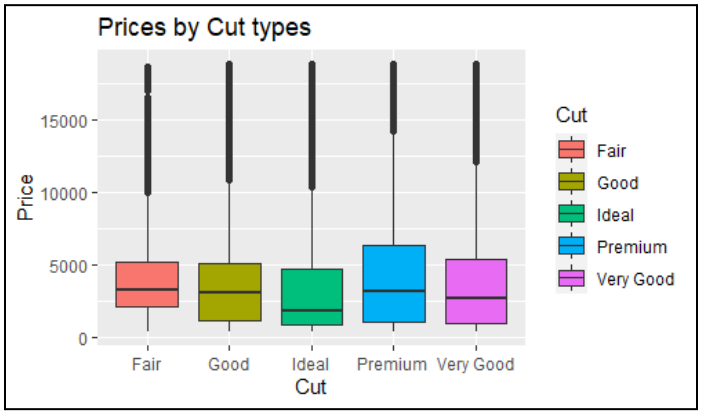
Levels	Counts	% of Total	Cumulative %
I1	741	1.37375	1.37375
IF	1790	3.31850	4.69225
SI1	13065	24.22136	28.91361
SI2	9194	17.04486	45.95847
VS1	8171	15.14831	61.10679
VS2	12258	22.72525	83.83204
VVS1	3655	6.77605	90.60808
VVS2	5066	9.39192	100.00000

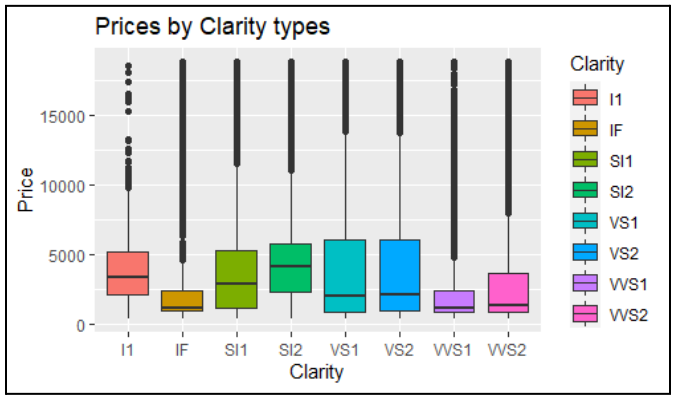
Now, we will visualize some important relationships between our variables.



In this scatter plot, we see the price of diamond broken down by its type of cut as a factor of carat. We can observe that as the type of cut improves, our prices start shifting upwards, i.e. to higher price points.

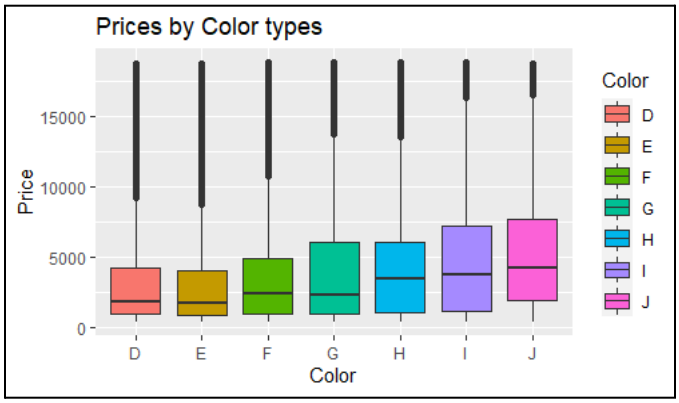
In this box plot, we can see the prices of our diamonds broken down by their type of cut. This visualization helps us comprehend that the Premium cut of diamonds have higher price points compared to the other types of diamonds. The interesting thing to note is that the Ideal cut has the lowest values despite not being the worst type of cut overall.



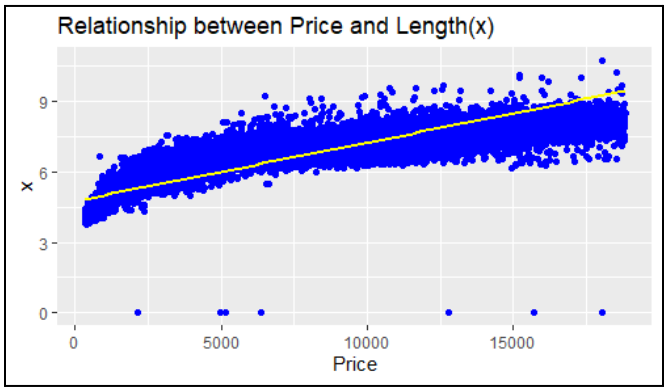


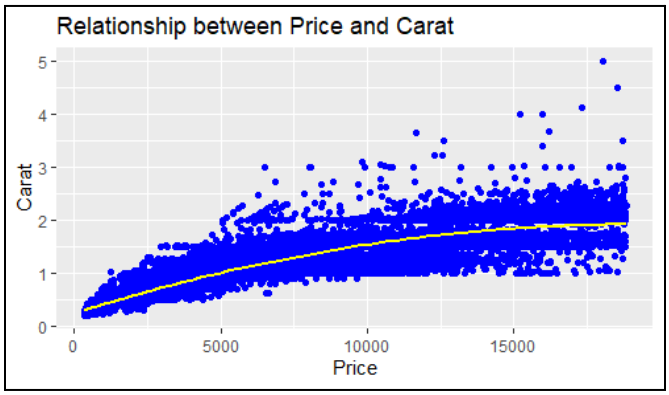
In this box plot, we can see the prices of our diamonds broken down by their type of clarity. This visualization helps us comprehend that diamonds with a clarity rank of SI2 have higher price points compared to the other types of diamonds despite being on the lower end of the spectrum.

In this box plot, we can see the prices of our diamonds broken down by their type of color. This visualization helps us comprehend that diamonds that have a color ranked as J, have higher price points compared to the other types of diamonds.



Finally, we would like to show two important relationships between the price of diamonds and their carat and length. We see that by increasing the length of a diamond, the price increases as well. Similarly, as the diamond increases in carat, so does the price.





Lastly, we would like to end this section by reviewing a pair plot of our variables without the length (x), width (y), and depth (z). This pairplot helps us visualize all the possible combinations of relationships that exist between our variables. We already expanded in detail on some of the more significant relationships earlier.

Part 2 - Prediction Analysis

In the model, we wanted to predict the price of a diamond given its attributes. As the dependent variable (price) is quantitative, four multiple linear regression models and two neural networks were created to compare performance.

Multiple Linear Regression Model

Looking at the plot of the variables, most variables seemed to have a relationship with the price of diamonds. As a result, to create the multiple linear regression model, we used backward selection to select the predictor variables.

Model 1

We first created a model containing all variables. The first model was statistically significant with a high adjusted R-squared value.

Below is a summary of the first model.

```
Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)    784.995    492.131   1.595    0.111
carat        11044.565     57.982 190.482  < 2e-16 ***
cutGood        562.309     41.557  13.531  < 2e-16 ***
cutIdeal       827.829     41.224  20.081  < 2e-16 ***
cutPremium     763.084     39.518  19.310  < 2e-16 ***
cutVery Good   699.260     40.160  17.412  < 2e-16 ***
colorE        -209.802     21.754  -9.644  < 2e-16 ***
colorF        -281.908     22.002 -12.813  < 2e-16 ***
colorG        -488.456     21.490 -22.730  < 2e-16 ***
colorH        -990.406     22.873 -43.300  < 2e-16 ***
colorI       -1453.564     25.676 -56.612  < 2e-16 ***
colorJ       -2369.219     31.763 -74.591  < 2e-16 ***
clarityIF     5452.574     61.260  89.007  < 2e-16 ***
claritySI1    3730.859     52.565  70.977  < 2e-16 ***
claritySI2    2777.096     52.767  52.629  < 2e-16 ***
clarityVS1    4653.980     53.706  86.657  < 2e-16 ***
clarityVS2    4343.202     52.812  82.239  < 2e-16 ***
clarityVVS1   5091.828     56.837  89.587  < 2e-16 ***
clarityVVS2   5024.535     55.245  90.949  < 2e-16 ***
depth          -51.559      5.433  -9.490  < 2e-16 ***
table          -22.246      3.532  -6.299 3.02e-10 ***
x            -1407.734     98.164 -14.341  < 2e-16 ***
y              495.752     97.386   5.091 3.59e-07 ***
z              -46.083     36.374  -1.267    0.205
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1145 on 37713 degrees of freedom
Multiple R-squared:  0.9186,    Adjusted R-squared:  0.9185
F-statistic: 1.85e+04 on 23 and 37713 DF,  p-value: < 2.2e-16
```

Model 2

Given the results of the Model 1, we removed the z variable as it was not significant. Additionally, we removed variables that were dependent on one another. Depth is calculated by using x, y and z. Since z was already removed, we compared the results of only including Depth or only including x and y among the other variables. The results of this comparison showed that x and y together improved the model more than Depth. The second model was statistically significant with a high R-squared value. However, the R-squared value dropped slightly and the residual standard error increased slightly.

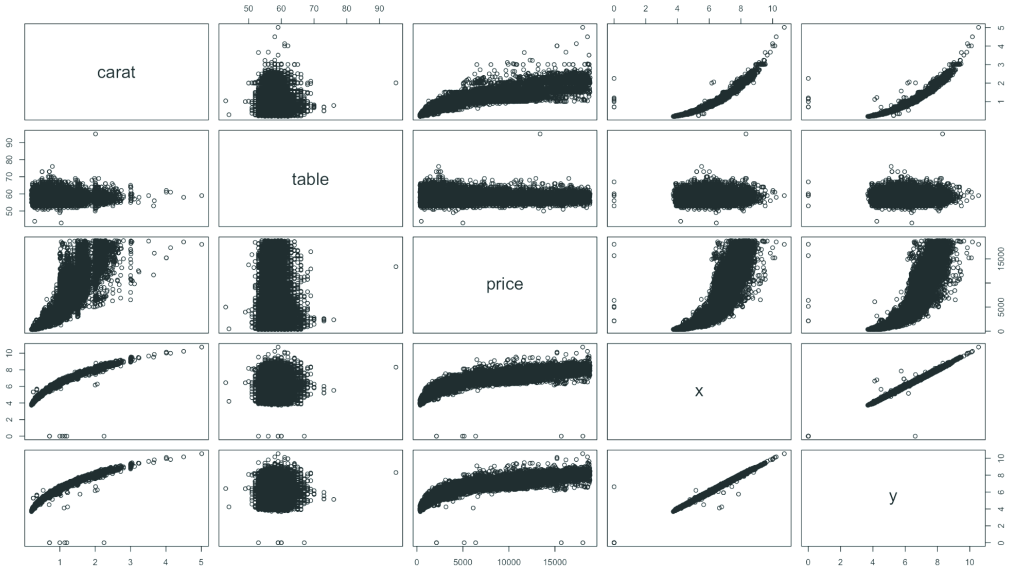
Below is a summary of the second model.

```
Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  -3672.582    211.593 -17.357  < 2e-16 ***
carat        10901.720     56.563 192.737  < 2e-16 ***
cutGood        643.359     40.953  15.710  < 2e-16 ***
cutIdeal       984.244     38.709  25.427  < 2e-16 ***
cutPremium     906.103     37.341  24.266  < 2e-16 ***
cutVery Good   818.626     38.694  21.156  < 2e-16 ***
colorE        -209.788     21.788  -9.629  < 2e-16 ***
colorF        -283.977     22.035 -12.887  < 2e-16 ***
colorG        -493.909     21.517 -22.954  < 2e-16 ***
colorH        -996.210     22.902 -43.498  < 2e-16 ***
colorI       -1459.736     25.710 -56.777  < 2e-16 ***
colorJ       -2376.532     31.805 -74.721  < 2e-16 ***
clarityIF     5485.126     61.281  89.507  < 2e-16 ***
claritySI1    3736.852     52.644  70.984  < 2e-16 ***
claritySI2    2789.062     52.838  52.785  < 2e-16 ***
clarityVS1    4670.394     53.767  86.863  < 2e-16 ***
clarityVS2    4355.681     52.882  82.367  < 2e-16 ***
clarityVVS1   5116.986     56.878  89.965  < 2e-16 ***
clarityVVS2   5045.743     55.297  91.248  < 2e-16 ***
table           -6.709      3.237  -2.073   0.0382 *
x            -1471.583     97.120 -15.152  < 2e-16 ***
y              592.489     96.733   6.125 9.16e-10 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1147 on 37715 degrees of freedom
Multiple R-squared:  0.9183,    Adjusted R-squared:  0.9183
F-statistic: 2.02e+04 on 21 and 37715 DF,  p-value: < 2.2e-16
```

Model 3

We choose to continue improving Model 2 as that model removed multicollinearity. Looking at the plots between variables, carat, x and y are not a straight linear relationship with price.



There is a curve to the plots. As such, we added the poly function to these variables. This improved the model. Additionally, we removed the variable table as its significance was low and did not impact the p-value, adjusted r-squared or the residual standard error when removed. Below is a summary of the third model.

```
Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)         326.72      58.06   5.627 1.84e-08 ***
poly(carat, 2)1  646023.91   14455.67  44.690  < 2e-16 ***
poly(carat, 2)2  -82990.28    2106.06 -39.406  < 2e-16 ***
cutGood             302.78      39.56   7.654 2.00e-14 ***
cutIdeal            589.90      37.06  15.917  < 2e-16 ***
cutPremium          492.33      37.01  13.304  < 2e-16 ***
cutVery Good        391.69      37.83  10.353  < 2e-16 ***
colorE             -212.10      20.69 -10.249  < 2e-16 ***
colorF             -283.54      20.93 -13.544  < 2e-16 ***
colorG             -496.58      20.44 -24.292  < 2e-16 ***
colorH            -1010.78      21.77 -46.428  < 2e-16 ***
colorI            -1476.61      24.42 -60.464  < 2e-16 ***
colorJ            -2374.99      30.22 -78.598  < 2e-16 ***
clarityIF          5119.54      58.49  87.524  < 2e-16 ***
claritySI1         3438.61      50.23  68.458  < 2e-16 ***
claritySI2         2498.99      50.40  49.582  < 2e-16 ***
clarityVS1         4334.96      51.34  84.440  < 2e-16 ***
clarityVS2         4032.47      50.49  79.873  < 2e-16 ***
clarityVVS1        4750.38      54.33  87.439  < 2e-16 ***
clarityVVS2        4685.43      52.83  88.697  < 2e-16 ***
poly(x, 2)1     -320925.71   23612.97 -13.591  < 2e-16 ***
poly(x, 2)2     -160987.45    8111.50 -19.847  < 2e-16 ***
poly(y, 2)1      462980.64   22185.53  20.869  < 2e-16 ***
poly(y, 2)2      321771.29    7919.44  40.631  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1090 on 37713 degrees of freedom
Multiple R-squared:  0.9263,    Adjusted R-squared:  0.9263
F-statistic: 2.062e+04 on 23 and 37713 DF,  p-value: < 2.2e-16
```

Model 4

Looking at the plots between variables, there appears to be interactions between carat and x. As x increases, carat increases. There's also an interaction between carat and y. As y increases carat increases. There's also an interaction between x and y. As y increases x increases. As such, we added these interactions to the model. This improved the model.

Below is a summary of the fourth model.

```
Coefficients:
                              Estimate Std. Error t value Pr(>|t|)
(Intercept)                  2.837e+02  2.348e+02   1.208    0.227
poly(carat, 2)1              7.112e+05  3.025e+04  23.509  < 2e-16 ***
poly(carat, 2)2             -1.958e+05  3.450e+04  -5.675 1.40e-08 ***
poly(x, 2)1                  1.714e+05  1.209e+05   1.418    0.156
poly(x, 2)2                 -2.353e+05  5.536e+04  -4.250 2.14e-05 ***
poly(y, 2)1                 -1.576e+05  1.241e+05  -1.270    0.204
poly(y, 2)2                  4.367e+05  5.972e+04   7.313 2.67e-13 ***
cutGood                      3.622e+02  3.843e+01   9.425  < 2e-16 ***
cutIdeal                     6.383e+02  3.634e+01  17.564  < 2e-16 ***
cutPremium                   5.102e+02  3.612e+01  14.127  < 2e-16 ***
cutVery Good                 4.527e+02  3.704e+01  12.220  < 2e-16 ***
colorE                      -2.108e+02  1.992e+01 -10.585  < 2e-16 ***
colorF                      -2.687e+02  2.015e+01 -13.335  < 2e-16 ***
colorG                      -5.040e+02  1.967e+01 -25.616  < 2e-16 ***
colorH                      -1.027e+03  2.097e+01 -48.979  < 2e-16 ***
colorI                      -1.509e+03  2.352e+01 -64.175  < 2e-16 ***
colorJ                      -2.406e+03  2.910e+01 -82.666  < 2e-16 ***
clarityIF                    4.987e+03  5.649e+01  88.283  < 2e-16 ***
claritySI1                   3.324e+03  4.849e+01  68.546  < 2e-16 ***
claritySI2                   2.399e+03  4.866e+01  49.298  < 2e-16 ***
clarityVS1                   4.215e+03  4.956e+01  85.041  < 2e-16 ***
clarityVS2                   3.905e+03  4.874e+01  80.117  < 2e-16 ***
clarityVVS1                  4.613e+03  5.245e+01  87.956  < 2e-16 ***
clarityVVS2                  4.558e+03  5.098e+01  89.397  < 2e-16 ***
poly(carat, 2)1:poly(x, 2)1  1.775e+08  2.988e+07   5.940 2.88e-09 ***
poly(carat, 2)2:poly(x, 2)1  3.721e+08  3.480e+07  10.695  < 2e-16 ***
poly(carat, 2)1:poly(x, 2)2 -2.679e+08  2.652e+07 -10.100  < 2e-16 ***
poly(carat, 2)2:poly(x, 2)2 -2.974e+06  2.337e+06  -1.273    0.203
poly(carat, 2)1:poly(y, 2)1 -1.707e+08  2.717e+07  -6.282 3.37e-10 ***
poly(carat, 2)2:poly(y, 2)1 -3.658e+08  3.476e+07 -10.525  < 2e-16 ***
poly(carat, 2)1:poly(y, 2)2  2.717e+08  2.653e+07  10.239  < 2e-16 ***
poly(carat, 2)2:poly(y, 2)2  9.663e+05  2.339e+06   0.413    0.679
poly(x, 2)1:poly(y, 2)1      2.880e+06  7.966e+06   0.362    0.718
poly(x, 2)2:poly(y, 2)1      2.218e+08  2.959e+07   7.494 6.85e-14 ***
poly(x, 2)1:poly(y, 2)2     -2.253e+08  2.958e+07  -7.618 2.63e-14 ***
poly(x, 2)2:poly(y, 2)2     -3.586e+06  3.591e+05  -9.986  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1048 on 37701 degrees of freedom
Multiple R-squared:  0.9318,    Adjusted R-squared:  0.9318
F-statistic: 1.472e+04 on 35 and 37701 DF,  p-value: < 2.2e-16
```

Model Performance Evaluation:

To evaluate the models and measure their performance we are setting up training data set and testing data set to compare the model performances and the mean absolute error (MAE) between the models:

Model 1:
Training Data:

- p-value: < 2.2e-16
- adjusted r-square: 0.9207
- residual standard error: 1121 on 37537 degrees of freedom

Testing Error: Mean Absolute Error: 747.66
The training model provides a better fit compared to the original model one.

Model 2:
Training Data:

- p-value: < 2.2e-16
- adjusted r-square: 0.9199
- residual standard error: 1127 on 37541 degrees of freedom

Testing Error: Mean Absolute Error: 751.87
Testing error on Model 2 is higher than model 1 and therefore further agrees that model 1 is statistically more significant.

Model 3:
Training Data:

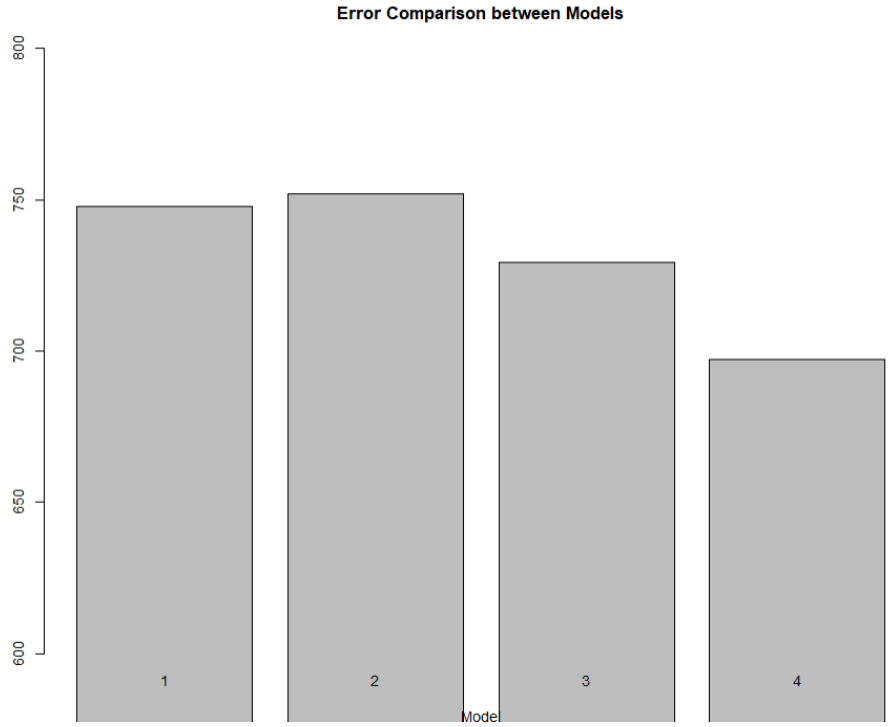
- p-value: < 2.2e-16
- adjusted r-square: 0.9257
- residual standard error: 1085 on 37538 degrees of freedom

Testing Error: Mean Absolute Error: 729.32
Testing error for Model 3 is much lower than model 1 and therefore model 3 is more improved than previous 2 models.

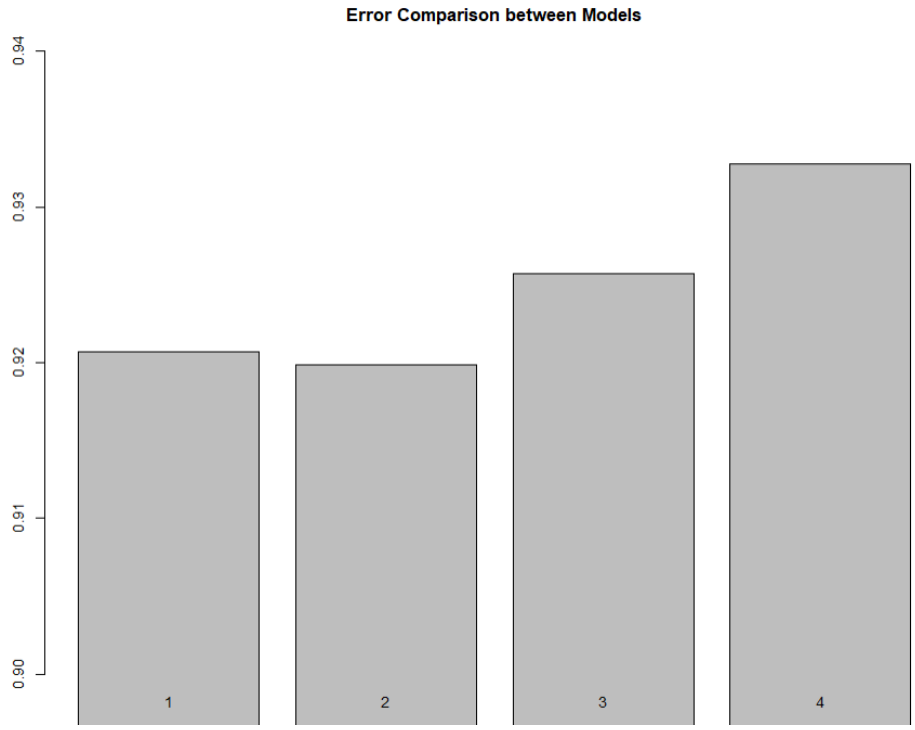
Model 4:
Training Data:

- p-value: < 2.2e-16
- adjusted r-square: 0.9328
- residual standard error: 1032 on 37526 degrees of freedom

Testing Error: Mean Absolute Error: 697.38
Model 4 has the lowest mean absolute error out of the 4 models and therefore we have improved the model and it is most significant out of the 4 models.



The figure above shows the comparative Mean Absolute error between the 4 models.



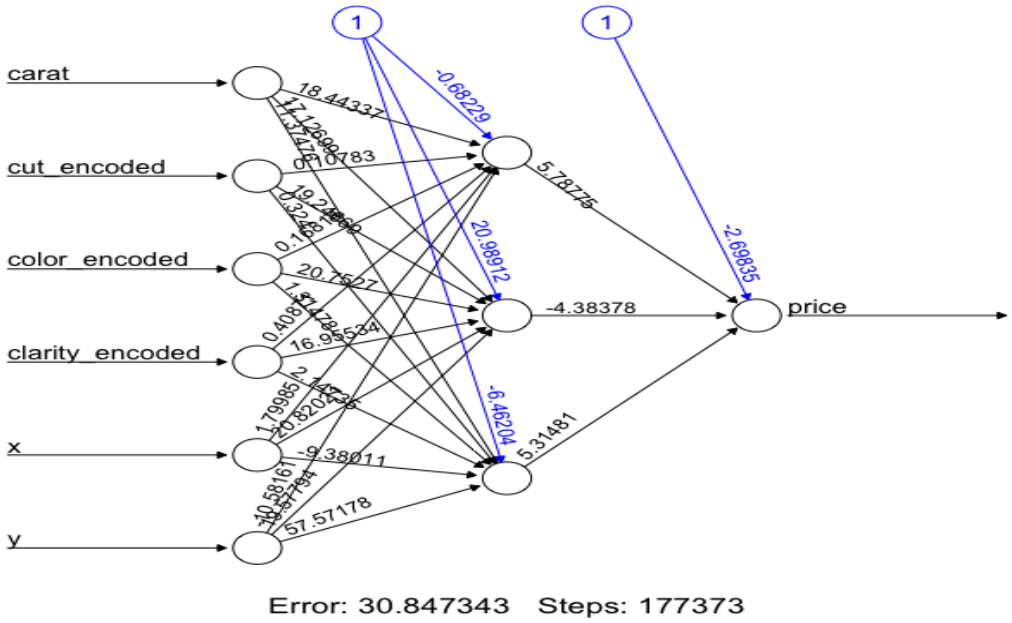
The figure above shows the Adjusted R-Square comparison between the 4 models.

With the results derived from the performance evaluation of the 4 regression models we could not conclude if these are the best possible models for our business case. Therefore, we have decided to explore other modeling options to optimize prediction accuracy and minimize errors.

Model 5: Neural Network 1

After developing the 4 linear regression models, we already have a good understanding of what variables can predict the price of the diamonds so we decide to use the variables in model 4, the best performing model, to be the inputs of our neural network. However, we do not add any poly function or interaction terms as the strength of a neural network is capturing the non-linear relationships. The first neural network is with 1 hidden layer which has 3 neurons.

Below is how the neural network 1 looks like.



Training Model and Testing Errors:

To further evaluate the models we are setting up training data set and testing data set to compare the model performances and the mean absolute error (MAE) between the models:

Train Data:

- Sum of squared error: 30.8473

It is important to note that this sum of squared error cannot be directly compared to the linear regression models' because we normalize the inputs before passing them into the neural network.

To make the linear regression models and neural networks be comparable, we denormalize the values in the testing part after getting the output from the neural networks.

Testing Error:

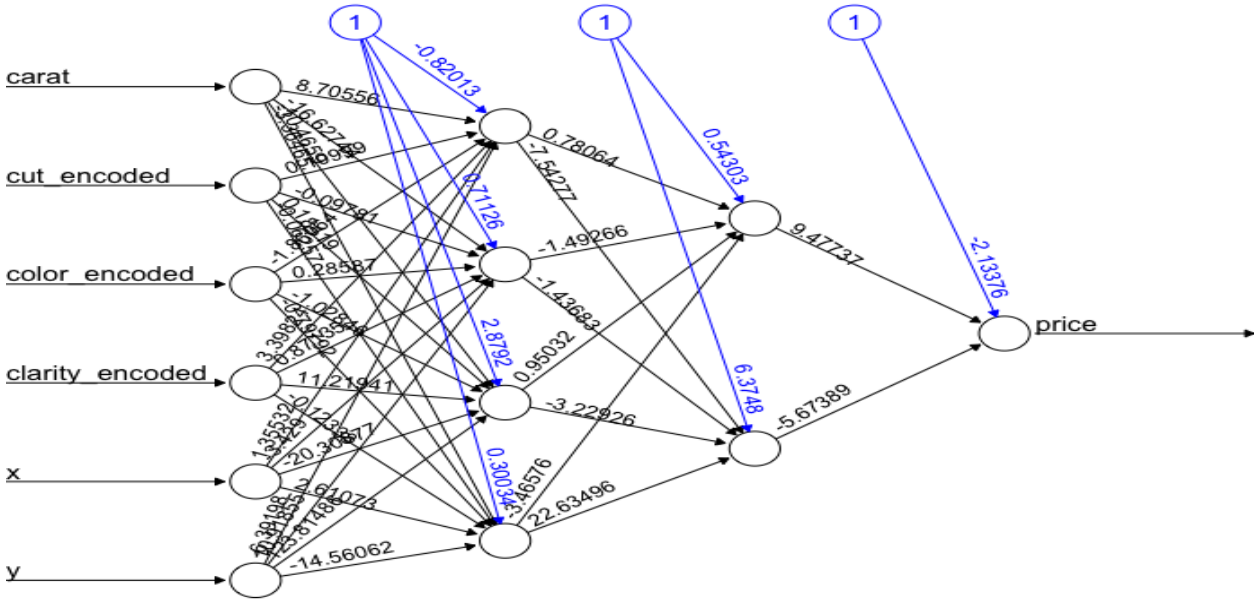
- Sum of squared error (before denormalizing): 14.4785
- Mean Absolute Error (denormalized): 366.6354

The MAE of neural network 1 is smaller than all the linear regression models so it is a better model in predicting the price of diamonds.

Model 6: Neural Network 2

Neural network 2 has the same inputs as neural network 1. The second neural network is with 2 hidden layers which have 4 neurons and 2 neurons respectively.

Below is how the neural network 1 looks like.



Training Model and Testing Errors:

To further evaluate the models we are setting up training data set and testing data set to compare the model performances and the mean absolute error (MAE) between the models:

Train Data:

- Sum of squared error: 17.5847

Same as the neural network 1, this sum of error is according to values that are normalized so we denormalize the values in the testing part after getting the output from the neural networks so that we can compare the neural network with the linear regression models.
Testing Error:

- Sum of squared error (before denormalizing): 8.3439
- Mean Absolute Error (denormalized): 330.0478

With the neural network 2 having a smaller training error and testing error than the neural network 1, we conclude that the neural network 2 is a better model as it performs better prediction and does not overfit as indicated by the smaller testing error. Neural network 2 also has the smallest MAE compared to all other models so it is the best model in predicting the price of diamonds.

Conclusion

In this research paper, our objective was to predict the price of a diamond for a jewelry store. We used historical data to build different variations of a regression model and neural networks. There were different variations done with different sets of variables to improve the accuracy of our models. As we saw in the study, the neural network models had a lower error rate compared to the multiple regression models. In that case, we would recommend our neural network 2 to the jewelry store to predict the price of the diamonds.

Appendix A: Code for descriptive statistics

```
# Descriptive Statistics

# Install packages and load libraries

install.packages(c('tibble', 'dplyr', 'readr', 'jmv'))
library(dplyr)
library(tidyverse)
library(jmv)
library(ggplot2)
library(GGally)

# Load the Diamonds data set.
diamonds <- read_csv("diamonds.csv")

# Convert character variables to factor
diamonds <- diamonds %>% mutate(cut=as.factor(cut),color=as.factor(color),
                        clarity=as.factor(clarity))

# Convert the diamonds type to a data frame
diamonds <- data.frame(diamonds)

# Drop first column as it is not needed
diamonds <- diamonds[-c(1)]

# Descriptive statistics with jmv

# Summary Statistics
summary_table1 <- descriptives(data = diamonds, vars = c("Carat", "Cut",
                                          "Color", "Clarity",
                                          "Price"),
          splitBy = NULL, freq = TRUE, n = TRUE, missing = FALSE,
          mean = TRUE, sd = TRUE, median = TRUE,
          min = TRUE, max = TRUE, pcEqGr = TRUE)

summary_table2 <- descriptives(data = diamonds, vars = c("Depth", "Table", "x",
                         "y", "z"),
          splitBy = NULL, freq = TRUE, n = TRUE, missing = FALSE,
          mean = TRUE, sd = TRUE, median = TRUE,
```

```
        min = TRUE, max = TRUE, pcEqGr = TRUE)

# Plots
# Relationship between Carat and Price - Detailed
ggplot(data = diamonds)+geom_point(mapping = aes(x=Carat,y=Price,col=Cut))+
  ggtitle("Relationship between Carat and Price")+xlab("Carat")+ylab("Price")+
  facet_wrap(~Cut,)

# Box Plot - Prices and Cut
ggplot(data = diamonds, aes(x=Cut, y=Price, fill=Cut)) +
  geom_boxplot() + ggtitle("Prices by Cut types")

# Box Plot - Prices and Clarity
ggplot(data = diamonds, aes(x=Clarity, y=Price, fill=Clarity)) +
  geom_boxplot() + ggtitle("Prices by Clarity types")

# Box Plot - Prices and Color
ggplot(data = diamonds, aes(x=Color, y=Price, fill=Color)) +
  geom_boxplot() + ggtitle("Prices by Color types")

# Scatter Plot - Price & Carat
ggplot(diamonds, aes(x=Price, y=Carat)) +
  geom_point(color="blue") + geom_smooth(formula = y ~ poly(x, 2),method = lm, color=
"yellow") +
  ggtitle("Relationship between Price and Carat")

# Scatter Plot - Price & Length (x)
ggplot(diamonds, aes(x=Price, y=x)) +
  geom_point(color="blue") + geom_smooth(formula = y ~ x,method = lm, color= "yellow") +
  ggtitle("Relationship between Price and Length(x)")

# Pair Plot
ggpairs(diamonds, columns = 1:7, aes(color = Cut, alpha = 0.5),
        upper = list(continuous = wrap("cor", size = 2.5)))
```

Appendix B: Code for linear models

```
#load libraries
library(tidyverse)

#read csv file
df_diamonds <- read_csv("diamonds.csv")

#create train and test set
set.seed(1)
ind <- sample(2, nrow(df_diamonds), replace=TRUE, prob=c(0.7, 0.3))

train_set <- df_diamonds[ind==1,]
test_set <- df_diamonds[ind==2,]

#model 1
lm_1 <- lm(formula = price ~ ., data = train_set)
summary(lm_1)

#model 2, removing y and z (not significant variables) and depth, which depends on x, y and z
but doesn't seem to impact the model
lm_2 <- lm(formula = price ~ carat + cut + color + clarity + table + x + y, data = train_set)
summary(lm_2)

#model 3, carat and x have a curved fitted line, adding poly
lm_3 <- lm(formula = price ~ poly(carat, 2) + cut + color + clarity + poly(x, 2) + poly(y, 2), data
= train_set)
summary(lm_3)

#model 4, adding interaction terms as their plots seem correlated
lm_4 <- lm(formula = price ~ poly(carat, 2) * poly(x, 2) + poly(carat, 2) * poly(y, 2) + poly(x, 2)
* poly(y, 2) + cut + color + clarity, data = train_set)
summary(lm_4)

#calculating Mean Absolute Error
mae(lm_1, test_set)
mae(lm_2, test_set)
mae(lm_3, test_set)
mae(lm_4, test_set)
```

Appendix C: Code for neural networks

```
# encode the categorical variables into numbers
diamonds_encoded<-diamonds
cut_fac<-factor(x=diamonds$cut)
order_cut<-c('Fair', 'Good', 'Very Good','Premium','Ideal')
cut_fac<-factor(x=diamonds$cut, levels=order_cut, ordered=T)
levels(cut_fac)<-c('1','2','3','4','5')
diamonds_encoded[["cut_encoded"]] <- cut_fac
diamonds_encoded[ , 12] <- as.data.frame(apply(diamonds_encoded['cut_encoded'], 2,
as.numeric))

color_fac<-factor(x=diamonds$color)
order_color<-c('J', 'I', 'H','G','F','E','D')
color_fac<-factor(x=diamonds$color, levels=order_color, ordered=T)
levels(color_fac)<-c('1','2','3','4','5','6','7')
diamonds_encoded[["color_encoded"]] <- color_fac
diamonds_encoded[ , 13] <- as.data.frame(apply(diamonds_encoded['color_encoded'], 2,
as.numeric))

clarity_fac<-factor(x=diamonds$clarity)
order_clarity<-c('I1', 'SI2', 'SI1','VS2','VS1','VVS2','VVS1','IF')
clarity_fac<-factor(x=diamonds$clarity, levels=order_clarity, ordered=T)
levels(clarity_fac)<-c('1','2','3','4','5','6','7','8')
diamonds_encoded[["clarity_encoded"]] <- clarity_fac
diamonds_encoded[ , 14] <- as.data.frame(apply(diamonds_encoded['clarity_encoded'], 2,
as.numeric))

diamonds_encoded <- diamonds_encoded[-c(1,3:5)]

# scale data for neural network
max = apply(diamonds_encoded , 2 , max)
min = apply(diamonds_encoded, 2 , min)
scaled = as.data.frame(scale(diamonds_encoded, center = min, scale = max - min))

# set up denoramlize function for later comparison
denormalize <- function(x,min,max) {
  x*(max-min) + min}

# load library
```

```
library(neuralnet)

# create training and testing set
trainNN = scaled[ind==1, ]
testNN = scaled[ind==2, ]

# create neural network 1
NN = neuralnet(price ~ carat + cut_encoded + color_encoded + clarity_encoded + x + y,
            trainNN, hidden = 3 , stepmax = 300000, linear.output = F )

# get SSE for training data
Train_NN1_Output <- compute(NN, trainNN)
NN1_Train_SSE <- sum((Train_NN1_Output$net.result - trainNN[, 4])^2)/2
NN1_Train_SSE

# get SSE and MAE for testing data
Test_NN1_Output <- compute(NN, testNN)
NN1_Test_SSE <- sum((Test_NN1_Output$net.result - testNN[, 4])^2)/2
NN1_Test_SSE
denorm_Test1_price<-denormalize(Test_NN1_Output$net.result,min[4],max[4])
NN1_Test_MAE <- (sum(abs(denorm_Test1_price - datatest[, 4])))/nrow(datatest)
NN1_Test_MAE



# create neural network 2
NN2 = neuralnet(price ~ carat + cut_encoded + color_encoded + clarity_encoded + x + y,
            trainNN, hidden = c(4,2) , stepmax = 500000, linear.output = F ) # it takes literally
more than an hour to run

# get SSE for training data
plot(NN2)

# get SSE and MAE for testing data
Test_NN2_Output <- compute(NN2, testNN)
NN2_Test_SSE <- sum((Test_NN2_Output$net.result - testNN[, 4])^2)/2
NN2_Test_SSE
denorm_Test2_price<-denormalize(Test_NN2_Output$net.result,min[4],max[4])
NN2_Test_MAE <- (sum(abs(denorm_Test2_price - datatest[, 4])))/nrow(datatest)
NN2_Test_MAE
```