Introduction

This report presents the results of the analysis on diamond data.

Analysis

```
# Load the dataset
data <- read.csv("C:/Users/lenovo/Documents/Projects/Data Analysis/DiamondDataComplete.csv")

# Data quality check
summary(data)
```

```
##      carat             cut                color              clarity
##  Min.   :0.2000   Length:50000       Length:50000       Length:50000
##  1st Qu.:0.4000   Class :character   Class :character   Class :character
##  Median :0.7000   Mode  :character   Mode  :character   Mode  :character
##  Mean   :0.7974
##  3rd Qu.:1.0400
##  Max.   :5.0100
##      depth           table           price             x
##  Min.   :43.00   Min.   :43.00   Min.   :  326   Min.   : 0.00
##  1st Qu.:61.00   1st Qu.:56.00   1st Qu.:  949   1st Qu.: 4.71
##  Median :61.80   Median :57.00   Median : 2401   Median : 5.70
##  Mean   :61.75   Mean   :57.45   Mean   : 3925   Mean   : 5.73
##  3rd Qu.:62.50   3rd Qu.:59.00   3rd Qu.: 5312   3rd Qu.: 6.54
##  Max.   :79.00   Max.   :95.00   Max.   :18823   Max.   :10.74
##        y                z
##  Min.   : 0.000   Min.   : 0.000
##  1st Qu.: 4.720   1st Qu.: 2.910
##  Median : 5.710   Median : 3.520
##  Mean   : 5.732   Mean   : 3.538
##  3rd Qu.: 6.540   3rd Qu.: 4.030
##  Max.   :31.800   Max.   :31.800
```

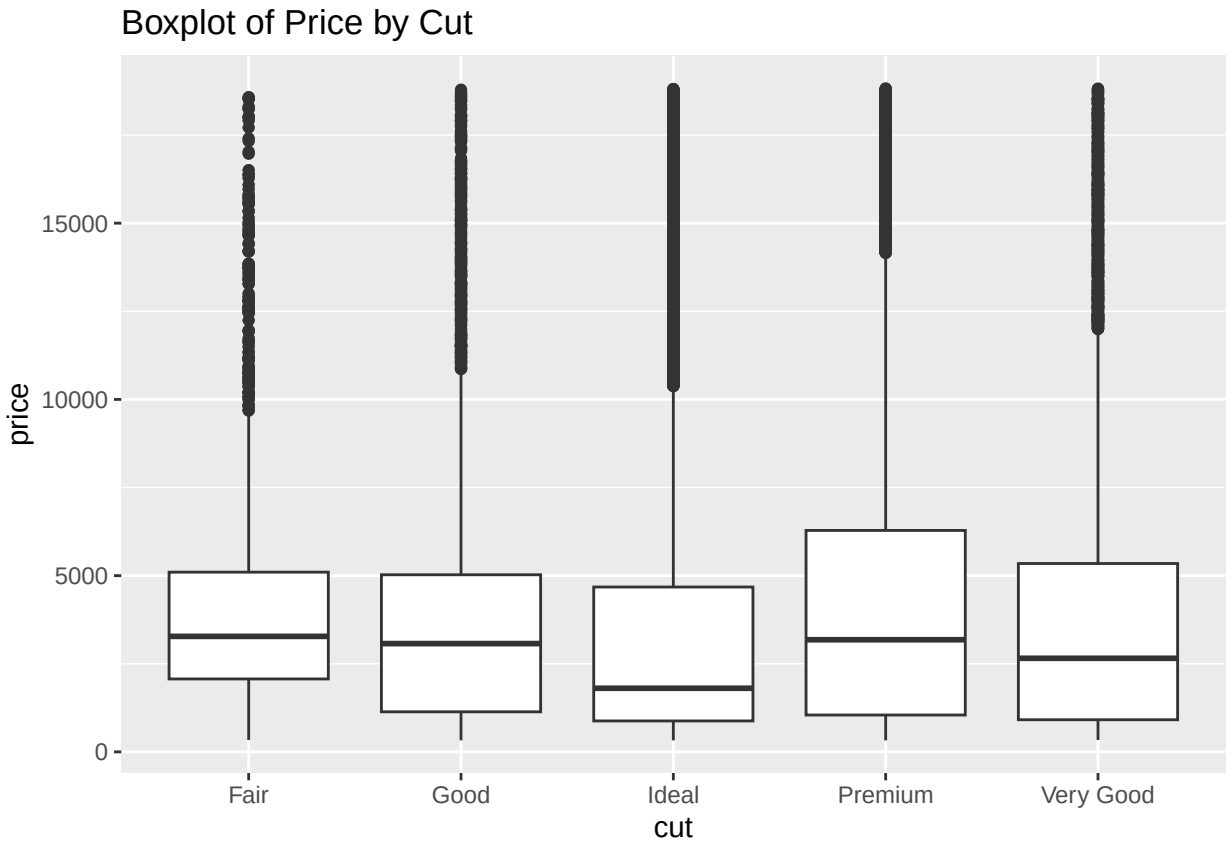
```
str(data)
```

```
## 'data.frame':    50000 obs. of  10 variables:
##  $ carat  : num  0.74 0.72 0.36 0.31 1 0.5 1.07 0.53 1.5 1.01 ...
##  $ cut    : chr  "Very Good" "Ideal" "Ideal" "Premium" ...
##  $ color  : chr  "D" "H" "D" "I" ...
##  $ clarity: chr  "VS2" "VS1" "VVS2" "VVS1" ...
##  $ depth  : num  59.8 61.6 61.9 61 59.1 61.4 60.6 58.5 63.6 62.9 ...
##  $ table  : num  58 59 53 58 62 61 66 61 55 57 ...
##  $ price  : int  3476 2642 957 732 3640 1172 4554 1950 13853 4858 ...
##  $ x      : num  5.9 5.75 4.57 4.39 6.5 5.14 6.65 5.39 7.27 6.35 ...
##  $ y      : num  5.94 5.78 4.6 4.33 6.47 5.09 6.46 5.28 7.22 6.41 ...
##  $ z      : num  3.54 3.55 2.84 2.66 3.83 3.14 3.97 3.12 4.61 4.01 ...
```

```
# Check for missing values
colSums(is.na(data))
```

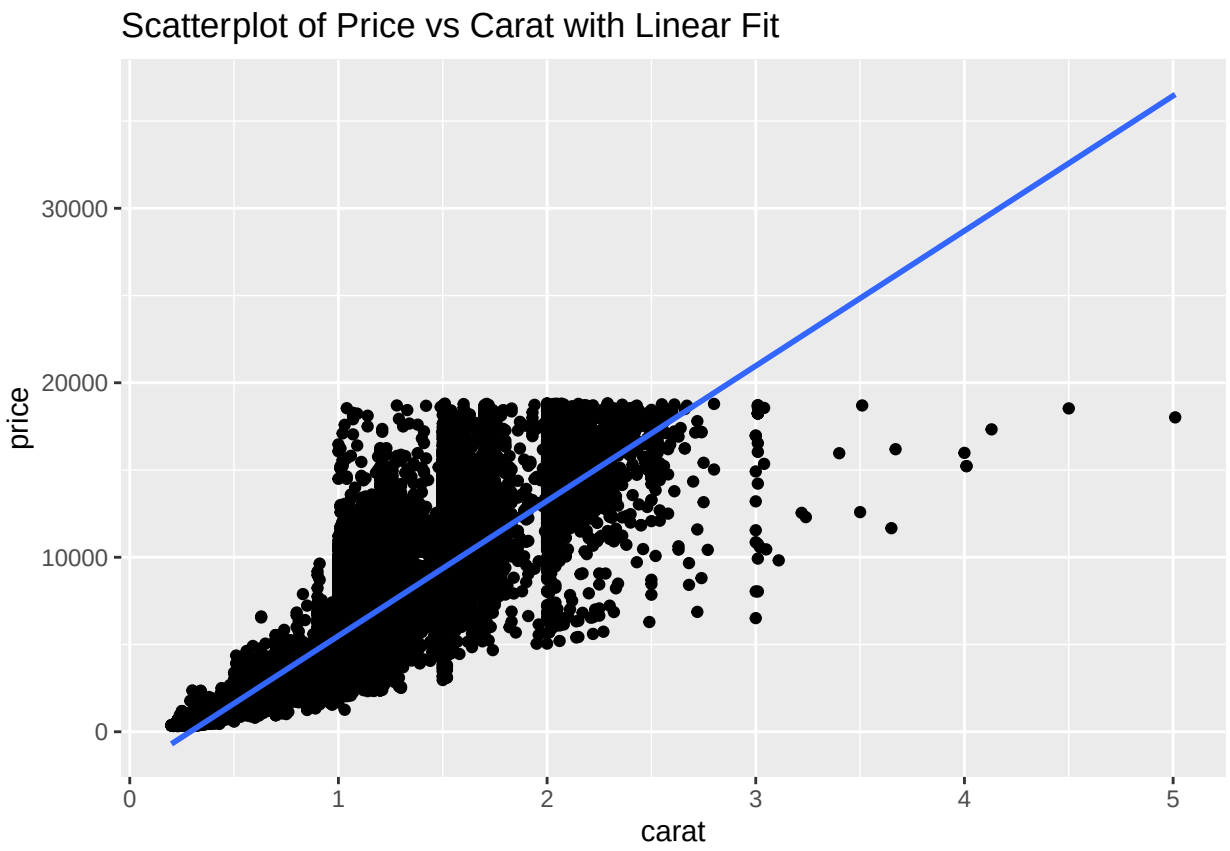
```
##   carat     cut   color clarity   depth   table   price       x       y       z
##       0       0       0       0       0       0       0       0       0       0
```

```r
# Boxplot of price by cut
library(ggplot2)
ggplot(data, aes(x = cut, y = price)) +
  geom_boxplot() +
  ggtitle("Boxplot of Price by Cut")
```




```r
# Scatterplot of price vs carat with linear fit
ggplot(data, aes(x = carat, y = price)) +
  geom_point() +
  geom_smooth(method = "lm") +
  ggtitle("Scatterplot of Price vs Carat with Linear Fit")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```




```r
library(caret)
```

```
## Loading required package: lattice
```

```r
set.seed(123)
trainIndex <- createDataPartition(data$price, p = 0.7, list = FALSE)
trainData <- data[trainIndex, ]
testData <- data[-trainIndex, ]

# Linear regression model
model <- lm(price ~ ., data = trainData)
summary(model)
```

```
##
## Call:
## lm(formula = price ~ ., data = trainData)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -20701.2   -590.2   -184.6    371.3  10695.7
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1647.999    557.064   2.958  0.00309 **
## carat        11041.922     60.197 183.431  < 2e-16 ***
```

```
## cutGood        582.035     41.735  13.946  < 2e-16 ***
## cutIdeal       834.009     41.491  20.101  < 2e-16 ***
## cutPremium     773.453     39.910  19.380  < 2e-16 ***
## cutVery Good   740.726     40.056  18.492  < 2e-16 ***
## colorE        -218.279     22.358  -9.763  < 2e-16 ***
## colorF        -271.326     22.523 -12.047  < 2e-16 ***
## colorG        -479.118     22.058 -21.721  < 2e-16 ***
## colorH        -977.745     23.413 -41.760  < 2e-16 ***
## colorI       -1440.213     26.379 -54.596  < 2e-16 ***
## colorJ       -2364.775     32.577 -72.591  < 2e-16 ***
## clarityIF     5447.662     63.374  85.960  < 2e-16 ***
## claritySI1    3778.105     54.240  69.655  < 2e-16 ***
## claritySI2    2826.826     54.426  51.939  < 2e-16 ***
## clarityVS1    4685.021     55.353  84.639  < 2e-16 ***
## clarityVS2    4384.419     54.517  80.422  < 2e-16 ***
## clarityVVS1   5100.920     58.645  86.980  < 2e-16 ***
## clarityVVS2   5067.827     57.034  88.857  < 2e-16 ***
## depth          -59.376      6.722  -8.833  < 2e-16 ***
## table          -28.661      3.622  -7.912 2.61e-15 ***
## x             -968.815     59.571 -16.263  < 2e-16 ***
## y               79.435     39.879   1.992  0.04639 *
## z             -105.458     74.015  -1.425  0.15422
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1136 on 34978 degrees of freedom
## Multiple R-squared:  0.9192, Adjusted R-squared:  0.9191
## F-statistic: 1.73e+04 on 23 and 34978 DF,  p-value: < 2.2e-16
```

```
### Predictions and Evaluation
library(Metrics)
```

```
##
## Attaching package: 'Metrics'
##
## The following objects are masked from 'package:caret':
##
##     precision, recall
```

```
# Predictions
tryCatch({
  predictions <- predict(model, newdata = testData)

  # Alternative RMSE calculation
  rmse_value <- rmse(predictions, testData$price)

  # Correlation between predictions and actual values
  correlation <- cor(predictions, testData$price)

  # Print results
  print("Evaluation Metrics:")
  print(paste("RMSE: ", rmse_value))
  print(paste("Correlation: ", correlation))
```

```
}, error = function(e) {
  print("Error during prediction or evaluation:")
  print(e)
})
```

```
## [1] "Evaluation Metrics:"
## [1] "RMSE:  1119.18715031787"
## [1] "Correlation:  0.959016523423911"
```

```
# Normalize data and perform KNN
library(class)
normData <- scale(data[, sapply(data, is.numeric)])
knn_result <- knn(
  train = normData[trainIndex, ],
  test = normData[-trainIndex, ],
  cl = data$cut[trainIndex],
  k = 5
)
knn_result[1:5]
```

```
## [1] Premium   Premium   Ideal     Fair      Very Good
## Levels: Fair Good Ideal Premium Very Good
```

```
# Train and predict using C5.0 model
library(C50)
trainData$cut <- as.factor(trainData$cut)
testData$cut <- as.factor(testData$cut)
c50_model <- C5.0(trainData[, -which(names(trainData) == "cut")], trainData$cut)
c50_pred <- predict(c50_model, testData)
head(c50_pred)
```

```
## [1] Good      Premium   Ideal     Fair      Very Good Ideal
## Levels: Fair Good Ideal Premium Very Good
```

```
# Train and predict using ANN
library(nnet)
ann_model <- nnet(cut ~ ., data = trainData, size = 10, linout = FALSE)
```

```
## # weights:  265
## initial  value 68680.363251
## iter  10 value 49600.374579
## final  value 48109.231198
## converged
```

```
ann_pred <- predict(ann_model, testData, type = "class")
head(ann_pred)
```

```
## [1] "Ideal" "Ideal" "Ideal" "Ideal" "Ideal" "Ideal"
```