ISDS 540 Group 02 – Rating Wines from the Piedmont Region of Italy

Edward De Avila, Sheetal Padmanabhan, Ishani Parkar, Salvador Rodriguez, Martin Zapata

1. Introduction

The *Wine Spectator* magazine reviewed and scored 475 wines from the Piedmont region of Italy using a 100-point scale (*Wine Spectator*, April 30, 2011). The following table shows how the *Wine Spectator* score each wine received is used to rate each wine as being ‘Classic’, ‘Outstanding’, ‘Very good’, ‘Good’, ‘Mediocre’, or ‘Not recommended’.

Score	Rating
95 – 100	Classic: a great wine
90 – 94	Outstanding: a wine of superior character and style
85 – 89	Very good: a wine with special qualities
80 - 84	Good: a solid, well-made wine
75 – 79	Mediocre: a drinkable wine that may have minor flaws
Below 75	Not Recommended

A key question for most consumers is whether paying more for a bottle of wine will result in a better wine. To investigate this question for wines from the Piedmont region we selected a random sample of 100 of the 475 wines that *Wine Spectator* reviewed. The data, contained in the file named “WineRatings.csv”, shows the Price ($), the Wine Spectator Score, and the Rating for each wine. Our goal is to use linear regression models to predict the score given the price of the wine.

2. Data Review

Reading the dataset into R:

```
rm(list=ls())
Wine <- read.csv("D:/Fall'23/ISDS 540/Group Project/WineRatings.csv")
```

A sample of top 10 records is shown below:

```
head(Wine, 10)

##    Number                         Wine Price Score      Rating
## 1       1                  Barolo 2006    52    91 Outstanding
## 2       2   Barbera d'Alba Superiore 2007    30    90 Outstanding
## 3       3 Barbera Piemonte Castelvero 2009    10    80        Good
## 4       4              Barolo Cannubi 2006   120    93 Outstanding
## 5       5            Barolo San Pietro 2006    75    91 Outstanding
## 6       6           Chardonnay Langhe 2009    20    83        Good
## 7       7                  Gavi Minaia 2009    39    87   Very Good
## 8       8               Langhe Faletto 2007    50    91 Outstanding
## 9       9      Favorita Langhe White 2008    18    87   Very Good
## 10     10  Langhe Brandini & Brandini 2008    31    88   Very Good
```

Select the ‘Price’ and ‘Score’ columns, i.e., the independent and dependent variables, respectively.

```
WineNew <- Wine[,(3:4)]
```

A sample of the top ten records.

```
head(WineNew, 10)
```

```
##    Price Score
## 1     52    91
## 2     30    90
## 3     10    80
## 4    120    93
## 5     75    91
## 6     20    83
## 7     39    87
## 8     50    91
## 9     18    87
## 10    31    88
```

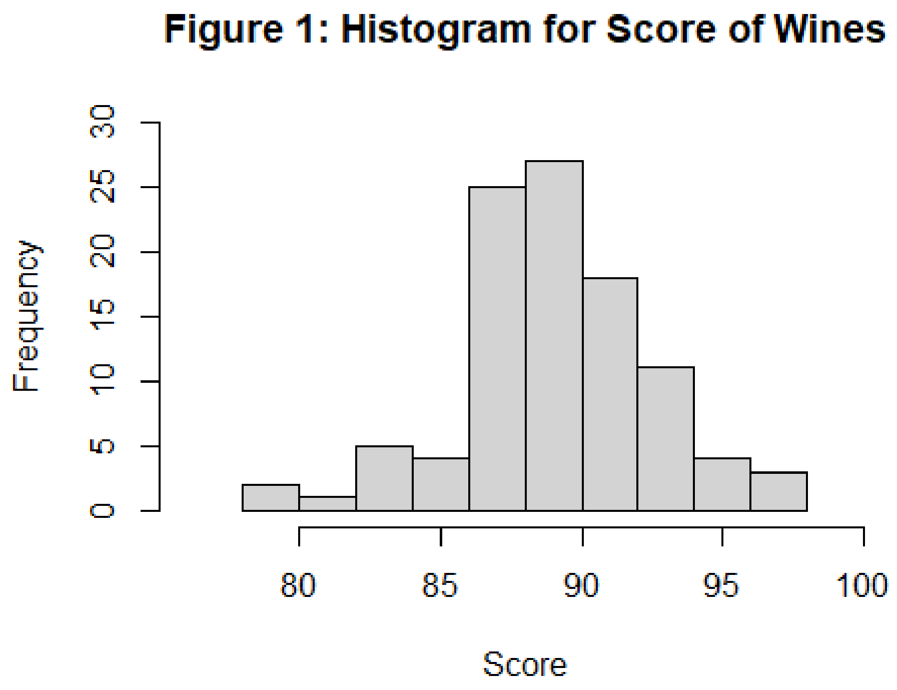
The final dataset has two variables.

- *Dependent Variable:* **Score**, an ordinal value indicating the comparative Wine quality between 0 and 100.
- *Independent Variable:* **Price**, a continuous variable indicating the price of a bottle of wine from 0 to 440 dollars (for the sample data of 100).

a. Exploratory Data Analysis

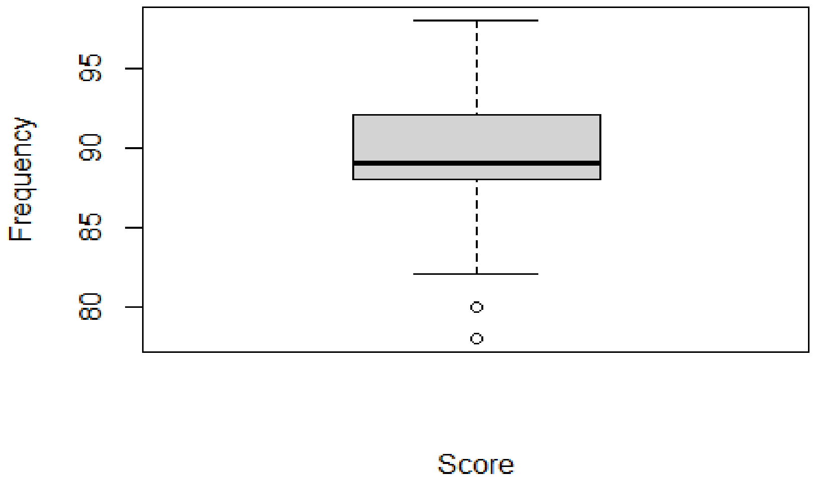
In the following Histogram, we see that the ‘Score’ variable is almost normally distributed, with the score between 86 to 92 having higher frequencies.

```
## Histogram

hist(WineNew$Score, xlim=c(76, 100), ylim=c(0, 30), xlab="Score",
    main="Figure 1: Histogram for Score of Wines")
```



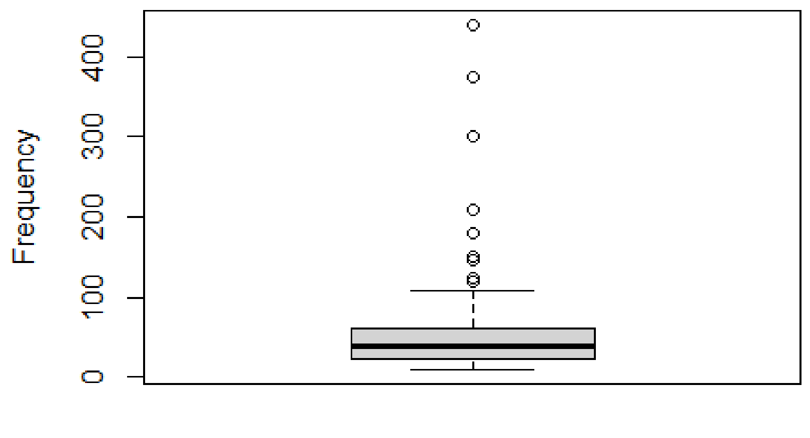
In the ‘Score’ boxplot below we see that there are very few outliers and majority of the score values are contained between the 1st and 3rd Quartile.

```
## Boxplot

boxplot(WineNew$Score)
```



The ‘Price’ boxplot below clearly indicates that the higher priced wines are considered outliers relative to the sample data and indicates that the Price variable is skewed. While some methodologies will remove these outliers from the data, for the purposes of this Case Study we have elected to leave them in the data set.

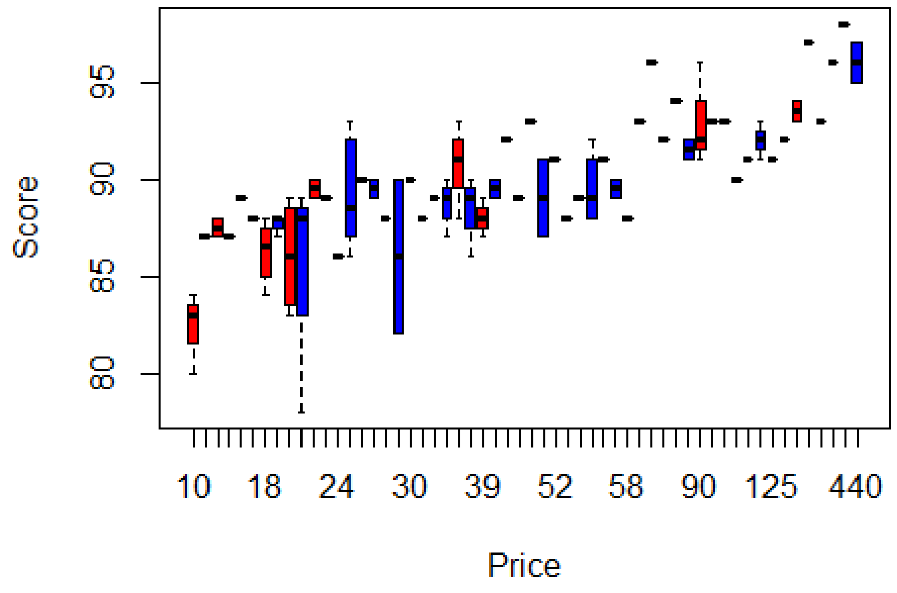
```
boxplot(WineNew$Price)
```



Price

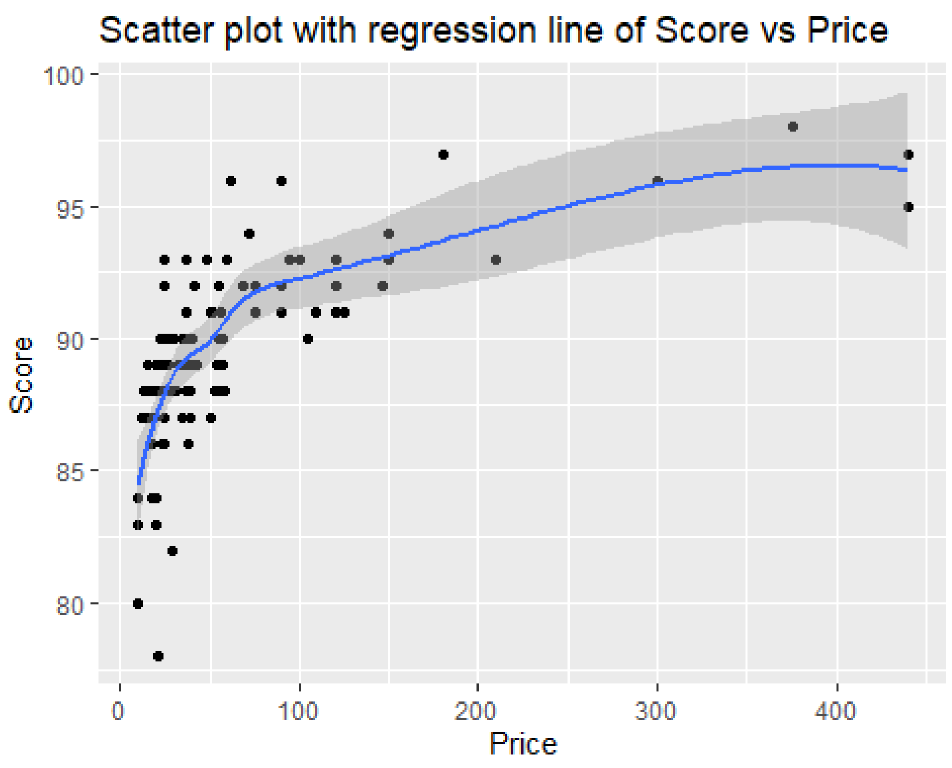
In the following boxplot, we see that most wines that score through 90 have a price range roughly between $20 and $80. Above a score of 90, we observe greater price variability with the top-rated wines being highly distributed between $80 and $440 dollars.

```
boxplot(Score ~ Price, data = WineNew, col = c("red", "blue"))
```



This scatterplot of ‘Score’ vs ‘Price’ below indicates a quadratic relationship between the variables.

```r
## Scatterplot

ggplot(WineNew, aes(Price, Score)) + labs(title = "Scatter plot with regressi
on line of Score vs Price") + geom_point() + geom_smooth()
```



```
# Price and Score have kind of a Quadratic relation #
```

3. Linear Regression: First Attempt

The model includes the linear regression functionality based on the sample ‘Price’ and ‘Score’ data.

```
#### Fit Initial Model ####

Wine.fit <- lm(Score ~ Price, data = WineNew)
summary(Wine.fit)
```

```
##
## Call:
## lm(formula = Score ~ Price, data = WineNew)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10.3511  -1.1412   0.1332   1.4879   6.5011
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 87.763226   0.342458 256.275  < 2e-16 ***
## Price        0.027995   0.003419   8.187 1.01e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.659 on 98 degrees of freedom
## Multiple R-squared:  0.4062, Adjusted R-squared:  0.4001
## F-statistic: 67.03 on 1 and 98 DF,  p-value: 1.01e-12
```

By looking at the R output, we know that the Price variable is significant (since the p-value is 1.01e-12).
Let's see if we can further improve our model.

4. Compare models using the test data

We want to build a linear model for predicting and explaining 'Score'. We partition the data into training (80%) and test (20%) sets. The training set is used to generate the predictive equation that is applied to the testing set.

Based on the observations from the scatter plot, we created few new variables to test and identify the best model.

```
#### Creating new variables ####

WineNew$Price_Log = log(WineNew$Price)
WineNew$Score_Log = log(WineNew$Score)
WineNew$Price_Sq = (WineNew$Price)^2
WineNew$Score_Sq = (WineNew$Score)^2


#### Fixing the train and test data sets ####

n = nrow(WineNew)
p = ncol(WineNew)
set.seed(456)
```

```
train.index <- sample(row.names(WineNew), floor(0.8*n))
test.index <- setdiff(row.names(WineNew), train.index)
train.df <- WineNew[train.index,]
test.df <- WineNew[test.index,]
```

a. Model 1

In Model 1, we fit a linear model using the predictor with the training set. Then we fit this model to the test set and calculate the RMSE.

```
#### Training and fitting Model 1 for train and test data sets ####

# Training the model

mod1 <- lm(Score ~ Price, data = train.df)
summary(mod1)

##
## Call:
## lm(formula = Score ~ Price, data = train.df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.1883 -1.2812  0.1482  1.5176  5.6885
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 87.922881   0.363521 241.865  < 2e-16 ***
## Price        0.026540   0.003515   7.551 6.95e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.527 on 78 degrees of freedom
## Multiple R-squared:  0.4223, Adjusted R-squared:  0.4149
## F-statistic: 57.01 on 1 and 78 DF,  p-value: 6.946e-11

# Fit the model on test set

mod1_test <- lm(Score~ Price, data = test.df)
summary(mod1_test)

##
## Call:
## lm(formula = Score ~ Price, data = test.df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.8131 -0.7673  0.4160  1.2591  6.7062
```

```
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 87.05477    0.94007  92.604   <2e-16 ***
## Price        0.03611    0.01095   3.298    0.004 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.218 on 18 degrees of freedom
## Multiple R-squared:  0.3766, Adjusted R-squared:  0.342
## F-statistic: 10.87 on 1 and 18 DF,  p-value: 0.004002

# Model 1 RMSE

preds.mod1 <- predict(mod1, newdata = test.df)
MSE1 <- mean((preds.mod1 - test.df$Score)^2)

RMSE1 <- sqrt(MSE1)
print(RMSE1)

## [1] 3.135059
```

b. Model 2

The next model iteration includes using the log and square of the Price and Score data, respectively.

```
#### Training and fitting Model 2 for train and test data sets ####

# Training the model

mod2 <- lm(Score_Sq ~ Price_Log, data = train.df)
summary(mod2)

##
## Call:
## lm(formula = Score_Sq ~ Price_Log, data = train.df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1097.99  -270.56    37.52   223.19   910.18
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  5935.09     192.07   30.90   <2e-16 ***
## Price_Log     560.36      49.76   11.26   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 367.2 on 78 degrees of freedom
## Multiple R-squared:  0.6191, Adjusted R-squared:  0.6142
## F-statistic: 126.8 on 1 and 78 DF,  p-value: < 2.2e-16

# Fit the model on test set

mod2_test <- lm(Score_Sq ~ Price_Log, data = test.df)
summary(mod2_test)

##
## Call:
## lm(formula = Score_Sq ~ Price_Log, data = test.df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1520.93  -204.10    39.12   259.67   963.08
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   5782.7      534.5  10.820 2.62e-09 ***
## Price_Log      598.5      144.5   4.142 0.000612 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 513.1 on 18 degrees of freedom
## Multiple R-squared:  0.488,  Adjusted R-squared:  0.4596
## F-statistic: 17.16 on 1 and 18 DF,  p-value: 0.0006121

# Model 2 RMSE

preds.mod2 <- predict(mod2, newdata = test.df)
preds2.Score <- sqrt(preds.mod2)
MSE2 <- mean((preds2.Score - test.df$Score)^2)

RMSE2 <- sqrt(MSE2)
print(RMSE2)

## [1] 2.829256
```

We see that the Root Mean Squared Error has dropped significantly when the new variables are used.

c. Model 3

The next model iteration includes using the log and square of the Price and the Score variable as it is.

```r
#### Training and fitting Model 3 for train and test data sets ####

# Training the model

mod3 <- lm(Score ~ Price_Log + Price_Sq, data = train.df)
summary(mod3)

##
## Call:
## lm(formula = Score ~ Price_Log + Price_Sq, data = train.df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.3927 -1.5096  0.2054  1.3183  5.0801
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  7.766e+01  1.380e+00  56.263  < 2e-16 ***
## Price_Log    3.189e+00  3.776e-01   8.445 1.41e-12 ***
## Price_Sq    -2.886e-06  9.230e-06  -0.313    0.755
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.085 on 77 degrees of freedom
## Multiple R-squared:  0.6118, Adjusted R-squared:  0.6017
## F-statistic: 60.68 on 2 and 77 DF,  p-value: < 2.2e-16

# Fit the model on test set

mod3_test <- lm(Score ~ Price_Log + Price_Sq, data = test.df)
summary(mod3_test)

##
## Call:
## lm(formula = Score ~ Price_Log + Price_Sq, data = test.df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.1454 -1.1081  0.2242  1.5339  5.1884
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  7.680e+01  4.575e+00  16.789 5.12e-12 ***
## Price_Log    3.398e+00  1.334e+00   2.547   0.0208 *
## Price_Sq    -3.619e-06  5.386e-05  -0.067   0.9472
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.06 on 17 degrees of freedom
## Multiple R-squared:  0.4677, Adjusted R-squared:  0.4051
## F-statistic: 7.468 on 2 and 17 DF,  p-value: 0.004703

# Model 3 RMSE

preds.mod3 <- predict(mod3, newdata = test.df)
MSE3 <- mean((preds.mod3 - test.df$Score)^2)

RMSE3 <- sqrt(MSE3)
print(RMSE3)

## [1] 2.826958
```

We see that when fitting the model for train set the square of Price variable is insignificant since p-value is much higher than 0.05 that is 5% significance level. So, we proceed with models with only log of Price variable.

d. Model 4

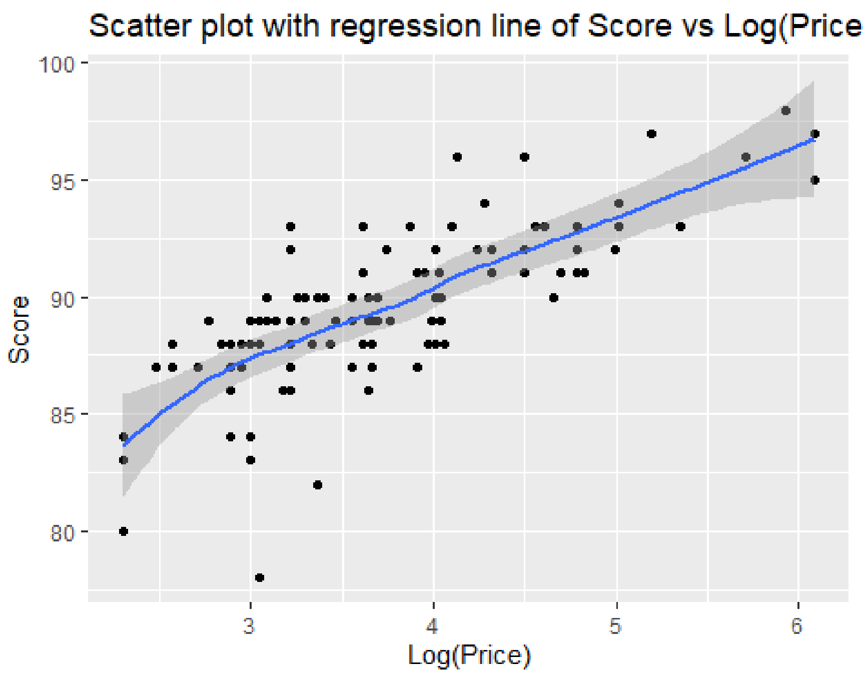
The next model iteration includes using the log of the Price and the Score variable as it is.

```
#### Training and fitting Model 4 for train and test data sets ####

# Training the model

mod4 <- lm(Score ~ Price_Log, data = train.df)
summary(mod4)

##
## Call:
## lm(formula = Score ~ Price_Log, data = train.df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.3958 -1.4586  0.1649  1.2806  5.0659
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  77.9201     1.0841   71.88   <2e-16 ***
## Price_Log     3.1110     0.2809   11.08   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.073 on 78 degrees of freedom
## Multiple R-squared:  0.6113, Adjusted R-squared:  0.6063
## F-statistic: 122.7 on 1 and 78 DF,  p-value: < 2.2e-16
```

```
# Fit the model on test set

mod4_test <- lm(Score ~ Price_Log, data = test.df)
summary(mod4_test)

##
## Call:
## lm(formula = Score ~ Price_Log, data = test.df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.1592 -1.0583  0.2345  1.5163  5.2364
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  77.0229     3.0974  24.867 2.18e-15 ***
## Price_Log     3.3294     0.8374   3.976 0.000886 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.974 on 18 degrees of freedom
## Multiple R-squared:  0.4676, Adjusted R-squared:  0.438
## F-statistic: 15.81 on 1 and 18 DF,  p-value: 0.0008865

# Model 4 RMSE

preds.mod4 <- predict(mod4, newdata = test.df)
MSE4 <- mean((preds.mod4 - test.df$Score)^2)

RMSE4 <- sqrt(MSE4)
print(RMSE4)

## [1] 2.828516
```

The RMSE value is marginally better than Model 3. Let’s look at the scatter plot with the regression line for Score vs Log(Price).

```
# Scatter Plot for Score ~ Log(Price)

ggplot(WineNew, aes(Price_Log, Score)) +
  labs(title = "Scatter plot with regression line of Score vs Log(Price)", x
= "Log(Price)", y = "Score") + geom_point() + geom_smooth()
```



We can see that the relation between Score and Log(Price) is almost linear.

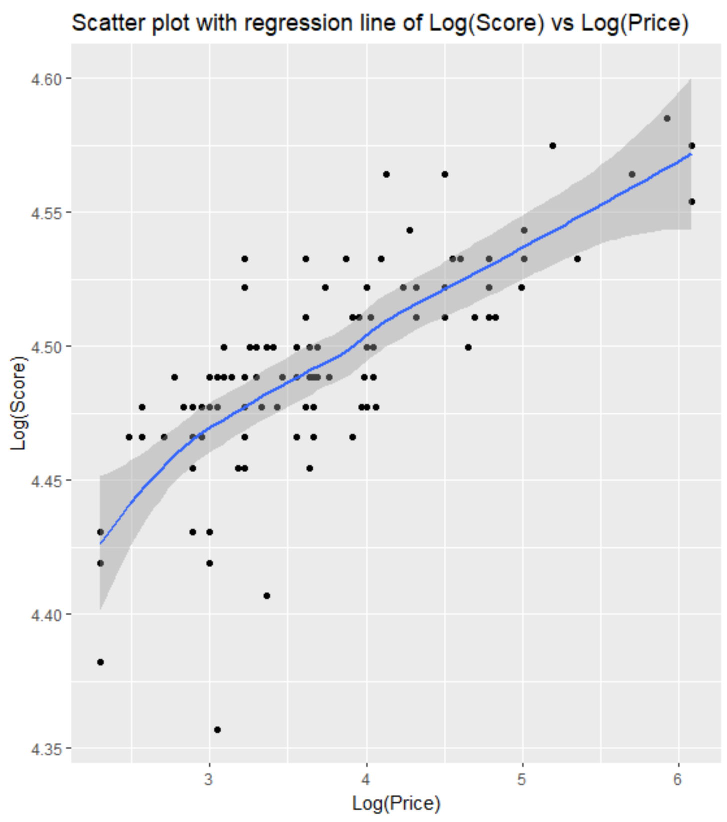
e. Model 5

The next model iteration includes using the log of Price and Score.

```
#### Training and fitting Model 5 for train and test data sets ####

# Training the model

mod5 <- lm(Score_Log ~ Price_Log, data = train.df)
summary(mod5)

##
## Call:
## lm(formula = Score_Log ~ Price_Log, data = train.df)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.074572 -0.015752  0.001745  0.014672  0.056443
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 4.364795   0.012284  355.33   <2e-16 ***
## Price_Log   0.034597   0.003183   10.87   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.02349 on 78 degrees of freedom
## Multiple R-squared:  0.6024, Adjusted R-squared:  0.5973
## F-statistic: 118.2 on 1 and 78 DF,  p-value: < 2.2e-16

# Fit the model on test set

mod5_test <- lm(Score_Log ~ Price_Log, data = test.df)
summary(mod5_test)

##
## Call:
## lm(formula = Score_Log ~ Price_Log, data = test.df)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.110451 -0.010927  0.002818  0.017744  0.057013
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 4.354181   0.036088 120.656   <2e-16 ***
## Price_Log   0.037109   0.009757   3.803   0.0013 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.03465 on 18 degrees of freedom
## Multiple R-squared:  0.4456, Adjusted R-squared:  0.4148
## F-statistic: 14.47 on 1 and 18 DF,  p-value: 0.001302

# Model 4 RMSE

preds.mod5 <- predict(mod5, newdata = test.df)
preds5.Score = exp(preds.mod5)
MSE5 <- mean((preds5.Score - test.df$Score)^2)

RMSE5 <- sqrt(MSE5)
print(RMSE5)

## [1] 2.828338
```

The RMSE value is marginally better than Model 4.

```
# Scatter Plot for Log(Score) ~ Log(Price)

ggplot(WineNew, aes(Price_Log, Score_Log)) +
  labs(title = "Scatter plot with regression line of Log(Score) vs Log(Price)
", x = "Log(Price)", y = "Log(Score)") +
  geom_point() + geom_smooth()
```



We can observe that the graph between Score ~ Log(Price) and Log(Score) ~ Log(Price) is almost similar. But when we look at the Root Mean Squared Error, we can see that it is marginally better than the one for Model 4.

```
#### Printing Mean Squared Error and Root Mean Squared Error for all models ####

print(c(MSE1, MSE2, MSE3, MSE4))

## [1] 9.828592 8.004691 8.000503 7.999494

print(c(RMSE1, RMSE2, RMSE3, RMSE4))

## [1] 3.135059 2.829256 2.828516 2.828338
```

f. Final Model

The model with the least RMSE is the best to predict our dependent variable – Score. We will be using Model 5 as the final model to predict Score using the Price of a bottle of Wine. To obtain the estimated regression equation, we can fit model 4 again using the combined training and test data.

```
#### Final model will be used to predict Score ####
```

```
full.df = rbind(train.df, test.df)
mod.final <- lm(Score_Log ~ Price_Log, data = full.df)
summary(mod.final)
```

```
##
## Call:
## lm(formula = Score_Log ~ Price_Log, data = full.df)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.112742 -0.014463  0.001817  0.014534  0.057024
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 4.362516   0.011978  364.22   <2e-16 ***
## Price_Log   0.035124   0.003129   11.22   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0257 on 98 degrees of freedom
## Multiple R-squared:  0.5625, Adjusted R-squared:  0.5581
## F-statistic:   126 on 1 and 98 DF,  p-value: < 2.2e-16
```

The resulting estimated regression equation is as follows:

$$log\left(\widehat{Score}\right) = 4.363 + \ 0.035 * \log\left(WinePrice\right)$$

.
We will use this regression equation to predict the quality Score for a Wine based on the Price. With an Adjusted R-Square score of 0.5581

5. Conclusion

The above equation predicts the score correctly only 56% of times using the Price of the bottle of Wine. We conclude that if we collect more data with regards to the vinification process, the quality of grapes and more, then we can build a model that would help us predict the Score of a Wine i.e., the quality of a wine more accurately.