# Student Performance Analysis

Matthew Boodhoo

## Project Outline

We will be analyzing the following Kaggle dataset:
https://www.kaggle.com/datasets/nikhil7280/student-performance-multiple-linear-regression/data

The above dataset contains the following variables:
**Dependent**: Performance Index
**Independent**: Hours Studied, Previous Scores, Extracurricular Activities, Sleep Hours, Sample Question Papers Practiced

Our goal is to determine a model that accurately fits and predicts the performance index given the independent variables in the dataset.

We will approach this in the following steps:

1. Clean the data
2. Split into training and test sets
3. Explore the data
4. Fit regression models
5. Residual Diagnostics
6. Assess predictive accuracy
7. Obtain prediction interval given new data using best model

## Cleaning the Data

```r
# Loading required packages
library(tidyverse)
library(skimr)
library(GGally)
library(corrplot)
library(psych)
library(gridExtra)
library(MASS)
library(car)
library(broom)
library(car)
```

```r
# Reading in Student Performance data
performance.data = read_csv('Student_Performance.csv')
```

```r
# Structure of data frame
glimpse(performance.data)
```

```
## Rows: 10,000
## Columns: 6
## $ `Hours Studied`                    <dbl> 7, 4, 8, 5, 7, 3, 7, 8, 5, 4, 8, 8,~
## $ `Previous Scores`                  <dbl> 99, 82, 51, 52, 75, 78, 73, 45, 77,~
## $ `Extracurricular Activities`       <chr> "Yes", "No", "Yes", "Yes", "No", "N~
## $ `Sleep Hours`                      <dbl> 9, 4, 7, 5, 8, 9, 5, 4, 8, 4, 4, 6,~
## $ `Sample Question Papers Practiced` <dbl> 1, 2, 2, 2, 5, 6, 6, 6, 2, 0, 5, 2,~
## $ `Performance Index`                <dbl> 91, 65, 45, 36, 66, 61, 63, 42, 61,~
```

```r
# Broad overview of data frame
skim(performance.data)
```

Table 1: Data summary

| | |
|---|---|
| Name | performance.data |
| Number of rows | 10000 |
| Number of columns | 6 |
| Column type frequency: | |
| character | 1 |
| numeric | 5 |
| Group variables | None |

**Variable type: character**

| skim_variable | n_missing | complete_rate | min | max | empty | n_unique | whitespace |
|---|---|---|---|---|---|---|---|
| Extracurricular Activities | 0 | 1 | 2 | 3 | 0 | 2 | 0 |

**Variable type: numeric**

| skim_variable | n_missing | complete_rate | mean | sd | p0 | p25 | p50 | p75 | p100 | hist |
|---|---|---|---|---|---|---|---|---|---|---|
| Hours Studied | 0 | 1 | 4.99 | 2.59 | 1 | 3 | 5 | 7 | 9 | |
| Previous Scores | 0 | 1 | 69.45 | 17.34 | 40 | 54 | 69 | 85 | 99 | |
| Sleep Hours | 0 | 1 | 6.53 | 1.70 | 4 | 5 | 7 | 8 | 9 | |
| Sample Question Papers Practiced | 0 | 1 | 4.58 | 2.87 | 0 | 2 | 5 | 7 | 9 | |
| Performance Index | 0 | 1 | 55.22 | 19.21 | 10 | 40 | 55 | 71 | 100 | |

```r
# Finding the number of duplicate rows
sum(duplicated(performance.data))
```

```
## [1] 127
```

We see that there are no missing values, and based on the generation of the data, we can say the duplicated values are not an issue (meaning they do not need to be removed), since there is no uniqueness requirement, for example in the form of unique student IDs.

We do, however, want to modify the "Extracurricular Activities" column. We will assign the "No" to 0 and the "Yes" to 1. This will help us significantly when calculating correlations later on.

```r
for (i in 1:10000) {
  if (performance.data$`Extracurricular Activities`[i] == "No") {
    performance.data$`Extracurricular Activities`[i] = 0
  } else {
    performance.data$`Extracurricular Activities`[i] = 1
  }
}
performance.data$`Extracurricular Activities` =
  as.numeric(performance.data$`Extracurricular Activities`)
```

We can proceed to split the data into training and test sets.

## Training and Test sets

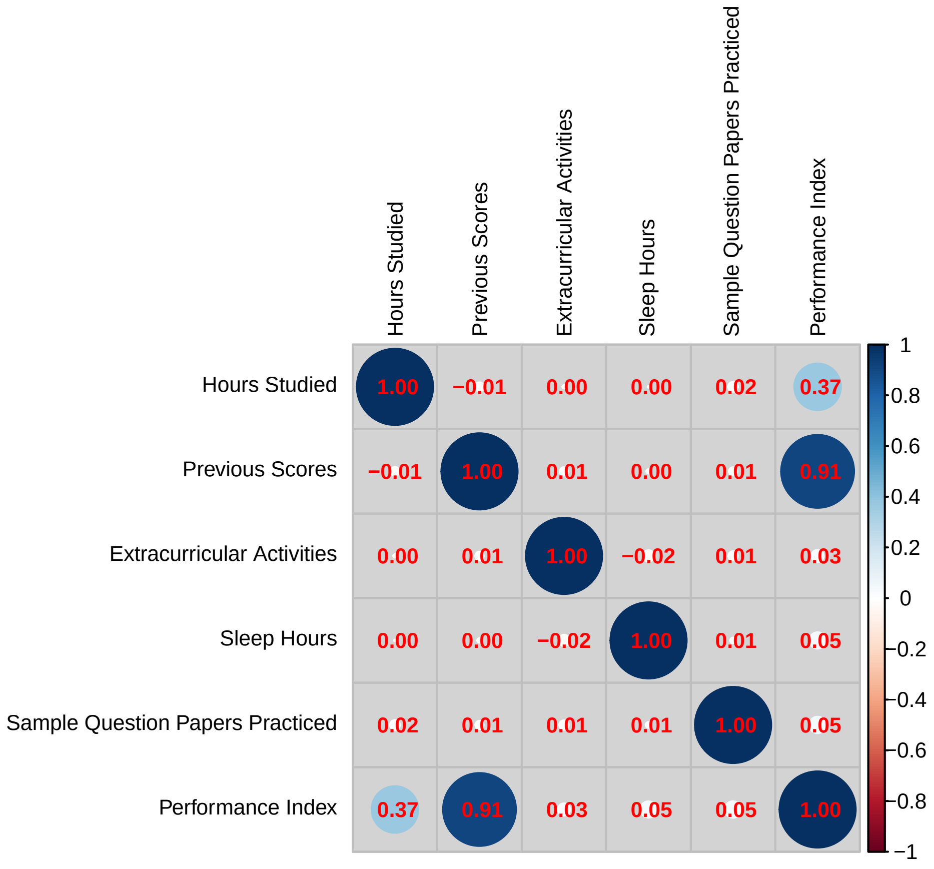
```r
# Creating training and test sets
set.seed(6)
index = sample(1:10000, 9000)
index = sort(index)
performance.data.training = performance.data[index,]
performance.data.test = performance.data[-index,]
```

We will use the training data for the next few sections until we start analyzing prediction power, where we will then use the test data.

## Exploring the Data

Below is the correlation matrix for the data.

```r
corrplot(cor(performance.data.training), method = 'circle', addCoef.col = 'red',
         tl.col = 'black', tl.cex = 0.7, number.cex = 0.7, cl.cex = 0.7,
         bg = 'lightgrey')
```



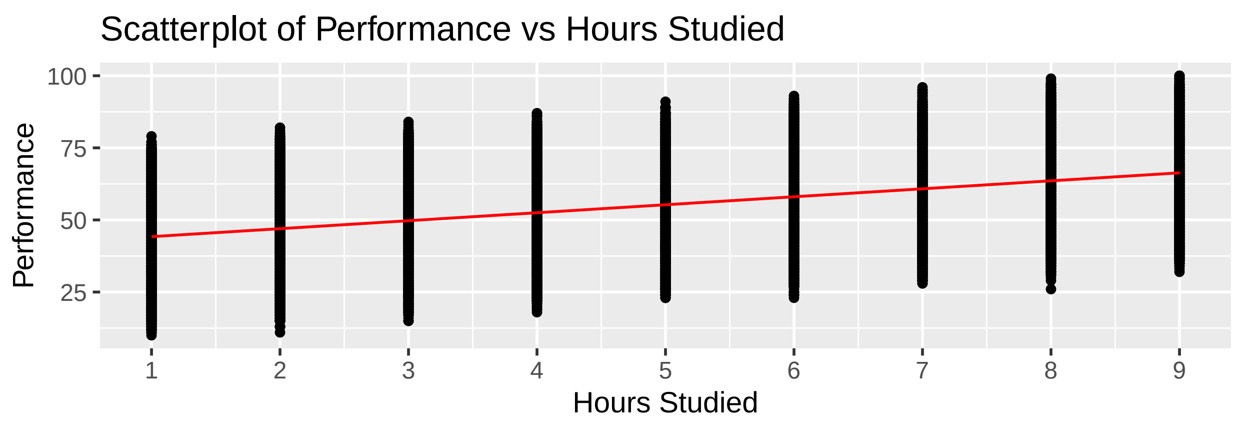
From the correlation plot, we see hours studied and previous test results seem to have a fair amount of correlation with performance. The other predictors may be dropped in the fitting process. Also, the explanatory variables don't seem to have much, if any, collinearity.

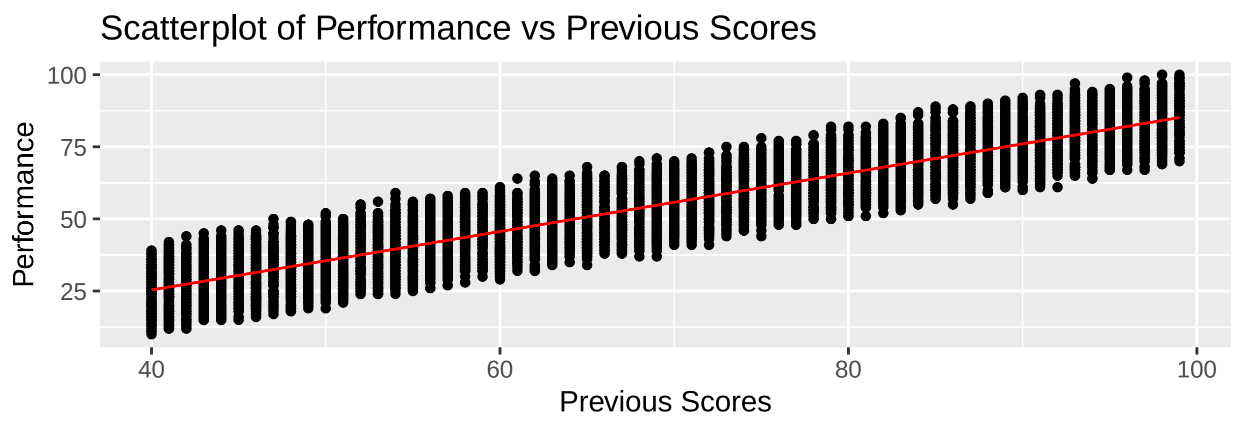
We now explore some visualizations of the notable relationships shown in the correlation matrix.

```r
# Scatterplot of Performance Index vs Hours Studied
x1 = performance.data.training$`Hours Studied`
y = performance.data.training$`Performance Index`

plot1 = ggplot(data = performance.data.training, aes(x=x1, y=y)) +
  geom_point(shape = 16) +
  geom_smooth(method = 'lm', se = FALSE, colour = 'red', linewidth = 0.5) +
  labs(title = 'Scatterplot of Performance vs Hours Studied', x = 'Hours Studied',
       y = 'Performance') +
  scale_x_continuous(breaks = seq(min(x1), max(x1), 1))

# Scatterplot of Performance Index vs Previous Scores
```

```
x2 = performance.data.training$`Previous Scores`
y = performance.data.training$`Performance Index`

plot2 = ggplot(data = performance.data.training, aes(x=x2, y=y)) +
  geom_point(shape = 16) +
  geom_smooth(method = 'lm', se = FALSE, colour = 'red', linewidth = 0.5) +
  labs(title = 'Scatterplot of Performance vs Previous Scores', x = 'Previous Scores',
       y = 'Performance')

grid.arrange(plot1, plot2, nrow = 2)
```

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





## Fitting Regression Models

```
# creating vectors for fit testing on the training data
x1.tr = as.vector(performance.data.training$`Hours Studied`)
x2.tr = as.vector(performance.data.training$`Previous Scores`)
x3.tr = as.vector(performance.data.training$`Extracurricular Activities`)
x4.tr = as.vector(performance.data.training$`Sleep Hours`)
x5.tr = as.vector(performance.data.training$`Sample Question Papers Practiced`)
```

```r
y.tr = as.vector(performance.data.training$`Performance Index`)
f.x3.tr = as.factor(x3.tr)

# creating vectors for prediction testing on the test data
x1.te = as.vector(performance.data.test$`Hours Studied`)
x2.te = as.vector(performance.data.test$`Previous Scores`)
x3.te = as.vector(performance.data.test$`Extracurricular Activities`)
x4.te = as.vector(performance.data.test$`Sleep Hours`)
x5.te = as.vector(performance.data.test$`Sample Question Papers Practiced`)
y.te = as.vector(performance.data.test$`Performance Index`)
f.x3.te = as.factor(x3.te)

df.tr = data.frame(Y = y.tr, X1 = x1.tr, X2 = x2.tr, X3 = f.x3.tr,
                   X4 = x4.tr, X5 = x5.tr)
df.te = data.frame(Y = y.te, X1 = x1.te, X2 = x2.te, X3 = f.x3.te,
                   X4 = x4.te, X5 = x5.te)
```

We will now perform multiple linear regression on the training data.

```r
# Multiple linear regression model
model1 = lm(Y ~., data = df.tr)
summary(model1)
```

```
##
## Call:
## lm(formula = Y ~ ., data = df.tr)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.6271 -1.3822 -0.0242  1.3521  8.7806
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -34.064784   0.134398 -253.46   <2e-16 ***
## X1            2.852189   0.008301  343.60   <2e-16 ***
## X2            1.018296   0.001242  820.16   <2e-16 ***
## X31           0.594223   0.043070   13.80   <2e-16 ***
## X4            0.482411   0.012695   38.00   <2e-16 ***
## X5            0.194457   0.007512   25.89   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.042 on 8994 degrees of freedom
## Multiple R-squared:  0.9887, Adjusted R-squared:  0.9887
## F-statistic: 1.574e+05 on 5 and 8994 DF,  p-value: < 2.2e-16
```

```r
# Variable selection using AIC
step_model = stepAIC(model1, direction = "both", trace = TRUE)
```

```
## Start:  AIC=12857.04
## Y ~ X1 + X2 + X3 + X4 + X5
##
```

```
##        Df Sum of Sq     RSS   AIC
## <none>                37504 12857
## - X3    1       794   38298 13044
## - X5    1      2794   40298 13502
## - X4    1      6021   43526 14195
## - X1    1    492314  529818 36688
## - X2    1   2804964 2842468 51807
```

```
# Variance inflation factor to check for multicollinearity
vif(model1)
```

```
##       X1       X2       X3       X4       X5
## 1.000450 1.000457 1.000830 1.000517 1.000645
```

We see that the p-values corresponding to each regression coefficient are extremely small, indicating they are all important. This is confirmed by the AIC variable selection process states all the explanatory variates are important.

This may be surprising given the correlation matrix only showed "hours studied" and "previous test scores" having meaningful correlation with the performance. However, it should be noted that the correlation matrix examined unconditional association between variables. Whereas, linear regression examines the conditional association between the independent variables and the dependent variable. As such, it is entirely possible for variables with low unconditional correlation to have high conditional correlation.

We also see that the VIF values for each variable are very low which, along with the correlation matrix examined earlier, indicates no multicollinearity in the data.

We will propose a smaller model with only hours studied and previous test scores as the predictors for future use in assessing prediction power. That said, we can also compare our models to test if they are significantly different in terms of fit.

```
# Reduced model with only highly correlated variables
model2 = lm(Y ~ X1 + X2, data = df.tr)
summary(model2)
```

```
##
## Call:
## lm(formula = Y ~ X1 + X2, data = df.tr)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -8.287 -1.537 -0.005  1.532  9.155
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -29.790317   0.110401  -269.8   <2e-16 ***
## X1            2.855435   0.009300   307.0   <2e-16 ***
## X2            1.019024   0.001391   732.5   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.288 on 8997 degrees of freedom
## Multiple R-squared:  0.9858, Adjusted R-squared:  0.9858
## F-statistic: 3.125e+05 on 2 and 8997 DF,  p-value: < 2.2e-16
```

```r
# Analysis of Variance Table comparing both models
anova(model2, model1)
```

```
## Analysis of Variance Table
##
## Model 1: Y ~ X1 + X2
## Model 2: Y ~ X1 + X2 + X3 + X4 + X5
##   Res.Df   RSS Df Sum of Sq      F    Pr(>F)
## 1   8997 47110
## 2   8994 37504  3    9605.5 767.84 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Comparing the two models, we see that have very similar R-squared and Adjusted R-squared values. This indicates they both provide a good fit to the data. However, the anova table has a very small p-value for the F-test of significance for the missing variables. This indicates the fit of the larger model is statistically significantly better than the reduced model.

Now that we have established the larger model is better, let us make sense of the coefficient estimates. Firstly, note that all of the coefficients of regression are positive, except for the intercept. However, in this case, the intercept does not provide any useful information, so we can ignore it when interpreting our regression summary. Going in the order presented in the summary output, we have:

1. For every 1 hr increase in study time, there is an expected 2.852189 increase in performance, all else being equal.
2. For every 1 mark increase in previous score, there is an expected 1.018296 increase in performance, all else being equal.
3. For students who participate in extracurricular activities, there is an expected 0.594223 increase in performance over students who don't participate in extracurricular activities, all else being equal.
4. For every 1 hr increase in sleep, there is an expected 0.482411 increase in performance, all else being equal.
5. For every 1 sample paper practiced increase, there is an expected 0.194457 increase in performance, all else being equal.

All of the predictors have a positive impact on student performance.

We will now investigate the residual diagnostics of both models to see if they are also suitable models for inference purposes (namely prediction intervals).

## Residual Diagnostics

We want to check both models to see if they satisfy the regression assumptions for inference purposes. That is:

1. The error terms are Normally distributed
2. The error terms have a mean of zero
3. The error terms have a constant variance
4. The error terms are independent of each other

We will extract the residuals from both models,

```r
# residuals from model1 and model2
res1 = model1$residuals
res2 = model2$residuals

# fitted values from model1 and model2
fitted1 = model1$fitted.values
fitted2 = model2$fitted.values
```

## Model 1

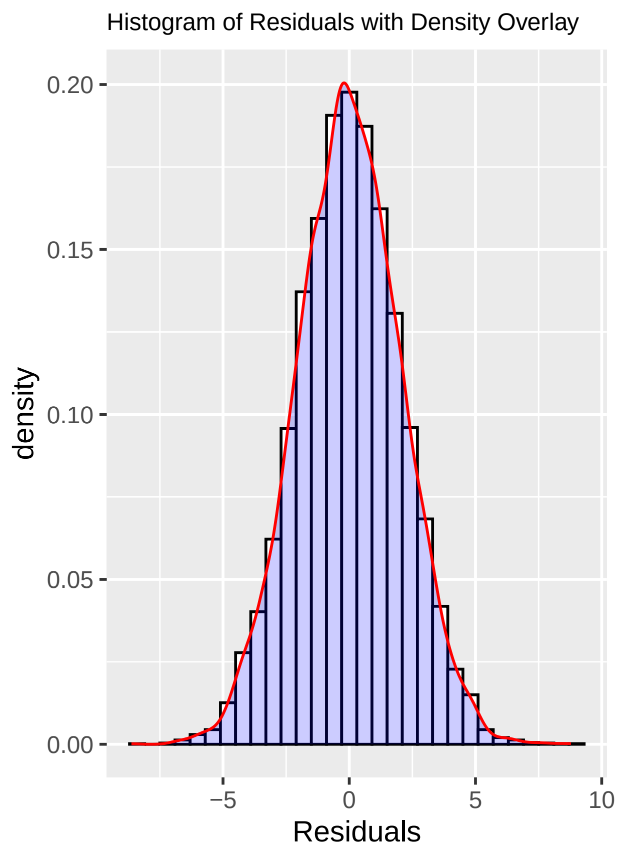
We will test the full model first.

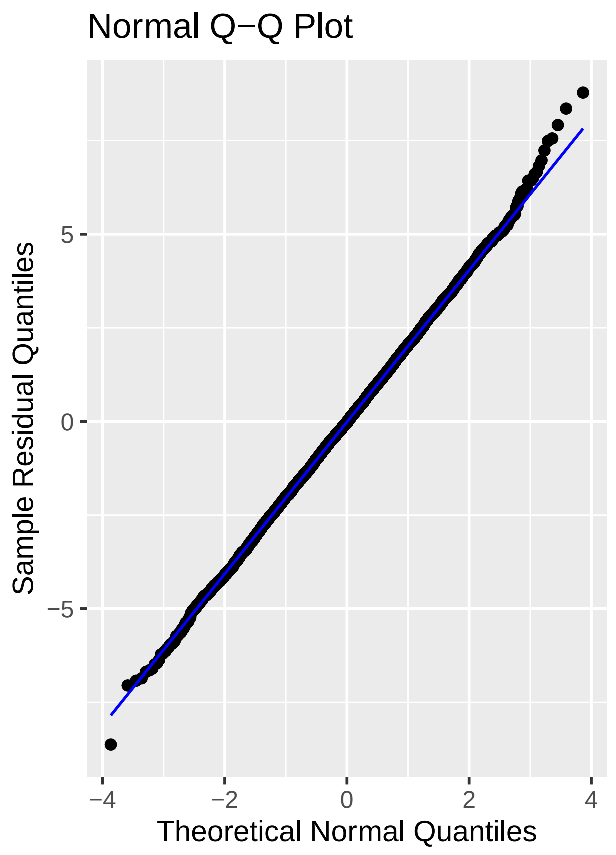
## Normality

To test this, we will look at both a histogram and Normal q-q plot of the data.

```r
df.1 = data.frame(res1, fitted1)

# Histogram of residuals
fig1 = ggplot(df.1, aes(x=res1)) +
  geom_histogram(aes(y = after_stat(density)), color = 'black', fill = 'white') +
  geom_density(color = 'red', fill = 'blue', alpha = 0.2) +
  labs(title = 'Histogram of Residuals with Density Overlay', x = 'Residuals') +
  theme(plot.title = element_text(size = 9))

# Normal Q-Q plot of residuals
fig2 = ggplot(df.1, aes(sample = res1)) +
  geom_qq() + geom_qq_line(color = 'blue') +
  labs(title = 'Normal Q-Q Plot', x = 'Theoretical Normal Quantiles',
       y = 'Sample Residual Quantiles')

grid.arrange(fig1, fig2, ncol=2)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



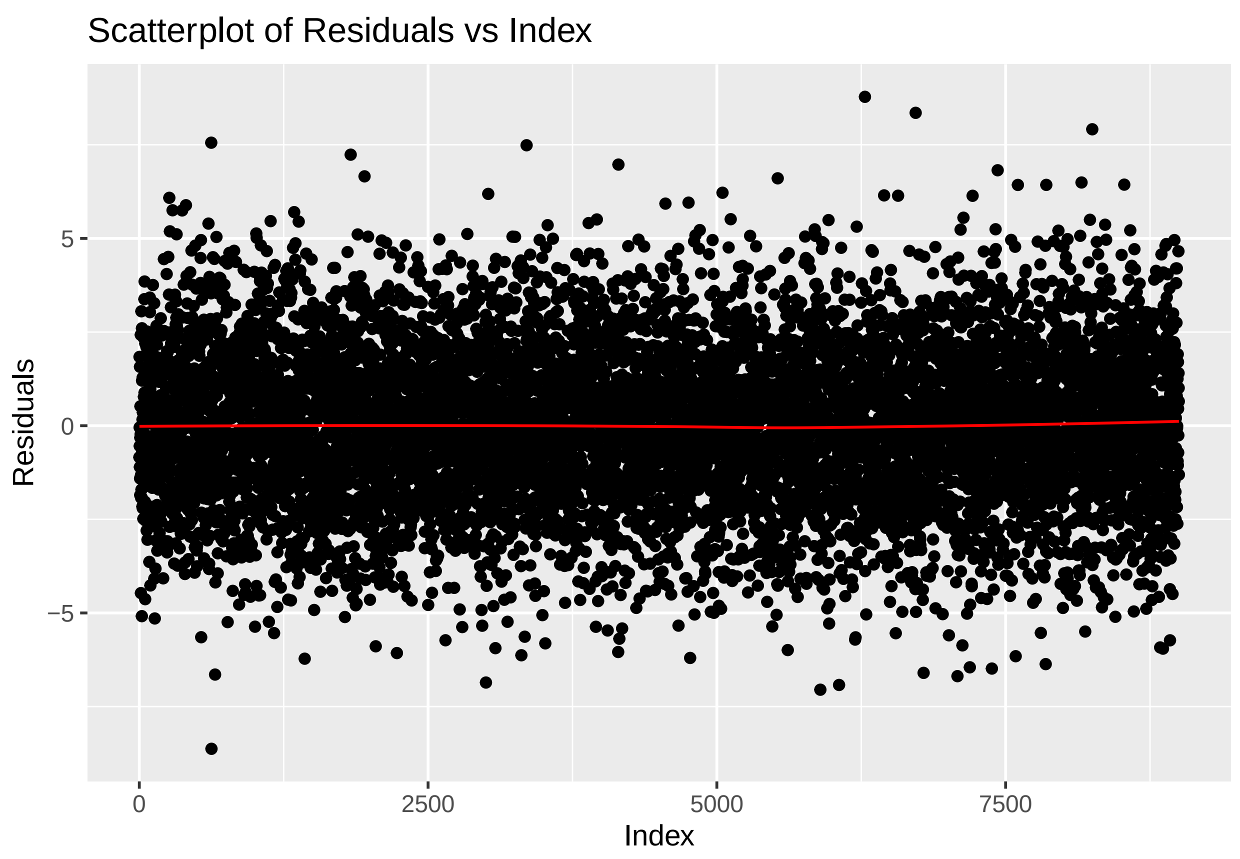


We can see the histogram has the classic bell shape we expect from a Normal Distribution. Also, the points seem to lie fairly well along the Q-Q plot, with no visible concerns. From this, we can say the Normality assumption is satisfied.

## Mean of Zero

```r
# Scatterplot of Residuals vs Index
index = 1:length(res1)
df.1$index = index
ggplot(df.1, aes(x = index, y = res1)) +
  geom_point() +
  geom_smooth(method = 'loess', se = FALSE, colour = 'red', linewidth = 0.5) +
  labs(title = 'Scatterplot of Residuals vs Index', x = 'Index', y = 'Residuals')
```

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



```r
mean(res1)
```

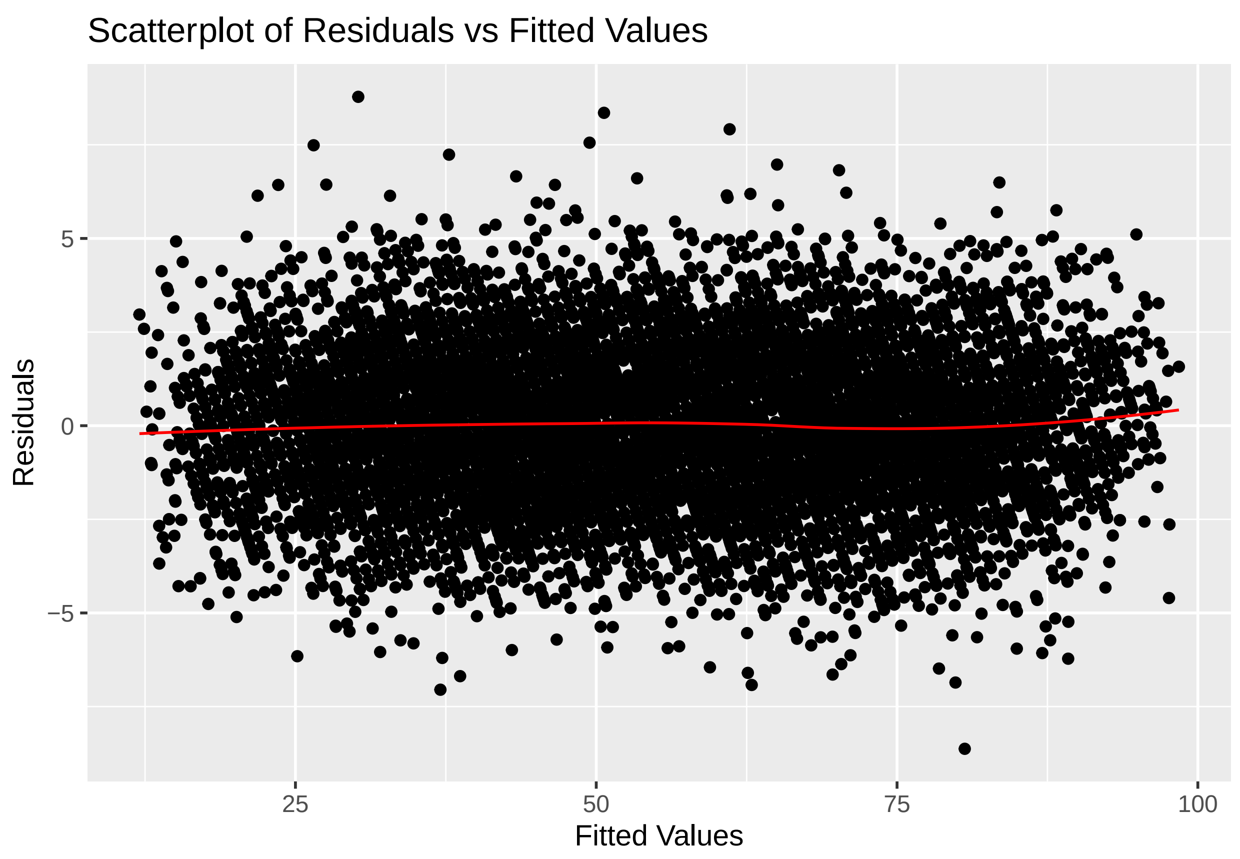
```
## [1] 2.925303e-17
```

We see from the above scatterplot that the points seem to be evenly spread about the horizontal line at zero, with no discernible pattern. We also see the mean of the residuals is very close to zero. We can say the zero mean assumption is satisfied.

### Constant Variance

```r
# Residuals vs Fitted Values
ggplot(df.1, aes(x = fitted1, y = res1)) +
  geom_point() +
  geom_smooth(method = 'loess', se = FALSE, colour = 'red', linewidth = 0.5) +
  labs(title = 'Scatterplot of Residuals vs Fitted Values', x = 'Fitted Values',
       y = 'Residuals')
```

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



We see there does not seem to be any change in spread of the data as we move right along the x-axis. This indicates the variance of the residuals is remaining constant throughout. The constant variance assumption is satisfied.

## Independence

If we re-examine the residuals vs fitted values scatterplot, we do not see any pattern, which indicates the residuals are independent of the independent variable.

Next, we check the correlation between the residuals and the predictor variables,

```
data.frame(cor(x = res1, y = performance.data.training[,1:5]), row.names = c('res1'))
```

```
##      Hours.Studied Previous.Scores Extracurricular.Activities   Sleep.Hours
## res1  2.238141e-17    3.114708e-16               2.065649e-16 -4.523403e-16
##      Sample.Question.Papers.Practiced
## res1                    -1.030257e-16
```

We see that they all have extremely low correlation with the residuals.

We will now perform two formal tests of randomness to further test for independence among the residuals.

```
randtests::difference.sign.test(res1)
```

```
##
```

```
##  Difference Sign Test
##
## data:  res1
## statistic = -0.45641, n = 9000, p-value = 0.6481
## alternative hypothesis: nonrandomness
```

```r
randtests::runs.test(res1)
```

```
##
##  Runs Test
##
## data:  res1
## statistic = 0, runs = 4501, n1 = 4500, n2 = 4500, n = 9000, p-value = 1
## alternative hypothesis: nonrandomness
```

All of our analysis has not presented any evidence against independence of the residuals. Therefore, we can say the independence assumption is verified.

We have verified all the necessary assumptions of linear regression. We can now use this model for inference.

## Model 2

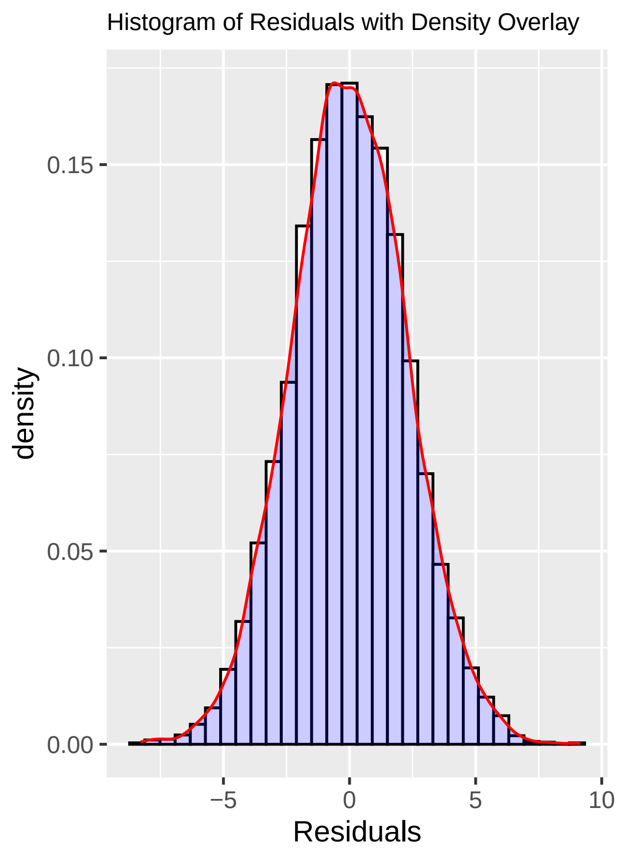
We will now do similar checks for the reduced model

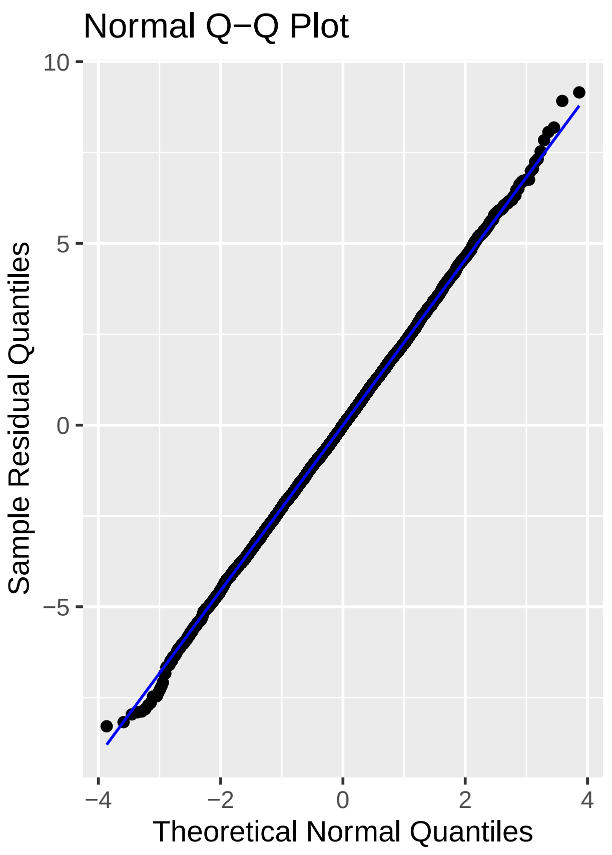
## Normality

To test this, we will look at both a histogram and Normal q-q plot of the data.

```r
df.2 = data.frame(res2, fitted2)

# Histogram of residuals
fig3 = ggplot(df.2, aes(x=res2)) +
  geom_histogram(aes(y = after_stat(density)), color = 'black', fill = 'white') +
  geom_density(color = 'red', fill = 'blue', alpha = 0.2) +
  labs(title = 'Histogram of Residuals with Density Overlay', x = 'Residuals') +
  theme(plot.title = element_text(size = 9))

# Normal Q-Q plot of residuals
fig4 = ggplot(df.2, aes(sample = res2)) +
  geom_qq() + geom_qq_line(color = 'blue') +
  labs(title = 'Normal Q-Q Plot', x = 'Theoretical Normal Quantiles',
       y = 'Sample Residual Quantiles')

grid.arrange(fig3, fig4, ncol=2)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



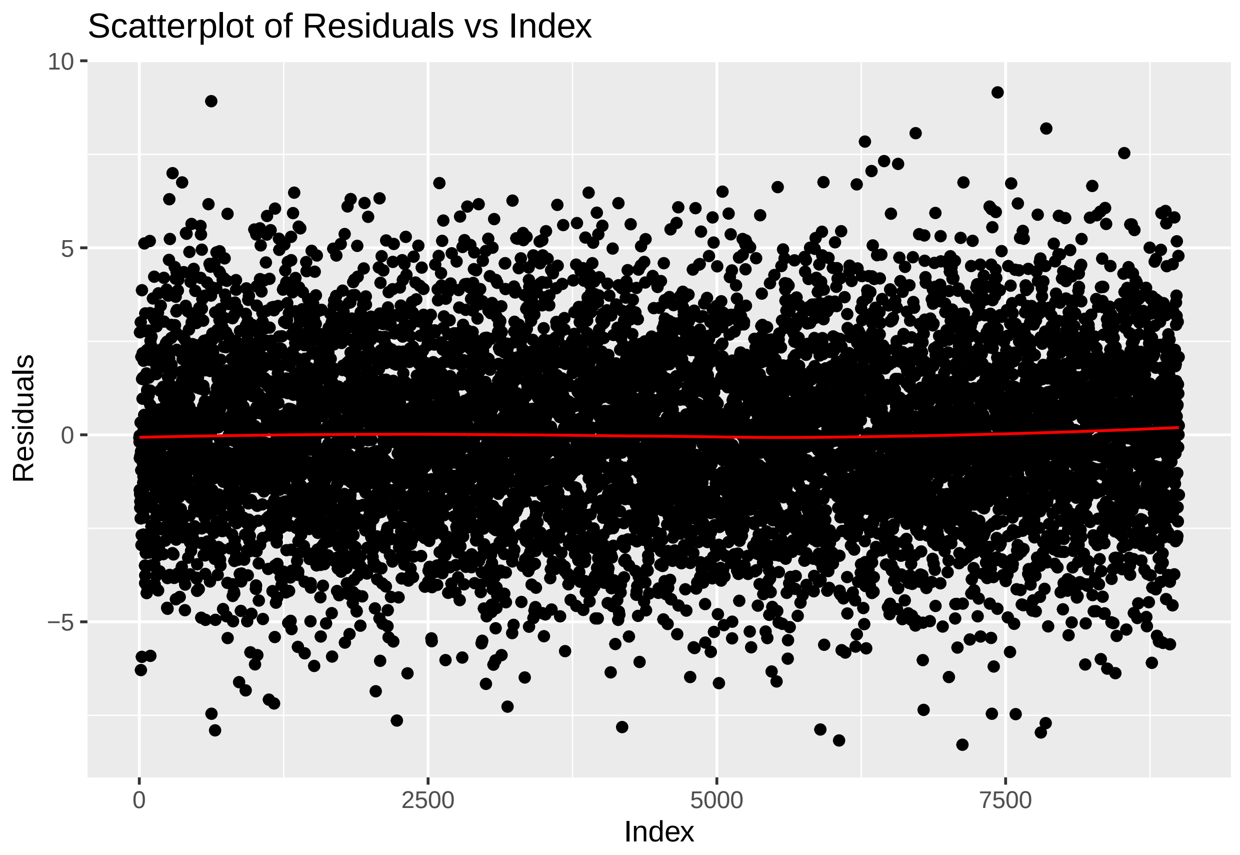


We can see the histogram has the classic bell shape we expect from a Normal Distribution. Also, the points seem to lie fairly well along the Q-Q plot, with no visible concerns. From this, we can say the Normality assumption is satisfied.

### Mean of Zero

```r
# Scatterplot of Residuals vs Index
index = 1:length(res2)
df.2$index = index
ggplot(df.2, aes(x = index, y = res2)) +
  geom_point() +
  geom_smooth(method = 'loess', se = FALSE, colour = 'red', linewidth = 0.5) +
  labs(title = 'Scatterplot of Residuals vs Index', x = 'Index', y = 'Residuals')
```

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



```
mean(res2)
```

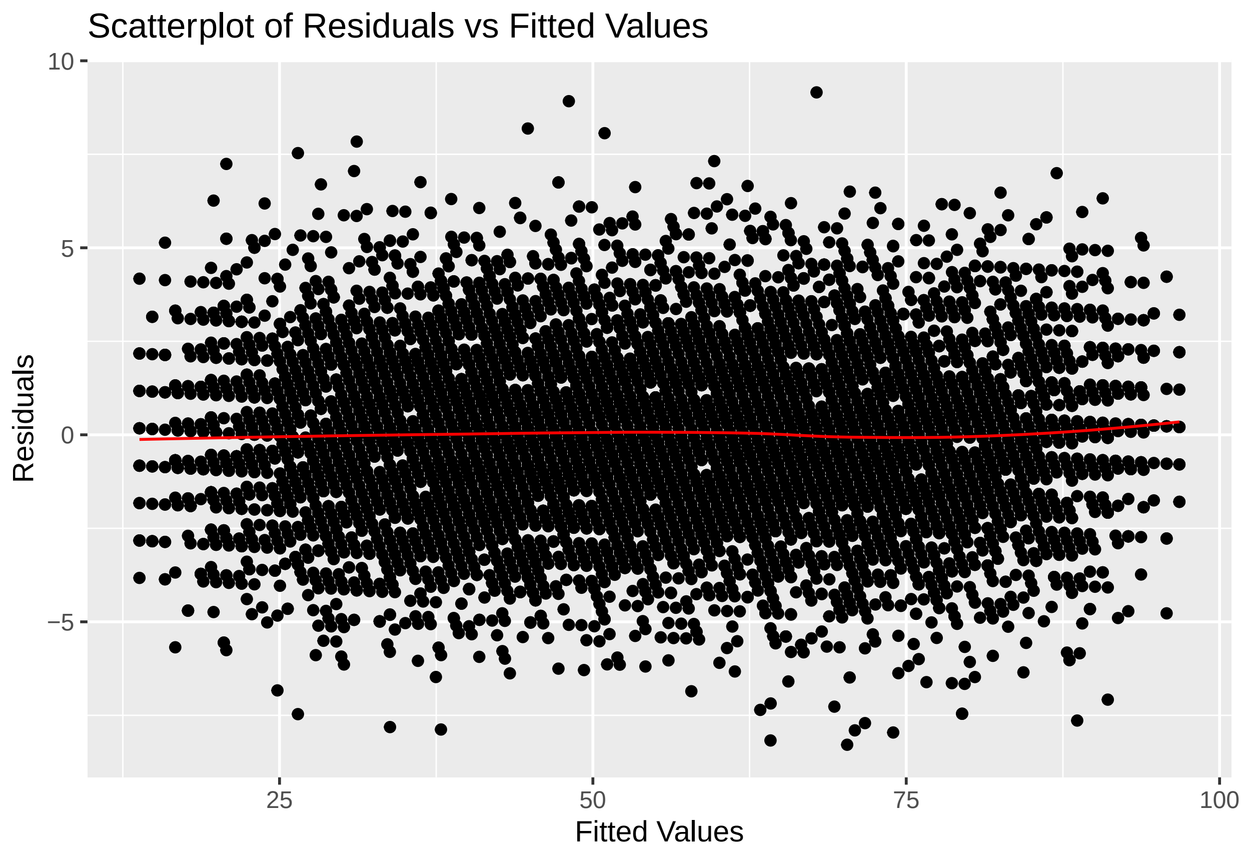
```
## [1] -1.081211e-16
```

We see from the above scatterplot that the points seem to be evenly spread about the horizontal line at zero, with no discernible pattern. We also see the mean of the residuals is very close to zero. We can say the zero mean assumption is satisfied.

### Constant Variance

```
# Residuals vs Fitted Values
ggplot(df.2, aes(x = fitted2, y = res2)) +
  geom_point() +
  geom_smooth(method = 'loess', se = FALSE, colour = 'red', linewidth = 0.5) +
  labs(title = 'Scatterplot of Residuals vs Fitted Values', x = 'Fitted Values',
       y = 'Residuals')
```

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



We see there does not seem to be any change in spread of the data as we move right along the x-axis. This indicates the variance of the residuals is remaining constant throughout. The constant variance assumption is satisfied.

### Independence

If we re-examine the residuals vs fitted values scatterplot, we do not see any pattern, which indicates the residuals are independent of the independent variable.

Next, we check the correlation between the residuals and the predictor variables,

```
data.frame(cor(x = res2, y = performance.data.training[,1:2]), row.names = c('res2'))
```

```
##      Hours.Studied Previous.Scores
## res2 -4.658094e-17    7.547116e-17
```

We see that they both have extremely low correlation with the residuals.

We will now perform two formal tests of randomness to further test for independence among the residuals.

```
randtests::difference.sign.test(res2)
```

```
##
##  Difference Sign Test
##
```

```
## data:  res2
## statistic = -1.3694, n = 8998, p-value = 0.1709
## alternative hypothesis: nonrandomness
```

```
randtests::runs.test(res2)
```

```
##
##  Runs Test
##
## data:  res2
## statistic = -0.67466, runs = 4469, n1 = 4500, n2 = 4500, n = 9000,
## p-value = 0.4999
## alternative hypothesis: nonrandomness
```

All of our analysis has not presented any evidence against independence of the residuals. Therefore, we can say the independence assumption is verified.

We have verified all the necessary assumptions of linear regression. We can now use this model for inference.

## Predictive Accuracy

We will now examine the predictive accuracy of the two models. Sometimes, because of the bias-variance trade-off, models which provide better fit may not be good models for prediction. Using Root Mean Square Error (RMSE), Mean Absolute Error (MAE), and Mean Absolute Percentage Error (MAPE), we will determine which model is better at predicting the performance of a student given new data.

```
# Obtaining the predicted values using the test set for both models
pre1 = predict(model1, newdata = df.te)
pre2 = predict(model2, newdata = df.te)
```

```
# Creating functions to calculate Root Mean Square Error (RMSE),
# Mean Absolute Error (MAE), and Mean Absolute Percentage Error (MAPE)

RMSE = function(obs, pred) {
  se = (obs - pred)^2
  mse = mean(se)
  rmse = sqrt(mse)
  return(round(rmse, 3))
}

MAE = function(obs, pred) {
  ae = abs(obs - pred)
  mae = mean(ae)
  return(round(mae, 3))
}

MAPE = function(obs, pred) {
  ape = abs((obs - pred)/obs)
  mape = mean(ape) * 100
  paste0(round(mape, 3), '%')
}
```

```r
rmse1 = RMSE(df.te$Y, pre1)
rmse2 = RMSE(df.te$Y, pre2)

mae1 = MAE(df.te$Y, pre1)
mae2 = MAE(df.te$Y, pre2)

mape1 = MAPE(df.te$Y, pre1)
mape2 = MAPE(df.te$Y, pre2)

# Table of values of prediction criteria for both models
data.frame(model1 = c(rmse1, mae1, mape1), model2 = c(rmse2, mae2, mape2),
           row.names = c('RMSE', 'MAE', 'MAPE'))
```

```
##      model1 model2
## RMSE  2.003  2.244
## MAE   1.568  1.764
## MAPE 3.407% 3.812%
```

We can see that the full model (model1) performs the best in every category. We will use the full model to predict performance using new data now, and produce a prediction interval.

We will predict the performance of a student with the following data:

Hours Studied = 5, Previous Score = 78, Extracurricular Activities = 0 (meaning ‘No’), Sleep Hours = 6, Sample Question Papers Practiced = 3.

```r
# Using full model to predict score based on new data
newdata = data.frame(X1 = 5, X2 = 78, X3 = as.factor(0), X4 = 6, X5 = 3)
pre.new = predict(model1, newdata = newdata, interval = 'prediction')
pre.new
```

```
##        fit      lwr     upr
## 1 63.10106 59.09762 67.1045
```

We see that a student with the aforementioned data is predicted to get a score of approximately 63, plus or minus around 4. The 95% prediction interval is reasonably small, which is desirable.

In conclusion, based on our analysis, we can say that the full model performs well in terms of both fit and prediction. This means it can be used to assist education professionals in designing study plans and making recommendations to their students on what to focus on to maximize their future test scores.