Experiment - 3: Multiple Linear Regression in Python and R

Why do we need to study Multiple Linear Regression?

1. **Predictive Modeling**: MLR predicts outcomes based on multiple input variables.
2. **Understanding Relationships**: It reveals how independent variables affect the dependent variable.
3. **Optimization**: MLR helps optimize processes by identifying influential factors.
4. **Interpretability**: Results are easy to understand and communicate.
5. **Model Evaluation**: It provides tools for assessing model fit and assumptions.
6. **Foundation for Advanced Techniques**: MLR forms the basis for more complex regression methods.
7. **Research and Analysis**: Widely used in research and data analysis across fields.
8. **Business Decision Making**: Assists in understanding customer behavior and optimizing strategies.
9. **Model Comparison**: Helps select the most suitable model based on various criteria.
10. **Continuous Learning**: Enhances statistical and analytical skills in a data-driven world.

In essence, studying MLR offers insights, predictive power, and analytical skills that are valuable across disciplines and industries.

```
!pip install pandas numpy scikit-learn matplotlib
```

```
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (1.5.3)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (1.26.4)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (1.2.2)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (3.7.1)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2023.4)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.11.4)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (3.2.0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.2.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (4.47.2)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.4.5)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (23.2)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (3.1.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.1->pandas) (1.16.0)
```

Download the dataset from UCI Repository / Kaggle

-- Load data into Google Colab

```
import pandas as pd
df = pd.read_csv('Student_Performance.csv')
```

-- Display the summary of the dataset

```
print(df.describe())
```

```
       Hours Studied  Previous Scores   Sleep Hours  \
count   10000.000000     10000.000000  10000.000000   
mean        4.992900        69.445700      6.530600   
std         2.589309        17.343152      1.695863   
min         1.000000        40.000000      4.000000   
25%         3.000000        54.000000      5.000000   
50%         5.000000        69.000000      7.000000   
75%         7.000000        85.000000      8.000000   
max         9.000000        99.000000      9.000000   

       Sample Question Papers Practiced  Performance Index  
count                      10000.000000       10000.000000  
mean                           4.583300          55.224800  
std                            2.867348          19.212558  
min                            0.000000          10.000000  
25%                            2.000000          40.000000  
50%                            5.000000          55.000000  
75%                            7.000000          71.000000  
max                            9.000000         100.000000  
```

```
df.head()
```

	Hours Studied	Previous Scores	Extracurricular Activities	Sleep Hours	Sample Question Papers Practiced	Performance Index
0	7	99	Yes	9	1	91.0
1	4	82	No	4	2	65.0
2	8	51	Yes	7	2	45.0
3	5	52	Yes	5	2	36.0

```
df.columns
```

```
Index(['Hours Studied', 'Previous Scores', 'Extracurricular Activities',
       'Sleep Hours', 'Sample Question Papers Practiced', 'Performance Index'],
      dtype='object')
```

```
!pip install --upgrade numpy
```

Create a model and fit it

```
from sklearn.linear_model import LinearRegression
```

```
X = df[['Hours Studied', 'Previous Scores', 'Sleep Hours']]
y = df['Performance Index']
```

```
model = LinearRegression()
model.fit(X, y)
```

```
▾ LinearRegression
LinearRegression()
```

Get the values : Coefficient of Determination, Intercept and Coefficients

Coefficient of determination (R-squared)bold text

```
r_squared = model.score(X, y)
```

```
model.score(X,y)
```

```
0.9876497723179762
```

Intercept

```
intercept = model.intercept_
```

```
model.intercept_
```

```
-32.91458717489678
```

Coefficients

```
coefficients = model.coef_
```

```
model.coef_
```

```
array([2.85722462, 1.01884437, 0.47762684])
```

Predict the response

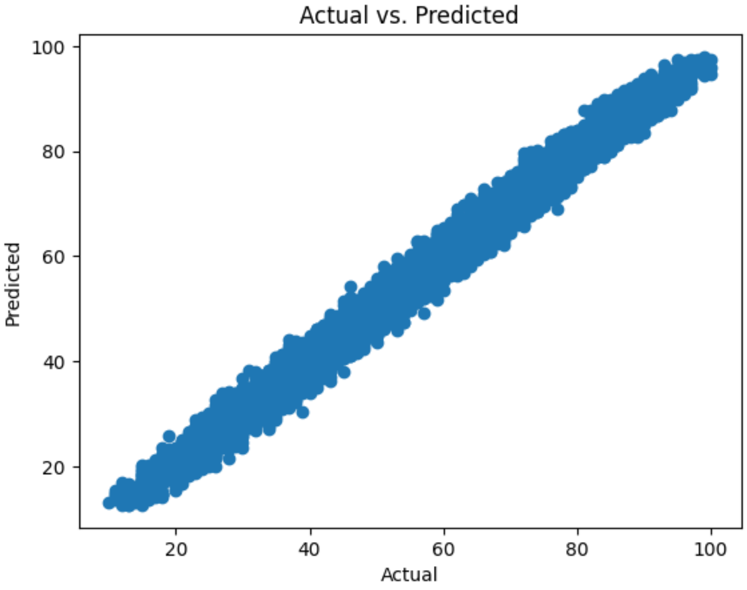
```
y_pred = model.predict(X)
model.predict(X)
```

```
array([92.25021944, 63.97005708, 45.24766058, ..., 72.61385805,
       94.97172626, 66.30148333])
```

Visualize the results with a graph

```
import matplotlib.pyplot as plt

# Scatter plot of actual vs. predicted values
plt.scatter(y, y_pred)
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.title('Actual vs. Predicted')
plt.show()
```



Implementation in R

```
install.packages(c("dplyr", "tidyr", "readr", "ggplot2", "caret", "lmtest"))
```

```
Installing packages into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)
```

```
data <- read.csv("Student_Performance.csv")
```

```
summary(data)
```

```
 Hours.Studied   Previous.Scores Extracurricular.Activities  Sleep.Hours
 Min.   :1.000   Min.   :40.00   Length:10000               Min.   :4.000
 1st Qu.:3.000   1st Qu.:54.00   Class :character           1st Qu.:5.000
 Median :5.000   Median :69.00   Mode  :character           Median :7.000
 Mean   :4.993   Mean   :69.45                              Mean   :6.531
 3rd Qu.:7.000   3rd Qu.:85.00                              3rd Qu.:8.000
 Max.   :9.000   Max.   :99.00                              Max.   :9.000
 Sample.Question.Papers.Practiced Performance.Index
 Min.   :0.000                    Min.   : 10.00
 1st Qu.:2.000                    1st Qu.: 40.00
 Median :5.000                    Median : 55.00
 Mean   :4.583                    Mean   : 55.22
 3rd Qu.:7.000                    3rd Qu.: 71.00
 Max.   :9.000                    Max.   :100.00
```

```
head(data)
```

A data.frame: 6 × 6

	Hours.Studied	Previous.Scores	Extracurricular.Activities	Sleep.Hours	Sample.Ques
	<int>	<int>	<chr>	<int>	
1	7	99	Yes	9	
2	4	82	No	4	
3	8	51	Yes	7	
4	5	52	Yes	5	
5	7	75	No	8	
6	3	78	No	9	

```
X_train <- data[, c("Hours.Studied", "Previous.Scores", "Sleep.Hours", "Sample.Question.Papers.Practiced")]
y_train <- data$Performance.Index
```

```
class(X_train)
```

'data.frame'

```
model <- lm(formula=Performance.Index~Hours.Studied+Previous.Scores+Sleep.Hours+Sample.Question.Papers.Practiced, data=data)
```

```
summary(model)
```

```
Call:
lm(formula = Performance.Index ~ Hours.Studied + Previous.Scores + 
    Sleep.Hours + Sample.Question.Papers.Practiced, data = data)

Residuals:
    Min      1Q  Median      3Q     Max 
-8.3299 -1.3831 -0.0062  1.3701  8.4864 

Coefficients:
                                   Estimate Std. Error t value Pr(>|t|)    
(Intercept)                      -33.763726   0.126841 -266.19   <2e-16 ***
Hours.Studied                      2.853429   0.007962  358.40   <2e-16 ***
Previous.Scores                    1.018584   0.001189  857.02   <2e-16 ***
Sleep.Hours                        0.476333   0.012153   39.19   <2e-16 ***
Sample.Question.Papers.Practiced   0.195198   0.007189   27.15   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.061 on 9995 degrees of freedom
Multiple R-squared:  0.9885,	Adjusted R-squared:  0.9885 
F-statistic: 2.147e+05 on 4 and 9995 DF,  p-value: < 2.2e-16
```

Coefficient of determination (R-squared)

```
r_squared <- summary(model)$r.squared
r_squared
```

0.988498121677258

Intercept

```
intercept <- coef(model)[1]
intercept
```

(Intercept): -33.7637260907948

Coefficients

```
coefficients <- coef(model)[-1]
coefficients
```

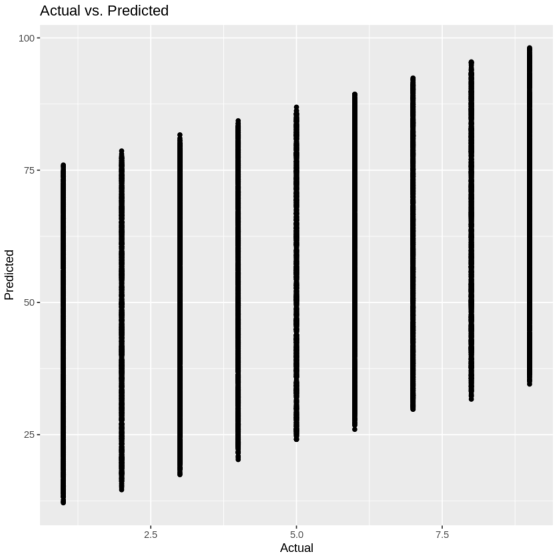
Hours.Studied: 2.85342921456769 Previous.Scores: 1.01858353850839 Sleep.Hours: 0.476332981977769 Sample.Question.Papers.Practiced: 0.195198296660483

Predict the response

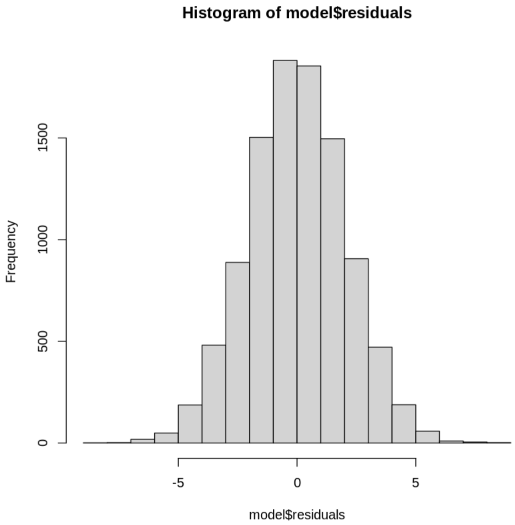
```
predictions <- predict(model, newdata = data)
```

Scatter plot of actual vs. predicted values

```
library(ggplot2)
ggplot(data, aes(x = Hours.Studied, y = predictions)) +
  geom_point() +
  geom_abline(intercept = intercept, slope = 1, color = "red") +
  labs(x = "Actual", y = "Predicted", title = "Actual vs. Predicted")
```



```
hist(model$residuals)
```


Histogram of model$residuals
Frequency
model$residuals