**Prajkta Kamat**

**Task-1:Prediction Using Supervised Machine Learning**

**Predict the percentage of marks of a student based on the number of study hours.**

**Simple linear regression**

**Data:**

Hours Scores

1 2.5 21

2 5.1 47

3 3.2 27

4 8.5 75

5 3.5 30

6 1.5 20

7 9.2 88

8 5.5 60

9 8.3 81

10 2.7 25

11 7.7 85

12 5.9 62

13 4.5 41

14 3.3 42

15 1.1 17

16 8.9 95

17 2.5 30

18 1.9 24

19 6.1 67

20 7.4 69

21 2.7 30

22 4.8 54

23 3.8 35

24 6.9 76

25 7.8 86

|  |
| --- |
| **Input data** |

**> library(readxl)**

**> data <- read\_excel("data.xlsx")**

**> View(data)**

**> head(data)**

Hours Scores

1 2.5 21

2 5.1 47

3 3.2 27

4 8.5 75

5 3.5 30

6 1.5 20

**> dim(data)**

[1] 25 2

**Data visualization**

|  |
| --- |
| **#Scatter Plot** |

**#visualize the linear relationship between the predictor and response.**

|  |
| --- |
| **> scatter.smooth(x=data$Hours, y=data$Scores, main="Hours~Scores") # scatterplot** |
|  |
| |  | | --- | |  | |

**The above scatter plot along with the smoothing line above suggests a linearly increasing relationship between the ‘Scores’ and ‘Hours’ variables.**

|  |
| --- |
| **#Boxplot-check for outliers** |

**# It displays data into quartiles and allows to detect any outliers**

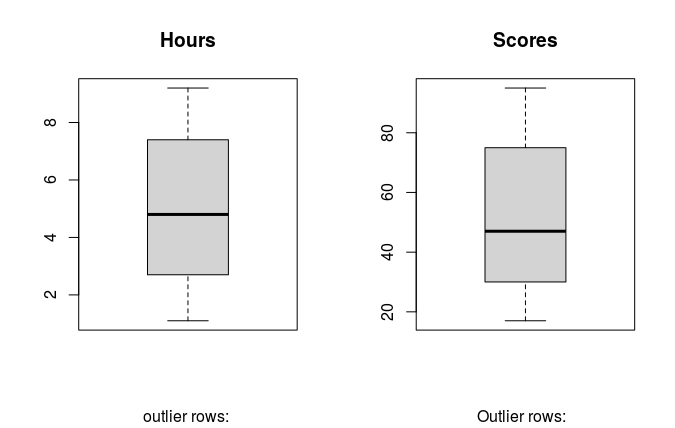
**> par(mfrow=c(1,2)) #divide graph area in 2 columns**

**# box plot for 'Hours'**

**> boxplot(data$Hours, main="Hours", sub=paste("Outlier rows: ", boxplot.stats(data$Hours)$out))**

**# box plot for 'Scores'**

**> boxplot(data$Scores, main="Scores", sub=paste("Outlier rows: ", boxplot.stats(data$Scores)$out))**

****

**Conclusion: No missing values and outliers present in the data set.**

|  |
| --- |
| **#Density plot** |

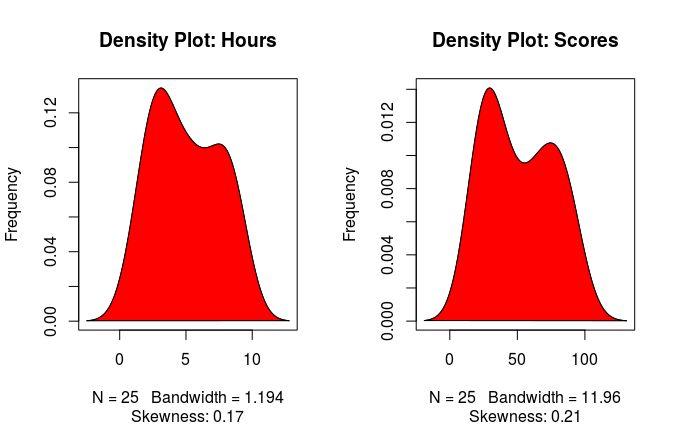
**#To see the distribution of the predictor variable. Ideally, a close to normal distribution (a bell shaped curve),without being skewed to the left or right is preferred.**

**> library(e1071)**

**> par(mfrow=c(1, 2)) # divide graph area in 2 columns**

**# density plot for 'Hours'**

|  |
| --- |
| **> plot(density(data$Hours), main="Density Plot: Hours", ylab="Frequency",**  **sub=paste("Skewness:", round(e1071::skewness(data$Hours), 2)))**  **> polygon(density(data$Hours), col="red")**  **# density plot for 'Scores'**  **> plot(density(data$Scores), main="Density Plot: Scores", ylab="Frequency",**  **sub=paste("Skewness:", round(e1071::skewness(data$Scores), 2)))**  **> polygon(density(data$Scores), col="red")** |
|  |
| |  | | --- | | > | |

****

**above graph is density plot of dependent variable scores ,which look like a normal probability plot so we conclude that our response variable is normally distributed.**

|  |
| --- |
| **#Correlation** |

**> cor(data$Scores,data$Hours) #correlation between Hours and Scores**

[1] 0.9761907

|  |
| --- |
| **#Build Linear Model** |

**> model = lm(data$Scores~ data$Hours)**

**> print(model)**

|  |
| --- |
| Call:  lm(formula = Scores ~ Hours, data = data)  Coefficients:  (Intercept) Hours  2.484 9.776 |
|  |
| |  | | --- | | > | |

|  |
| --- |
| **#Linear Regression Diagnostics** |

**> summary(model) #Model summary**

Call:

lm(formula = data$Scores ~ data$Hours)

Residuals:

Min 1Q Median 3Q Max

-10.578 -5.340 1.839 4.593 7.265

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) 2.4837 2.5317 0.981 0.337

data$Hours 9.7758 0.4529 21.583 <2e-16 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 5.603 on 23 degrees of freedom

Multiple R-squared: 0.9529, Adjusted R-squared: 0.9509

F-statistic: 465.8 on 1 and 23 DF, p-value: < 2.2e-16

|  |
| --- |
| #Predicting Linear Models |

|  |
| --- |
|  |
|  |
| |  | | --- | |  | |

#### Step 1: Create the training and test data samples from original data.

|  |
| --- |
| **> # Create Training and Test data -**  **> set.seed(18) # setting seed to reproduce result of random sampling**  **> trainingRowIndex <- sample(1:nrow(data), 0.8\*nrow(data)) # row indices for training data**  **> trainingData <- data[trainingRowIndex, ] # model training data**  **> testData <- data[-trainingRowIndex, ] # test data** |
|  |
| Step 2: Develop the model on the training data and use it to predict the Scores on test data.  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  | | --- | | **> # Build the model on training data**  **> lmMod <- lm(Scores ~ Hours, data=trainingData) # build the model**  **> ScoresPred <- predict(lmMod, testData) # predict Scores** Step 3: Review diagnostic measures. **> summary (lmMod)**  Call:  lm(formula = Scores ~ Hours, data = trainingData)  Residuals:  Min 1Q Median 3Q Max  -10.8659 -4.4552 0.0822 4.9335 7.2194  Coefficients:  Estimate Std. Error t value Pr(>|t|)  (Intercept) -0.04092 3.01821 -0.014 0.989  Hours 10.10669 0.54331 18.602 3.35e-13 \*\*\*  ---  Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1  Residual standard error: 5.517 on 18 degrees of freedom  Multiple R-squared: 0.9506, Adjusted R-squared: 0.9478  F-statistic: 346 on 1 and 18 DF, p-value: 3.353e-13  **Here R-squared= 0.9506, means the model explain 95.06 % of variability in the dependent variable (Score) by independent variable (Hours).** | |  | | |  |  |  |  |  | | --- | --- | --- | --- | --- | | Step 4: Calculate prediction accuracy and error rates. **# make actuals\_predicteds dataframe.**   |  | | --- | | **>actuals\_preds <- data.frame(cbind(actuals=testData$Scores, predicteds=ScoresPred))**  **> correlation\_accuracy <- cor(actuals\_preds) # 82.7%**  **> head(actuals\_preds)**    actuals predicted  1 88 92.94063  2 42 33.31116  3 17 11.07644  4 30 25.22581  5 86 78.79126 | |  | | |  | | --- | | > | | | | | |

|  |
| --- |
| **> min\_max\_accuracy = mean(apply(actuals\_preds, 1, min) /apply(actuals\_preds, 1, max))**  # => 0.8297, min\_max accuracy  **> mape=mean(abs((actuals\_preds$predicteds - actuals\_preds$actuals))/actuals\_preds$actuals)** |
|  |
| |  | | --- | |  | |

# => 0.1708, mean absolute percentage deviation