Supervised ML Using Linear Regression

This is my first task as a Data Science and BI Intern at The Sparks Foundation. This task is related to supervised ML using Linear Regression.

We are given the data of 25 students scoring certain marks (in percentage) in a test and devoting certain amount of hours. Hence, we can *predict* the marks of the student based on the number of hours devoted using **Linear Regression**.

Preparing a Linear Regression Model

A simple 2-variable linear regression is used to find out the causality of two variables. In our case, we are interested to find out the causality between the marks and number of hours studied.

Linear regression tries to fit a straight line such that the distance between the points of the data set and this regression line is minimum, and there is minimal residual, i.e., the value of the actual and predicted values of the data set.

You can find the data here.

Importing the data

First, let us import the data under the variable name `student_score`.

```
student_scores=read.csv("http://bit.ly/w-data")
head(student_scores)
```

```
##   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
```

Standard Form of a Simple Linear Regression Model

The standard form of a simple linear regression model is

$y = ax + b$

where

x=independent variable y=response variable a=intercept of the regression line and b=slope of the regression line.

In our pursuit to find out a relation between *Score* and *Marks*, we can start off by calculating the **correlation** between these two variables.

Finding Correlation

Let us store the number of hours as `x` and the marks scored as`y` for our convenience. Clearly, number of hours, i.e., `x` is our independent variable and the marks scored, i.e., `y`, is our response variable.

```
x=student_scores$Hours
x
```

```
##  [1] 2.5 5.1 3.2 8.5 3.5 1.5 9.2 5.5 8.3 2.7 7.7 5.9 4.5 3.3 1.1 8.9 2.5 1.9 6.1
## [20] 7.4 2.7 4.8 3.8 6.9 7.8
```

```
y=student_scores$Scores
y
```

```
##  [1] 21 47 27 75 30 20 88 60 81 25 85 62 41 42 17 95 30 24 67 69 30 54 35 76 86
```

Using the `cor` and `cor.test` functions, we get the following results.

```
cor(x,y)
```

```
## [1] 0.9761907
```

```
cor.test(x,y)
```

```
##
##  Pearson's product-moment correlation
##
## data:  x and y
## t = 21.583, df = 23, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.9459248 0.9896072
## sample estimates:
##       cor
## 0.9761907
```

The correlation between `x` and `y` is 0.97, indicating that there is a **strong positive correlation** between the marks and the number of hours.

Using the `cor.test` we can see that the p-value is really small, indicating that the null hypothesis that there is no correlation between `x` and `y` is rejected.

Look at the 95% confidence interval of the correlation. It also indicates that the **actual correlation value lies in this confidence interval**.

Fitting the Linear Regression Model

Now that we know that there is a correlation between `x` and `y`, we move ahead to form a simple linear regression model to understand the dependency of scores on the number of hours using `lm` function.

We will store the model under the variable named `model`.

```
model=lm(y~x)
model
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Coefficients:
## (Intercept)            x
##       2.484        9.776
```

From the above result, we can see that the intercept of this model is 2.484 and the slope is 9.776.

Intercept of this linear regression model is equal to y when x is 0. That means, when the number of hours devoted for the test is zero, the predicted minimum marks a student would get is 2.484.

The slope tells how would `y` change when `x` changes. Simply put in our case, it tells us how much would marks change if the number of hours devoted changes. The slope is a positive number in this case. That means, the marks increase by a factor of 9.776 if the hours change be a factor of 1.

Thus, the equation of our regression line is

$y = 2.484 + 9.776x + e$

where e indicates the residuals. We can use this equation to find the predicted marks.

We use `summary` function to know more about this model.

```
summary(model)
```

```
##
## Call:
## lm(formula = y ~ x)
##
## 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
## x             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
```

The Predicted Marks

To see the predicted marks, use the commands `fitted` or `predict`.

```
fitted(model)
```

```
##        1        2        3        4        5        6        7        8
## 26.92318 52.34027 33.76624 85.57800 36.69899 17.14738 92.42106 56.25059
##        9       10       11       12       13       14       15       16
## 83.62284 28.87834 77.75736 60.16091 46.47479 34.74382 13.23706 89.48832
##       17       18       19       20       21       22       23       24
## 26.92318 21.05770 62.11607 74.82462 28.87834 49.40753 39.63173 69.93672
##       25
## 78.73494
```

```
predict(model)
```

```
##        1        2        3        4        5        6        7        8
## 26.92318 52.34027 33.76624 85.57800 36.69899 17.14738 92.42106 56.25059
##        9       10       11       12       13       14       15       16
## 83.62284 28.87834 77.75736 60.16091 46.47479 34.74382 13.23706 89.48832
##       17       18       19       20       21       22       23       24
## 26.92318 21.05770 62.11607 74.82462 28.87834 49.40753 39.63173 69.93672
##       25
## 78.73494
```

We are in a position to *superficially* compare the actual and the predicted marks.

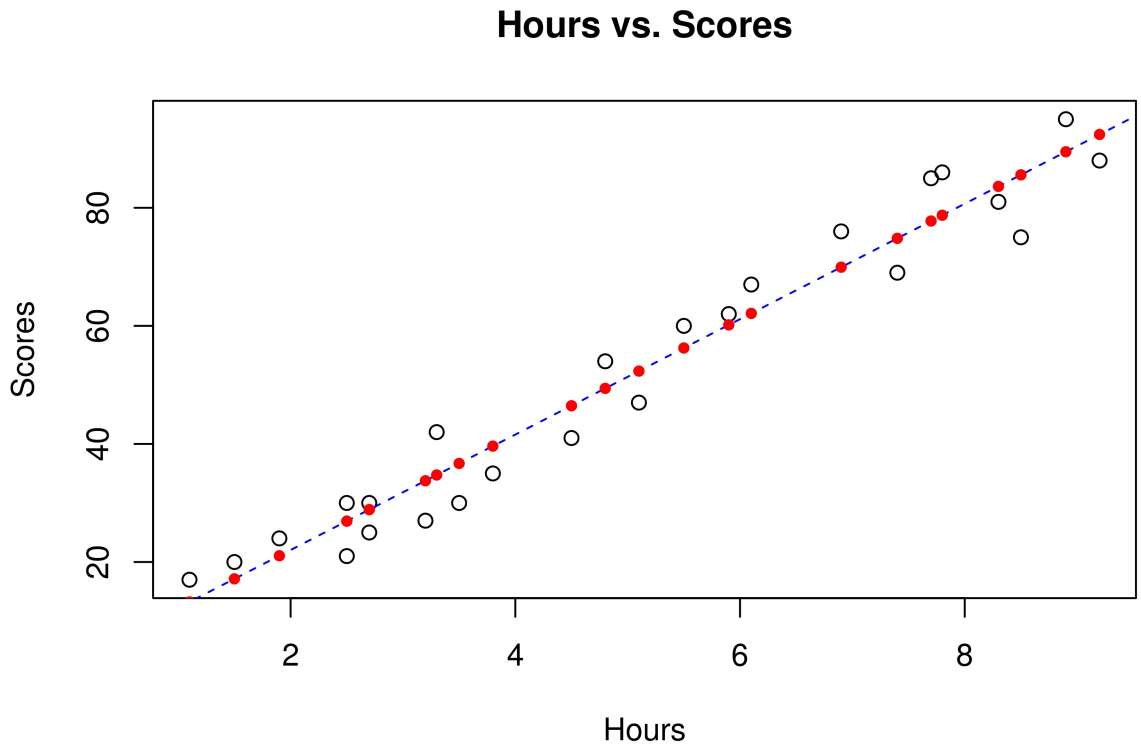
```
cbind(y,predict(model))
```

```
##     y
## 1  21 26.92318
## 2  47 52.34027
## 3  27 33.76624
## 4  75 85.57800
## 5  30 36.69899
## 6  20 17.14738
## 7  88 92.42106
## 8  60 56.25059
## 9  81 83.62284
## 10 25 28.87834
## 11 85 77.75736
## 12 62 60.16091
## 13 41 46.47479
## 14 42 34.74382
## 15 17 13.23706
## 16 95 89.48832
## 17 30 26.92318
## 18 24 21.05770
## 19 67 62.11607
## 20 69 74.82462
## 21 30 28.87834
## 22 54 49.40753
## 23 35 39.63173
## 24 76 69.93672
## 25 86 78.73494
```

Assessing the Fit

To see whether our linear regression is a good fit, we can see it using a scatter plot and superimposing the regression line on the plot.

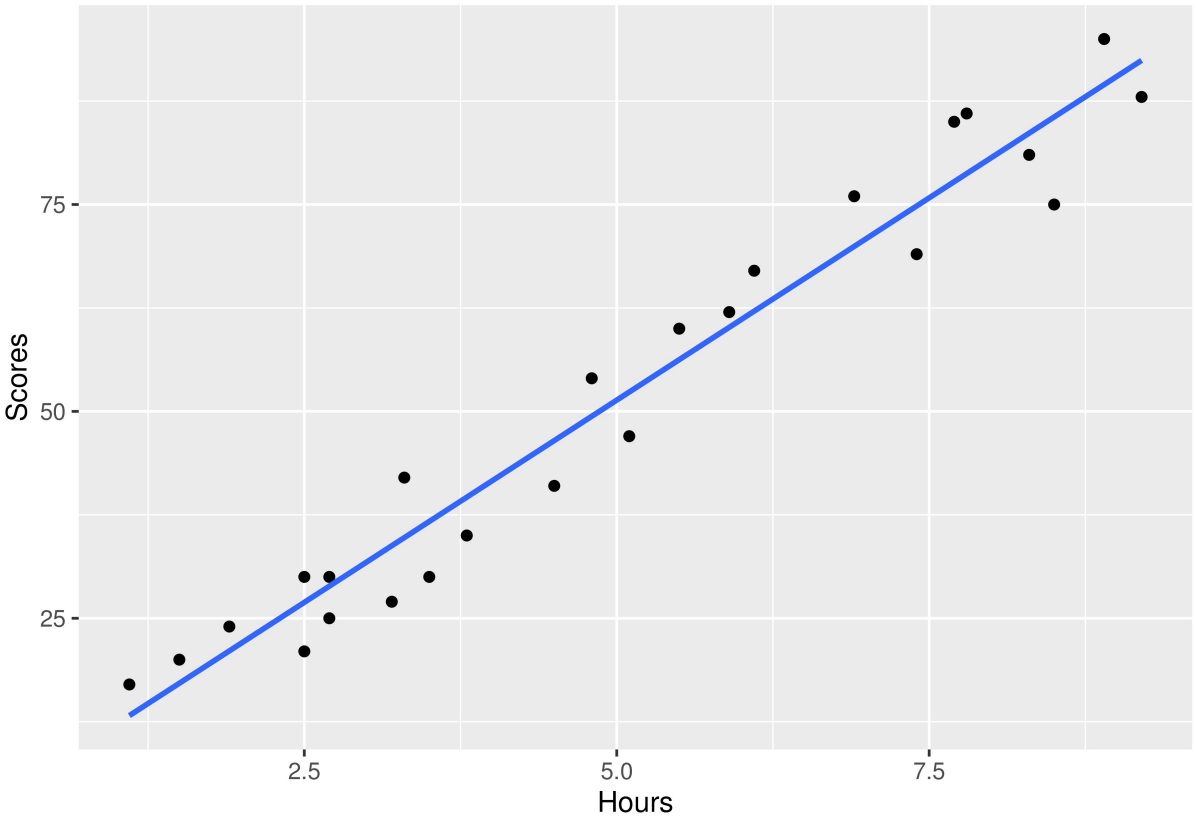
```
plot(x,y,main="Hours vs. Scores",xlab="Hours",ylab="Scores")
abline(model,col="blue",lty="dashed")
points(x,fitted(model),pch=20,col="red")
```



For aesthetics, `ggplot2` package can be used as well.

```
library(ggplot2)
ggplot(student_scores,aes(Hours,Scores))+geom_point()+geom_smooth(method=lm,se=FALSE,fullrange=TRUE)
```

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



Graphically, we can see that the regression line is a good fit. We can prove this, even without any graphics.

R^2, i.e., the goodness of fit, is a rudimentary measure of the accuracy of the model. higher the value of R^2, the greater would be the fit of the model.

R^2 is found out by squaring the correlation coefficient. You can either square the correlation coefficient or extract R^2 value from the`summary` of your model.

```
R2=summary(model)$r.squared
R2
```

```
## [1] 0.9529482
```

```
(cor(x,y))^2
```

```
## [1] 0.9529482
```

Since the value of R^2 is quite high, we can say that this model is a good fit. But, this is not enough to be a good fit. For that, the mean value of residuals should be zero and serially uncorrelated.

First, let us see the residuals.

```
resid(model)
```

```
##          1          2          3          4          5          6          7
##  -5.923182  -5.340271  -6.766244 -10.578002  -6.698985   2.852622  -4.421065
```

```
##          8          9         10         11         12         13         14
##   3.749408  -2.622842  -3.878343   7.242640   1.839087  -5.474789   7.256175
##         15         16         17         18         19         20         21
##   3.762943   5.511676   3.076818   2.942300   4.883926  -5.824618   1.121657
##         22         23         24         25
##   4.592470  -4.631726   6.063283   7.265060
```

```
mean(resid(model))
```

```
## [1] -2.485165e-16
```

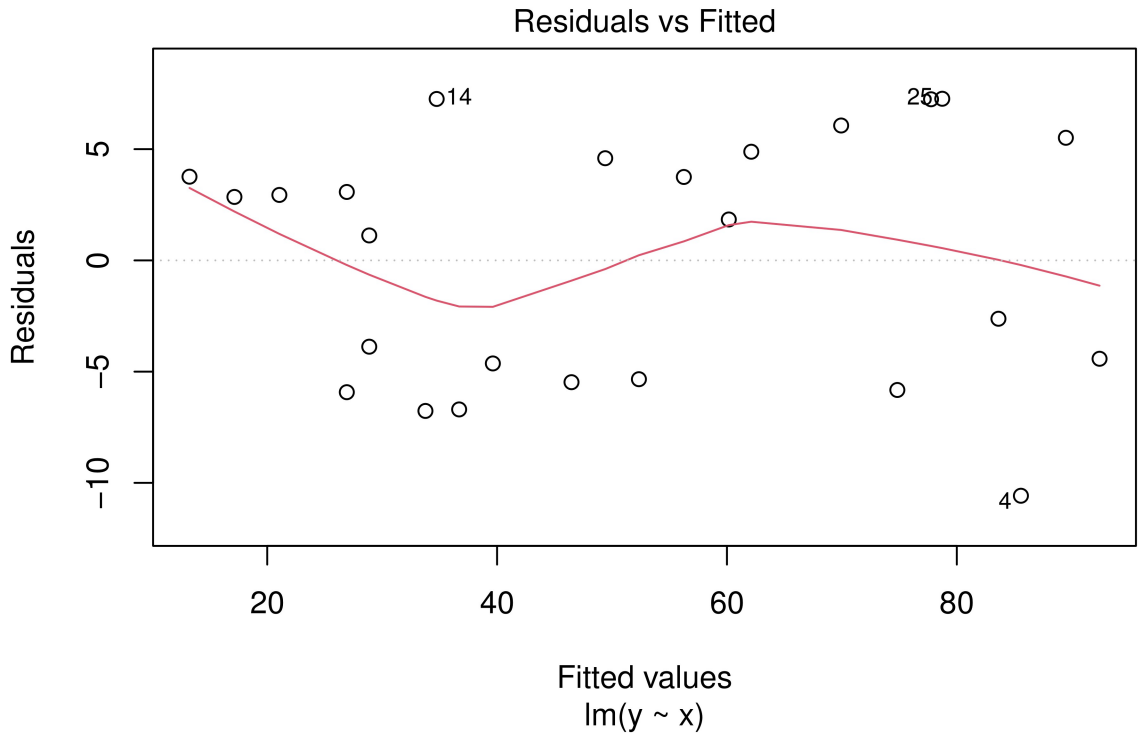
```
var(resid(model))
```

```
## [1] 30.08618
```

```
cor(resid(model),fitted(model))
```

```
## [1] 1.052932e-16
```

```
plot(model,1)
```



The mean value of residuals is too small, and the variance is quite large (indicating that the residuals are spread, and not constricted to one value).

Moreover, there is hardly any correlation between the residual and the fitted values, indicating that the residuals have nothing to do with the predictions made. Thus, it is a good fit.

This graph supports the above-mentioned arguments.

ANOVA Test

Checking the fit through R^2 is not enough. We use ANOVA test to check for linearity of this model. The null hypothesis of ANOVA test is that the value of the intercept is 0, and its alternative hypothesis is otherwise, i.e. not equal to 0.

```
anova(model)
```

```
## Analysis of Variance Table
##
## Response: y
##           Df  Sum Sq Mean Sq F value    Pr(>F)
## x          1 14624.2 14624.2  465.82 < 2.2e-16 ***
## Residuals 23   722.1    31.4
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Since the p-value is significantly small, it means that there is a linear relationship between `x` and `y`.

Overall, **linear regression is a good fit for predicting the marks from the number of hours devoted to studies.**

Predicting scores using the model

Now that we can predict the score using the equation of the regression line, i.e.,

$y = 2.484 + 9.776x$,

we can easily predict the score when the hours devoted is 9.25 hours, i.e., inline equation: $x = 9.25$, using two methods:

1. Data frame method

In this method, we use `data.frame` function to create a new data point in x with a value 9.25. Thus, `x=9.25`.

Then, we will use `predict` function to predict the score at this particular value of `x`.

```
a=data.frame(x=9.25)
predict(model,a)
```

```
##        1
## 92.90985
```

2. First Principle Method

Since scores can be predicted using the regression line equation, put the value $x = 9.25$ in the equation $y = 2.484 + 9.776x$ to get the predicted score at this value of x.

```
coef(model)[1]+coef(model)[2]*9.25
```

```
## (Intercept)
##    92.90985
```

Through both the methods, we will get the same answer.