TSF-GRIP (Feb`23)

Name : **Bhanuprakash**

Task : **Prediction using Supervised ML (Task#1)**

To Do : **Predict the percentage of a student based on the no. of study hours.**

Tool : **R**

03/02/2023

Loading dataset of Scores & Hours.

```
df=read.csv("C:\\Users\\Bhanu\\OneDrive\\Desktop\\Tasks\\StudentScores.csv",header = T);df
```

```
##    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
```

```
attach(df)
```

Summary of the data

```
summary(df)
```

```
##      Hours           Scores
##  Min.   :1.100   Min.   :17.00
##  1st Qu.:2.700   1st Qu.:30.00
##  Median :4.800   Median :47.00
##  Mean   :5.012   Mean   :51.48
##  3rd Qu.:7.400   3rd Qu.:75.00
##  Max.   :9.200   Max.   :95.00
```

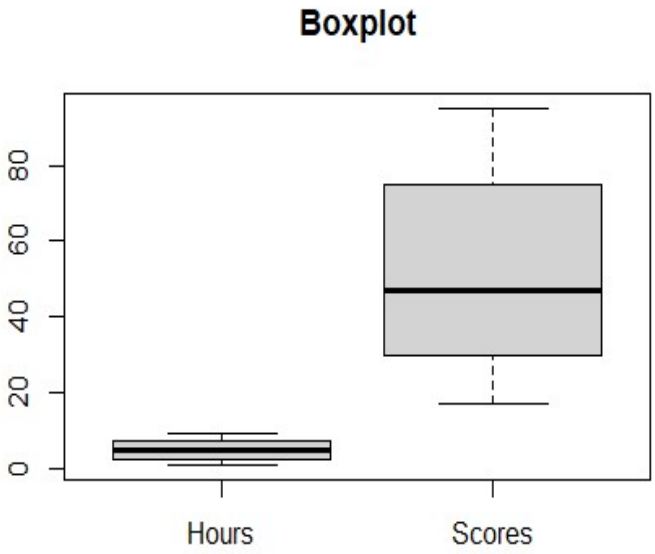
Plotting the data of Hours v/s Scores

```
scatter.smooth(Hours,Scores,xlab='Hours',ylab='Scores',main='Hours vs. Scores')
```

Interpretation: Scores are linearly increasing with hours of study too.

Identifying Outliers in Scores

```
boxplot(df,main='Boxplot')
```



Interpretation: No outliers and most of the data lies on upper part.(i.e..right skewed)

Preparing training and testing sets for modelling

```
library(caret)

set.seed(600)
train_index=createDataPartition(df$Scores,p=0.7,list = F)
trainset=df[train_index,]
testset=df[-train_index,]
print("----Trained----")

## [1] "----Trained----"
```

Model Building...

```
model=lm(Scores~Hours,data = trainset)
summary(model)

##
## Call:
```

```
## lm(formula = Scores ~ Hours, data = trainset)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -9.823 -4.784  1.534  4.312  7.903
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   3.1478     2.9274   1.075    0.296
## Hours         9.6088     0.5054  19.012  2.3e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.603 on 18 degrees of freedom
## Multiple R-squared:  0.9526, Adjusted R-squared:  0.9499
## F-statistic: 361.5 on 1 and 18 DF,  p-value: 2.305e-13
```

Coefficient of determination(R.Sq)

```
summary(model)$r.squared*100

## [1] 95.2565
```

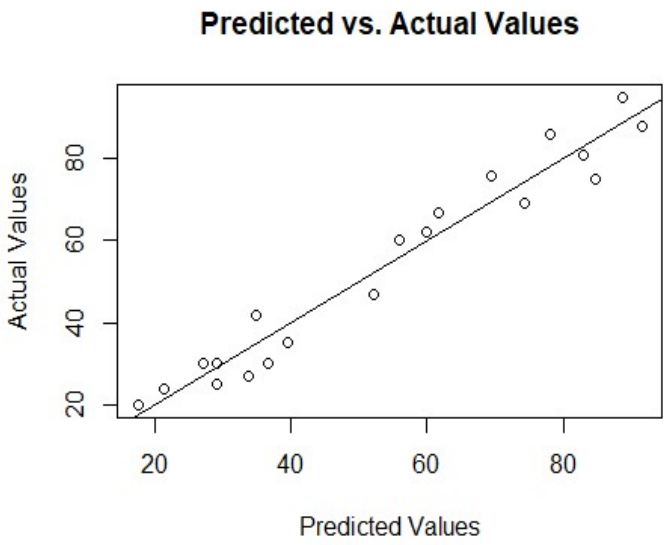
95.2565 % of the variation in the exam scores can be explained by the number of hours studied .

Predicting Scores based on Model

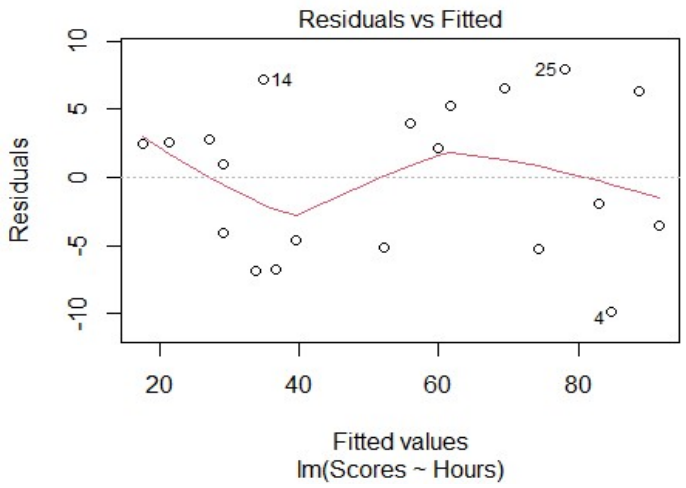
```
fit.model=fitted(model) ; fit.model

##        2        3        4        5        6        7        8        9
## 52.15281 33.89605 84.82281 36.77869 17.56105 91.54898 55.99634 82.90104
##       10       12       14       16       17       18       19       20
## 29.09163 59.83987 34.85693 88.66634 27.16987 21.40458 61.76163 74.25310
##       21       23       24       25
## 29.09163 39.66134 69.44869 78.09663
```

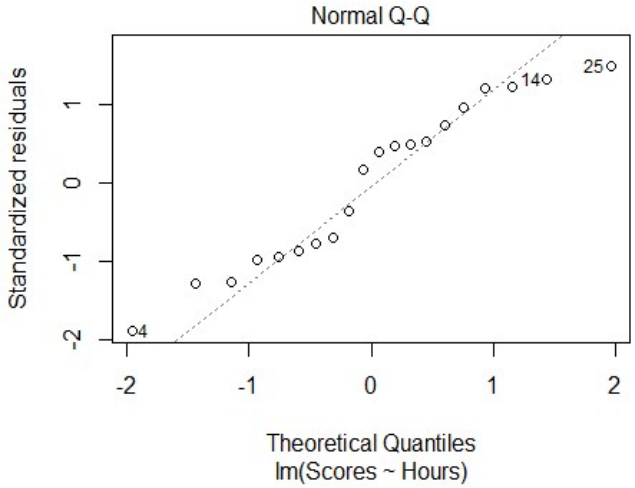
Model Diagnostic Plots

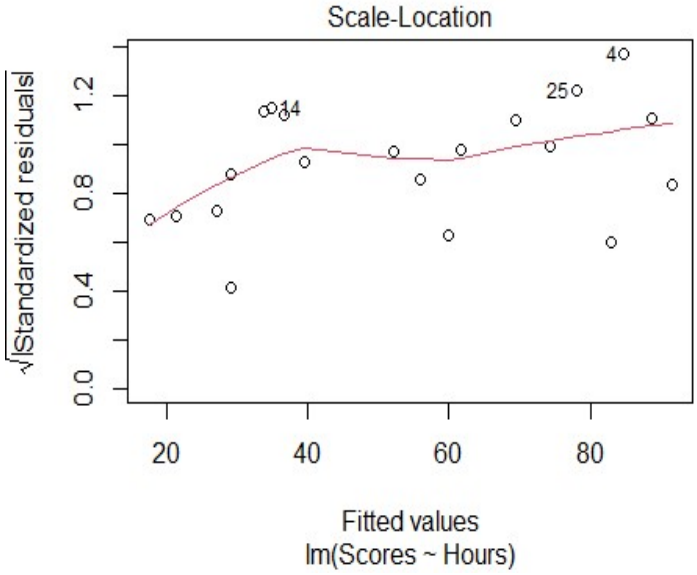
```
# Plot for Predicted v/s Actual Scores
plot(x=predict(model), y= trainset$Scores,
     xlab='Predicted Values',
     ylab='Actual Values',
     main='Predicted vs. Actual Values')
abline(a=0, b=1)
```

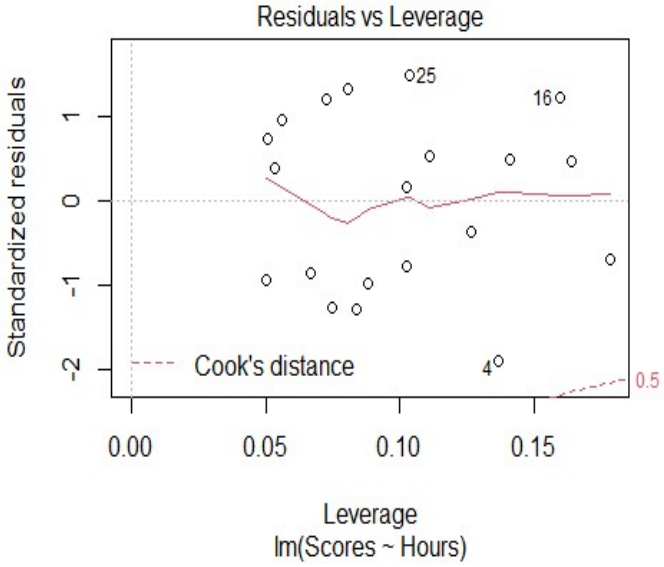



```
plot(lm(Scores~Hours,data = trainset))
```



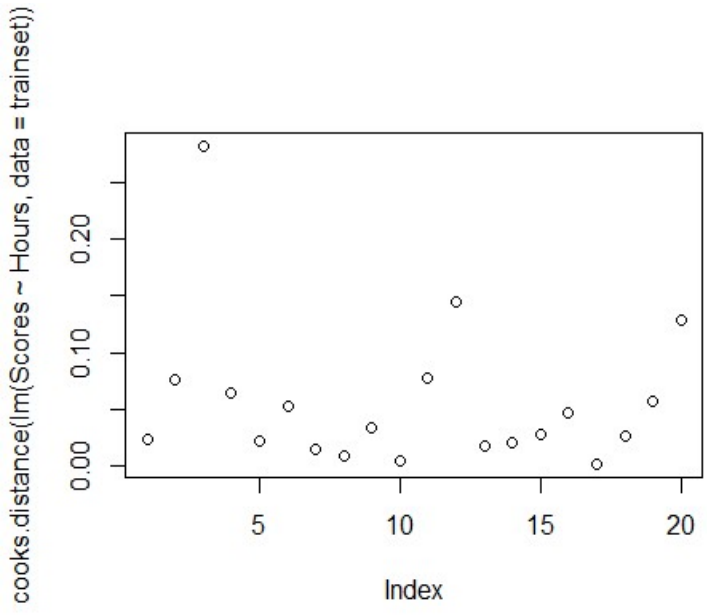






Plot for cook's distance

```
plot(cooks.distance(lm(Scores~Hours,data = trainset)))
```



creating function for predicting score

```
Score=function(hrs){
  score=model[[1]][1]+(model[[1]][2])*hrs
  if (hrs >= 2){
      cat("The score is",score,"for studying",hrs,"hours.","\n")
    }
  else{
    cat("The score is",score,"for studying",hrs,"hour.","\n")
    }
}
```

Testing function()

```
Score(1)

## The score is 12.75664 for studying 1 hour.

Score(9.25)

## The score is 92.02942 for studying 9.25 hours.
```

Evaluating Model Accuracy

```
#Root Mean Square Error
#predicted values,testdata
rmse=function(prevar,testdt) sqrt(mean(((testdt - prevar)^2)))

a=rmse(fit.model,testset$Scores)
cat("The performance of the model value is about ",a)

## The performance of the model value is about  35.17616
```

——-Thank You——-