# The Sparks Foundation - Data Science & Business Analytics Internship

## TASK 1 - Prediction using Supervised Machine Learning

## Author : Vandana

#### In this task we have to predict the percentage score of a student based on the numbers of hours studied. The task has two variables where the features is the number of hours studied and the target value of percentage score. This can be solved using simple linear regression

### Steps:

*Step 1 - Importing the data

*Step 2 - Visualizing the dataset

*Step 3 - Data preparation

*Step 4 - Training the algorithm

*Step 5 - Visualizing the model

*Step 6 - Making predcitions

*Step 7 - Evaluating the model

### STEP 1 - Importing the dataset

In this step, we will import the dataset through the link with the help of pandas library and then we will observe the data

In [1]:

```
# Importing all the required libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns

# To ignore the warnings
import warnings as wg
wg.filterwarnings("ignore")
```

In [4]:

```
# Reading data from remote link
url = "https://raw.githubusercontent.com/AdiPersonalWorks/Random/master/student_scor
df = pd.read_csv(url)
```

In [5]:

```
# now let's observe the dataset
df.head()
```

Out[5]:

| | Hours | Scores |
|---|---|---|
| **0** | 2.5 | 21 |
| **1** | 5.1 | 47 |
| **2** | 3.2 | 27 |
| **3** | 8.5 | 75 |
| **4** | 3.5 | 30 |

In [6]:

```
df.tail()
```

Out[6]:

| | Hours | Scores |
|---|---|---|
| **20** | 2.7 | 30 |
| **21** | 4.8 | 54 |
| **22** | 3.8 | 35 |
| **23** | 6.9 | 76 |
| **24** | 7.8 | 86 |

In [7]:

```
# To find the number of columns and rows
df.shape
```

Out[7]:

```
(25, 2)
```

In [8]:

```
# To find more information about our dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25 entries, 0 to 24
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Hours   25 non-null     float64
 1   Scores  25 non-null     int64  
dtypes: float64(1), int64(1)
memory usage: 528.0 bytes
```

In [9]:

```
df.describe()
```

Out[9]:

| | Hours | Scores |
|---|---|---|
| **count** | 25.000000 | 25.000000 |
| **mean** | 5.012000 | 51.480000 |
| **std** | 2.525094 | 25.286887 |
| **min** | 1.100000 | 17.000000 |
| **25%** | 2.700000 | 30.000000 |
| **50%** | 4.800000 | 47.000000 |

|  | Hours | Scores |
|---|---|---|
| **75%** | 7.400000 | 75.000000 |
| **max** | 9.200000 | 95.000000 |

In [10]:

```
# now we will check if our dataset contains null or missings values
df.isnull().sum()
```

Out[10]:

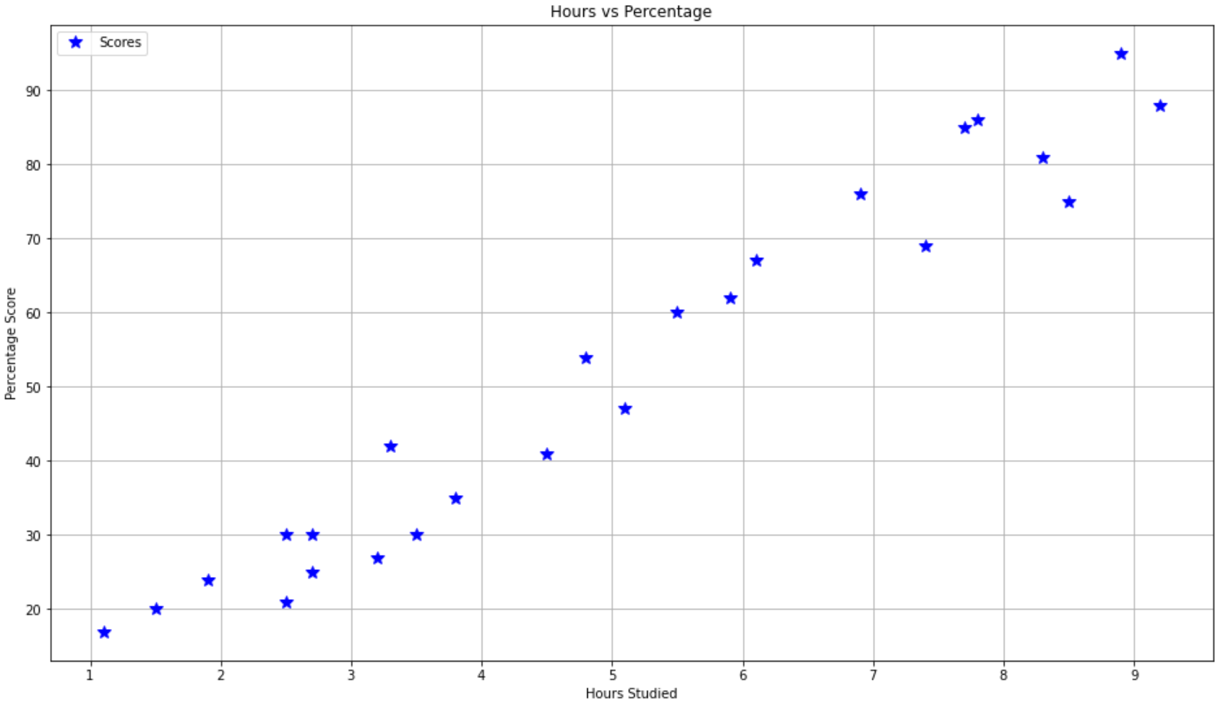
```
Hours     0
Scores    0
dtype: int64
```

## STEP 2 - Visualizing the dataset

In this we will plot the dataset to check whether we can observe any relation between the two variables or not.

In [11]:

```
# Plotting the dataset
plt.rcParams["figure.figsize"] = [16,9]
df.plot(x='Hours', y='Scores', style='*', color='blue', markersize=10)
plt.title('Hours vs Percentage')
plt.xlabel('Hours Studied')
plt.ylabel('Percentage Score')
plt.grid()
plt.show()
```



From the graph above, we can observe that there is a linear relationship between "hours studied" and "percentage score". So, we can use the linear regression supervised machine model on it to predict further values.

In [12]:

```
# we can also use .corr to determine the corelation between the variables
df.corr()
```

Out[12]:

|  | Hours | Scores |
|---|---|---|

|  | Hours | Scores |
|---|---|---|
| **Hours** | 1.000000 | 0.976191 |
| **Scores** | 0.976191 | 1.000000 |

## STEP 3 - Data preparation

In this step we will divide the data into "features" (inputs) and "labels" (outputs). After that we will split the whole dataset into 2 parts - testing data and training data.

In [13]:

```
df.head()
```

Out[13]:

|  | Hours | Scores |
|---|---|---|
| **0** | 2.5 | 21 |
| **1** | 5.1 | 47 |
| **2** | 3.2 | 27 |
| **3** | 8.5 | 75 |
| **4** | 3.5 | 30 |

In [14]:

```
# using iloc function we will divide the data
X = df.iloc[:, :1].values
y = df.iloc[:, 1:].values
```

In [15]:

```
X
```

Out[15]:

```
array([[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],
       [7.4],
       [2.7],
       [4.8],
       [3.8],
       [6.9],
       [7.8]])
```

In [16]:

```
y
```

Out[16]:
```
array([[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]], dtype=int64)
```

In [18]:
```
# Splitting data into training and testing data

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
                            test_size=0.2, random_state=0)
```

## STEP 4 - Training the Algorithm

We have splited our data into training and testing sets, and now we will train our Model.

In [20]:
```
from sklearn.linear_model import LinearRegression

model = LinearRegression()
model.fit(X_train, y_train)
```

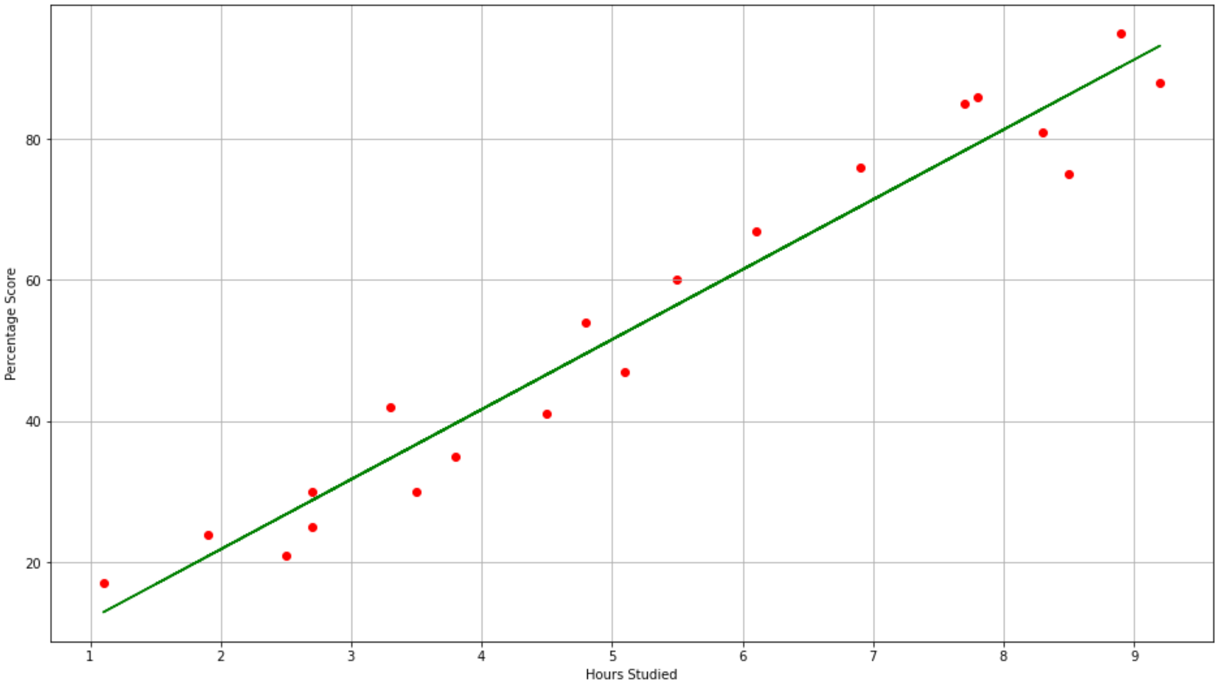
Out[20]:
```
LinearRegression()
```

## STEP 5 - Visualizing the model

After training the model, now its time to visualize it.

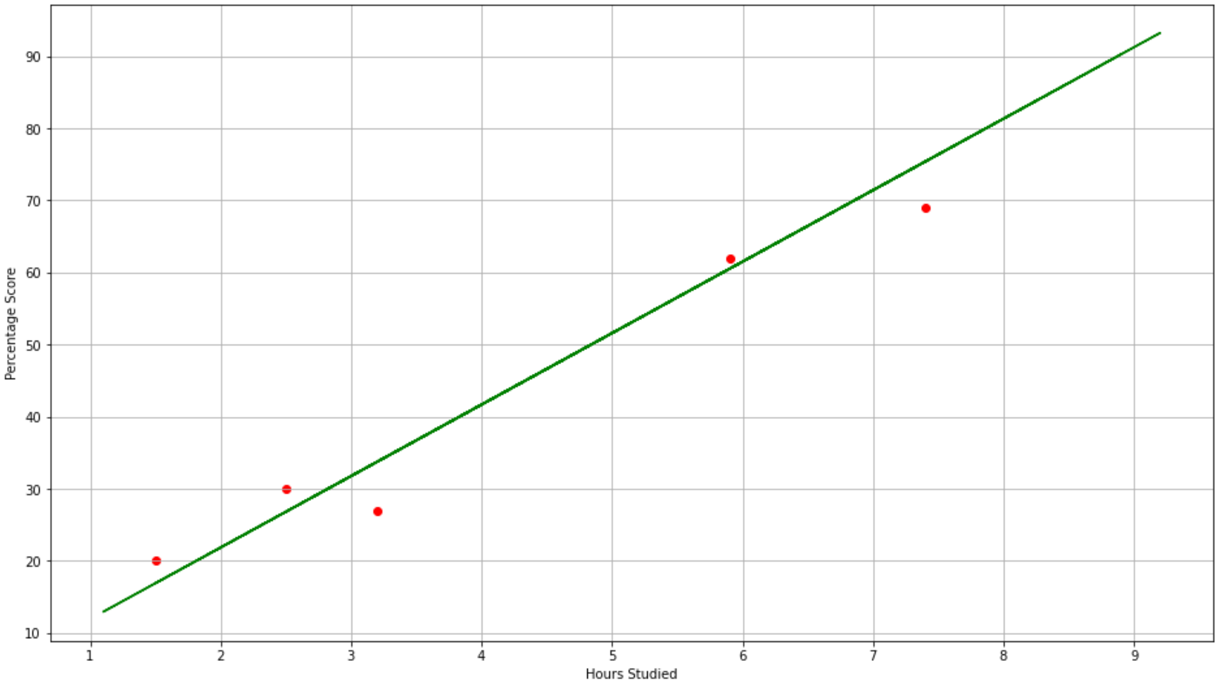
In [21]:
```
line = model.coef_*X + model.intercept_

# Plotting for the training data
plt.rcParams["figure.figsize"] = [16,9]
plt.scatter(X_train, y_train, color='red')
plt.plot(X, line, color='green');
plt.xlabel('Hours Studied')
plt.ylabel('Percentage Score')
plt.grid()
plt.show()
```



In [22]:

```
# Plotting for the testing data
plt.rcParams["figure.figsize"] = [16,9]
plt.scatter(X_test, y_test, color='red')
plt.plot(X, line, color='green');
plt.xlabel('Hours Studied')
plt.ylabel('Percentage Score')
plt.grid()
plt.show()
```



## STEP 6 - Making Predictions

Now that we have trained our algorithm, it's time to make some predictions

In [23]:

```
print(X_test) # Testing data - In Hours
y_pred = model.predict(X_test) # Predicting the scores
```

```
[[1.5]
```

```
 [3.2]
 [7.4]
 [2.5]
 [5.9]]
```

In [24]:

```
# Comparing Actual vs Predicted

y_test
```

Out[24]:

```
array([[20],
       [27],
       [69],
       [30],
       [62]], dtype=int64)
```

In [25]:

```
y_pred
```

Out[25]:

```
array([[16.88414476],
       [33.73226078],
       [75.357018  ],
       [26.79480124],
       [60.49103328]])
```

In [26]:

```
# Comparing Actual vs Predicted
comp = pd.DataFrame({ 'Actual':[y_test],'Predicted':[y_pred] })
comp
```

Out[26]:

| | Actual | Predicted |
|---|---|---|
| **0** | [[20], [27], [69], [30], [62]] | [[16.884144762398037], [33.73226077948984], [7... |

In [27]:

```
# Testing with your own data

hours = 9.25
own_pred = model.predict([[hours]])
print("The predicted score if a person studies for",hours,"hours is",own_pred[0])
```

```
The predicted score if a person studies for 9.25 hours is [93.69173249]
```

**Hence, it can be concluded that the predicted score if a person studies for 9.25 hours is 93.69173249**

## STEP 7 - Evaluating the model

In the last step, we are going to evaluate our trained model by calculating mean absolute error

In [28]:

```
from sklearn import metrics

print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
```

```
Mean Absolute Error: 4.183859899002975
```