# GRIP : The Sparks Foundation

**Data Science & Business Analytics**

**Author: Ume Salma Khan**

**Task 1 : Prediction using Supervised ML**

***In this task we have to predict the percentage score of a student based on the number of hours studied. The task has two variables where the feature is the no. of hours studied and the target value is the percentage score. This can be solved using Simple Linear Regression.***

In [38]:
```
# Importing required libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import linear_model
```

In [39]:
```
# Importing the Data

dataset = pd.read_csv("study_hours.csv")
hours= dataset.iloc[:, :-1].values
score= dataset.iloc[:, -1].values
```

In [40]:
```
# Describing the data

dataset.head()
df = pd.read_csv("study_hours.csv")
df
```

Out[40]:

| | Hours | Scores |
|---|---|---|
| 0 | 2.5 | 21 |
| 1 | 5.1 | 47 |
| 2 | 3.2 | 27 |
| 3 | 8.5 | 75 |
| 4 | 3.5 | 30 |
| 5 | 1.5 | 20 |
| 6 | 9.2 | 88 |
| 7 | 5.5 | 60 |
| 8 | 8.3 | 81 |
| 9 | 2.7 | 25 |
| 10 | 7.7 | 85 |
| 11 | 5.9 | 62 |
| 12 | 4.5 | 41 |
| 13 | 3.3 | 42 |
| 14 | 1.1 | 17 |
| 15 | 8.9 | 95 |
| 16 | 2.5 | 30 |
| 17 | 1.9 | 24 |
| 18 | 6.1 | 67 |
| 19 | 7.4 | 69 |
| 20 | 2.7 | 30 |
| 21 | 4.8 | 54 |
| 22 | 3.8 | 35 |
| 23 | 6.9 | 76 |
| 24 | 7.8 | 86 |

In [41]:
```
print(hours)
```

```
[[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 [37]:
```
print(score)
```

```
[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]
```

In [15]:
```
#Splitting dataset into the Training set and Test set

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(hours, score, test_size= 0.2, random_state=0 )
```

In [17]:
```
#Training the Simple Linear Regression model on the Training set

from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(x_train, y_train)
```

Out[17]:
```
LinearRegression()
```

In [18]:
```
# Predicting the Test Set Results

y_pred = regressor.predict(x_test)
```

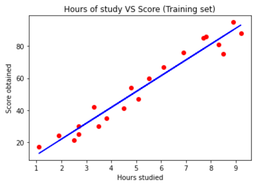
In [19]:
```
print(y_pred)
```

```
[16.88414476 33.73226078 75.357018   26.79480124 60.49103328]
```

Visualization :

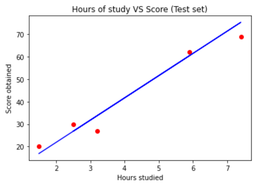
In [20]:
```
# Visualizing of Training Set Results

plt.scatter(x_train, y_train, color = "red")
plt.plot(x_train, regressor.predict(x_train), color= "blue")
plt.title("Hours of study VS Score (Training set)")
plt.xlabel("Hours studied")
plt.ylabel("Score obtained")
plt.show()
```




In [21]:
```
# Visualising the Test Set Results

plt.scatter(x_test, y_test, color = "red")
plt.plot(x_test, y_pred , color= "blue")
plt.title("Hours of study VS Score (Test set)")
plt.xlabel("Hours studied")
plt.ylabel("Score obtained")
plt.show()
```




Prediction :

What will be the predicted score if a student studies for 9.25 hrs/day?

In [22]:
```
y_sample = regressor.predict([[9.25]])
```

In [24]:
```
print("The student who studies for 9.25 hrs/day will get a score of :", y_sample)
```

```
The student who studies for 9.25 hrs/day will get a score of : [93.69173249]
```

In [25]:
```
from sklearn.metrics import mean_squared_error
from math import sqrt
print(sqrt(mean_squared_error(y_test, y_pred)))
```

```
4.647447612100368
```

Evaluation :

In [26]:
```
#Calculate Root Mean-Square Error(RMSE)
from sklearn.metrics import mean_squared_error
from math import sqrt
print(sqrt(mean_squared_error(y_test, y_pred)))
```

```
4.647447612100368
```

In [27]:
```
from sklearn.metrics import mean_absolute_error
print(mean_squared_error(y_test, y_pred))
```

```
21.598769307217413
```

In [28]:
```
#Calculate R2 Score
from sklearn.metrics import r2_score
r2_score(y_test, y_pred)
```

Out[28]:
```
0.9454906892105355
```

***The best possible R2 Score is 1 (when the model has no error and all true values match with the prediction values) and it can be negative (because the model can be arbitrarily worse) or value 0 (for all inputs it predicts the same output). Hence, 0.945 R2 Score tells our model is pretty good.***