GRIP : The Sparks Foundation

Data Science and Business Analytics Intern

Author : Aditya Kumar Mishra

Task 1 : Prediction Using Supervised ML

**In this task we have to predict the percentage score of a student 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 soloved using

In [1]:
```
# Importing reguired libaries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

**Reading data from url

In [2]:
```
url="http://bit.ly/w-data"
data=pd.read_csv(url)
```

Exploring Data

In [3]:
```
print(data.shape)
data.head()
```

```
(25, 2)
```

Out[3]:

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

In [4]:
```
data. describe()
```

Out[4]:

	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
75%	7.400000	75.000000
max	9.200000	95.000000

In [5]:
```
data
```

Out[5]:

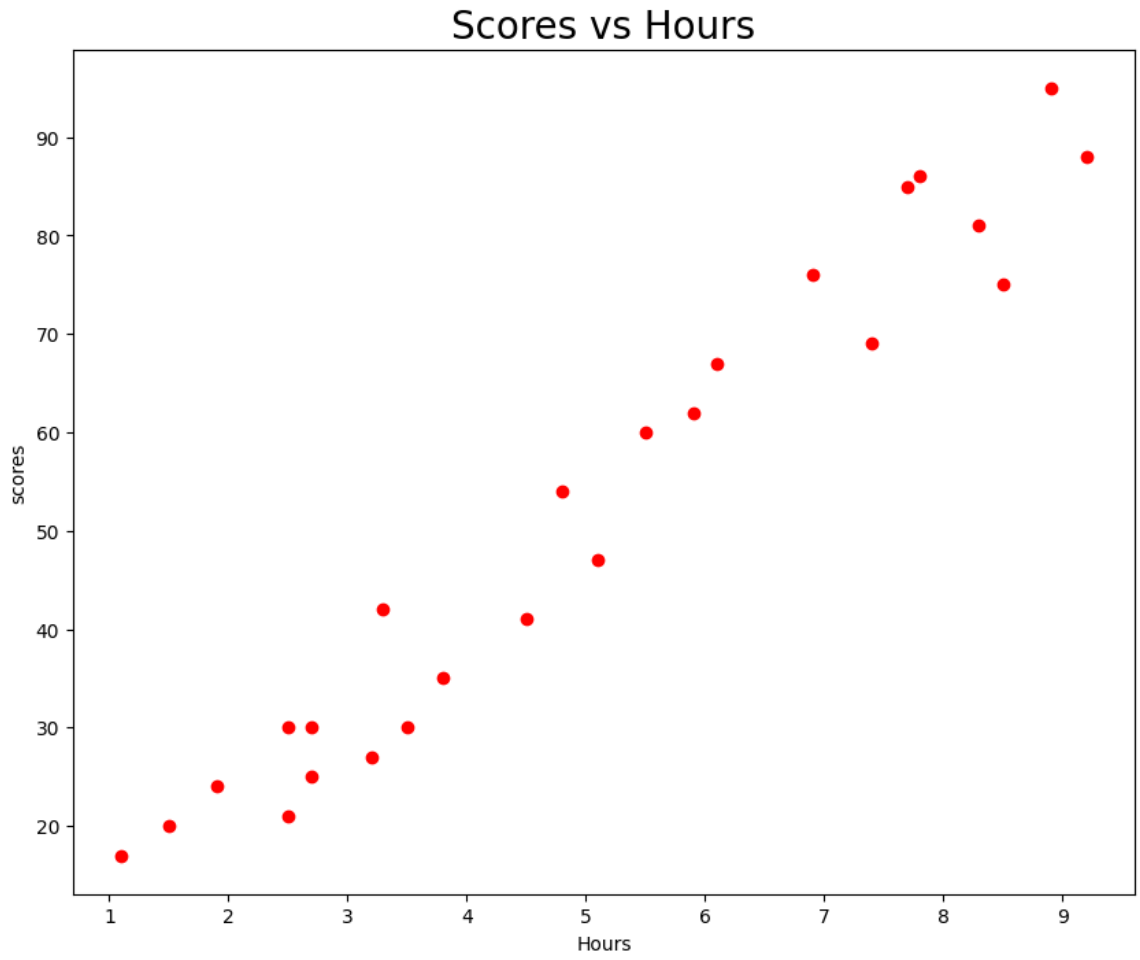
	Hours	Scores
0	2.5	21
1	5.1	47
2	3.2	27
3	8.5	75
4	3.5	30
5	1.5	20
6	9.2	88
7	5.5	60
8	8.3	81
9	2.7	25
10	7.7	85
11	5.9	62
12	4.5	41
13	3.3	42
14	1.1	17
15	8.9	95
16	2.5	30
17	1.9	24
18	6.1	67
19	7.4	69
20	2.7	30
21	4.8	54
22	3.8	35
23	6.9	76
24	7.8	86

values stored in variables X , y respectively

In [7]:
```
X = data['Hours']
y = data['Scores']
```

Visualizing Data

In [20]:
```
plt.figure(figsize=(10,8))
plt.title('Scores vs Hours',size=20)
plt.scatter(X,y,c='red')
plt.xlabel('Hours')
plt.ylabel('scores')
plt.show()
```



Preparing The Data

In [25]:
```
#Dividing the data into "attributes"(inputs) and "labels"(outputs).Using
x=data.iloc[:, :-1].values
y=data.iloc[:,1].values
```

In [26]:
```
x
```

Out[26]:
```
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 [27]:
```
y
```

Out[27]:
```
array([21, 47, 27, 75, 30, 20, 88, 60, 81, 25, 85, 62, 41, 42, 17, 95, 3
0,
       24, 67, 69, 30, 54, 35, 76, 86], dtype=int64)
```

Splitting Dataset Into Train and Test

In [28]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_s
```

Training the model

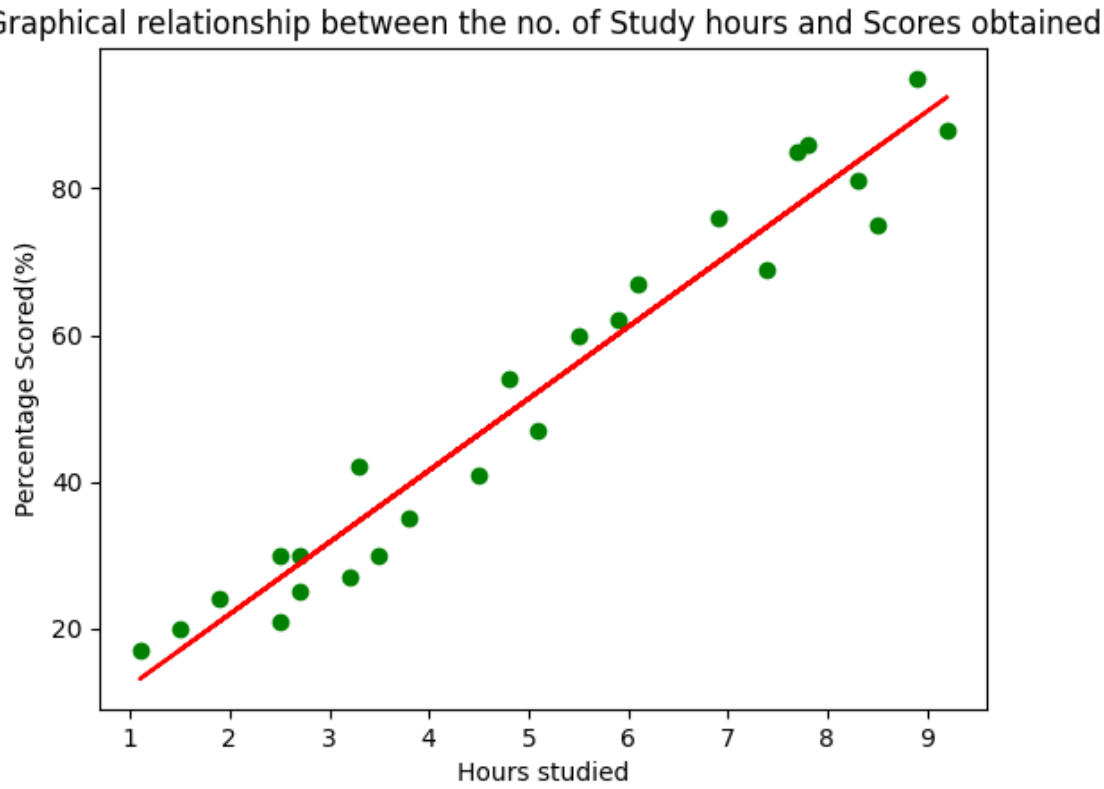
In [29]:
```
regressor=LinearRegression()
regressor.fit(x_train.reshape(-1,1),y_train)
print("Training of the model is complete")
```

```
Training of the model is complete
```

Plotting Linear Regression Model

In [31]:
```
#Plotting the regression line
line=regressor.coef_*x+regressor.intercept_

#Plotting the scatter plot with the regression line
plt.scatter(x,y,color='green',marker='o')
plt.plot(x,line,color='red');
plt.title('Graphical relationship between the no. of Study hours and Scor
plt.xlabel('Hours studied')
plt.ylabel('Percentage Scored(%)')
plt.show()
```



In [32]:
```
print('intercept={}, slope coefficient={}'.format(regressor.intercept_,re
```

```
intercept=2.370815382341881, slope coefficient=[9.78856669]
```

Testing data

In [33]:
```
print(x_test)
y_pred=regressor.predict(x_test)
```

```
[[1.5]
 [3.2]
 [7.4]
 [2.5]
 [5.9]
 [3.8]
 [1.9]
 [7.8]]
```

Comparing Actual and Predicted Results

In [34]:
```
data=pd.DataFrame({'Actual': y_test,'Predicted':y_pred})
data
```

Out[34]:

	Actual	Predicted
0	20	17.053665
1	27	33.694229
2	69	74.806209
3	30	26.842232
4	62	60.123359
5	35	39.567369

6	24	20.969092
7	86	78.721636

**Accurcy of model

In [36]:
```
#Test with own data
from sklearn import metrics
metrics.r2_score(y_test,y_pred)
```

Out[36]:
```
0.9568211104435257
```

**Predicting Error

In [37]:
```
from sklearn import metrics
print('Mean Absolute Error:',metrics.mean_absolute_error(y_test,y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test,y_pred))
print('Root Mean Squared Error:',np.sqrt(metrics.mean_squared_error(y_tes
print('R2:',metrics.r2_score(y_test,y_pred))
```

```
Mean Absolute Error: 4.419727808027652
Mean Squared Error: 22.96509721270043
Root Mean Squared Error: 4.792191274636315
R2: 0.9568211104435257
```

**What will be predicted score if student studies for 9.25hr?

In [40]:
```
hours =9.25
test=np.array([hours])
test=test.reshape(-1,1)
own_pred=regressor.predict(test)
print("No of hours={}".format(hours))
print("Predicted Score={}".format(own_pred[0]))
```

```
No of hours=9.25
Predicted Score=92.91505723477056
```

In [ ]:
```

```