DATA SCIENCE AND BUSINESS ANALYTIC INTERN

NAME - SHIVAM SINGH

TASK-1 PREDICTION USING SUPERVISED ML

In this task we have to predict the percentage score of a student based on the number of hour 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 [1]:

```
#importing required libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import r2_score,mean_squared_error
from math import sqrt

%matplotlib inline
```

In [2]:

```
#reading data from remote link
url= "http://bit.ly/w-data"
data=pd.read_csv(url)
print("Data imported successfully")
data.head(10)
```

```
Data imported successfully
```

Out[2]:

	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

Data imported successfully

In [3]:

```
data.shape
```

Out[3]:

```
(25, 2)
```

In [4]:

```
data.describe()
```

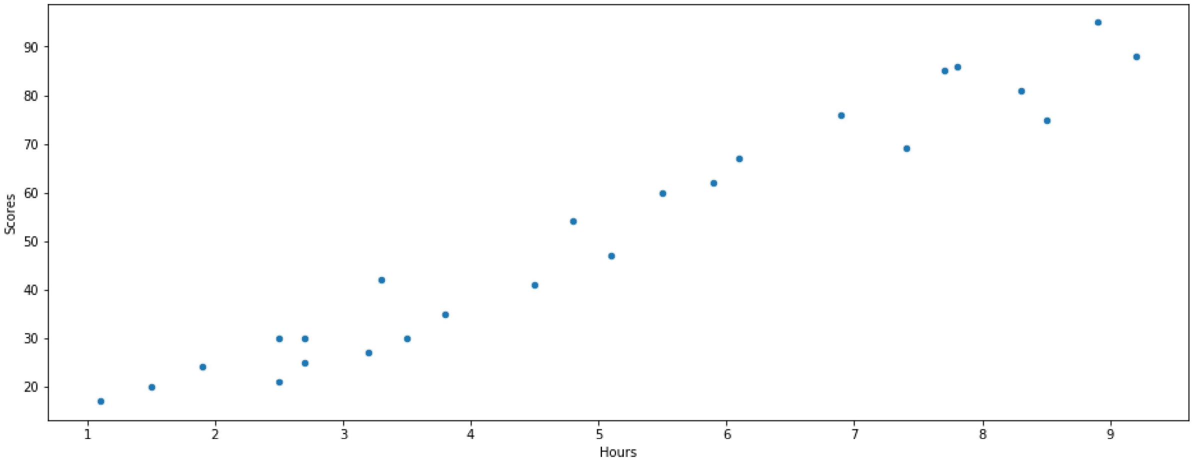
Out[4]:

	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

visualization

In [5]:
```
data.plot(kind="scatter",x="Hours",y="Scores",figsize=(16,6))
```

Out[5]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1c458ec1608>
```



In [6]:
```
x = data.iloc[:,:-1].values
y = data.iloc[:,1].values
```

Training the Algorithm

In [7]:
```
from sklearn.model_selection import train_test_split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size = 1/3, ran
dom_state = 0)
```

In [8]:
```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

Out[8]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=Fals
e)
```

Making Predictions

In [9]:
```
y_pred = regressor.predict(X_test)
print(y_pred)
```

```
[17.04289179 33.51695377 74.21757747 26.73351648 59.68164043 39.33132858
 20.91914167 78.09382734 69.37226512]
```

In [10]:
```
from statsmodels.sandbox.regression.predstd import wls_prediction_std
import statsmodels.api as sm
```

In [11]:
```
model1=sm.OLS(y_train,X_train)
result = model1.fit()
result.summary()
```

```
C:\Users\Shivam\anaconda3\lib\site-packages\scipy\stats\stats.py:1535: UserWa
rning: kurtosistest only valid for n>=20 ... continuing anyway, n=16
  "anyway, n=%i" % int(n))
```

Out[11]:

OLS Regression Results

Dep. Variable:	y	**R-squared (uncentered):**	0.991
Model:	OLS	**Adj. R-squared (uncentered):**	0.990
Method:	Least Squares	**F-statistic:**	1611.
Date:	Thu, 11 Mar 2021	**Prob (F-statistic):**	1.11e-16
Time:	16:08:15	**Log-Likelihood:**	-50.502
No. Observations:	16	**AIC:**	103.0
Df Residuals:	15	**BIC:**	103.8
Df Model:	1		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
x1	10.0780	0.251	40.132	0.000	9.543	10.613

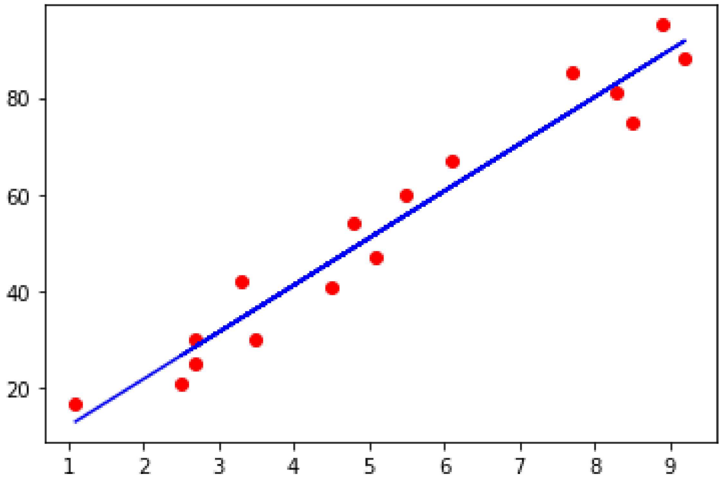
Omnibus:	2.476	**Durbin-Watson:**	2.079
Prob(Omnibus):	0.290	**Jarque-Bera (JB):**	1.124
Skew:	-0.191	**Prob(JB):**	0.570
Kurtosis:	1.759	**Cond. No.**	1.00

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Visualising the Training set results

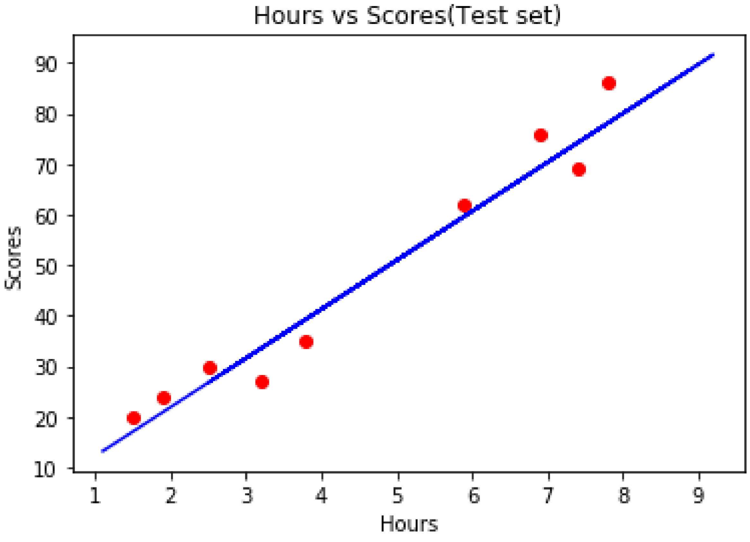
```
In [12]: plt.scatter(X_train, y_train, color = 'red')
         plt.plot(X_train, regressor.predict(X_train), color = 'blue')
```

```
Out[12]: [<matplotlib.lines.Line2D at 0x1c45a3cf248>]
```



Visualising the Test set results

```
In [13]: plt.scatter(X_test, y_test, color = 'red')
         plt.plot(X_train, regressor.predict(X_train), color = 'blue')
         plt.title('Hours vs Scores(Test set)')
         plt.xlabel('Hours')
         plt.ylabel('Scores')
         plt.show()
```



Question

what will be prdicted score if a student studies for 9.25 hrs/day?

```
In [14]: Hr=pd.DataFrame({'Hours':[9.25]})
         regressor.predict(Hr)

Out[14]: array([92.14523315])
```

Evaluating the model

```
In [15]: # from sklearn.metrics import mean_absolute_error
         from sklearn.metrics import mean_absolute_error
         y_pre=regressor.predict(X_test)
         mae = mean_absolute_error(y_test,y_pre)
```

```
In [16]: mae

Out[16]: 4.6913974413974415
```

```
In [ ]:
```