The Sparks Foundation

Graduate Rotational Internship Program(GRIP)

Domain-Data Science and Business Analytics

Name-Diksha Khade

Task -Prediction Using Supervised ML

Importing all the required libraries

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics
%matplotlib inline
```

Importing the data

In [3]:

```
url = 'http://bit.ly/w-data'
data_df = pd.read_csv(url)
```

Exploring the data

In [31]:

```
data_df.head()
```

Out[31]:

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

In [7]:

```
data_df.shape
```

Out[7]:

```
(25, 2)
```

In [8]:

```
data_df.describe()
```

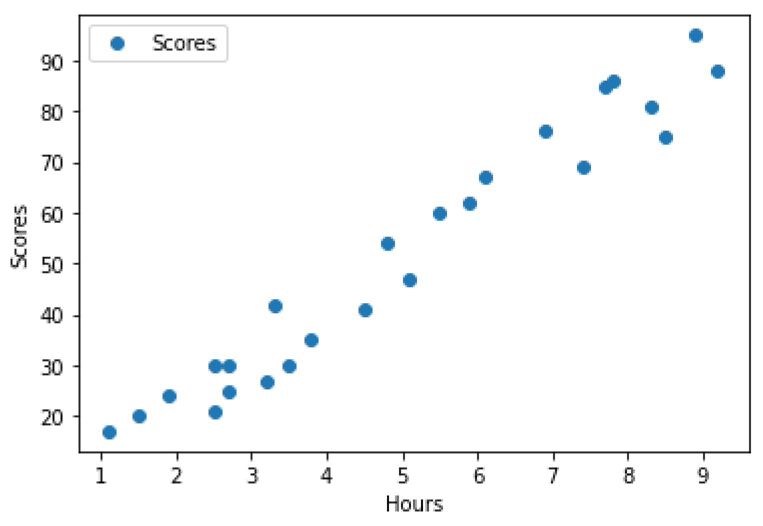
Out[8]:

	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

Plotting the data points

In [9]:

```
data_df.plot(x='Hours',y='Scores',style='o')
plt.xlabel('Hours')
plt.ylabel('Scores')
plt.show()
```



Preparing the data

In [10]:

```
X = data_df['Hours'].values.reshape(-1,1)
y = data_df['Scores'].values.reshape(-1,1)
```

In [16]:

```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2,random_state=0)
```

Training the algorithm

In [17]:

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

Out[17]:

```
LinearRegression()
```

In [18]:

```
#To retrieve the intercept
print(reg.intercept_)

#To retrieve the slope
print(reg.coef_)
```

```
[2.01816004]
[[9.91065648]]
```

Making Predictions

In [19]:

```
y_pred = reg.predict(X_test)
```

In [20]:

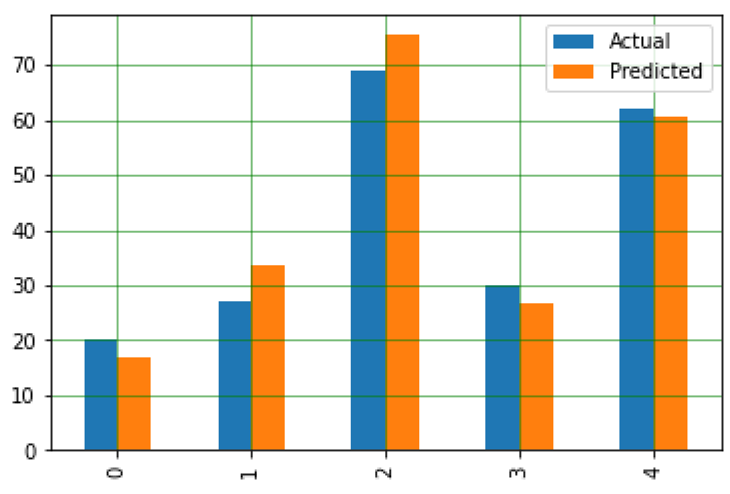
```
df = pd.DataFrame({'Actual':y_test.flatten(),'Predicted':y_pred.flatten()})
df
```

Out[20]:

	Actual	Predicted
0	20	16.884145
1	27	33.732261
2	69	75.357018
3	30	26.794801
4	62	60.491033

In [22]:

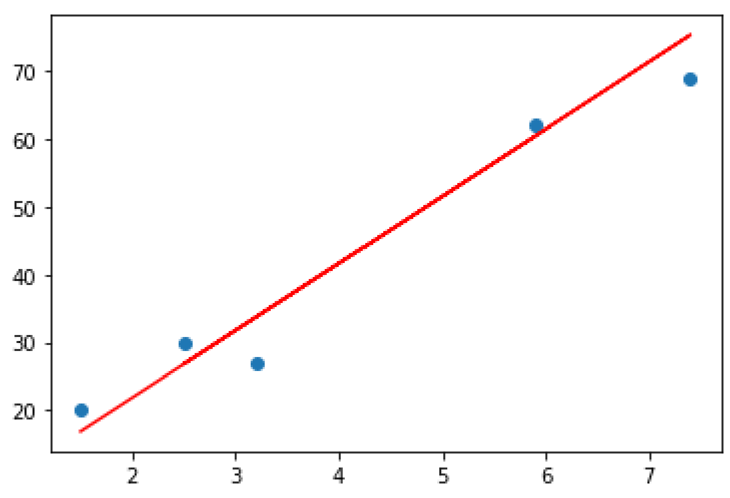
```
df.plot(kind='bar')
plt.grid(which='major',linestyle='-',linewidth='0.5',color='green')
plt.grid(which='minor',linestyle=':',linewidth='0.5',color='black')
plt.show()
```



Plotting straight line with test data

In [24]:

```
plt.scatter(X_test,y_test)
plt.plot(X_test,y_pred,color='red')
plt.show()
```



Evaluate Performance of the algorithm

In [25]:

```
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_test,y_pred)))
```

```
Mean Absolute Error = 4.183859899002975
Mean Squared Error = 21.598769307217406
Root Mean Squared Error = 4.647447612100367
```

what will be the predicted score if the student studies 9.25 hours/day ?

In [29]:

```
hours = [[9.25]]
reg.predict(hours)
```

Out[29]:

```
array([[93.69173249]])
```

The student will score 93 if studies for 9.25 hours/day