The Spark Foundation GRIPJUNE 2021

Task - 1 Prediction using Supervised ML

Importing the Packages

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

Reading the Data Set

In [2]:
```
url = "https://raw.githubusercontent.com/AdiPersonalWorks/Random/master/stude
```

In [3]:
```
df = pd.read_csv(url)
```

In [4]:
```
df.head()
```

Out[4]:

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

In [5]:
```
# Exploratry Data Analysis
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 [6]:
```
df.describe()
```

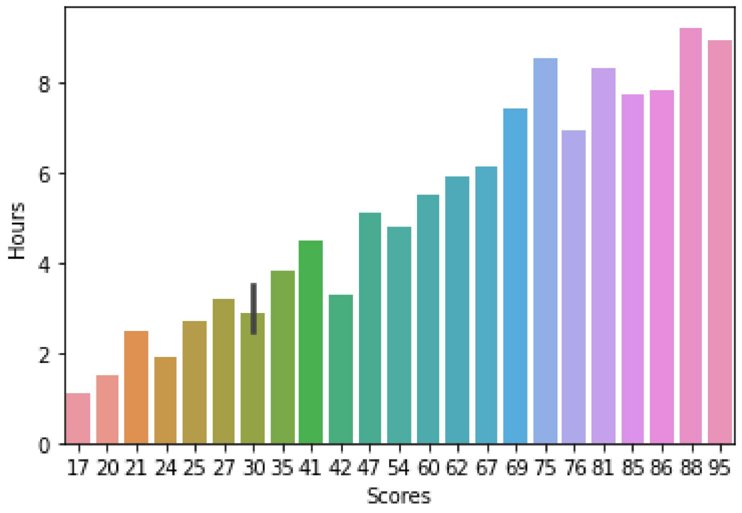
Out[6]:

	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

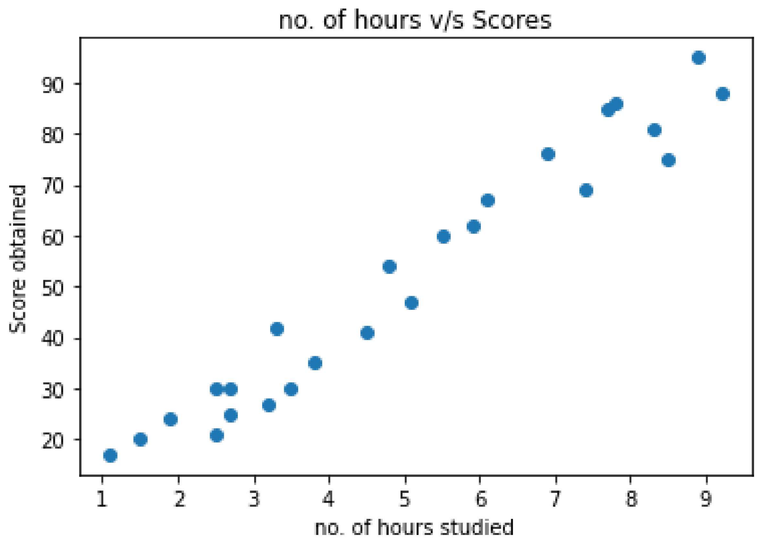
Data Visualization

In [7]:
```
# visualizing the data using bar graph
sns.barplot(x='Scores',y='Hours',data=df,estimator=np.mean,alpha=1)
```

Out[7]:
```
<AxesSubplot:xlabel='Scores', ylabel='Hours'>
```



In [8]:
```
# visualizing the data using scatter plot
plt.scatter(x="Hours",y="Scores",data=df)
plt.title('no. of hours v/s Scores')
plt.xlabel('no. of hours studied')
plt.ylabel('Score obtained')
plt.show()
```



Feature Selection

In [9]:
```
from sklearn.model_selection import train_test_split
```

In [10]:
```
x = df[["Hours"]]
y = df["Scores"]
```

Training and Testing the Data

In [11]:
```
x_train,x_test,y_train,y_test = train_test_split(x, y,test_size=0.25, random_
In [12]: x_train.head()
```

```
Out[12]:
```

	Hours
24	7.8
23	6.9
14	1.1
1	5.1
10	7.7

Training the Model

```
In [13]: from sklearn.linear_model import LinearRegression
```

```
In [14]: lm = LinearRegression()
```

Fitting the Data

```
In [15]: lm.fit(x_train,y_train)

Out[15]: LinearRegression()
```

```
In [16]: lm.intercept_

Out[16]: 1.932204253151646
```

```
In [17]: lm.coef_

Out[17]: array([9.94167834])
```

Predicting the Data

```
In [18]: predict = lm.predict([[9.25]])
```

```
In [19]: predict

Out[19]: array([93.89272889])
```

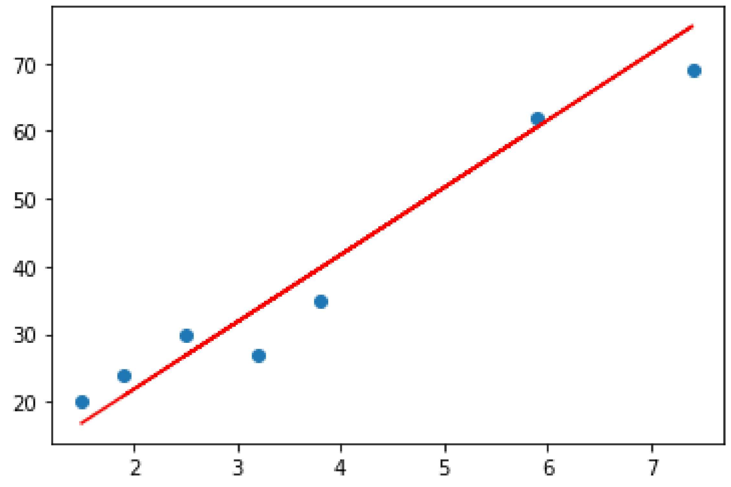
```
In [20]: p = lm.predict(x_test)
```

In [21]:
```
# Table for predicted values and tested values
df1=pd.DataFrame({'Actual':y_test,'Predicted':p})
df1
```

Out[21]:

	Actual	Predicted
5	20	16.844722
2	27	33.745575
19	69	75.500624
16	30	26.786400
11	62	60.588106
22	35	39.710582
17	24	20.821393

In [22]:
```
# Plotting the best fit line
plt.scatter(x_test, y_test)
plt.plot(x_test,p,color='red')
plt.show()
```



Evaluating the Data

In [23]:
```
from sklearn import metrics
```

In [24]:
```
r1=metrics.mean_absolute_error(y_test,p)
print("Mean Absolute Error    : ",r1)
```

```
Mean Absolute Error    :  4.130879918502486
```

In [25]:
```
r2=metrics.mean_squared_error(y_test,p)
print("Mean Squared Error      : ",r2)
```

```
Mean Squared Error      :  20.33292367497997
```

In [26]:
```
r3=np.sqrt(r2)
print("Root Mean Squared Error : ", r3)
```

```
Root Mean Squared Error :  4.5092043283688055
```

In [27]:
```
#answer for the question
hours=9.35
print(f"Number of hours of study are {hours}")
print(f"Predicted score {predict}")
```

```
Number of hours of study are 9.35
Predicted score [93.89272889]
```