PROBLEM STATEMENT:Predicting marks of a student, can score based on number of Hours of study.

APPROACH: To get the best optimised results we will use the supervised learning techniques to train various predictive models.

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

Loading Data Set

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

In [3]:
```
data
```

Out[3]:

	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

Overview of Dataset

In [4]:
```
data.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]:
```
data.tail
```

Out[5]:
```
<bound method NDFrame.tail of     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 [6]:
```
data.shape
```

Out[6]:
```
(25, 2)
```

Getting Data Information

In [7]:
```
data.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 [8]:
```
data.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

Checkning The Null Values

In [9]:
```
data.isnull().sum()
```

Out[9]:
```
Hours     0
Scores    0
dtype: int64
```

Split the dataset into dependent & independent variable:

In [10]:
```
marks=data.drop("Scores",axis="columns")
duration=data.drop("Hours",axis="columns")
```

In [11]:
```
marks.shape
```

Out[11]:
```
(25, 1)
```

In [12]:
```
duration.shape
```

Out[12]:
```
(25, 1)
```

train_test_split

In [13]:
```
from sklearn.model_selection import train_test_split
marks_train,marks_test,duration_train,duration_test=train_test_split(marks,duration,test_size=0.2,random_state=0)
```

In [14]:
```
marks_train.shape
```

Out[14]:
```
(20, 1)
```

In [15]:
```
duration_train.shape
```

Out[15]:
```
(20, 1)
```

In [16]:
```
duration_test.shape
```

Out[16]:
```
(5, 1)
```

In [17]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(marks_train,duration_train)
```

Out[17]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

Printing slope value(y=mx+c)

In [18]:
```
lr.coef_[0][0].round(2)
```

Out[18]:
```
9.91
```

Printing intercept value(y=mx+c)

In [19]:
```
lr.intercept_[0],round(2)
```

Out[19]:
```
(2.0181600414346683, 2)
```

In [20]:
```
m=9.91
c=2.02
y="9.25*c
y
```

Out[20]:
```
93.6875
```

In [21]:
```
lr.predict([[9.25]]).round(2)
```

Out[21]:
```
array([[93.69]])
```

In [22]:
```
score_pred=lr.predict(marks_test)
pd.DataFrame(np.c_[marks_test,duration_test,score_pred],columns=["Hours","originalscore","predicted score"])
```

Out[22]:

	Hours	originalscore	predicted score
0	1.5	20.0	16.884145
1	3.2	27.0	33.732261
2	7.4	69.0	75.357018
3	2.5	30.0	26.794801
4	5.9	62.0	60.491033

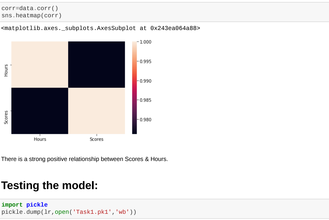
In [23]:
```
lr.score(marks_test,duration_test)
```

Out[23]:
```
0.9454906892105356
```

Select a model and train it:

In [24]:
```
graph=plt.figure(figsize=(15,8))
g1=graph.add_subplot(121)
g2=graph.add_subplot(122)
g1.set_title('Hours Distribution')
g2.set_title('Scores Distribution')
p1=sns.distplot(data['Hours'],label='Hours',ax=g1,color='red')
p2=sns.distplot(data['Scores'],label='Scores',ax=g2,color='blue')
p1.legend()
p2.legend()
```

Out[24]:
```
<matplotlib.legend.Legend at 0x243e9efa108>
```



Visualizing Relationship between the varibles

In [25]:
```
sns.regplot(x=data['Hours'],y=data['Scores'],color='r')
```

Out[25]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x243ea2353c8>
```

In [26]:
```
corr=data.corr()
sns.heatmap(corr)
```

Out[26]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x243ea964a88>
```


There is a strong positive relationship between Scores & Hours.

Testing the model:

In [27]:
```
import pickle
pickle.dump(lr,open('Task1.pk1','wb'))
```

In [28]:
```
model=pickle.load(open('Task1.pk1','rb'))
print(model.predict([[9.25]]))
```

```
[[93.69173249]]
```

After studying for 9.25 hours,The Expected Score should be 93.69.