Prediction Using Supervised Machine Lea

Name: Tejas Dhabu

Task: 1

In [2]:

```
# Importing all the necessary libraries
import numpy as np
from sklearn.linear_model import LinearRegression
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

In [3]:

```
# Reading data from given link
url = "http://bit.ly/w-data"
df = pd.read_csv(url)
print("Data imported successfully")

df
```

```
Data imported successfully
```

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

	Hours	Scores
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 [4]:

```
#Assigning the variable x
x = df.iloc[:, 0].values.reshape(-1,1)
x
```

Out[4]:

```
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 [5]:

```
# the independent variable should always be in 2 dimension in order perform regressi
x.shape
```

Out[5]:

```
(25, 1)
```

In [6]:

```
#Assigning the y variable
y = df.iloc[:, 1].values
y
```

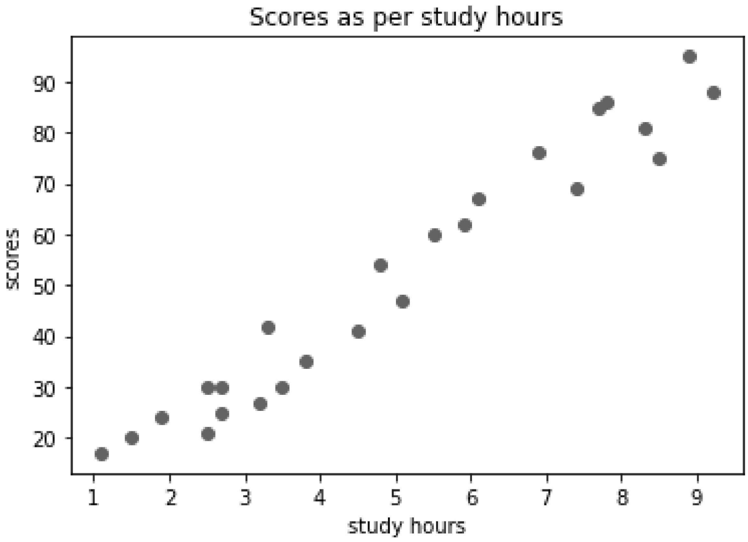
Out[6]:

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

In [9]:

```
#Ploting a scatter plot for the given dataset
plt.scatter(x,y)
plt.title("Scores as per study hours")
plt.xlabel("study hours")
plt.ylabel("scores")
```

Out[9]:
```
Text(0, 0.5, 'scores')
```



Step 2: Creating Simple Linear Regression Model and training it.

In [10]:
```
print("x: ", x,"y: ", y)
```

```
x:  [[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]] y:  [21 47 27 75 30 20 88 60 81 25 85 62 41 42 17 95 30 24 67 69 30 54 35 76
 86]
```

Spliting the Data: Training and Test Data

In [11]:
```
from sklearn.model_selection import train_test_split
```

In [12]:
```
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size = .3, random_stat
```

In [13]:

```
#Training the model
```

In [14]:

```
#printing the training data from the original dataset
print("x training data:", x_train)
print("y training data:", y_train)
```

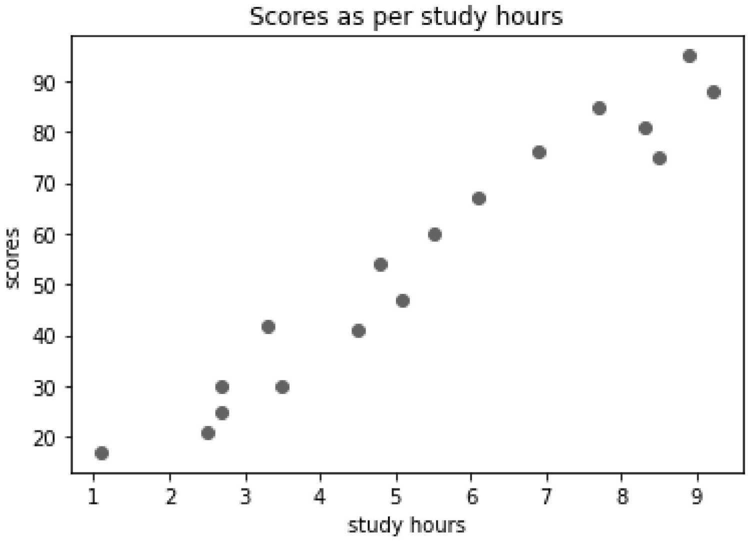
```
x training data: [[6.9]
 [1.1]
 [5.1]
 [7.7]
 [3.3]
 [8.3]
 [9.2]
 [6.1]
 [3.5]
 [2.7]
 [5.5]
 [2.7]
 [8.5]
 [2.5]
 [4.8]
 [8.9]
 [4.5]]
y training data: [76 17 47 85 42 81 88 67 30 25 60 30 75 21 54 95 41]
```

In [16]:

```
#scatter ploting the training dataset
plt.scatter(x_train,y_train)
plt.title("Scores as per study hours")
plt.xlabel("study hours")
plt.ylabel("scores")
```

Out[16]:

```
Text(0, 0.5, 'scores')
```



In [17]:

```
#creating a simple regression model for the training dataset
lr = LinearRegression()
model = lr.fit(x_train,y_train)
print("Training complete.")
```

```
Training complete.
```

In [18]:

```
model.intercept_ #y-intercept(regression constant)
```

Out[18]:

```
2.3708153823418883
```

```
In [19]: model.coef_ #regression coefficient, slope
```

```
Out[19]: array([9.78856669])
```

Formula: y =ax+b

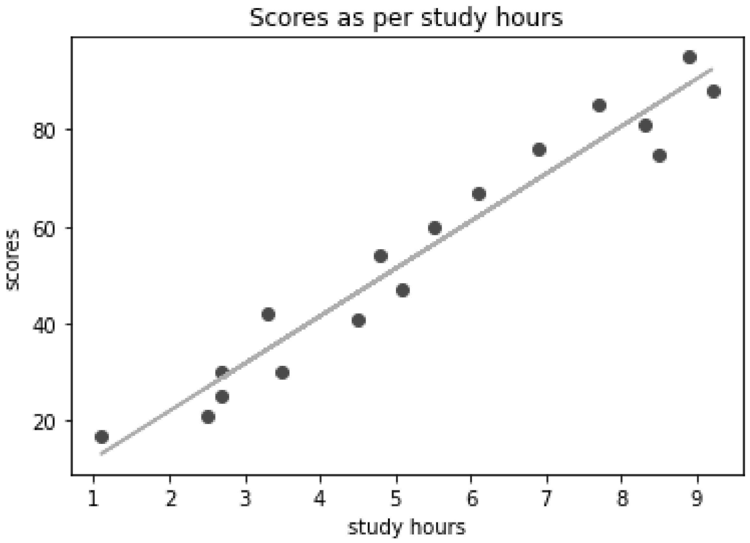
- **a** is the slope(estimated coefficient)
- **b** is the y-intercept

```
In [20]: #inserting the regression line in scatter plot of training dataset
         y_train_pred = model.coef_*x_train + model.intercept_

         plt.scatter(x_train,y_train, color = "green")
         plt.plot(x_train, y_train_pred, color = "orange")

         plt.title("Scores as per study hours")
         plt.xlabel("study hours")
         plt.ylabel("scores")
```

```
Out[20]: Text(0, 0.5, 'scores')
```



Step 3: Predicting using test data

```
In [21]: #predicting the scores for test data
         y_test_pred = model.predict(x_test)
         y_test_pred, y_test
```

```
Out[21]: (array([17.05366541, 33.69422878, 74.80620886, 26.8422321 , 60.12335883,
                39.56736879, 20.96909209, 78.72163554]),
          array([20, 27, 69, 30, 62, 35, 24, 86], dtype=int64))
```

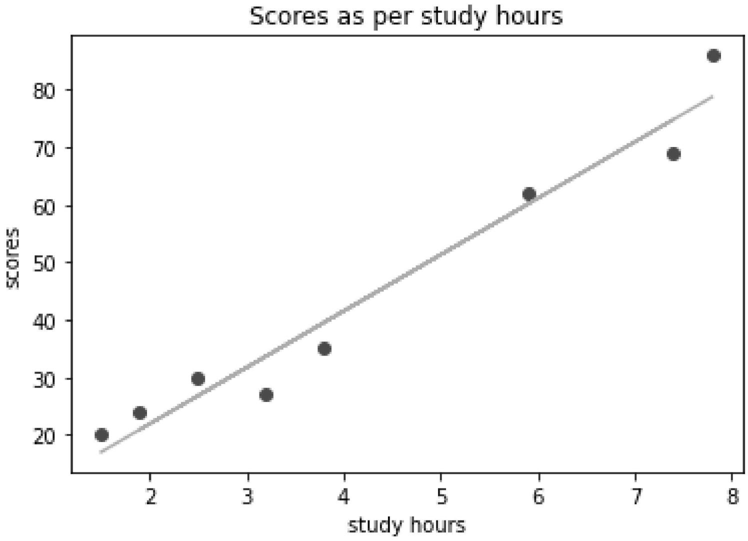
```
In [22]: #inserting the regression line in scatter plot of test dataset

         plt.scatter(x_test,y_test, color = "green")
         plt.plot(x_test, y_test_pred, color = "orange")

         plt.title("Scores as per study hours")
         plt.xlabel("study hours")
         plt.ylabel("scores")
```

Out[22]:

```
Text(0, 0.5, 'scores')
```



In [23]:

```
# we can also test with our own data
hours = 9.25
own_pred = model.predict([[hours]])
own_pred

print("At study hours,", hours, "per day, the model predicts the score to be: ", own
```

```
At study hours, 9.25 per day, the model predicts the score to be:  [92.91505723]
```

R-squared will give you an estimate of the relationship between movements of a dependent variable based on an independent variable's movements.

In [24]:

```
print("R Squared value: ",model.score(x_test,y_test)) # R squared - coefficient of d
```

```
R Squared value:  0.9568211104435257
```

In [25]:

```
# Comparing Actual vs Predicted
df = pd.DataFrame({'Actual': y_test, 'Predicted': y_test_pred})
df
```

Out[25]:

	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

Evaluating the performance of the model

The MAE measures the average magnitude of the errors in a set of forecasts, without considering their direction. It measures accuracy for continuous variables.

In [26]:

```
from sklearn import metrics
print('Mean Absolute Error:',
      metrics.mean_absolute_error(y_test, y_test_pred))
```

```
Mean Absolute Error: 4.419727808027651
```