THE SPARKS FOUNDATION

TASK 1

NAME: Yashika kushwaha

Task :Data Science & Buisness Analytic Supervised ML

OBJECTIVE:

1) Predict the percentage of an student based on the no. of study hours using supervised ML.

2) Predict what will be score if a student studies for 9.25 hrs/ day.

In [2]:

```
# Importing all libraries required in this notebook
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
%matplotlib inline
```

In [4]:

```
# Reading data from remote link
data = pd.read_csv("http://bit.ly/w-data")
print("Dataset successfully imported!")
```

```
Dataset successfully imported!
```

In [5]:

```
data
```

Out[5]:

	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

	Hours	Scores
24	7.8	86

In [6]:

```
# head of dataset
data.head()
```

Out[6]:

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

In [7]:

```
#tail of the dataset
data.tail()
```

Out[7]:

	Hours	Scores
20	2.7	30
21	4.8	54
22	3.8	35
23	6.9	76
24	7.8	86

In [8]:

```
data.shape
```

Out[8]:

```
(25, 2)
```

In [9]:

```
data.describe()
```

Out[9]:

	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

In [10]:

```
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 [11]:

```
#check and tell if dataset has null value it will return true otherwise false.
data.isnull()
```

Out[11]:

	Hours	Scores
0	False	False
1	False	False
2	False	False
3	False	False
4	False	False
5	False	False
6	False	False
7	False	False
8	False	False
9	False	False
10	False	False
11	False	False
12	False	False
13	False	False
14	False	False
15	False	False
16	False	False
17	False	False
18	False	False
19	False	False
20	False	False
21	False	False
22	False	False

	Hours	Scores
23	False	False
24	False	False

In [12]:

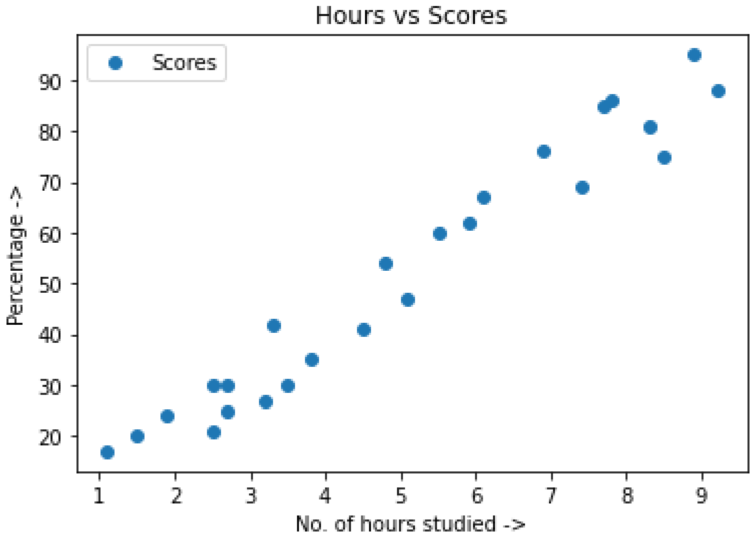
```
data.isnull().sum()
```

Out[12]:

```
Hours     0
Scores    0
dtype: int64
```

In [13]:

```
#scatter plot
data.plot(x='Hours', y='Scores', style = 'o')
plt.title("Hours vs Scores")
plt.xlabel("No. of hours studied ->")
plt.ylabel("Percentage ->")
plt.show()
```

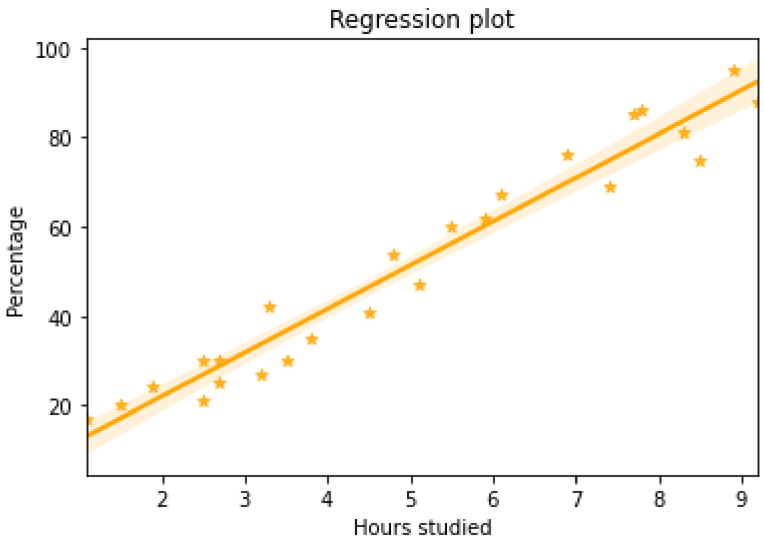


From the above graph, we can infer that there is a positive linear relation between the number of hours studied and percentage of score.

In [14]:

```
#regression plot

sns.regplot(x = data['Hours'], y = data['Scores'], marker= '*', color= 'orange')
plt.title('Regression plot')
plt.xlabel('Hours studied')
plt.ylabel('Percentage')
plt.show()
```

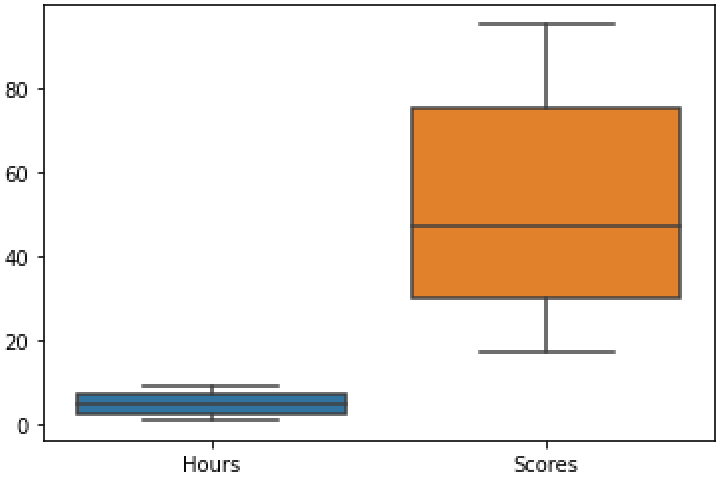


In [15]:

```
#Box plot
sns.boxplot(data= data)
```

Out[15]:

```
<AxesSubplot:>
```



From the above, plot we can see there are not outiler in dataset.

In [16]:

```
# Data pre-processing to fetch input / independent variable & dependent attribute

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

In [17]:

```
x
```

Out[17]:

```
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 [18]:

```
y
```

Out[18]:

```
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 [19]:

```
# Training and test spiltting
X_train , X_test , Y_train, Y_test = train_test_split(x , y , test_size=0.2, random_state = 0)
```

Implementing linear regression algorithm

In [20]:

```
from sklearn.linear_model import LinearRegression

regressor = LinearRegression()
regressor.fit(X_train , Y_train)      # Complete training
```

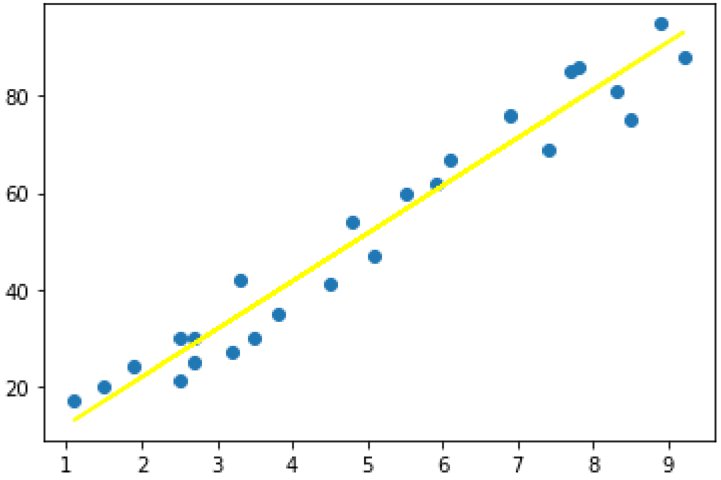
Out[20]:

```
LinearRegression()
```

In [22]:

```
# Plotting the regression line
line = regressor.coef_*x+regressor.intercept_

# plotting for the test data
plt.scatter(x, y)
plt.plot(x, line, color ='yellow')
plt.show()
```



In [23]:

```
# Testing our Algorithm

print(X_test)  # Testing data - In Hours
y_pred = regressor.predict(X_test)     # Predicting the scores
```

```
[[1.5]
 [3.2]
 [7.4]
 [2.5]
 [5.9]]
```

In [24]:

```
# Creating a data frame of actual and predicted values

data_frame = pd.DataFrame({'Actual' : Y_test, 'Predicted' : y_pred})
data_frame
```

Out[24]:

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

In [25]:

```
# Checking the percentage on the given data point(studey hours = 9.25)

hours = [[9.25]]
own_pred = regressor.predict(hours)
print("No of Hours ={}".format(hours))
print("Predicted Score = {}".format(own_pred[0]))
```

```
No of Hours =[[9.25]]
Predicted Score = 93.69173248737538
```

In [26]:

```python
# Checking the performance of algorithm

from sklearn import metrics
print("Mean Absolute error is : " , metrics.mean_absolute_error(Y_test, y_pred))
```

```
Mean Absolute error is :  4.183859899002975
```

Conclusion : Hence, we concluded that if a study studies for 9.25 per day, then their is a possibilty of perentage comes out to be 93.6917%

In [ ]:

```

```