The Sparks Foundation

Graduate Rotational Internship Program (GRIP)

NAME - BARIRA SIDDIQUI

DATA SCIENCE AND BUSINESS ANALYTICS INTERN AT THE SPARKS FOUNDATION

TASK 1 - PREDICTING USING SUPERVISED MACHINE LEARNING

AIM- TO PREDICT THE SCORES OF STUDENTS BASED ON NUMBER OF HOURS STUDIED

LINK OF THE DATA- http://bit.ly/w-data

IMPORTING THE LIBRARIES

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

IMPORTING DATA

In [3]:
```
student_data=pd.read_csv("http://bit.ly/w-data")#data set stored in variable student_data
student_data.head(5)#printing the 1st 5 entries of the data set
```

Out[3]:

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

VISUALIZING THE DATA SET

In [6]:
```
student_data.shape
```

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

In [7]:
```
student_data.isnull().sum() # checking for null values
```

Out[7]:
```
Hours     0
Scores    0
dtype: int64
```

In [4]:
```
student_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
```

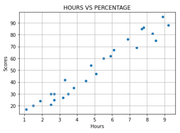
In [5]:
```
student_data.describe()
```

Out[5]:

	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

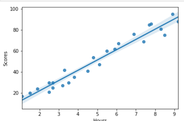
PLOTTING THE SCATTERPLOT

In [37]:
```
student_data.plot.scatter(x='Hours',y='Scores')
plt.title('HOURS VS PERCENTAGE')
plt.grid()
plt.show()
```



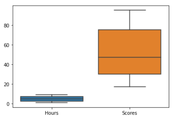
PLOTTING DIFFERENT GRAPHS OF THE DATA USING SEABORN

In [9]:
```
#Scatterplot
sbn.regplot(x=student_data['Hours'],y=student_data['Scores']);
```



In [10]:
```
#Boxplot
sbn.boxplot(data=student_data[["Hours","Scores"]])
```

Out[10]:
```
<AxesSubplot:>
```



STORING DATA INTO AN ARRAY

In [12]:
```
x=student_data.iloc[:,:-1].values
y=student_data.iloc[:,1].values
```

In [13]:
```
x
```

Out[13]:
```
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 [15]:
```
y
```

Out[15]:
```
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)
```

SPLITTING THE STUDENT_DATA FOR TRAINING AND TEST SET

In [16]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y,
                                   test_size=0.25, random_state=0)

#We have allocated 25% of student_data for testing and 75% of student_data for training purpose
```

TRAINING THE ALGORITHM

Simple Linear Regression

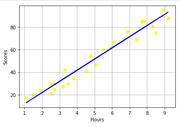
We will predict the percentage of marks of that a student is expected to score based on number of hours he/she has studied.

In [17]:
```
from sklearn.linear_model import LinearRegression
regressor=LinearRegression()
regressor.fit(x_train,y_train)
```

Out[17]:
```
LinearRegression()
```

Plotting Regression line

In [19]:
```
line = regressor.coef_*x+regressor.intercept_
student_data.plot.scatter(x="Hours", y="Scores")
plt.scatter(x,y, color= 'yellow');
plt.plot(x, line,color='blue');
plt.grid()
plt.show()
```



The above regression graph clearly shows the positive relation between the scores and hours studied by a student

In [21]:
```
y_pred=regressor.predict(x_test)
print(y_pred)
```

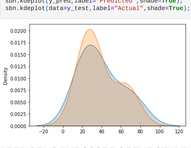
```
[16.84472176 33.74557494 75.50062397 26.7864001  60.58810646 39.71058194
 20.8213931 ]
```

In [22]:
```
df= pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
df
```

Out[22]:

	Actual	Predicted
0	20	16.844722
1	27	33.745575
2	69	75.500624
3	30	26.786400
4	62	60.588106
5	35	39.710582
6	24	20.821393

In [23]:
```
sbn.kdeplot(y_pred,label="Predicted",shade=True);
sbn.kdeplot(data=y_test,label="Actual",shade=True);
```



PREDICTING THE SCORE OF THE STUDENT STUDIES 9.5 HR/DAY

In [34]:
```
hours=9.5
predic=regressor.predict([[hours]])
print("Number of Hours = {}".format(hours))
print("Predicted score = {:.2f}".format(predic[0]))
```

```
Number of Hours = 9.5
Predicted score = 96.38
```

Evaluating the Model

In [35]:
```
from sklearn import metrics
print('Mean Absolute Error:',
      metrics.mean_absolute_error(y_test, y_pred))
```

```
Mean Absolute Error: 4.130879918502486
```