TASK I: PREDICTION USING SUPERVISED MACHINE LEARNING

AUTHOR: NEHA HRISHIKESH DEODHAR

DATASET USED FOR THE FOLLOWING PROJECT:http://bit.ly/w-data

OVERVIEW: PREDICTION OF PERCENTAGE ATTAINED BY THE STUDENTS BASED UPON THE NUMBER OF HOURS OF STUDY

DEPENDANT VARIABLE: Percentage Of Students

INDEPENDANT VARIABLE: Number Of Hours Of Study

In [1]:

```
#Importing Required Libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:

```
#Importing Dataset From The Given Link And Displaying It
data=pd.read_csv("http://bit.ly/w-data")
data
```

Out[2]:

	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
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 [3]:

```
#Checking The Dimensions Of The Dataset
data.shape
```

Out[3]:

```
(25, 2)
```

In [4]:

```
data.describe
```

Out[4]:

```
<bound method NDFrame.describe 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 [5]:

```
#Checking The Datatypes
data.dtypes
```

Out[5]:

```
Hours     float64
Scores      int64
dtype: object
```

In [6]:

```
#Checking For Null Values
data.isnull().sum()
```

Out[6]:

```
Hours     0
Scores    0
```

```
dtype: int64
```

In [7]:

```
#Checking For Duplicated Values
data.duplicated().sum()
```

Out[7]:

```
0
```

In [8]:

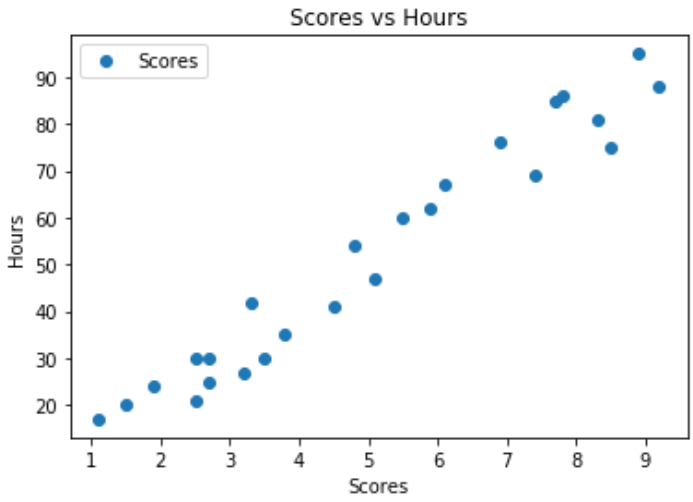
```
#Displaying The Column Names
data.columns
```

Out[8]:

```
Index(['Hours', 'Scores'], dtype='object')
```

In [9]:

```
#Scatterplot Of Hours VS Scores
data.plot(x='Hours',y='Scores',style='o')
plt.title("Scores vs Hours")
plt.xlabel("Scores")
plt.ylabel("Hours")
plt.show()
```

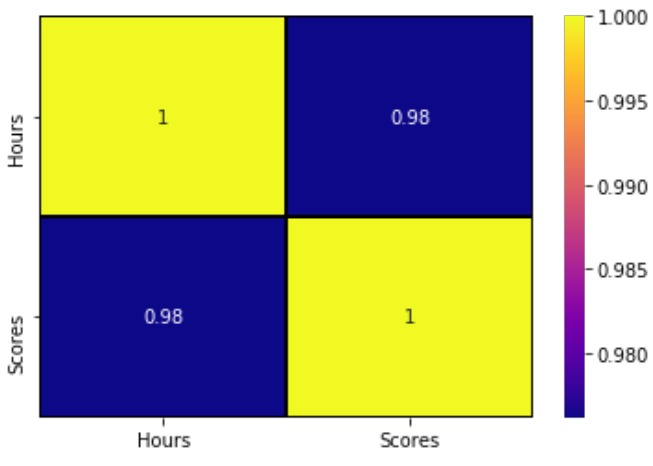


In [10]:

```
#HeatMap To Generate Correlation
tc = data.corr()
sns.heatmap(tc, annot = True, cmap ='plasma',
            linecolor ='black', linewidths = 1)
```

Out[10]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2ab275afb88>
```



In [11]:

```
#Reshaping The Data
X = data.iloc[:, :-1].values
y = data.iloc[:, 1].values
X=X.reshape(-1,1)
```

In [12]:

```
#Splitting The Dataset
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                test_size=0.2, random_state=0)
```

In [13]:

```
#Training The DataSet
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train,y_train)
print("Dataset Training Completed!!")
```

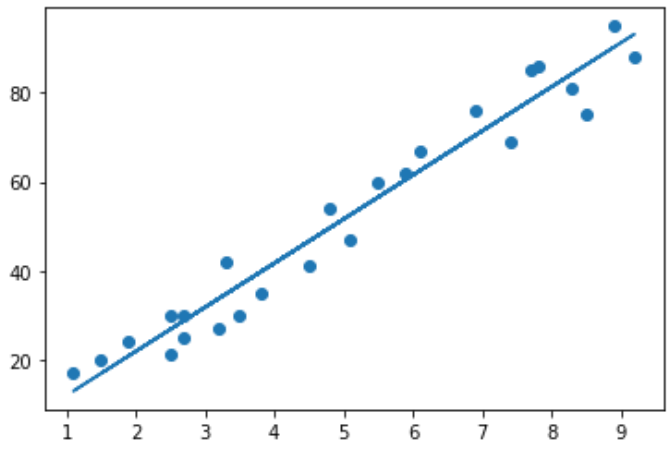
```
Dataset Training Completed!!
```

In [14]:

```
# Plotting The Regression Line
a = regressor.coef_
b = regressor.intercept_,
print(a)
print(b)
line = a*X+b

# Plotting Regression Line For Test Data
plt.scatter(X, y)
plt.plot(X, line);
plt.show()
```

```
[9.91065648]
(2.018160041434683,)
```



In [15]:

```
#Predicting The Scores
print(X_test)
y_pred = regressor.predict(X_test)
```

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

In [16]:

```
#Comparison Between Expected And Predicted Results
```

```
#Comparison Between Expected And Predicted Results
data = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred,'Error':((y_test-y_pred)*100)
/y_test})
data
```

Out[16]:

	Actual	Predicted	Error
0	20	16.884145	15.579276
1	27	33.732261	-24.934299
2	69	75.357018	-9.213070
3	30	26.794801	10.683996
4	62	60.491033	2.433817

In [17]:

```
#Predicting Score For 9.25 Hours/Day Study
print('For X = 9.25 hours Predicted Score:',regressor.predict([[9.25]]))
```

```
For X = 9.25 hours Predicted Score: [93.69173249]
```

In [18]:

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

```
Mean Absolute Error: 4.183859899002975
```

In [19]:

```
#Calculating R2 Score
from sklearn.metrics import r2_score
print ("R2 Score: {:.4f}".format(r2_score(y_test, y_pred)))
```

```
R2 Score: 0.9455
```

FROM THE ABOVE IT IS DEDUCED THAT THE ACCURACY OF THE MODEL IS 94.55%