Ponnuru Sai Venkatesh

Data Science & Business Analytics

GRIP @ The Sparks Foundation

Task 1 - Prediction using Supervised ML

- Predict the percentage of an student based on the no. of study hours.

In [ ]:

```
# Import necessary libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

In [3]:

```
# Importing/Reading the data

url = "http://bit.ly/w-data"
data = pd.read_csv(url)
```

In [4]:

```
# Displaying head of the data

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 [54]:

```
# Displaying tail of the data


data.tail()
```

Out[54]:

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

In [11]:

```
# Finding the data type of the data

data.dtypes
```

Out[11]:

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

In [8]:

```
# Describing the data

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

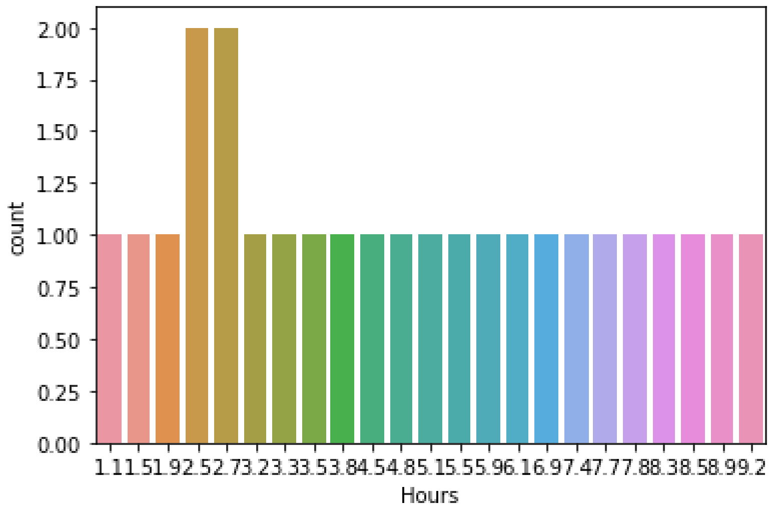
In [14]:

```
# Countplot for "Hours"

sns.countplot(x="Hours",data=data)
```

Out[14]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1c023cb8730>
```



- This plot shows the count for number of hours

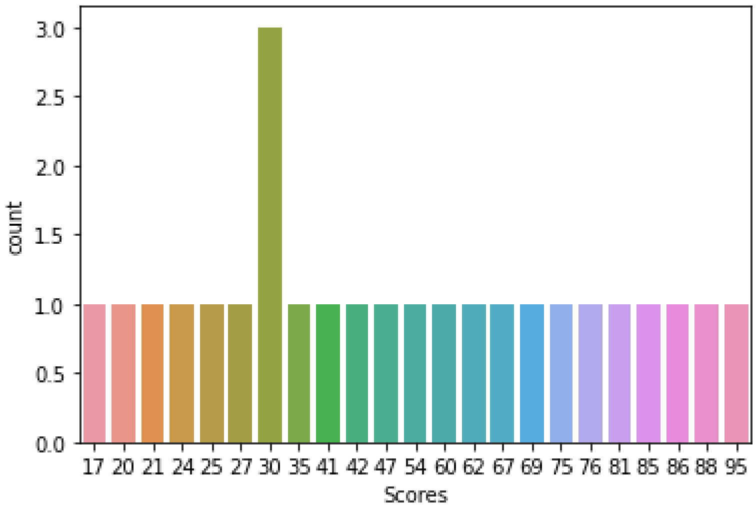
In [55]:

```
# Countplot for "Scores"


sns.countplot(x="Scores",data=data)
```

Out[55]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1c025b17be0>
```

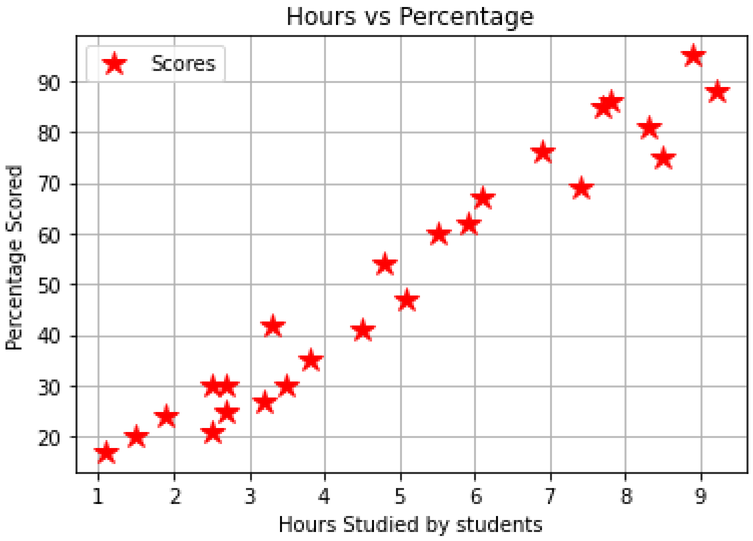


- This plot shows the count for Percentage Scored

In [58]:

```
# Plotting the distribution of scores

data.plot(x='Hours', y='Scores', style='*', color='red', markersize=13)
plt.title('Hours vs Percentage')
plt.xlabel('Hours Studied by students')
plt.ylabel('Percentage Scored')
plt.grid()
plt.show()
```



- This plot depicts the positive linear relation between the number of hours studied and percentage scored.

In [25]:

```
# spliting the dataset into dependent and independent values by using  "iloc" Function

X = data.iloc[:, :-1].values
Y = data.iloc[:, 1].values
```

In [ ]:

```
# training and testing the dataset using "train-test-split" function.

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 [26]:

```
# Training the Algorithm

from sklearn.linear_model import LinearRegression
model = LinearRegression()
```

In [27]:

```
# Fitting the model

model.fit(X_train, Y_train)
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

Out[27]:

```
LinearRegression()
```

In [28]:

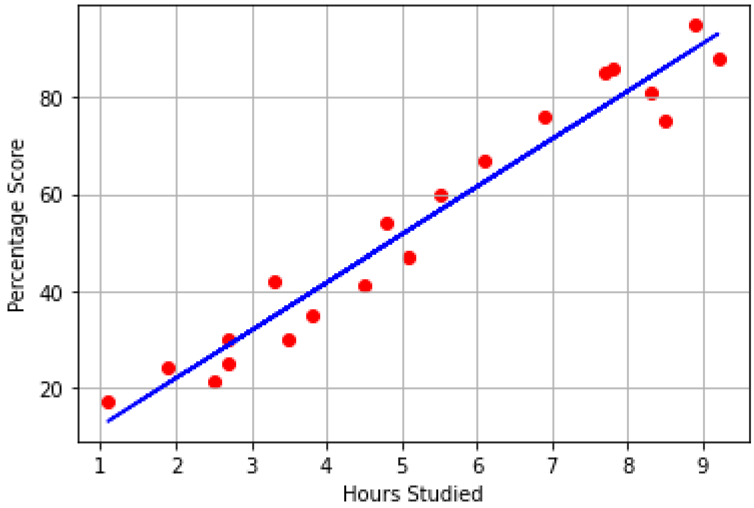
```
# Plotting the regression line

line = model.coef_*X + model.intercept_
```

In [33]:

```
# Plotting for the train data


plt.scatter(X_train, Y_train, color='red')
plt.plot(X, line, color='blue');
plt.xlabel('Hours Studied')
plt.ylabel('Percentage Score')
plt.grid()
plt.show()
```

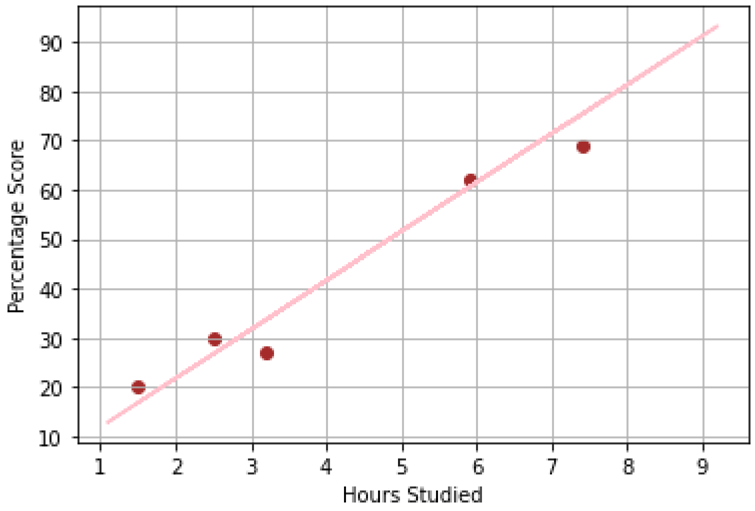


- This Scatter Plot depicts "Percentage scored by the students" with respect to their "Hours studied" for trained data.

In [60]:

```
# Plotting for the test data


plt.scatter(X_test, Y_test, color='brown')
plt.plot(X, line, color='pink');
plt.xlabel('Hours Studied')
plt.ylabel('Percentage Score')
plt.grid()
plt.show()
```



- This Scatter Plot depicts "Percentage scored by the students" with respect to their "Hours studied" for testing data.

In [65]:

```
# Predicting the model

Y_predicted = model.predict(X_test)
```

In [68]:

```
# Comparision of Real and Predicted Class values

df = pd.DataFrame({'Actual score': Y_test, 'Predicted score': Y_predicted})
df
```

Out[68]:

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

In [63]:

```
# Finding the Percentage

hrs = 9.25
Score_prediction = model.predict([[hrs]])
print("The predicted score, if a person studies for",hrs,"hours is",Score_prediction[0])
```

```
The predicted score, if a person studies for 9.25 hours is 93.69173248737538
```

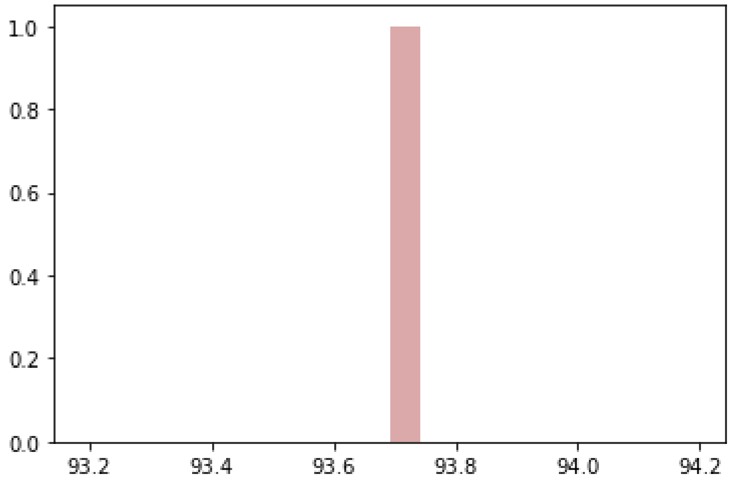
In [69]:

```
# Plot for Percentage Scored

sns.distplot(own_prediction[0],color='brown',bins=20,kde=False)
```

Out[69]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1c0245ebbb0>
```



Evaluating the model

In [70]:

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

```
Mean Absolute Error: 4.183859899002975
```