MEDAM.BHANUTEJ

Data Science & BusinessAnalytics

GRIP @ The Sparks Foundation

TASK 1 (Prediction using Supervised ML)

Predicting the percentage of an student based on the no. of study hours.

In [61]:

```
#import neccesarry libraries

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

In [62]:

```
#importing data

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

Out[62]:

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

In [4]:

```
data.tail(5)
```

Out[4]:

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

In [5]:

```
#describing the data


data.describe()
```

Out[5]:

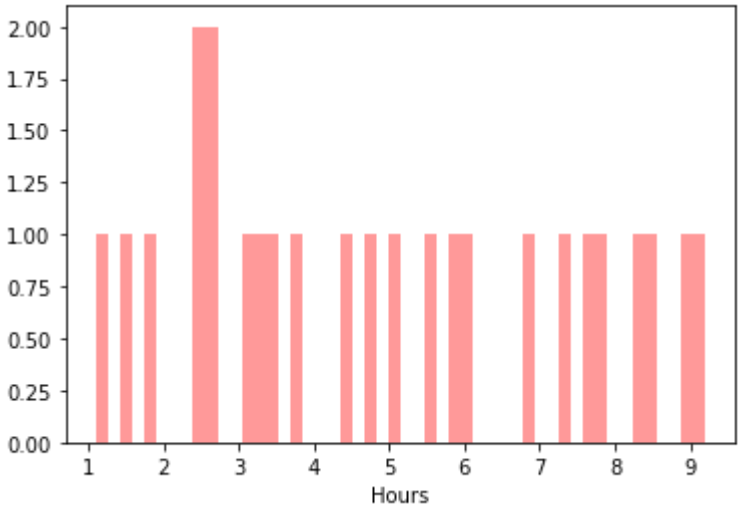
	Hours	Scores
count	25.000000	25.000000
mean	5.012000	51.480000
std	2.525094	25.286887
min	1.100000	17.000000
25%	2.700000	30.000000
50%	4.800000	47.000000
75%	7.400000	75.000000
max	9.200000	95.000000

In [49]:

```
#plotting graph on hours


sns.distplot(data['Hours'],color='red',bins=50,kde=False)
```

Out[49]:

<AxesSubplot:xlabel='Hours'>

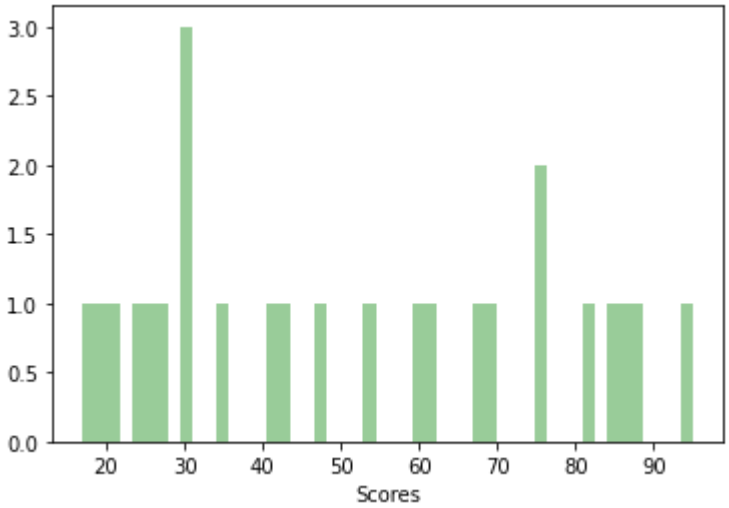


In [14]:

```
#plotting the graph on  scores


sns.distplot(data['Scores'],color='green',bins=50,kde=False)
```

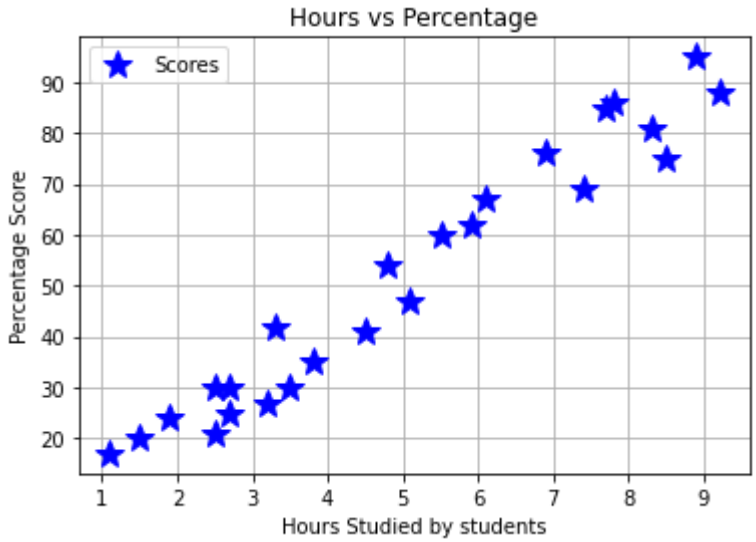
Out[14]:

<AxesSubplot:xlabel='Scores'>



In [24]:

```
#hours vs scores  graph

data.plot(x='Hours', y='Scores', style='*', color='blue', markersize=15)
plt.title('Hours vs Percentage')
plt.xlabel('Hours Studied by students')
plt.ylabel('Percentage Score')
plt.grid()
plt.show()
```



In [63]:

```
#splitting the data into dependent and independent values

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

In [64]:

```

#training and testing the data set

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 [65]:

```
from sklearn.linear_model import LinearRegression
```

In [66]:

```
#fitting the model data

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

Out[66]:

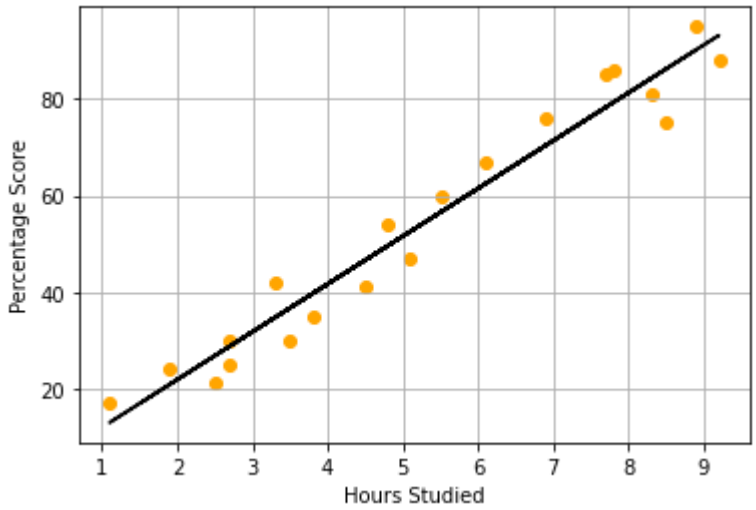
```
LinearRegression()
```

In [30]:

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

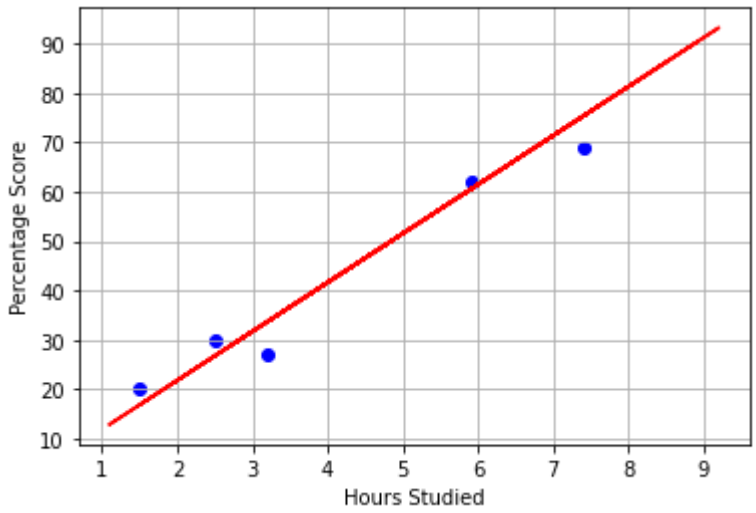
In [68]:

```
#graph plotting for training data

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



In [38]:

```
#graph plotting for testing data

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



In [12]:

```
#predicting the model

Y_predicted = model.predict(X_test)
```

In [13]:

```
#comparision between real class values and predicted class values

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

Out[13]:

	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 [52]:

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

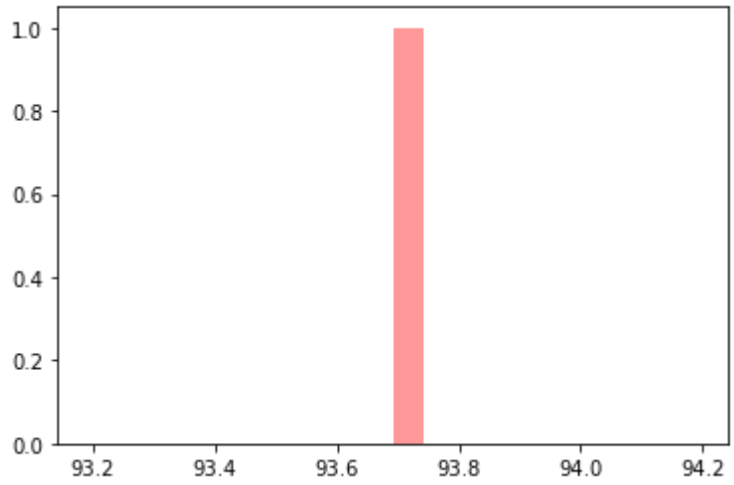
```
The predicted score if a person studies for 9.25 hours is 93.69173248737538
```

In [59]:

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

Out[59]:

<AxesSubplot:>



In [69]:

```
#evaluating the model data


from sklearn import metrics
```

In [16]:

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

```
Mean Absolute Error: 4.183859899002975
```

In [ ]:

```

```