Predict the percentage of a student based on the no. of study hours

Model : Simple Linear Regression

In this task,we will predict the percentage of marks that a student is expected to score based upon the number of hours they studied. This is a simple linear regression task as it involves just two variables

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

In [2]:
```
url="http://bit.ly/w-data"
data = pd.read_csv(url)
```

Data Overview

In [3]:
```
data.head(10)
```

Out[3]:

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

	Hours	Scores
5	1.5	20
6	9.2	88
7	5.5	60
8	8.3	81
9	2.7	25

In [4]:
```
data.describe()
```

Out[4]:

	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

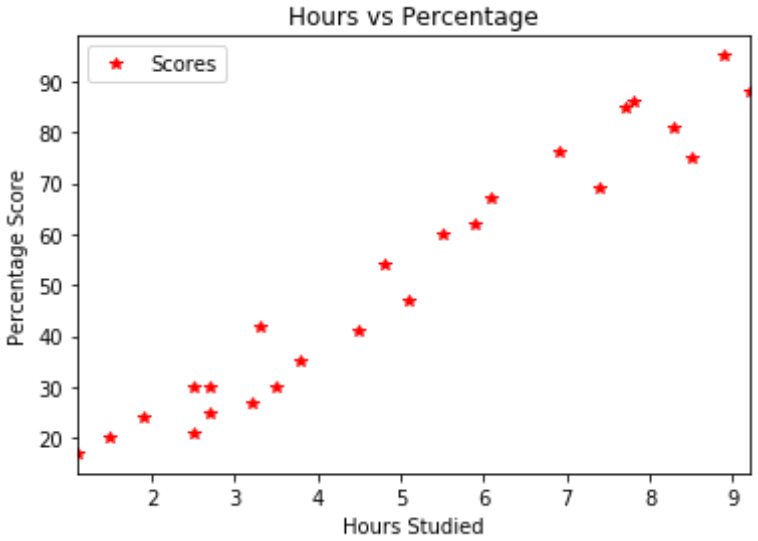
In [5]:
```
data.isnull().sum()
```

Out[5]:
```
Hours     0
Scores    0
dtype: int64
```

Let's plot our data points on 2-D graph to eyeball our dataset and see if we can manually find any relationship between the data. We can create the plot with the following script:

In [6]:
```
data.plot(x='Hours',y='Scores',style='*',color='red')
```

```
plt.title('Hours vs Percentage')
plt.xlabel('Hours Studied')
plt.ylabel('Percentage Score')
plt.show()
```



The above graph shows that there is a positive linear relationship between hours studied and percentage score

Making the data to divide into "attributes"(inputs) and "labels"(outputs)

In [7]:
```
X=data.iloc[:,:-1].values
y=data.iloc[:,1].values
```

As we have our attributes and labels. Further step is to split the data into training and testing sets by using train_test_split() method()

In [8]:
```
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,rand
```

```
om_state=0)
```

Training the Algorithm

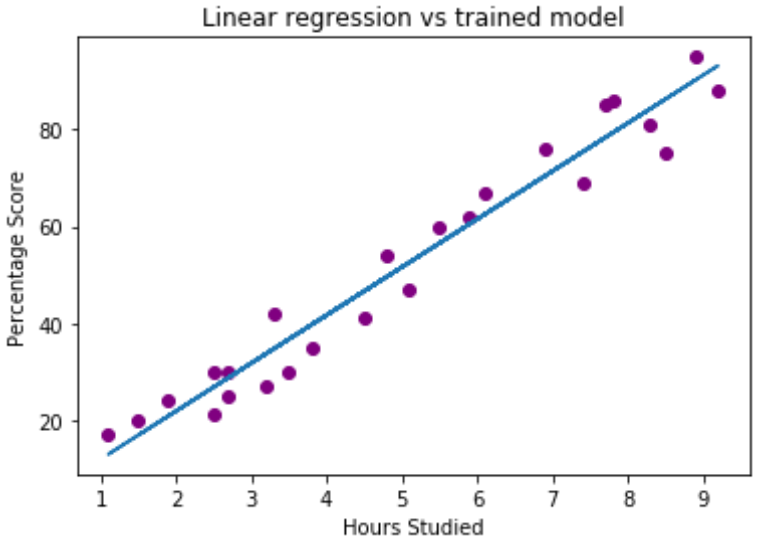
We have split our data into training and testing sets, and now is finally the time to train our algorithm.

In [9]:
```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)

print("Training complete.")
```

```
Training complete.
```

In [10]:
```
# Plotting the regression line
line = regressor.coef_*X+regressor.intercept_
# Plotting for the test data
plt.title("Linear regression vs trained model")
plt.scatter(X,y,color='purple')
plt.xlabel('Hours Studied')
plt.ylabel('Percentage Score')
plt.plot(X,line)
plt.show()
```



Making Predictions

Now that we have trained our algorithm, it's time to make some predictions.

In [11]:
```
print(X_test)
y_pred = regressor.predict(X_test)
```

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

In [12]:
```
df = pd.DataFrame({'Actual':y_test, 'Predicted':y_pred})
df
```

Out[12]:

	Actual	Predicted
0	20	16.884145

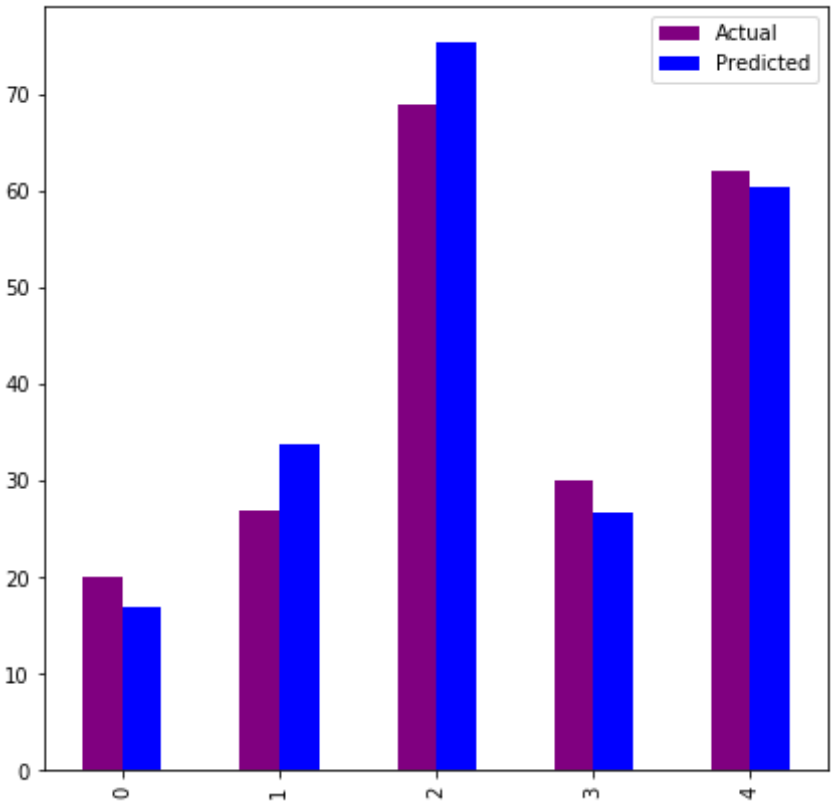
	Actual	Predicted
1	27	33.732261
2	69	75.357018
3	30	26.794801
4	62	60.491033

In [13]:
```
print("Training Score ",regressor.score(X_train,y_train))
print("Testing Score ",regressor.score(X_test,y_test))
```

```
Training Score  0.9515510725211553
Testing Score  0.9454906892105356
```

plotting the bar char to depict the actual and predict value

In [14]:
```
df.plot(kind='bar',figsize=(7,7),color=('purple','blue'))
plt.show()
```



In [15]:
```
hours = 9.25
test = np.array([hours])
test = test.reshape(-1,1)
pred = regressor.predict([[9.5]])
print("NO. of hours = {}".format(hours))
print("Predicted Score = {}".format(pred[0]))
```

```
NO. of hours = 9.25
Predicted Score = 96.16939660753593
```

Evaluating the model

The final step is to evaluate the performance of algorithm. This step is particularly important to compare how well different algorithms perform on a particular dataset.

In [16]:
```
from sklearn import metrics
print('Mean Absolute Error:',metrics.mean_absolute_error(y_test,y_pred
))
print('Mean Squared Error:',metrics.mean_squared_error(y_test,y_pred))
print('Root Mean Squared Error:',np.sqrt(metrics.mean_squared_error(y_t
est,y_pred)))
print('Explained Variance Score:',metrics.explained_variance_score(y_te
st,y_pred))
```

```
Mean Absolute Error: 4.183859899002975
Mean Squared Error: 21.5987693072174
Root Mean Squared Error: 4.6474476121003665
Explained Variance Score: 0.9482829156738147
```

Result:We have predicted that after studying for 9.25 hours student will get 96.16 percentage.

In [ ]:
```

```