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

Name-shital more

Importing Libraries

In [2]:

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

The following command imports the CSV dataset using pandas:

In [13]:

```
df=pd.read_csv("C:\\Users\Shri Krupa\\Downloads\\student_scores.csv")
```

In [15]:

```
df.head()
```

Out[15]:

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

In [16]:

```
df.shape
```

Out[16]:

```
(25, 2)
```

In [18]:
```
df.describe()
```

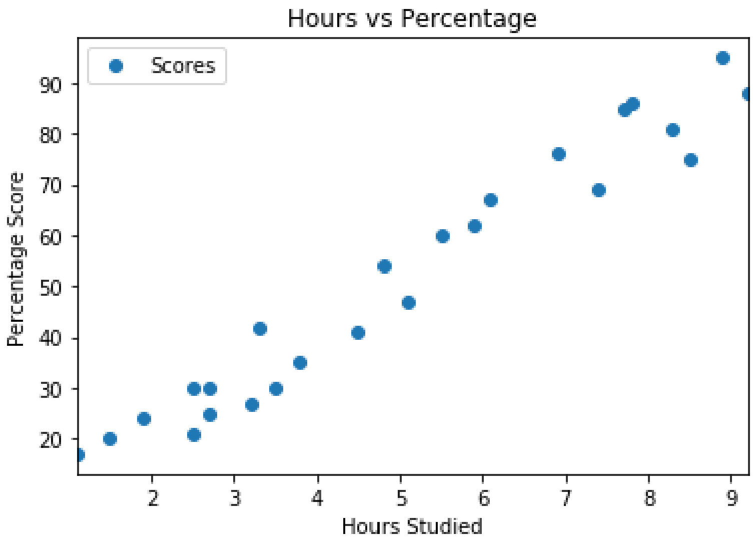
Out[18]:

	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

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 [19]:
```
df.plot(x='Hours', y='Scores', style='o')
plt.title('Hours vs Percentage')# Name title of the graph
plt.xlabel('Hours Studied') # Assign the name of the x axis
plt.ylabel('Percentage Score')#Assign the name of the y axis
plt.show()
```



we can clearly see that there is a positive linear relation between the number of hours studied and percentage of score.

Preparing the Data

```
In [39]: X = df.iloc[:, :-1].values
         y = df.iloc[:, 1].values
```

Using Scikit-Learn's built-in train_test_split() method:

```
In [22]: 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)
```

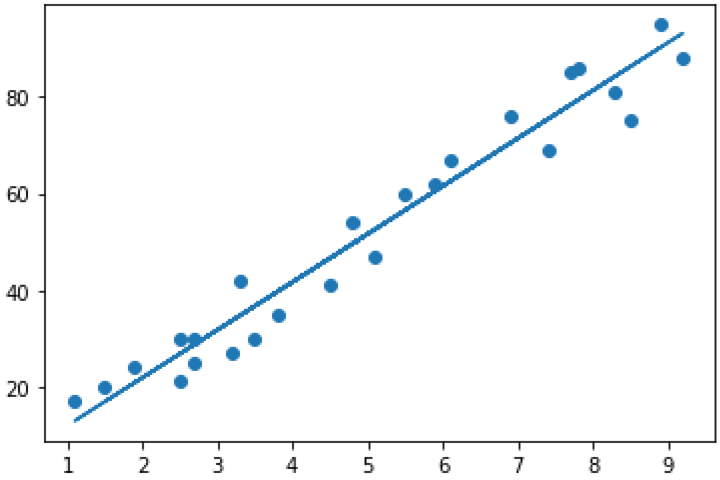
Training the Algorithm

```
In [23]: from sklearn.linear_model import LinearRegression
         regressor = LinearRegression()
         regressor.fit(X_train, y_train)
```

```
Out[23]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

```
In [29]: line=regressor.coef_*X+regressor.intercept_
```

```
In [33]: plt.scatter(X,y)
         plt.plot(X,line);
         plt.show()
```



To retrieve the intercept

In [24]:

```
print(regressor.intercept_)
```

```
2.018160041434683
```

In [25]:

```
print(regressor.coef_)
```

```
[9.91065648]
```

Making Predictions

In [26]:

```
y_pred = regressor.predict(X_test)
```

In [27]:

```
df1 = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
df1
```

Out[27]:

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

In [38]:

```
hours=9.25
test=np.array([hours])
test=test.reshape(-1,1)
pred=regressor.predict(test)
print(" No of Hours={}".format(hours))
print("predicted score={}".format(pred[0]))
```

```
 No of Hours=9.25
predicted score=93.69173248737538
```

Let's find the values for these metrics using our test data

In [28]:

```
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_test, y
_pred)))
```

```
Mean Absolute Error: 4.183859899002975
Mean Squared Error: 21.5987693072174
Root Mean Squared Error: 4.6474476121003665
```

In this article we studied on of the most fundamental machine learning algorithms i.e. linear regression. We implemented both simple linear regression and multiple linear regression with the help of the Scikit-Learn machine learning library.

conclusion

Above final step is to evaluate the performance of algorithm. this step is particularly important to compare how well different algorithm perform on particular Dataset

In [ ]: