Data Science and Business Analytics

#GRIPMAY21

Author: Debapriya Bhowal

Task 1: Prediction using Supervised ML

Predict the percentage of marks that a student is expected to score based upon the number of houres studied.

In [5]:

```
#Importing the libraries required
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

%matplotlib inline
```

In [6]:

```
#Reading data from a remote file
url= "http://bit.ly/w-data"
df= pd.read_csv(url)
print("Data imported successfully.")
df.head(10)
```

```
Data imported successfully.
```

Out[6]:

	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

In [8]:

```
df.shape
```

Out[8]:

```
(25, 2)
```

So here in this dataset we have 25 entries and 2 columns.

In [10]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25 entries, 0 to 24
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Hours   25 non-null     float64
 1   Scores  25 non-null     int64  
dtypes: float64(1), int64(1)
memory usage: 528.0 bytes
```

In [11]:

```
#Check if there are any missing values
df.isnull()
```

Out[11]:

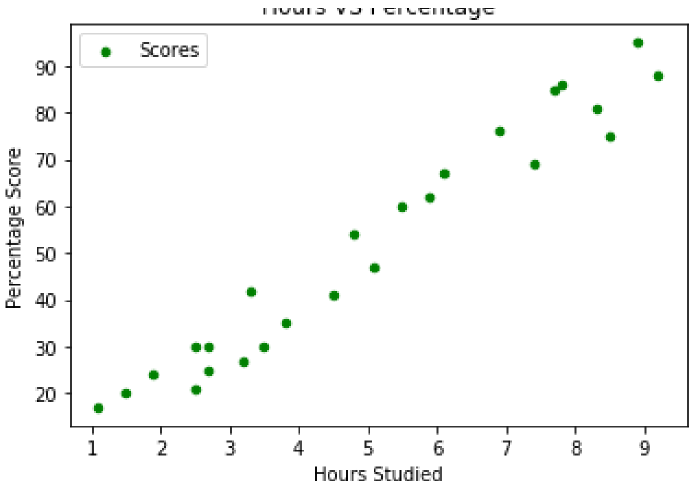
	Hours	Scores
0	False	False
1	False	False
2	False	False
3	False	False
4	False	False
5	False	False
6	False	False
7	False	False
8	False	False
9	False	False
10	False	False
11	False	False
12	False	False
13	False	False
14	False	False
15	False	False
16	False	False
17	False	False
18	False	False
19	False	False
20	False	False
21	False	False
22	False	False
23	False	False
24	False	False

There is no missing value as we can see.

In [12]:

```
#Plotting the distribution of scores
df.plot(x= "Hours", y= "Scores", kind= "scatter", color="green", label="Scores")
plt.title("Hours VS Percentage")
plt.xlabel('Hours Studied')
plt.ylabel('Percentage Score')
plt.show()
```

Hours VS Percentage



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

In [21]:

```
df.corr()
```

Out[21]:

	Hours	Scores
Hours	1.000000	0.976191
Scores	0.976191	1.000000

The next step is to divide the data into "attributes" (inputs) and "labels" (outputs).

In [13]:

```
x_value = df.iloc[:,:-1].values
y_value = df.iloc[:, 1].values
print (x_value, y_value)
```

```
[[2.5]
 [5.1]
 [3.2]
 [8.5]
 [3.5]
 [1.5]
 [9.2]
 [5.5]
 [8.3]
 [2.7]
 [7.7]
 [5.9]
 [4.5]
 [3.3]
 [1.1]
 [8.9]
 [2.5]
 [1.9]
 [6.1]
 [7.4]
 [2.7]
 [4.8]
 [3.8]
 [6.9]
 [7.8]] [21 47 27 75 30 20 88 60 81 25 85 62 41 42 17 95 30 24 67 69 30 54 35 76
 86]
```

The next step is to split this data into training and test sets.

In [17]:

In [17]:

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x_value, y_value, test_size=0.2, ran
dom_state=0)
```

In [18]:

```
#Training the model
from sklearn.linear_model import LinearRegression
r = LinearRegression()
r.fit(x_train, y_train)
print("Training completed.")
```

```
Training completed.
```

In [22]:

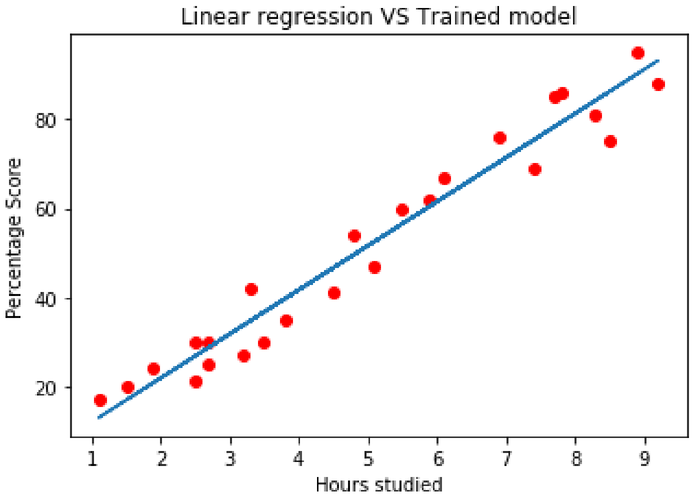
```
print("Intercept value=", r.intercept_)
print("Linear coefficient =", r.coef_)
```

```
Intercept value= 2.018160041434662
Linear coefficient = [9.91065648]
```

In [26]:

```
#Plotting the Regression line
line= r.coef_*x_value + r.intercept_

plt.scatter(x_value, y_value, color= "red")
plt.title("Linear regression VS Trained model")
plt.xlabel("Hours studied")
plt.ylabel("Percentage Score")
plt.plot(x_value,line)
plt.show()
```



Predicting the Model

In [32]:

```
#Testing the data
print(x_test)
```

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

In [33]:

```
#Predicting scores
y_predict= r.predict(x_test)
```

```
y_predict
```

Out[33]:

```
array([16.88414476, 33.73226078, 75.357018  , 26.79480124, 60.49103328])
```

In [31]:

```
# Comparing Actual vs Predicted
df = pd.DataFrame({'Actual': y_test, 'Predicted': y_predict})
```

In [34]:

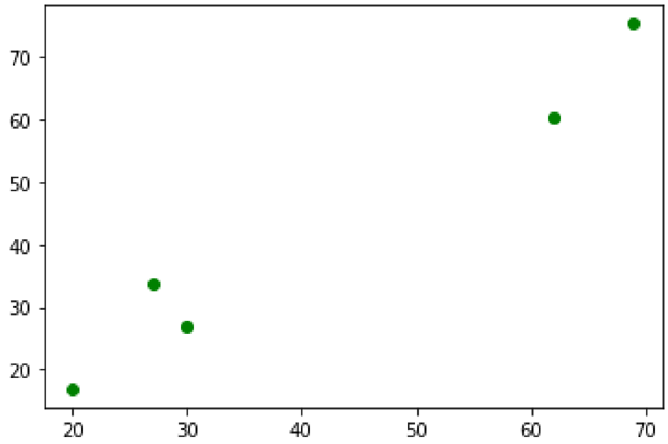
```
df
```

Out[34]:

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

In [36]:

```
plt.scatter(y_test, y_predict, color="green")
plt.show()
```



Predicted score if a student studies 9.5 hours a day?

In [38]:

```
hours = np.array(9.25).reshape(-1,1)
pred_score = r.predict(hours)
print("No of Hours = {}".format(hours[0][0]))
print("Predicted Score = {}".format(pred_score[0]))
```

```
No of Hours = 9.25
Predicted Score = 93.69173248737539
```

Evaluating the model

The final step is to evaluate the performance of algorithm.

In [39]:

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

```
Mean Absolute Error: 4.183859899002982
```

The task has been completed.

In [ ]:

```

```