The Spark Foundation

Data Science and Business Analytics Internship (GRIP June 2021)

TASK 1-Prediction Using Supervised Machine Learning

Author : Aman Alvi

Problem Statement

- Predict the percentage of a student based on the number of study hours
- What will be the predicted score if a student studies for 9.25 hrs?

In [3]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

In [17]:
```
### Loading the dataset

url = 'https://bit.ly/w-data'
data = pd.read_csv(url)
print("Data imported successfully")
```

```
Data imported successfully
```

Exploratory Data Analysis

In [18]:
```
#first five rows of dataframe
data.head()
```

Out[18]:

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

In [6]:
```
# concise summary of dataframe
data.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 [7]:
```
# descriptive statistics of data
data.describe()
```

Out[7]:

	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 [8]:
```
# descriptive statistics of 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

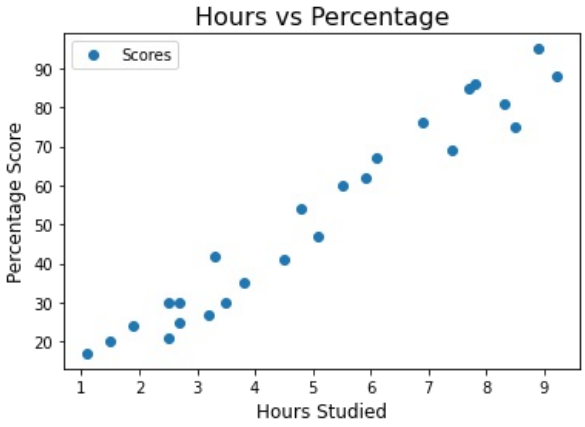
In [9]:
```
# checking for null values
data.isnull().sum()
```

Out[9]:
```
Hours     0
Scores    0
dtype: int64
```

Visualizing Data

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 Plotting the distribution of scores

In [11]:
```
data.plot(x='Hours', y='Scores', style='o')
plt.title('Hours vs Percentage', fontsize=16)
plt.xlabel('Hours Studied',fontsize=12)
plt.ylabel('Percentage Score', fontsize=12)
plt.show()
```



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.

Preparing the data

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

In [19]:
```
X = data.iloc[:, :-1].values
y = data.iloc[:, 1].values
```

Now that we have our attributes and labels, the next step is to split this data into training and test sets. We'll do this by using Scikit-Learn's built-in train_test_split() method:

In [20]:
```
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 Model

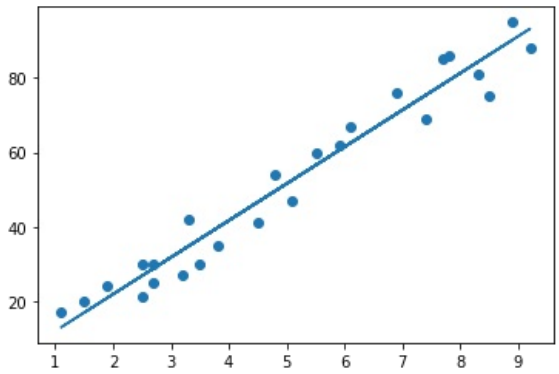
We have split our data into training and testing sets, and now is finally the time to train our algorithm.

In [21]:
```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)

print("Training complete.")
```

```
Training complete.
```

Plotting the regression line

In [22]:
```
# Plotting the regression line
line = regressor.coef_*X+regressor.intercept_

# Plotting for the test data
plt.scatter(X, y)
plt.plot(X, line);
plt.show()
```



Making Predictions

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

In [23]:
```
print(X_test) # Testing data - In Hours
y_pred = regressor.predict(X_test) # Predicting the scores
```

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

In [24]:
```
# Comparing Actual vs Predicted
df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
df
```

Out[24]:

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

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. For simplicity here, we have chosen the mean square error. There are many such metrics

In [27]:
```
from sklearn import metrics
from sklearn.metrics import r2_score
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print ('R2 Score:', r2_score(y_test,y_pred))
```

```
Mean Absolute Error: 4.183859899002975
R2 Score: 0.9454906892105356
```

What will be predicted score if a student studies for 9.25 hrs/ day?

In [28]:
```
hours = 9.25
own_pred = regressor.predict([[hours]])
print("No of Hours = {}".format(hours))
print("Predicted Score = {}".format(own_pred[0]))
```

```
No of Hours = 9.25
Predicted Score = 93.69173248737538
```

Thank you

In [ ]:
```

```