The Sparks Foundation-Data Science & Business Analytics Internship

Task 1 - Prediction using Supervised Machine Learninng

In this task it is required to predict the percentge of a student on the basis of number of hours studied using the Linear Regression supervised machinne learning algorithm.

Simple Linear Regresion

In this Regression 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.

Author: Chupriya Venkatesan

Importing all required libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.model_selection import train_test_split
%matplotlib inline
```

In [3]:
```
#Reading data from remote link
url="http://bit.ly/w-data"
data = pd.read_csv(url)
print(data)
```

```
    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
10    7.7      85
11    5.9      62
12    4.5      41
13    3.3      42
14    1.1      17
15    8.9      95
16    2.5      30
17    1.9      24
18    6.1      67
19    7.4      69
```

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

In [4]:
```
data.head(10)
```

Out[4]:

	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 [5]:
```
# The shape of dataset
data.shape
```

Out[5]:
```
(25, 2)
```

In [6]:
```
#check the info of data
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]:
```
#check the description of student_score data
data.describe()
```

Out[7]:

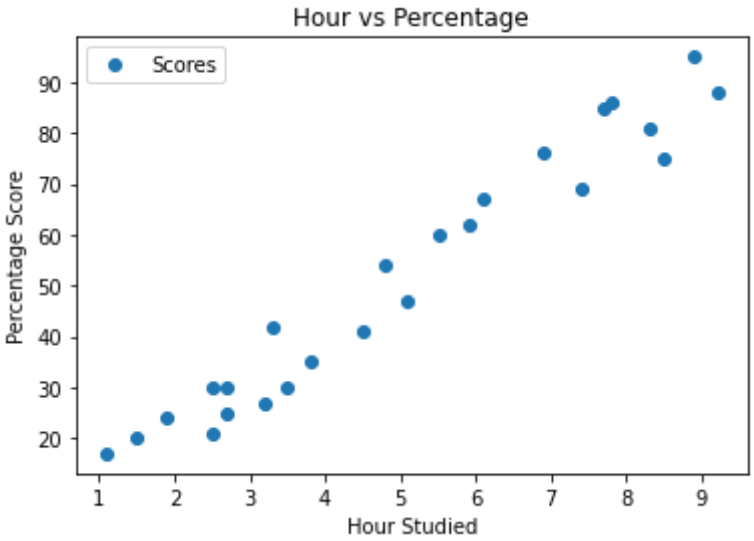
	Hours	Scores
count	25.000000	25.000000
mean	5.012000	51.480000
std	2.525094	25.286887
min	1.100000	17.000000
25%	2.700000	30.000000
50%	4.800000	47.000000

	Hours	Scores
75%	7.400000	75.000000
max	9.200000	95.000000

Data Visualization

Now let's plot a graph of our data so that it will give us clear idea about data

In [9]:
```
#Plotting the distribution of scores
data.plot(x='Hours',y='Scores',style='o')
plt.title('Hour vs Percentage')
plt.xlabel('Hour Studied')
plt.ylabel('Percentage Score')
plt.show()
```



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

Linear Regression Model

Now we prepare the data and split it in test data.

In [12]:
```
X=data.iloc[:,:-1].values
Y=data.iloc[:,1].values
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,train_size=0.80,test_size=0.2)
```

Training the model

In [13]:
```
from sklearn.linear_model import LinearRegression
linearReg=LinearRegression()
linearReg.fit(X_train,Y_train)
Y_predict=linearReg.predict(X_train)
```

Training the Algorithm

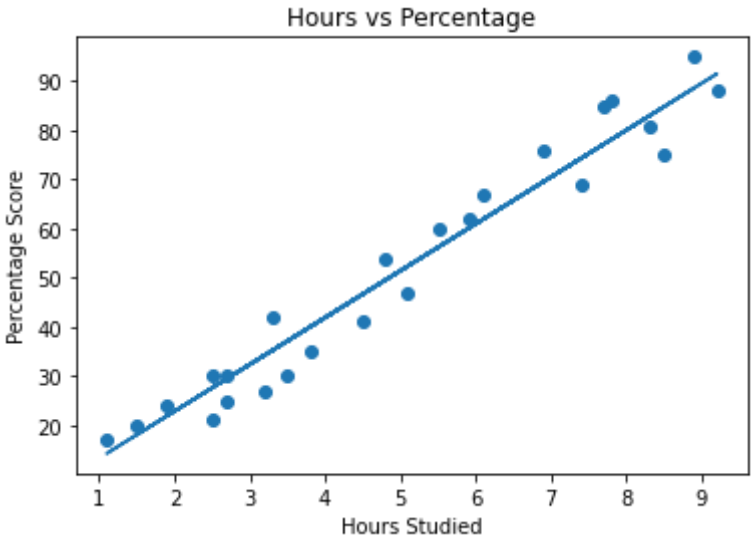
Now the spliting of our data into training and testing set is done, now it's time to train our algorithm.

In [14]:
```
Reg= LinearRegression()
Reg.fit(X_train,Y_train)

print("Training complete")
```

```
Training complete
```

In [15]:
```
# Plotting the regression line
line= Reg.coef_*X+Reg.intercept_
# Plotting for the test data
plt.scatter(X,Y)
plt.plot(X,line);
plt.title('Hours vs Percentage')
plt.xlabel('Hours Studied')
plt.ylabel('Percentage Score')
plt.show()
```



Checking the accuracy scores for training and test set

In [16]:
```
print('Test Score')
print(Reg.score(X_test,Y_test))
print('Train Score')
print(Reg.score(X_train,Y_train))
```

```
Test Score
0.9572617158681459
Train Score
0.9503216112851878
```

In [17]:
```
Y_test
```

Out[17]:
```
array([42, 21, 27, 95, 81], dtype=int64)
```

In [18]:
```
Y_predict
```

Out[18]:
```
array([27.63735323, 74.36727957, 84.8576712 , 69.59891974, 21.91532144,
       52.43282435, 37.17407289, 91.53337496, 77.22829547, 49.57180845,
```

```
       29.54469716, 18.10063357, 40.03508879, 29.54469716, 60.06220008,
       46.71079255, 78.18196744, 61.96954401, 14.28594571, 56.24751222])
```

In [19]:
```
Y_predict[:5]
```

Out[19]:
```
array([27.63735323, 74.36727957, 84.8576712 , 69.59891974, 21.91532144])
```

In [22]:
```
data=pd.DataFrame({'Actual':Y_test,'Predicted':Y_predict[:5]})
data
```

Out[22]:

	Actual	Predicted
0	42	27.637353
1	21	74.367280
2	27	84.857671
3	95	69.598920
4	81	21.915321

In [24]:
```
# Let's predict the score for 9.2 hours
print('Score of student who studied for 9.2 hours a day',Reg.predict([[9.2]]))
```

```
Score of student who studied for 9.2 hours a day [91.53337496]
```

Model Evaluation Metrics

In [25]:
```
# Checking the efficiency of model
mean_squ_error=mean_squared_error(Y_test,Y_predict[:5])
mean_abs_error=mean_absolute_error(Y_test,Y_predict[:5])
print("Mean Squared Error:",mean_squ_error)
print("Men Absolute Error",mean_abs_error)
```

```
Mean Squared Error: 2107.6152773386143
Men Absolute Error 42.01467127181587
```

In [ ]:
```

```