Data Science and Business Analytics

#GRIPMAY21

Author: Antara Das

Task 1: Prediction using Supervised ML

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

In [1]:
```
# Importing all the libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
#Reading data from remote file
url="http://bit.ly/w-data"
df=pd.read_csv(url)
```

In [3]:
```
print("Printing the first 10 data")
df.head(10)
```

```
Printing the first 10 data
```

Out[3]:

	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 [4]:
```
df.shape
```

Out[4]:
```
(25, 2)
```

So there are 25 rows and 2 columns in our dataset.

In [5]:
```
df.dtypes
```

Out[5]:
```
Hours      float64
Scores       int64
dtype: object
```

In [6]:
```
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: 464.0 bytes
```

In [7]:
```
#check whether any missing value is there
df.isnull().sum()
```

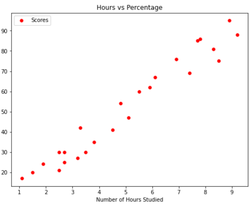
Out[7]:
```
Hours     0
Scores    0
dtype: int64
```

In [8]:
```
#check how Hours and Scores are correlated
df.corr()
```

Out[8]:

	Hours	Scores
Hours	1.000000	0.976191
Scores	0.976191	1.000000

In [9]:
```
#Plotting the distribution of scores
df.plot(kind='scatter',x='Hours',y='Scores',color='Red',s=30,label='Scores',figsize=(8,6))
plt.title('Hours vs Percentage')
plt.xlabel('Number of Hours Studied')
plt.ylabel('Percentage Score')
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.

In [10]:
```
#divide data into attributes and labels
x_data=df[['Hours']].values
y_data=df['Scores'].values
print(x_data,y_data)
```

```
[[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]
```

In [11]:
```
#splitting our data into training and testing sets
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x_data,y_data,test_size=0.20,random_state=0)
```

In [12]:
```
#create linear regression object and fit the linear model
from sklearn.linear_model import LinearRegression
lm=LinearRegression()
lm.fit(x_train,y_train)
print("Training completed.")
```

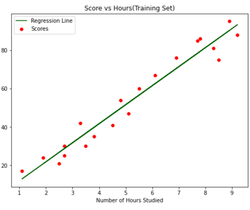
```
Training completed.
```

In [13]:
```
#testing the accuracy of the model
print(lm.score(x_test,y_test))
```

```
0.9454906892105355
```

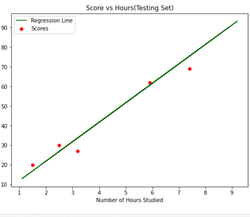
In [14]:
```
#Plotting the training set

plt.figure(figsize=(8,6))
plt.scatter(x_train,y_train,s=30,color='Red',label='Scores')
plt.plot(x_train,lm.predict(x_train),color='DarkGreen',label='Regression Line')
plt.title('Score vs Hours(Training Set)')
plt.xlabel('Number of Hours Studied')
plt.ylabel('Percentage score')
plt.legend()
plt.show()
```



In [15]:
```
#Plotting the testing set

plt.figure(figsize=(8,6))
plt.scatter(x_test,y_test,s=30,color='Red',label='Scores')
plt.plot(x_train,lm.predict(x_train),color='DarkGreen',label='Regression Line')
plt.title('Score vs Hours(Testing Set)')
plt.xlabel('Number of Hours Studied')
plt.ylabel('Percentage score')
plt.legend()
plt.show()
```



In [16]:
```
#Actual value vs Predicted value
y_hat=lm.predict(x_test)
df2=pd.DataFrame({"Actual": y_test ,"Predicted": y_hat})
df2
```

Out[16]:

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

Task is to check what will be the score if a student studies for 9.25 hours per day.

In [17]:
```
hours = np.array(9.25).reshape(-1,1)
score_predict = lm.predict(hours)
print("No of Hours = {}".format(hours[0][0]))
print("Predicted Score = {}".format(score_predict[0]))
```

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

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 absolute error.

In [18]:
```
from sklearn.metrics import mean_absolute_error
mse=mean_absolute_error(y_test,y_hat)
print('Mean absolute error is: {}'.format(mse))
```

```
Mean absolute error is: 4.183859899002982
```

Task is completed.