Data Science and Business Analytics

GRIPMAR21

Author : Shruti Chittora

Task 1 : Prediction using Supervised ML

A simple linear regression task as it involves just 2 variables i.e hours and score

Importing libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

Getting the data

In [4]:
```
url='http://bit.ly/w-data'
data=pd.read_csv(url)
print('Done!')
```

```
Done!
```

In [5]:
```
data.head(5)
```

Out[5]:

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

In [6]:
```
data.dtypes
```

Out[6]:
```
Hours      float64
Scores       int64
dtype: object
```

In [7]:
```
data.shape
```

Out[7]:
```
(25, 2)
```

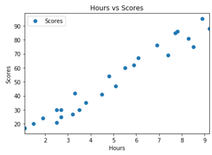
In [8]:
```
data.describe()
```

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

Data Visualization

In [18]:
```
data.plot(x='Hours',y='Scores',style='o')
plt.xlabel('Hours')
plt.ylabel('Scores')
plt.title('Hours vs Scores')
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 [34]:
```
X = data.iloc[:, :-1].values
y = data.iloc[:, 1].values
print(x)
print(y)
```

```
Scores
[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]
```

splitting data into training and test set We will do this by using Scikit-Learn's built-in train_test_split() method:

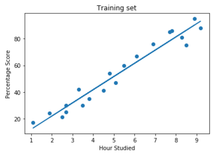
In [21]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
                            test_size=0.2, random_state=0)
```

Training the ML Algorithm for this we requires a linear regression Algorithm

In [22]:
```
from sklearn.linear_model import LinearRegression
reg=LinearRegression()
reg.fit(X_train,y_train)
```

Out[22]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

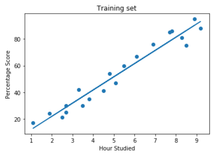
In [23]:
```
# Plotting for the test data
plt.scatter(X_train,y_train)
plt.title('Training set')
plt.plot(X_train,reg.predict(X_train))
plt.xlabel('Hour Studied')
plt.ylabel('Percentage Score')
plt.show()
```



In [24]:
```
#acuracy of training
reg.score(X_train,y_train)
```

Out[24]:
```
0.9515510725211553
```

In [25]:
```
# Plotting the regression line
line = reg.coef_*X+reg.intercept_
# Plotting for the test data
plt.scatter(X_train,y_train)
plt.title('Training set')
plt.plot(X_train,reg.predict(X_train))
plt.xlabel('Hour Studied')
plt.ylabel('Percentage Score')
plt.show()
```



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

In [26]:
```
print(X_test) # Testing data - In Hours
y_pred = reg.predict(X_test) # Predicting the scores
```

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

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

In [27]:
```
hours=9.25
own_pred=reg.predict([[hours]])
print("No of Hours = {}".format(hours))
print("Predicted Score = {}".format((own_pred)[0]))
```

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

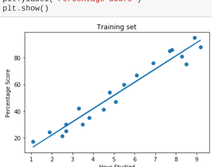
Evaluating the model

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 square error. There are many such metrics.

In [28]:
```
from sklearn import metrics
print('Mean Absolute Error:',
      metrics.mean_absolute_error(y_test, y_pred))
```

```
Mean Absolute Error: 4.183859899002975
```

In [29]:
```
# Visualizing the training set
plt.scatter(X_train,y_train)
plt.title('Training set')
plt.plot(X_train,reg.predict(X_train))
plt.xlabel('Hour Studied')
plt.ylabel('Percentage Score')
plt.show()
```



Hence Task1 is completed!

In [ ]:
```
```