GRIP JANUARY''22 THE SPARKS FOUNDATION

DATA SCIENCE AND BUSSINESS ANALYTICS INTERN

Name -- Aditya kumar

Task 1 Prediction using supervised machine learning

Linear Regression with Python Scikit Learn

In this section we will see how the Python Scikit-Learn library for machine learning can be used to implement regression functions. We will start with simple linear regression involving two variables.

Simple Linear Regression

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.

Description :: To predict the percentage of an student based on the no of study hours.

IMPORTING ALL THE REQUIRED LIBRARY

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

IMPORT DATASET

In [2]:
```
url = "http://bit.ly/w-data"
s_data = pd.read_csv(url)
print("Data imported successfully")

s_data.head(10)
```

```
Data imported successfully
```

Out[2]:

	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 [3]:
```
s_data.shape
```

Out[3]:
```
(25, 2)
```

In [4]:
```
s_data.columns
```

Out[4]:
```
Index(['Hours', 'Scores'], dtype='object')
```

In [5]:
```
s_data.describe()
```

Out[5]:

	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

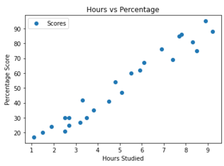
CHECKING FOR NULL VALUES

In [6]:
```
s_data.isnull().sum()
```

Out[6]:
```
Hours     0
Scores    0
dtype: int64
```

SCORE DISTRIBUTION PLOT

In [7]:
```
s_data.plot(x='Hours', y='Scores', style='o')
plt.title('Hours vs Percentage')
plt.xlabel('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.

Preparing the data

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

In [8]:
```
X = s_data.iloc[:, :-1].values
y = s_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 [9]:
```
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 Algorithm

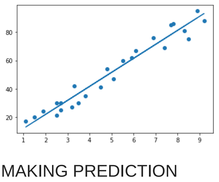
In [10]:
```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

Out[10]:
```
LinearRegression()
```

PLOTTING THE REGRESSION LINE

In [11]:
```
line = regressor.coef_*X+regressor.intercept_

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



MAKING PREDICTION

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

In [12]:
```
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]]
```

COMPARISON BETWEEN ACTUAL AND PREDICTED

In [13]:
```
df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
df
```

Out[13]:

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

Predicting score of student based on based on hour studied

Here our aim is to predict the score of student if he/she studies for 9.25 hours

In [14]:
```
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
```

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 [15]:
```
from sklearn import metrics
print('Mean Absolute Error:',
      metrics.mean_absolute_error(y_test, y_pred))
```

```
Mean Absolute Error: 4.183859899002975
```