TSF-TASK 2: Score Prediction using Linear Regression

To Explore Supervised Machine Learning in this Regression problem.

Objective

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.

Loading Libraries

In [25]:
```
#Importing all libraries
import pandas as pd
import requests
import io
import numpy as np                          # for matrix operations
import matplotlib.pyplot as plt             # for visualization
%matplotlib inline
import seaborn as sns
```

Loading Data

In [26]:
```
data=pd.read_csv('http://bit.ly/w-data')
print("Data imported successfully")
data.head()
```

```
Data imported successfully
```

Out[26]:

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

In [27]:
```
data.shape
```

Out[27]:
```
(25, 2)
```

In [28]:
```
data.describe()
```

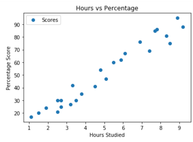
Out[28]:

	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

Visualization

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:

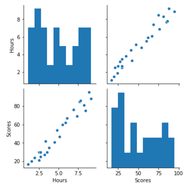
In [29]:
```
# Plotting the distribution of scores
data.plot(x='Hours', y='Scores', style='o')
plt.title('Hours vs Percentage')
plt.xlabel('Hours Studied')
plt.ylabel('Percentage Score')
plt.legend()
plt.show()
```



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.

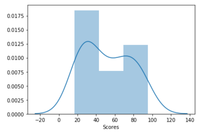
In [30]:
```
sns.pairplot(data)
```

Out[30]:
```
<seaborn.axisgrid.PairGrid at 0x1aa0734550>
```



In [31]:
```
sns.distplot(data['Scores'])
```

Out[31]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1aa0760fa88>
```



Data Preprocessing

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

In [12]:
```
X = data.iloc[:, :-1].values
y = data.iloc[:, 1].values
```

Splitting the Data sets in Training and Test using Scikit-Learn's

In [13]:
```
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

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

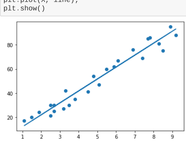
In [14]:
```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)

print("Training complete.")
```

```
Training complete.
```

In [17]:
```
# 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 [19]:
```
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]]
```

Performance Evaluating

We have predicted the outputs on our test dataset.Now is the time to evaluate the performance of our model using r2_score, mean_squared_error and mean_absolute_error.we will import these metrices from scikit_learn library

In [37]:
```
from sklearn.metrics import r2_score, mean_squared_error,mean_absolute_error
score =r2_score(y_pred,y_test)
mse =mean_squared_error(y_pred,y_test)
mae =mean_absolute_error(y_pred,y_test)


print('r2_score:{}'.format(score))
print ('mean_squared_error:{}'.format(mse))
print('Mean Absolute Error:', mae)
```

```
r2_score:0.9546785947197246
mean_squared_error:21.598769307217 4
Mean Absolute Error: 4.183859899002975
```

In [41]:
```
# Comparing Actual vs Predicted
df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
df
```

Out[41]:

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

In [44]:
```
df.plot()
```

Out[44]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1aa07830548>
```

Predicting Score

Now we will be predicting the scores with respect to 9.25hours of study using our trained model.

In [47]:
```
hour=[[9.25]]
predict_score=regressor.predict(hour)
print('hour_studied:{}'.format(hour))
print('Score_prediction:{}'.format(predict_score))
```

```
hour_studied:[[9.25]]
Score_prediction:[93.69173249]
```