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 and then we will move towards linear regression involving multiple 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.

Importing Libraries

In [14]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

Importing dataset

In [15]:

```
url = r'https://raw.githubusercontent.com/AdiPersonalWorks/Random/master/student_scores%20-%
20student_scores.csv'
df = pd.read_csv(url)
df.head()
```

Out[15]:

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

Data Analysis

In [16]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25 entries, 0 to 24
Data columns (total 2 columns):
Hours     25 non-null float64
Scores    25 non-null int64
dtypes: float64(1), int64(1)
memory usage: 480.0 bytes
```

In [17]:

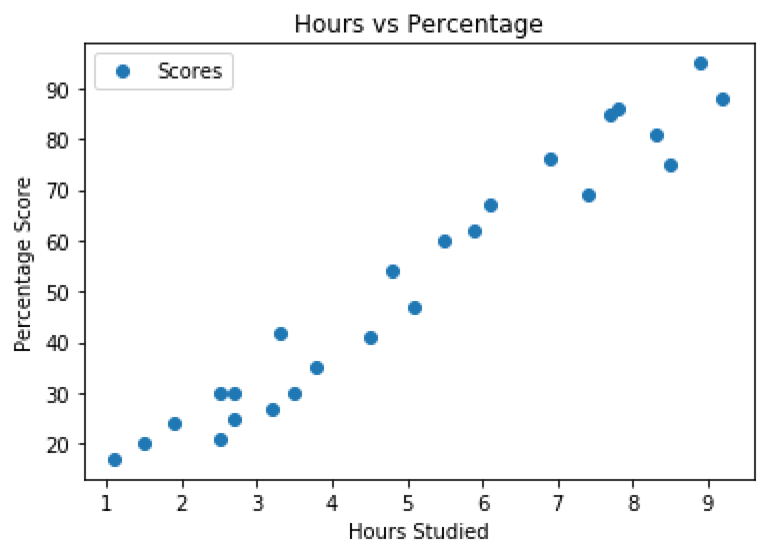
```
df.describe()
```

Out[17]:

	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

In [18]:

```
df.plot(x='Hours', y='Scores', style='o')
plt.title('Hours vs Percentage')
plt.xlabel('Hours Studied')
plt.ylabel('Percentage Score')
plt.show()
```



Preparing the data

In [19]:

```
X = df.iloc[:, :-1].values
y = df.iloc[:, 1].values
```

In [20]:

```
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 [21]:

```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

Out[21]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

In [22]:

```
#To retrieve the intercept:
print(regressor.intercept_)
```

```
2.018160041434683
```

In [23]:

```
#For retrieving the slope (coefficient of x):
print(regressor.coef_)
```

```
[9.91065648]
```

This means that for every one unit of change in hours studied, the change in the score is about 9.91%. Or in simpler words, if a student studies one hour more than they previously studied for an exam, they can expect to achieve an increase of 9.91% in the score achieved by the student previously.

Making Predictions

In [24]:

```
y_pred = regressor.predict(X_test)
```

The y_pred is a numpy array that contains all the predicted values for the input values in the X_test series.

To compare the actual output values for X_test with the predicted values, execute the following script:

In [25]:

```
dif = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
dif
```

Out[25]:

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

Though my model is not very precise, the predicted percentages are close to the actual ones.

Note : The values in the columns above may be different in your case because the **train_test_split** function randomly splits data into train and test sets, and your splits are likely different from the one shown here.

Evaluating the Algorithm

In [26]:

```
from sklearn import metrics
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

```
Mean Absolute Error: 4.183859899002975
Mean Squared Error: 21.5987693072174
Root Mean Squared Error: 4.6474476121003665
```

You can see that the value of root mean squared error is 4.64, which is less than 10% of the mean value of the percentages of all the students i.e. 51.48. This means that my algorithm did a decent job.

Making Prediction

What will be predicted score if a student study for 9.25 hrs in a day?

In [27]:

```
regressor.predict(np.array([[9.25]]))
```

Out[27]:

```
array([93.69173249])
```

Conclusion