xxxxxxxxxx

```
## **Task 1 - Prediction using Supervised ML**

### **Submitted by - Rishika Rai**

### **Simple Linear Regression**
```

Task 1 - Prediction using Supervised ML¶

Submitted by - Rishika Rai¶

Simple Linear Regression¶

xxxxxxxxxx

```
### Importing the Required Libraries
```

Importing the Required Libraries¶

In [1]:

xxxxxxxxxx

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib
%matplotlib inline
from matplotlib import pyplot as plt
```

xxxxxxxxxx

```
## Read the Data
```

Read the Data¶

In [2]:

xxxxxxxxxx

```
DataFrame = pd.read_csv("http://bit.ly/w-data")
```

In [3]:

xxxxxxxxxx

```
DataFrame.head()
```

Out[3]:

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

In [4]:

xxxxxxxxxx

```
DataFrame.tail()
```

Out[4]:

	Hours	Scores
20	2.7	30
21	4.8	54
22	3.8	35
23	6.9	76
24	7.8	86

In [5]:

xxxxxxxxxx

```
DataFrame.shape
```

Out[5]:

```
(25, 2)
```

xxxxxxxxxx

```
## Lets check the data type
```

Lets check the data type¶

In [6]:

xxxxxxxxxx

```
DataFrame.dtypes
```

Out[6]:

```
Hours     float64
Scores      int64
dtype: object
```

In [7]:

xxxxxxxxxx

```
DataFrame.count()
```

Out[7]:

```
Hours     25
Scores    25
dtype: int64
```

xxxxxxxxxx

```
## Let's check the summary of data
```

Let's check the summary of data¶

In [8]:

xxxxxxxxxx

```
DataFrame.info()

# summary of data
```

```
<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: 528.0 bytes
```

In [9]:

xxxxxxxxxx

```
DataFrame.describe().T
```

Out[9]:

	count	mean	std	min	25%	50%	75%	max
Hours	25.0	5.012	2.525094	1.1	2.7	4.8	7.4	9.2
Scores	25.0	51.480	25.286887	17.0	30.0	47.0	75.0	95.0

xxxxxxxxxx

```
## Total Unique Value
```

Total Unique Value¶

In [10]:

xxxxxxxxxx

```
DataFrame.nunique()
```

Out[10]:

```
Hours     23
Scores    23
dtype: int64
```

xxxxxxxxxx

```
## Missing Value
```

Missing Value¶

In [11]:

xxxxxxxxxx

```
DataFrame.isnull().sum()
```

Out[11]:

```
Hours     0
Scores    0
dtype: int64
```

xxxxxxxxxx

```
As we see, there is no missing value in the dataset.
```

As we see, there is no missing value in the dataset.

xxxxxxxxxx

```
## Data Visualizing
```

Data Visualizing¶

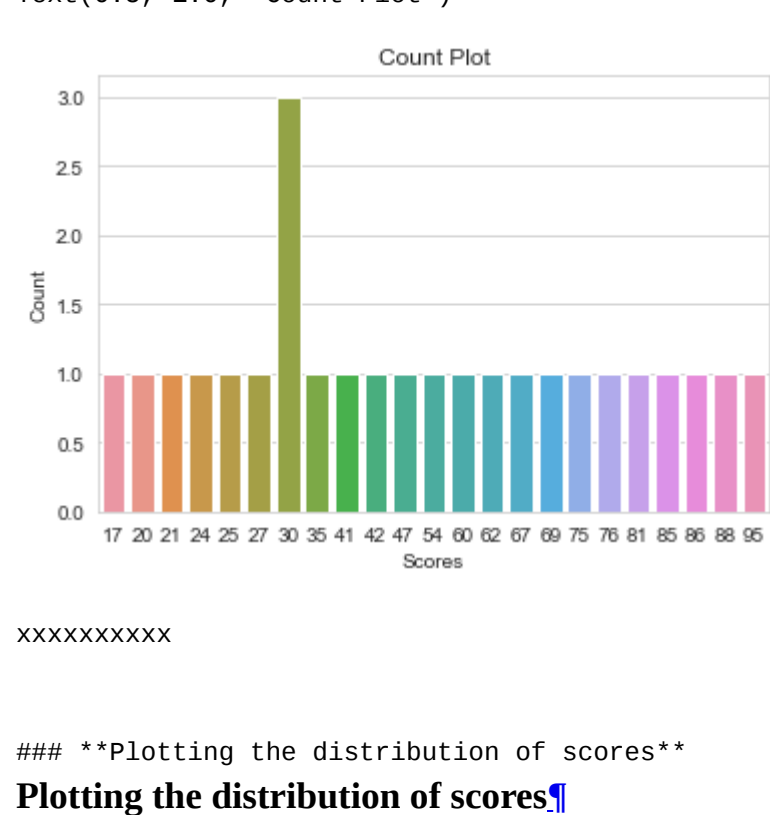
In [12]:

xxxxxxxxxx

```
sns.set_style("whitegrid")
sns.countplot(x = 'Hours', data = DataFrame)
plt.xlabel('Hours')
plt.ylabel('Count')
plt.title('Count Plot')
```

Out[12]:

```
Text(0.5, 1.0, 'Count Plot')
```



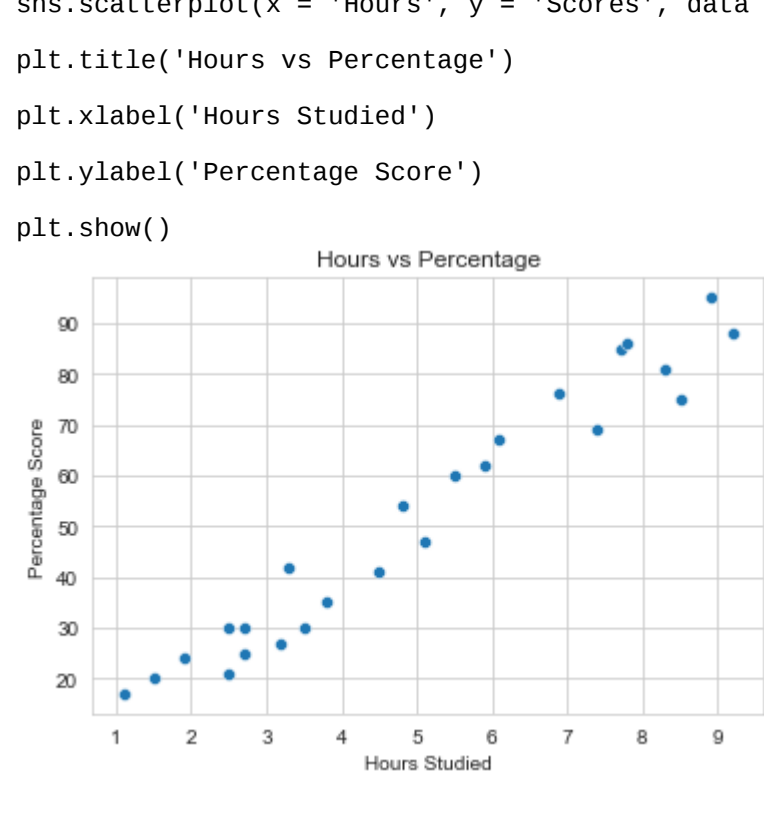
In [13]:

xxxxxxxxxx

```
sns.set_style("whitegrid")
sns.countplot(x = 'Scores', data = DataFrame)
plt.xlabel('Scores')
plt.ylabel('Count')
plt.title('Count Plot')
```

Out[13]:

```
Text(0.5, 1.0, 'Count Plot')
```



xxxxxxxxxx

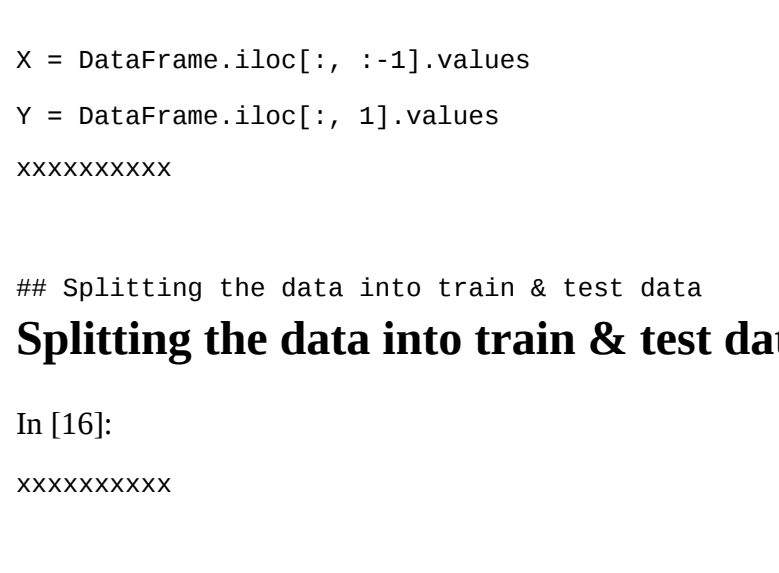
```
### **Plotting the distribution of scores**
```

Plotting the distribution of scores¶

In [14]:

xxxxxxxxxx

```
sns.scatterplot(x = 'Hours', y = 'Scores', data = DataFrame)
plt.title('Hours vs Percentage')
plt.xlabel('Hours Studied')
plt.ylabel('Percentage Score')
plt.show()
```



xxxxxxxxxx

```
 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.
```

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.

xxxxxxxxxx

```
### **Preparing the data**
```

Preparing the data¶

In [15]:

xxxxxxxxxx

```
X = DataFrame.iloc[:, :-1].values
Y = DataFrame.iloc[:, 1].values
```

xxxxxxxxxx

```
## Splitting the data into train & test data
```

Splitting the data into train & test data¶

In [16]:

xxxxxxxxxx

```
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)
```

xxxxxxxxxx

```
## Perform Linear Regresssion
```

Perform Linear Regresssion¶

In [17]:

xxxxxxxxxx

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

print("Training complete.")
```

```
Training complete.
```

xxxxxxxxxx

```
### Model Evaluation
```

Model Evaluation¶

In [18]:

xxxxxxxxxx

```
slope = regressor.coef_
slope
```

Out[18]:

```
array([9.91065648])
```

In [19]:

xxxxxxxxxx

```
intercept = regressor.intercept_
intercept
```

Out[19]:

```
2.018160041434683
```

xxxxxxxxxx

```
###  Plotting the Regression Line
```

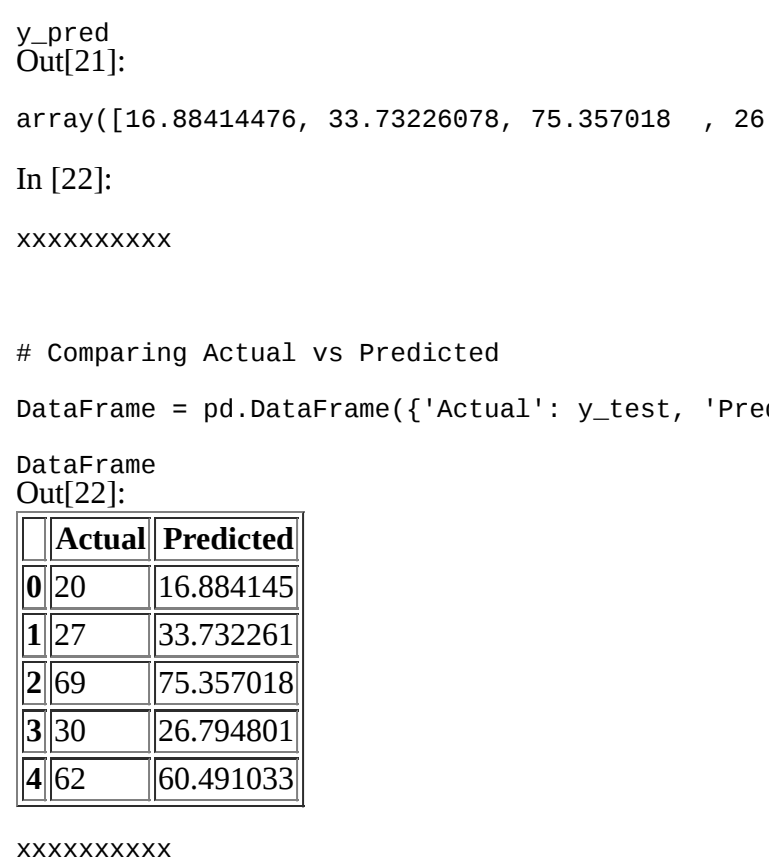
Plotting the Regression Line¶

In [20]:

xxxxxxxxxx

```
line = slope*X+intercept

plt.scatter(X, Y,color = "#5CC8A0" ,label = "scatter plot")
plt.plot(X, line,color = "#080808" ,label = "Regression Line");
plt.legend()
plt.show()
```



xxxxxxxxxx

```
## Making Predictions for Scores
```

Making Predictions for Scores¶

In [21]:

xxxxxxxxxx

```
y_pred = regressor.predict(x_test)

y_pred
```

Out[21]:

```
array([16.88414476, 33.73226078, 75.357018  , 26.79480124, 60.49103328])
```

In [22]:

xxxxxxxxxx

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

Out[22]:

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

xxxxxxxxxx

```
## Predicting Scores at 9.25 hours
```

Predicting Scores at 9.25 hours¶

In [23]:

xxxxxxxxxx

```
pred = regressor.predict([[9.25]])
print("No of Hours = 9.25")
print("Predicted Score = {}".format(pred[0]))
```

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

xxxxxxxxxx

```
## Evaluating the model
```

Evaluating the model¶

In [24]:

xxxxxxxxxx

```
from sklearn import metrics
print('Mean Absolute Error:',

      metrics.mean_absolute_error(y_test, y_pred))
```

```
Mean Absolute Error: 4.183859899002975
```