Supervised Machine Learning-

Supervised learning, also known as supervised machine learning, is a subcategory of machine learning and artificial intelligence. It is defined by its use of labeled datasets to train algorithms that to classify data or predict outcomes accurately

Different Types of Supervised Learning- Regression. In regression, a single output value is produced using training data. ... Classification. It involves grouping the data into classes. ... Naive Bayesian Model. ... Random Forest Model. ... Neural Networks. ... Support Vector Machines.

Linear Regression with Python Scikit Learn

In this section we will see how the Python Scikit-Learn library for ML can be used to implement Regression Function. 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 students is expected to score based upon the no of Hours they studied. This is a simple linear Regression task as it involves just Two Variables.

Importing Library and Data

In [1]:

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

In [2]:

```
data=pd.read_csv('C:/Users/LENOVO/Desktop/SparkInternship/Prediction Using Supervised ML/SupervisedML.csv')
```

In [3]:

```
data.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

Feature Engineering

In [4]:

```
data.dtypes
```

Out[4]:

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

In [6]:

```
data.isnull().sum()
```

Out[6]:

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

In [8]:

```
data.isna().sum()
```

Out[8]:

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

In [10]:

```
data.shape
```

Out[10]:

```
(25, 2)
```

In [12]:

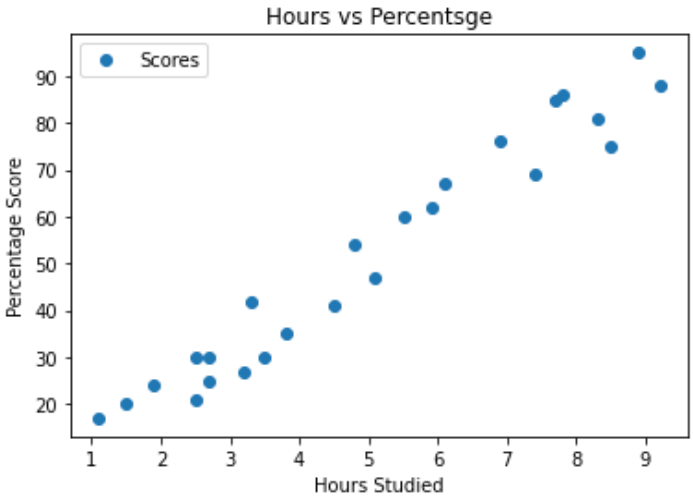
```
data.columns
```

Out[12]:

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

In [16]:

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



Create Trainning Data and Testing Data

In [17]:

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

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)
```

Import Linear Regression Model

In [24]:

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

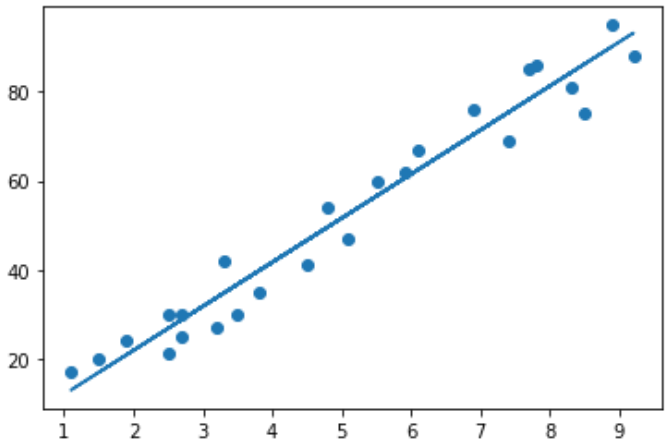
Out[24]:

```
LinearRegression()
```

In [25]:

```
line=regressor.coef_*X+regressor.intercept_

plt.scatter(X,Y)
plt.plot(X,line);
plt.show()
```



In [26]:

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

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

In [28]:

```
df=pd.DataFrame({'Actual':Y_test,'Predicted':y_pred})
df
```

Out[28]:

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

	Actual	Predicted
4	62	60.491033

In [29]:

```
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
```

Now, its time to evaluate our model using mean squared error as well as using mean absolute error

In [31]:

```
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))
```

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

From This We can Say that if student studied for 9.25hrs/day the predicted score will be 93.69....!

In [ ]:

```

```