GRIPJULY21-(THE SPARK FOUNDATION)

Author : Mohammed Asif ¶

Task 1-Prediction using Supervised ML(Level - Beginner)

In this task,will predict the parcentage of an student based on the no of study hours with the help of simple linear regression task and atleast two variable should be use in this task.

In [16]:
```
#Importing require libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
```

step 1 :- Extracting the data from online source

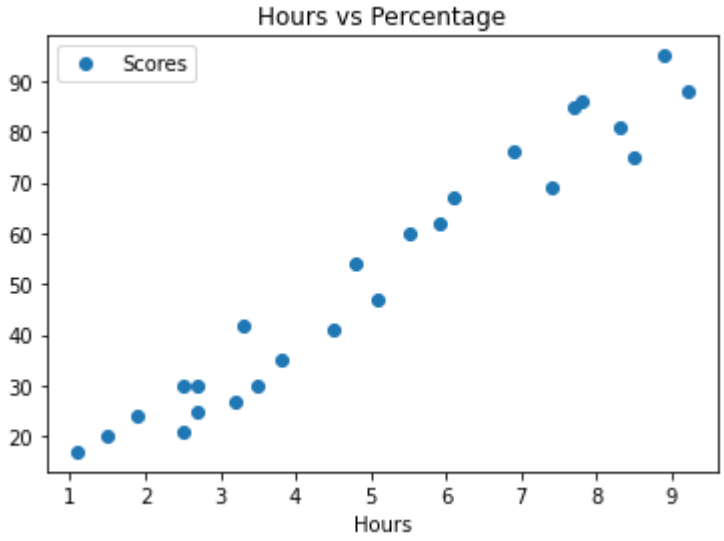
In [17]:
```
#Loading the data from link to read
data = pd.read_csv('http://bit.ly/w-data')
data.head(15)
```

Out[17]:

	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
10	7.7	85
11	5.9	62
12	4.5	41
13	3.3	42
14	1.1	17

step-2 : Data Visualization

In [19]:
```
 #Plotting the distribution of scores
data.plot(x='Hours', y='Scores', style='o')
plt.title('Hours vs Percentage')
plt.show()
```



Step 3:- Preparing The Data

The next step is to separate the data into "attributes" and "labels"

In [20]:
```
x = data.iloc[:,:-1].values
y = data.iloc[:, -1].values
```

Step -4 :- Algorithm Training

Separating the data into training data-set and test data_set. Start training the algorithm

In [21]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_s
regressor = LinearRegression()
regressor.fit(x_train.reshape(-1,1), y_train)
```

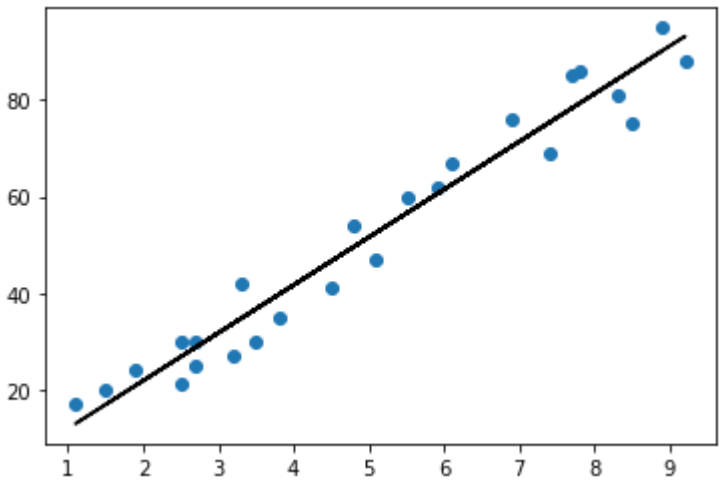
Out[21]:
```
LinearRegression()
```

In [22]:
```
print("Training Over!!")
```

```
Training Over!!
```

Step 5 :- Ploting the line of regression

In [23]:
```
line = regressor.coef_*x+regressor.intercept_
#test data plotting
plt.scatter(x,y)
plt.plot(x,line,color = 'Black')
plt.show()
```



Step 6:- Marking Predictions

We have trained the algorithm, time to make some predictions

In [24]:
```
#Testing data - In Hours
print(x_test)

# predicting the score

y_pred = regressor.predict(x_test)
```

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

In [25]:
```
import pandas as pd
```

Step-7 :- Comparing Actual vs predicted

In [28]:
```
# Comparing actual data vs predicted
data = pd.DataFrame({'Actual': y_test,'Predicted': y_pred})
```

In [29]:
```
data
```

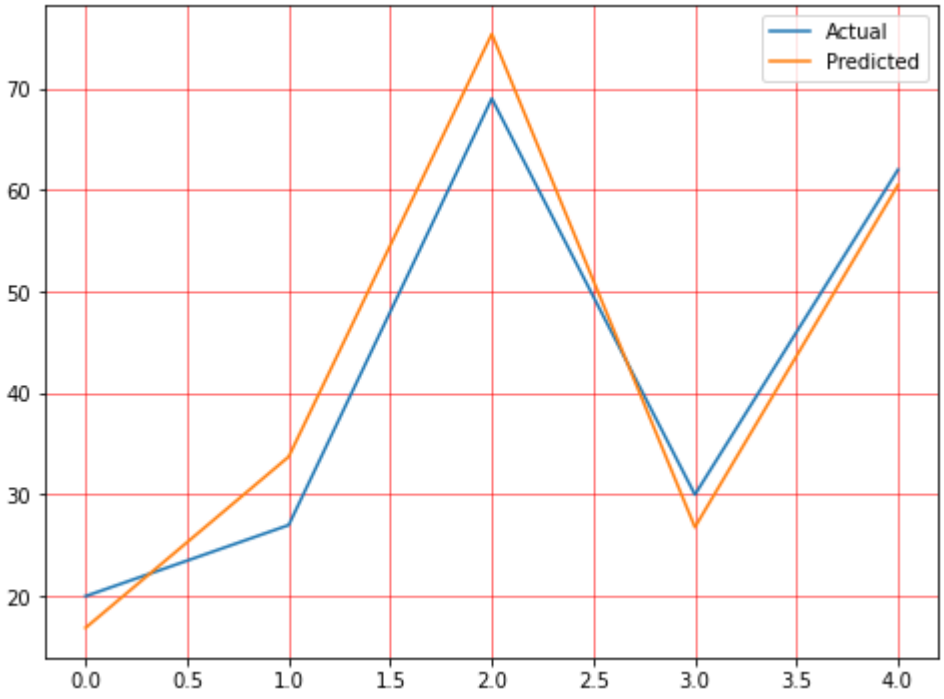
Out[29]:

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

In [30]:
```
#Evaluate the training data and test data score
print("Training score",regressor.score(x_train, y_train))
print("Testing score", regressor.score(x_test, y_test))
```

```
Training score 0.9515510725211552
Testing score 0.9454906892105356
```

In [32]:
```
#create a line graph to depict the difference between the actual and predicted va
data.plot(kind = 'line', figsize = (8,6))
plt.grid(which ='major', linewidth='0.5', color = 'black')
plt.grid(which ='major', linewidth='0.5', color = 'red')
plt.show()
```



In [36]:
```
#Testing the our data
hours = 9.25
test = np.array([hours])
test = test.reshape(-1,1)
own_pred = regressor.predict(test)
print("No of hours={}".format(hours))
print("Predicted Score = {}".format(own_pred[0]))
```

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

Step -8 Evaluating the model

last step to evaluate the performance of algorithm. it is a crucial step to compare how good different algorithm work on a perticular data, We have chosen the mean square error, There are more matrices like that.

In [42]:
```
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_pr
```

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

In [ ]:
```

```