Task-7

GO_STP_5247

Simple Linear Regression

Find the students scores based on their study hours. This is a simple Regression problem type because it has only two variables.

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import sklearn
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score,mean_squared_error
stu= pd.read_csv('/StudentHoursScores.csv')
```

[4]
```
stu.shape
```

```
(23, 2)
```

[5]
```
stu.describe()
```

	Hours	Scores
count	23.000000	23.000000
mean	4.817391	47.695652
std	2.709688	27.103228

✓ 0s completed at 10:50 PM

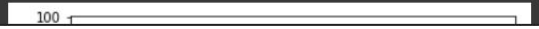
[5]

std	2.709688	27.103228
min	1.100000	12.000000
25%	2.650000	27.000000
50%	4.100000	40.000000
75%	7.100000	72.500000
max	9.600000	96.000000

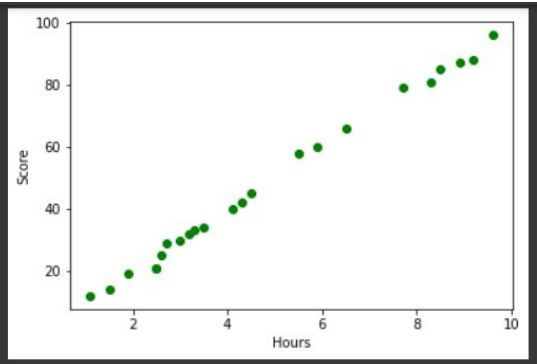
[6]
```
stu.head()
```

	Hours	Scores
0	7.7	79
1	5.9	60
2	4.5	45
3	3.3	33
4	1.1	12

[7]
```
X=stu['Hours']
Y=stu['Scores']

plt.xlabel("Hours")
plt.ylabel("Score")
plt.scatter(X,Y,color = 'green',marker='o')
plt.plot()
plt.show()
```



[7]



[8]
```
X=stu.iloc[:,:-1].values
Y=stu.iloc[:,1].values
print('X: ',X)
print('Y: ',Y)
```

```
X:  [[7.7]
 [5.9]
 [4.5]
 [3.3]
 [1.1]
 [8.9]
 [2.5]
 [1.9]
 [2.7]
 [8.3]
 [5.5]
 [9.2]
 [1.5]
 [3.5]
 [8.5]
 [3.2]
```

```
 [8.5]
 [3.2]
 [6.5]
 [2.5]
 [9.6]
 [4.3]
 [4.1]
 [3. ]
 [2.6]]
Y:  [79 60 45 33 12 87 21 19 29 81 58 88 14 34 85 32 66 21 96 42 40 30 25]
```

```
[10] X_train, X_test, Y_train, Y_test = sklearn.model_selection.train_test_split(X, Y, test_size = 0.33, random_state = 42)
```

```
[11] lin = LinearRegression()

     mod=lin.fit(X_train, Y_train)

     Y_train_pred = lin.predict(X_train)

     Y_test_pred = lin.predict(X_test)
```

```
[12] mod.coef_

     array([9.90147351])
```

```
[13] mod.intercept_

     -0.0374166764121
```

```
[14] mod.predict([[5]])

     array([49.46995088])
```

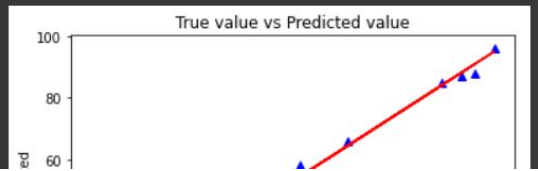
```
df=pd.DataFrame(Y_test_pred,Y_test)

print(df.head())
```

```
            0
32  31.647299
81  82.144813
79  76.203929
29  26.696562
21  24.716267
```

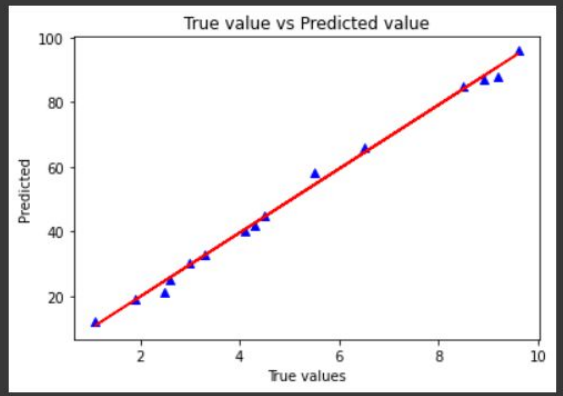
```
[16] print("Accuracy is: ", r2_score(Y_test,Y_test_pred))
     print("MSE: ",mean_squared_error(Y_test,Y_test_pred))
```

```
Accuracy is:  0.9932521891609714
MSE:  4.0044039947860615
```

```
[18] plt.scatter(X_train ,Y_train,c='blue',marker='^',label='Training data')
     plt.plot(X_train,mod.predict(X_train),c='r')

     plt.xlabel('True values')
     plt.ylabel('Predicted')

     plt.title("True value vs Predicted value")

     plt.plot()
     plt.show()
```



```
[18] plt.ylabel('Predicted')

     plt.title("True value vs Predicted value")

     plt.plot()
     plt.show()
```




```
[19] plt.scatter(X_test ,Y_test,c='blue',marker='o',label='Training data')
     plt.plot(X_test,mod.predict(X_test),c='r')
     plt.scatter(X_test ,Y_test_pred,c='lightgreen',marker='s',label='Training data')

     plt.xlabel('True values')
     plt.ylabel('Predicted')
     plt.title("True value vs Predicted value")

     plt.legend()
     plt.plot()
     plt.show()
```

```
[19] plt.scatter(X_test ,Y_test,c='blue',marker='o',label='Training data')
     plt.plot(X_test,mod.predict(X_test),c='r')
     plt.scatter(X_test ,Y_test_pred,c='lightgreen',marker='s',label='Training data')

     plt.xlabel('True values')
     plt.ylabel('Predicted')
     plt.title("True value vs Predicted value")

     plt.legend()
     plt.plot()
     plt.show()
```