```
import numpy as np
import pandas as pd
df=pd.read_csv('https://raw.githubusercontent.com/arib168/data/main/50_Startups.csv')  #data taken from git
df
```

	R&D Spend	Administration	Marketing Spend	State	Profit
0	165349.20	136897.80	471784.10	New York	192261.83
1	162597.70	151377.59	443898.53	California	191792.06
2	153441.51	101145.55	407934.54	Florida	191050.39
3	144372.41	118671.85	383199.62	New York	182901.99
4	142107.34	91391.77	366168.42	Florida	166187.94
5	131876.90	99814.71	362861.36	New York	156991.12
6	134615.46	147198.87	127716.82	California	156122.51
7	130298.13	145530.06	323876.68	Florida	155752.60
8	120542.52	148718.95	311613.29	New York	152211.77
9	123334.88	108679.17	304981.62	California	149759.96
10	101913.08	110594.11	229160.95	Florida	146121.95
11	100671.96	91790.61	249744.55	California	144259.40
12	93863.75	127320.38	249839.44	Florida	141585.52
13	91992.39	135495.07	252664.93	California	134307.35
14	119943.24	156547.42	256512.92	Florida	132602.65
15	114523.61	122616.84	261776.23	New York	129917.04
16	78013.11	121597.55	264346.06	California	126992.93
17	94657.16	145077.58	282574.31	New York	125370.37
18	91749.16	114175.79	294919.57	Florida	124266.90
19	86419.70	153514.11	0.00	New York	122776.86

21	78389.47	153773.43	299737.29	New York	111313.02
22	73994.56	122782.75	303319.26	Florida	110352.25
23	67532.53	105751.03	304768.73	Florida	108733.99
24	77044.01	99281.34	140574.81	New York	108552.04
25	64664.71	139553.16	137962.62	California	107404.34
26	75328.87	144135.98	134050.07	Florida	105733.54
27	72107.60	127864.55	353183.81	New York	105008.31
28	66051.52	182645.56	118148.20	Florida	103282.38
29	65605.48	153032.06	107138.38	New York	101004.64
30	61994.48	115641.28	91131.24	Florida	99937.59
31	61136.38	152701.92	88218.23	New York	97483.56
32	63408.86	129219.61	46085.25	California	97427.84
33	55493.95	103057.49	214634.81	Florida	96778.92
34	46426.07	157693.92	210797.67	California	96712.80
35	46014.02	85047.44	205517.64	New York	96479.51
36	28663.76	127056.21	201126.82	Florida	90708.19
37	44069.95	51283.14	197029.42	California	89949.14
38	20229.59	65947.93	185265.10	New York	81229.06
39	38558.51	82982.09	174999.30	California	81005.76
40	28754.33	118546.05	172795.67	California	78239.91
41	27892.92	84710.77	164470.71	Florida	77798.83
42	23640.93	96189.63	148001.11	California	71498.49
43	15505.73	127382.30	35534.17	New York	69758.98
44	22177.74	154806.14	28334.72	California	65200.33
45	1000.23	124153.04	1903.93	New York	64926.08

46	1315.46	115816.21	297114.46	Florida	49490.75
47	0.00	135426.92	0.00	California	42559.73
48	542.05	51743.15	0.00	New York	35673.41
49	0.00	116983.80	45173.06	California	14681.40

```
df.head()
```

	R&D Spend	Administration	Marketing Spend	State	Profit
0	165349.20	136897.80	471784.10	New York	192261.83
1	162597.70	151377.59	443898.53	California	191792.06
2	153441.51	101145.55	407934.54	Florida	191050.39
3	144372.41	118671.85	383199.62	New York	182901.99
4	142107.34	91391.77	366168.42	Florida	166187.94

```
df.tail()
```

	R&D Spend	Administration	Marketing Spend	State	Profit
45	1000.23	124153.04	1903.93	New York	64926.08
46	1315.46	115816.21	297114.46	Florida	49490.75
47	0.00	135426.92	0.00	California	42559.73
48	542.05	51743.15	0.00	New York	35673.41
49	0.00	116983.80	45173.06	California	14681.40

```
df.shape
```

```
(50, 5)
```

```
df.columns
```

```
Index(['R&D Spend', 'Administration', 'Marketing Spend', 'State', 'Profit'], dtype='object')
```

```
df.isna().sum()
```

```
R&D Spend          0
Administration     0
Marketing Spend    0
State              0
Profit             0
dtype: int64
```

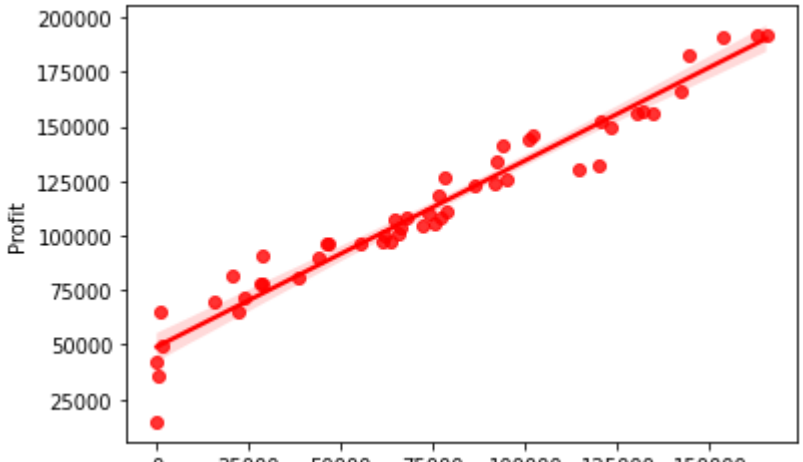
```
x=df.iloc[:,:-1]
x
y=df.iloc[:,-1]
y
```

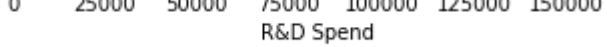
```
0     192261.83
1     191792.06
2     191050.39
3     182901.99
4     166187.94
5     156991.12
6     156122.51
7     155752.60
8     152211.77
9     149759.96
10    146121.95
11    144259.40
12    141585.52
13    134307.35
14    132602.65
15    129917.04
16    126992.93
17    125370.37
18    124266.90
19    122776.86
20    118474.03
21    111313.02
22    110352.25
23    108733.99
24    108552.04
25    107404.34
```

```
26    105733.54
27    105008.31
28    103282.38
29    101004.64
30     99937.59
31     97483.56
32     97427.84
33     96778.92
34     96712.80
35     96479.51
36     90708.19
37     89949.14
38     81229.06
39     81005.76
40     78239.91
41     77798.83
42     71498.49
43     69758.98
44     65200.33
45     64926.08
46     49490.75
47     42559.73
48     35673.41
49     14681.40
Name: Profit, dtype: float64
```

```
import seaborn as sns
sns.regplot(x=df['R&D Spend'],y=y,color='red')
```

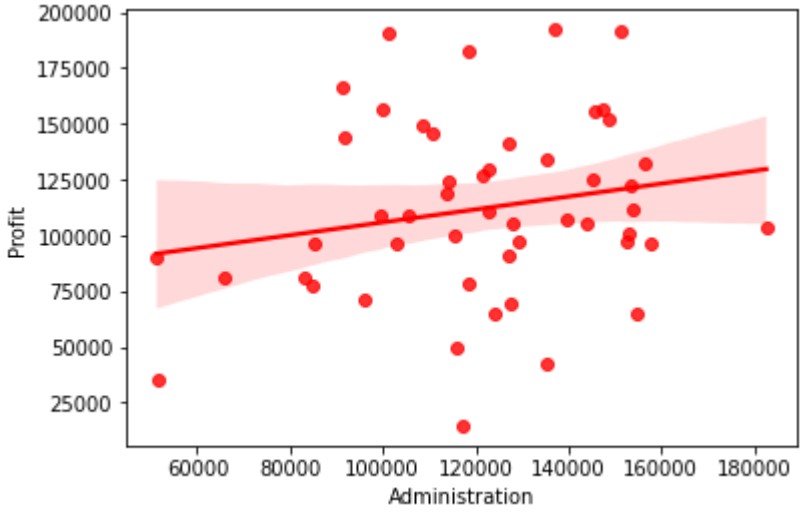
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4901009b20>
```





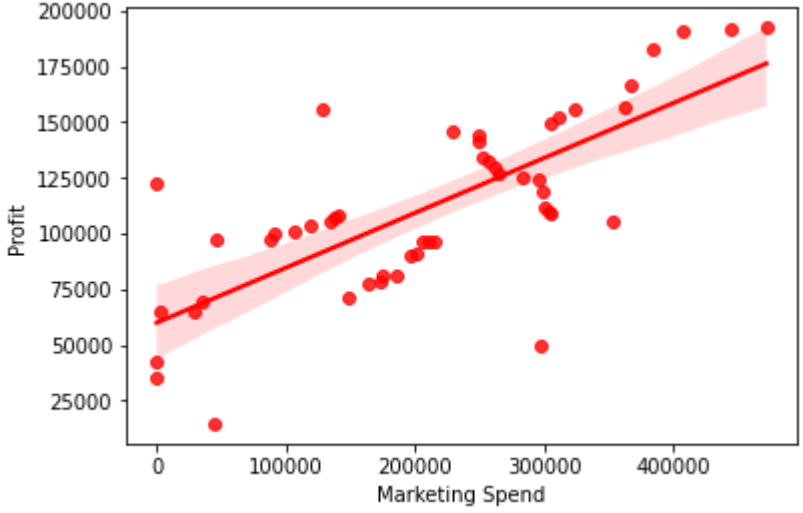

```
sns.regplot(x=df['Administration'],y=y,color='red')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4900d854f0>
```




```
sns.regplot(x=df['Marketing Spend'],y=y,color='red')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4900d66b20>
```




```
df.groupby('State')['State'].count()
```

```
State
California    17
Florida       16
New York      17
Name: State, dtype: int64
```

```
# state act as object
# ENCODINg(To convert numerical)

# here we use "one hot encoding"
#   *incrase te number of features,expanding number of columns

# Drawbacks of label encoding
  #  *forming a hyrarchy(higher value have higher priority  and  lower value have lower priority)
```

```
df.shape
```

```
(50, 5)
```

```
from sklearn.compose import make_column_transformer  #different column
from sklearn.preprocessing import OneHotEncoder

# handle_unknown='ignore'====>ignore the unknown data when comes into testing
# remainder=passthrough===>pass the remaining data
col_trans=make_column_transformer((OneHotEncoder(handle_unknown='ignore'),['State']),remainder='passthrough')

x=col_trans.fit_transform(x)
x
```

```
array([[0.0000000e+00, 0.0000000e+00, 1.0000000e+00, 1.6534920e+05,
        1.3689780e+05, 4.7178410e+05],
       [1.0000000e+00, 0.0000000e+00, 0.0000000e+00, 1.6259770e+05,
        1.5137759e+05, 4.4389853e+05],
       [0.0000000e+00, 1.0000000e+00, 0.0000000e+00, 1.5344151e+05,
        1.0114555e+05, 4.0793454e+05],
       [0.0000000e+00, 0.0000000e+00, 1.0000000e+00, 1.4437241e+05,
        1.1867185e+05, 3.8319962e+05],
       [0.0000000e+00, 1.0000000e+00, 0.0000000e+00, 1.4210734e+05,
        9.1391770e+04, 3.6616842e+05],
       [0.0000000e+00, 0.0000000e+00, 1.0000000e+00, 1.3187690e+05,
```

```
 9.9814710e+04, 3.6286136e+05],
[1.0000000e+00, 0.0000000e+00, 0.0000000e+00, 1.3461546e+05,
 1.4719887e+05, 1.2771682e+05],
[0.0000000e+00, 1.0000000e+00, 0.0000000e+00, 1.3029813e+05,
 1.4553006e+05, 3.2387668e+05],
[0.0000000e+00, 0.0000000e+00, 1.0000000e+00, 1.2054252e+05,
 1.4871895e+05, 3.1161329e+05],
[1.0000000e+00, 0.0000000e+00, 0.0000000e+00, 1.2333488e+05,
 1.0867917e+05, 3.0498162e+05],
[0.0000000e+00, 1.0000000e+00, 0.0000000e+00, 1.0191308e+05,
 1.1059411e+05, 2.2916095e+05],
[1.0000000e+00, 0.0000000e+00, 0.0000000e+00, 1.0067196e+05,
 9.1790610e+04, 2.4974455e+05],
[0.0000000e+00, 1.0000000e+00, 0.0000000e+00, 9.3863750e+04,
 1.2732038e+05, 2.4983944e+05],
[1.0000000e+00, 0.0000000e+00, 0.0000000e+00, 9.1992390e+04,
 1.3549507e+05, 2.5266493e+05],
[0.0000000e+00, 1.0000000e+00, 0.0000000e+00, 1.1994324e+05,
 1.5654742e+05, 2.5651292e+05],
[0.0000000e+00, 0.0000000e+00, 1.0000000e+00, 1.1452361e+05,
 1.2261684e+05, 2.6177623e+05],
[1.0000000e+00, 0.0000000e+00, 0.0000000e+00, 7.8013110e+04,
 1.2159755e+05, 2.6434606e+05],
[0.0000000e+00, 0.0000000e+00, 1.0000000e+00, 9.4657160e+04,
 1.4507758e+05, 2.8257431e+05],
[0.0000000e+00, 1.0000000e+00, 0.0000000e+00, 9.1749160e+04,
 1.1417579e+05, 2.9491957e+05],
[0.0000000e+00, 0.0000000e+00, 1.0000000e+00, 8.6419700e+04,
 1.5351411e+05, 0.0000000e+00],
[1.0000000e+00, 0.0000000e+00, 0.0000000e+00, 7.6253860e+04,
 1.1386730e+05, 2.9866447e+05],
[0.0000000e+00, 0.0000000e+00, 1.0000000e+00, 7.8389470e+04,
 1.5377343e+05, 2.9973729e+05],
[0.0000000e+00, 1.0000000e+00, 0.0000000e+00, 7.3994560e+04,
 1.2278275e+05, 3.0331926e+05],
[0.0000000e+00, 1.0000000e+00, 0.0000000e+00, 6.7532530e+04,
 1.0575103e+05, 3.0476873e+05],
[0.0000000e+00, 0.0000000e+00, 1.0000000e+00, 7.7044010e+04,
 9.9281340e+04, 1.4057481e+05],
[1.0000000e+00, 0.0000000e+00, 0.0000000e+00, 6.4664710e+04,
 1.3955316e+05, 1.3796262e+05],
[0.0000000e+00, 1.0000000e+00, 0.0000000e+00, 7.5328870e+04,
 1.4413598e+05, 1.3405007e+05],
[0.0000000e+00, 0.0000000e+00, 1.0000000e+00, 7.2107600e+04,
 1.2786455e+05, 3.5318381e+05],
[0.0000000e+00, 1.0000000e+00, 0.0000000e+00, 6.6051520e+04,
```

```
           1.8264556e+05, 1.1814820e+05],
```

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30,random_state=42)
x_train
```

```
    array([[1.0000000e+00, 0.0000000e+00, 0.0000000e+00, 1.3461546e+05,
            1.4719887e+05, 1.2771682e+05],
           [0.0000000e+00, 1.0000000e+00, 0.0000000e+00, 2.7892920e+04,
            8.4710770e+04, 1.6447071e+05],
           [0.0000000e+00, 1.0000000e+00, 0.0000000e+00, 1.3154600e+03,
            1.1581621e+05, 2.9711446e+05],
           [1.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
            1.3542692e+05, 0.0000000e+00],
           [0.0000000e+00, 0.0000000e+00, 1.0000000e+00, 1.1452361e+05,
            1.2261684e+05, 2.6177623e+05],
           [1.0000000e+00, 0.0000000e+00, 0.0000000e+00, 1.2333488e+05,
            1.0867917e+05, 3.0498162e+05],
           [1.0000000e+00, 0.0000000e+00, 0.0000000e+00, 7.8013110e+04,
            1.2159755e+05, 2.6434606e+05],
           [0.0000000e+00, 0.0000000e+00, 1.0000000e+00, 7.7044010e+04,
            9.9281340e+04, 1.4057481e+05],
           [1.0000000e+00, 0.0000000e+00, 0.0000000e+00, 4.6426070e+04,
            1.5769392e+05, 2.1079767e+05],
           [0.0000000e+00, 0.0000000e+00, 1.0000000e+00, 6.1136380e+04,
            1.5270192e+05, 8.8218230e+04],
           [0.0000000e+00, 0.0000000e+00, 1.0000000e+00, 1.6534920e+05,
            1.3689780e+05, 4.7178410e+05],
           [1.0000000e+00, 0.0000000e+00, 0.0000000e+00, 2.2177740e+04,
            1.5480614e+05, 2.8334720e+04],
           [0.0000000e+00, 0.0000000e+00, 1.0000000e+00, 7.2107600e+04,
            1.2786455e+05, 3.5318381e+05],
           [0.0000000e+00, 1.0000000e+00, 0.0000000e+00, 5.5493950e+04,
            1.0305749e+05, 2.1463481e+05],
           [0.0000000e+00, 0.0000000e+00, 1.0000000e+00, 1.3187690e+05,
            9.9814710e+04, 3.6286136e+05],
           [0.0000000e+00, 0.0000000e+00, 1.0000000e+00, 6.5605480e+04,
            1.5303206e+05, 1.0713838e+05],
           [1.0000000e+00, 0.0000000e+00, 0.0000000e+00, 1.0067196e+05,
            9.1790610e+04, 2.4974455e+05],
           [0.0000000e+00, 1.0000000e+00, 0.0000000e+00, 2.8663760e+04,
            1.2705621e+05, 2.0112682e+05],
           [1.0000000e+00, 0.0000000e+00, 0.0000000e+00, 1.6259770e+05,
            1.5137759e+05, 4.4389853e+05],
           [0.0000000e+00, 0.0000000e+00, 1.0000000e+00, 7.8389470e+04,
```

```
 1.5377343e+05, 2.9973729e+05],
[0.0000000e+00, 1.0000000e+00, 0.0000000e+00, 1.5344151e+05,
 1.0114555e+05, 4.0793454e+05],
[0.0000000e+00, 0.0000000e+00, 1.0000000e+00, 1.5505730e+04,
 1.2738230e+05, 3.5534170e+04],
[0.0000000e+00, 0.0000000e+00, 1.0000000e+00, 4.6014020e+04,
 8.5047440e+04, 2.0551764e+05],
[0.0000000e+00, 1.0000000e+00, 0.0000000e+00, 6.7532530e+04,
 1.0575103e+05, 3.0476873e+05],
[1.0000000e+00, 0.0000000e+00, 0.0000000e+00, 2.8754330e+04,
 1.1854605e+05, 1.7279567e+05],
[0.0000000e+00, 1.0000000e+00, 0.0000000e+00, 1.0191308e+05,
 1.1059411e+05, 2.2916095e+05],
[0.0000000e+00, 1.0000000e+00, 0.0000000e+00, 7.3994560e+04,
 1.2278275e+05, 3.0331926e+05],
[0.0000000e+00, 1.0000000e+00, 0.0000000e+00, 9.1749160e+04,
 1.1417579e+05, 2.9491957e+05],
[1.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
 1.1698380e+05, 4.5173060e+04],
```

```
from sklearn.linear_model import LinearRegression
model=LinearRegression()
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
y_pred
```

```
array([126187.39411505,  85788.82259512,  99777.02815177,  45706.12238326,
       127062.20722772,  51891.83884457, 109114.62977494, 100600.61123701,
        97953.99874714, 111730.57706807, 128818.49200668, 174195.35772633,
        93736.28538439, 148381.04097161, 172313.8713939 ])
```

```
df=pd.DataFrame({'actual_value':y_test,'predicted_value':y_pred})
df
```

	actual_value	predicted_value
13	134307.35	126187.394115
39	81005.76	85788.822595
30	99937.59	99777.028152
45	64926.08	45706.122383

17	125370.37	127062.207228
48	35673.41	51891.838845
26	105733.54	109114.629775
25	107404.34	100600.611237
32	97427.84	97953.998747
19	122776.86	111730.577068
12	141585.52	128818.492007
4	166187.94	174195.357726
37	89949.14	93736.285384
8	152211.77	148381.040972
3	182901.99	172313.871394

```
print("intercept",model.intercept_)
print("slope is",model.coef_)
```

```
intercept 57153.61206241345
slope is [ 2.59028652e+02  7.17099427e+02 -9.76128080e+02  8.04937292e-01
 -9.12577104e-02  2.80672826e-02]
```

```
list(zip(x,model.coef_))
```

```
[(array([0.000000e+00, 0.000000e+00, 1.000000e+00, 1.653492e+05,
         1.368978e+05, 4.717841e+05]), 259.0286523053593),
 (array([1.0000000e+00, 0.0000000e+00, 0.0000000e+00, 1.6259770e+05,
         1.5137759e+05, 4.4389853e+05]), 717.0994272258821),
 (array([0.0000000e+00, 1.0000000e+00, 0.0000000e+00, 1.5344151e+05,
         1.0114555e+05, 4.0793454e+05]), -976.1280795289858),
 (array([0.0000000e+00, 0.0000000e+00, 1.0000000e+00, 1.4437241e+05,
         1.1867185e+05, 3.8319962e+05]), 0.8049372918011102),
 (array([0.0000000e+00, 1.0000000e+00, 0.0000000e+00, 1.4210734e+05,
         9.1391770e+04, 3.6616842e+05]), -0.09125771038947761),
 (array([0.0000000e+00, 0.0000000e+00, 1.0000000e+00, 1.3187690e+05,
         9.9814710e+04, 3.6286136e+05]), 0.028067282565416463)]
```

```
from sklearn.metrics import mean_absolute_error
print("error is",mean_absolute_error(y_test,y_pred))

from sklearn.metrics import mean_absolute_percentage_error
print("percentage error",mean_absolute_percentage_error(y_test,y_pred))

from sklearn.metrics import mean_squared_error
print("squared error is",mean_squared_error(y_test,y_pred))

from sklearn.metrics import mean_squared_error
z=mean_squared_error(y_test,y_pred)
print("root_mean_squared_error is",np.sqrt(z))

from sklearn.metrics import r2_score
print("r2 score is",r2_score(y_test,y_pred))
```

```
error is 7395.4335315232565
percentage error 0.08929865344171896
squared error is 84826955.03534976
root_mean_squared_error is 9210.154995186007
r2 score is 0.9397108063355675
```

Colab paid products - Cancel contracts here