Multiple Linear Regression is one of the important regression algorithms which models the linear relationship between a single dependent continuous variable and more than one independent variable.

In [ ]:
```
"""Problem Description:

We have a dataset of 50 start-up companies.
This dataset contains five main information:
R&D Spend, Administration Spend, Marketing Spend, State, and Profit for a fi
Our goal is to create a model that can easily determine which company has a
and which is the most affecting factor for the profit of a company."""
```

Out[1]:
```
'Problem Description:\n\nWe have a dataset of 50 start-up companies. \nThi
s dataset contains five main information: \nR&D Spend, Administration Spen
d, Marketing Spend, State, and Profit for a financial year. \nOur goal is
to create a model that can easily determine which company has a maximum pr
ofit, and which is the most affecting factor for the profit of a company.'
```

In [ ]:
```
#Steps to implement Machine Learning
"""duplicated Values Remove
Null values Remove
Categorical Column remove
standardised data
spliting the data
train the data
testing the data"""
```

Out[2]:
```
'duplicated Values Remove\nNull values Remove\nCategorical Column remove\n
standardised data\nspliting the data\ntrain the data\ntesting the data'
```

In [ ]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [ ]:
```
df=pd.read_csv("/content/sample_data/35 Startups_Multiple_Linear_Regression
```

In [ ]:
```
df.head()
```

Out[6]:

	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

In [ ]:
```
df.shape
```

Out[8]:
```
(50, 5)
```

In [ ]:
```
df.size
```

Out[9]:
```
250
```

In [ ]:
```
df.dtypes
```

Out[10]:
```
R&D Spend          float64
Administration     float64
Marketing Spend    float64
State               object
Profit             float64
dtype: object
```

In [ ]:
```
df.describe()
```

Out[11]:

	R&D Spend	Administration	Marketing Spend	Profit
count	50.000000	50.000000	50.000000	50.000000
mean	73721.615600	121344.639600	211025.097800	112012.639200
std	45902.256482	28017.802755	122290.310726	40306.180338
min	0.000000	51283.140000	0.000000	14681.400000
25%	39936.370000	103730.875000	129300.132500	90138.902500
50%	73051.080000	122699.795000	212716.240000	107978.190000
75%	101602.800000	144842.180000	299469.085000	139765.977500
max	165349.200000	182645.560000	471784.100000	192261.830000

In [ ]:
```
df.columns
```

Out[12]:
```
Index(['R&D Spend', 'Administration', 'Marketing Spend', 'State', 'Profi
t'], dtype='object')
```

In [ ]:
```
df.nunique()
```

Out[13]:
```
R&D Spend          49
Administration     50
Marketing Spend    48
State               3
Profit             50
dtype: int64
```

In [ ]:
```
df.duplicated()
```

Out[14]:
```
0     False
1     False
2     False
3     False
4     False
5     False
6     False
7     False
8     False
9     False
10    False
11    False
12    False
13    False
14    False
15    False
16    False
17    False
18    False
19    False
20    False
21    False
22    False
23    False
24    False
25    False
26    False
27    False
28    False
29    False
30    False
31    False
32    False
33    False
34    False
35    False
36    False
37    False
38    False
39    False
40    False
41    False
42    False
43    False
44    False
45    False
46    False
47    False
48    False
49    False
dtype: bool
```

In [ ]:
```
df.duplicated().sum()
```

Out[15]:
```
0
```

In [ ]:
```
df.drop_duplicates(inplace=True)
```

In [ ]:
```
df.duplicated().sum()
```

Out[17]:
```
0
```

In [ ]:
```
df.dropna() #dropping NA values
```

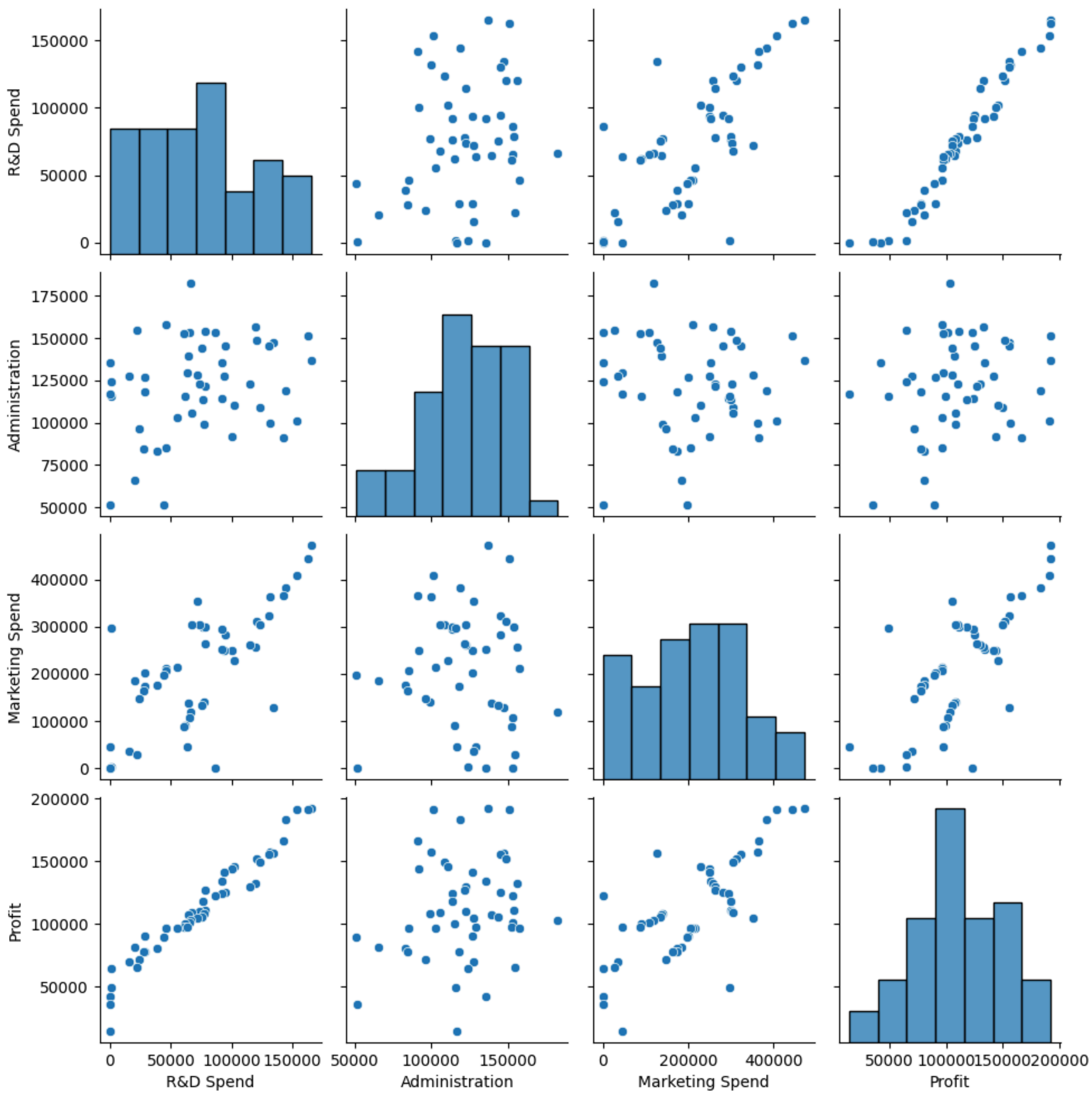
Out[18]:

	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
20	76253.86	113867.30	298664.47	California	118474.03
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

	R&D Spend	Administration	Marketing Spend	State	Profit
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

In [ ]:
```
sns.pairplot(df)
```

Out[19]: <seaborn.axisgrid.PairGrid at 0x7b6f8ae9a2f0>



In [ ]:
```
X=df[["R&D Spend","Administration","Marketing Spend"]]
```

In [ ]:
```
X.shape
```

Out[23]:
```
(50, 3)
```

In [ ]:
```
from sklearn.preprocessing import StandardScaler #sacleing the features bec
```

In [ ]:
```
ss=StandardScaler()
X=ss.fit_transform(X)
```

In [ ]:
```
X
```

Out[32]:
```
array([[ 2.01641149e+00,  5.60752915e-01,  2.15394309e+00],
       [ 1.95586034e+00,  1.08280658e+00,  1.92360040e+00],
       [ 1.75436374e+00, -7.28257028e-01,  1.62652767e+00],
       [ 1.55478369e+00, -9.63646307e-02,  1.42221024e+00],
       [ 1.50493720e+00, -1.07991935e+00,  1.28152771e+00],
       [ 1.27980001e+00, -7.76239071e-01,  1.25421046e+00],
       [ 1.34006641e+00,  9.32147208e-01, -6.88149930e-01],
       [ 1.24505666e+00,  8.71980011e-01,  9.32185978e-01],
       [ 1.03036886e+00,  9.86952101e-01,  8.30886909e-01],
       [ 1.09181921e+00, -4.56640246e-01,  7.76107440e-01],
       [ 6.20398248e-01, -3.87599089e-01,  1.49807267e-01],
       [ 5.93085418e-01, -1.06553960e+00,  3.19833623e-01],
       [ 4.43259872e-01,  2.15449064e-01,  3.20617441e-01],
       [ 4.02077603e-01,  5.10178953e-01,  3.43956788e-01],
       [ 1.01718075e+00,  1.26919939e+00,  3.75742273e-01],
       [ 8.97913123e-01,  4.58678535e-02,  4.19218702e-01],
       [ 9.44411957e-02,  9.11841968e-03,  4.40446224e-01],
       [ 4.60720127e-01,  8.55666318e-01,  5.91016724e-01],
       [ 3.96724938e-01, -2.58465367e-01,  6.92992062e-01],
       [ 2.79441650e-01,  1.15983657e+00, -1.74312698e+00],
       [ 5.57260867e-02, -2.69587651e-01,  7.23925995e-01],
       [ 1.02723599e-01,  1.16918609e+00,  7.32787791e-01],
       [ 6.00657792e-03,  5.18495648e-02,  7.62375876e-01],
       [-1.36200724e-01, -5.62211268e-01,  7.74348908e-01],
       [ 7.31146008e-02, -7.95469167e-01, -5.81939297e-01],
       [-1.99311688e-01,  6.56489139e-01, -6.03516725e-01],
       [ 3.53702028e-02,  8.21717916e-01, -6.35835495e-01],
       [-3.55189938e-02,  2.35068543e-01,  1.17427116e+00],
       [-1.68792717e-01,  2.21014050e+00, -7.67189437e-01],
       [-1.78608540e-01,  1.14245677e+00, -8.58133663e-01],
       [-2.58074369e-01, -2.05628659e-01, -9.90357166e-01],
       [-2.76958231e-01,  1.13055391e+00, -1.01441945e+00],
       [-2.26948675e-01,  2.83923813e-01, -1.36244978e+00],
       [-4.01128925e-01, -6.59324033e-01,  2.98172434e-02],
       [-6.00682122e-01,  1.31053525e+00, -1.87861793e-03],
       [-6.09749941e-01, -1.30865753e+00, -4.54931587e-02],
       [-9.91570153e-01,  2.05924691e-01, -8.17625734e-02],
       [-6.52532310e-01, -2.52599402e+00, -1.15608256e-01],
       [-1.17717755e+00, -1.99727037e+00, -2.12784866e-01],
       [-7.73820359e-01, -1.38312156e+00, -2.97583276e-01],
       [-9.89577015e-01, -1.00900218e-01, -3.15785883e-01],
       [-1.00853372e+00, -1.32079581e+00, -3.84552407e-01],
       [-1.10210556e+00, -9.06937535e-01, -5.20595959e-01],
       [-1.28113364e+00,  2.17681524e-01, -1.44960468e+00],
       [-1.13430539e+00,  1.20641936e+00, -1.50907418e+00],
       [-1.60035036e+00,  1.01253936e-01, -1.72739998e+00],
       [-1.59341322e+00, -1.99321741e-01,  7.11122474e-01],
       [-1.62236202e+00,  5.07721876e-01, -1.74312698e+00],
       [-1.61043334e+00, -2.50940884e+00, -1.74312698e+00],
       [-1.62236202e+00, -1.57225506e-01, -1.36998473e+00]])
```

In [ ]:
```
Y=df["Profit"]
```

In [ ]:
```
Y.shape
```

Out[35]:
```
(50,)
```

In [ ]:
```
from sklearn.model_selection import train_test_split
```

In [ ]:
```
#split the dataset into training data and training data
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.25,random_st
```

In [ ]:
```
X_train.shape,X_test.shape,Y_train.shape,Y_test.shape
```

Out[38]:
```
((37, 3), (13, 3), (37,), (13,))
```

In [ ]:
```
#now implement the Linear Regression Model
```

In [ ]:
```
from sklearn.linear_model import LinearRegression
```

In [ ]:
```
lr=LinearRegression()
```

In [ ]:
```
lr.fit(X_train,Y_train)
```

Out[42]:
```
LinearRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [ ]:
```
y_pred=lr.predict(X_test)
```

In [ ]:
```
y_pred
```

Out[44]:
```
array([ 88361.6924659 , 109068.75037541,  66233.18132181,  70645.38100143,
        48118.47333074, 115786.66944536, 171799.96557761,  99617.55808099,
       159031.78297409, 157877.26074356,  83222.30531514, 179714.94106163,
        75105.99525989])
```

In [ ]:
```
lr.score(X_train,Y_train)
```

Out[45]:
```
0.9310605936487033
```

In [ ]:
```
lr.score(X_test,Y_test)
```

Out[46]:
```
0.9878392927652377
```

In [ ]:
```
lr.intercept_
```

Out[47]:
```
111675.78913907126
```

In [ ]:
```
lr.coef_
```

Out[48]:
```
array([36077.48534245,  -219.17000538,  2819.81544887])
```

In [ ]:
```
#now find the error/ cost function/loss function
```

In [ ]:
```
from sklearn.metrics import mean_absolute_error, mean_absolute_percentage_e
```

In [ ]:
```
mean_absolute_error(Y_test,y_pred)
```

Out[51]:
```
3476.6285513802627
```

In [ ]:
```
mean_absolute_percentage_error(Y_test,y_pred)
```

Out[52]:
```
0.032672215269165125
```

Method 2 from sir point of views- refer from project no 8

In [ ]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [ ]:
```
df2=pd.read_csv("/content/sample_data/35 Startups_Multiple_Linear_Regressio
```

In [ ]:
```
df2.head(5)
```

Out[4]:

	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

In [ ]:
```
df2.shape
```

Out[5]:
```
(50, 5)
```

In [ ]:
```
df2.size
```

Out[6]:
```
250
```

In [ ]:
```
df2.dtypes
```

Out[7]:
```
R&D Spend          float64
Administration     float64
Marketing Spend    float64
State               object
Profit             float64
dtype: object
```

In [ ]:
```
df2.ndim
```

Out[8]:
```
2
```

In [ ]:
```
df2.describe()
```

Out[10]:

	R&D Spend	Administration	Marketing Spend	Profit
count	50.000000	50.000000	50.000000	50.000000
mean	73721.615600	121344.639600	211025.097800	112012.639200
std	45902.256482	28017.802755	122290.310726	40306.180338
min	0.000000	51283.140000	0.000000	14681.400000
25%	39936.370000	103730.875000	129300.132500	90138.902500
50%	73051.080000	122699.795000	212716.240000	107978.190000
75%	101602.800000	144842.180000	299469.085000	139765.977500
max	165349.200000	182645.560000	471784.100000	192261.830000

In [ ]:
```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   R&D Spend        50 non-null     float64
 1   Administration   50 non-null     float64
 2   Marketing Spend  50 non-null     float64
 3   State            50 non-null     object 
 4   Profit           50 non-null     float64
dtypes: float64(4), object(1)
memory usage: 2.1+ KB
```

In [ ]:
```
df2.columns
```

Out[16]:
```
Index(['R&D Spend', 'Administration', 'Marketing Spend', 'State', 'Profi
t'], dtype='object')
```

In [ ]:
```
df2.nunique()
```

Out[17]:
```
R&D Spend          49
Administration     50
Marketing Spend    48
State               3
Profit             50
dtype: int64
```

In [ ]:
```
df2["R&D Spend"].unique()
```

Out[18]:
```
array([165349.2 , 162597.7 , 153441.51, 144372.41, 142107.34, 131876.9 ,
       134615.46, 130298.13, 120542.52, 123334.88, 101913.08, 100671.96,
        93863.75,  91992.39, 119943.24, 114523.61,  78013.11,  94657.16,
        91749.16,  86419.7 ,  76253.86,  78389.47,  73994.56,  67532.53,
        77044.01,  64664.71,  75328.87,  72107.6 ,  66051.52,  65605.48,
        61994.48,  61136.38,  63408.86,  55493.95,  46426.07,  46014.02,
        28663.76,  44069.95,  20229.59,  38558.51,  28754.33,  27892.92,
        23640.93,  15505.73,  22177.74,   1000.23,   1315.46,      0.  ,
          542.05])
```

In [ ]:
```
df2["Administration"].unique()
```

Out[19]:
```
array([136897.8 , 151377.59, 101145.55, 118671.85,  91391.77,  99814.71,
       147198.87, 145530.06, 148718.95, 108679.17, 110594.11,  91790.61,
       127320.38, 135495.07, 156547.42, 122616.84, 121597.55, 145077.58,
       114175.79, 153514.11, 113867.3 , 153773.43, 122782.75, 105751.03,
        99281.34, 139553.16, 144135.98, 127864.55, 182645.56, 153032.06,
       115641.28, 152701.92, 129219.61, 103057.49, 157693.92,  85047.44,
       127056.21,  51283.14,  65947.93,  82982.09, 118546.05,  84710.77,
        96189.63, 127382.3 , 154806.14, 124153.04, 115816.21, 135426.92,
        51743.15, 116983.8 ])
```

In [ ]:
```
df2["Marketing Spend"].unique()
```

Out[20]:
```
array([471784.1 , 443898.53, 407934.54, 383199.62, 366168.42, 362861.36,
       127716.82, 323876.68, 311613.29, 304981.62, 229160.95, 249744.55,
       249839.44, 252664.93, 256512.92, 261776.23, 264346.06, 282574.31,
       294919.57,      0.  , 298664.47, 299737.29, 303319.26, 304768.73,
       140574.81, 137962.62, 134050.07, 353183.81, 118148.2 , 107138.38,
        91131.24,  88218.23,  46085.25, 214634.81, 210797.67, 205517.64,
       201126.82, 197029.42, 185265.1 , 174999.3 , 172795.67, 164470.71,
       148001.11,  35534.17,  28334.72,   1903.93, 297114.46,  45173.06])
```

In [ ]:
```
df2["State"].unique()
```

Out[21]:
```
array(['New York', 'California', 'Florida'], dtype=object)
```

In [ ]:
```
df2["Profit"].unique()
```

Out[22]:
```
array([192261.83, 191792.06, 191050.39, 182901.99, 166187.94, 156991.12,
       156122.51, 155752.6 , 152211.77, 149759.96, 146121.95, 144259.4 ,
       141585.52, 134307.35, 132602.65, 129917.04, 126992.93, 125370.37,
       124266.9 , 122776.86, 118474.03, 111313.02, 110352.25, 108733.99,
       108552.04, 107404.34, 105733.54, 105008.31, 103282.38, 101004.64,
        99937.59,  97483.56,  97427.84,  96778.92,  96712.8 ,  96479.51,
        90708.19,  89949.14,  81229.06,  81005.76,  78239.91,  77798.83,
        71498.49,  69758.98,  65200.33,  64926.08,  49490.75,  42559.73,
        35673.41,  14681.4 ])
```

In [ ]:
```
df2.duplicated()
```

Out[25]:
```
0     False
1     False
2     False
3     False
4     False
5     False
6     False
7     False
8     False
9     False
10    False
11    False
12    False
13    False
14    False
15    False
16    False
17    False
18    False
19    False
20    False
21    False
22    False
23    False
24    False
25    False
26    False
27    False
28    False
29    False
30    False
31    False
32    False
33    False
34    False
35    False
36    False
37    False
38    False
39    False
40    False
41    False
42    False
43    False
44    False
45    False
46    False
47    False
48    False
49    False
dtype: bool
```

In [ ]:
```
df2.duplicated().count()
```

Out[26]:
```
50
```

In [ ]:
```
df2.duplicated().sum()
```

Out[27]:
```
0
```

In [ ]:
```
df2.drop_duplicates(inplace=True)
```

In [ ]:
```
df2.duplicated()
```

Out[29]:
```
0     False
1     False
2     False
3     False
4     False
5     False
6     False
7     False
8     False
9     False
10    False
11    False
12    False
13    False
14    False
15    False
16    False
17    False
18    False
19    False
20    False
21    False
22    False
23    False
24    False
25    False
26    False
27    False
28    False
29    False
30    False
31    False
32    False
33    False
34    False
35    False
36    False
37    False
38    False
39    False
40    False
41    False
42    False
43    False
44    False
45    False
46    False
47    False
48    False
49    False
dtype: bool
```

In [ ]:
```
df2.dropna()
```

Out[30]:

	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
20	76253.86	113867.30	298664.47	California	118474.03
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

	R&D Spend	Administration	Marketing Spend	State	Profit
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

In [ ]:
```
x=df2.iloc[:,:-1].values #features
```

In [ ]:
```
x
```

Out[32]:
```
array([[165349.2, 136897.8, 471784.1, 'New York'],
       [162597.7, 151377.59, 443898.53, 'California'],
       [153441.51, 101145.55, 407934.54, 'Florida'],
       [144372.41, 118671.85, 383199.62, 'New York'],
       [142107.34, 91391.77, 366168.42, 'Florida'],
       [131876.9, 99814.71, 362861.36, 'New York'],
       [134615.46, 147198.87, 127716.82, 'California'],
       [130298.13, 145530.06, 323876.68, 'Florida'],
       [120542.52, 148718.95, 311613.29, 'New York'],
       [123334.88, 108679.17, 304981.62, 'California'],
       [101913.08, 110594.11, 229160.95, 'Florida'],
       [100671.96, 91790.61, 249744.55, 'California'],
       [93863.75, 127320.38, 249839.44, 'Florida'],
       [91992.39, 135495.07, 252664.93, 'California'],
       [119943.24, 156547.42, 256512.92, 'Florida'],
       [114523.61, 122616.84, 261776.23, 'New York'],
       [78013.11, 121597.55, 264346.06, 'California'],
       [94657.16, 145077.58, 282574.31, 'New York'],
       [91749.16, 114175.79, 294919.57, 'Florida'],
       [86419.7, 153514.11, 0.0, 'New York'],
       [76253.86, 113867.3, 298664.47, 'California'],
       [78389.47, 153773.43, 299737.29, 'New York'],
       [73994.56, 122782.75, 303319.26, 'Florida'],
       [67532.53, 105751.03, 304768.73, 'Florida'],
       [77044.01, 99281.34, 140574.81, 'New York'],
       [64664.71, 139553.16, 137962.62, 'California'],
       [75328.87, 144135.98, 134050.07, 'Florida'],
       [72107.6, 127864.55, 353183.81, 'New York'],
       [66051.52, 182645.56, 118148.2, 'Florida'],
       [65605.48, 153032.06, 107138.38, 'New York'],
       [61994.48, 115641.28, 91131.24, 'Florida'],
       [61136.38, 152701.92, 88218.23, 'New York'],
       [63408.86, 129219.61, 46085.25, 'California'],
       [55493.95, 103057.49, 214634.81, 'Florida'],
       [46426.07, 157693.92, 210797.67, 'California'],
       [46014.02, 85047.44, 205517.64, 'New York'],
       [28663.76, 127056.21, 201126.82, 'Florida'],
       [44069.95, 51283.14, 197029.42, 'California'],
       [20229.59, 65947.93, 185265.1, 'New York'],
       [38558.51, 82982.09, 174999.3, 'California'],
       [28754.33, 118546.05, 172795.67, 'California'],
       [27892.92, 84710.77, 164470.71, 'Florida'],
       [23640.93, 96189.63, 148001.11, 'California'],
       [15505.73, 127382.3, 35534.17, 'New York'],
       [22177.74, 154806.14, 28334.72, 'California'],
       [1000.23, 124153.04, 1903.93, 'New York'],
       [1315.46, 115816.21, 297114.46, 'Florida'],
       [0.0, 135426.92, 0.0, 'California'],
       [542.05, 51743.15, 0.0, 'New York'],
       [0.0, 116983.8, 45173.06, 'California']], dtype=object)
```

In [ ]:
```
y=df2.iloc[:,-1].values #labels
```

In [ ]:
```
y
```

Out[34]:
```
array([192261.83, 191792.06, 191050.39, 182901.99, 166187.94, 156991.12,
       156122.51, 155752.6 , 152211.77, 149759.96, 146121.95, 144259.4 ,
       141585.52, 134307.35, 132602.65, 129917.04, 126992.93, 125370.37,
       124266.9 , 122776.86, 118474.03, 111313.02, 110352.25, 108733.99,
       108552.04, 107404.34, 105733.54, 105008.31, 103282.38, 101004.64,
        99937.59,  97483.56,  97427.84,  96778.92,  96712.8 ,  96479.51,
        90708.19,  89949.14,  81229.06,  81005.76,  78239.91,  77798.83,
        71498.49,  69758.98,  65200.33,  64926.08,  49490.75,  42559.73,
        35673.41,  14681.4 ])
```

In [ ]:
```
#now we have to handle the categorical column usning column transformer and
```

In [ ]:
```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
```

In [ ]:
```
from sklearn.preprocessing import OneHotEncoder
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(),[3])], rem
x= np.array(ct.fit_transform(x))
```

In [ ]:
```
x
```

Out[39]:
```
array([[0.0, 0.0, 1.0, 165349.2, 136897.8, 471784.1],
       [1.0, 0.0, 0.0, 162597.7, 151377.59, 443898.53],
       [0.0, 1.0, 0.0, 153441.51, 101145.55, 407934.54],
       [0.0, 0.0, 1.0, 144372.41, 118671.85, 383199.62],
       [0.0, 1.0, 0.0, 142107.34, 91391.77, 366168.42],
       [0.0, 0.0, 1.0, 131876.9, 99814.71, 362861.36],
       [1.0, 0.0, 0.0, 134615.46, 147198.87, 127716.82],
       [0.0, 1.0, 0.0, 130298.13, 145530.06, 323876.68],
       [0.0, 0.0, 1.0, 120542.52, 148718.95, 311613.29],
       [1.0, 0.0, 0.0, 123334.88, 108679.17, 304981.62],
       [0.0, 1.0, 0.0, 101913.08, 110594.11, 229160.95],
       [1.0, 0.0, 0.0, 100671.96, 91790.61, 249744.55],
       [0.0, 1.0, 0.0, 93863.75, 127320.38, 249839.44],
       [1.0, 0.0, 0.0, 91992.39, 135495.07, 252664.93],
       [0.0, 1.0, 0.0, 119943.24, 156547.42, 256512.92],
       [0.0, 0.0, 1.0, 114523.61, 122616.84, 261776.23],
       [1.0, 0.0, 0.0, 78013.11, 121597.55, 264346.06],
       [0.0, 0.0, 1.0, 94657.16, 145077.58, 282574.31],
       [0.0, 1.0, 0.0, 91749.16, 114175.79, 294919.57],
       [0.0, 0.0, 1.0, 86419.7, 153514.11, 0.0],
       [1.0, 0.0, 0.0, 76253.86, 113867.3, 298664.47],
       [0.0, 0.0, 1.0, 78389.47, 153773.43, 299737.29],
       [0.0, 1.0, 0.0, 73994.56, 122782.75, 303319.26],
       [0.0, 1.0, 0.0, 67532.53, 105751.03, 304768.73],
       [0.0, 0.0, 1.0, 77044.01, 99281.34, 140574.81],
       [1.0, 0.0, 0.0, 64664.71, 139553.16, 137962.62],
       [0.0, 1.0, 0.0, 75328.87, 144135.98, 134050.07],
       [0.0, 0.0, 1.0, 72107.6, 127864.55, 353183.81],
       [0.0, 1.0, 0.0, 66051.52, 182645.56, 118148.2],
       [0.0, 0.0, 1.0, 65605.48, 153032.06, 107138.38],
       [0.0, 1.0, 0.0, 61994.48, 115641.28, 91131.24],
       [0.0, 0.0, 1.0, 61136.38, 152701.92, 88218.23],
       [1.0, 0.0, 0.0, 63408.86, 129219.61, 46085.25],
       [0.0, 1.0, 0.0, 55493.95, 103057.49, 214634.81],
       [1.0, 0.0, 0.0, 46426.07, 157693.92, 210797.67],
       [0.0, 0.0, 1.0, 46014.02, 85047.44, 205517.64],
       [0.0, 1.0, 0.0, 28663.76, 127056.21, 201126.82],
       [1.0, 0.0, 0.0, 44069.95, 51283.14, 197029.42],
       [0.0, 0.0, 1.0, 20229.59, 65947.93, 185265.1],
       [1.0, 0.0, 0.0, 38558.51, 82982.09, 174999.3],
       [1.0, 0.0, 0.0, 28754.33, 118546.05, 172795.67],
       [0.0, 1.0, 0.0, 27892.92, 84710.77, 164470.71],
       [1.0, 0.0, 0.0, 23640.93, 96189.63, 148001.11],
       [0.0, 0.0, 1.0, 15505.73, 127382.3, 35534.17],
       [1.0, 0.0, 0.0, 22177.74, 154806.14, 28334.72],
       [0.0, 0.0, 1.0, 1000.23, 124153.04, 1903.93],
       [0.0, 1.0, 0.0, 1315.46, 115816.21, 297114.46],
       [1.0, 0.0, 0.0, 0.0, 135426.92, 0.0],
       [0.0, 0.0, 1.0, 542.05, 51743.15, 0.0],
       [1.0, 0.0, 0.0, 0.0, 116983.8, 45173.06]], dtype=object)
```

In [ ]:
```
#now we can implement the ML model train test split
```

In [ ]:
```
from sklearn.model_selection import train_test_split
```

In [ ]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25,random_st
```

In [ ]:
```
x_train
```

Out[43]:
```
array([[1.0, 0.0, 0.0, 63408.86, 129219.61, 46085.25],
       [0.0, 1.0, 0.0, 101913.08, 110594.11, 229160.95],
       [0.0, 0.0, 1.0, 78389.47, 153773.43, 299737.29],
       [0.0, 0.0, 1.0, 46014.02, 85047.44, 205517.64],
       [0.0, 0.0, 1.0, 72107.6, 127864.55, 353183.81],
       [0.0, 1.0, 0.0, 91749.16, 114175.79, 294919.57],
       [0.0, 0.0, 1.0, 61136.38, 152701.92, 88218.23],
       [1.0, 0.0, 0.0, 162597.7, 151377.59, 443898.53],
       [0.0, 1.0, 0.0, 93863.75, 127320.38, 249839.44],
       [1.0, 0.0, 0.0, 46426.07, 157693.92, 210797.67],
       [0.0, 0.0, 1.0, 1000.23, 124153.04, 1903.93],
       [0.0, 1.0, 0.0, 75328.87, 144135.98, 134050.07],
       [0.0, 0.0, 1.0, 131876.9, 99814.71, 362861.36],
       [1.0, 0.0, 0.0, 91992.39, 135495.07, 252664.93],
       [0.0, 1.0, 0.0, 73994.56, 122782.75, 303319.26],
       [0.0, 0.0, 1.0, 86419.7, 153514.11, 0.0],
       [0.0, 0.0, 1.0, 94657.16, 145077.58, 282574.31],
       [0.0, 1.0, 0.0, 119943.24, 156547.42, 256512.92],
       [0.0, 1.0, 0.0, 142107.34, 91391.77, 366168.42],
       [0.0, 1.0, 0.0, 27892.92, 84710.77, 164470.71],
       [0.0, 1.0, 0.0, 55493.95, 103057.49, 214634.81],
       [0.0, 0.0, 1.0, 77044.01, 99281.34, 140574.81],
       [1.0, 0.0, 0.0, 100671.96, 91790.61, 249744.55],
       [0.0, 0.0, 1.0, 20229.59, 65947.93, 185265.1],
       [1.0, 0.0, 0.0, 78013.11, 121597.55, 264346.06],
       [0.0, 0.0, 1.0, 542.05, 51743.15, 0.0],
       [0.0, 1.0, 0.0, 1315.46, 115816.21, 297114.46],
       [1.0, 0.0, 0.0, 0.0, 116983.8, 45173.06],
       [0.0, 0.0, 1.0, 120542.52, 148718.95, 311613.29],
       [0.0, 0.0, 1.0, 15505.73, 127382.3, 35534.17],
       [0.0, 0.0, 1.0, 65605.48, 153032.06, 107138.38],
       [1.0, 0.0, 0.0, 64664.71, 139553.16, 137962.62],
       [0.0, 1.0, 0.0, 66051.52, 182645.56, 118148.2],
       [0.0, 0.0, 1.0, 165349.2, 136897.8, 471784.1],
       [0.0, 0.0, 1.0, 114523.61, 122616.84, 261776.23],
       [0.0, 1.0, 0.0, 28663.76, 127056.21, 201126.82],
       [1.0, 0.0, 0.0, 123334.88, 108679.17, 304981.62]], dtype=object)
```

In [ ]:
```
y_train
```

Out[44]:
```
array([ 97427.84, 146121.95, 111313.02,  96479.51, 105008.31, 124266.9 ,
        97483.56, 191792.06, 141585.52,  96712.8 ,  64926.08, 105733.54,
       156991.12, 134307.35, 110352.25, 122776.86, 125370.37, 132602.65,
       166187.94,  77798.83,  96778.92, 108552.04, 144259.4 ,  81229.06,
       126992.93,  35673.41,  49490.75,  14681.4 , 152211.77,  69758.98,
       101004.64, 107404.34, 103282.38, 192261.83, 129917.04,  90708.19,
       149759.96])
```

In [ ]:
```
y_train
```

Out[45]:
```
array([ 97427.84, 146121.95, 111313.02,  96479.51, 105008.31, 124266.9 ,
        97483.56, 191792.06, 141585.52,  96712.8 ,  64926.08, 105733.54,
       156991.12, 134307.35, 110352.25, 122776.86, 125370.37, 132602.65,
       166187.94,  77798.83,  96778.92, 108552.04, 144259.4 ,  81229.06,
       126992.93,  35673.41,  49490.75,  14681.4 , 152211.77,  69758.98,
       101004.64, 107404.34, 103282.38, 192261.83, 129917.04,  90708.19,
       149759.96])
```

In [ ]:
```
y_test
```

Out[46]:
```
array([ 89949.14, 108733.99,  65200.33,  71498.49,  42559.73, 118474.03,
       182901.99,  99937.59, 155752.6 , 156122.51,  81005.76, 191050.39,
        78239.91])
```

In [ ]:
```
#now implement the linear regression model to predict the new outcomes
```

In [ ]:
```
from sklearn.linear_model import LinearRegression
```

In [ ]:
```
reg=LinearRegression()
```

In [ ]:
```
reg.fit(x_train,y_train)
```

Out[50]:
```
LinearRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [ ]:
```
reg.coef_
```

Out[51]:
```
array([ 3.33308401e+02,  1.90646775e+02, -5.23955177e+02,  7.94908963e-01,
       -9.24199896e-03,  2.26310528e-02])
```

In [ ]:
```
reg.intercept_
```

Out[52]:
```
49507.17141495356
```

In [ ]:
```
reg.score(x_train,y_train)
```

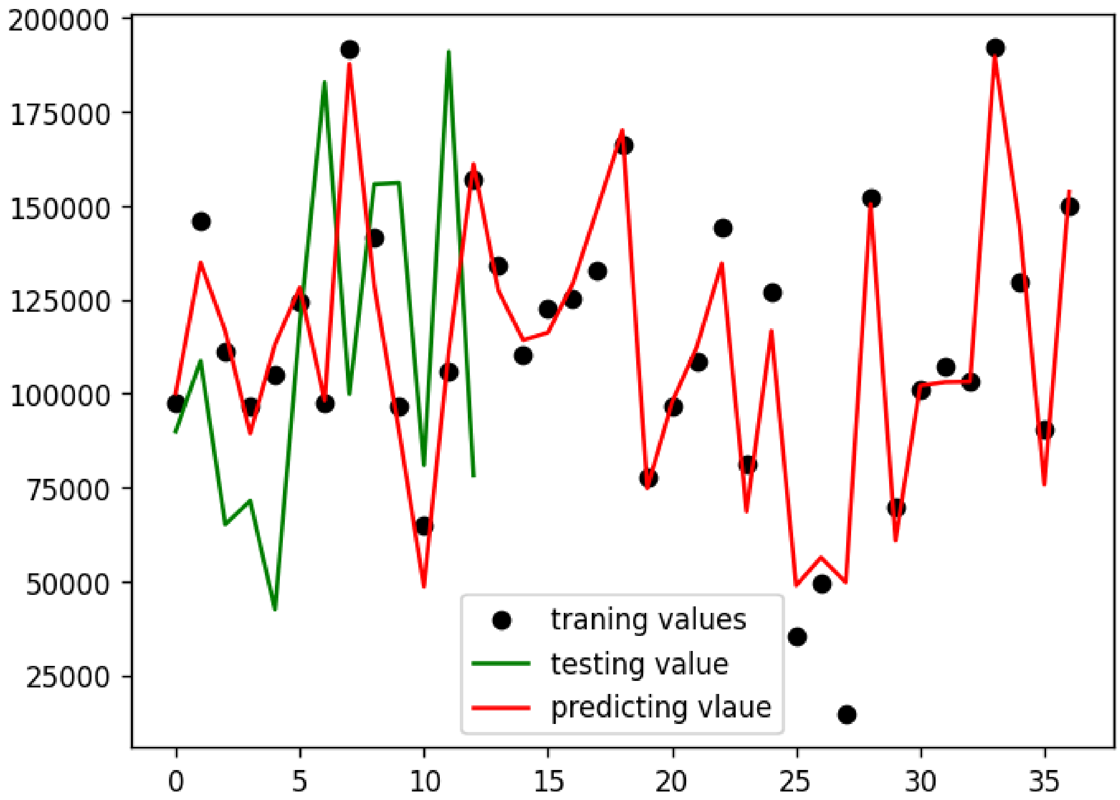
Out[53]:
```
0.9311628200093645
```

In [ ]:
```
reg.score(x_test,y_test)
```

Out[54]:
```
0.9874051459541274
```

In [ ]:
```
plt.figure(dpi=120)
plt.plot(y_train,"o",color="Black",label="traning values")
plt.plot(y_test,color="Green",label="testing value")
plt.plot(reg.predict(x_train),color="red",label="predicting vlaue")
plt.legend()
```

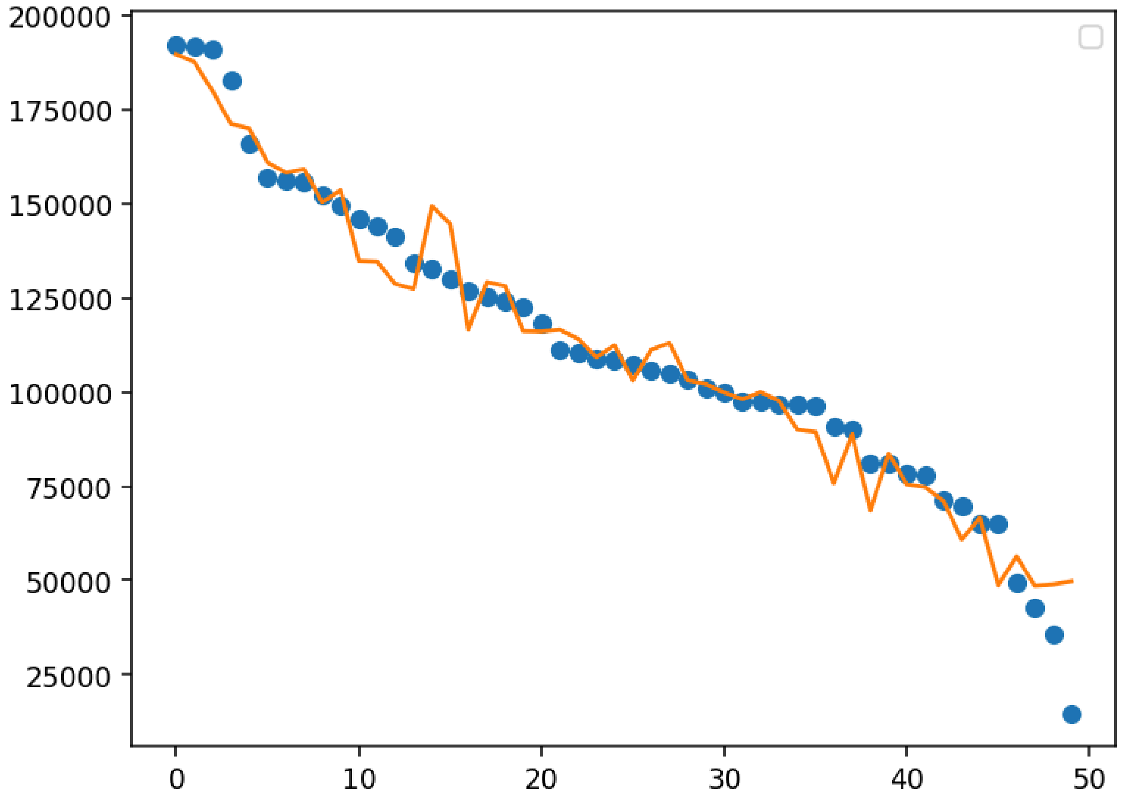
Out[56]: <matplotlib.legend.Legend at 0x7d597ff94b50>



In [ ]:
```
plt.figure(dpi=135)
plt.plot(y,"o")
plt.plot(reg.predict(x))  #best fit line
plt.legend()
```

```
WARNING:matplotlib.legend:No artists with labels found to put in legend.
Note that artists whose label start with an underscore are ignored when le
gend() is called with no argument.
```

Out[57]: <matplotlib.legend.Legend at 0x7d598000e080>



In [ ]:
```
df3=pd.DataFrame()  #need to check
df3["ActualValue"]=y_test
df3["PredictedValue"]=reg.predict(x_test)
print(df3)
```

```
    ActualValue  PredictedValue
0      89949.14    88857.102563
1     108733.99   109299.917912
2      65200.33    66680.290487
3      71498.49    71093.303448
4      42559.73    48588.864362
5     118474.03   116162.086537
6     182901.99   171321.584718
7      99937.59    99971.425331
8     155752.60   159257.651210
9     156122.51   158377.469861
10     81005.76    83684.483035
11    191050.39   179967.050876
12     78239.91    75512.499927
```

In [ ]: