In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.preprocessing import LabelEncoder,OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import statsmodels.api as sm
import statsmodels.regression.linear_model as lm
```

In [2]:

```
data=pd.read_csv("Startups.csv")
data.head(10)
```

Out[2]:

	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

In [3]:

```
real_x=data.iloc[:,0:4].values
real_y=data.iloc[:,4].values
```

In [4]:

```
le=LabelEncoder()
real_x[:,3]=le.fit_transform(real_x[:,3])
oneHE=OneHotEncoder()
real_x=oneHE.fit_transform(real_x).toarray()
```

In [5]:

```
real_x=real_x[:,1:]
```

In [6]:

```
training_x,test_x,training_y,test_y=train_test_split(real_x,real_y,
                                                     test_size=0.2,random_state=
```

In [7]:

```
MLR=LinearRegression()
MLR.fit(training_x,training_y)
```

Out[7]:

```
LinearRegression()
```

In [8]:

```
pred_y=MLR.predict(test_x)
pred_y
```

Out[8]:

```
array([122629.67939821, 103103.18776136, 114542.21097071, 100892.89701
574,
       116688.41484393, 117949.2268177 , 118090.69121311, 117930.63315
589,
       112572.93581677, 116688.41484393])
```

In [9]:

```
test_y
```

Out[9]:

```
array([103282.38, 144259.4 , 146121.95,  77798.83, 191050.39, 105008.3
1,
        81229.06,  97483.56, 110352.25, 166187.94])
```

In [10]:

```
MLR.coef_
```

Out[10]:

```
array([-1.76704566e+04, -1.09433183e+04, -3.34903441e+04, -6.95039891e
+03,
        -2.53951476e+03, -1.18483632e+04, -2.33160709e+04, -1.57955178e
+04,
        -2.86591133e+04, -9.15369154e+03, -1.88504280e+04,  3.16023006e
+03,
         7.12202793e+03, -1.04765256e+03, -6.67528641e+03, -2.69957282e
+03,
        -4.37127551e+02, -1.68249921e+03,  1.38302238e+04, -7.61040835e
+03,
         5.94126455e+03, -6.46567569e+03, -2.68097915e+03, -4.11547903e
+03,
        -2.55228721e+03,  8.45721427e+03, -6.29746175e+03,  1.05841619e
+04,
        -9.85050010e+02,  1.85907781e+04, -3.41489072e+02,  1.47381697e
+04,
         7.05705535e+03,  2.60212330e+02,  3.17759762e+03, -2.14620387e
+03,
         1.12091850e+03,  3.87297451e+03,  8.93643981e+03,  1.66114566e
+04,
         1.30213951e+04,  1.21203047e+04,  1.87323066e+04,  0.00000000e
+00,
         2.07572613e+04,  0.00000000e+00,  3.06221566e+04,  2.38772080e
+04,
        -6.56834010e+03, -3.21507428e+04,  0.00000000e+00, -3.47010858e
+01,
         0.00000000e+00, -1.56363620e+04,  0.00000000e+00,  0.00000000e
+00,
        -2.55551461e+03, -2.89035211e+03,  1.21203047e+04,  0.00000000e
+00,
        -6.61710422e+03, -7.44374578e+02,  1.66114566e+04,  0.00000000e
+00,
         5.04561280e+03,  3.95998711e+03, -8.15684865e+03, -1.68536604e
+04,
        -4.26220951e+04, -6.26599430e+03,  2.07572613e+04,  8.24158899e
+03,
         4.08295776e+03,  0.00000000e+00, -2.23804039e+04,  1.33944421e
+03,
         8.92002490e+03, -2.19604135e+04,  0.00000000e+00, -4.07625461e
+02,
        -2.22302635e+04,  9.82179508e+03,  2.38772080e+04, -3.17573698e
+03,
        -4.20129381e+03,  2.23997585e+03,  1.30213951e+04,  1.87323066e
+04,
         1.13225621e+04,  3.06221566e+04,  0.00000000e+00, -6.00757881e
+03,
         1.86914725e+04, -4.16606798e+03, -1.14384485e+04,  6.02063032e
+03,
        -1.08256879e+03,  0.00000000e+00, -3.51355966e+04, -2.23804039e
+04,
        -1.14384485e+04, -2.19604135e+04, -4.26220951e+04, -4.07625461e
+02,
        0.00000000e+00, -8.15684865e+03, -6.00757881e+03,  0.00000000e
+00,
        1.87323066e+04, -4.20129381e+03, -3.17573698e+03, -2.89035211e
+03,
       -2.55551461e+03,  0.00000000e+00, -6.26599430e+03, -3.47010858e
+01,
        0.00000000e+00, -6.56834010e+03,  1.33944421e+03, -1.56363620e
+04,
       -1.08256879e+03, -6.61710422e+03,  0.00000000e+00,  0.00000000e
+00,
        8.92002490e+03,  9.82179508e+03,  6.02063032e+03,  4.08295776e
+03,
        8.24158899e+03,  2.23997585e+03,  3.95998711e+03, -1.68536604e
+04,
        5.04561280e+03, -4.16606798e+03,  0.00000000e+00, -7.44374578e
+02,
        1.66114566e+04,  1.13225621e+04,  1.30213951e+04,  0.00000000e
+00,
        1.21203047e+04,  0.00000000e+00,  2.07572613e+04,  0.00000000e
+00,
        3.06221566e+04,  2.38772080e+04, -1.53834689e+04,  1.37935585e
+03,
        5.32114698e+03])
```

In [11]:

```
MLR.intercept_
```

Out[11]:

```
115309.05898987639
```

In [12]:

```
real_x = np.append(arr=np.ones((50,1)).astype(int),values=real_x,axis=1)
```

In [13]:

```
 x_opt =  real_x[:,[0,1,2,3,4,5]]
```

In [14]:

```

# OLS=mls.ols( data,endog=real_y,exog=x_opt).fit()
```

In [15]:

```
reg_OLS = sm.OLS(endog = real_y, exog = x_opt).fit()
```

In [16]:

```
reg_OLS.summary()
```

Out[16]:

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.203
Model:	OLS	**Adj. R-squared:**	0.113
Method:	Least Squares	**F-statistic:**	2.244
Date:	Sun, 13 Sep 2020	**Prob (F-statistic):**	0.0665
Time:	14:16:13	**Log-Likelihood:**	-594.98
No. Observations:	50	**AIC:**	1202.
Df Residuals:	44	**BIC:**	1213.
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	1.178e+05	5659.867	20.808	0.000	1.06e+05	1.29e+05
x1	-8.209e+04	3.84e+04	-2.139	0.038	-1.59e+05	-4730.369
x2	-5.284e+04	3.84e+04	-1.377	0.176	-1.3e+05	2.45e+04
x3	-6.828e+04	3.84e+04	-1.779	0.082	-1.46e+05	9086.971
x4	-4.801e+04	3.84e+04	-1.251	0.218	-1.25e+05	2.94e+04
x5	-3.654e+04	3.84e+04	-0.952	0.346	-1.14e+05	4.08e+04

Omnibus:	1.590	**Durbin-Watson:**	0.512
Prob(Omnibus):	0.452	**Jarque-Bera (JB):**	0.795
Skew:	-0.059	**Prob(JB):**	0.672
Kurtosis:	3.606	**Cond. No.**	7.47

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [17]:

```
x_opt =  real_x[:,[0,1,2,3,4,5]]
reg_OLS = sm.OLS(endog = real_y, exog = x_opt).fit()
reg_OLS.summary()
```

Out[17]:

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.203
Model:	OLS	**Adj. R-squared:**	0.113
Method:	Least Squares	**F-statistic:**	2.244
Date:	Sun, 13 Sep 2020	**Prob (F-statistic):**	0.0665
Time:	14:16:13	**Log-Likelihood:**	-594.98
No. Observations:	50	**AIC:**	1202.
Df Residuals:	44	**BIC:**	1213.
Df Model:	5		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	1.178e+05	5659.867	20.808	0.000	1.06e+05	1.29e+05
x1	-8.209e+04	3.84e+04	-2.139	0.038	-1.59e+05	-4730.369
x2	-5.284e+04	3.84e+04	-1.377	0.176	-1.3e+05	2.45e+04
x3	-6.828e+04	3.84e+04	-1.779	0.082	-1.46e+05	9086.971
x4	-4.801e+04	3.84e+04	-1.251	0.218	-1.25e+05	2.94e+04
x5	-3.654e+04	3.84e+04	-0.952	0.346	-1.14e+05	4.08e+04

Omnibus:	1.590	**Durbin-Watson:**	0.512
Prob(Omnibus):	0.452	**Jarque-Bera (JB):**	0.795
Skew:	-0.059	**Prob(JB):**	0.672
Kurtosis:	3.606	**Cond. No.**	7.47

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [18]:

```
x_opt =  real_x[:,[0,1,2,3,4]]
reg_OLS = sm.OLS(endog = real_y, exog = x_opt).fit()
reg_OLS.summary()
```

Out[18]:

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.187
Model:	OLS	**Adj. R-squared:**	0.115
Method:	Least Squares	**F-statistic:**	2.584
Date:	Sun, 13 Sep 2020	**Prob (F-statistic):**	0.0496
Time:	14:16:13	**Log-Likelihood:**	-595.48
No. Observations:	50	**AIC:**	1201.
Df Residuals:	45	**BIC:**	1211.
Df Model:	4		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	1.17e+05	5592.160	20.917	0.000	1.06e+05	1.28e+05
x1	-8.13e+04	3.83e+04	-2.121	0.039	-1.59e+05	-4083.603
x2	-5.205e+04	3.83e+04	-1.358	0.181	-1.29e+05	2.52e+04
x3	-6.748e+04	3.83e+04	-1.760	0.085	-1.45e+05	9733.737
x4	-4.721e+04	3.83e+04	-1.232	0.225	-1.24e+05	3e+04

Omnibus:	1.208	**Durbin-Watson:**	0.460
Prob(Omnibus):	0.547	**Jarque-Bera (JB):**	0.481
Skew:	-0.012	**Prob(JB):**	0.786
Kurtosis:	3.480	**Cond. No.**	7.38

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [19]:

```
x_opt =  real_x[:,[0,1,2,3]]
reg_OLS = sm.OLS(endog = real_y, exog = x_opt).fit()
reg_OLS.summary()
```

Out[19]:

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.159
Model:	OLS	**Adj. R-squared:**	0.105
Method:	Least Squares	**F-statistic:**	2.908
Date:	Sun, 13 Sep 2020	**Prob (F-statistic):**	0.0445
Time:	14:16:13	**Log-Likelihood:**	-596.31
No. Observations:	50	**AIC:**	1201.
Df Residuals:	46	**BIC:**	1208.
Df Model:	3		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	1.16e+05	5563.332	20.845	0.000	1.05e+05	1.27e+05
x1	-8.03e+04	3.85e+04	-2.083	0.043	-1.58e+05	-2710.722
x2	-5.104e+04	3.85e+04	-1.324	0.192	-1.29e+05	2.65e+04
x3	-6.648e+04	3.85e+04	-1.725	0.091	-1.44e+05	1.11e+04

Omnibus:	0.787	**Durbin-Watson:**	0.365
Prob(Omnibus):	0.675	**Jarque-Bera (JB):**	0.207
Skew:	0.028	**Prob(JB):**	0.902
Kurtosis:	3.310	**Cond. No.**	7.30

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [20]:

```
x_opt =  real_x[:,[0,1,3]]
reg_OLS = sm.OLS(endog = real_y, exog = x_opt).fit()
reg_OLS.summary()
```

Out[20]:

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.127
Model:	OLS	**Adj. R-squared:**	0.090
Method:	Least Squares	**F-statistic:**	3.430
Date:	Sun, 13 Sep 2020	**Prob (F-statistic):**	0.0407
Time:	14:16:13	**Log-Likelihood:**	-597.25
No. Observations:	50	**AIC:**	1200.
Df Residuals:	47	**BIC:**	1206.
Df Model:	2		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	1.149e+05	5549.041	20.707	0.000	1.04e+05	1.26e+05
x1	-7.923e+04	3.88e+04	-2.040	0.047	-1.57e+05	-1089.548
x2	-6.541e+04	3.88e+04	-1.684	0.099	-1.44e+05	1.27e+04

Omnibus:	0.458	**Durbin-Watson:**	0.343
Prob(Omnibus):	0.796	**Jarque-Bera (JB):**	0.074
Skew:	0.061	**Prob(JB):**	0.964
Kurtosis:	3.144	**Cond. No.**	7.22

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [21]:

```
x_opt =  real_x[:,[0,1]]
reg_OLS = sm.OLS(endog = real_y, exog = x_opt).fit()
reg_OLS.summary()
```

Out[21]:

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.075
Model:	OLS	**Adj. R-squared:**	0.055
Method:	Least Squares	**F-statistic:**	3.875
Date:	Sun, 13 Sep 2020	**Prob (F-statistic):**	0.0548
Time:	14:16:13	**Log-Likelihood:**	-598.71
No. Observations:	50	**AIC:**	1201.
Df Residuals:	48	**BIC:**	1205.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	1.136e+05	5596.183	20.294	0.000	1.02e+05	1.25e+05
x1	-7.79e+04	3.96e+04	-1.969	0.055	-1.57e+05	1665.637

Omnibus:	0.172	**Durbin-Watson:**	0.216
Prob(Omnibus):	0.918	**Jarque-Bera (JB):**	0.033
Skew:	0.061	**Prob(JB):**	0.984
Kurtosis:	2.966	**Cond. No.**	7.15

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [22]:

```
x_opt =  real_x[:,[0]]
reg_OLS = sm.OLS(endog = real_y, exog = x_opt).fit()
reg_OLS.summary()
```

Out[22]:

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.000
Model:	OLS	**Adj. R-squared:**	0.000
Method:	Least Squares	**F-statistic:**	nan
Date:	Sun, 13 Sep 2020	**Prob (F-statistic):**	nan
Time:	14:16:14	**Log-Likelihood:**	-600.65
No. Observations:	50	**AIC:**	1203.
Df Residuals:	49	**BIC:**	1205.
Df Model:	0		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	1.12e+05	5700.155	19.651	0.000	1.01e+05	1.23e+05

Omnibus:	0.018	**Durbin-Watson:**	0.020
Prob(Omnibus):	0.991	**Jarque-Bera (JB):**	0.068
Skew:	0.023	**Prob(JB):**	0.966
Kurtosis:	2.825	**Cond. No.**	1.00

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.