

Multiple Regression on estimating the profit of Startups:

Importing the libraries

```
In [33]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Importing the data set and EDA process

```
In [2]: data=pd.read_csv('50_Startups.csv')
```

```
In [3]: data
```

```
Out[3]:
```

	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

	R&D Spend	Administration	Marketing Spend	State	Profit
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
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 [4]:

```
for i in data.columns:
    print(i)
```

R&D Spend

Administration
Marketing Spend
State
Profit

```
In [5]: data.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 [6]: X= data.iloc[:, :-1].values
        y=data.iloc[:, -1].values
```

Preprocessing the data

```
In [7]: # No null values
```

```
In [8]: # Categorical??? yes - Encoding
```

```
In [9]: from sklearn.compose import ColumnTransformer
        from sklearn.preprocessing import OneHotEncoder
        ct= ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [3])], remainder='passthru')
```

```
In [10]: X=np.array(ct.fit_transform(X))
         print(X)

[[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]]
```

Splitting the test and train data set

```
In [11]: from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test=train_test_split(X,y, test_size=0.2, random_state=0)
```

Linear Regression Model

```
In [12]: #Multiple regression
from sklearn.linear_model import LinearRegression
regressor=LinearRegression()
regressor.fit(X_train, y_train)
```

```
Out[12]: LinearRegression()
```

Predicting for X_test

```
In [43]: y_predict=regressor.predict(X_test)
print(y_predict)
print(y_test)
```

```
[103015.2 132582.28 132447.74 71976.1 178537.48 116161.24 67851.69
```

```
98791.73 113969.44 167921.07]
[103282.38 144259.4 146121.95 77798.83 191050.39 105008.31 81229.06
97483.56 110352.25 166187.94]
```

In [35]:

```
np.set_printoptions(precision=2)
comparision=pd.DataFrame(np.concatenate((y_predict.reshape(len(y_predict),1),y_test.res
comparision.columns=['Predicted Value', 'Actual Value']
comparision
```

Out[35]:

	Predicted Value	Actual Value
0	103015.201598	103282.38
1	132582.277608	144259.40
2	132447.738452	146121.95
3	71976.098513	77798.83
4	178537.482211	191050.39
5	116161.242302	105008.31
6	67851.692097	81229.06
7	98791.733747	97483.56
8	113969.435330	110352.25
9	167921.065696	166187.94

In [49]:

```
print(regressor.predict([[1,0.0, 1.0,120542.52,148718.95,311613.29]]))
[152874.59]
```

In []:

```
print(regressor.coef_)
print(regressor.intercept_)
```