

# Multiple Linear Regression

- 50 Startup Project

## Importing the Libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

## Import the Dataset

```
dataset=pd.read_csv("50_Startups.csv")
X=dataset.iloc[:, :-1].values
y=dataset.iloc[:, -1].values
```

```
dataset.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

```
print(X)
```

```
[[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']]
```

## Encoding categorical data

Encoding the Independent Variable

```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
ct=ColumnTransformer(transformers=[('encoder',OneHotEncoder(), [3])], remainder='passthrough')
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 dataset into Training and Test set

```
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)
```

## Training the Multiple Linear Regression model on Training set

```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

```
LinearRegression()
```

## Predicting the Test set results

```
y_pred = regressor.predict(X_test)
np.set_printoptions(precision=2)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))
```

```
[[103015.2 103282.38]
 [132582.28 144259.4 ]
 [132447.74 146121.95]
 [ 71976.1  77798.83]
 [178537.48 191050.39]
 [116161.24 105008.31]
 [ 67851.69  81229.06]
 [ 98791.73  97483.56]
 [113969.44 110352.25]
 [167921.07 166187.94]]
```

## Making a single prediction (for example the profit of a startup with R&D Spend = 160000, Administration Spend = 130000, Marketing Spend = 300000 and State = 'California')

```
print(regressor.predict([[1,0,0,160000,130000, 300000]]))

[181566.92]
```

Therefore, our model predicts that the profit of a Californian startup which spent 160000 in R&D, 130000 in Administration and 300000 in Marketing is \$ 181566,92.

**Important note 1:** Notice that the values of the features were all input in a double pair of square brackets. That's because the "predict" method always expects a 2D array as the format of its inputs. And putting our values into a double pair of square brackets makes the input exactly a 2D array. Simply put:

1,0,0,160000,130000,300000→scalars

[1,0,0,160000,130000,300000]→1D array

[[1,0,0,160000,130000,300000]]→2D array

**Important note 2:** Notice also that the "California" state was not input as a string in the last column but as "1, 0, 0" in the first three columns. That's because of course the predict method expects the one-hot-encoded values of the state, and as we see in the second row of the matrix of features X, "California" was encoded as "1, 0, 0". And be careful to include these values in the first three columns, not the last three ones, because the dummy variables are always created in the first columns.

## Getting the final linear regression equation with the values of the coefficients

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

[ 8.66e+01 -8.73e+02  7.86e+02  7.73e-01  3.29e-02  3.66e-02]
42467.52924853204
```

Therefore, the equation of our multiple linear regression model is:

$Profit = (86.6 \times \text{Dummy State 1}) - (873 \times \text{Dummy State 2}) + (786 \times \text{Dummy State 3}) + (0.773 \times \text{R\&D Spend}) + (0.0329 \times \text{Administration}) + (0.0366 \times \text{Marketing Spend}) + (42467.53)$

**Important Note:** To get these coefficients we called the "coef\_" and "intercept\_" attributes from our regressor object. Attributes in Python are different than methods and usually return a simple value or an array of values.