In [10]:

```
import pandas as pd
path="C:\\Users\\TANUJA HARISH\\Desktop\\ML and DL Summer
Internship\\50_Startups.csv"
dataset=pd.read_csv(path)
dataset.head()
```

Out[10]:

	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 [6]:

```
dataset.shape
```

Out[6]:

```
(50, 5)
```

Preparing dataset features and target values¶

In [56]:

```
# sepearte dataset into x and y
import numpy as np
x=np.array(dataset.iloc[:,0:3])
y=np.array(dataset[["Profit"]])
print(x)
print(y)
print(x.shape)
print(y.shape)
```

```
[[165349.2  136897.8  471784.1 ]
 [162597.7  151377.59 443898.53]
 [153441.51 101145.55 407934.54]
 [144372.41 118671.85 383199.62]
 [142107.34  91391.77 366168.42]
 [131876.9   99814.71 362861.36]
 [134615.46 147198.87 127716.82]
 [130298.13 145530.06 323876.68]
 [120542.52 148718.95 311613.29]
 [123334.88 108679.17 304981.62]
 [101913.08 110594.11 229160.95]
 [100671.96  91790.61 249744.55]
 [ 93863.75 127320.38 249839.44]
 [ 91992.39 135495.07 252664.93]
 [119943.24 156547.42 256512.92]
```

```
 [114523.61 122616.84 261776.23]
 [ 78013.11 121597.55 264346.06]
 [ 94657.16 145077.58 282574.31]
 [ 91749.16 114175.79 294919.57]
 [ 86419.7  153514.11      0.  ]
 [ 76253.86 113867.3  298664.47]
 [ 78389.47 153773.43 299737.29]
 [ 73994.56 122782.75 303319.26]
 [ 67532.53 105751.03 304768.73]
 [ 77044.01  99281.34 140574.81]
 [ 64664.71 139553.16 137962.62]
 [ 75328.87 144135.98 134050.07]
 [ 72107.6  127864.55 353183.81]
 [ 66051.52 182645.56 118148.2 ]
 [ 65605.48 153032.06 107138.38]
 [ 61994.48 115641.28  91131.24]
 [ 61136.38 152701.92  88218.23]
 [ 63408.86 129219.61  46085.25]
 [ 55493.95 103057.49 214634.81]
 [ 46426.07 157693.92 210797.67]
 [ 46014.02  85047.44 205517.64]
 [ 28663.76 127056.21 201126.82]
 [ 44069.95  51283.14 197029.42]
 [ 20229.59  65947.93 185265.1 ]
 [ 38558.51  82982.09 174999.3 ]
 [ 28754.33 118546.05 172795.67]
 [ 27892.92  84710.77 164470.71]
 [ 23640.93  96189.63 148001.11]
 [ 15505.73 127382.3   35534.17]
 [ 22177.74 154806.14  28334.72]
 [  1000.23 124153.04   1903.93]
 [  1315.46 115816.21 297114.46]
 [     0.   135426.92      0.  ]
 [   542.05  51743.15      0.  ]
 [     0.   116983.8   45173.06]]
[[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 ]]
(50, 3)
```

```
(50, 1)
```

visualize the data¶

In [18]:

```
import seaborn as sns
sns.pairplot(dataset)
```

Out[18]:

```
<seaborn.axisgrid.PairGrid at 0x1543ce0ff40>
```

Data feature Standadization¶

In [57]:

```
# Using scikit learn lib
#Using StandardScaler
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x_norm=sc.fit_transform(x)
```

```
y_norm=sc.fit_transform(y)
print(x_norm)
print(y_norm)
```

```
[[ 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]]
[[ 2.01120333]
 [ 1.99942997]
 [ 1.98084225]
 [ 1.77662724]
 [ 1.35774012]
 [ 1.12724963]
 [ 1.10548055]
 [ 1.09620987]
 [ 1.00746967]
 [ 0.94602247]
 [ 0.85484675]
 [ 0.80816756]
 [ 0.74115484]
 [ 0.55874952]
 [ 0.51602637]
 [ 0.44871967]
 [ 0.3754357 ]
 [ 0.33477114]
 [ 0.307116  ]
 [ 0.26977265]
 [ 0.16193522]
 [-0.01753384]
 [-0.04161264]
 [-0.08216943]
 [-0.08672946]
 [-0.11549309]
 [-0.15736664]
 [-0.17554233]
```

```
[-0.21879755]
[-0.27588222]
[-0.3026246 ]
[-0.36412744]
[-0.36552389]
[-0.38178711]
[-0.38344421]
[-0.38929092]
[-0.53393161]
[-0.5529549 ]
[-0.77149734]
[-0.77709368]
[-0.84641135]
[-0.85746568]
[-1.01536466]
[-1.05896021]
[-1.17320899]
[-1.18008224]
[-1.56692212]
[-1.74062718]
[-1.91321197]
[-2.43931323]]
```

Split data into train and test¶

In [60]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x_norm,y_norm,test_size=0.2)
print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)
```

```
(40, 3)
(40, 1)
(10, 3)
(10, 1)
```

Import the algorithm¶

In [106]:

```
from sklearn.neural_network import MLPRegressor
MLPregressor=MLPRegressor(hidden_layer_sizes=(5),max_iter=100,activation="r
elu",solver="adam",random_state=42)
MLPregressor.fit(x_train,y_train)
```

```
C:\Users\TANUJA
HARISH\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptr
on.py:1599: DataConversionWarning: A column-vector y was passed when a 1d
array was expected. Please change the shape of y to (n_samples, ), for example
using ravel().
  y = column_or_1d(y, warn=True)
C:\Users\TANUJA
HARISH\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptr
on.py:692: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (100)
reached and the optimization hasn't converged yet.
  warnings.warn(
```

Out[106]:

```
MLPRegressor(hidden_layer_sizes=5, max_iter=100, random_state=42)
```

Prediction¶

In [107]:

```
y_pred=MLPregressor.predict(x_test)
y_pred=y_pred.reshape(-1,1)
y_pred=sc.inverse_transform(y_pred)
print(y_pred.shape)
print(y_test.shape)
conc=np.c_[y_test,y_pred]
print(conc)
```

```
(10, 1)
(10, 1)
[[-8.46411346e-01  1.24966353e+05]
 [-8.21694292e-02  1.18282540e+05]
 [-5.52954899e-01  1.05599578e+05]
 [-3.83444211e-01  1.20962778e+05]
 [-1.57366637e-01  1.21109933e+05]
 [ 1.35774012e+00  6.38071997e+04]
 [-5.33931605e-01  1.27028218e+05]
 [ 2.01120333e+00  5.53964118e+04]
 [ 1.12724963e+00  7.36476955e+04]
 [-7.77093678e-01  1.31699249e+05]]
```

Calculating error¶

In [108]:

```
from sklearn.metrics import mean_squared_error,mean_absolute_error
from math import sqrt
print(np.sqrt(mean_squared_error(y_test,y_pred)))
print(mean_absolute_error(y_test,y_pred))
```

```
107751.31099362503
104249.87919791148
```

In [ ]:

In [ ]:

In [ ]:

In [ ]: