INTELLIGENT VECHICLE DAMAGE ASSESSMENT ANDCOST ESTIMATORFOR INSURANCE COMPANIES

Team member

Name	Register number
Team leader :VENUGOPAL. M	623119104031
Team Member:KAVINKUMAR. R	623119104008
Team Member:SIVAKUMAR. R	623119104024
Team Member:PRAVEENKUMAR. S	623119104020

```
   R&D Spend  ...     Profit
0  165349.20  ...  192261.83
1  162597.70  ...  191792.06
2  153441.51  ...  191050.39
3  144372.41  ...  182901.99
4  142107.34  ...  166187.94

[5 rows x 5 columns]
> 
```

Numerical/Statistical analysis of the dataset

```
dataset.describe()
```

Output:

	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

Dimensions of dataset

```
print('There are ',dataset.shape[0],'
```

Output:

```
There are  50 rows and  5 columns in
```

Here we are trying to check if there are repeated values in the dataset or not.

```
print('There are',dataset.duplicated(
```

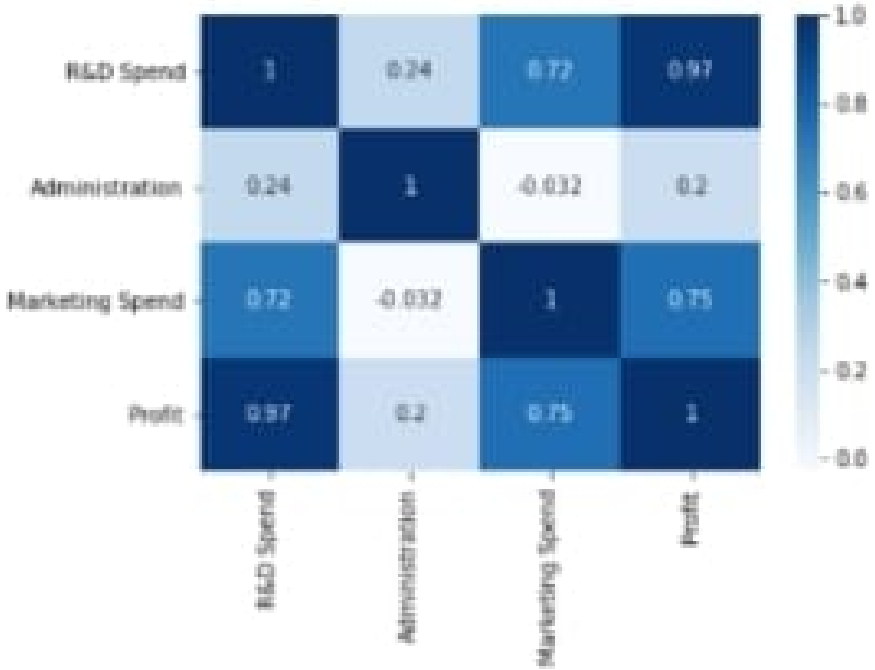
Output:

```
There are no repeating values in the
```

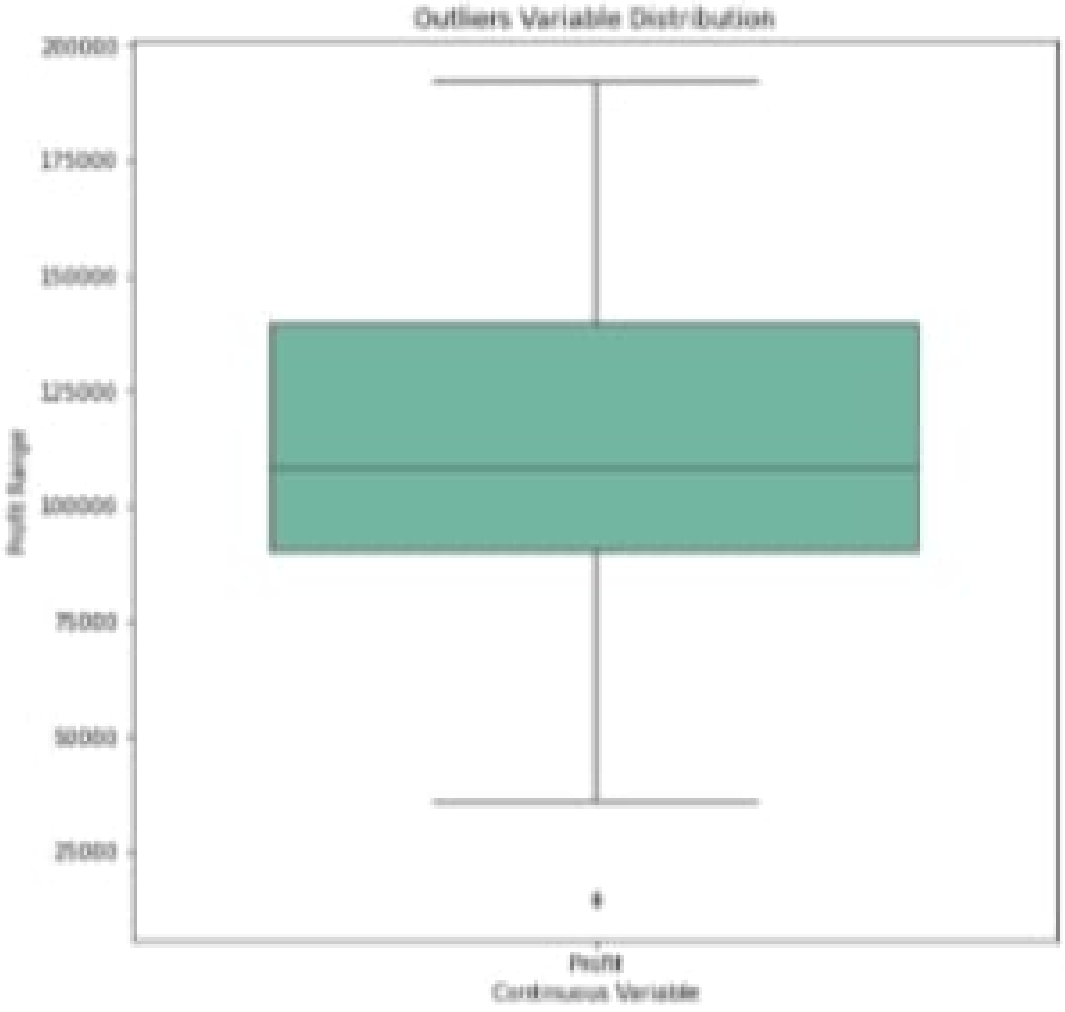
```
dataset.isnull().sum()
```

Output:

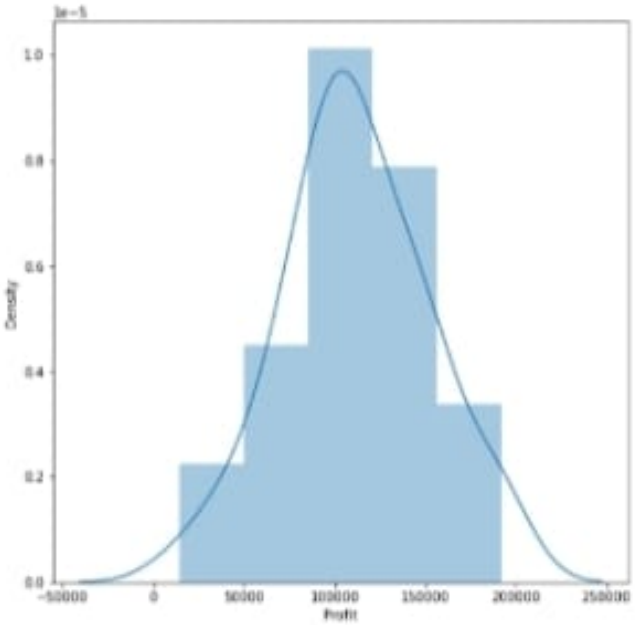
```
R&D Spend          0
Administration     0
Marketing Spend    0
State              0
Profit             0
dtype: int64
```


R&D Spend
Administration
Marketing Spend
Profit
1
0.24
0.72
0.97
0.24
1
-0.032
0.2
0.72
-0.032
1
0.75
0.97
0.2
0.75
1
R&D Spend
Administration
Marketing Spend
Profit
1.0
0.8
0.6
0.4
0.2
0.0

A	B	C	D	E
R&D Spend	**Administration**	**Marketing Spend**	**State**	**Profit**
165349.2	136897.8	471784.1	New York	192261.83
162597.7	151377.59	443898.53	California	191792.06
153441.51	101145.55	407934.54	Florida	191050.39
144372.41	118671.85	383199.62	New York	182901.99
142107.34	91391.77	366168.42	Florida	166187.94
131876.9	99814.71	362861.36	New York	156991.12
134615.46	147198.87	127716.82	California	156122.51
130298.13	145530.06	323876.68	Florida	155752.6
120542.52	148718.95	311613.29	New York	152211.77
123334.88	108679.17	304981.62	California	149759.96
101913.08	110594.11	229160.95	Florida	146121.95
100671.96	91790.61	249744.55	California	144259.4


Outliers Variable Distribution
200000
175000
150000
125000
100000
75000
50000
25000
Profit Range
Profit
Continuous Variable

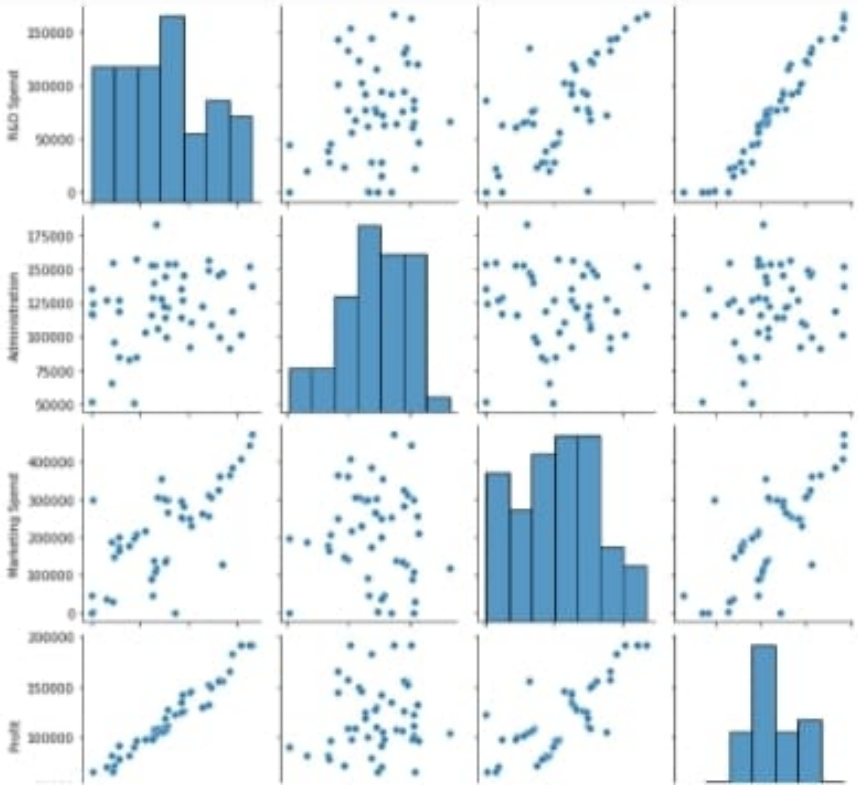
	R&D Spend	Administration	Marketing Spend	Profit
count	50.000000	50.000000	50.000000	50.000000
mean	73721.615600	121344.639600	211025.097800	112012.639200
std	45902.256482	28017.802755	122290.310726	40306.180338
min	0.000000	51283.140000	0.000000	14681.400000
25%	39936.370000	103730.875000	129300.132500	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



Inference: The average profit (which is 100k) is the most frequent i.e. this should be in the category of distribution plot.

```
sns.pairplot(dataset)
plt.show()
```

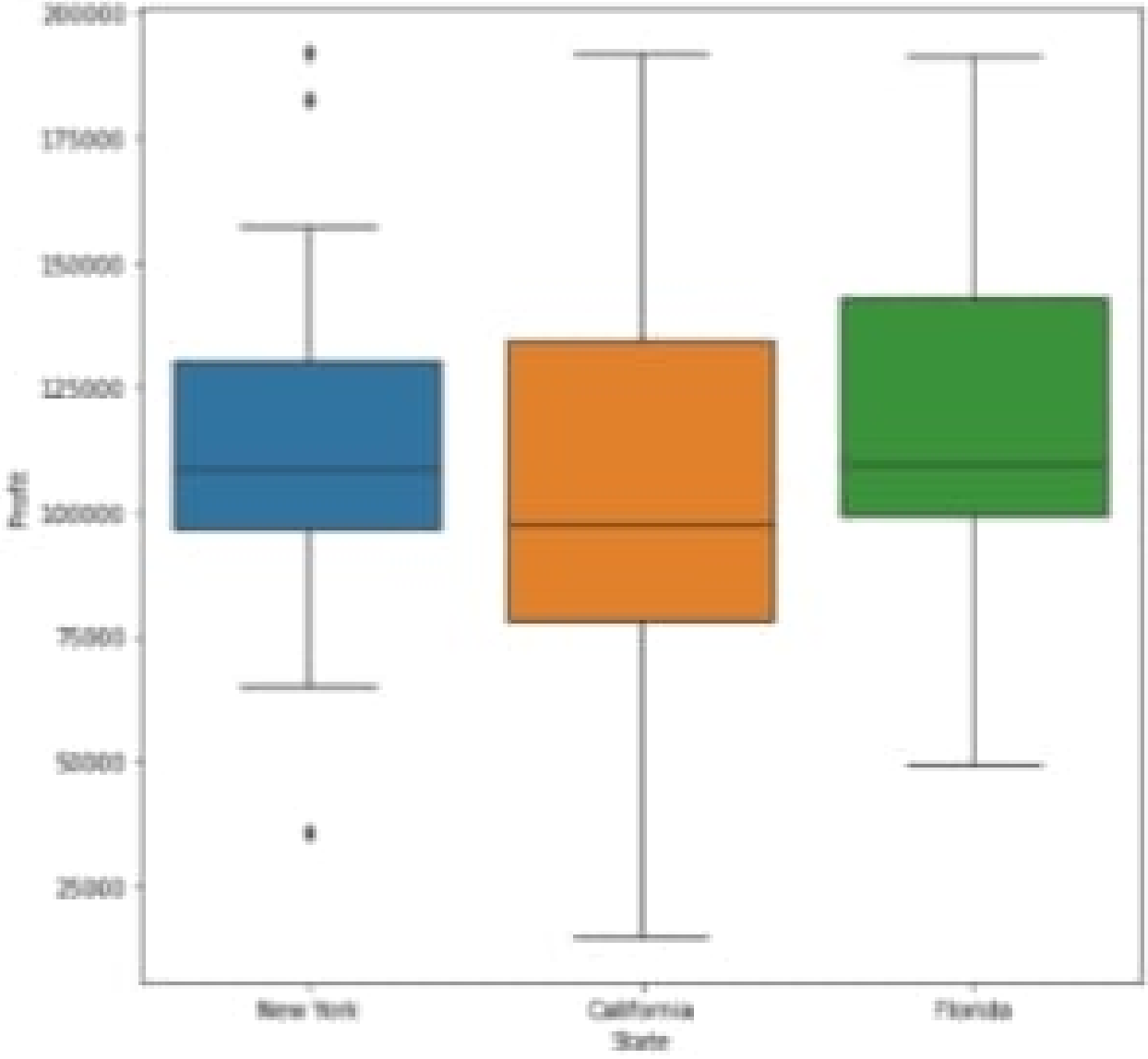
Output:



	Predicted value	Actual Value
0	104055.184238	103282.38
1	132557.602897	144259.40
2	133633.012845	146121.95
3	72336.280811	77798.83
4	179658.272109	191050.39
5	114889.631334	105008.31
6	66514.822490	81229.06
7	98461.693213	97483.56
8	114294.704870	110352.25
9	169090.511275	166187.94
10	96281.907934	96778.92
11	88108.300579	96479.51
12	110687.117232	105733.54
13	90536.342031	96712.80
14	127785.379386	124266.90

Inference :

As we can see that the **predicted value is close to the actual values** i.e the one present in the testing set, **Hence we can use this model for prediction**. But first, we need to calculate how much is the error generated.


200000
175000
150000
125000
100000
75000
50000
25000
Profit
New York
California
Florida
State

```
rray([[130298.13, 145530.06, 323876.6
       [119943.24, 156547.42, 256512.9
       [1000.23, 124153.04, 1903.93, 2
       [542.05, 51743.15, 0.0, 2],
       [65605.48, 153032.06, 107138.38
       [114523.61, 122616.84, 261776.2
       [61994.48, 115641.28, 91131.24,
       [63408.86, 129219.61, 46085.25,
       [78013.11, 121597.55, 264346.06
       [23640.93, 96189.63, 148001.11,
       [76253.86, 113867.3, 298664.47,
       [15505.73, 127382.3, 35534.17,
       [120542.52, 148718.95, 311613.2
       [91992.39, 135495.07, 252664.93
       [64664.71, 139553.16, 137962.62
       [131876.9, 99814.71, 362861.36,
       [94657.16, 145077.58, 282574.31
       [28754.33, 118546.05, 172795.67
       [0.0, 116983.8, 45173.06, 0],
       [162597.7, 151377.59, 443898.53
       [93863.75, 127320.38, 249839.44
       [44069.95, 51283.14, 197029.42,
       [77044.01, 99281.34, 140574.81,
       [134615.46, 147198.87, 127716.8
       [67532.53, 105751.03, 304768.73
       [28663.76, 127056.21, 201126.82
       [78389.47, 153773.43, 299737.29
       [86419.7, 153514.11, 0.0, 2],
       [123334.88, 108670.17, 304081.4
```

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

From the **corr function**, we can find the correlation between the columns.

```
c = dataset.corr()
c
```

Output:

	R&D Spend	Administration	Marketing Spend	Profit
R&D Spend	1.000000	0.241955	0.724248	0.972900
Administration	0.241955	1.000000	-0.032154	0.200717
Marketing Spend	0.724248	-0.032154	1.000000	0.747766
Profit	0.972900	0.200717	0.747766	1.000000

```
y_pred = model.predict(x_test)
y_pred
```

Output:

```
array([104055.1842384 , 132557.6028970
       179658.27210893, 114689.6313339
       114294.70487032, 169090.5112746
       110687.1172322 ,  90536.3420308
```

Testing scores

```
testing_data_model_score = model.score
print("Model Score/Performance on Test

training_data_model_score = model.scor
print("Model Score/Performance on Trai
```

Output:

```
Model Score/Performance on Testing dat
Model Score/Performance on Training da
```

```
labelencoder = LabelEncoder()
X[:, 3] = labelencoder.fit_transform(X
X1 = pd.DataFrame(X)
X1.head()
```

Output:

	0	1	2	3
0	165349	136898	471784	2
1	162598	151378	443899	0
2	153442	101146	407935	1
3	144372	118672	383200	2
4	142107	91391.8	366168	1


Thank
you!