REGRESSION

Regression analysis is a statistical method to find the relationship between target variables and predictor variables. It helps to understand how the dependent variable is changing corresponding to the independent variables. It mainly predict the continuous variable such as temperature,age,salary etc.

It is a supervised machine learning which helps to find the correlation between variables. It also help to predict the continuous variables with respect to more than one or more independent variables.It is mainly used for prediction,forecasting,time series etc.

" Regression shows a line or curve that passes through all the points on both target and predicted variables in such a way that the distance between the points and the regression line is minimum"

There are various types of regression

- Linear Regression
- Logistic Regression
- Polynomial Regression
- Support Vector Regression
- Decision Tree Regression
- Random Forest Regression
- Ridge Regression
- Lasso Regression

LINEAR REGRESSION

Linear regression is the simple and easy algorithm used for predictive analysis.It shows linear relationship between the independent variable and the dependent variable.

If there is only one input variable(x), then such linear regression is called simple linear regression.

The mathematical equation for simple linear regression is given by:

$$Y = aX + b + \epsilon$$

Where Y=Target variable
X=Predicted variable

a=Slope
b=intercept
Ɛ=random error

A linear line showing relationship between the dependent variable and independent variable is called regression line. If both the variable increases then the corresponding line is called positive linear relationship. If dependent variable increases and independent variable decrease then the line is called negative linear relationship.

In Regression , their will be more line so we have to find the best fit line such that the difference between actual value and predicted value is minimum.

For Linear Regression, we use the **Mean Squared Error (MSE)** cost function, which is the average of squared error occurred between the predicted values and actual values.

Residuals

The distance between the actual value and predicted values is called residual. If the observed points are far from the regression line, then the residual will be high, and so cost function will high. If the scatter points are close to the regression line, then the residual will be small and hence the cost function.

There are some assumptions for linear regression

1. Linear relationship between target and predicted variable.
2. No multicollinearity.
3. Homoscedasticity Assumption.
4. No autocorrelation
5. Random error should follow normal distribution.

MULTIPLE LINEAR REGRESSION

If there is more than one input variables, then such linear regression is called multiple linear regression. In Multiple Linear Regression, the target variable(Y) is a linear combination of multiple predictor variables x_1, x_2, x_3, ...,x_n.

The mathematical equation corresponding to MLR is given by:

$$yi=\beta0+\beta1xi,1+\beta2xi,2+\ldots+\beta kxi,k+\epsilon i.$$

where yi = dependent variables

xi=independent variables

βi=coefficients corresponding to independent variables.

Ɛi=Random error.

In the following sections, we will implement MLR using Python.

Problem description:

We have a dataset of 50 start-up companies. This dataset contains five main information: R&D Spend, Administration Spend, Marketing Spend, State, and Profit for a financial year. The aim of the company is to create a model that can determine which company has a maximum profit, and which is the most affecting factor for the profit of a company.

Here we take profit as dependent variable and other datas as independent variables. The main steps for implementing data in machine learning are:

1. Getting the dataset.
2. Importing libraries.
3. Importing dataset.
4. Finding missing data.
5. Encoding Categorical data.
6. Splitting datasets into training and test data.
7. Feature Scaling.

In MLR, Feature Scaling is not done manually because it is done by library itself.

Step-1:Getting the datasets

The data is collected from the secondary source in 'CSV' formatted files.

Step-2: Importing libraries

Firstly we will import the library which will help in building the model.The code is given below

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

Step-3: Importing datasets

We will import datasets of 50 companies using the code pd.read_csv.

```
data=pd.read_csv('/content/drive/MyDrive/archive (6).zip')
```

The output will be as follows:

Unnamed: 0	R&D Spend	Administration	Marketing Spend	State	Profit	
0	0	165349.30	136897.90	471784.20	New York	192261.93
1	1	162597.80	151377.69	443898.63	California	191792.16
2	2	153441.61	101145.65	407934.64	Florida	191050.49
3	3	144372.51	118671.95	383199.72	New York	182902.09
4	4	142107.44	91391.87	366168.52	Florida	166188.04
5	5	131877.00	99814.81	362861.46	New York	156991.22
6	6	134615.56	147198.97	127716.92	California	156122.61
7	7	130298.23	145530.16	323876.78	Florida	155752.70
8	8	120542.62	148719.05	311613.39	New York	152211.87
9	9	123334.98	108679.27	304981.72	California	149760.06
10	10	101913.18	110594.21	229161.05	Florida	146122.05
11	11	100672.06	91790.71	249744.65	California	144259.50
12	12	93863.85	127320.48	249839.54	Florida	141585.62
13	13	91992.49	135495.17	252665.03	California	134307.45
14	14	119943.34	156547.52	256513.02	Florida	132602.75
15	15	114523.71	122616.94	261776.33	New York	129917.14
16	16	78013.21	121597.65	264346.16	California	126993.03
17	17	94657.26	145077.68	282574.41	New York	125370.47
18	18	91749.26	114175.89	294919.67	Florida	124267.00
19	19	86419.80	153514.21	0.10	New York	122776.96
20	20	76253.96	113867.40	298664.57	California	118474.13

Unnamed: 0	R&D Spend	Administration	Marketing Spend	State	Profit	
21	21	78389.57	153773.53	299737.39	New York	111313.12
22	22	73994.66	122782.85	303319.36	Florida	110352.35
23	23	67532.63	105751.13	304768.83	Florida	108734.09
24	24	77044.11	99281.44	140574.91	New York	108552.14
25	25	64664.81	139553.26	137962.72	California	107404.44
26	26	75328.97	144136.08	134050.17	Florida	105733.64
27	27	72107.70	127864.65	353183.91	New York	105008.41
28	28	66051.62	182645.66	118148.30	Florida	103282.48
29	29	65605.58	153032.16	107138.48	New York	101004.74
30	30	61994.58	115641.38	91131.34	Florida	99937.69
31	31	61136.48	152702.02	88218.33	New York	97483.66
32	32	63408.96	129219.71	46085.35	California	97427.94
33	33	55494.05	103057.59	214634.91	Florida	96779.02
34	34	46426.17	157694.02	210797.77	California	96712.90
35	35	46014.12	85047.54	205517.74	New York	96479.61
36	36	28663.86	127056.31	201126.92	Florida	90708.29
37	37	44070.05	51283.24	197029.52	California	89949.24
38	38	20229.69	65948.03	185265.20	New York	81229.16
39	39	38558.61	82982.19	174999.40	California	81005.86
40	40	28754.43	118546.15	172795.77	California	78240.01
41	41	27893.02	84710.87	164470.81	Florida	77798.93
42	42	23641.03	96189.73	148001.21	California	71498.59
43	43	15505.83	127382.40	35534.27	New York	69759.08
44	44	22177.84	154806.24	28334.82	California	65200.43
45	45	1000.33	124153.14	1904.03	New York	64926.18
46	46	1315.56	115816.31	297114.56	Florida	49490.85

	Unnamed: 0	R&D Spend	Administration	Marketing Spend	State	Profit
47	47	0.10	135427.02	0.10	California	42559.83
48	48	542.15	51743.25	0.10	New York	35673.51
49	49	0.10	116983.90	45173.16	California	14681.50

The table information is taken by the command data.info().The output is

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 6 columns):

#	Column	Non-Null Count	Dtype
0	Unnamed: 0	50 non-null	int64
1	R&D Spend	50 non-null	float64
2	Administration	50 non-null	float64
3	Marketing Spend	50 non-null	float64
4	State	50 non-null	object
5	Profit	50 non-null	float64

dtypes: float64(4), int64(1), object(1)

memory usage: 2.5+ KB

From this we get that this dataframe contain 50 entries and 6 variables. Among the variables,4 are in float datatypes,1 each in integer and object data type.

The statistical description of the above data is found by data.describe().

	Unnamed: 0	R&D Spend	Administration	Marketing Spend	Profit
count	50.00000	50.000000	50.000000	50.000000	50.000000
mean	24.50000	73721.715600	121344.739600	211025.197800	112012.739200
std	14.57738	45902.256482	28017.802755	122290.310726	40306.180338
min	0.00000	0.100000	51283.240000	0.100000	14681.500000
25%	12.25000	39936.470000	103730.975000	129300.232500	90139.002500

	Unnamed: 0	R&D Spend	Administration	Marketing Spend	Profit
50%	24.50000	73051.180000	122699.895000	212716.340000	107978.290000
75%	36.75000	101602.900000	144842.280000	299469.185000	139766.077500
max	49.00000	165349.300000	182645.660000	471784.200000	192261.930000

Step-4: Finding missing data

The missing data is find by the command data.isna().sum()

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

Here the data doesn't contain any missing values from the above table.

We have to find which company has maximum profit with respect to other variables so profit is taken as the dependent variables and other as the independent variables. The extraction of target and predicted variable is given by

```
y=data[['Profit']].values
x=data[['R&D Spend','Administration','Marketing Spend','State']].values
y
```

```
array([[192261.93],
       [191792.16],
       [191050.49],
       [182902.09],
       [166188.04],
       [156991.22],
       [156122.61],
       [155752.7 ],
       [152211.87],
       [149760.06],
       [146122.05],
       [144259.5 ],
       [141585.62],
       [134307.45],
```

```
       [132602.75],
       [129917.14],
       [126993.03],
       [125370.47],
       [124267.  ],
       [122776.96],
       [118474.13],
       [111313.12],
       [110352.35],
       [108734.09],
       [108552.14],
       [107404.44],
       [105733.64],
       [105008.41],
       [103282.48],
       [101004.74],
       [ 99937.69],
       [ 97483.66],
       [ 97427.94],
       [ 96779.02],
       [ 96712.9 ],
       [ 96479.61],
       [ 90708.29],
       [ 89949.24],
       [ 81229.16],
       [ 81005.86],
       [ 78240.01],
       [ 77798.93],
       [ 71498.59],
       [ 69759.08],
       [ 65200.43],
       [ 64926.18],
       [ 49490.85],
       [ 42559.83],
       [ 35673.51],
       [ 14681.5 ]])
```

Step 5-Encoding categorical data

We use one-hot encoding when there are categorical values in our dataset. Here for us, there is a state column that is categorical, so we have to use one-hot encoding to convert them.So, import One-HotEncoder from scikit learn library.

```
from sklearn.preprocessing import OneHotEncoder
ohe=OneHotEncoder(sparse=False)
x=ohe.fit_transform(data[['State']])
x
```

```
array([[0., 0., 1.],
       [1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.],
```

```
[0., 1., 0.],
[0., 0., 1.],
[1., 0., 0.],
[0., 1., 0.],
[0., 0., 1.],
[1., 0., 0.],
[0., 1., 0.],
[1., 0., 0.],
[0., 1., 0.],
[1., 0., 0.],
[0., 1., 0.],
[0., 0., 1.],
[1., 0., 0.],
[0., 0., 1.],
[0., 1., 0.],
[0., 0., 1.],
[1., 0., 0.],
[0., 0., 1.],
[0., 1., 0.],
[0., 1., 0.],
[0., 0., 1.],
[1., 0., 0.],
[0., 1., 0.],
[0., 0., 1.],
[0., 1., 0.],
[0., 0., 1.],
[0., 1., 0.],
[0., 0., 1.],
[1., 0., 0.],
[0., 1., 0.],
[1., 0., 0.],
[0., 0., 1.],
[0., 1., 0.],
[1., 0., 0.],
[0., 0., 1.],
[1., 0., 0.],
[1., 0., 0.],
[0., 1., 0.],
[1., 0., 0.],
[0., 0., 1.],
[1., 0., 0.],
[0., 0., 1.],
[0., 1., 0.],
[1., 0., 0.],
[0., 0., 1.],
[1., 0., 0.]])
```

Step-7:Splitting the dataset into training and test datasets

Now we will split the dataset into training and test set. The code for this is given below:

```
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)
x_train,x_test,y_train,y_test
```

The above code will split the dataset into training and test data.the output will be

```
(array([[55494.05, 103057.59, 214634.91, 1],
        [46014.12, 85047.54000000001, 205517.74, 2],
        [75328.97, 144136.08000000002, 134050.17, 1],
        [46426.17, 157694.02000000002, 210797.77, 0],
        [91749.26, 114175.89, 294919.67, 1],
        [130298.23, 145530.16, 323876.78, 1],
        [119943.34, 156547.52000000002, 256513.02, 1],
        [1000.33, 124153.14, 1904.03, 2],
        [542.15, 51743.25, 0.1, 2],
        [65605.58, 153032.16, 107138.48, 2],
        [114523.71, 122616.94, 261776.33, 2],
        [61994.58, 115641.38, 91131.34, 1],
        [63408.96, 129219.71, 46085.35, 0],
        [78013.21, 121597.65, 264346.16, 0],
        [23641.03, 96189.73, 148001.21, 0],
        [76253.96, 113867.4, 298664.56999999995, 0],
        [15505.83, 127382.4, 35534.27, 2],
        [120542.62, 148719.05000000002, 311613.39, 2],
        [91992.49, 135495.17, 252665.03, 0],
        [64664.81, 139553.26, 137962.72, 0],
        [131877.0, 99814.81, 362861.46, 2],
        [94657.26, 145077.68, 282574.41, 2],
        [28754.43, 118546.15, 172795.77000000002, 0],
        [0.1, 116983.9, 45173.16, 0],
        [162597.80000000002, 151377.69, 443898.63, 0],
        [93863.85, 127320.48, 249839.54, 1],
        [44070.05, 51283.24, 197029.52, 0],
        [77044.11, 99281.44, 140574.91, 2],
        [134615.56, 147198.97, 127716.92, 0],
        [67532.63, 105751.13, 304768.83, 1],
        [28663.86, 127056.31, 201126.92, 1],
        [78389.57, 153773.53, 299737.39, 2],
        [86419.8, 153514.21, 0.1, 2],
        [123334.98, 108679.27, 304981.72, 0],
        [38558.61, 82982.19, 174999.4, 0],
        [1315.56, 115816.31, 297114.56, 1],
        [144372.51, 118671.95, 383199.72, 2],
        [165349.30000000002, 136897.9, 471784.2, 2],
        [0.1, 135427.02000000002, 0.1, 0],
        [22177.84, 154806.24000000002, 28334.82, 0]], dtype=object),
 array([[66051.62000000001, 182645.66, 118148.3, 1],
        [100672.06, 91790.71, 249744.65, 0],
        [101913.18, 110594.21, 229161.05, 1],
        [27893.02, 84710.87000000001, 164470.81, 1],
        [153441.61000000002, 101145.65, 407934.64, 1],
        [72107.70000000001, 127864.65, 353183.91, 2],
        [20229.69, 65948.03, 185265.2, 2],
        [61136.48, 152702.02000000002, 88218.33, 2],
        [73994.66, 122782.85, 303319.36, 1],
```

```
       [142107.44, 91391.87, 366168.52, 1]], dtype=object),
array([[ 96779.02],
       [ 96479.61],
       [105733.64],
       [ 96712.9 ],
       [124267.  ],
       [155752.7 ],
       [132602.75],
       [ 64926.18],
       [ 35673.51],
       [101004.74],
       [129917.14],
       [ 99937.69],
       [ 97427.94],
       [126993.03],
       [ 71498.59],
       [118474.13],
       [ 69759.08],
       [152211.87],
       [134307.45],
       [107404.44],
       [156991.22],
       [125370.47],
       [ 78240.01],
       [ 14681.5 ],
       [191792.16],
       [141585.62],
       [ 89949.24],
       [108552.14],
       [156122.61],
       [108734.09],
       [ 90708.29],
       [111313.12],
       [122776.96],
       [149760.06],
       [ 81005.86],
       [ 49490.85],
       [182902.09],
       [192261.93],
       [ 42559.83],
       [ 65200.43]]),
array([[103282.48],
       [144259.5 ],
       [146122.05],
       [ 77798.93],
       [191050.49],
       [105008.41],
       [ 81229.16],
       [ 97483.66],
       [110352.35],
       [166188.04]]))
```

In MLR,step 7 is not important so next we proceed to fit the model. Now we have a well prepared datasets inorder to provide training we will fit the regression model to the training set. The code for this

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

Now, we have successfully trained our model using the training dataset. In the next step, we will test the performance of the model using the test dataset.To test the performance of the dataset,we will do it by predicting the test result.For prediction we will use y_pred .

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

```
array([[110559.165   ],
       [101383.13625 ],
       [110559.165   ],
       [110559.165   ],
       [110559.165   ],
       [117867.14714286],
       [117867.14714286],
       [117867.14714286],
       [110559.165   ],
       [110559.165   ]])
```

Evaluate the model

To find the accuracy of the model and we use r2_score and is calculated by using the below code:

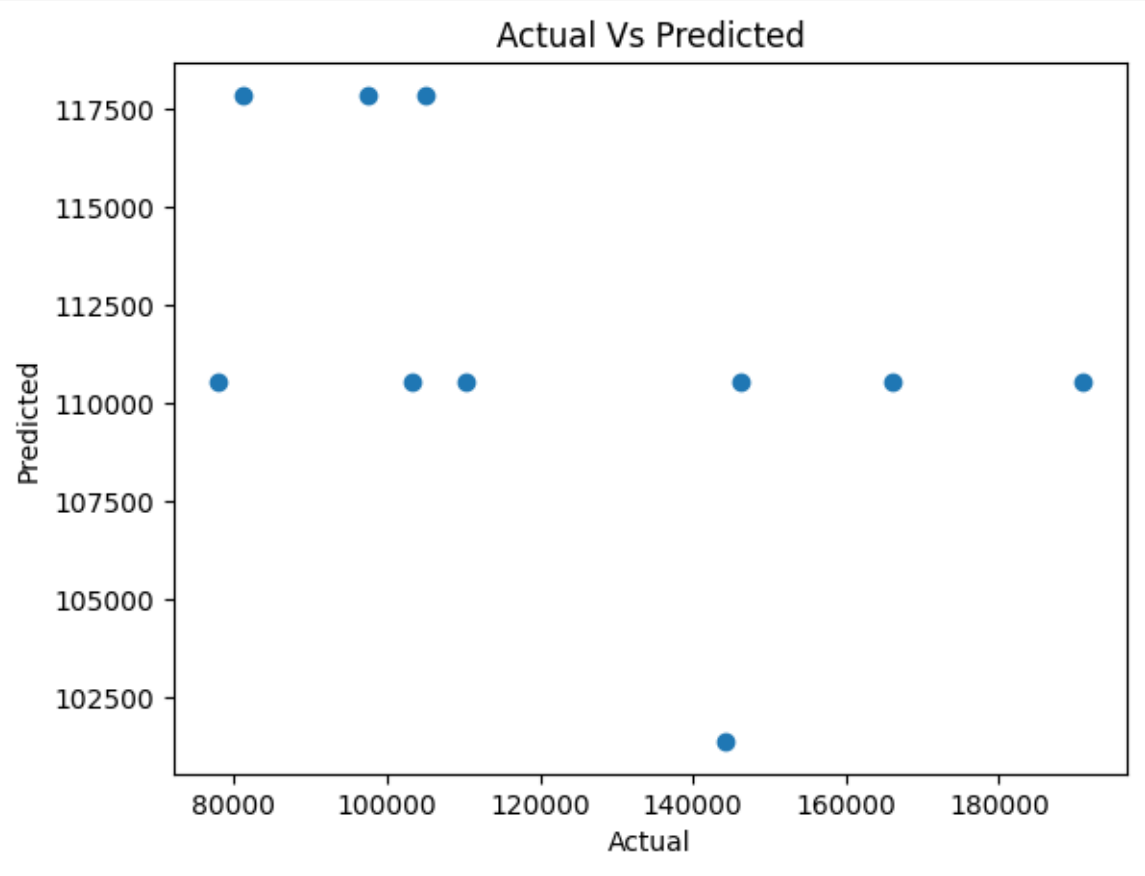
```
from sklearn.metrics import r2_score
Accuracy=r2_score(y_test,y_pred)*100
print(" Accuracy of the model is:",Accuracy)
```

Accuracy of the model is: -22.965294816117066

Plot the results:

We will plot the scatter plot between actual values and predicted values. Use xlabel to label the x-axis and use ylabel to label the y-axis.

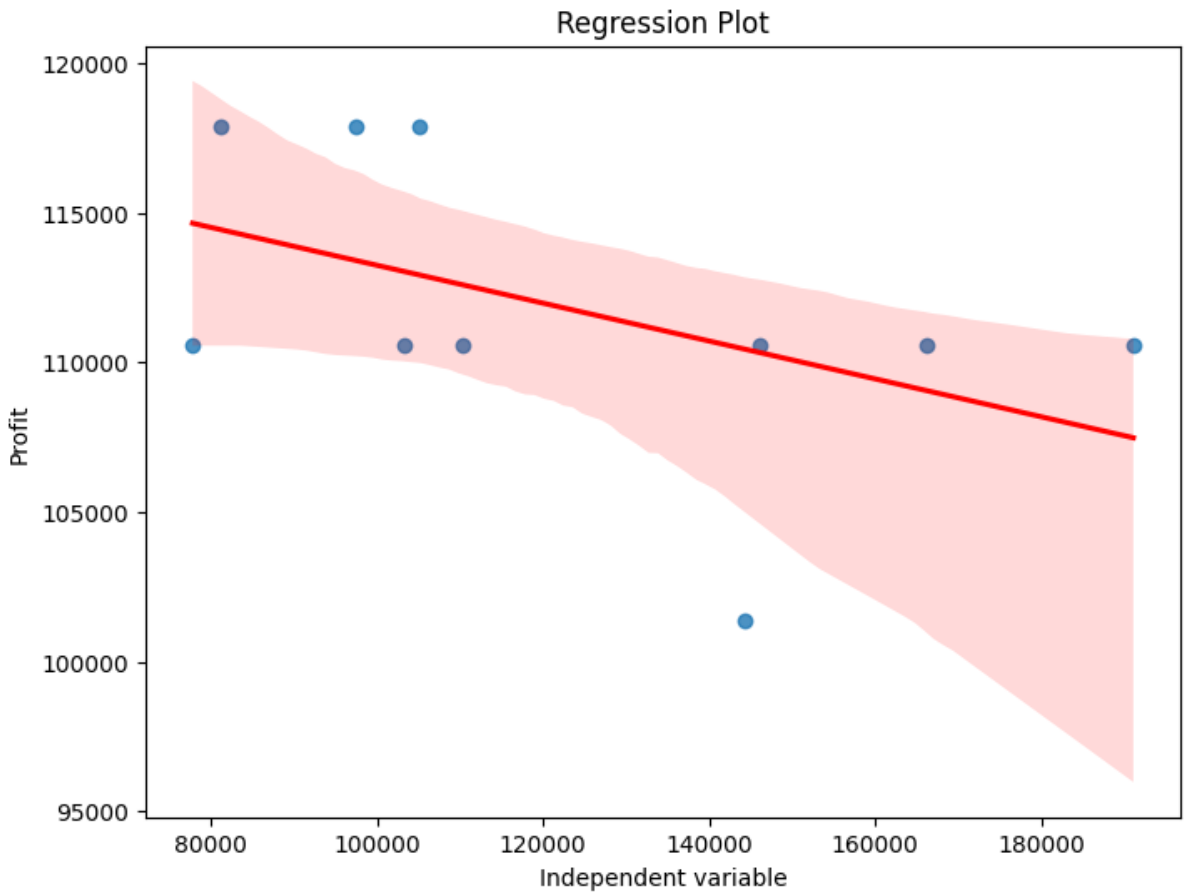
```
plt.scatter(y_test,y_pred)
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.title('Actual Vs Predicted')
```



Regression plot

A regression plot is useful to understand the linear relationship between two parameters. It creates a regression line in-between those parameters and then plots a scatter plot of those data points.

```
plt.figure(figsize=(8, 6))
sns.regplot(x=y_test, y=y_pred,line_kws={'color': 'red'})
plt.title('Regression Plot')
plt.xlabel('Independent variable')
plt.ylabel('Profit')
plt.show()
```



From the graph, it can be say that the data are slightly scattered from the regression line. The regression line is falling downwards.

Predicted values

Create a new data frame that contains actual values, predicted values, and differences between them so that we will understand how near the model predicts its actual value.

```
pred_df=pd.DataFrame({'Actual Value':y_test,'Predicted Value':y_pred,'Difference':y_test-y_pred})
pred_df
```

Actual Value	Predicted Value	Difference	
28	103282.48	110559.165000	-7276.685000

Actual Value	Predicted Value	Difference	
11	144259.50	101383.136250	42876.363750
10	146122.05	110559.165000	35562.885000
41	77798.93	110559.165000	-32760.235000
2	191050.49	110559.165000	80491.325000
27	105008.41	117867.147143	-12858.737143
38	81229.16	117867.147143	-36637.987143
31	97483.66	117867.147143	-20383.487143
22	110352.35	110559.165000	-206.815000
4	166188.04	110559.165000	55628.875000

We can see that the difference between actual values and predicted values are not very high.

Conclusion

In this article we have discussed about different types of regression .A datasets of 50 companies are fitted for multiple linear regression .In that set,we have

cleaned the data and we learned how to perform One-Hot Encoding and where to perform it. We trained the model, predicted the results, evaluated the model using r2_score metrics, and plotted the results.