Practical 3

Aim:To build the model for prediction of profit

Importing the libraries

In [39]:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
```

Reading the dataset and taking into dataframe

In [13]:

```
file_path = 'practical2.csv'
```

In [14]:

```
df = pd.read_csv(file_path)
```

reading head of data

In [15]:

```
print(df.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
```

Selecting relevant columns for the model

In [29]:

```
correlation = df.corr(numeric_only=True)['Profit']
```

In [30]:

```
print(correlation)
```

```
R&D Spend          0.972900
Administration     0.200717
Marketing Spend    0.747766
Profit             1.000000
Name: Profit, dtype: float64
```

As correlation coeafficient with R&D Spend is more than any other attribute so we will selecting it for our prediction model as Independent variable

In [16]:

```
x = df[['R&D Spend']]
```

In [17]:

```
X=x.to_numpy()
```

Target variable

In [18]:

```
y = df['Profit']
```

In [19]:

```
Y=y.to_numpy()
```

Splitting the data into training and testing sets

In [20]:

```
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, ra
```

In [21]:

```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

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

Training the model

In [22]:

```
from sklearn.linear_model import LinearRegression
regressor=LinearRegression()
regressor.fit(x_train,y_train) # Training the algorithm
```

Out[22]:

```
LinearRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

Predictions on the test set

In [23]:

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

In [24]:

```
print(y_pred)
```

```
[104667.27805998 134150.83410578 135207.80019517  72170.54428856
 179090.58602508 109824.77386586  65644.27773757 100481.43277139
 111431.75202432 169438.14843539]
```

Evaluating the model

In [34]:

```
print(f'The Error y-y`: {y_test-y_pred}')
```

```
The Error y-y`: [-1384.89805998 10108.56589422 10914.14980483  5628.285711
44
 11959.80397492 -4816.46386586 15584.78226243 -2997.87277139
 -1079.50202432 -3250.20843539]
```

In [42]:

```
e_array=(y_test-y_pred)**2
print(f'The Error y-y`: {e_array}')
```

```
The Error y-y`: [1.91794264e+06 1.02183104e+08 1.19118666e+08 3.16776000e+
07
 1.43036911e+08 2.31983242e+07 2.42885438e+08 8.98724115e+06
 1.16532462e+06 1.05638549e+07]
```

In [44]:

```
# Calculate the sum of elements
array_sum = np.sum(e_array)

# Display the result
print("Sum of array elements:", array_sum)

# Taking the Mean
print("Mean of array elements:", array_sum/10)
```

```
Sum of array elements: 684734407.1905932
Mean of array elements: 68473440.71905932
```

Above are the actual calculation and below using the predefined functions

In [25]:

```
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
```

In [26]:

```
print(f'Mean Squared Error: {mse}')
print(f'R-squared: {r2}')
```

```
Mean Squared Error: 68473440.71905932
R-squared: 0.9464587607787219
```