Sales Prediction using Linear from Scratch

Predicting Total Sales for Advertising Amount (in 1000s of Dollars) spent

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df = pd.read_csv("Advertising.csv")
```

In [3]:
```
df.head()
```

Out[3]:

	TV	radio	newspaper	sales
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	9.3
3	151.5	41.3	58.5	18.5
4	180.8	10.8	58.4	12.9

Finding relation between advertising and total sales

Total Sales = Sales of Tv + Radio + Newspaper

In [4]:
```
df['total_spend'] = df['TV'] + df['radio'] + df['newspaper']
```

In [5]:
```
df.head()
```

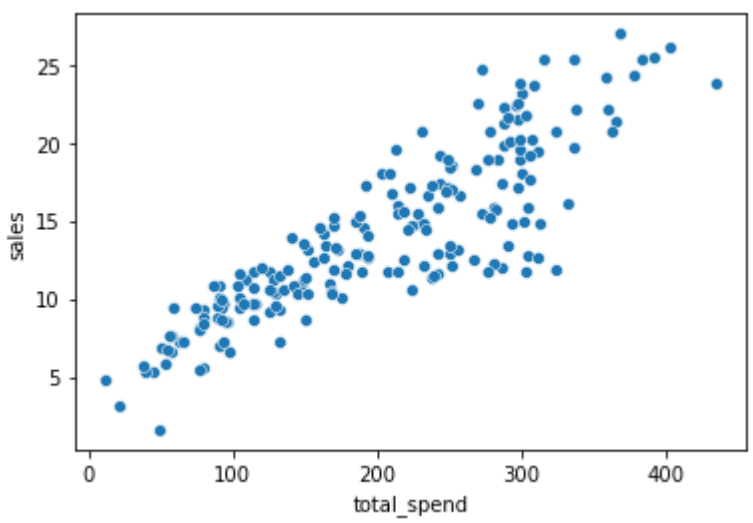
Out[5]:

	TV	radio	newspaper	sales	total_spend
0	230.1	37.8	69.2	22.1	337.1
1	44.5	39.3	45.1	10.4	128.9
2	17.2	45.9	69.3	9.3	132.4
3	151.5	41.3	58.5	18.5	251.3
4	180.8	10.8	58.4	12.9	250.0

Realising if there is any relation or not

In [6]:
```
sns.scatterplot(data=df, x='total_spend', y='sales')
```

Out[6]:
```
<AxesSubplot:xlabel='total_spend', ylabel='sales'>
```



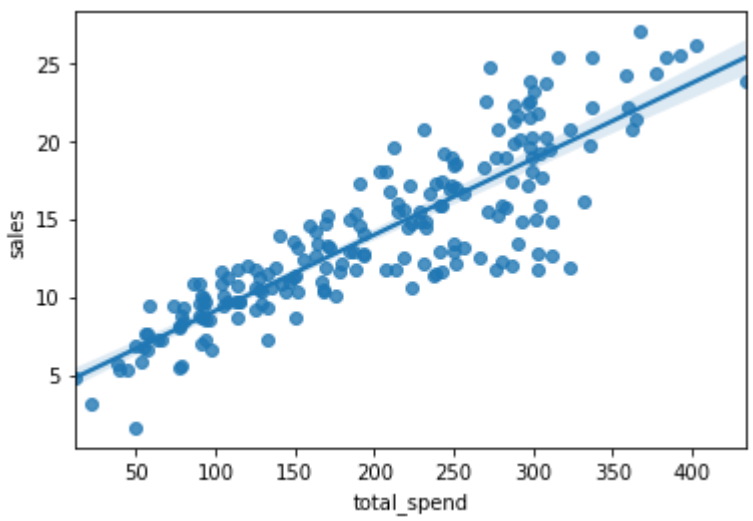
Seems to have a relation that if total_spend increases overall sales increased

Now the task is that what might be the relationship between sales and total_spend...
It seems that there exists a linear relationship for the most so let's plot a regression line..

Using Pre-defined Methods

In [7]:
```
sns.regplot(data=df, x='total_spend', y='sales')
```

Out[7]:
```
<AxesSubplot:xlabel='total_spend', ylabel='sales'>
```



Lets test the linear regression with the help of a feature

In [8]:
```
X = df['total_spend']
y = df['sales']
```

Lets use Oridinary Least Squares approach to this. The approach isn't so particular but a basic approach is fine for this linear shaped plot

Ordinary Least squares fit from scratch for Verification

y = b + mx (Simple Linear Regression)

y = B0 + B1x1 + B2x2 + ... (Multiple Linear Regression - Degree = 1)

y = B0 + B1x^1 + B2x^2 + ... (Polynomial Regression - Degree > 1)

Lets find the beta-coefficients of the X

In [9]:
```
myBetas = np.polyfit(X, y, deg=1)
myBetas
```

Out[9]:
```
array([0.04868788, 4.24302822])
```

In [10]:
```
B1, B0 = myBetas
```

In [11]:
```
B1
```

Out[11]:
```
0.048687879319048145
```

In [12]:
```
B0
```

Out[12]:
```
4.2430282160363255
```

Lets try a prediction

In [13]:
```
spending = np.linspace(0, 500, 100)
```

In [14]:
```
predicted_sales = B0 + B1*spending
```

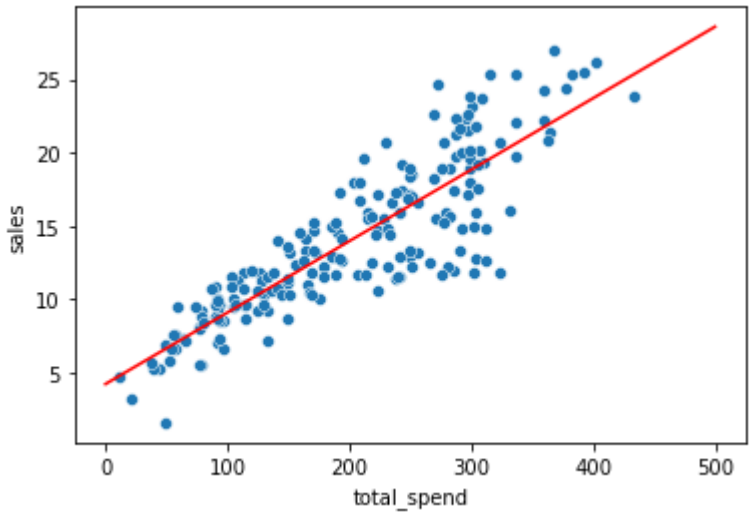
In [17]:
```
predicted_sales
```

Out[17]:
```
array([ 4.24302822,  4.4889266 ,  4.73482498,  4.98072336,  5.22662174,
        5.47252012,  5.7184185 ,  5.96431688,  6.21021526,  6.45611364,
        6.70201202,  6.9479104 ,  7.19380878,  7.43970716,  7.68560554,
        7.93150392,  8.1774023 ,  8.42330068,  8.66919906,  8.91509744,
        9.16099582,  9.4068942 ,  9.65279258,  9.89869097, 10.14458935,
       10.39048773, 10.63638611, 10.88228449, 11.12818287, 11.37408125,
       11.61997963, 11.86587801, 12.11177639, 12.35767477, 12.60357315,
       12.84947153, 13.09536991, 13.34126829, 13.58716667, 13.83306505,
       14.07896343, 14.32486181, 14.57076019, 14.81665857, 15.06255695,
       15.30845533, 15.55435371, 15.80025209, 16.04615048, 16.29204886,
       16.53794724, 16.78384562, 17.029744  , 17.27564238, 17.52154076,
       17.76743914, 18.01333752, 18.2592359 , 18.50513428, 18.75103266,
       18.99693104, 19.24282942, 19.4887278 , 19.73462618, 19.98052456,
       20.22642294, 20.47232132, 20.7182197 , 20.96411808, 21.21001646,
       21.45591484, 21.70181322, 21.9477116 , 22.19360999, 22.43950837,
       22.68540675, 22.93130513, 23.17720351, 23.42310189, 23.66900027,
       23.91489865, 24.16079703, 24.40669541, 24.65259379, 24.89849217,
       25.14439055, 25.39028893, 25.63618731, 25.88208569, 26.12798407,
       26.37388245, 26.61978083, 26.86567921, 27.11157759, 27.35747597,
       27.60337435, 27.84927273, 28.09517111, 28.3410695 , 28.58696788])
```

Lets see the plotting

In [18]:
```
sns.scatterplot(x='total_spend', y='sales', data=df)
plt.plot(spending, predicted_sales, color='red')
```

Out[18]:
```
[<matplotlib.lines.Line2D at 0x1bb6042a448>]
```



Lets Randomly give a spend value

In [19]:
```
spend = 200
predicted_sales = B0 + spend*B1
```

In [22]:
```
print("The Predicted Sale =", int(predicted_sales.round(1)))
```

```
The Predicted Sale = 14
```