Sales prediction for broadcasting media using Linear Regression

Ninad Mandavkar
12 min read · Mar 14, 2024

GitHub link

Broadcasting media all over the globe has been a rapidly evolving segment when it comes to making people cognizant about the society. The contemporary world needs to constantly be in concurrence with the cutting edge technology, economic advances, gloabalization and much more. The advent of AI has also been considered as a threat to the existing media houses, where the adaptive technology might come with an alternative that will get people information at one click without the need of subscribing to any media model. Lucks for us, there is one thing that is still a major problem when it comes to handling these big media houses, something which as of now cannot be handled purely by a humanoid. The answer is “data”! Along with the colossal availability of data, the task of handling such datasets is still a major issue. Even a bigger dilemma is prediction of judicial advertising of media houses and use of business acumen to channel the advertising revenue to media houses that may reap higher dividends.

Today, we are going to work on such a use case which takes the data of 3 such media houses “TV, Radio & Newspaper” and tries to scrutinize using

hyperparameters on which particular media house is more remunerative and how with the help of Machine Learning one can predict the Sales for the future.

Before beginning with the actual code, one should understand the concept of Linear Regression on which this entire problem revolves. Let us try to imbibe some theory before we implement the algorithm pragmatically.

Concept of Linear regression

The equation used in Linear Regression is

$$y = b0 + b1x1 + b2x2 + b3x3 \ldots\ldots bnxn$$

Here b0 = y-intercept: value of y at x=0, where b0, b1 & b2 are coefficients of linear regression.
Here x1, x2, x3....= independent variables & y = dependent variable

By making use of the past data we already have y and x. After applying Linear regression on these variables we can easily find b0, b1 & b2.

Using these coefficients for a given values of x, we can then predict the value of y in future!

Concept of Best-fit line

Linear Regression works on the concept of “Best-fit-line”.
Best-fit-line is the line that pass closest to all the datapoints in the X-Y co-ordinate system. (Similar to Trendline).
So equation of this best fit line is nothing but the equation of linear regression above:

$$y = b0 + b1x1 + b2x2 + b3x3 \ldots\ldots bnxn$$

With the help of this best-fit-line one can predict y for any given value of x.

To check which particular independent variable (x1, x2,x3...xn) impacts the dependent variable(y) more, we need to check which particular coefficient obtained after Linear Regression has the highest value. Higher the coefficient more the impact.

Concept of RMSE & R squared

Evaluation of a Regression model is done using Root Mean Square Error (RMSE) & R squared. Lesser the RMSE better is the model. Always remember Error is the difference between Actual and Predicted value.

Actual Value — Predicted value = error

So one can also infer that lesser RMSE infers to lesser error and hence more enhanced model. To define a RMSE, it is a measure of the goodness of fit of a Linear Regression model.

R squared tells you how well the regression model is predicting as compared to the mean model.
It lies between (0–1).

If R squared is close to 1 → very good model.
If R squared is close to 0.5 → Needs tuning.
If R squared is close to 0 → Not a good model.
If R squared is less than 0 → Mean model is better than the regression model.

. . .

> *Code*

1. Import all the libraries & read the dataset.

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

```
data = pd.read_csv("advertising.csv")
```

```
data
```

```
data
```

	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	12.0
3	151.5	41.3	58.5	16.5
4	180.8	10.8	58.4	17.9
...	...	...	...	...
195	38.2	3.7	13.8	7.6
196	94.2	4.9	8.1	14.0
197	177.0	9.3	6.4	14.8
198	283.6	42.0	66.2	25.5
199	232.1	8.6	8.7	18.4

Apparently the dataset is about the investment in dollars done on media segments be it TV, Radio & Newspaper and their respective Sales, the final Sales column is a dependent column which is a corollary of individual advertising revenue.

So if one does a layman check it is very clear that investment done in Television Media (TV) is much more compared to that of Radio and Newspaper and it does feel that majority of Sales might be derived from TV as a media.

However, I would be happy to debunk your assumptions soon :)

Let us focus on the next step of the algorithm, which is '*Assumption Checking*'.

2. Assumption checking

Before moving further with the regression, one should do Assumption checking. Assumption checking is a pre-requisite for any machine Learning algorithm. It involves checking the data for any null values, outliers, linearity, multi-co-linearity, categorical columns and treating the data if applicable.

Some of the important Assumption checks are:

2.1 Linearity : Independent variable (x) & dependent variable (Y) should have a linear relation. It can be checked using Correlation either by heatmap/scatter plot.

2.2 No multi co-linearity: There should not exist a linear relation between the independent columns (x). In such cases, one has to drop one of those columns.

2.3 No auto correlation: Auto correlation indicates if the values in a column are dependent on their previous values from the same column, in such cases the column is auto-correlated. No Auto-correlation for any column should exist.(Only time-series data is auto correlated)

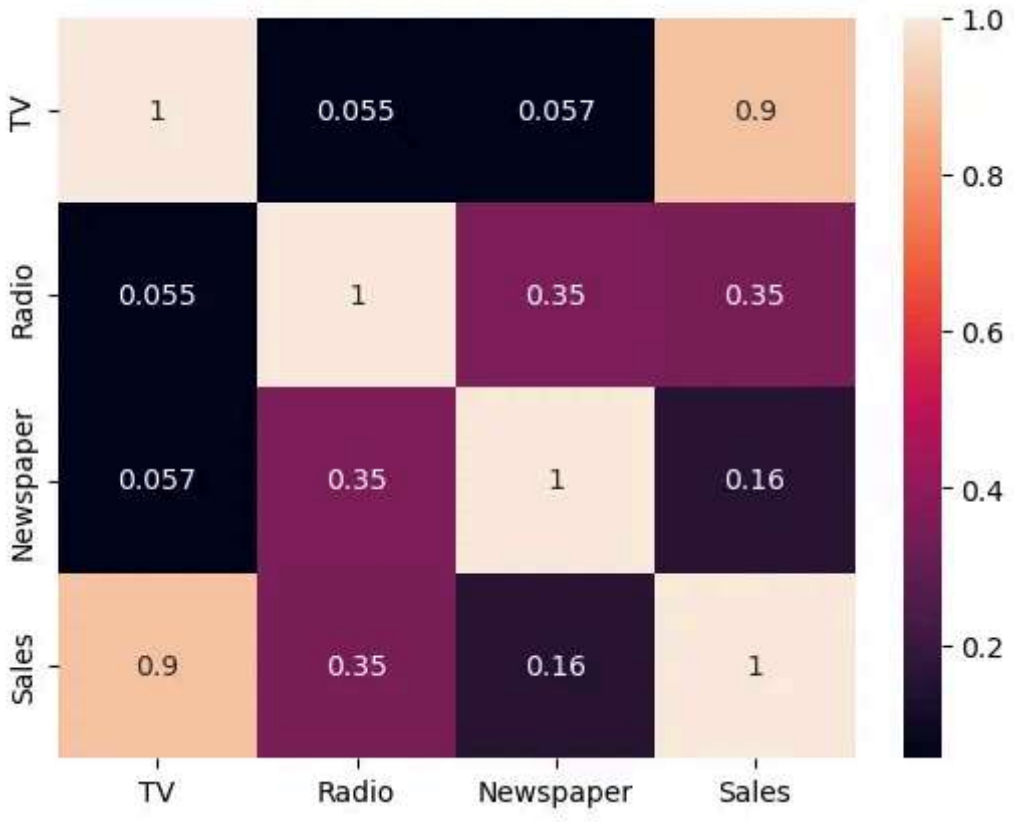
2.4 No outliers: Outliers are the extreme values in the dataset that skew away from the original datapoints. No outliers should be present. Outliers can be identified using a boxplot.

2.5 No null values: Dataset should not contain null values.

2.6 No Categorical columns : No Categorical columns should be present. If they exist, they should be mapped using Label Encoder. Character to numbers (only applies when we have categorical columns).

Let us Check for Linearity, Multi-co-linearity & Auto correlation using heatmap.

```
2
3 sns.heatmap(data.corr(),annot=True)
```



The threshold for linear correlation = 0.1
The relation in heatmap below 0.1 is NOT correlated
The relation in heatmap above 0.1 is linearly correlated
The relation in heatmap = 0 is NOT correlated

In the boxplot above
Independent col : x: Newspaper, Radio, TV
Dependent col : y: Sales

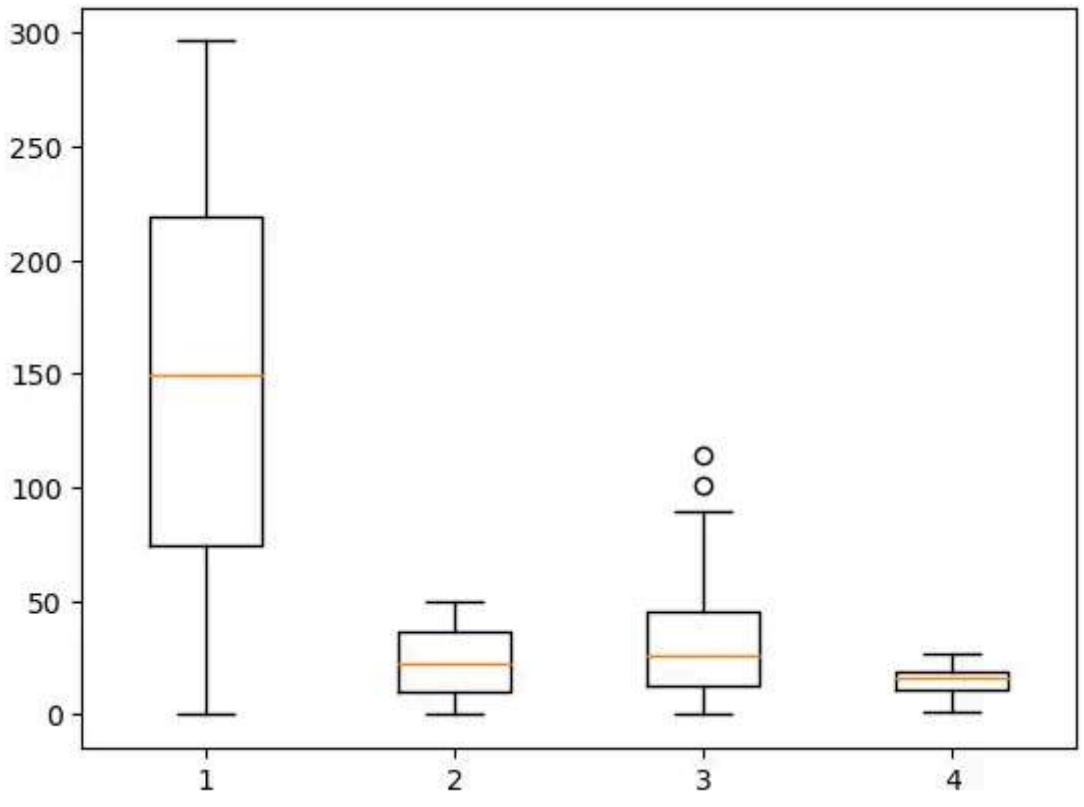
As evident from the boxplot, the correlation of Sales wrt to Newspaper, Radio & TV is 0.16,0.35 & 0.9 (leftmost col in the boxplot) and all of them are greater than 0.1 ***Hence there exists linearity and correlation.***

Now let us check for 'Multi correlation',(look at 1st 3 cols in the plot above), in this case the correlation between Newspaper & TV for one scenario is 0.057 > 0.1. *Hence* ***there exists Multi correlation b/w Newspaper & TV.*** However, one should not eliminate the columns since we have lack of columns. This is called Feature selection/variable selection.

Also since the values in every independent column above are not anyhow correlated to their previous value ***we have no autocorrelation*** as well.(Only time-series data is auto correlated)

Let us check for outliers now,

```
1 plt.boxplot(data)
```



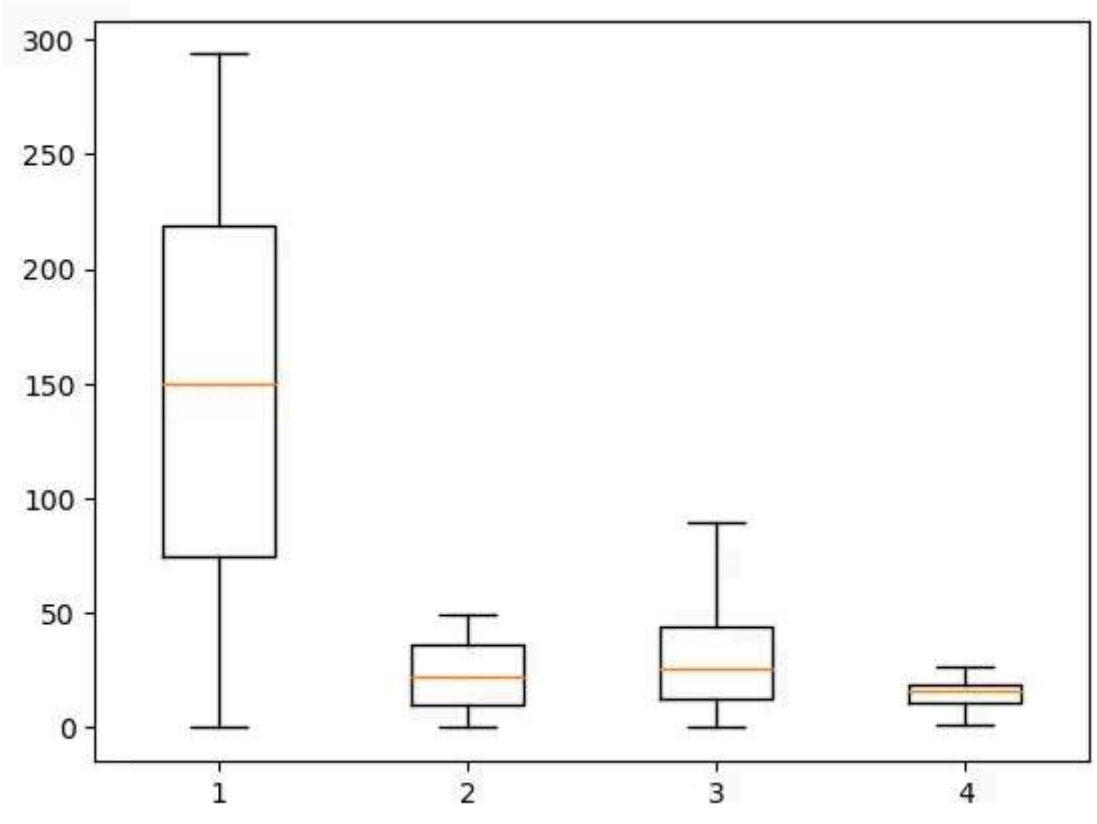
As one can see, we do see 2 outliers for the 3rd column “Newspaper” which should be eliminated.

```
data_new = data[data["Newspaper"]< 90]    # Fetching data below 90 so that we
                                          # exclude the outlier in Newspaper column.
```

```
data_new
```

	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	12.0
3	151.5	41.3	58.5	16.5
4	180.8	10.8	58.4	17.9
...	...	...	...	...
195	38.2	3.7	13.8	7.6
196	94.2	4.9	8.1	14.0
197	177.0	9.3	6.4	14.8
198	283.6	42.0	66.2	25.5
199	232.1	8.6	8.7	18.4

```
plt.boxplot(data_new)
```



Now let us check for Null values,

```
data_new.isnull().sum()
```

```
TV           0
Radio        0
Newspaper    0
Sales        0
dtype: int64
```

As we have got No null values from above, we have no need to do Null Value treatment on data_new.

However, if we have got some null values, we can call the following *Null_Value_Treatment(data_new)* function where we replace integer and float columns having null values with mean and rest other columns having null values with mode.

```
def Null_Value_Treatment(data_new):
    for i in data_new.columns:
        if data_new[i].dtypes == "int64" or data_new[i].dtypes == "float64":
            data_new[i].fillna(data_new[i].mean(),inplace=True)
        else:
            data_new[i].fillna(data_new[i].mode()[0],inplace=True)
    return data_new.isnull().sum()
```

Even if we do not have null values, there is a chance that the dataset contains special characters. Hence, a check for special characters should be done as follows:

```
for i in data_new.columns:
    print(f"Unique value for coulmn: {i}\n\n{data_new[i].unique()}\n")
```

unique() functions checks for special character in the dataset.

```
Unique value for coulmn: TV

[230.1  44.5  17.2 151.5 180.8   8.7  57.5 120.2   8.6 199.8  66.1 214.7
```

```
 23.8  97.5 204.1 195.4 281.4  69.2 147.3 218.4 237.4  13.2 228.3  62.3
262.9 142.9 240.1 248.8  70.6 292.9 112.9  97.2 265.6  95.7 290.7 266.9
 74.7  43.1 228.  202.5 177.  293.6 206.9  25.1 175.1  89.7 239.9 227.2
 66.9 100.4 216.4 182.6 262.7 198.9   7.3 136.2 210.8 210.7  53.5 261.3
239.3 102.7 131.1  69.   31.5 139.3 216.8 199.1 109.8  26.8 129.4 213.4
 16.9  27.5 120.5   5.4 116.   76.4 239.8  75.3  68.4 213.5 193.2  76.3
110.7  88.3 134.3  28.6 217.7 250.9 107.4 163.3 197.6 184.9 289.7 135.2
222.4 280.2 187.9 238.2 137.9  25.   90.4  13.1 255.4 225.8 241.7 175.7
209.6  78.2  75.1 139.2 125.7  19.4 141.3  18.8 224.  123.1 229.5  87.2
  7.8  80.2 220.3  59.6   0.7 265.2   8.4 219.8  36.9  48.3  25.6 273.7
 43.   73.4 193.7 220.5 104.6  96.2 140.3 243.2  38.   44.7 280.7 121.
171.3 187.8   4.1  93.9 149.8  11.7 131.7 172.5  85.7 188.4 163.5 117.2
234.5  17.9 206.8 215.4 284.3  50.  164.5  19.6 168.4 276.9 248.4 170.2
276.7 165.6 156.6 218.5  56.2 287.6 253.8 205.  139.5 191.1 286.   18.7
 39.5  75.5 166.8 149.7  38.2  94.2 283.6 232.1]

Unique value for coulmn: Radio

[37.8 39.3 45.9 41.3 10.8 48.9 32.8 19.6  2.1  2.6  5.8 24.  35.1  7.6
 32.9 47.7 39.6 20.5 23.9 27.7  5.1 15.9 16.9 12.6  3.5 29.3 16.7 27.1
 16.  28.3 17.4  1.5 20.   1.4  4.1 43.8 49.4 26.7 37.7 22.3 33.4  8.4
 25.7 22.5  9.9 41.5 15.8 11.7  3.1  9.6 41.7 46.2 28.8 28.1 19.2 49.6
 29.5  2.  42.7 15.5 29.6 42.8  9.3 24.6 14.5 27.5 43.9 30.6 14.3 33.
  5.7 43.7  1.6 28.5 29.9  7.7 20.3 44.5 43.  18.4 40.6 25.5 47.8  4.9
 33.5 36.5 14.  31.6 21.  42.3  4.3 10.1 17.2 34.3 46.4 11.   0.3  0.4
 26.9  8.2 38.  15.4 20.6 46.8 35.   0.8 36.9 26.8 21.7  2.4 34.6 32.3
 11.8 38.9  0.  49.  12.   2.9 27.2 38.6 47.  39.  28.9 25.9 17.  35.4
 33.2 14.8  1.9  7.3 40.3 25.8 13.9 23.3 39.7 21.1 11.6 43.5  1.3 18.1
 35.8 36.8 14.7  3.4 37.6  5.2 23.6 10.6 20.9 20.1  7.1 30.2  7.8  2.3
 10.   5.4 21.3 45.1 28.7 12.1 41.1 42.  35.6  3.7  8.6]

Unique value for coulmn: Newspaper

[69.2 45.1 69.3 58.5 58.4 75.  23.5 11.6  1.  21.2 24.2  4.  65.9  7.2
 46.  52.9 55.8 18.3 19.1 53.4 49.6 26.2 19.5 12.6 22.9 40.8 43.2 38.6
 30.   0.3  7.4  8.5  5.  45.7 35.1 32.  31.6 38.7  1.8 26.4 43.3 31.5
 35.7 18.5 49.9 36.8 34.6  3.6 39.6 58.7 15.9 60.  41.4 16.6 37.7  9.3
 21.4 54.7 27.3  8.4 28.9  0.9  2.2 10.2 11.  27.2 31.7 19.3 31.3 13.1
 89.4 20.7 14.2  9.4 23.1 22.3 36.9 32.5 35.6 33.8 65.7 16.  63.2 73.4
 51.4 33.  59.  72.3 10.9  5.9 22.  51.2 45.9 49.8 17.9  5.3 29.7 23.2
 25.6  5.5 56.5  2.4 10.7 34.5 52.7 14.8 79.2 46.2 50.4 15.6 12.4 74.2
 25.9 50.6  9.2  3.2 43.1  8.7 43.   2.1 65.6 59.7 20.5  1.7 12.9 75.6
 37.9 34.4 38.9  9.  44.3 11.9 20.6 37.  48.7  9.5  5.7 50.5 24.3 45.2
 30.7 49.3  5.4 84.8 21.6 19.4 57.6  6.4 18.4 47.4 17.  12.8 41.8 20.3
 35.2 23.7 17.6  8.3 27.4 71.8 19.6 26.6 18.2  3.7 23.4  5.8  6.  13.8
  8.1 66.2]

Unique value for coulmn: Sales

[22.1 10.4 12.  16.5 17.9  7.2 11.8 13.2  4.8 15.6 12.6 17.4  9.2 13.7
 19.  22.4 24.4 11.3 14.6 18.  17.5  5.6 20.5  9.7 17.  15.  20.9 18.9
 10.5 21.4 11.9 17.8 25.4 14.7 10.1 21.5 16.6 17.1 20.7  8.5 16.1 10.6
 23.2 19.8 16.4 10.7 22.6 21.2 20.2 23.7  5.5 23.8 18.4  8.1 24.2 14.
 16.  11.  13.4 22.3 18.3 12.4  8.8  8.7  6.9 14.2  5.3 17.3 13.6 21.7
 12.9 16.7  7.3 19.4 22.2 11.5 16.9 17.2 19.7 21.8 12.2  9.4 15.9  6.6
 15.5  7.  15.2 24.7  1.6 17.7  5.7 19.6 10.8 11.6  9.5 20.8  9.6 10.9
 19.2 20.1 12.3 10.3 18.2 20.6  3.2 15.3 13.3 19.9  8.  20.   8.4  7.6
 27.  16.8 17.6 26.2  6.7  5.9 14.8 25.5]
```

As one can clearly see, we have got no special characters like “?” in the dataset, so our data is now ready for getting trained and tested on ML algorithms.

3. Splitting the data using train_test_split

Before we train the data on ML algorithm it is important that we split the data into independent variables (x) and dependent variable (y)

```
x = data_new[["TV","Radio","Newspaper"]]    ## While assigning multiple parameters we use
                                            ## mutiple brackets
y = data_new[["Sales"]]
```

```
x
```

	TV	Radio	Newspaper
0	230.1	37.8	69.2
1	44.5	39.3	45.1
2	17.2	45.9	69.3
3	151.5	41.3	58.5
4	180.8	10.8	58.4
...	...	...	...
195	38.2	3.7	13.8
196	94.2	4.9	8.1
197	177.0	9.3	6.4
198	283.6	42.0	66.2
199	232.1	8.6	8.7

```
y
```

	Sales
0	22.1
1	10.4
2	12.0
3	16.5
4	17.9
...	...
195	7.6
196	14.0
197	14.8
198	25.5
199	18.4

Let us check the size of the dataset,

```
x.shape ##Indicates 3 columns in x for the given example
```

```
(198, 3)
```

```
y.shape  ##Indicates 1 column in y for the given example
```

```
(198, 1)
```

```
data.shape
```

```
(200, 4)
```

Shape function returns the size of the dataset. Now let us split the dependent variable (y) and independent variables (x) into train and test data.

```
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)

```

Here we are using ‘sklearn’ library which is using another sub library ‘model_selection’ to import a splitting function *‘train_test_split’* which splits the independent variable ‘x’ & dependent variable ‘y ’ into train and test sizes

based on the ratio 70–80 % for train size and 20 % (0.2) for the test size for each variable.

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

```
(158, 3)
(40, 3)

(158, 1)
(40, 1)
```

On printing the size of the train & test data we can make the following conclusion on comparing it with the original shape of x and y:

x.shape = 198 splitted in x_train = 158, x_test = 40
y.shape = 198 splitted in y_train = 158, y_test = 40

4. Importing Linear Regression function

Once done with splitting the data, we have to use Linear Regression on the trained data using fit function.
But before that we need to call LinearRegression() function by importing it from sklearn.linear_model library.

```
from sklearn.linear_model import LinearRegression
lm = LinearRegression()
```

```
lm.fit(x_train,y_train)
```

```
print(lm.coef_)
```

```
[[0.05381805 0.10159698 0.00186049]]
```

Here *lm.coef* will return all the coefficients *(b1, b2 & b3)* that we obtain after performing LinearRegression using LinearRegression()

Here the value of coefficient b2=0.10159698 is higher than the other coefficients.

From this we can conclude that higher b2 corresponds to larger impact on y. Hence for the above example, investment in advertising for Radio will give us higher return at Sales.

```
print(lm.intercept_)
```

```
[4.76105011]
```

lm.intercept returns the value of y-intercept (b0)

To summarise the above result,
I/p to a LinearRegression — -> x_train, y_train
O/p of a LinearRegression — -> b0, b1, b2, b3

So since we already have the coefficients now, for the given value of x in the future, we can easily predict y for the future.

```
x1 =250
x2= 14
x3 = 67

y = 4.76105011 + (0.05381805*x1)+(0.10159698*x2)+(0.00186049*x3)
```

```
y
```

```
19.76257316
```

5. Predicting the y_pred based on x_test data

y (i.e. Sales) can be now predicted based on the test data as follows:

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

```
y_pred
```

```
array([[ 8.08582129],
       [11.23169063],
       [16.4456282 ],
       [19.39121801],
       [10.79384298],
       [16.45502156],
       [17.8845832 ],
       [ 8.6647108 ],
       [ 9.93272352],
       [17.15031685],
       [ 8.10690585],
       [20.07766535],
       [10.66075919],
       [10.79062784],
       [16.79853713],
       [18.14214081],
```

6. Constructing a DataFrame

Once y (Sales) is predicted let us check the closeness of the actual and predicted data by constructing a data frame.

```
# DataFrame is used for constructing a table

new_df = pd.DataFrame()
new_df = x_test
new_df['actual sales']= y_test
new_df['predicted sales']= y_pred
new_df
```

	TV	Radio	Newspaper	actual sales	predicted sales
56	7.3	28.1	41.4	5.5	8.085821
6	57.5	32.8	23.5	11.8	11.231691
194	149.7	35.6	6.0	17.3	16.445628
53	182.6	46.2	58.7	21.2	19.391218
13	97.5	7.6	7.2	13.7	10.793843
167	206.8	5.2	19.4	17.2	16.455022
165	234.5	3.4	84.8	16.9	17.884583
170	50.0	11.6	18.4	8.4	8.664711
191	75.5	10.8	6.0	11.9	9.932724
85	193.2	18.4	65.7	20.2	17.150317
78	5.4	29.9	9.4	5.3	8.106906

7. Evaluating the model using Root Mean Square Error and R squared

The evaluation of Linear Regression model is done to check how accurate the model is performing. This accuracy can be checked using parameters like Root Mean Square Error and R squared.

```
lm.score(x_train,y_train)*100
```

```
89.88614375458548
```

```
from sklearn.metrics import r2_score,mean_squared_error
import numpy as np

r2=r2_score(y_test,y_pred)
print("R-squared:",r2)

rmse=np.sqrt(mean_squared_error(y_test,y_pred))
print("RMSE:",rmse)
```

```
R-squared: 0.9069156913074427
RMSE: 1.6344492185626285
```

R-squared around 90% (i.e. 0.9069...) is an indication that the model is highly accurate. Also, a RMSE value as high as 1.63 confirms the evaluation efficiency as well.

To conclude we can say that, amongst the broadcasting medium of TV Radio and Newspaper, investments should be directed towards radio since it will garner higher return of Sales.

Since, the Newspaper as a medium has the lowest impact, broadcasting agency should focus on advertising in Newspapers to gather more attention of consumers to the otherwise obsolete medium.

To know more about the source code, check out my GitHub.

Machine Learning