

```
In [57]: import pandas as pd
```

```
In [58]: data=pd.read_csv("/home/placement/Desktop/divyasri/Advertising.csv")
```

```
In [59]: data.describe()
```

```
Out[59]:
```

	Unnamed: 0	TV	radio	newspaper	sales
count	200.000000	200.000000	200.000000	200.000000	200.000000
mean	100.500000	147.042500	23.264000	30.554000	14.022500
std	57.879185	85.854236	14.846809	21.778621	5.217457
min	1.000000	0.700000	0.000000	0.300000	1.600000
25%	50.750000	74.375000	9.975000	12.750000	10.375000
50%	100.500000	149.750000	22.900000	25.750000	12.900000
75%	150.250000	218.825000	36.525000	45.100000	17.400000
max	200.000000	296.400000	49.600000	114.000000	27.000000

```
In [60]: data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
#   Column      Non-Null Count  Dtype
---  -
0   Unnamed: 0   200 non-null   int64
1   TV           200 non-null   float64
2   radio        200 non-null   float64
3   newspaper    200 non-null   float64
4   sales        200 non-null   float64
dtypes: float64(4), int64(1)
memory usage: 7.9 KB
```

```
In [61]: data.isna().sum() #null values
```

```
Out[61]: Unnamed: 0      0  
         TV           0  
         radio        0  
         newspaper    0  
         sales        0  
         dtype: int64
```

```
In [62]: data1=data.drop(['Unnamed: 0'],axis=1)
```

```
In [63]: data1
```

```
Out[63]:
```

	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
...
195	38.2	3.7	13.8	7.6
196	94.2	4.9	8.1	9.7
197	177.0	9.3	6.4	12.8
198	283.6	42.0	66.2	25.5
199	232.1	8.6	8.7	13.4

200 rows × 4 columns

```
In [64]: cor=data1.corr() #correlation
```

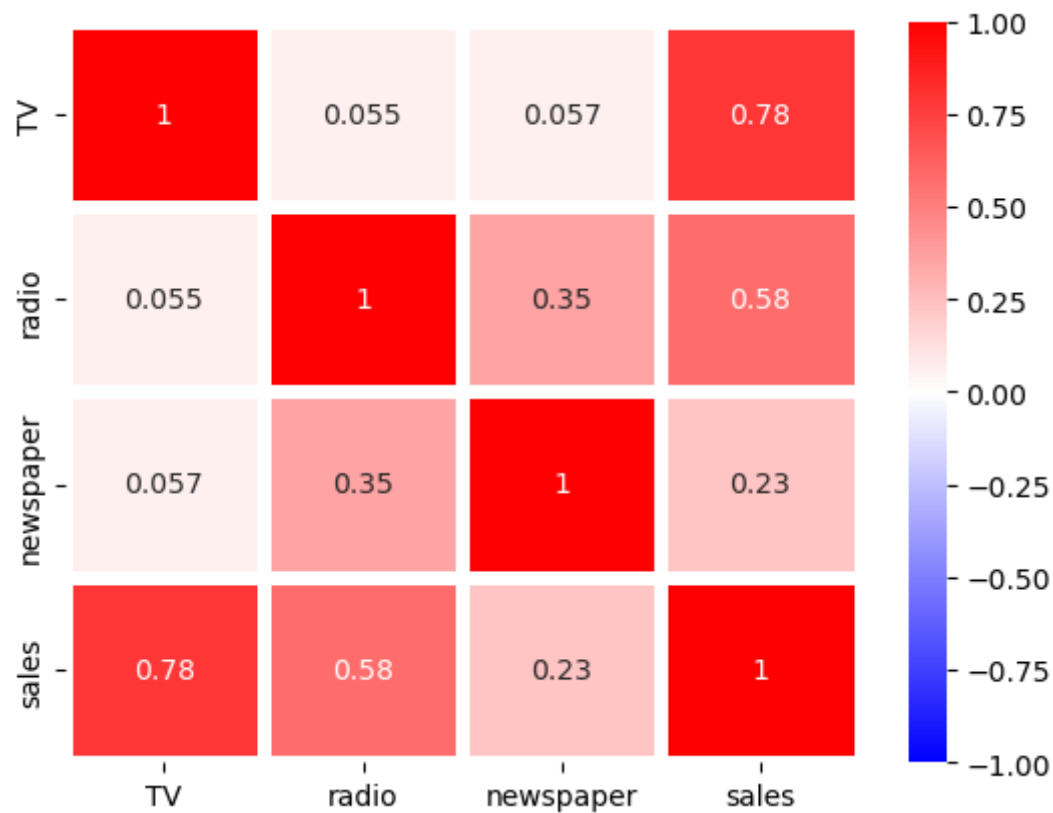
In [65]: cor

Out[65]:

	TV	radio	newspaper	sales
TV	1.000000	0.054809	0.056648	0.782224
radio	0.054809	1.000000	0.354104	0.576223
newspaper	0.056648	0.354104	1.000000	0.228299
sales	0.782224	0.576223	0.228299	1.000000

```
In [66]: import seaborn as sns                                #correlation matrix
sns.heatmap(cor, vmax=1, vmin=-1, annot=True, linewidths=5, cmap='bwr')
```

Out[66]: <Axes: >



```
In [67]: list(data1)
```

Out[67]: ['TV', 'radio', 'newspaper', 'sales']

```
In [68]: y=data1['sales']
x=data1.drop(['sales'],axis=1)
```

In [69]:

y

Out[69]:

```
0      22.1
1      10.4
2       9.3
3      18.5
4      12.9
...
195     7.6
196     9.7
197    12.8
198    25.5
199    13.4
Name: sales, Length: 200, dtype: float64
```

In [70]:

x

Out[70]:

	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

200 rows × 3 columns

```
In [71]: from sklearn.model_selection import train_test_split #splitting of training and testing
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.33,random_state=42)
```

```
In [72]: x_train.head(5)
```

```
Out[72]:
```

	TV	radio	newspaper
42	293.6	27.7	1.8
189	18.7	12.1	23.4
90	134.3	4.9	9.3
136	25.6	39.0	9.3
51	100.4	9.6	3.6

```
In [73]: x_test.head(5)
```

```
Out[73]:
```

	TV	radio	newspaper
95	163.3	31.6	52.9
15	195.4	47.7	52.9
30	292.9	28.3	43.2
158	11.7	36.9	45.2
128	220.3	49.0	3.2

```
In [74]: y_train.head(5)
```

```
Out[74]: 42      20.7
189      6.7
90      11.2
136      9.5
51      10.7
Name: sales, dtype: float64
```

```
In [75]: y_test.head(5)
```

```
Out[75]: 95      16.9  
        15      22.4  
        30      21.4  
        158     7.3  
        128     24.7  
        Name: sales, dtype: float64
```

```
In [76]: from sklearn.linear_model import LinearRegression  
        reg=LinearRegression()#creating object of LinearRegression  
        reg.fit(x_train,y_train)#training and fitting LR object using training data
```

```
Out[76]: 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.
```

```
In [77]: ypred=reg.predict(x_test)#prediction of values(x_test*reg)
```

```
In [78]: ypred
```

```
Out[78]: array([16.58673085, 21.18622524, 21.66752973, 10.81086512, 22.25210881,  
                13.31459455, 21.23875284,  7.38400509, 13.43971113, 15.19445383,  
                9.01548612,  6.56945204, 14.4156926 ,  8.93560138,  9.56335776,  
                12.10760805,  8.86091137, 16.25163621, 10.31036304, 18.83571624,  
                19.81058732, 13.67550716, 12.45182294, 21.58072583,  7.67409148,  
                5.67090757, 20.95448184, 11.89301758,  9.13043149,  8.49435255,  
                12.32217788,  9.99097553, 21.71995241, 12.64869606, 18.25348116,  
                20.17390876, 14.20864218, 21.02816483, 10.91608737,  4.42671034,  
                9.59359543, 12.53133363, 10.14637196,  8.1294087 , 13.32973122,  
                5.27563699,  9.30534511, 14.15272317,  8.75979349, 11.67053724,  
                15.66273733, 11.75350353, 13.21744723, 11.06273296,  6.41769181,  
                9.84865789,  9.45756213, 24.32601732,  7.68903682, 12.30794356,  
                17.57952015, 15.27952025, 11.45659815, 11.12311877, 16.60003773,  
                6.90611478])
```

```
In [79]: from sklearn.metrics import r2_score#efficiency
r2_score(y_test,ypred)#y_test is actual value #ypred is predicted value
```

```
Out[79]: 0.8555568430680086
```

```
In [87]: import warnings
warnings.filterwarnings("ignore")
```

```
In [81]: from sklearn.metrics import mean_squared_error #to calculate rmse
mean_squared_error(ypred,y_test)
```

```
Out[81]: 3.7279283306815105
```

```
In [ ]: #ELASTIC REGRESSION
```

```
In [82]: from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import ElasticNet #for elastic net model
elastic = ElasticNet()
parameters = {'alpha': [1e-15, 1e-10, 1e-8, 1e-4, 1e-3, 1e-2, 1, 5, 10, 20]}
elastic_regressor = GridSearchCV(elastic, parameters)
elastic_regressor.fit(x_train, y_train)
```

```
Out[82]: GridSearchCV(estimator=ElasticNet(),
                      param_grid={'alpha': [1e-15, 1e-10, 1e-08, 0.0001, 0.001, 0.01, 1,
                                             5, 10, 20]})
```

**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.**

```
In [83]: elastic_regressor.best_params_ #alpha value
```

```
Out[83]: {'alpha': 1}
```

```
In [84]: elastic=ElasticNet(alpha=1)
elastic.fit(x_train,y_train)
y_pred_elastic=elastic.predict(x_test) #predicted value
```



```
In [85]: from sklearn.metrics import mean_squared_error #rmse value  
ElasticNet_Error=mean_squared_error(y_pred_elastic,y_test)  
ElasticNet_Error
```

Out[85]: 3.678636493022797

```
In [86]: from sklearn.metrics import r2_score  
r2_score(y_test,y_pred_elastic) #efficiency
```

Out[86]: 0.8574667157937812

```
In [88]: x_test
```

Out[88]:

	TV	radio	newspaper
95	163.3	31.6	52.9
15	195.4	47.7	52.9
30	292.9	28.3	43.2
158	11.7	36.9	45.2
128	220.3	49.0	3.2
...
97	184.9	21.0	22.0
31	112.9	17.4	38.6
12	23.8	35.1	65.9
35	290.7	4.1	8.5
119	19.4	16.0	22.3

66 rows × 3 columns

```
In [97]: test=[[110,33,21]]                                #predicting of single value  
         y_pred_elastic=elastic.predict(test)  
         y_pred_elastic
```

```
Out[97]: array([14.27162918])
```

```
In [98]: test=[[110,33,21],[220,66,13]]                  #predicting of two values  
         y_pred_elastic=elastic.predict(test)  
         y_pred_elastic
```

```
Out[98]: array([14.27162918, 25.54573805])
```

```
In [ ]:
```