In [1]:
```
import pandas as pd
```

In [2]:
```
import warnings
warnings.filterwarnings("ignore")
```

In [3]:
```
data=pd.read_csv("/home/placement/Desktop/BhanuSiva4K8/Advertising.csv")
```

In [4]:
```
data.describe()
```

Out[4]:

	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 [5]:
```
data.isna().sum()
```

Out[5]:
```
Unnamed: 0    0
TV            0
radio         0
newspaper     0
sales         0
dtype: int64
```

In [6]:
```
data=data.drop(['Unnamed: 0'],axis=1)
data
```

Out[6]:

	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

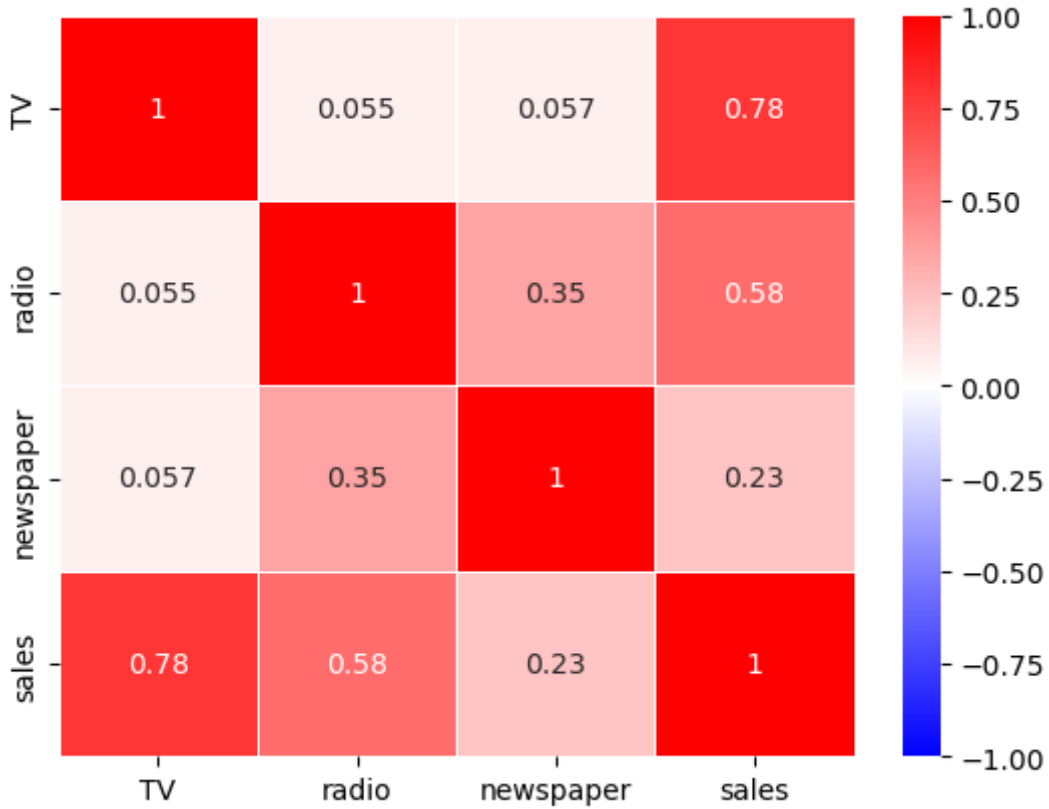
In [7]:
```
cor=data.corr()
cor
```

Out[7]:

	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 [8]:
```
import  seaborn as sns
sns.heatmap(cor,vmax=1,vmin=-1,annot=True,linewidths=.5,cmap='bwr')
```

Out[8]: <Axes: >



In [9]:
```
list(data)
```

Out[9]: ['TV', 'radio', 'newspaper', 'sales']

In [10]:
```
y=data['sales']
x=data.drop('sales',axis=1)
```

In [11]:
```
x
```

Out[11]:

	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 [12]:
```
list(data)
```

Out[12]:
```
['TV', 'radio', 'newspaper', 'sales']
```

In [13]:
```
data1=data.drop(['sales'],axis=1)
data1
```

Out[13]:

	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 [14]:
```
list(data1)
```

Out[14]:
```
['TV', 'radio', 'newspaper']
```

Linear Regression

In [15]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.33,random_state=42)
```

In [16]:
```
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[16]:
```
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 [17]:
```
y_pred=reg.predict(x_test)
y_pred
```

Out[17]:
```
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 [18]:
```
from sklearn.metrics import r2_score
r2_score(y_test,y_pred)
```

Out[18]:
```
0.8555568430680086
```

In [19]:
```
from sklearn.metrics import mean_squared_error
mean_squared_error(y_test,y_pred)
```

Out[19]:
```
3.7279283306815105
```

Elastic Net Regression

In [20]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import ElasticNet

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[20]:
```
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 [21]:
```
elastic_regressor.best_params_
```

Out[21]:
```
{'alpha': 1}
```

In [22]:
```
elastic=ElasticNet(alpha=0.01)
elastic.fit(x_train,y_train)
y_pred_elastic=elastic.predict(x_test)
```

In [23]:
```
from sklearn.metrics import r2_score
r2_score(y_test,y_pred_elastic)
```

Out[23]:
```
0.855576715693211
```

In [24]:
```
from sklearn.metrics import mean_squared_error
elastic_Error=mean_squared_error(y_pred_elastic,y_test)
elastic_Error
```

Out[24]:
```
3.7274154388002283
```

In [25]:
```
x_test
```

Out[25]:

	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 [26]:
```
y_pred_elastic=elastic.predict(x_test)
y_pred_elastic
```

Out[26]:
```
array([16.586402  , 21.18549064, 21.66731146, 10.81048594, 22.25163555,
       13.31420282, 21.23826213,  7.38440465, 13.44030631, 15.19447632,
        9.01566567,  6.56992818, 14.41585343,  8.93561237,  9.56392271,
       12.10797318,  8.86077385, 16.25173792, 10.31045666, 18.83572422,
       19.81009787, 13.6747085 , 12.45155408, 21.58013901,  7.67464897,
        5.67152586, 20.95397442, 11.89337441,  9.13077249,  8.49447362,
       12.32274924,  9.99115106, 21.71913221, 12.64788135, 18.25365935,
       20.17378258, 14.20822564, 21.02783675, 10.91647318,  4.42734865,
        9.5940482 , 12.53183345, 10.14629887,  8.12978131, 13.33033574,
        5.27626244,  9.30549626, 14.15279198,  8.76023033, 11.67055177,
       15.66216243, 11.75402123, 13.21659238, 11.06227267,  6.41837431,
        9.84910774,  9.45785583, 24.32540514,  7.68924136, 12.30858524,
       17.5799634 , 15.27963482, 11.45671827, 11.12265678, 16.60062774,
        6.90638894])
```

In [29]:
```
test=[[110,33,21]]
y_pred_elastic=elastic.predict(test)
y_pred_elastic
```

Out[29]:
```
array([14.28742973])
```

In [30]:
```
test=[[110,33,21],[220,66,13]]
y_pred_elastic=elastic.predict(test)
y_pred_elastic
```

Out[30]:
```
array([14.28742973, 25.63999643])
```

In [ ]:
```

```