In [1]:
```
import pandas as pd
import warnings
warnings.filterwarnings('ignore')
data=pd.read_csv('/home/placement/Downloads/Advertising.csv')#read the titanic csv file
print(data)
```

```
     Unnamed: 0     TV  radio  newspaper  sales
0             1  230.1   37.8       69.2   22.1
1             2   44.5   39.3       45.1   10.4
2             3   17.2   45.9       69.3    9.3
3             4  151.5   41.3       58.5   18.5
4             5  180.8   10.8       58.4   12.9
..          ...    ...    ...        ...    ...
195         196   38.2    3.7       13.8    7.6
196         197   94.2    4.9        8.1    9.7
197         198  177.0    9.3        6.4   12.8
198         199  283.6   42.0       66.2   25.5
199         200  232.1    8.6        8.7   13.4

[200 rows x 5 columns]
```

In [2]:
```
data.describe()
```

Out[2]:

	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 [3]:
```
data.shape
```

Out[3]:
```
(200, 5)
```

In [4]:
```
data.columns
```

Out[4]:
```
Index(['Unnamed: 0', 'TV', 'radio', 'newspaper', 'sales'], dtype='object')
```

In [8]:
```
data1=data.drop(columns='Unnamed: 0')
```

In [9]:
```
data1
```

Out[9]:

	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 [10]:
```
data1.isna().sum()
```

Out[10]:
```
TV           0
radio        0
newspaper    0
sales        0
dtype: int64
```

In [11]:
```
cor=data1.corr()
```

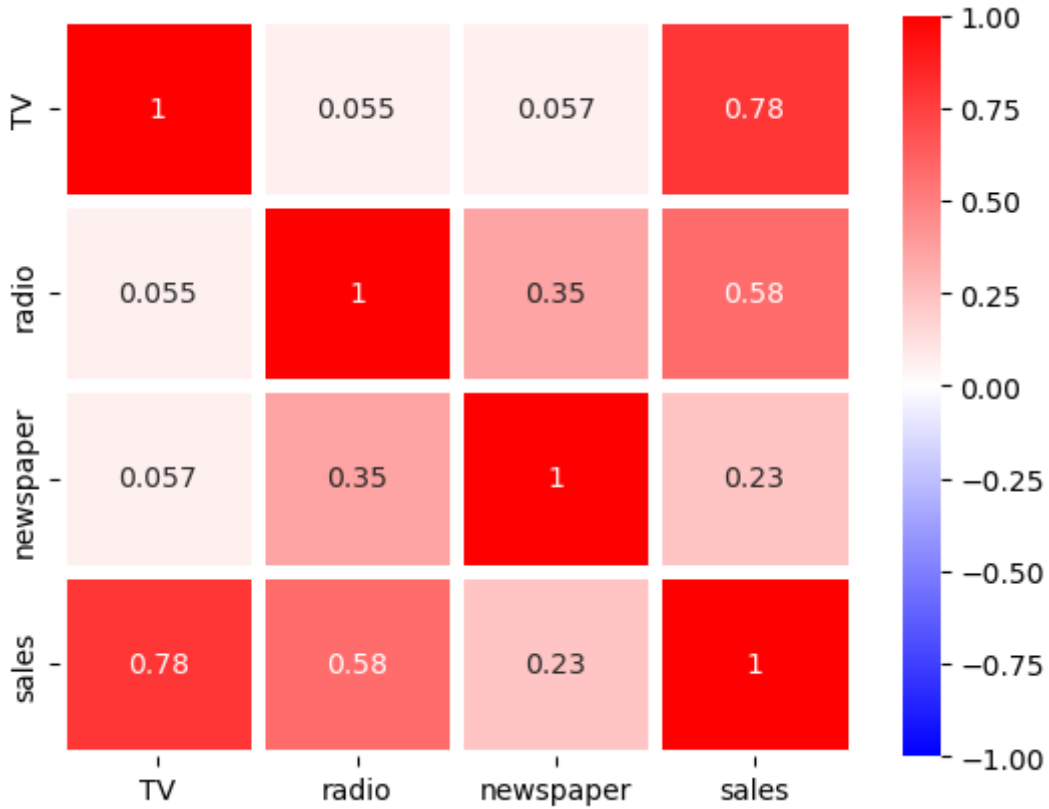
In [12]:
```
cor
```

Out[12]:

	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 [13]:
```
import seaborn as sns
sns.heatmap(cor,vmax=1,vmin=-1,annot=True,linewidth=5,cmap='bwr')
```

Out[13]: <Axes: >



In [14]:
```
y=data1['sales']
x=data1.drop(columns='sales')
```

In [15]:
```
x
```

Out[15]:

	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 [16]:
```
y
```

Out[16]:
```
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 [52]:
```
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 [53]:
```
x_train
```

Out[53]:

	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
...	...	...	...
106	25.0	11.0	29.7
14	204.1	32.9	46.0
92	217.7	33.5	59.0
179	165.6	10.0	17.6
102	280.2	10.1	21.4

134 rows × 3 columns

In [54]:
```
x_test
```

Out[54]:

	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

LinearRegression

In [55]:
```
from sklearn.linear_model import LinearRegression
reg=LinearRegression()
reg.fit(x_train,y_train)
```

Out[55]:
```
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 [56]:
```
y_pred_reg=reg.predict(x_test)
```

In [57]:
```
y_pred_reg
```

Out[57]:
```
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 [58]:
```
from sklearn.metrics import r2_score
r2_score(y_test,y_pred_reg)
```

Out[58]:
```
0.8555568430680086
```

In [59]:
```
from sklearn.metrics import mean_squared_error
mean_squared_error(y_test,y_pred_reg)
```

Out[59]:
```
3.7279283306815105
```

RidgeRegression

In [60]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import Ridge
alpha=[1e-15,1e-10,1e-4,1e-3,1e-2,15,10,20,30]
ridge=Ridge()                                        #creating an object for Ridge
parameters={'alpha':alpha}
ridge_regressor=GridSearchCV(ridge,parameters)
ridge_regressor.fit(x_train,y_train)                 #training and fitting
```

Out[60]:
```
GridSearchCV(estimator=Ridge(),
             param_grid={'alpha': [1e-15, 1e-10, 0.0001, 0.001, 0.01, 15, 10,
                                   20, 30]})
```

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 [61]:
```
ridge_regressor.best_params_
```

Out[61]:
```
{'alpha': 30}
```

In [62]:
```
ridge=Ridge(alpha=30)
ridge.fit(x_train,y_train)
y_pred_ridge=ridge.predict(x_test)
```

In [63]:
```
from sklearn.metrics import r2_score #to know the efficiency of the predicted price
r2_score(y_test,y_pred_ridge)
```

Out[63]:
```
0.8556430695733936
```

In [64]:
```
from sklearn.metrics import mean_squared_error
mean_squared_error(y_test,y_pred_ridge)
```

Out[64]:
```
3.7257029138524405
```

In [ ]:
```

```

ElasticRegression

In [65]:
```
from sklearn.linear_model import ElasticNet
from sklearn.model_selection import GridSearchCV
elastic=ElasticNet()                                  #creating an object for ElasticNet
parameters={'alpha':[1e-15,1e-10,1e-8,1e-4,1e-3,1e-2,15,10,20]}
elastic_regressor=GridSearchCV(elastic,parameters)
elastic_regressor.fit(x_train,y_train)    #training and fitting
```

Out[65]:
```
GridSearchCV(estimator=ElasticNet(),
             param_grid={'alpha': [1e-15, 1e-10, 1e-08, 0.0001, 0.001, 0.01, 15,
                                   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 [66]:
```
elastic_regressor.best_params_
```

Out[66]:
```
{'alpha': 0.01}
```

In [67]:
```
elastic=ElasticNet(alpha=0.01)
elastic.fit(x_train,y_train)
y_pred_enet=elastic.predict(x_test)
```

In [68]:
```
from sklearn.metrics import r2_score #to know the efficiency of the predicted price
r2_score(y_test,y_pred_enet)
```

Out[68]:
```
0.855576715693211
```

In [69]:
```
from sklearn.metrics import mean_squared_error
mean_squared_error(y_test,y_pred_enet)
```

Out[69]:
```
3.7274154388002283
```

In [80]:
```
test=[[110,33,21],[220,6,13]]
y_pred_enet=elastic.predict(test)
```

In [81]:
```
y_pred_enet   #sales we get from testdata
```

Out[81]:
```
array([14.28742973, 13.84367754])
```

In [ ]: