In [37]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import preprocessing,svm
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Lasso
from sklearn.linear_model import Ridge
from sklearn.preprocessing import StandardScaler


df=pd.read_csv(r"C:\Users\DELL\Downloads\Advertising.csv")
df
```

Out[37]:

	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

200 rows × 4 columns

In [38]:

```
df.head(10)
```

Out[38]:

	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
5	8.7	48.9	75.0	7.2
6	57.5	32.8	23.5	11.8
7	120.2	19.6	11.6	13.2
8	8.6	2.1	1.0	4.8
9	199.8	2.6	21.2	15.6

In [39]:

```
df.tail(10)
```

Out[39]:

	TV	Radio	Newspaper	Sales
190	39.5	41.1	5.8	10.8
191	75.5	10.8	6.0	11.9
192	17.2	4.1	31.6	5.9
193	166.8	42.0	3.6	19.6
194	149.7	35.6	6.0	17.3
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

In [40]:

```
df.query("TV>50")
```

Out[40]:

	TV	Radio	Newspaper	Sales
0	230.1	37.8	69.2	22.1
3	151.5	41.3	58.5	16.5
4	180.8	10.8	58.4	17.9
6	57.5	32.8	23.5	11.8
7	120.2	19.6	11.6	13.2
...	...	...	...	...
194	149.7	35.6	6.0	17.3
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

163 rows × 4 columns

In [41]:

```
df.sort_values("Radio")
```

Out[41]:

	TV	Radio	Newspaper	Sales
127	80.2	0.0	9.2	11.9
107	90.4	0.3	23.2	12.0
108	13.1	0.4	25.6	5.3
117	76.4	0.8	14.8	9.4
157	149.8	1.3	24.3	10.1
...	...	...	...	...
128	220.3	49.0	3.2	24.7
147	243.2	49.0	44.3	25.4
37	74.7	49.4	45.7	14.7
55	198.9	49.4	60.0	23.7
58	210.8	49.6	37.7	23.8

200 rows × 4 columns

In [42]:

```
df.nlargest(10,"Radio")
```

Out[42]:

	TV	Radio	Newspaper	Sales
58	210.8	49.6	37.7	23.8
37	74.7	49.4	45.7	14.7
55	198.9	49.4	60.0	23.7
128	220.3	49.0	3.2	24.7
147	243.2	49.0	44.3	25.4
5	8.7	48.9	75.0	7.2
175	276.9	48.9	41.8	27.0
89	109.8	47.8	51.4	16.7
15	195.4	47.7	52.9	22.4
135	48.3	47.0	8.5	11.6

In [43]:

```
df.nsmallest(5,"TV")
```

Out[43]:

	TV	Radio	Newspaper	Sales
130	0.7	39.6	8.7	1.6
155	4.1	11.6	5.7	3.2
78	5.4	29.9	9.4	5.3
56	7.3	28.1	41.4	5.5
126	7.8	38.9	50.6	6.6

In [44]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   TV         200 non-null    float64
 1   Radio      200 non-null    float64
 2   Newspaper  200 non-null    float64
 3   Sales      200 non-null    float64
dtypes: float64(4)
memory usage: 6.4 KB
```

In [45]:

```
df.isnull().sum()
```

Out[45]:

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

In [46]:

```
df.describe()
```

Out[46]:

	TV	Radio	Newspaper	Sales
count	200.000000	200.000000	200.000000	200.000000
mean	147.042500	23.264000	30.554000	15.130500
std	85.854236	14.846809	21.778621	5.283892
min	0.700000	0.000000	0.300000	1.600000
25%	74.375000	9.975000	12.750000	11.000000
50%	149.750000	22.900000	25.750000	16.000000
75%	218.825000	36.525000	45.100000	19.050000
max	296.400000	49.600000	114.000000	27.000000

In [47]:

```
df=df[["TV","Sales"]]
df.columns=['tv','sales']
```

In [48]:

```
df.head(10)
```

Out[48]:

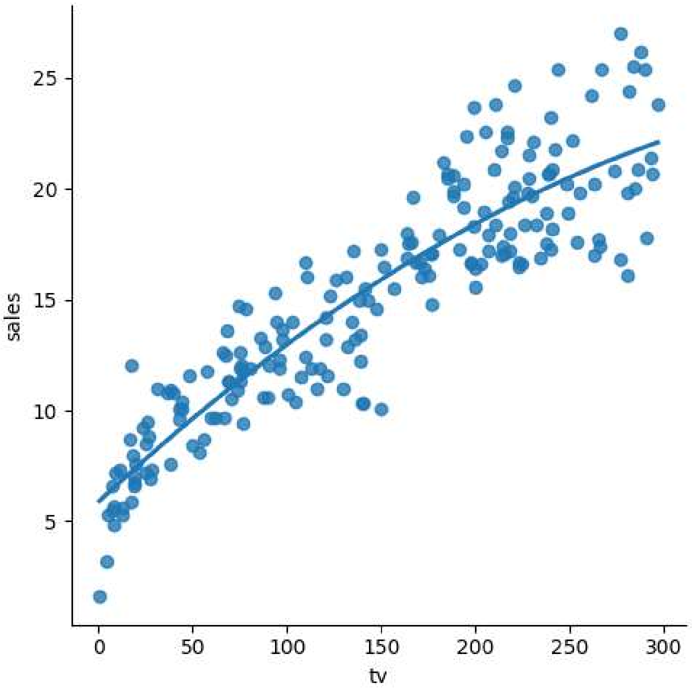
	tv	sales
0	230.1	22.1
1	44.5	10.4
2	17.2	12.0
3	151.5	16.5
4	180.8	17.9
5	8.7	7.2
6	57.5	11.8
7	120.2	13.2
8	8.6	4.8
9	199.8	15.6

In [49]:

```
sns.lmplot(x='tv',y='sales',data=df,order=2,ci=None)
```

Out[49]:

```
<seaborn.axisgrid.FacetGrid at 0x181ca9fb150>
```



In [50]:
```
df.fillna(method='ffill')
```

Out[50]:

	tv	sales
0	230.1	22.1
1	44.5	10.4
2	17.2	12.0
3	151.5	16.5
4	180.8	17.9
...	...	...
195	38.2	7.6
196	94.2	14.0
197	177.0	14.8
198	283.6	25.5
199	232.1	18.4

200 rows × 2 columns

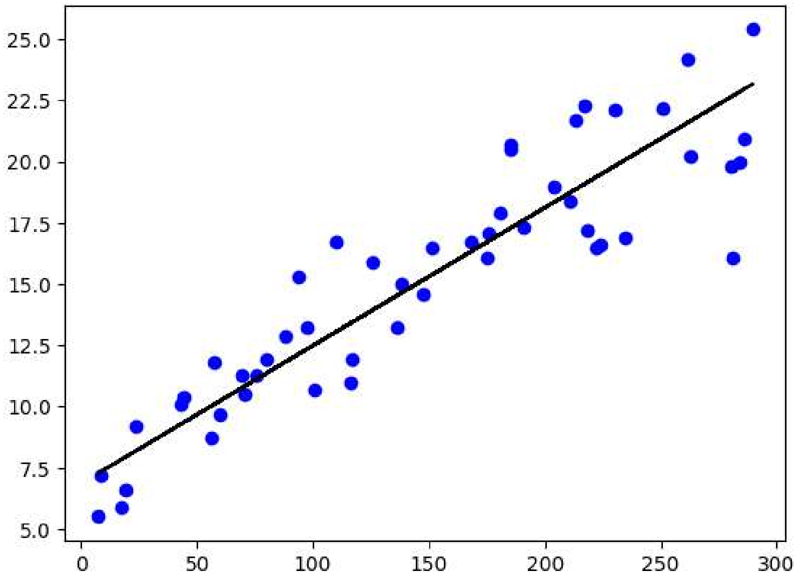
In [51]:
```
x=np.array(df['tv']).reshape(-1,1)
y=np.array(df['sales']).reshape(-1,1)
```

In [52]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)
```

In [53]:
```
regr=LinearRegression()
regr.fit(x_train,y_train)
print(regr.score(x_test,y_test))
```

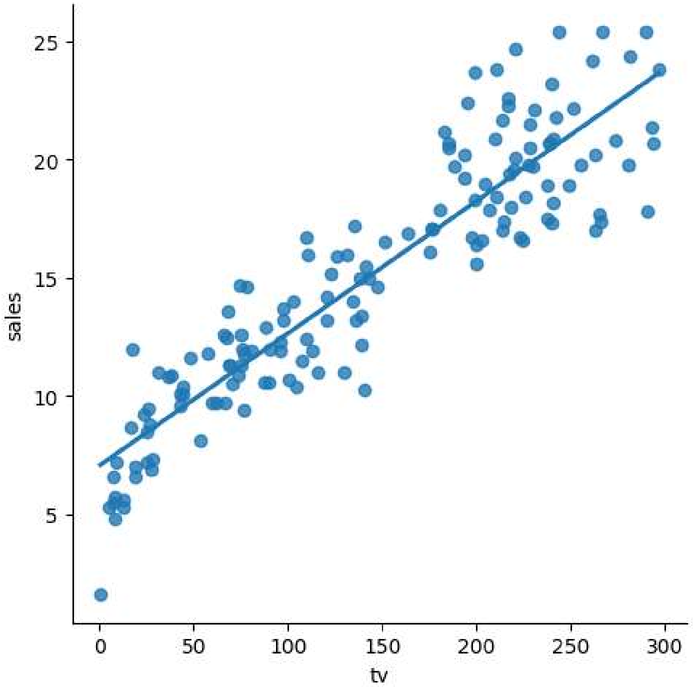
```
0.8190555254437374
```

In [54]:
```
y_pred=regr.predict(x_test)
plt.scatter(x_test,y_test,color='b')
plt.plot(x_test,y_pred,color='k')
plt.show()
```



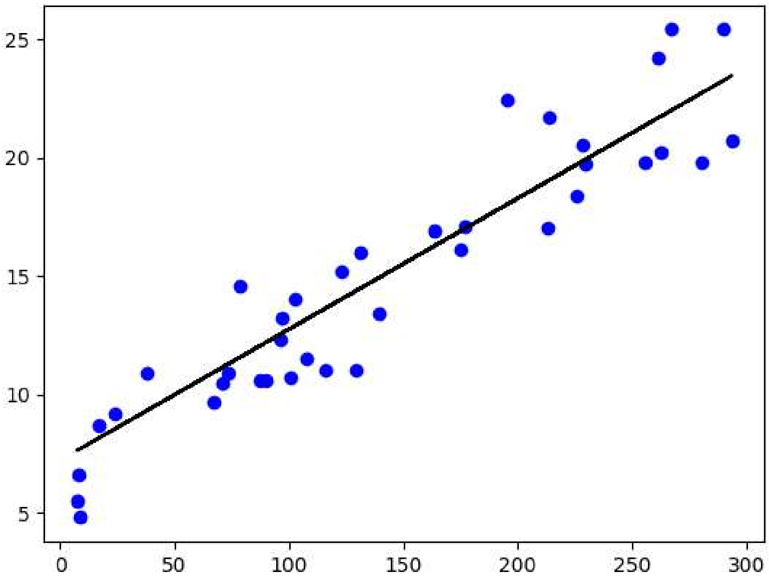
In [55]:
```
df150=df[:][:150]
sns.lmplot(x='tv',y='sales',data=df150,order=1,ci=None)
```

Out[55]:
```
<seaborn.axisgrid.FacetGrid at 0x181cacaaa90>
```



In [56]:
```
df150.fillna(method='ffill',inplace=True)
x=np.array(df150['tv']).reshape(-1,1)
y=np.array(df150['sales']).reshape(-1,1)
df150.dropna(inplace=True)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)
regr=LinearRegression()
regr.fit(x_train,y_train)
print("Regression:",regr.score(x_test,y_test))
y_pred=regr.predict(x_test)
plt.scatter(x_test,y_test,color='b')
plt.plot(x_test,y_pred,color='k')
plt.show()
```

```
Regression: 0.8718357652786353
```



In [57]:

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
model=LinearRegression()
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
r2=r2_score(y_test,y_pred)
print("R2 score:",r2)
```

```
R2 score: 0.8718357652786353
```

In [58]:

```
#conclusion:this model is best fit for linear regression
```

In [59]:

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import RidgeCV
ridge_cv=RidgeCV(alphas=[0.0001,0.001,0.01,0.1,1,10]).fit(x_train,y_train)
print("The train score for ridge model is {}".format(ridge_cv.score(x_train,y_train)))
print("The train score for ridge model is {}".format(ridge_cv.score(x_test,y_test)))
```

```
The train score for ridge model is 0.8142438327763156
The train score for ridge model is 0.871834693279092
```

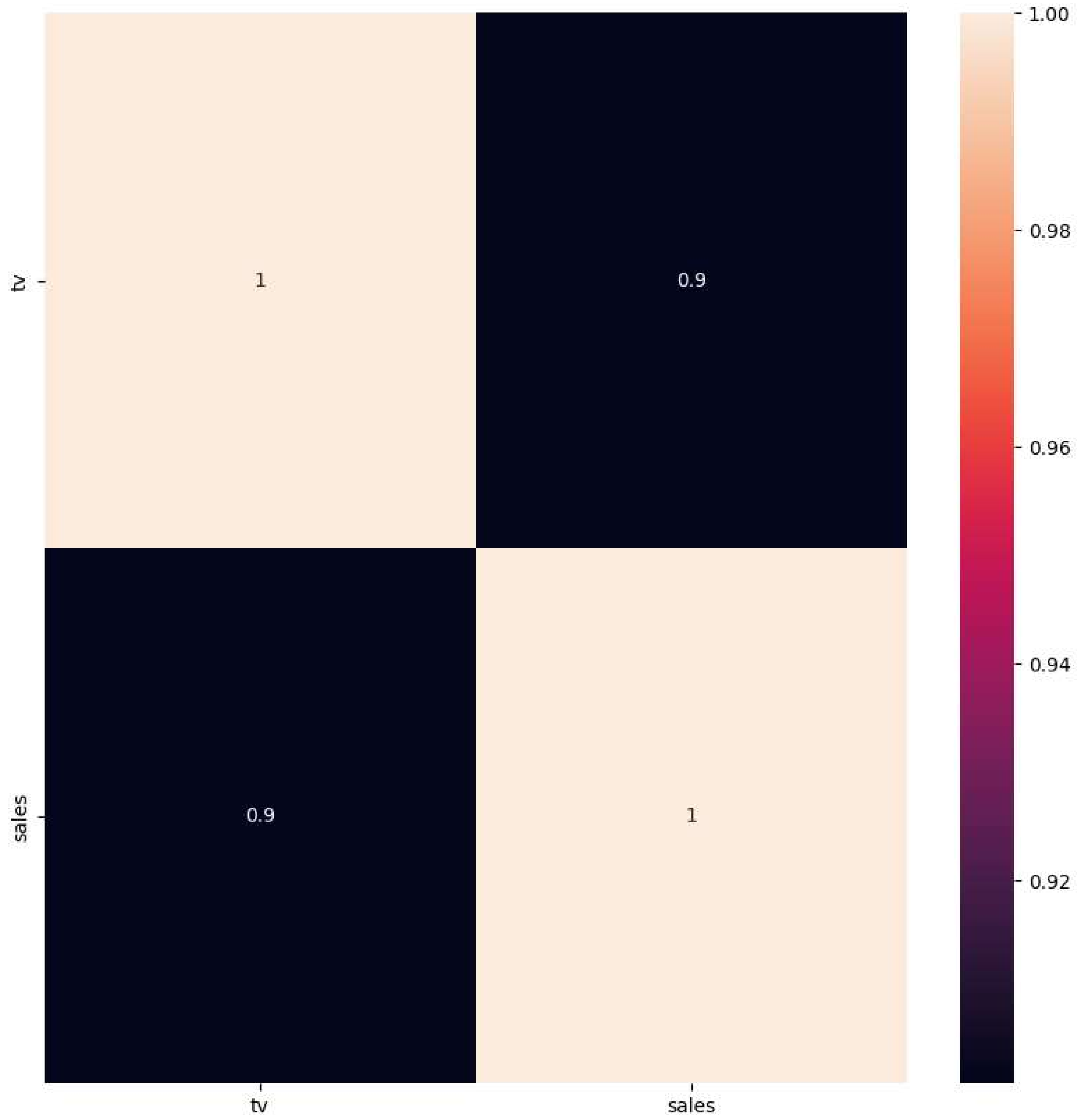
In [60]:

```
ridgeReg=Ridge(alpha=10)
ridgeReg.fit(x_train,y_train)
train_score_ridge=ridgeReg.score(x_train,y_train)
test_score_ridge=ridgeReg.score(x_train,y_train)
print("Ridge Model")
print("The train score for ridge model is {}".format(train_score_ridge))
print("The test score for ridge model is {}".format(test_score_ridge))
```

```
Ridge Model
The train score for ridge model is 0.8142438327763157
The test score for ridge model is 0.8142438327763157
```

In [61]:
```
plt.figure(figsize = (10, 10))
sns.heatmap(df.corr(), annot = True)
```

Out[61]:
```
<Axes: >
```



In [62]:
```
features = df.columns[0:2]
target = df.columns[-1]
#X and y values
X = df[features].values
y = df[target].values
#splot
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=17)
print("The dimension of X_train is {}".format(X_train.shape))
print("The dimension of X_test is {}".format(X_test.shape))
#Scale features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
The dimension of X_train is (140, 2)
The dimension of X_test is (60, 2)
```

In [63]:

```
lr = LinearRegression()
#Fit model
lr.fit(X_train, y_train)
#predict
#prediction = lr.predict(X_test)
#actual
actual = y_test
train_score_lr = lr.score(X_train, y_train)
test_score_lr = lr.score(X_test, y_test)
print("\nLinear Regression Model:\n")
print("The train score for lr model is {}".format(train_score_lr))
print("The test score for lr model is {}".format(test_score_lr))
```

```
Linear Regression Model:

The train score for lr model is 1.0
The test score for lr model is 1.0
```

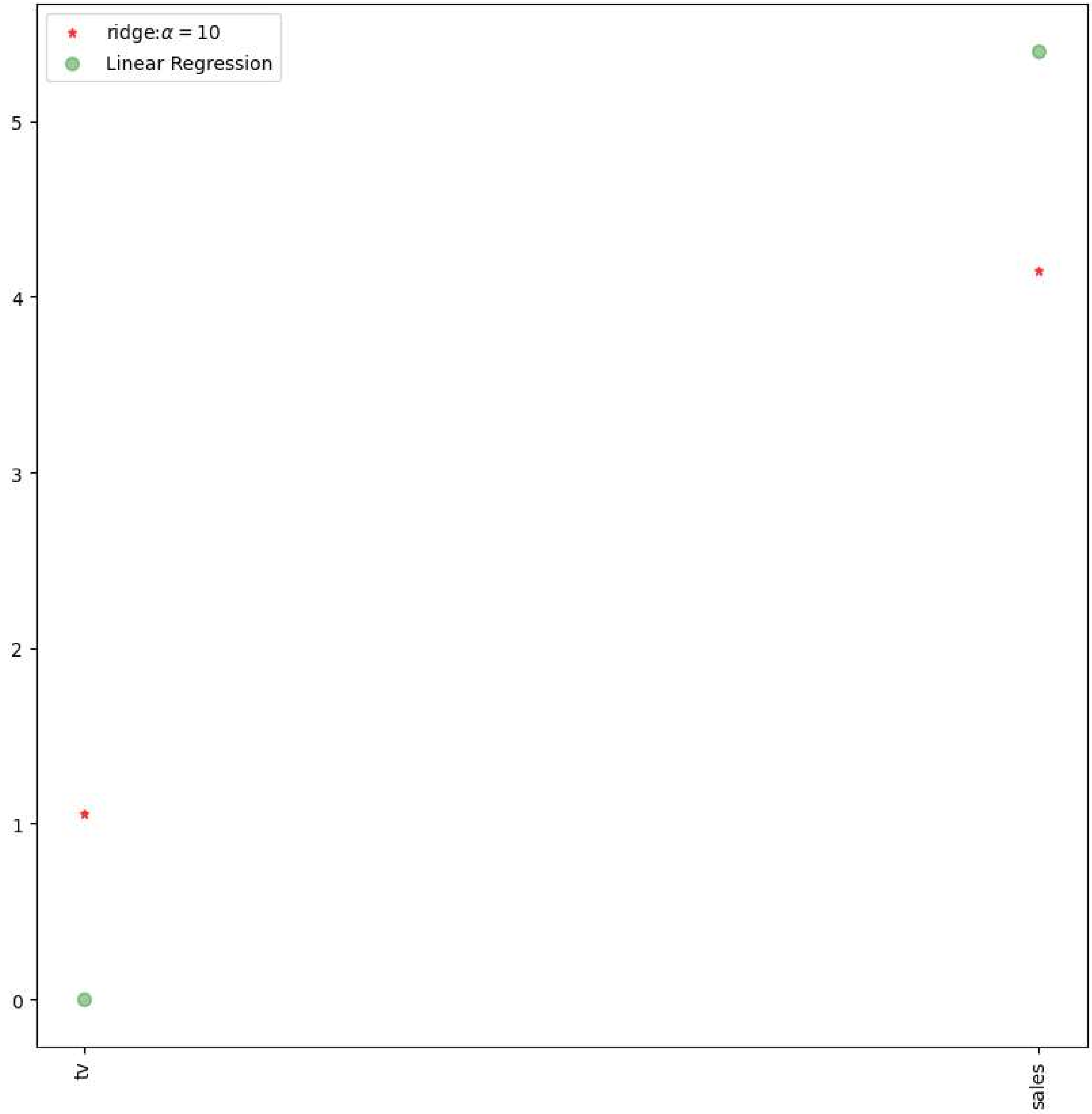
In [64]:

```
ridgeReg = Ridge(alpha=10)
ridgeReg.fit(X_train,y_train)
#train and test scorefor ridge regression
train_score_ridge = ridgeReg.score(X_train, y_train)
test_score_ridge = ridgeReg.score(X_test, y_test)
print("\nRidge Model:\n")
print("The train score for ridge model is {}".format(train_score_ridge))
print("The test score for ridge model is {}".format(test_score_ridge))
```

```
Ridge Model:

The train score for ridge model is 0.9900167746680466
The test score for ridge model is 0.9888279083610404
```

In [65]:

```
plt.figure(figsize=(10,10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5,color='red',label=r'ridge:$\alpha
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color='green',label='Linear Regression'
plt.xticks(rotation=90)
plt.legend()
plt.show()
```



In [66]:

```
#Lasso regression model
print("\nLasso Model: \n")
lasso = Lasso(alpha = 10)
lasso.fit(X_train,y_train)
train_score_ls =lasso.score(X_train,y_train)
test_score_ls =lasso.score(X_test,y_test)
print("The train score for ls model is {}".format(train_score_ls))
print("The test score for ls model is {}".format(test_score_ls))
```

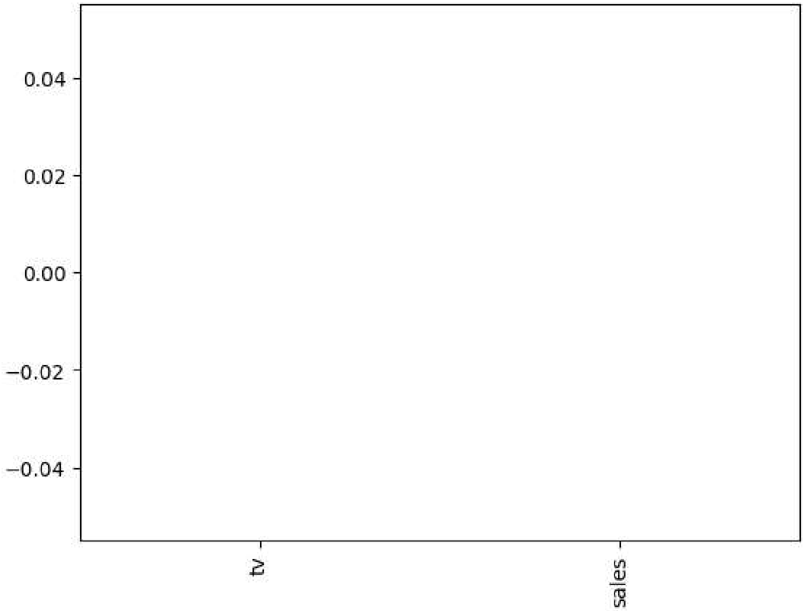
```
Lasso Model:

The train score for ls model is 0.0
The test score for ls model is -0.0064111102763571015
```

In [67]:

```
pd.Series(lasso.coef_, features).sort_values(ascending = True).plot(kind = "bar")
```

Out[67]:

```
<Axes: >
```



In [68]:

```
from sklearn.linear_model import LassoCV
#Lasso Cross validation
lasso_cv = LassoCV(alphas = [0.0001, 0.001,0.01, 0.1, 1, 10], random_state=0).fit(X_train, y_train)
#score
print(lasso_cv.score(X_train, y_train))
print(lasso_cv.score(X_test, y_test))
```

```
0.9999999677147366
0.9999999641980227
```

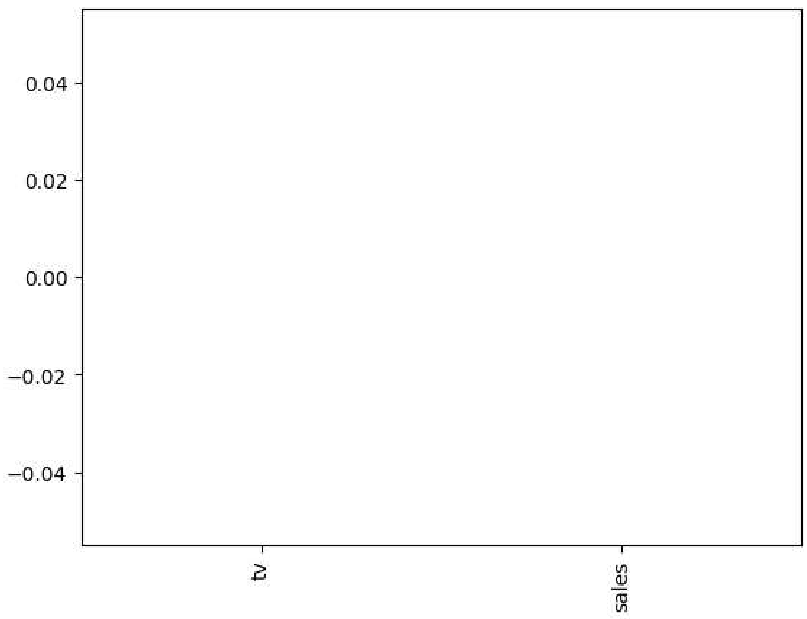
In [69]:

```
#Lasso regression model
print("\nLasso Model: \n")
lasso = Lasso(alpha = 10)
lasso.fit(X_train,y_train)
train_score_ls =lasso.score(X_train,y_train)
test_score_ls =lasso.score(X_test,y_test)
print("The train score for ls model is {}".format(train_score_ls))
print("The test score for ls model is {}".format(test_score_ls))
```

```

Lasso Model: 

The train score for ls model is 0.0
The test score for ls model is -0.0064111102763571015
```

```
In [70]: pd.Series(lasso.coef_, features).sort_values(ascending = True).plot(kind = "bar")
```

```
Out[70]: <Axes: >
```

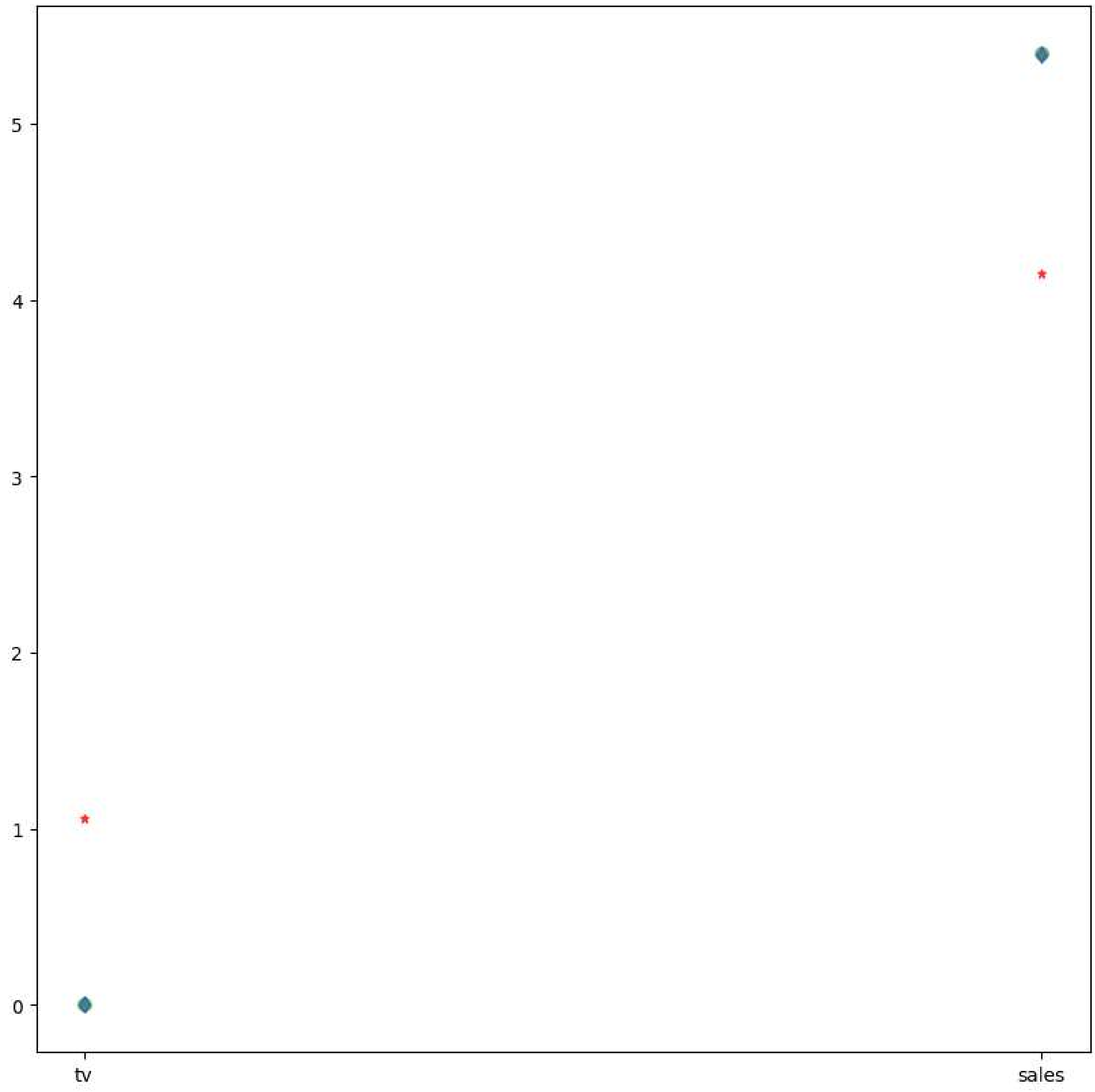


In [71]:

```

ze = (10, 10))
dge regression
s,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5,color='red',label=r'Ridge; $\alppha = 10$',zorder=
sso regression
v.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color='blue',label=r'lasso; $\alpha = grid$')
near model
s,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color='green',label='Linear Regression')
```

Out[71]:

```
[<matplotlib.lines.Line2D at 0x181ca03f010>]
```



Elastic Net

In [72]:

```
from sklearn.linear_model import ElasticNet
regr=ElasticNet()
regr.fit(X,y)
print(regr.coef_)
print(regr.intercept_)
```

```
[0.00938134 0.82969623]
1.197325903826
```

In [73]:

```
y_pred_elastic=regr.predict(X_train)
```

In [74]:

```
mean_squared_error=np.mean((y_pred_elastic-y_train)**2)
print("Mean Squared Error on test set",mean_squared_error)
```

```
Mean Squared Error on test set 218.26629572962375
```

In [ ]: