In [29]:

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

In [30]:

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

Out[30]:

```
        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 [31]:

```
k.describe()
```

Out[31]:

	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 [32]:

```
k.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 [33]:

```
k.head()
```

Out[33]:

	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

In [34]:

```
k.columns
```

Out[34]:

```
Index(['TV', 'Radio', 'Newspaper', 'Sales'], dtype='object')
```

In [35]:

```
k.shape
```

Out[35]:

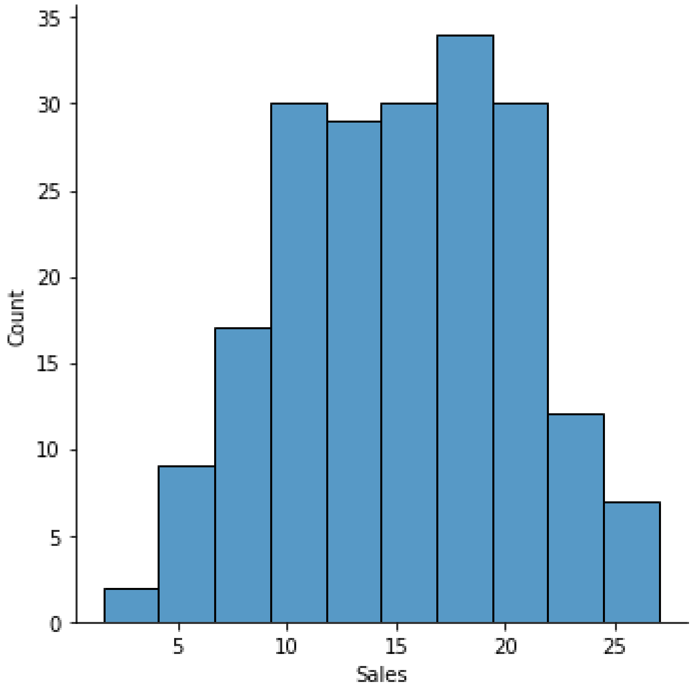
```
(200, 4)
```

In [37]:

```
sns.displot(k['Sales'])
```

Out[37]:

```
<seaborn.axisgrid.FacetGrid at 0x23a8628ec40>
```

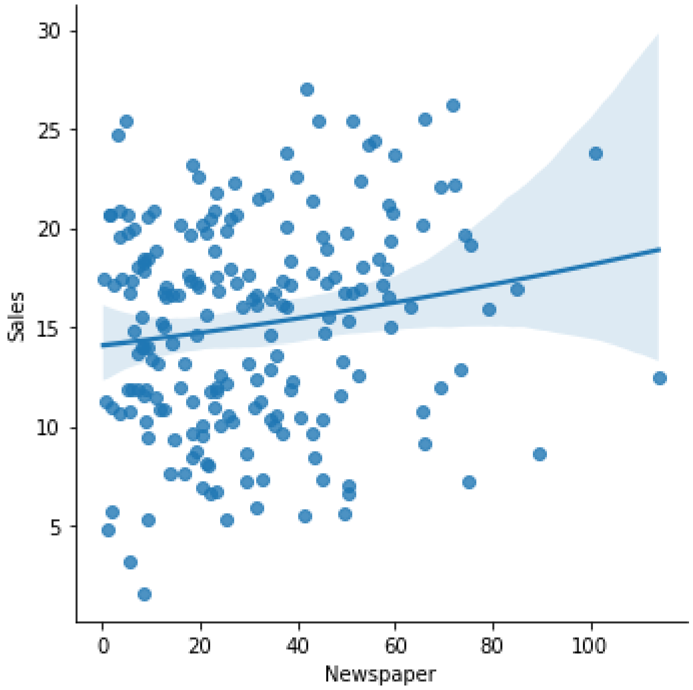


In [38]:

```
sns.lmplot(y='Sales',x='Newspaper',data=k,order=2)
```

Out[38]:

```
<seaborn.axisgrid.FacetGrid at 0x23a86a18970>
```

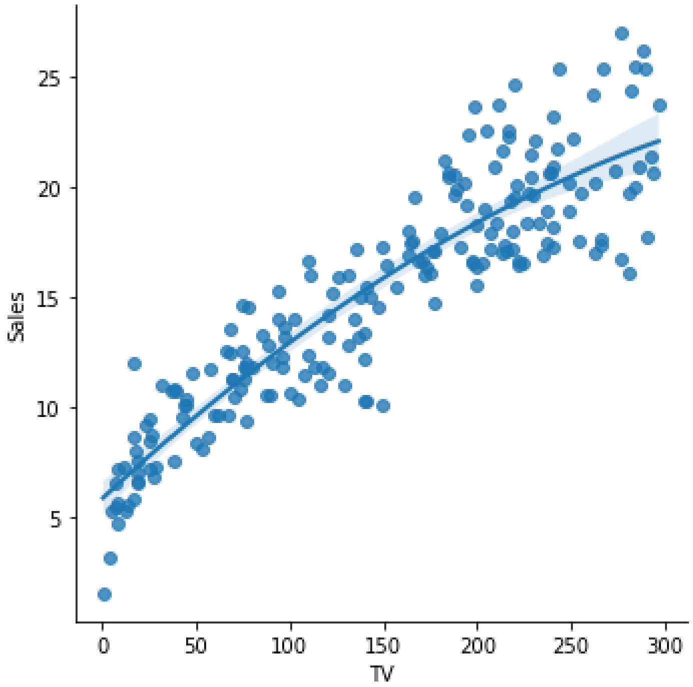


In [39]:

```
sns.lmplot(y='Sales',x='TV',data=k,order=2)
```

Out[39]:

```
<seaborn.axisgrid.FacetGrid at 0x23a86c62f40>
```

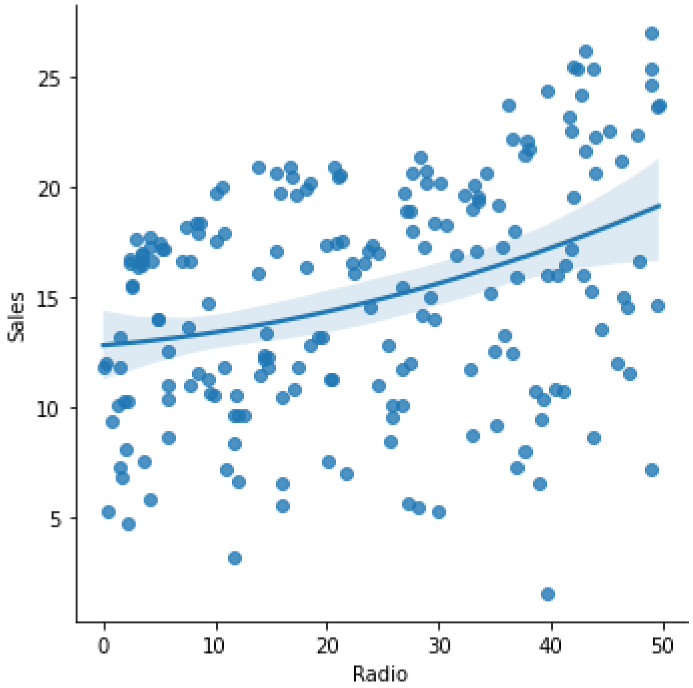


In [40]:

```
sns.lmplot(y='Sales',x='Radio',data=k,order=2)
```

Out[40]:

```
<seaborn.axisgrid.FacetGrid at 0x23a86d1b130>
```



In [41]:

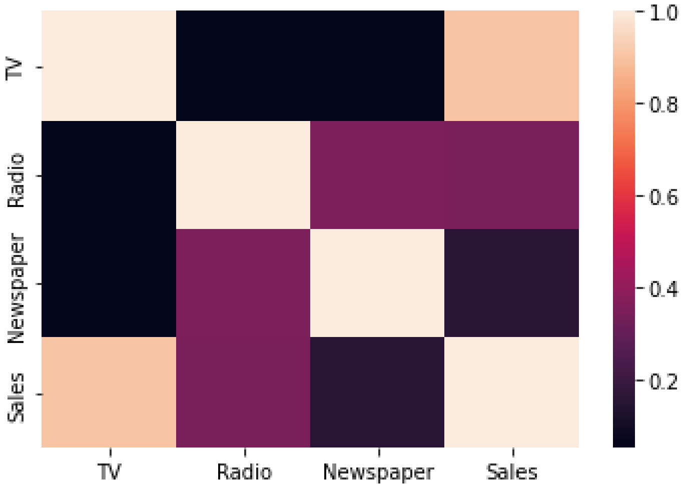
```
hk=k[['TV', 'Radio', 'Newspaper', 'Sales']]
```

In [42]:

```
sns.heatmap(hk.corr())
```

Out[42]:

```
<AxesSubplot:>
```



In [43]:

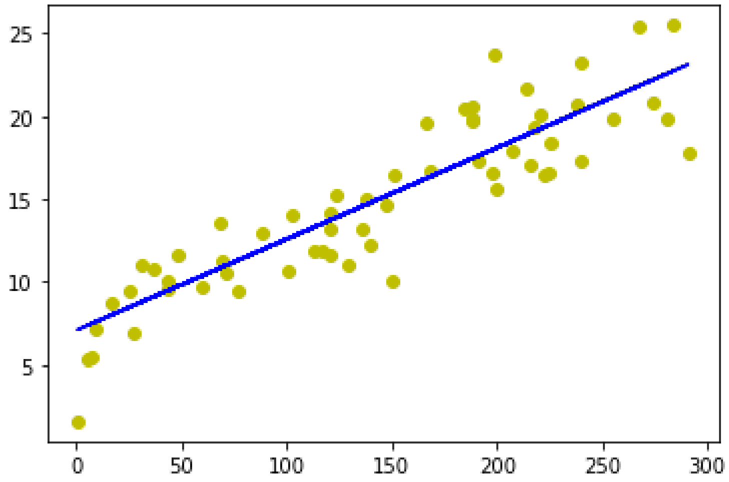
```
k.fillna(method='ffill',inplace=True)
regr=LinearRegression()
x=np.array(k['TV']).reshape(-1,1)
y=np.array(k['Sales']).reshape(-1,1)
k.dropna(inplace=True)
X_train,X_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
regr.fit(X_train,y_train)
regr.fit(X_train,y_train)
```

Out[43]:

```
LinearRegression()
```

In [44]:

```
y_pred=regr.predict(X_test)
plt.scatter(X_test,y_test,color='y')
plt.plot(X_test,y_pred,color='b')
plt.show()
```

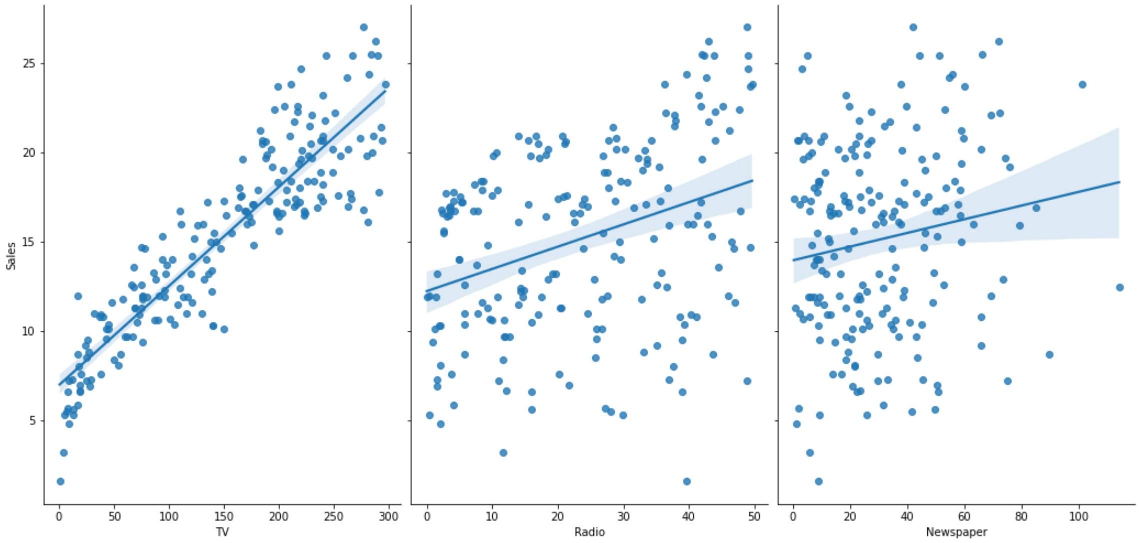


In [45]:

```
sns.pairplot(k,x_vars=['TV', 'Radio', 'Newspaper'],y_vars='Sales',height=7,aspect=0.7,ki
```

Out[45]:

```
<seaborn.axisgrid.PairGrid at 0x23a86e4d5e0>
```



In [46]:

```
#accuracy
regr=LinearRegression()
regr.fit(X_train,y_train)
regr.fit(X_train,y_train)
print(regr.score(X_test,y_test))
```

```
0.7994861274416293
```

In [47]:

```


#ridge regression model
ridgeReg=Ridge(alpha=10)
ridgeReg.fit(X_train,y_train)
#train and test score for 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 0.8167435664946732
The test score for ridge model is 0.7994858234484326
```

In [74]:

```
#using the linear cv model for ridge regression
from sklearn.linear_model import RidgeCV
#ridge cross validation
ridge_cv=RidgeCV(alphas=[0.0001,0.001,0.01,0.1,1,10]).fit(X_train,y_train)
#score
print(ridge_cv.score(X_train,y_train))
print(ridge_cv.score(X_test,y_test))
```

```
0.9999999999976276
0.9999999999962478
```

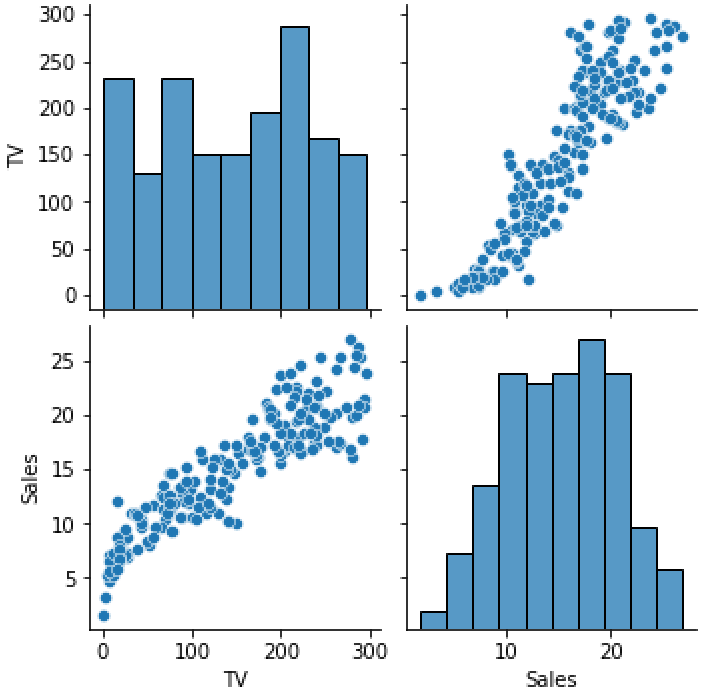
In [75]:

```
#using the linear cv model for lasso regression
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))
```

```
0.9999999343798134
0.9999999152638072
```

In [50]:

```
k.drop(columns = ["Radio", "Newspaper"], inplace = True)
#pairplot
sns.pairplot(k)
k.Sales = np.log(k.Sales)
```



In [54]:

```
features = k.columns[0:2]
target = k.columns[-1]
#X and y values
X = k[features].values
y = k[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 [56]:

```
#Model
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 [57]:

```
#Ridge Regression Model
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))
```

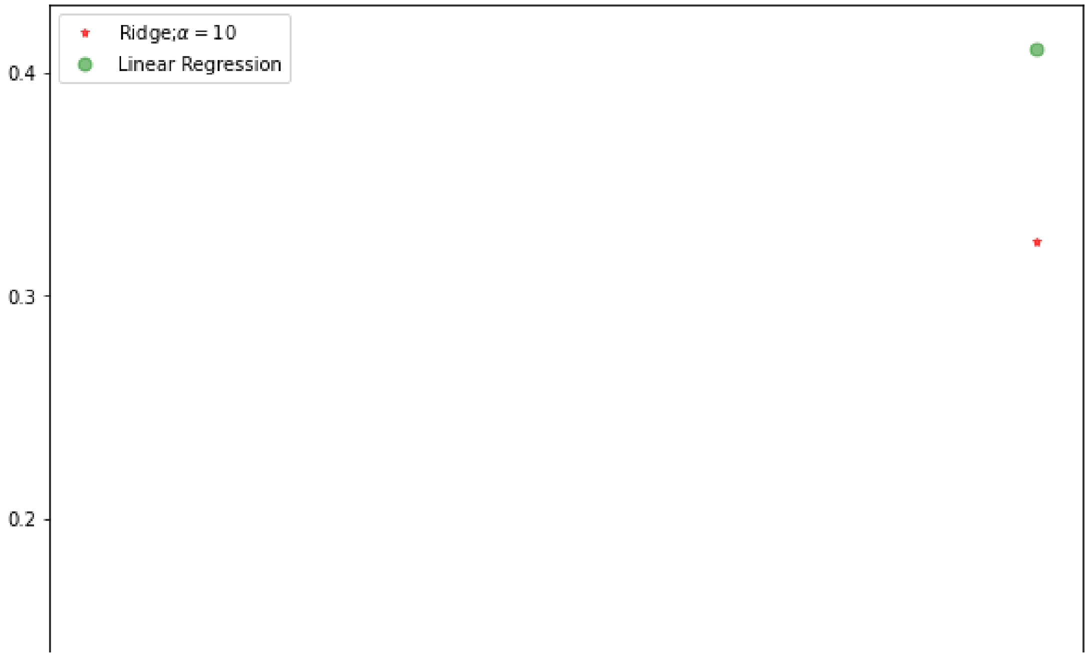
```

Ridge Model:

The train score for ridge model is 0.9902871391941607
The test score for ridge model is 0.9844266285141215
```

In [59]:

```
plt.figure(figsize=(10,10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',
         marker='*',markersize=5,color='red',
         label=r'Ridge;$\alpha=10$',zorder=7)
plt.plot(features,lr.coef_,alpha=0.5,linestyle='none',marker='o',
         markersize=7,color='green',
         label='Linear Regression')
plt.xticks(rotation=90)
plt.legend()
plt.show()
```



In [65]:

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

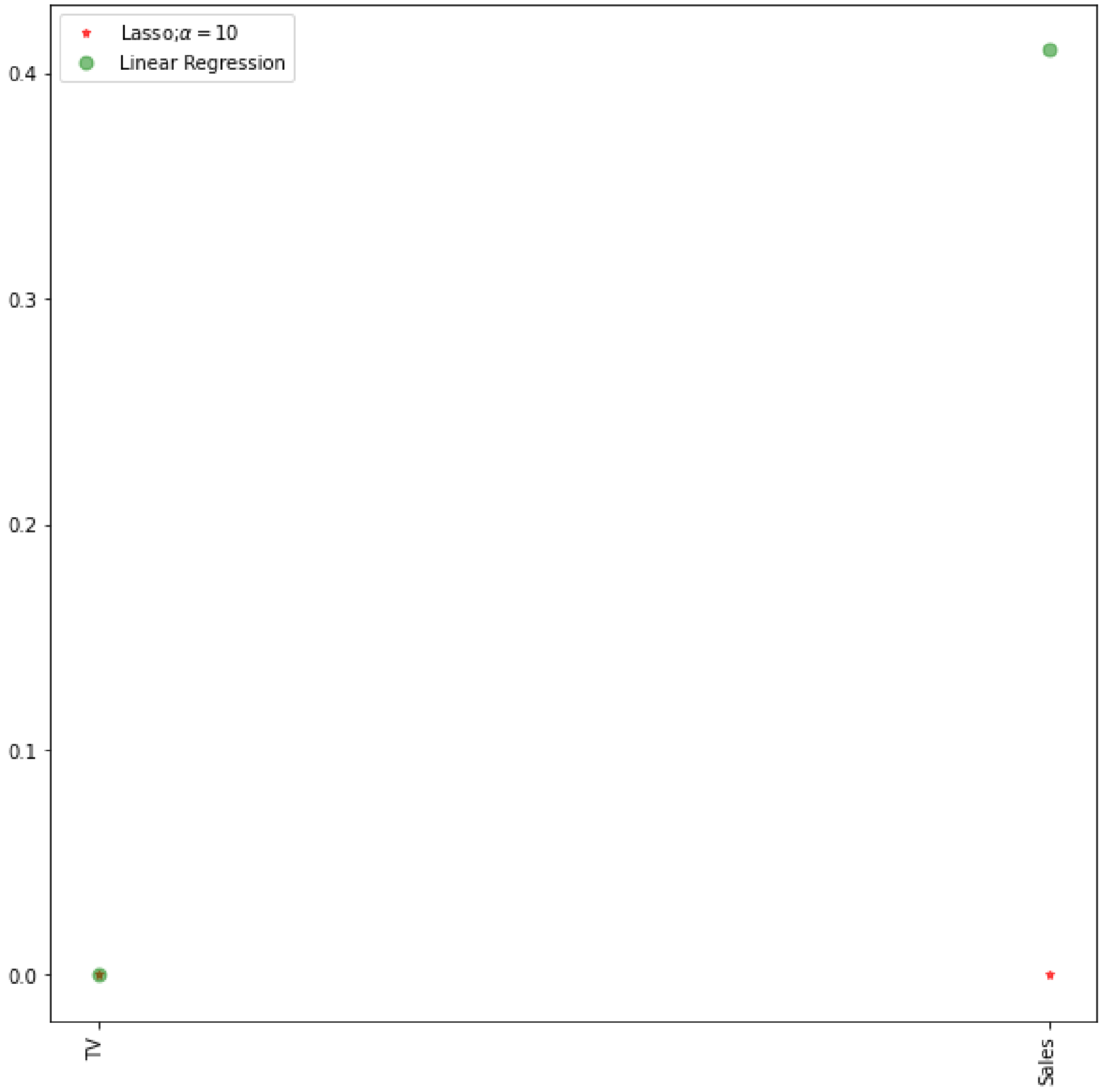
```

Lasso model:

The train score for lasso model is 0.0
The test score for lasso model is -0.0042092253233847465
```

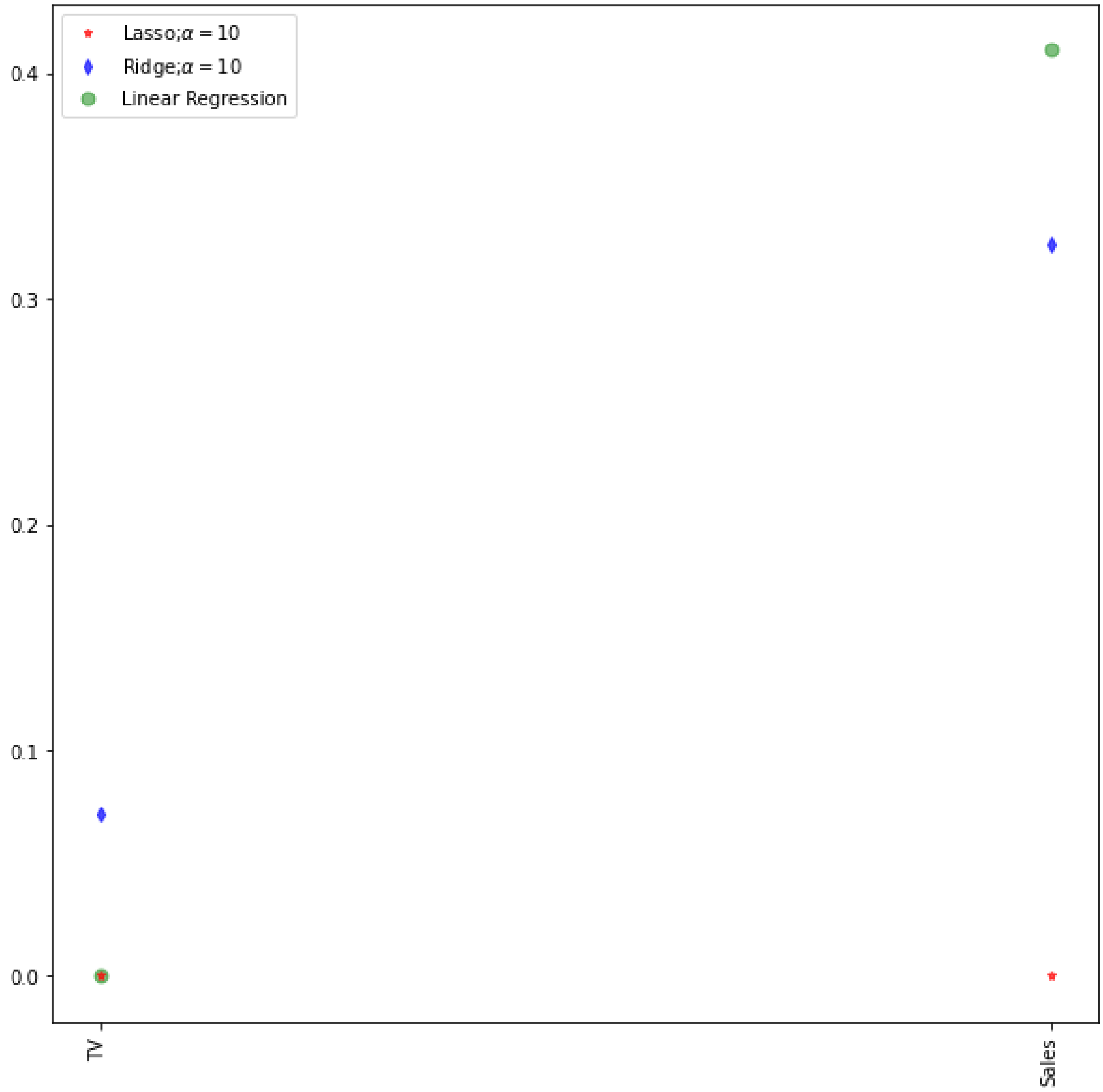
In [68]:

```
plt.figure(figsize=(10,10))
plt.plot(features,lassoReg.coef_,alpha=0.7,linestyle='none',
         marker='*',markersize=5,color='red',
         label=r'Lasso;$\alpha=10$')
plt.plot(features,lr.coef_,alpha=0.5,linestyle='none',marker='o',
         markersize=7,color='green',
         label='Linear Regression')
plt.xticks(rotation=90)
plt.legend()
plt.show()
```



In [70]:

```
plt.figure(figsize=(10,10))
#for lasso model
plt.plot(features,lassoReg.coef_,alpha=0.7,linestyle='none',
         marker='*',markersize=5,color='red',
         label=r'Lasso;$\alpha=10$',zorder=7)
#for ridge model
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',
         marker='d',markersize=5,color='b',
         label=r'Ridge;$\alpha=10$',zorder=7)
#for linear model
plt.plot(features,lr.coef_,alpha=0.5,linestyle='none',marker='o'
         ,markersize=7,color='green',
         label='Linear Regression')
#plottingg
plt.xticks(rotation=90)
plt.legend()
plt.show()
```

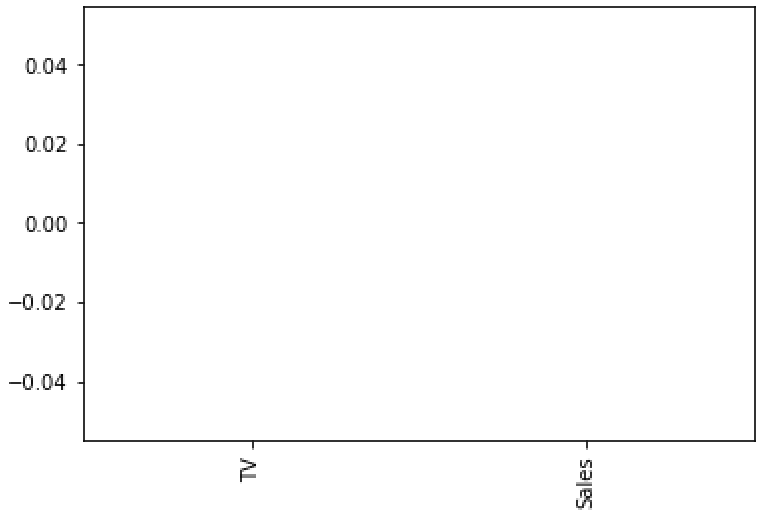


In [76]:

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

Out[76]:

```
<AxesSubplot:>
```



In [ ]:

```
```