In [1]:

```
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 [2]:

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

Out[2]:

	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 [3]:

```
k.describe()
```

Out[3]:

	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 [4]:

```
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 [5]:

```
k.head()
```

Out[5]:

	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 [6]:

```
k.columns
```

Out[6]:

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

In [7]:

```
k.shape
```

Out[7]:

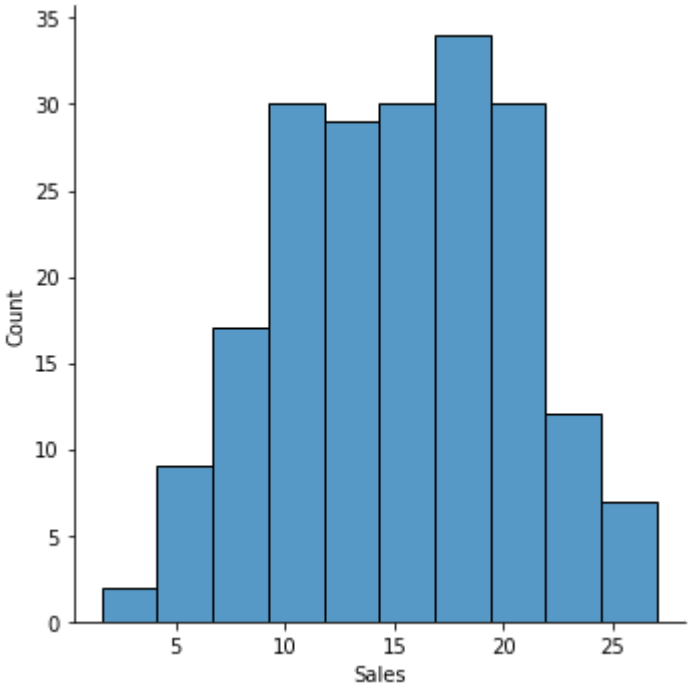
```
(200, 4)
```

In [8]:

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

Out[8]:

```
<seaborn.axisgrid.FacetGrid at 0x1885a74d070>
```

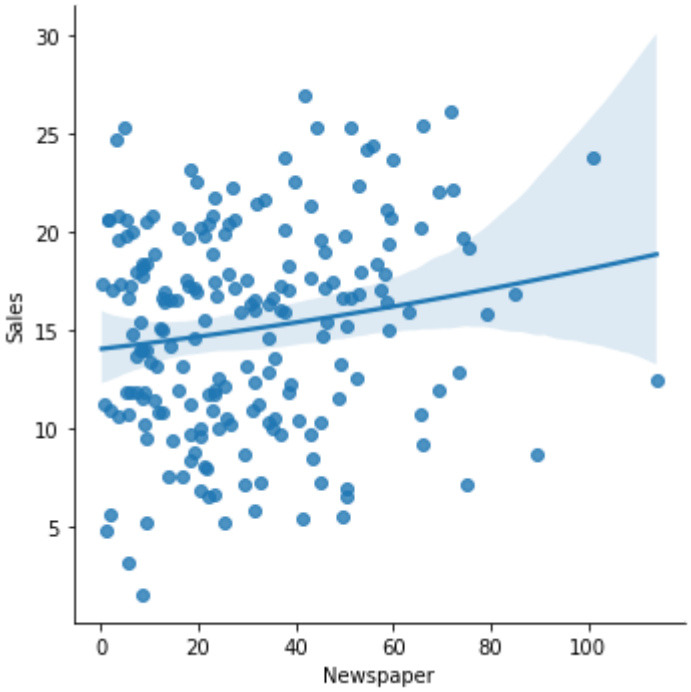


In [9]:

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

Out[9]:

```
<seaborn.axisgrid.FacetGrid at 0x1885a741430>
```

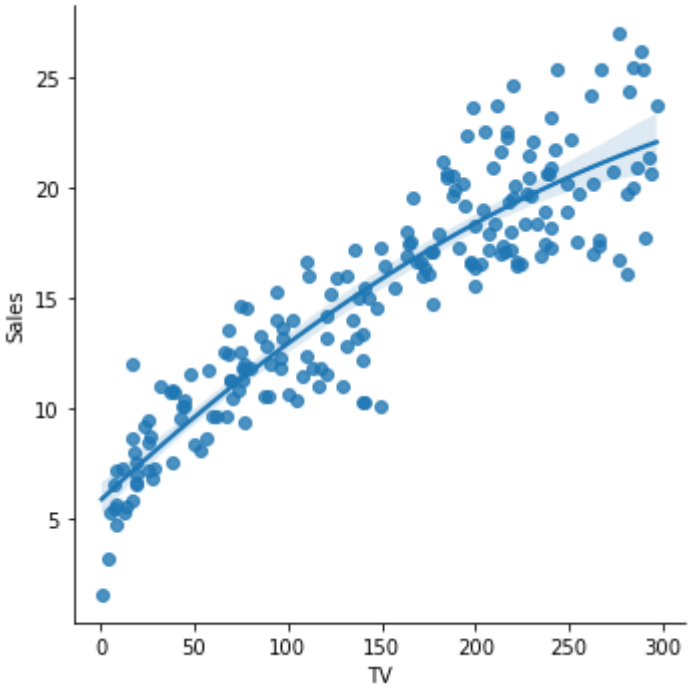


In [10]:

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

Out[10]:

```
<seaborn.axisgrid.FacetGrid at 0x1885a7e4f40>
```

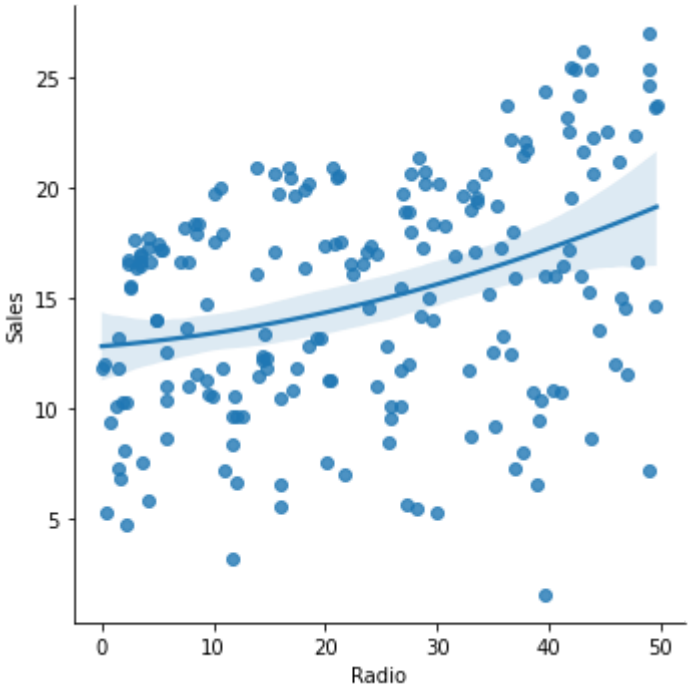


In [11]:

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

Out[11]:

```
<seaborn.axisgrid.FacetGrid at 0x1885a741e80>
```



In [12]:

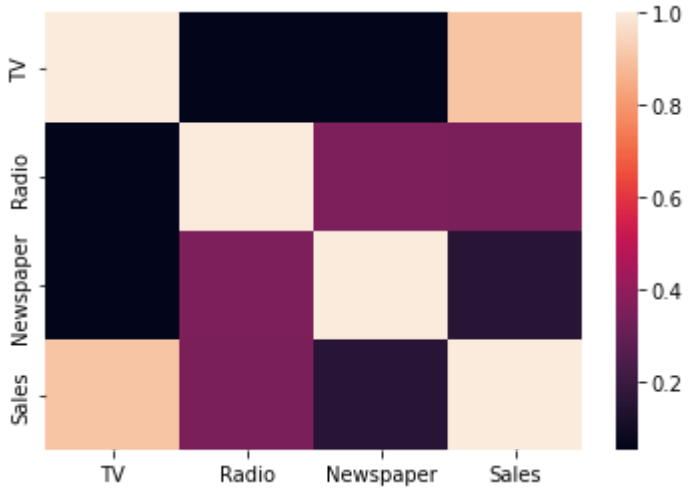
```
hk=k[['TV', 'Radio', 'Newspaper', 'Sales']]
```

In [13]:

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

Out[13]:

```
<AxesSubplot:>
```



In [14]:

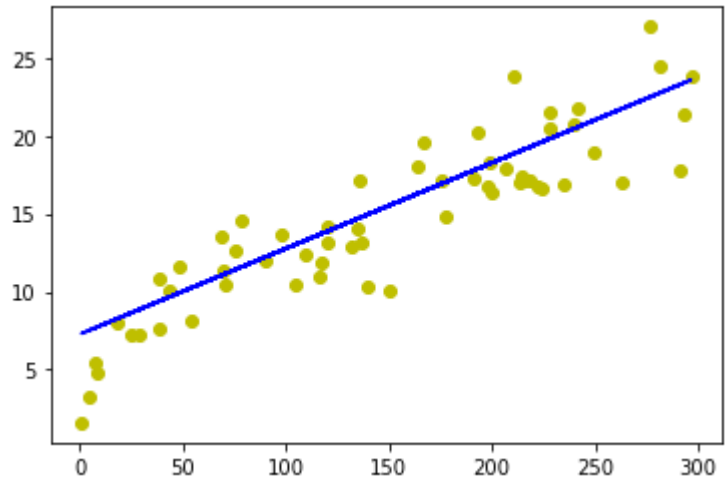
```
k.fillna(method='ffill',inplace=True)
regr=LinearRegression()
x=np.array(k['TV']).reshape(-1,1)
y=np.array(k['Sales']).reshape(-1,1)
k.dropna(inplace=True)
X_train,X_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
regr.fit(X_train,y_train)
regr.fit(X_train,y_train)
```

Out[14]:

```
LinearRegression()
```

In [15]:

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

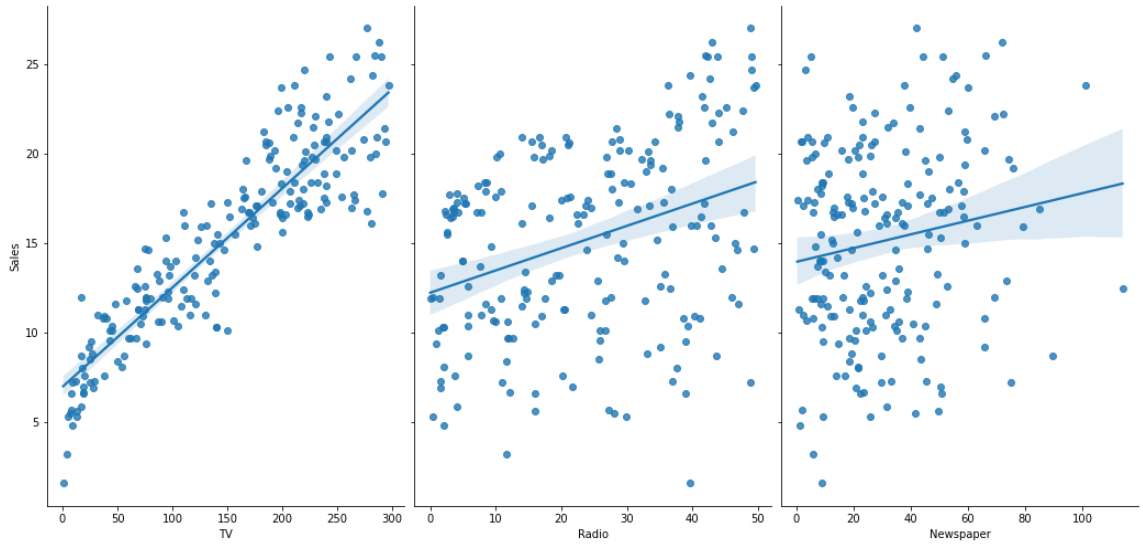


In [16]:

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

Out[16]:

```
<seaborn.axisgrid.PairGrid at 0x1885c1da880>
```



In [17]:

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

```
0.786971701148369
```

In [18]:

```

#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 0.8196413133140588
The test score for ridge model is 0.7869712853611057
```

In [19]:

```
#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))
```

```
0.8196413133140585
0.7869712853605555
```

In [20]:

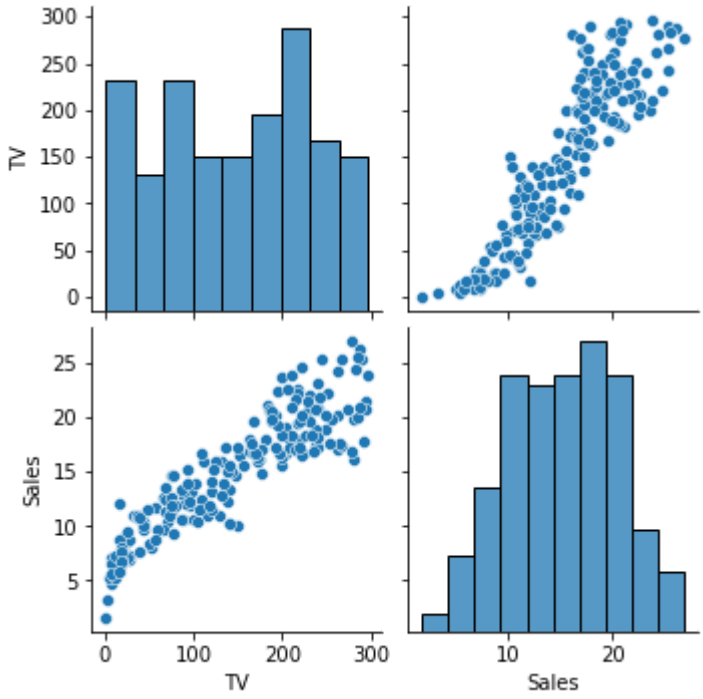
```
#using the linear cv model for lasso regression
from sklearn.linear_model import LassoCV
#lasso cross validation
lasso_cv=LassoCV(alphas=[0.0001,0.001,0.01,0.1,1,10],random_state=0).fit(X_train,y_train
#score
print(lasso_cv.score(X_train,y_train))
print(lasso_cv.score(X_test,y_test))
```

```
0.8196413133908109
0.7869716905645195

C:\Users\shaik\anaconda3\lib\site-packages\sklearn\linear_model\_coordinat
e_descent.py:1571: DataConversionWarning: A column-vector y was passed whe
n a 1d array was expected. Please change the shape of y to (n_samples, ),
for example using ravel().
  y = column_or_1d(y, warn=True)
```

In [21]:

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



In [22]:

```
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 [23]:

```
#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 [24]:

```
#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 [25]:

```
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 [26]:

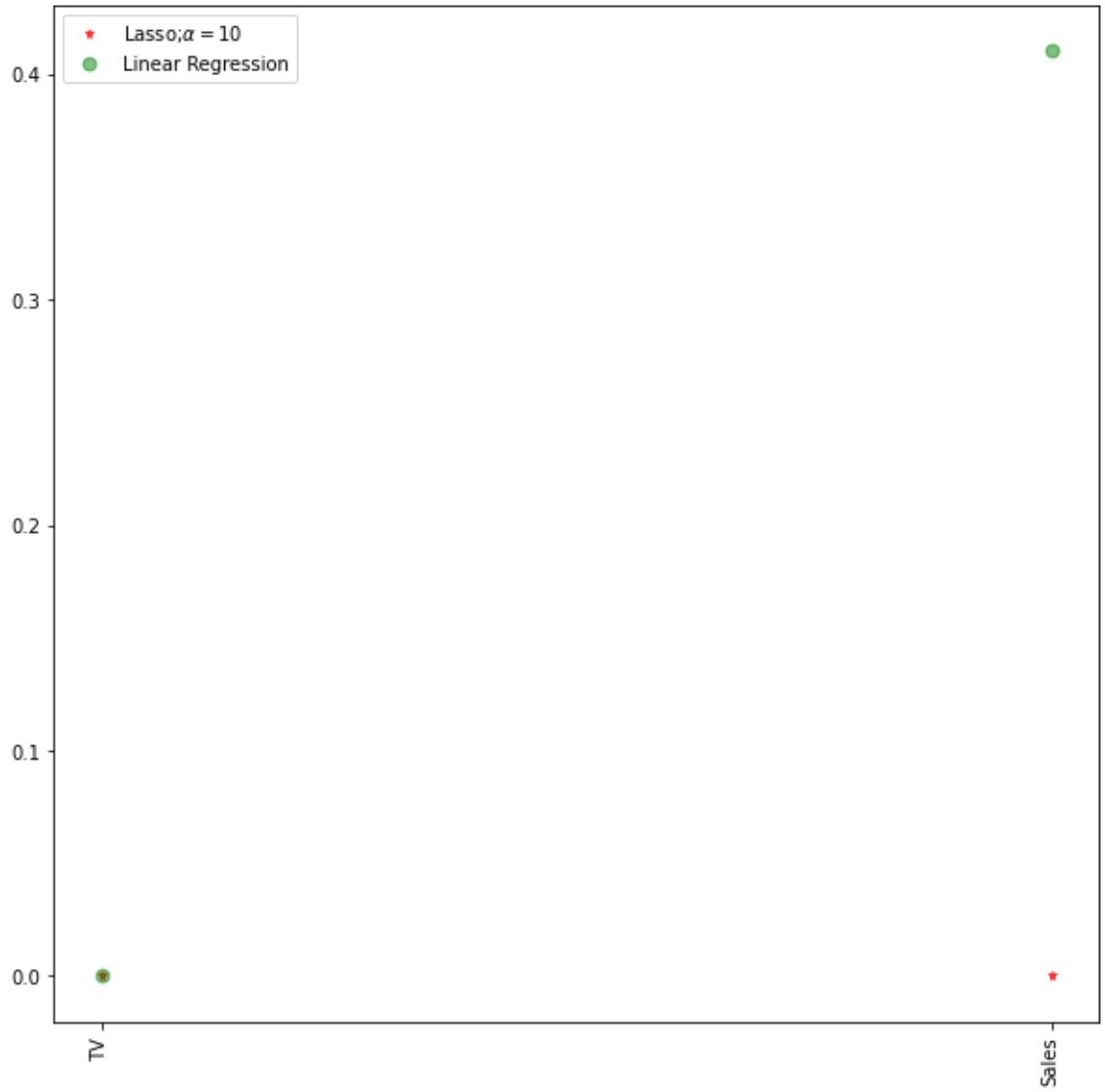
```
#Lasso regression model
lassoReg=Lasso( alpha = 10)
lassoReg.fit(X_train,y_train)
#train and test score for lasso regression
train_score_lasso=lassoReg.score(X_train,y_train)
test_score_lasso=lassoReg.score(X_test,y_test)
print("\nLasso model:\n")
print("The train score for lasso model is {}".format(train_score_lasso))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

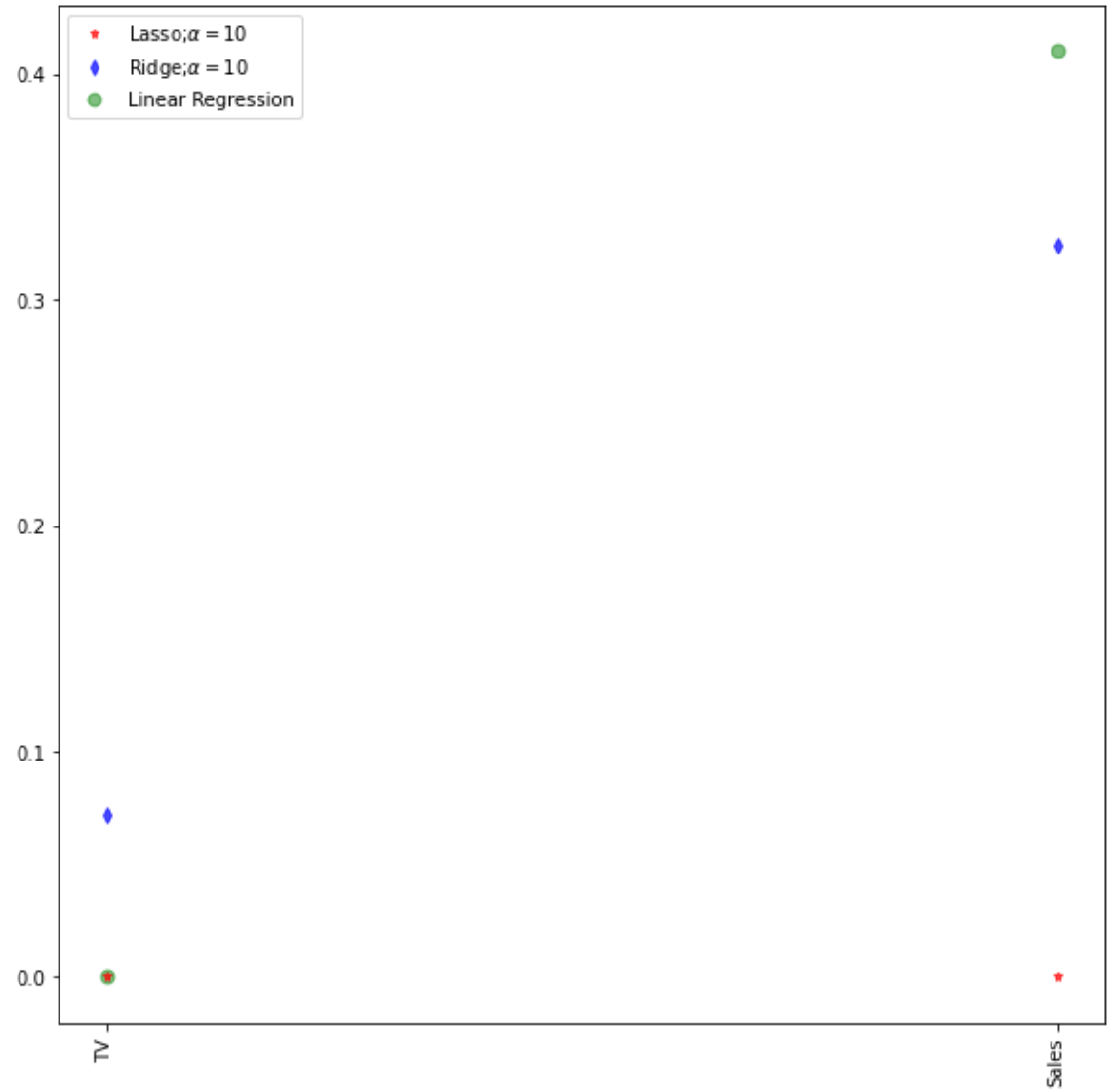
In [27]:

```
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 [28]:

```
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 [31]:

```
#elasticnet
from sklearn.linear_model import ElasticNet
regr=ElasticNet()
regr.fit(X,y)
print(regr.coef_)
print(regr.intercept_)
y_pred_elastic=regr.predict(X_train)
mean_squared_error=np.mean((y_pred_elastic-y_train)**2)
print("Mean Squared Error on test set",mean_squared_error)
```

```
[0.00417976 0.        ]
2.0263839193110043
Mean Squared Error on test set 0.5538818050142152
```

In [ ]: