In [20]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn import preprocessing,svm
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge,RidgeCV,Lasso
from sklearn.preprocessing import StandardScaler
```

In [21]:
```
df=pd.read_csv(r"C:\Users\Lenovo\OneDrive\Desktop\Data Sets\Advertising.csv")
df
```

Out[21]:

	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 [22]:
```
df.head()
```

Out[22]:

	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 [23]:
```
df.tail()
```

Out[23]:

	TV	Radio	Newspaper	Sales
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 [24]:
```
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 [25]:
```
df.describe()
```

Out[25]:

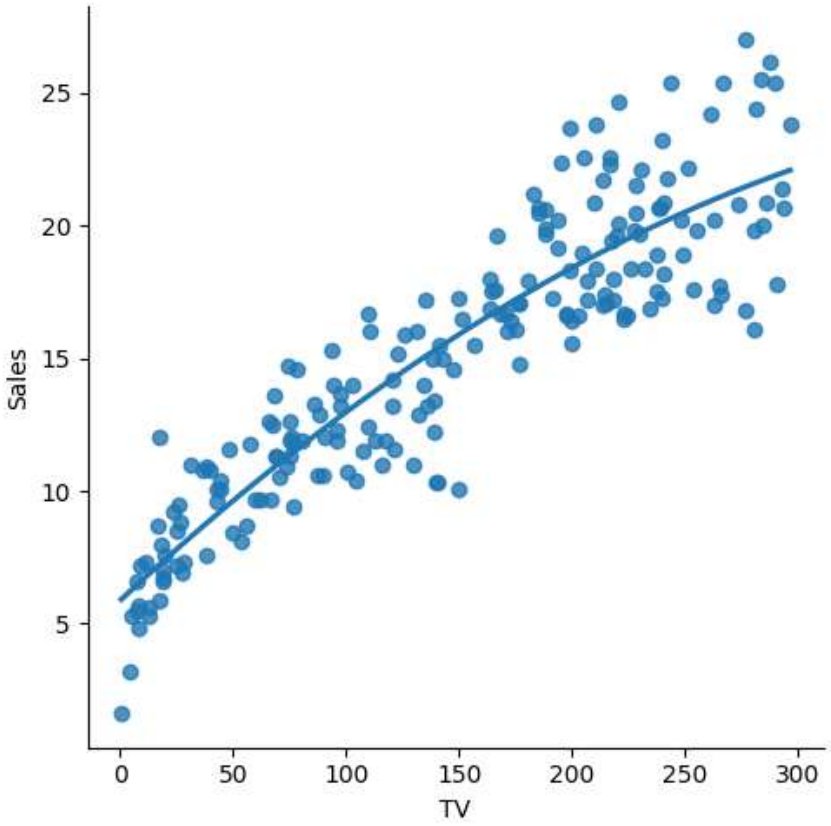
	TV	Radio	Newspaper	Sales
count	200.000000	200.000000	200.000000	200.000000
mean	147.042500	23.264000	30.554000	15.130500
std	85.854236	14.846809	21.778621	5.283892
min	0.700000	0.000000	0.300000	1.600000
25%	74.375000	9.975000	12.750000	11.000000
50%	149.750000	22.900000	25.750000	16.000000
75%	218.825000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

In [26]:
```
df.shape
```

Out[26]:
```
(200, 4)
```

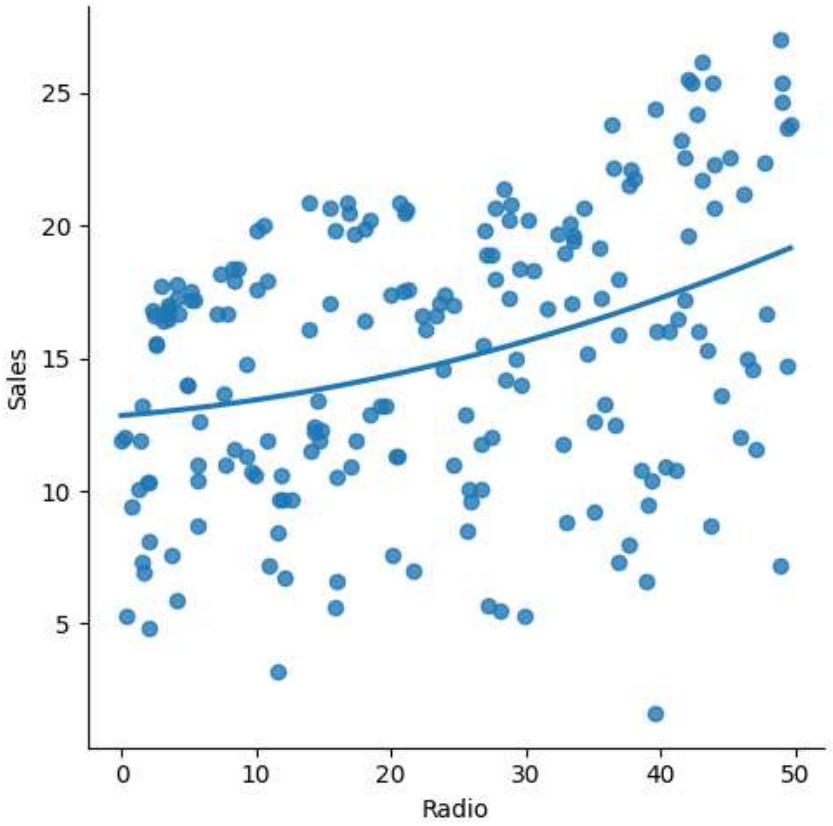
In [27]:
```
sns.lmplot(x="TV",y="Sales",data=df,order=2,ci=None)
```

Out[27]:
```
<seaborn.axisgrid.FacetGrid at 0x12903eeb1d0>
```



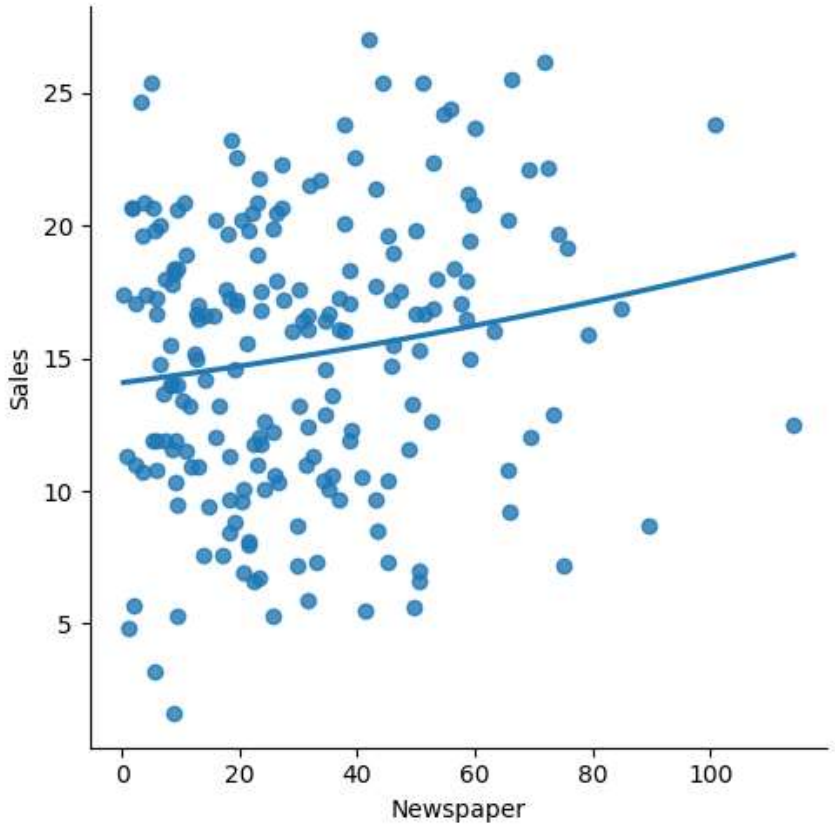
In [28]:
```
sns.lmplot(x="Radio",y="Sales",data=df,order=2,ci=None)
```

Out[28]:
```
<seaborn.axisgrid.FacetGrid at 0x1290474e0d0>
```



In [29]:
```
sns.lmplot(x="Newspaper",y="Sales",data=df,order=2,ci=None)
```

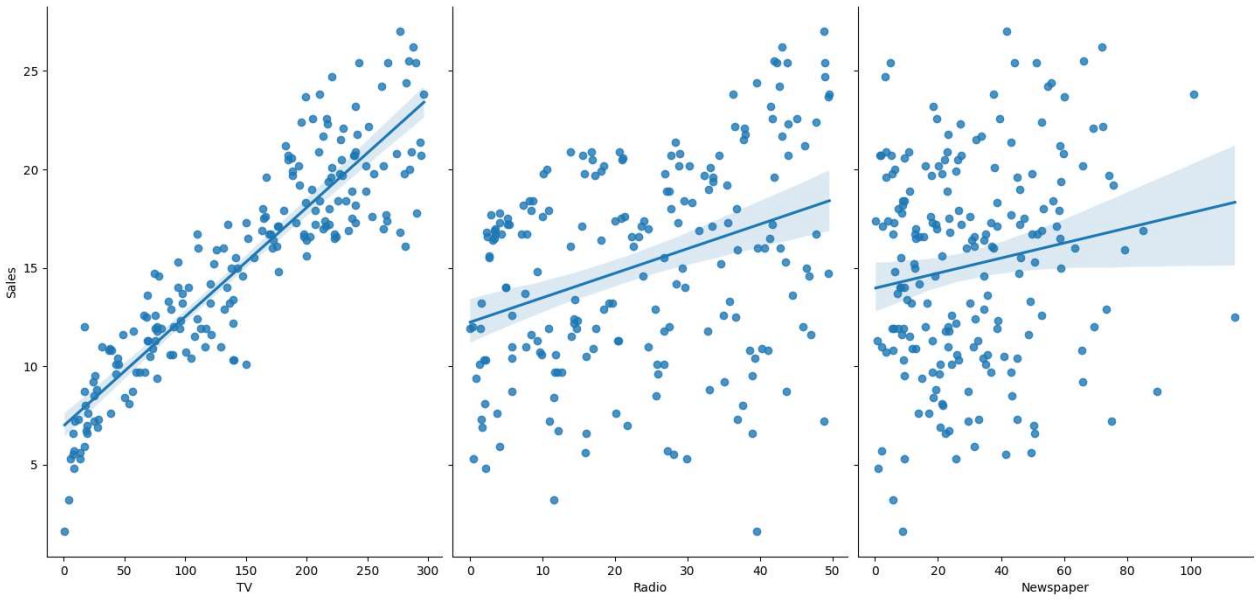
Out[29]:
```
<seaborn.axisgrid.FacetGrid at 0x12904756550>
```



In [30]:
```
df.fillna(method='ffill',inplace=True)
```

In [31]:
```
sns.pairplot(df,x_vars=['TV','Radio','Newspaper'],y_vars='Sales',height=7,aspect=0.7,kind='reg'
```

Out[31]:
```
<seaborn.axisgrid.PairGrid at 0x129006ea390>
```



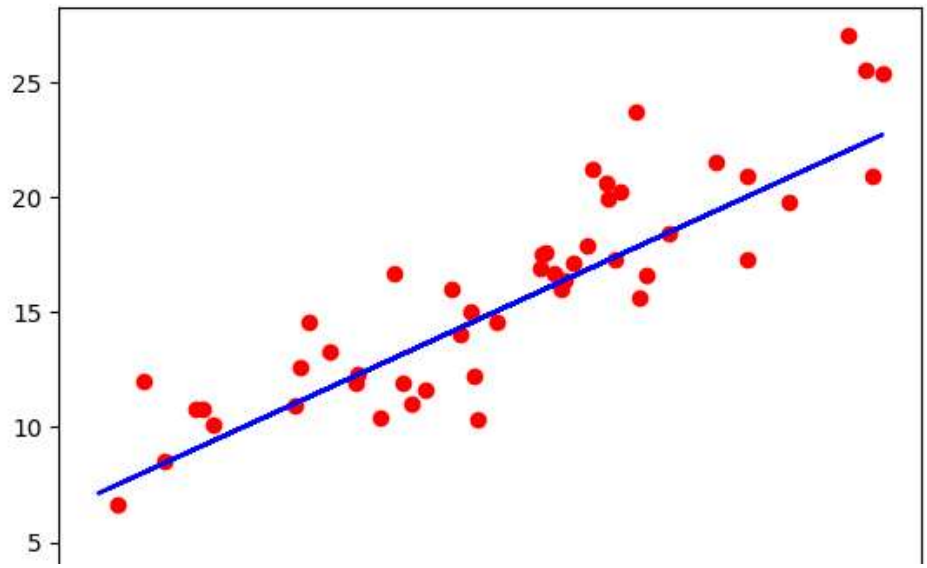
In [32]:
```
x=np.array(df['TV']).reshape(-1,1)
y=np.array(df['Sales']).reshape(-1,1)
df.dropna(inplace=True)
```

In [33]:
```
regr=LinearRegression()
```

In [34]:
```
X_train,X_test,Y_train,Y_test=train_test_split(x,y,test_size=0.25)
regr.fit(X_train,Y_train)
regr.fit(X_train,Y_train)
```

Out[34]:
```
LinearRegression
LinearRegression()
```

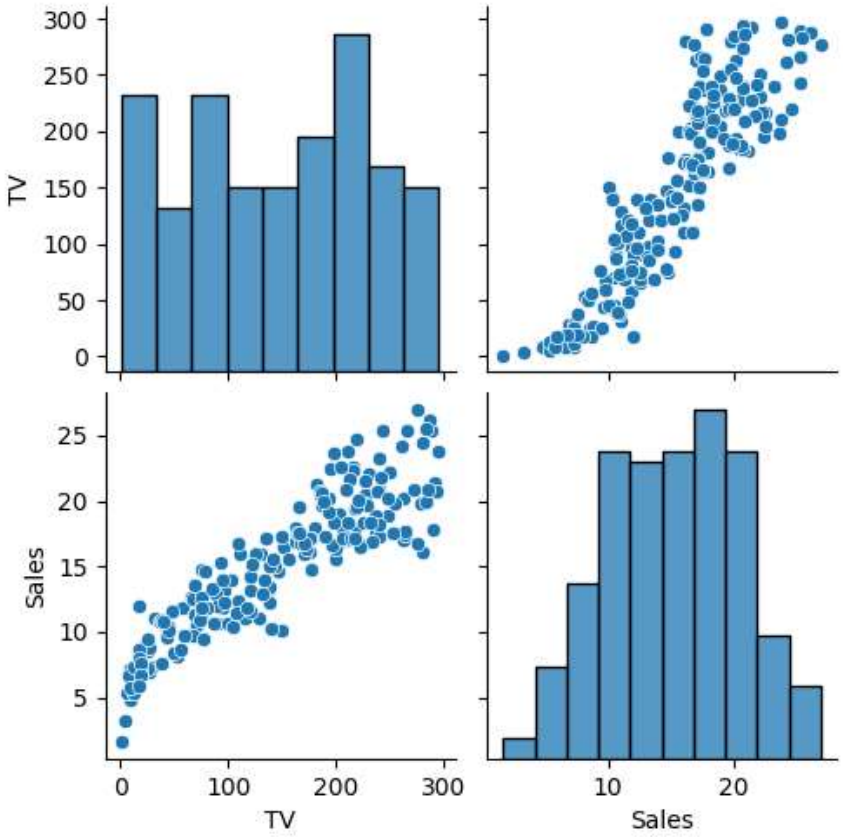
In [35]:
```
y_pred=regr.predict(X_test)
plt.scatter(X_test,Y_test,color='r')
plt.plot(X_test,y_pred,color='b')
plt.show()
```



In [37]:
```
plt.figure(figsize=(10,10))
sns.heatmap(df.corr(),annot=True)
```

Out[37]: <Axes: >

In [38]:
```
df.drop(columns=["Radio","Newspaper"],inplace=True)

#pairplot
sns.pairplot(df)

df.Sales=np.log(df.Sales)
```



In [39]:
```
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=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 (183, 2)
The dimension of X test is (17, 2)
```

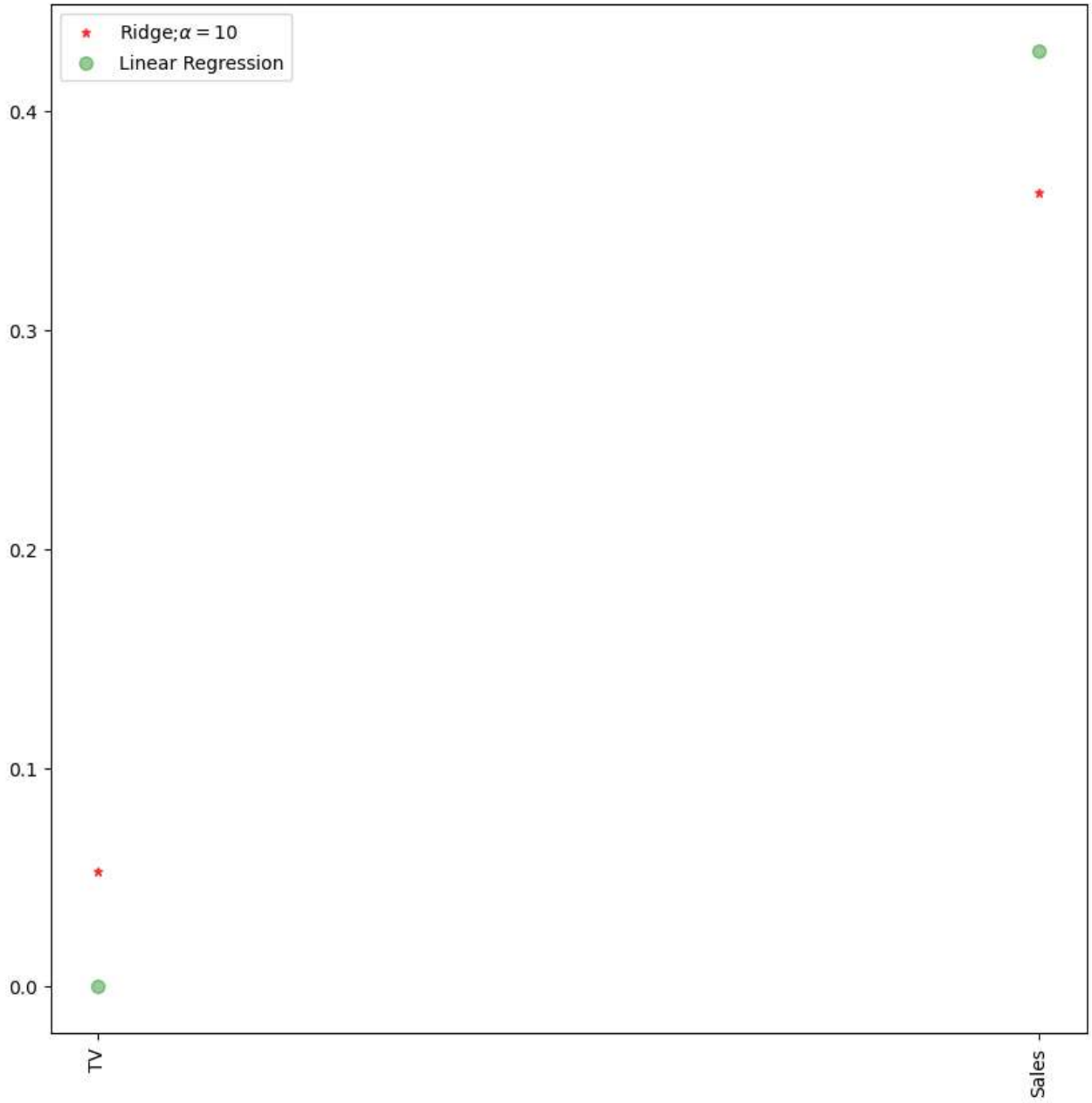
In [40]:
```
#model
lr=LinearRegression()

#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 [41]:
```
#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("\nLinear Regression Model:\n")
print("The train score for lr model is {}".format(train_score_ridge))
print("The test score for lr model is {}".format(test_score_ridge))
```

```
Linear Regression Model:

The train score for lr model is 0.9938003534333717
The test score for lr model is 0.9979567956770432
```

In [42]:
```
plt.figure(figsize=(10,10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5,color='red'
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color='green',lab

plt.xticks(rotation=90)
plt.legend()
plt.show()
```

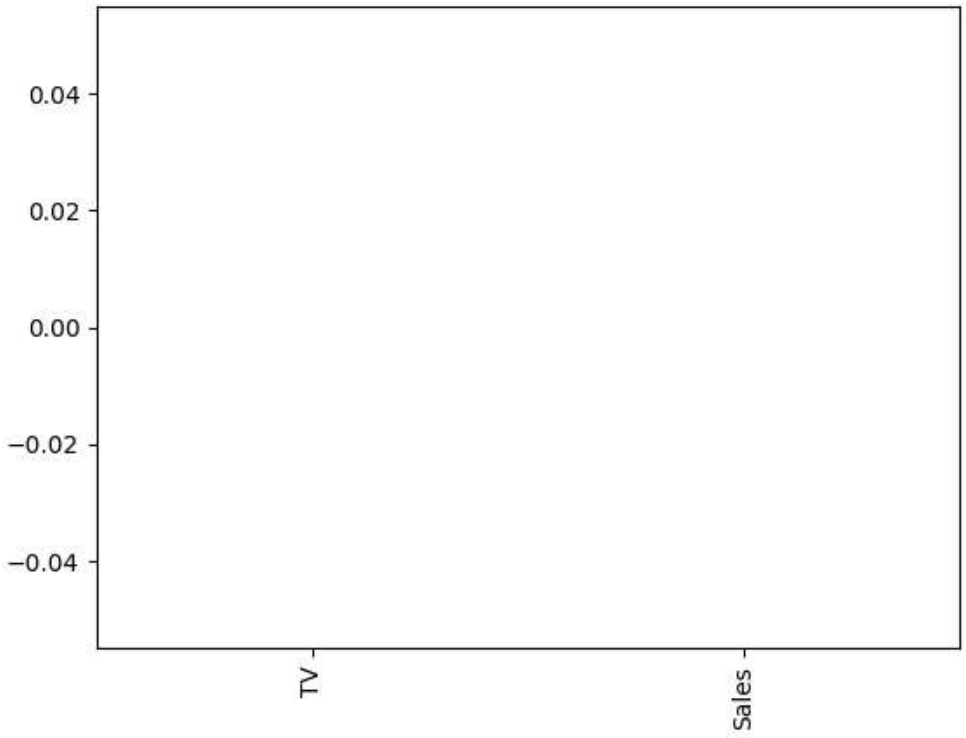


In [43]:
```
#Lasso regression model
print("\nLasso Model: \n")
lasso=Lasso(alpha=10)
lasso.fit(X_train,y_train)
train_score_ls=lasso.score(X_train,y_train)
test_score_ls=lasso.score(X_test,y_test)

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 -0.1764851111612895
```

In [44]:
```
pd.Series(lasso.coef_,features).sort_values(ascending=True).plot(kind="bar")
```

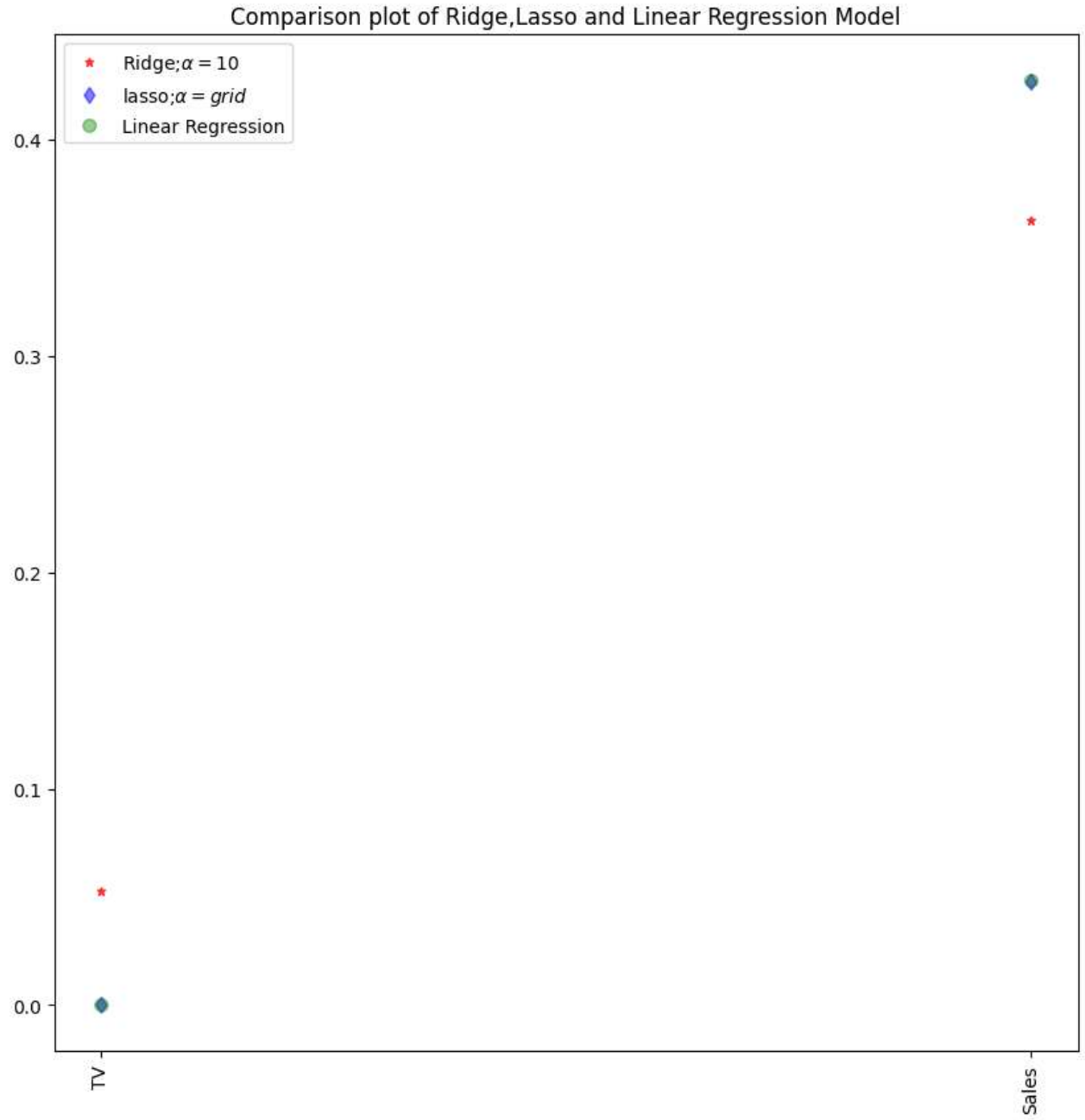
Out[44]: <Axes: >



In [45]:
```
#using the linear CV model
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))
```

```
0.9999999406813933
0.9999999492433705
```

In [46]:
```
plt.figure(figsize=(10,10))

plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5,color='red'
plt.plot(lasso_cv.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color='blue',label=r
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color='green',lab

plt.xticks(rotation=90)
plt.legend()
plt.title("Comparison plot of Ridge,Lasso and Linear Regression Model")
plt.show()
```



In [47]:
```
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 0.999999999998885
The train score for ridge model is 0.9999999999996397
```