In [1]:

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

In [2]:

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

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]:

```
data.head()
```

Out[3]:

	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 [4]:

```
data.tail()
```

Out[4]:

	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 [5]:

```
data.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 [6]:

```
data.describe()
```

Out[6]:

	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 [7]:

```
data.shape
```

Out[7]:

```
(200, 4)
```

In [8]:

```
data.columns
```

Out[8]:

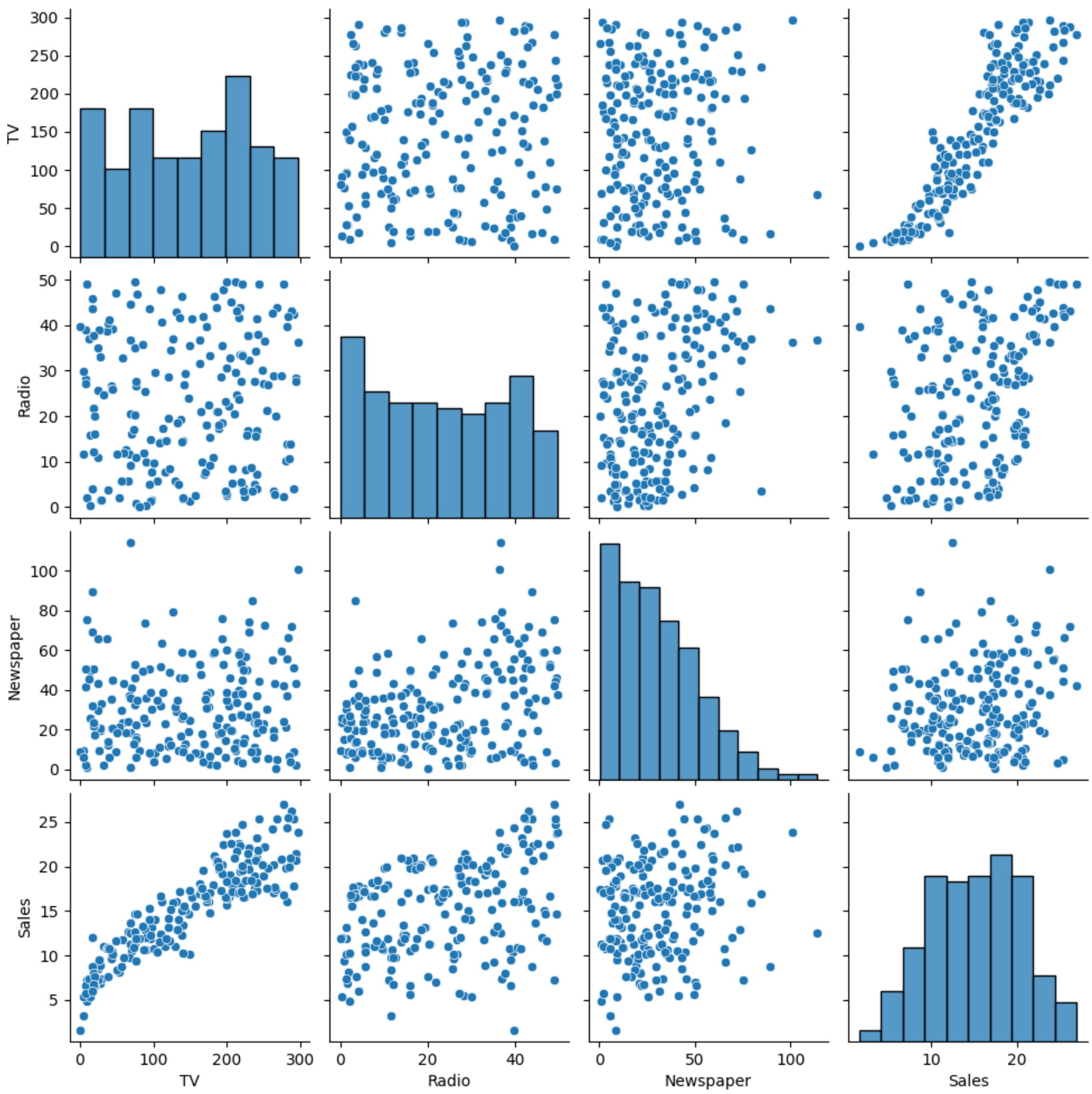
```
Index(['TV', 'Radio', 'Newspaper', 'Sales'], dtype='object')
```

In [9]:

```
sns.pairplot(data)
```

Out[9]:

```
<seaborn.axisgrid.PairGrid at 0x1744c213550>
```

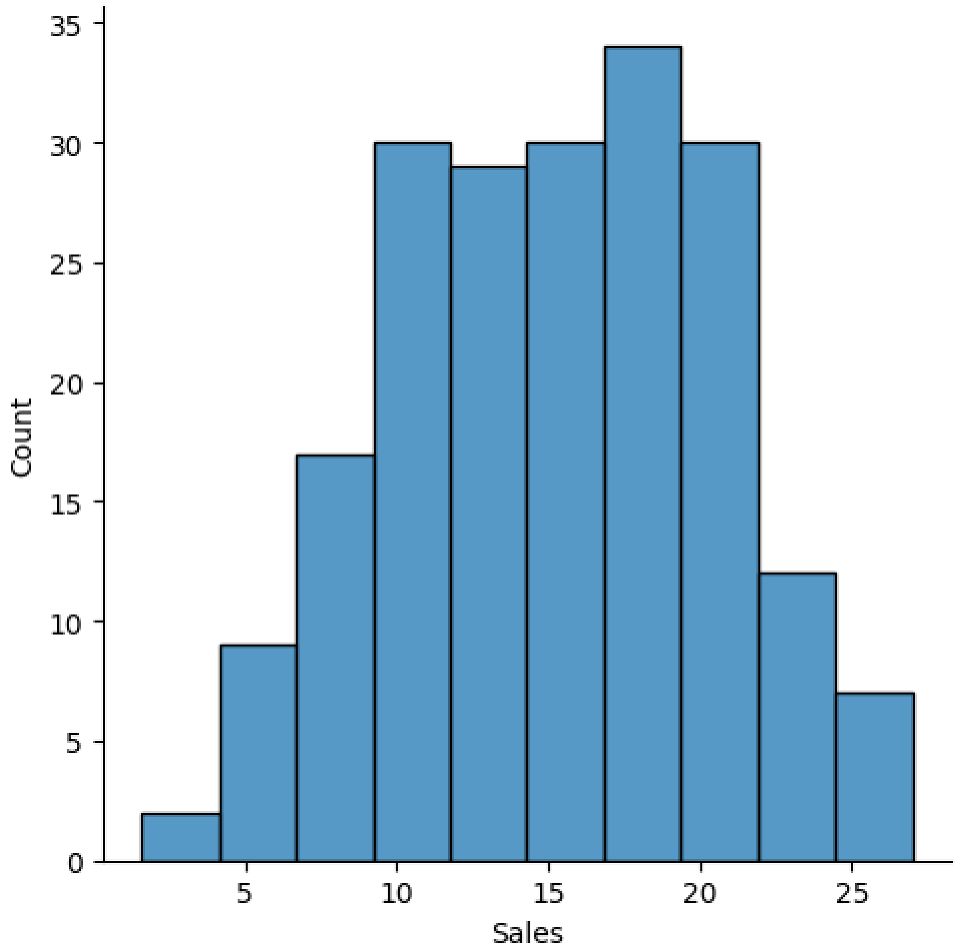


In [10]:

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

Out[10]:

```
<seaborn.axisgrid.FacetGrid at 0x1744f449710>
```

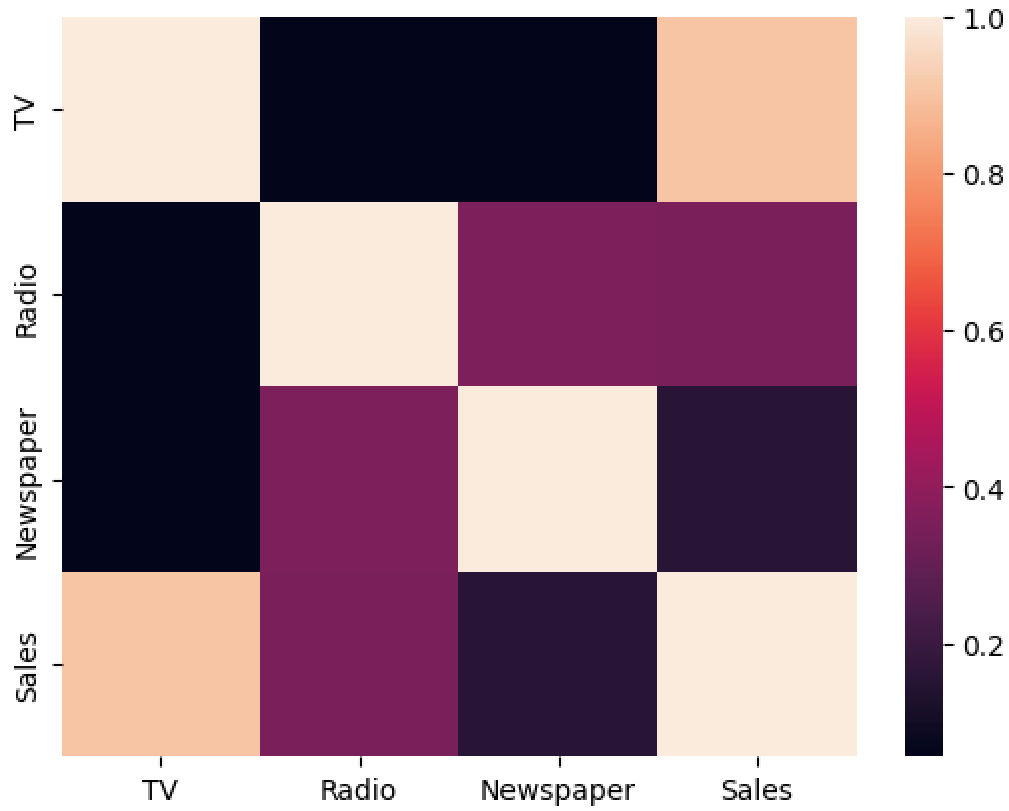


In [11]:

```
addf=data[['TV', 'Radio', 'Newspaper', 'Sales']]
sns.heatmap(addf.corr())
```

Out[11]:

```
<Axes: >
```



In [12]:

```
X=addf[['TV', 'Radio', 'Newspaper']]
y=data['Sales']
```

In [13]:

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=101)
from sklearn.linear_model import LinearRegression
lm=LinearRegression()
lm.fit(X_train,y_train)
print(lm.intercept_)
```

```
4.681232151484295
```

In [14]:

```
coeff_data=pd.DataFrame(lm.coef_,X.columns,columns=['coefficient'])
coeff_data
```

Out[14]:

	coefficient
TV	0.054930
Radio	0.109558
Newspaper	-0.006194

In [15]:

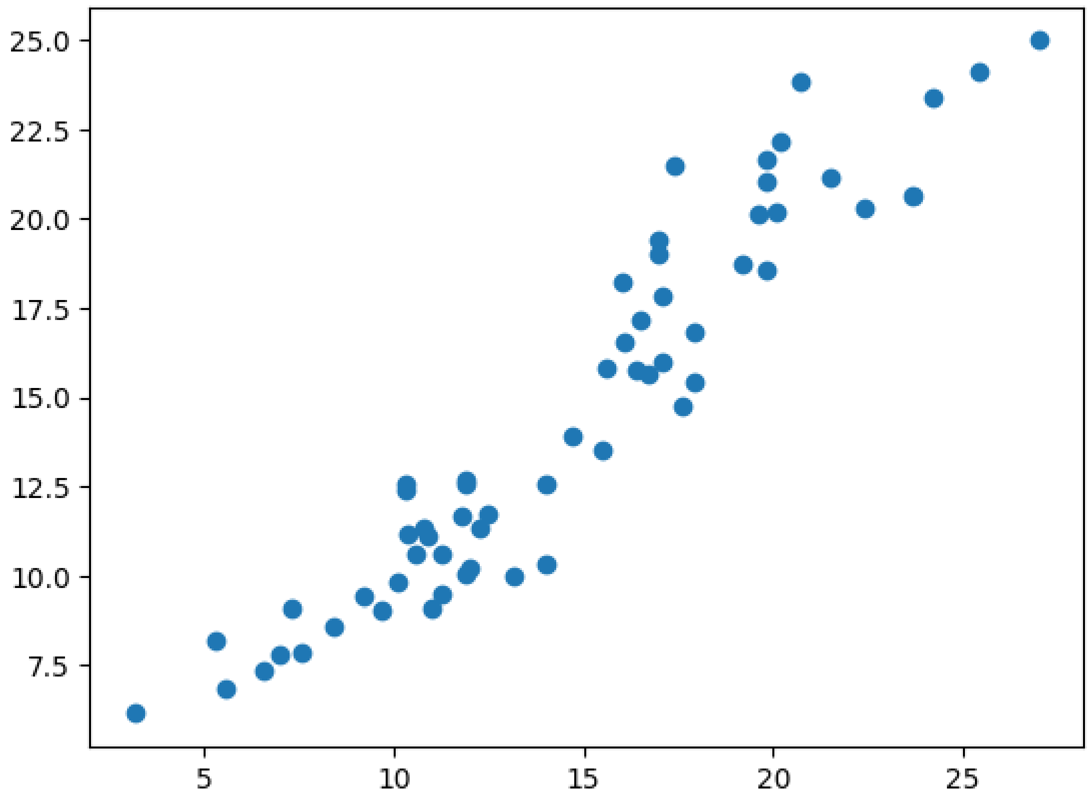
```
predictions=lm.predict(X_test)
```

In [16]:

```
plt.scatter(y_test,predictions)
```

Out[16]:

```
<matplotlib.collections.PathCollection at 0x1744fa4db50>
```

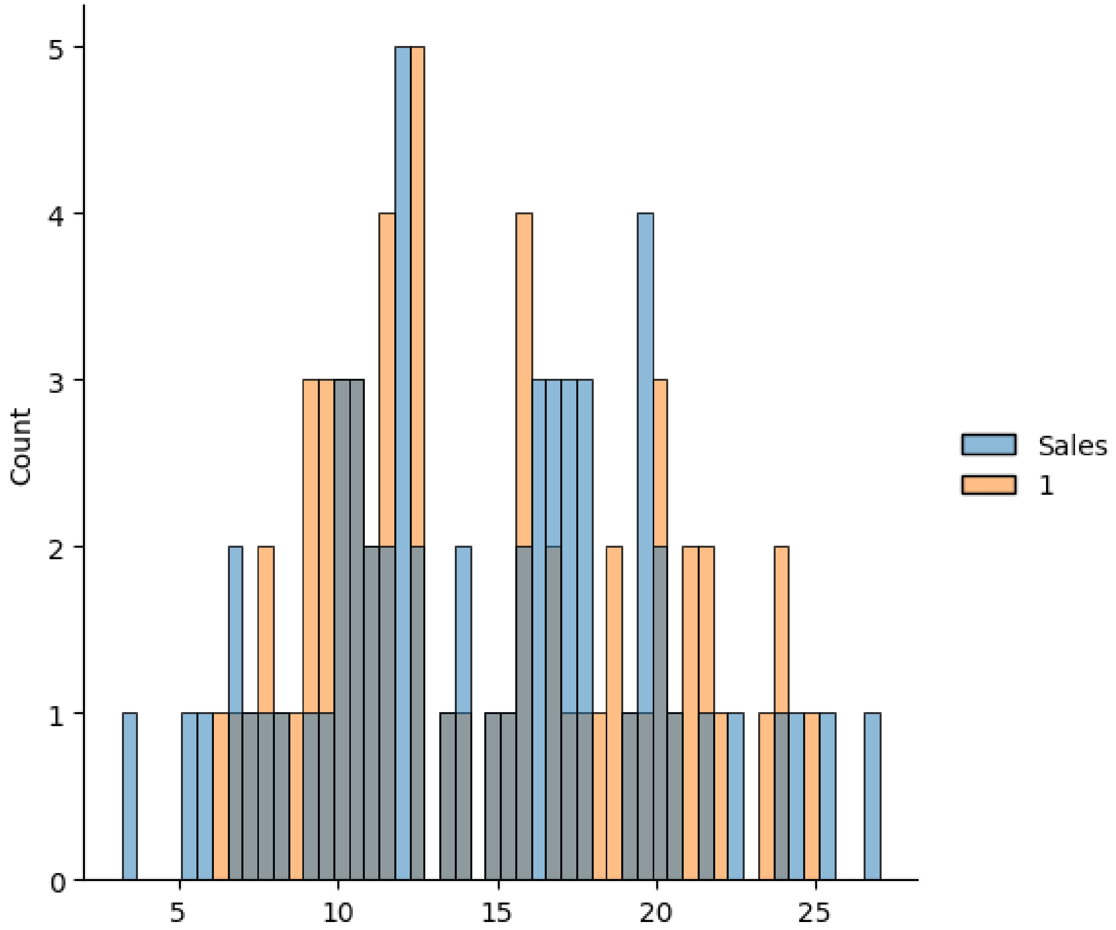


In [17]:

```
sns.displot((y_test,predictions),bins=50)
```

Out[17]:

<seaborn.axisgrid.FacetGrid at 0x1744fa77f10>



In [18]:

```
from sklearn import metrics
print('MAE:',metrics.mean_absolute_error(y_test,predictions))
print('MSE:',metrics.mean_squared_error(y_test,predictions))
print('MAE:',np.sqrt(metrics.mean_squared_error(y_test,predictions)))
```

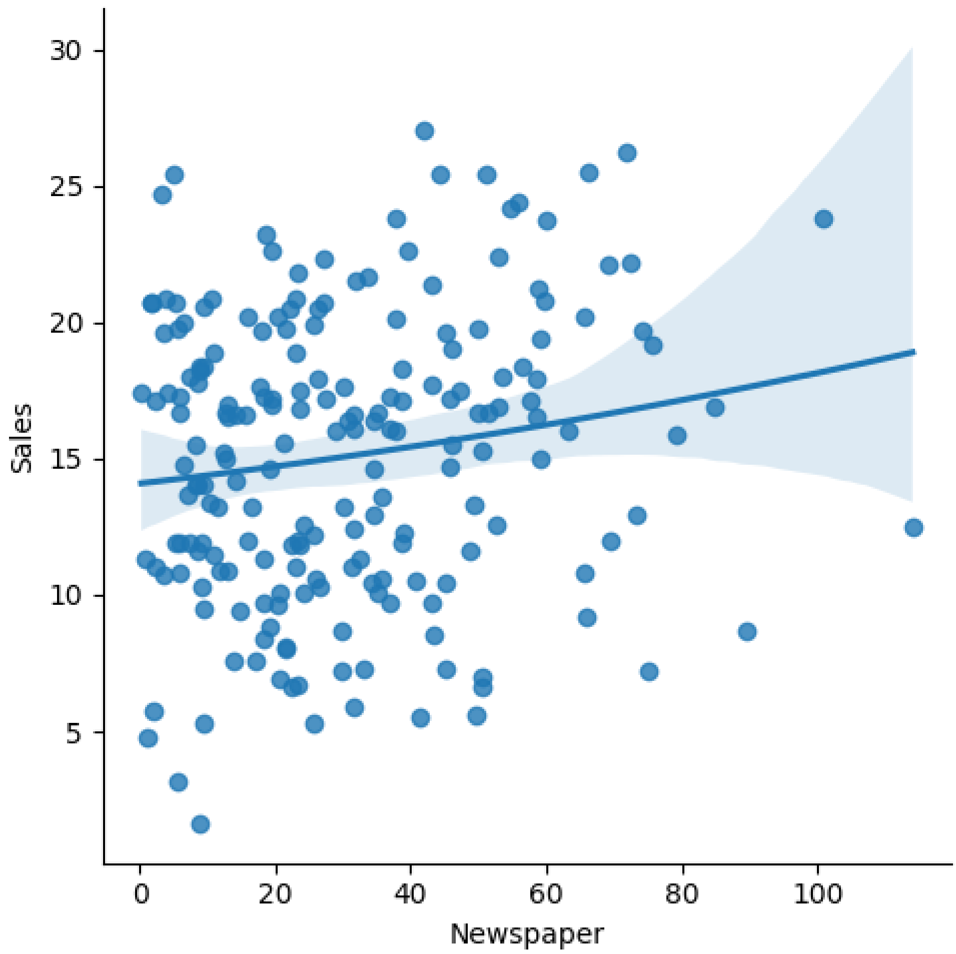
```
MAE: 1.3731200698367851
MSE: 2.8685706338964967
MAE: 1.6936855180040056
```

In [19]:

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

Out[19]:

<seaborn.axisgrid.FacetGrid at 0x1744f576d90>

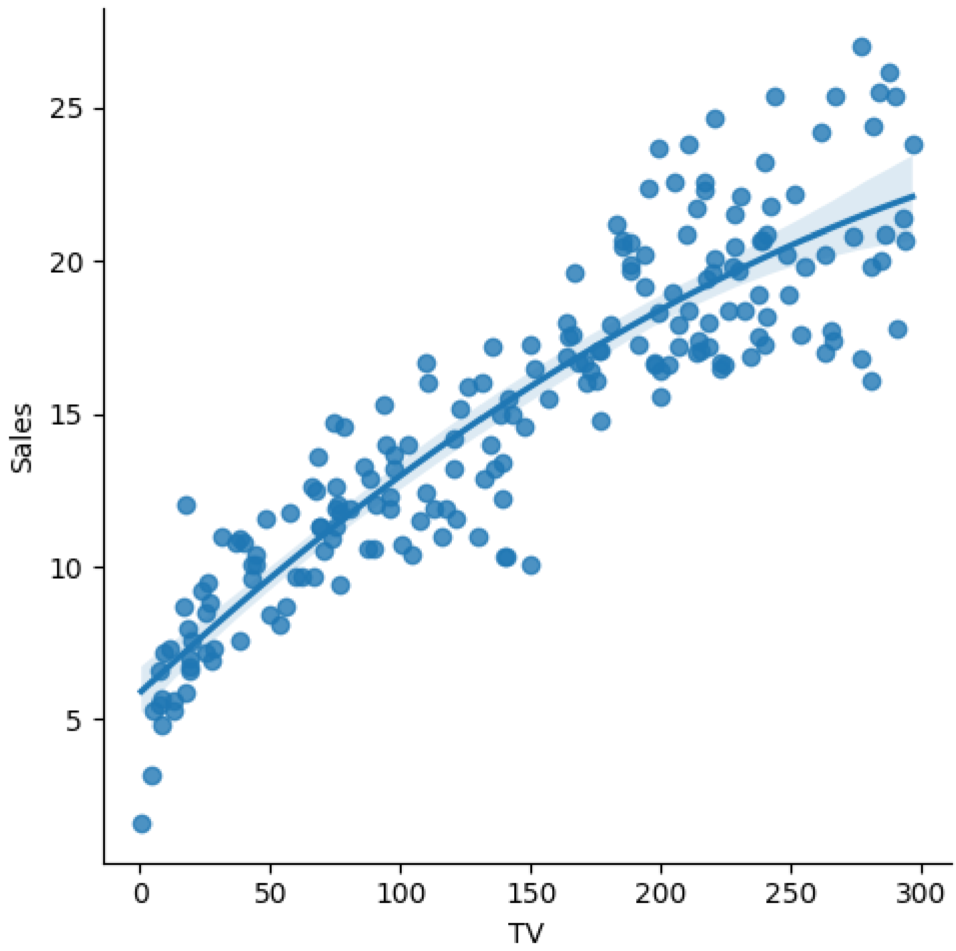


In [20]:

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

Out[20]:

<seaborn.axisgrid.FacetGrid at 0x17452cb8cd0>

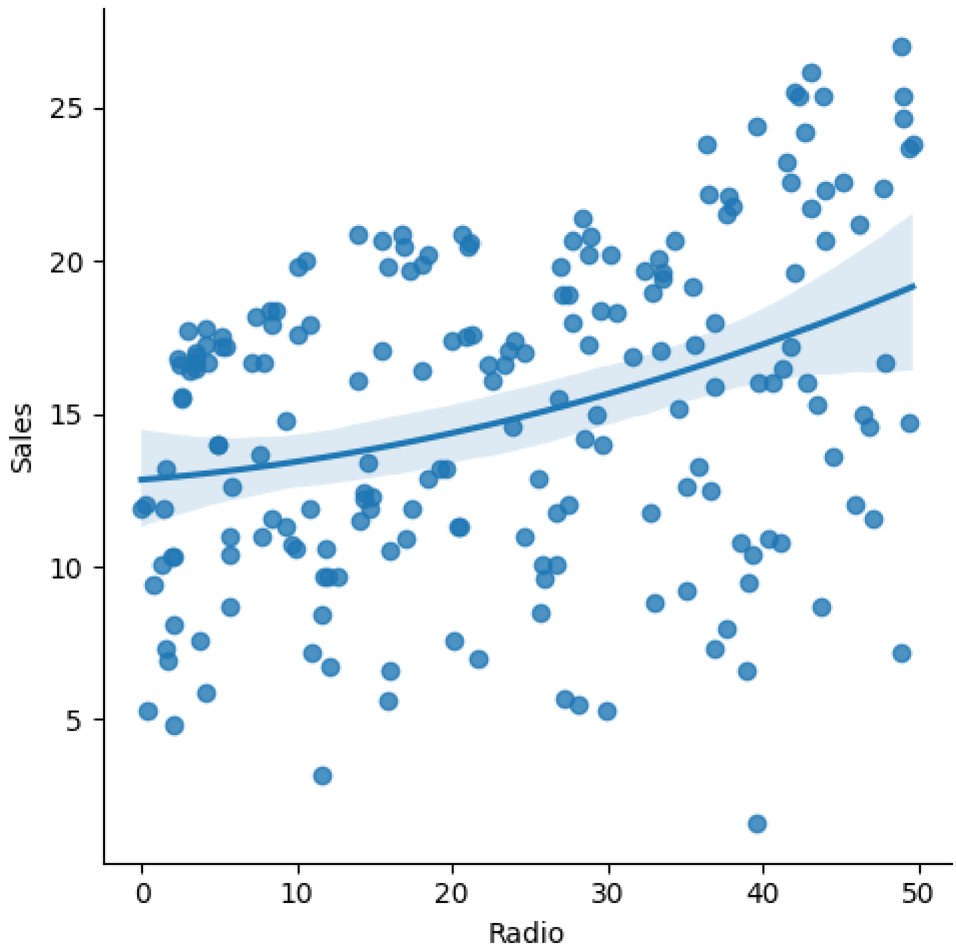


In [21]:

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

Out[21]:

<seaborn.axisgrid.FacetGrid at 0x17452d70c50>



In [22]:

```
data.fillna(method='ffill',inplace=True)
```

In [23]:

```
regr=LinearRegression()
```

In [24]:

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

In [25]:

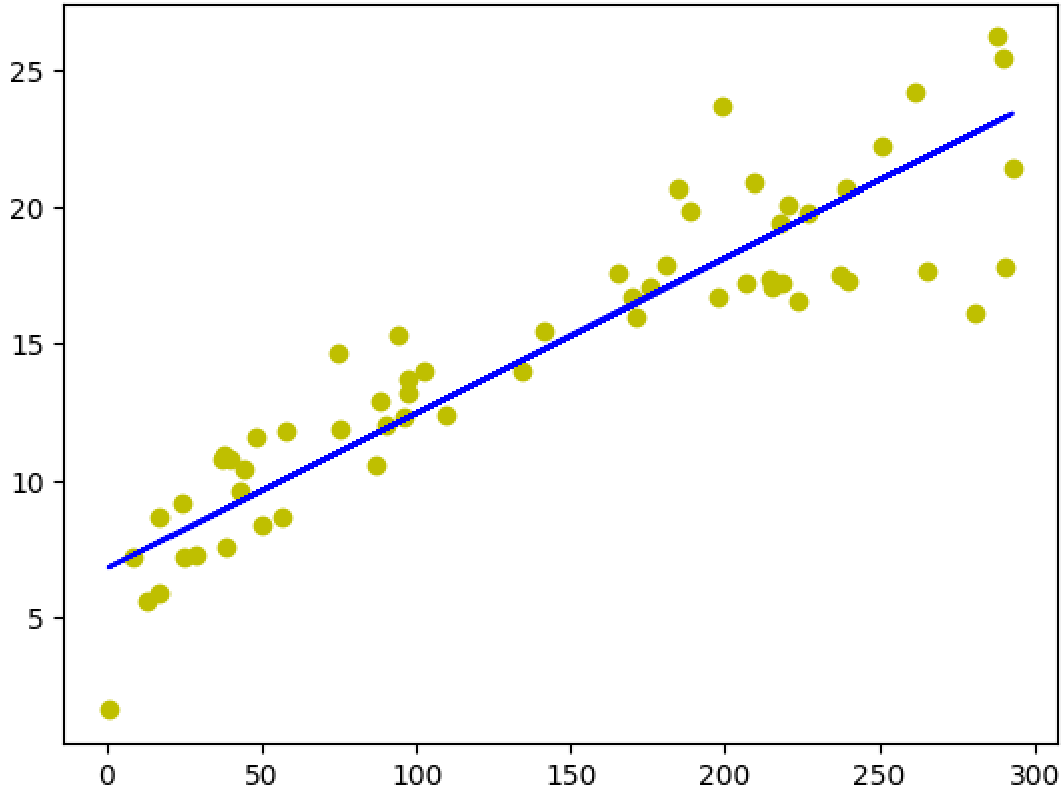
```
X_train,X_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
regr.fit(X_train,y_train)
regr.fit(X_train,y_train)
```

Out[25]:

```
▾ LinearRegression
LinearRegression()
```

In [26]:

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

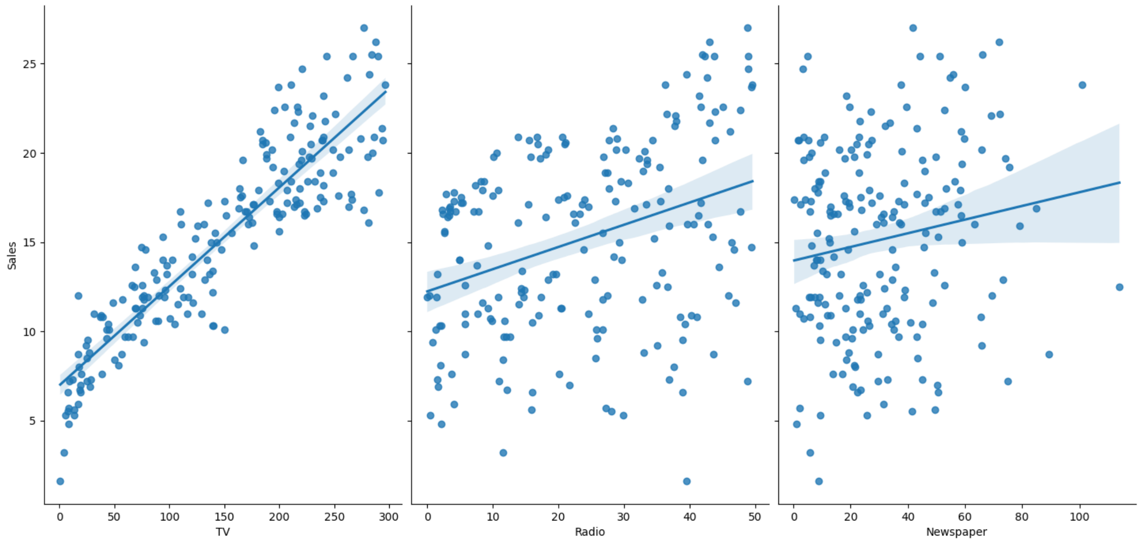


In [27]:

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

Out[27]:

```
<seaborn.axisgrid.PairGrid at 0x1744f3d8690>
```



In [28]:

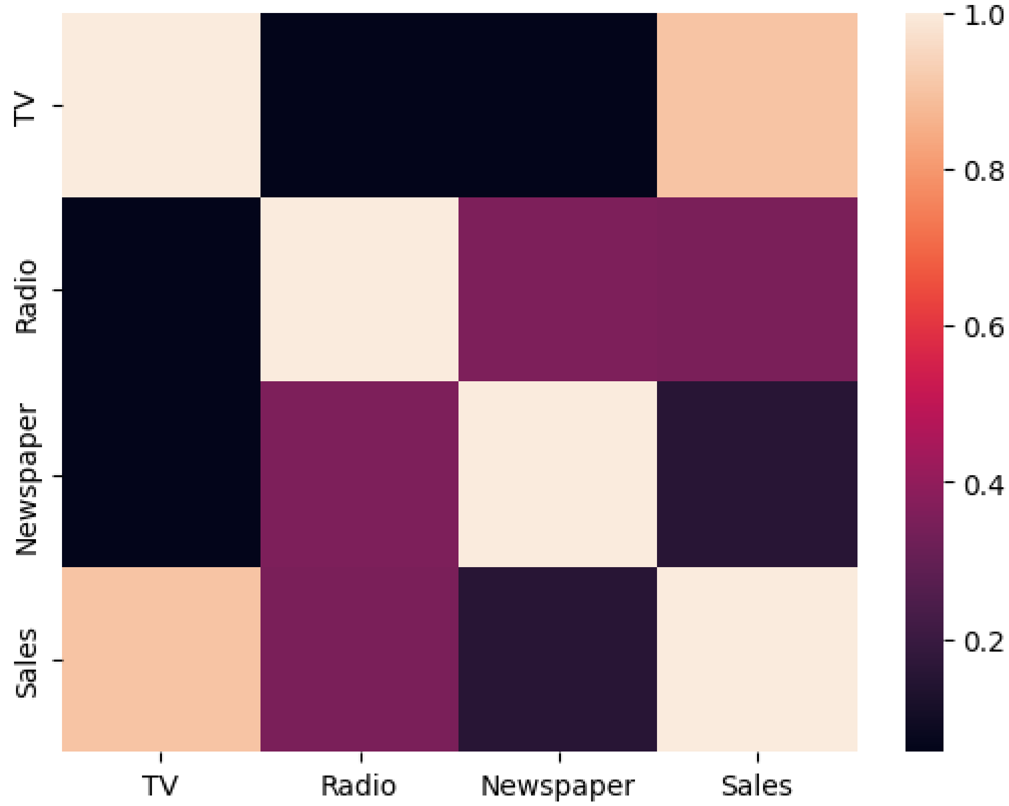
```
hk=data[['TV','Radio','Newspaper','Sales']]
```

In [29]:

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

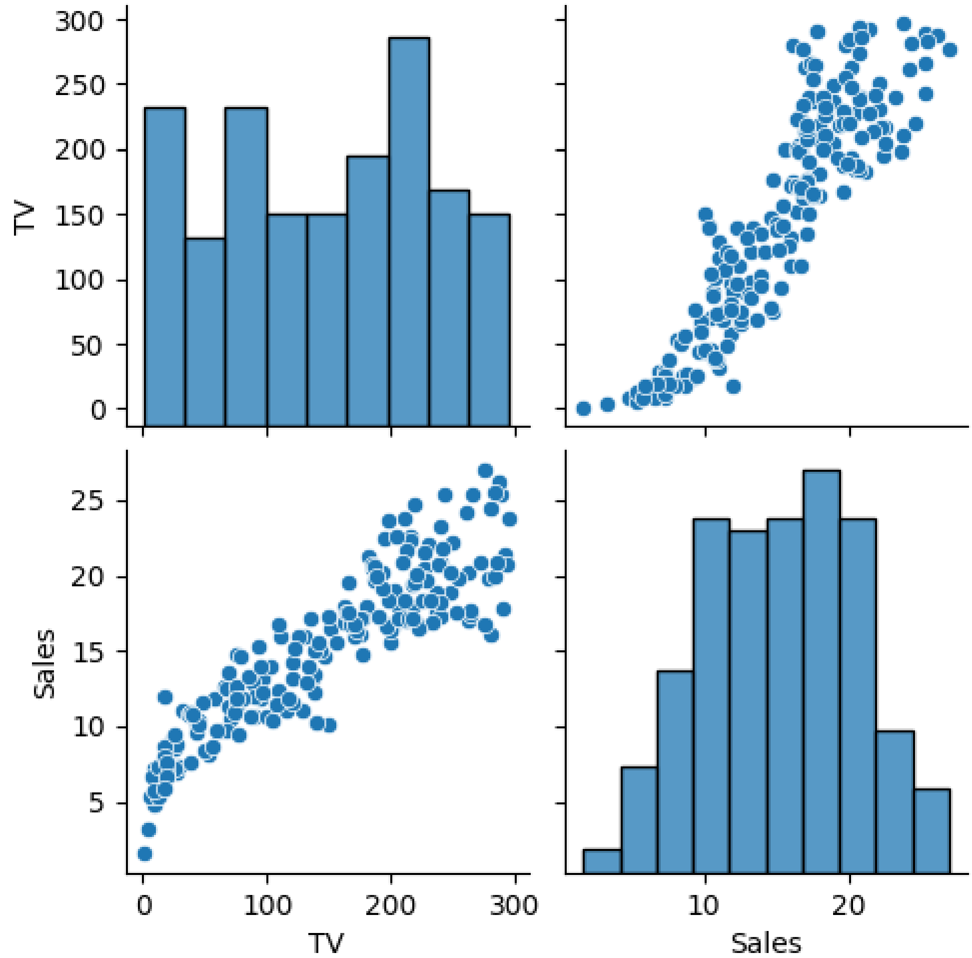
Out[29]:

```
<Axes: >
```



In [30]:

```
data.drop(columns=['Radio','Newspaper'],inplace=True)
sns.pairplot(data)
data.Sales=np.log(data.Sales)
```



In [32]:

```
features=data.columns[0:2]
target=data.columns[-1]
X=data[features].values
y=data[target].values
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))
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 [33]:

```
from sklearn.linear_model import Lasso,Ridge
```

In [34]:

```
lr=LinearRegression()
lr.fit(X_train,y_train)
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 train score lr model is {}".format(test_score_lr))
```

```
Linear Regression Model:

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

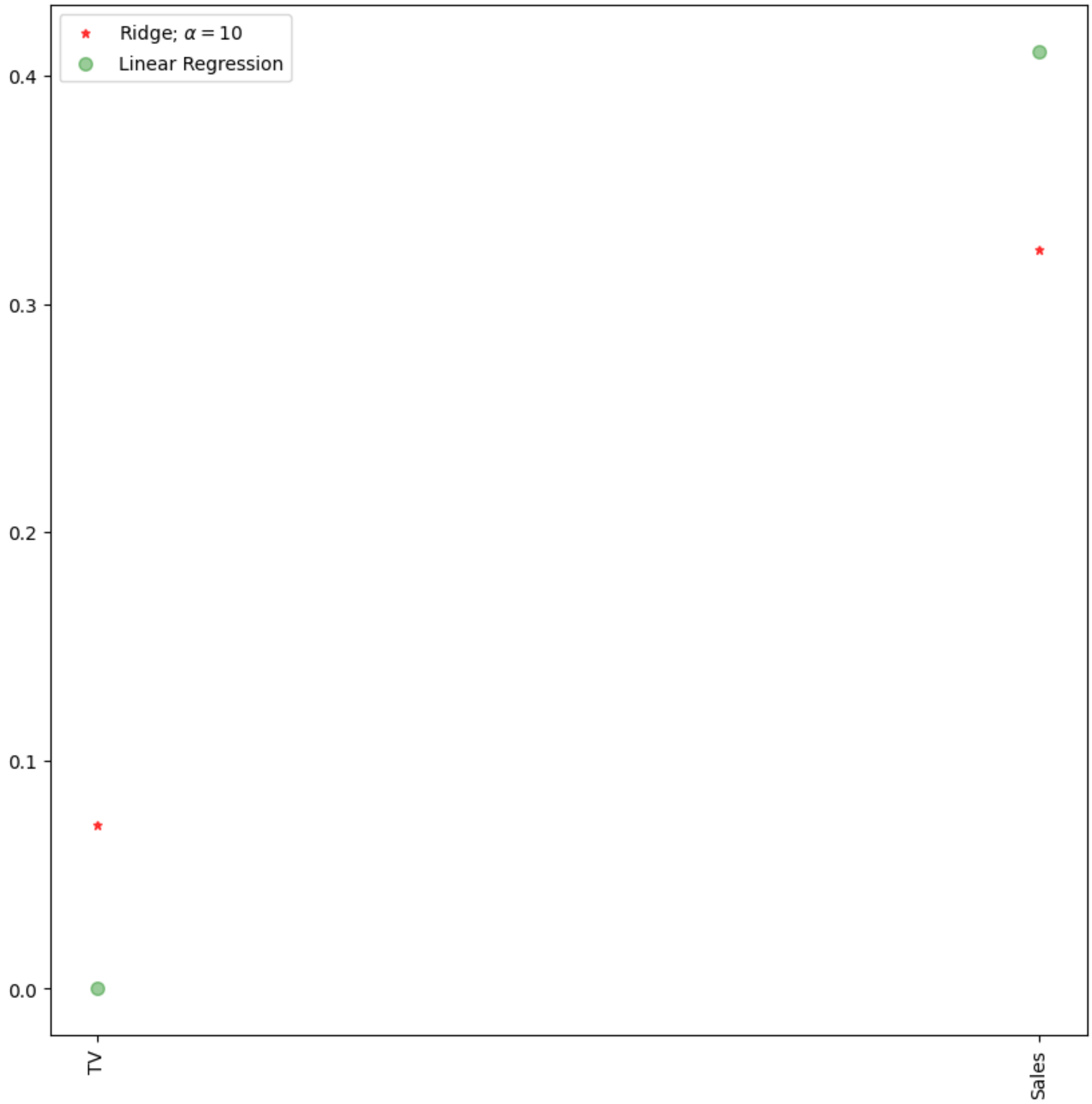
In [35]:

```
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_test,y_test)
print("\nRidge model\:\n")
print("The train score for ridge model is {}".format(train_score_ridge))
print("The train score for ridge model is {}".format(test_score_ridge))
```

```
Ridge model\:

The train score for ridge model is 0.9902871391941609
The train score for ridge model is 0.984426628514122
```

In [37]:

```
plt.figure(figsize=(10,10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5,colo
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color='gre
plt.xticks(rotation=90)
plt.legend()
plt.show()
```

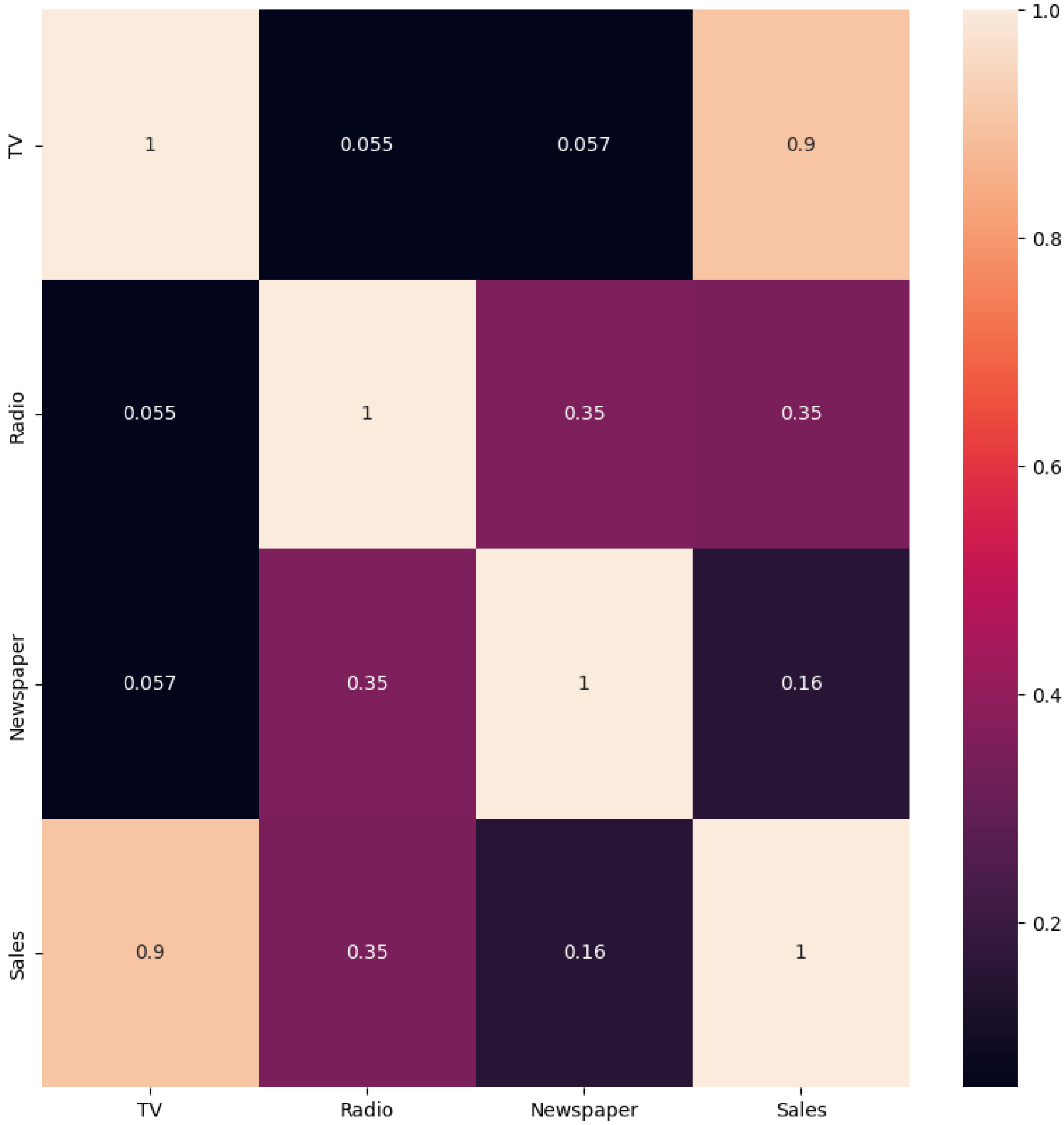


In [41]:

```
plt.figure(figsize = (10, 10))
sns.heatmap(data.corr(), annot = True)
```

Out[41]:

<Axes: >



In [13]:

```
#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))
```

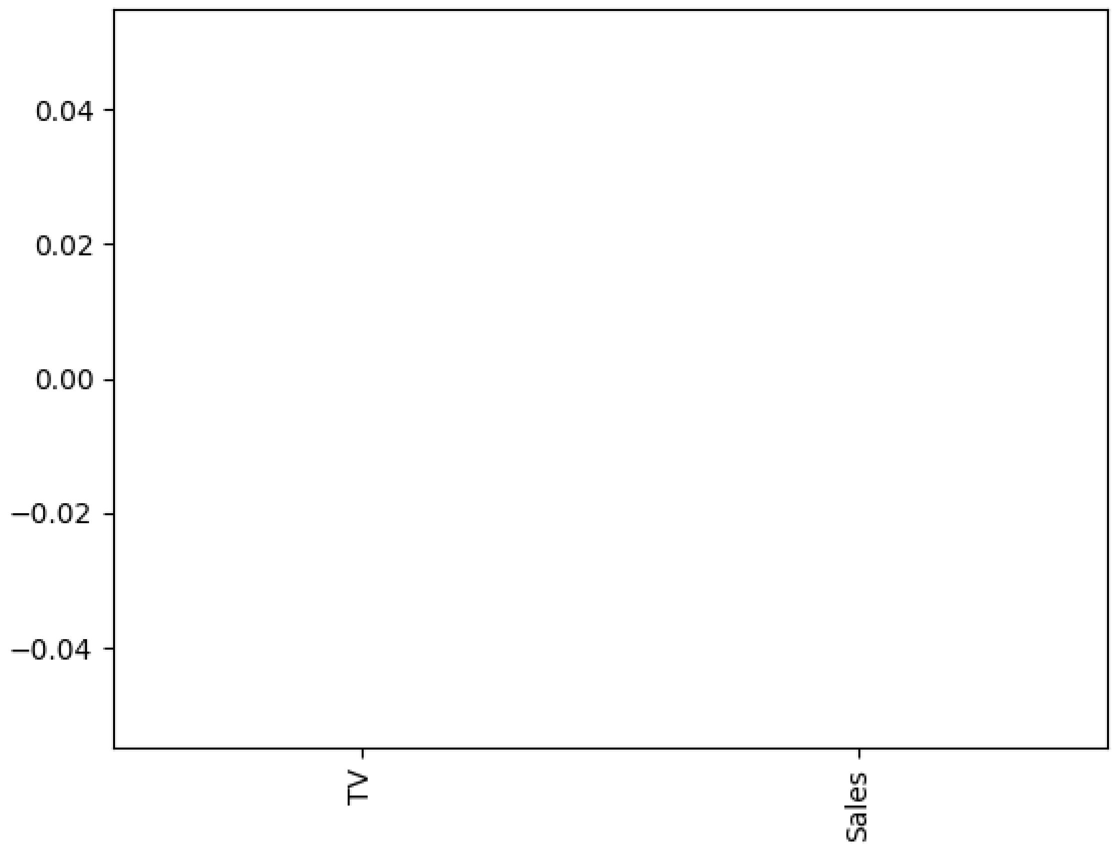
```
Lasso Model:

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

In [14]:

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

Out[14]:

```
<Axes: >
```



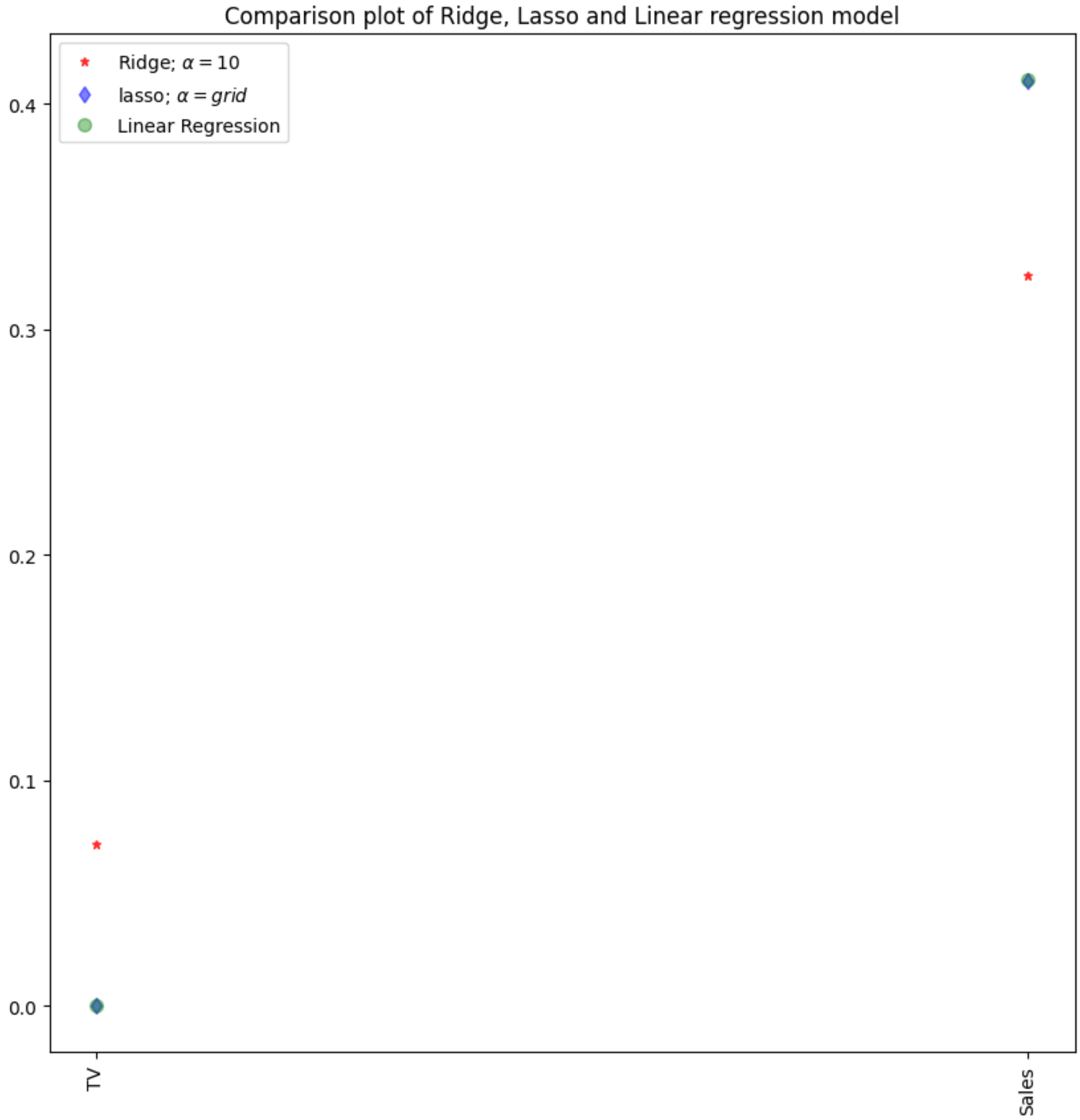
In [15]:

```
#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_trai
#score
print(lasso_cv.score(X_train, y_train))
print(lasso_cv.score(X_test, y_test))
```

```
0.9999999343798134
0.9999999152638072
```

In [16]:

```
#plot size
plt.figure(figsize = (10, 10))
#add plot for ridge regression
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5,colo
#add plot for lasso regression
plt.plot(lasso_cv.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color='blue',
#add plot for linear model
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color='gre
#rotate axis
plt.xticks(rotation = 90)
plt.legend()
plt.title("Comparison plot of Ridge, Lasso and Linear regression model")
plt.show()
```



In [17]:

```
#Using the linear CV model
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("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 0.999999999997627
The train score for ridge model is 0.9999999999962466
```

In [38]:

```
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 the tset set",mean_squared_error)
```

```
[0.00417976 0.        ]
2.026383919311004
mean Squared Error on the tset set 0.5538818050142158
```