LINEAR REGRESSION ADVERTISING ELASTICNET

In [4]:

```
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.linear_model import LinearRegression
from sklearn import metrics
from sklearn import preprocessing,svm
```

In [5]:

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

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
...	...	...	...	...
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 [6]:

```
df.tail()
```

Out[6]:

	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 [7]:

```
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 [8]:

```
df.describe()
```

Out[8]:

	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 [9]:

```
df.shape
```

Out[9]:

```
(200, 4)
```

In [10]:

```
df.columns
```

Out[10]:

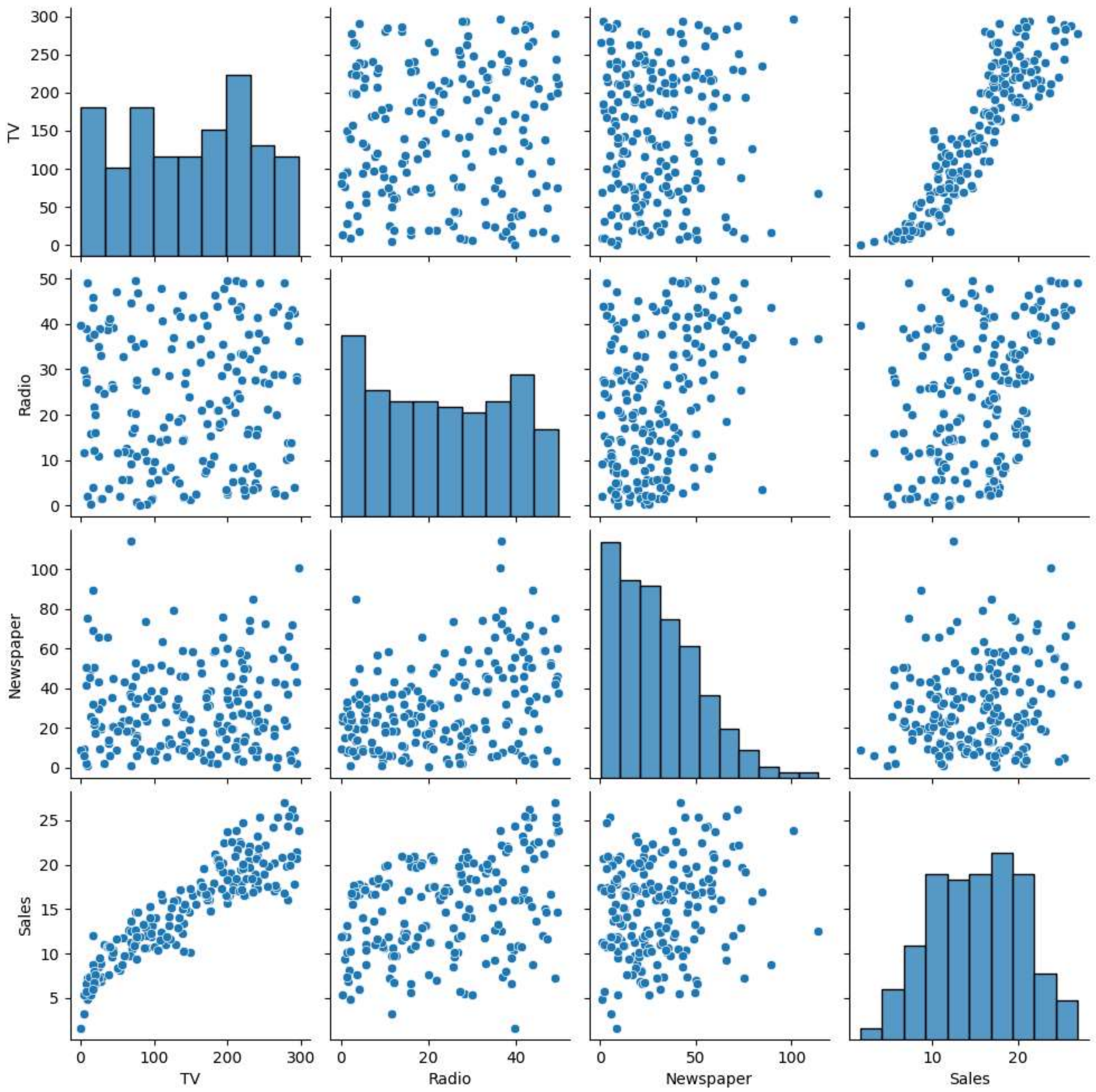
```
Index(['TV', 'Radio', 'Newspaper', 'Sales'], dtype='object')
```

In [11]:

```
sns.pairplot(df)
```

Out[11]:

<seaborn.axisgrid.PairGrid at 0x239ccfe9290>

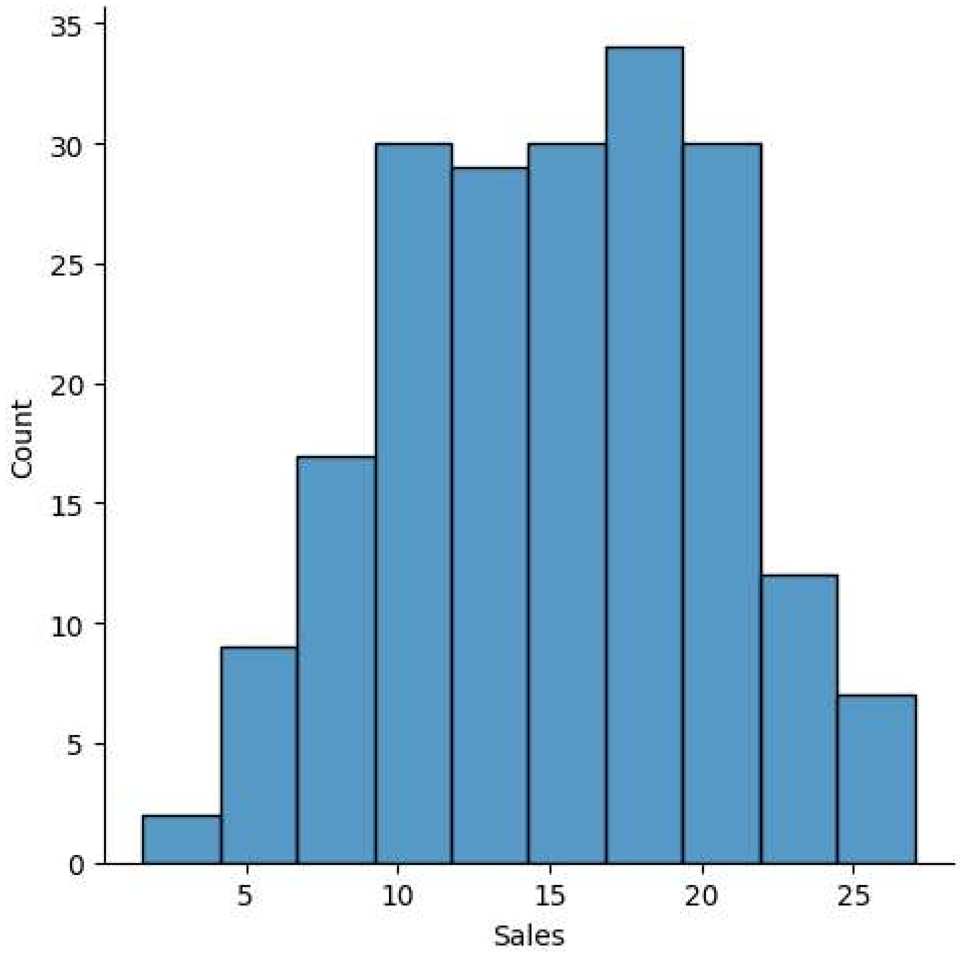


In [12]:

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

Out[12]:

<seaborn.axisgrid.FacetGrid at 0x239e3a8c3d0>

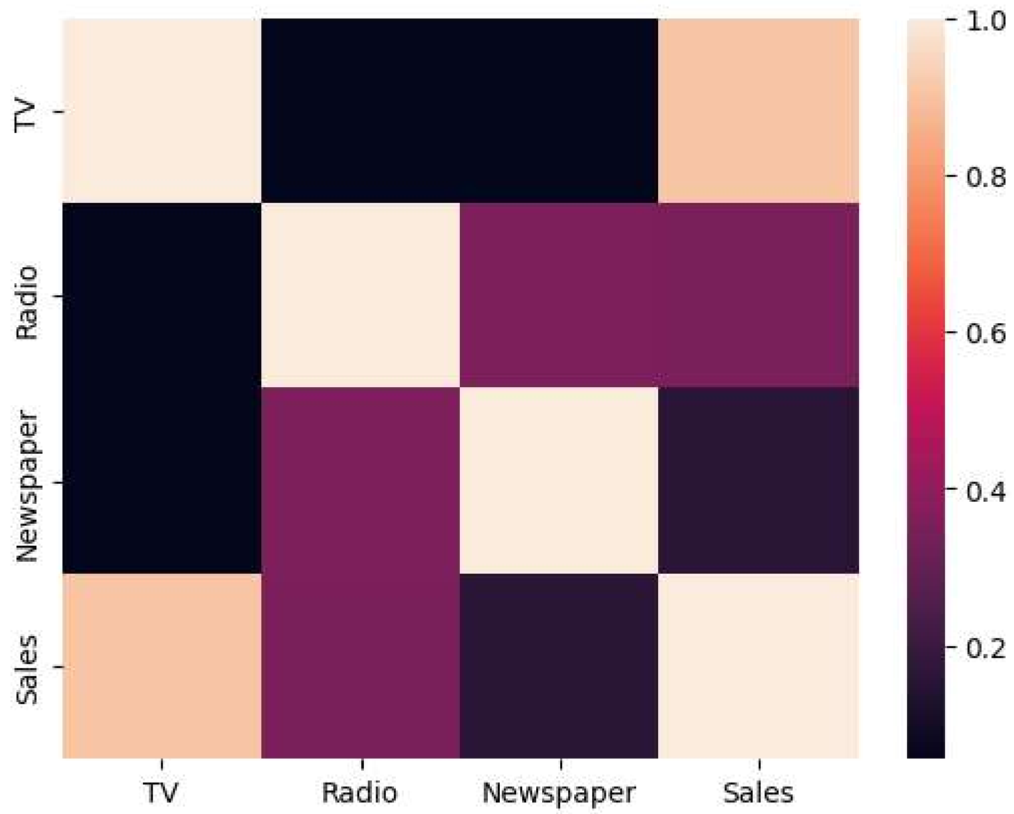


In [13]:

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

Out[13]:

```
<Axes: >
```



In [14]:

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

In [15]:

```
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 [16]:

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

Out[16]:

	coefficient
TV	0.054930
Radio	0.109558
Newspaper	-0.006194

In [17]:

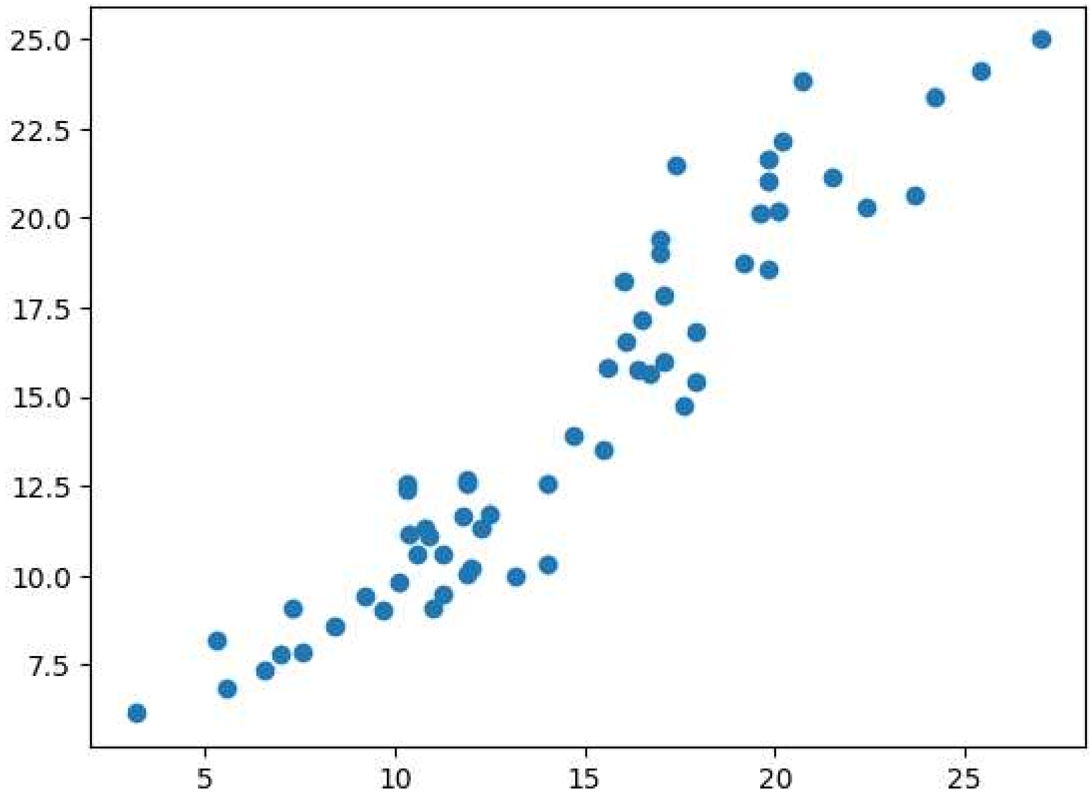
```
predictions=lm.predict(X_test)
```

In [18]:

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

Out[18]:

```
<matplotlib.collections.PathCollection at 0x239e3f88710>
```

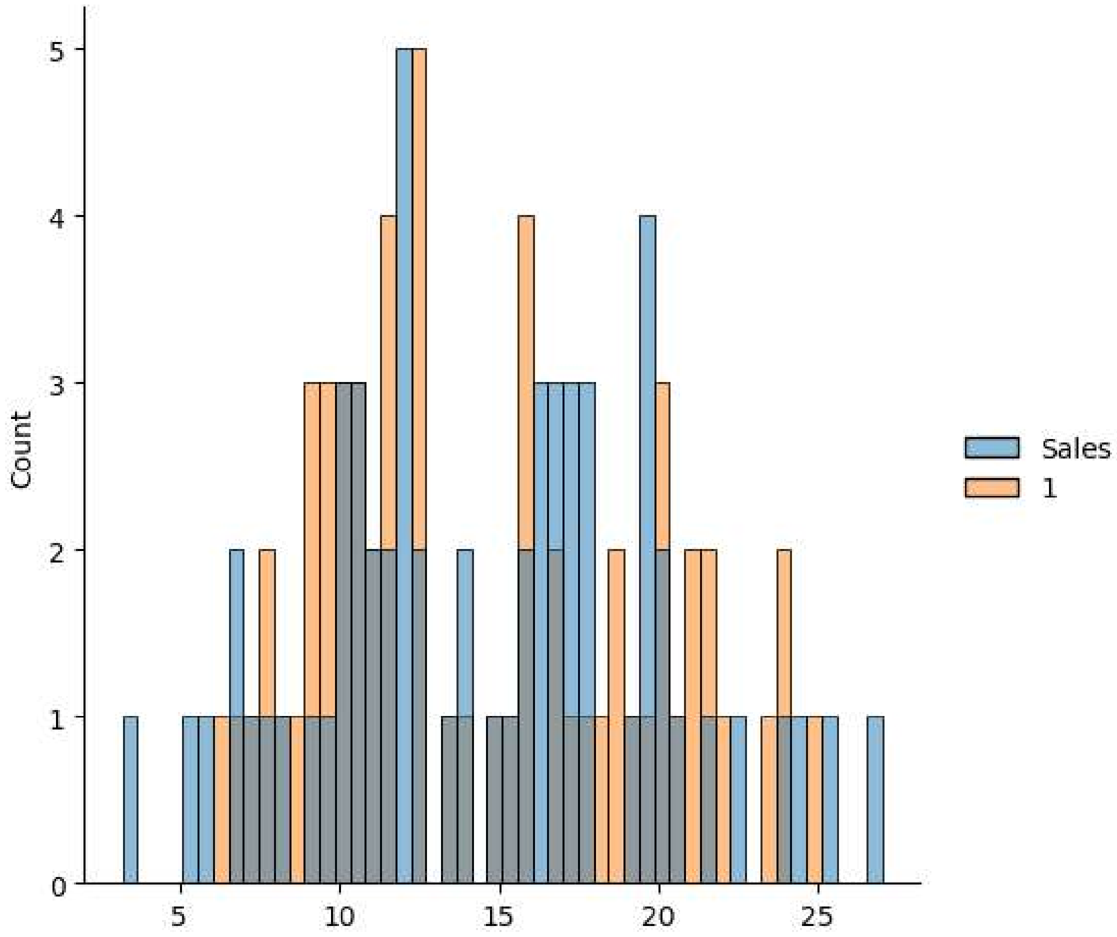


In [19]:

```
sns.displot((y_test,predictions),bins=50)#without semicolon
```

Out[19]:

```
<seaborn.axisgrid.FacetGrid at 0x239e3edfcd0>
```



In [20]:

```
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)))
```

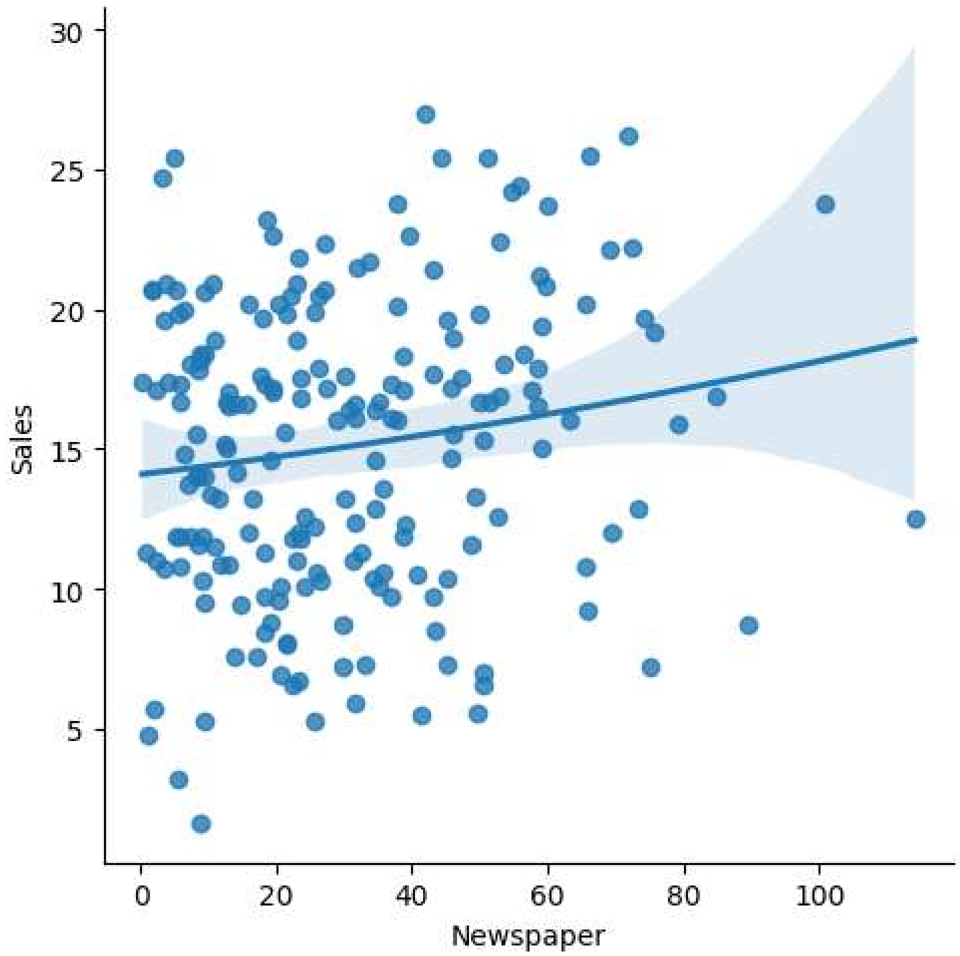
```
MAE: 1.3731200698367851
MSE: 2.8685706338964967
MAE: 1.6936855180040056
```

In [21]:

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

Out[21]:

<seaborn.axisgrid.FacetGrid at 0x239e708a9d0>

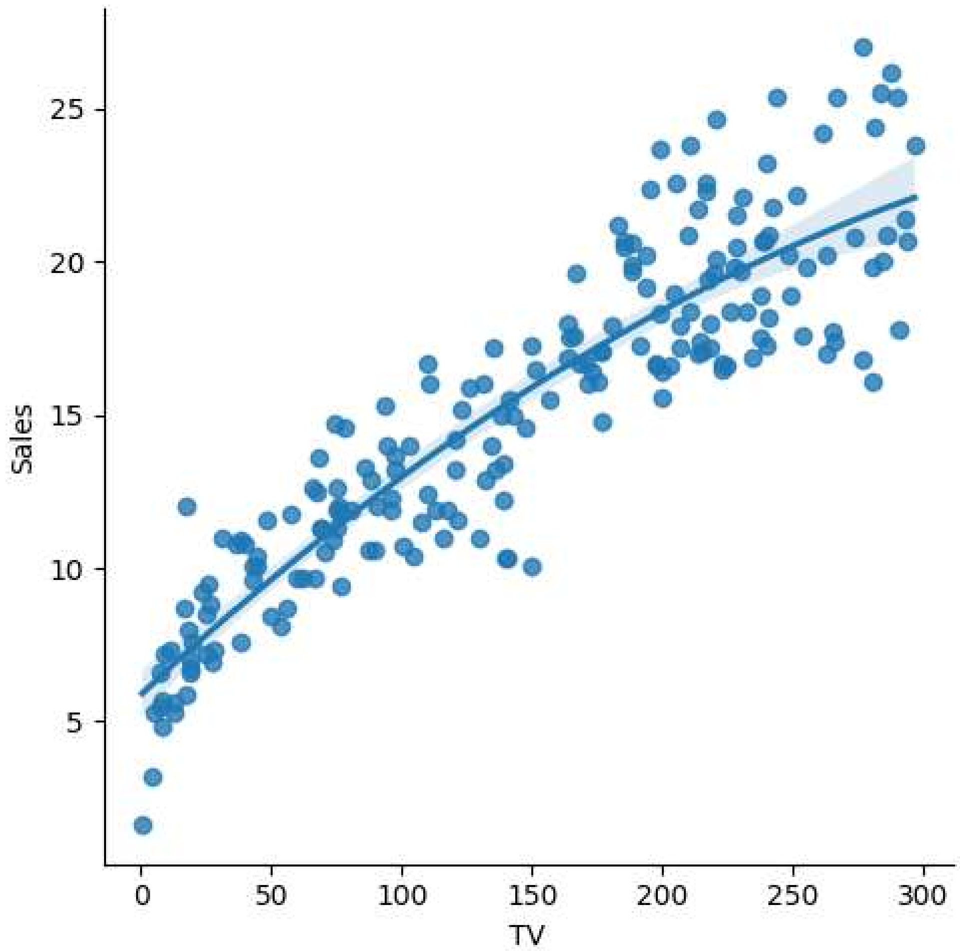


In [22]:

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

Out[22]:

```
&lt;seaborn.axisgrid.FacetGrid at 0x239e6fae190&gt;
```

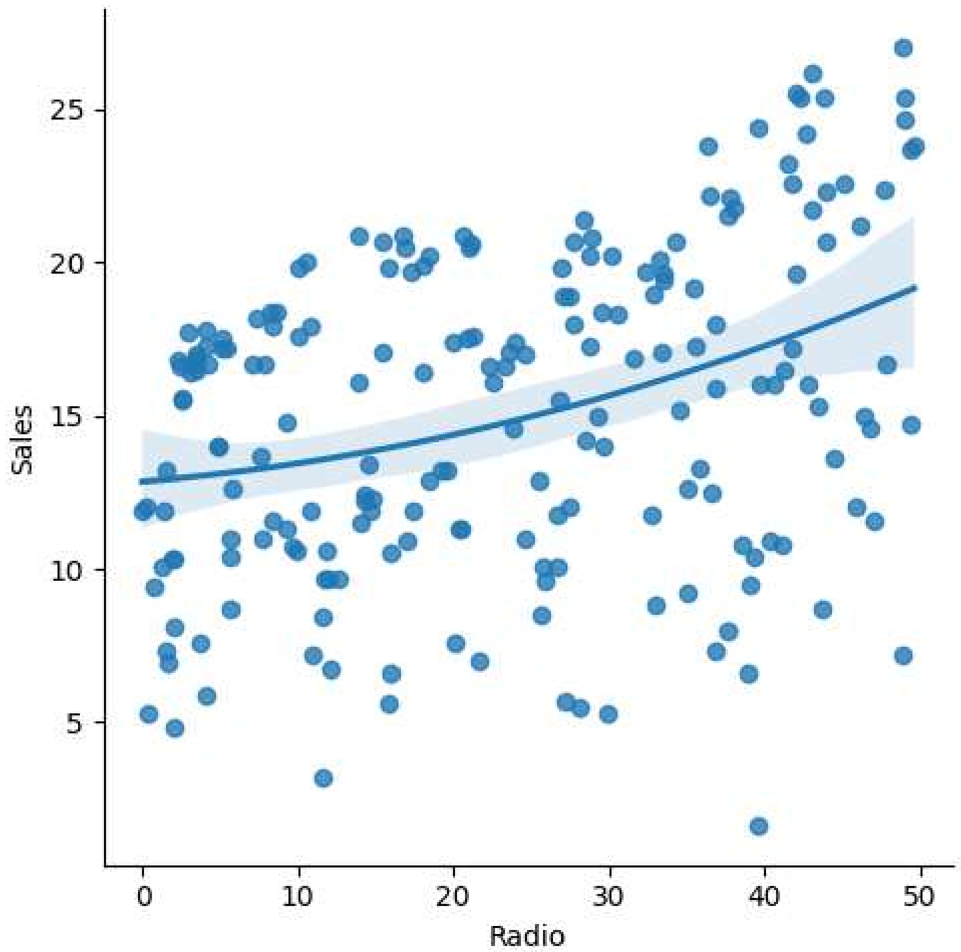


In [23]:

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

Out[23]:

```
<seaborn.axisgrid.FacetGrid at 0x239e7185f10>
```



In [24]:

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

In [25]:

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

In [26]:

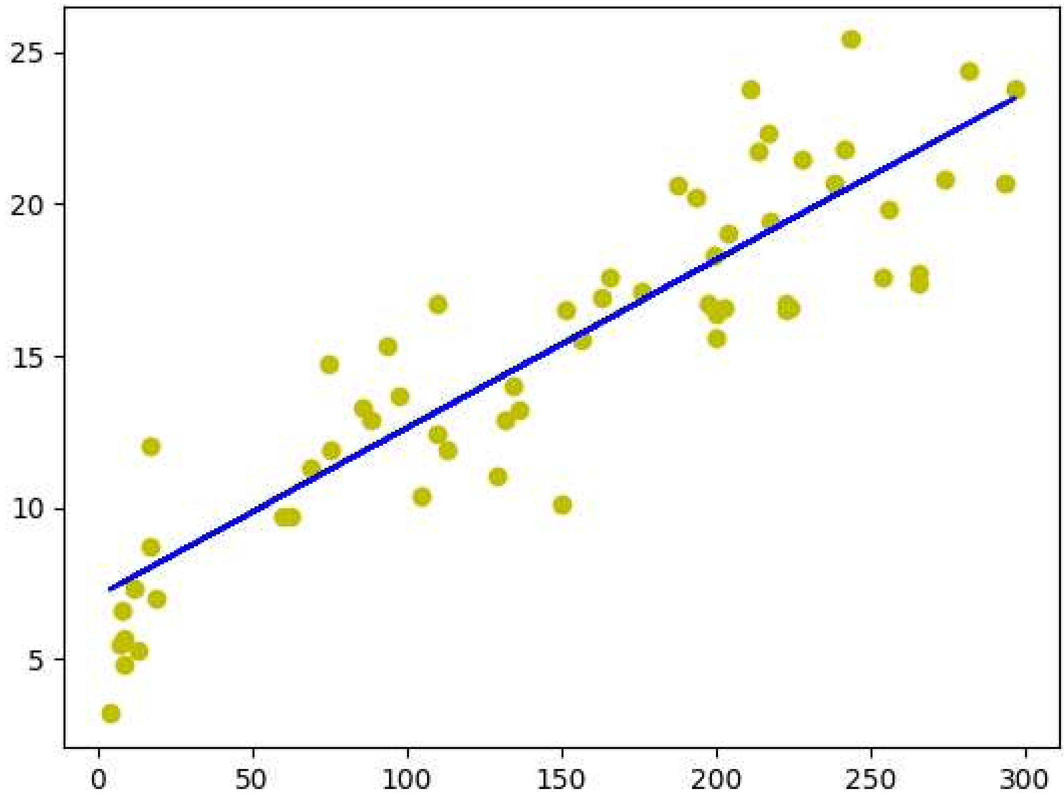
```
X_train,X_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
regr.fit(X_train,y_train)
regr.fit(X_train,y_train)
```

Out[26]:

```
▾ LinearRegression
LinearRegression()
```

In [27]:

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

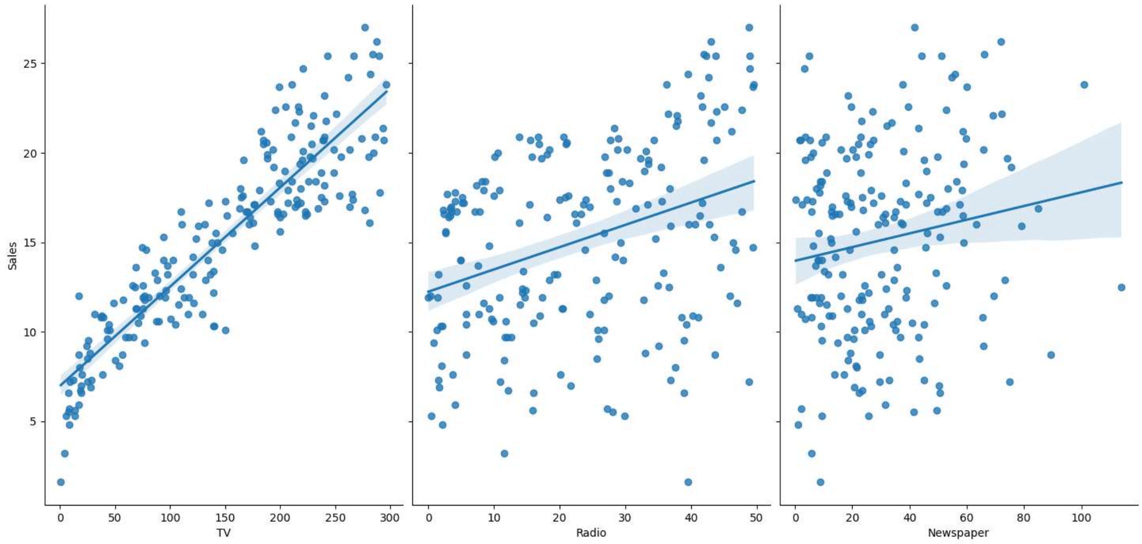


In [28]:

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

Out[28]:

```
<seaborn.axisgrid.PairGrid at 0x239e71ee390>
```



In [29]:

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

```
0.805593041457866
```

In [30]:

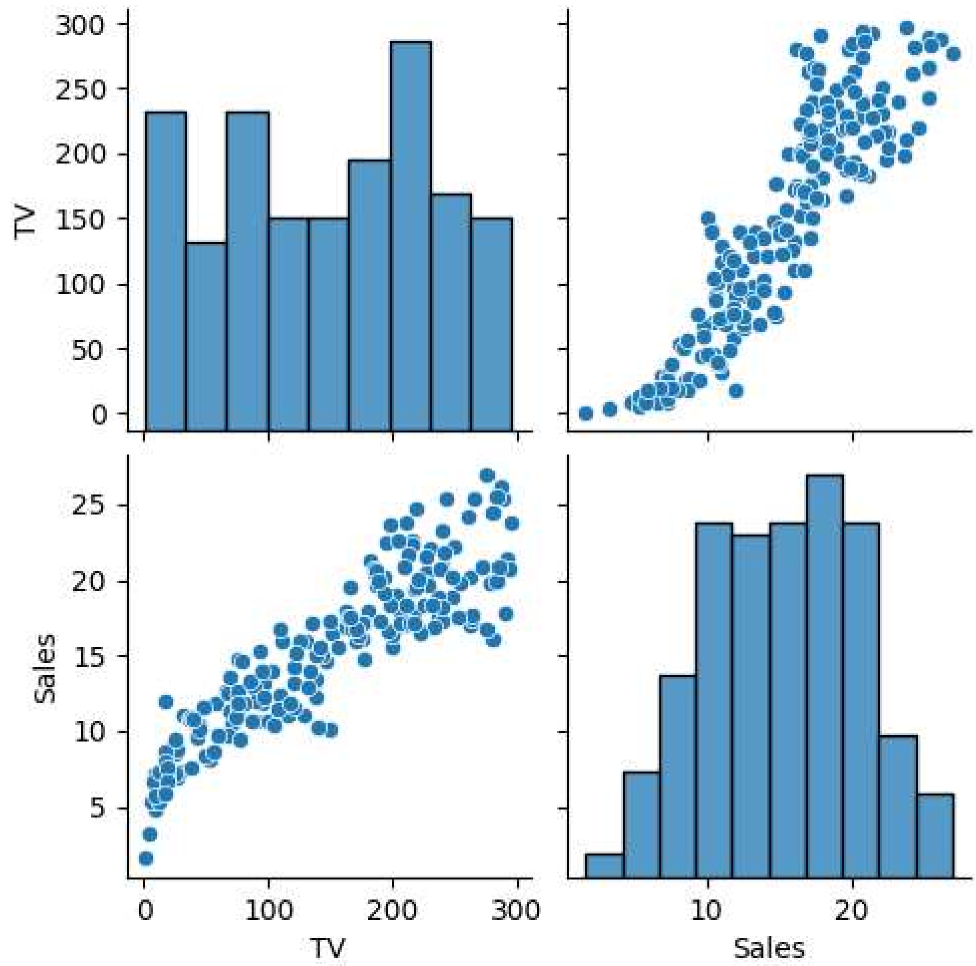
```
from sklearn.linear_model import Lasso,Ridge
from sklearn.preprocessing import StandardScaler
```

In [31]:

```
ddf=df[['TV', 'Radio', 'Newspaper', 'Sales']]
```

In [32]:

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



In [33]:

```
features=df.columns[0:2]
target=df.columns[-1]
X=df[features].values
y=df[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 [34]:

```
#Linear regression model
regr=LinearRegression()
regr.fit(X_train,y_train)
actual=y_test #actual value
train_score_regr=regr.score(X_train,y_train)
test_score_regr=regr.score(X_test,y_test)
print("\nLinear model:\n")
print("The train score for Linear model is {}".format(train_score_regr))
print("The test score for Linear model is {}".format(test_score_regr))
```

```
Linear model:

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

In [35]:

```
#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 0.9902871391941609
The test score for ridge model is 0.984426628514122
```

In [36]:

```
#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))
```

```
0.999999999997627
0.9999999999962466
```

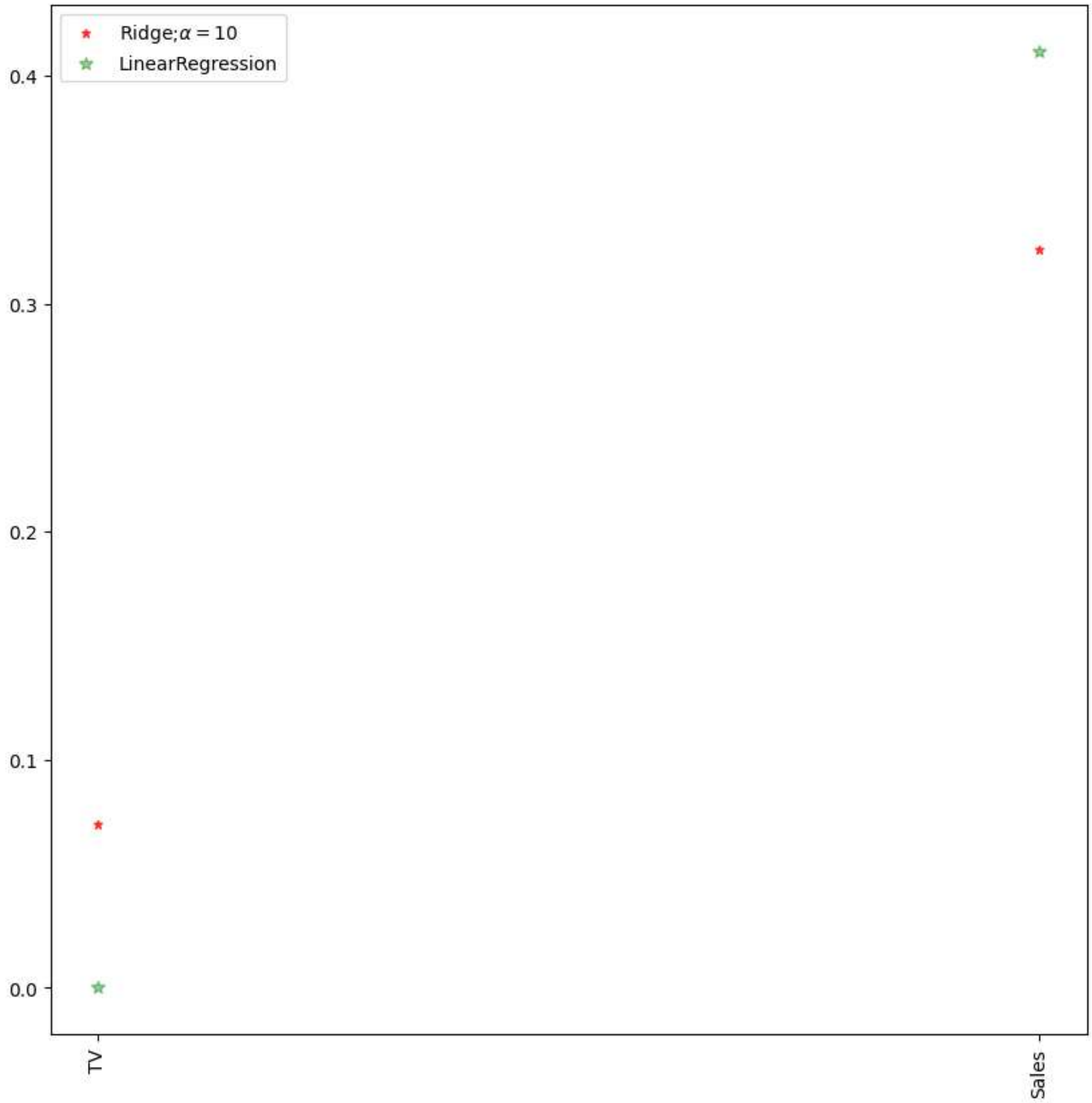
In [37]:

```
#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 [38]:

```
plt.figure(figsize=(10,10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5,colo
 label=r'Ridge;$\alpha=10$',zorder=7)
plt.plot(features,regr.coef_,alpha=0.4,linestyle='none',marker='*',markersize=7,color='g
plt.xticks(rotation=90)
plt.legend()
plt.show()
```

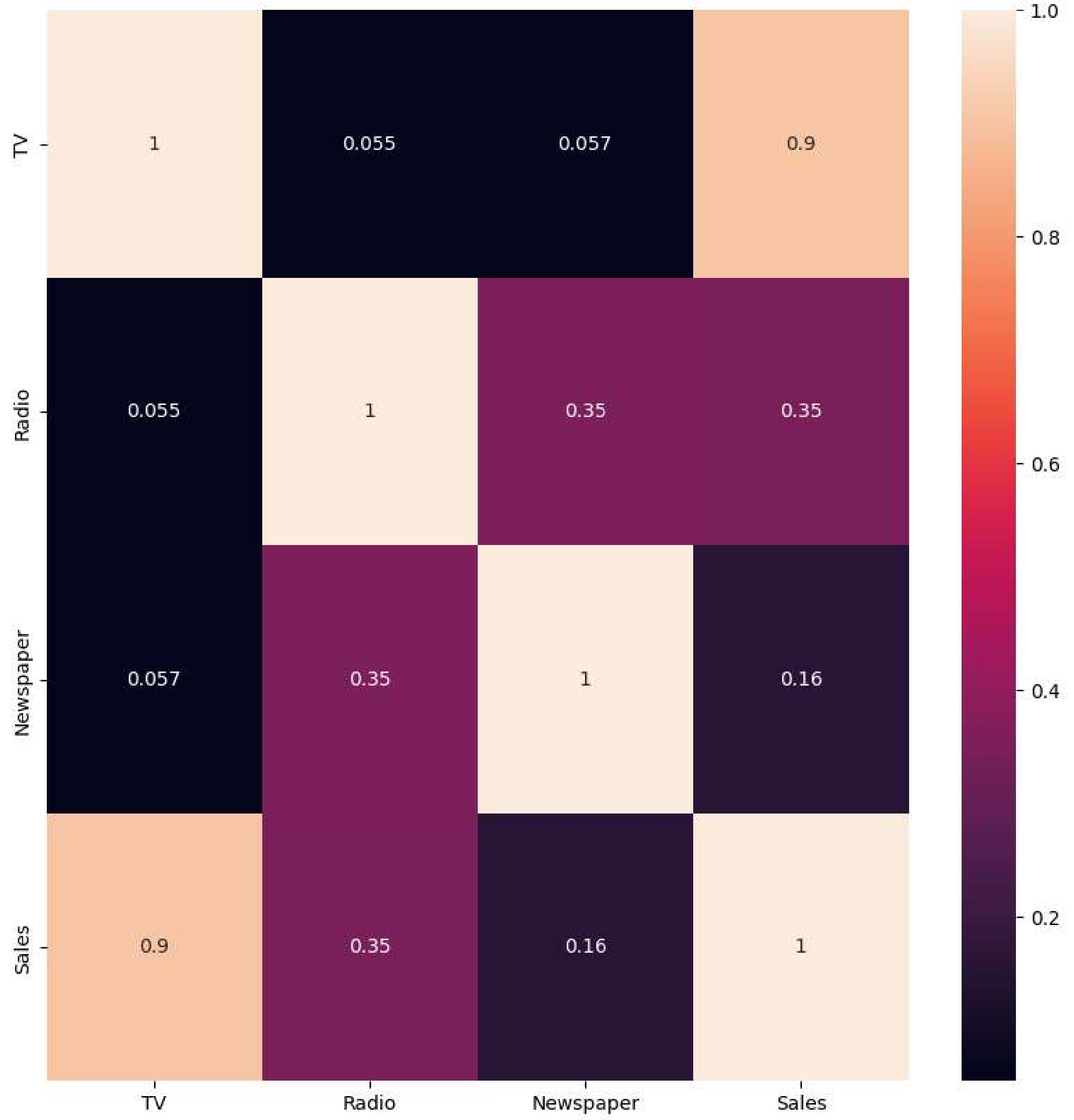


In [39]:

```
#ridge regression
plt.figure(figsize=(10,10))
sns.heatmap(ddf.corr(),annot=True)
```

Out[39]:

```
<Axes: >
```



In [40]:

```
#lasso regression model
lassoReg=Lasso(alpha=10)
lassoReg.fit(X_train,y_train)
#train and test score for ridge 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))
```

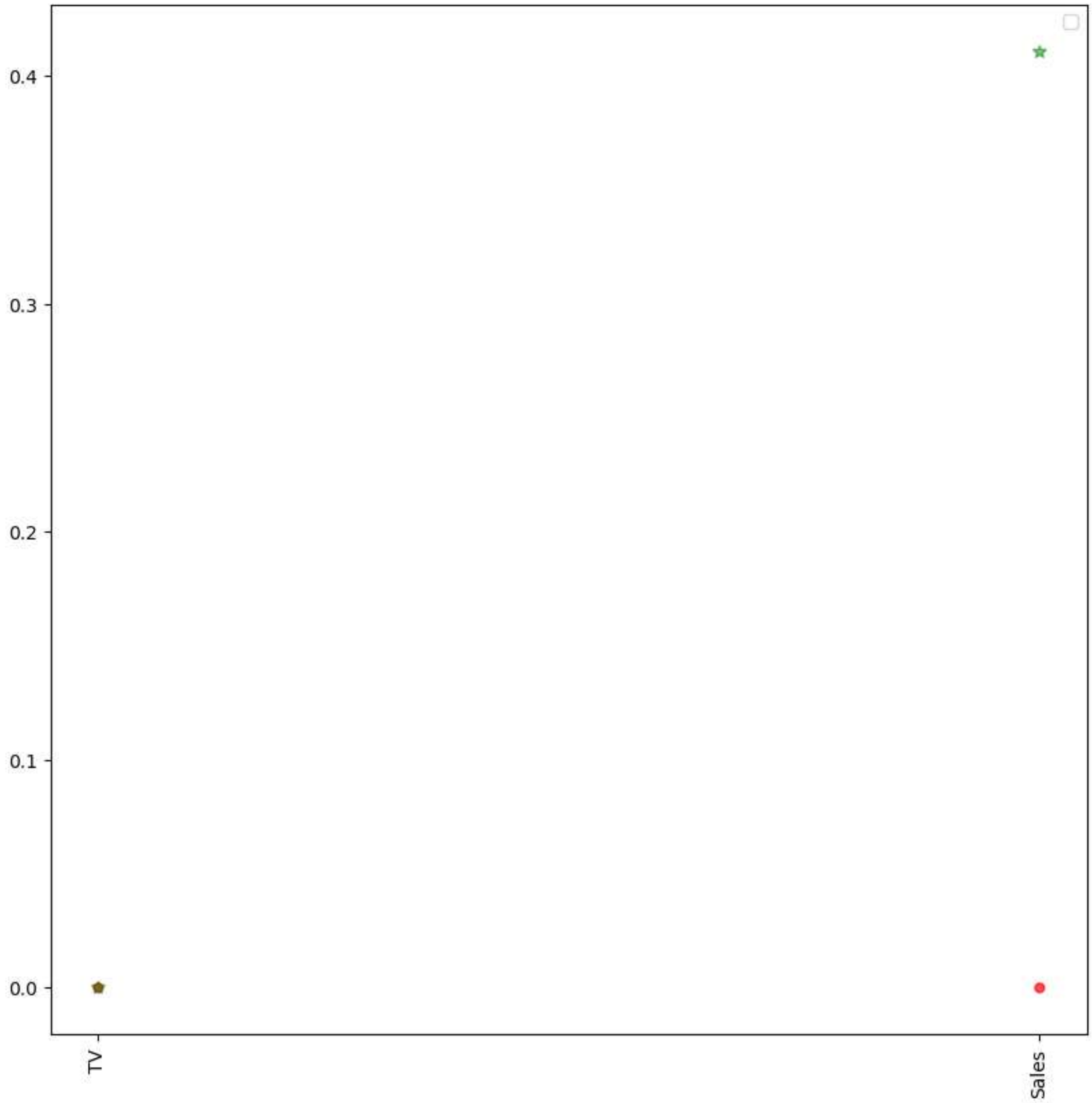
```
Lasso model:

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

In [41]:

```
plt.figure(figsize=(10,10))
plt.plot(features,lassoReg.coef_,alpha=0.7,linestyle='none',marker='o',markersize=5,colo
plt.plot(features,regr.coef_,alpha=0.5,linestyle='none',marker='*',markersize=7,color='g
plt.xticks(rotation=90)
plt.legend()
plt.show()
```

```
No artists with labels found to put in legend.  Note that artists whose la
bel start with an underscore are ignored when legend() is called with no a
rgument.
```

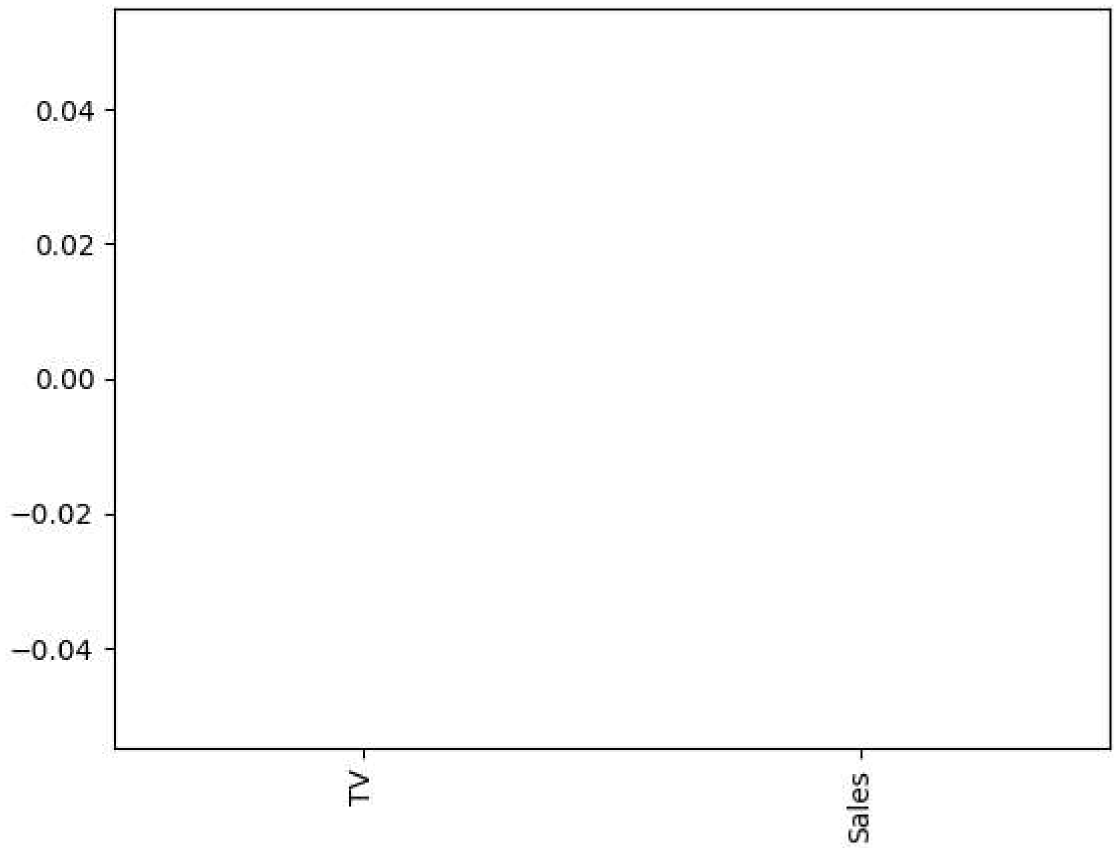


In [42]:

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

Out[42]:

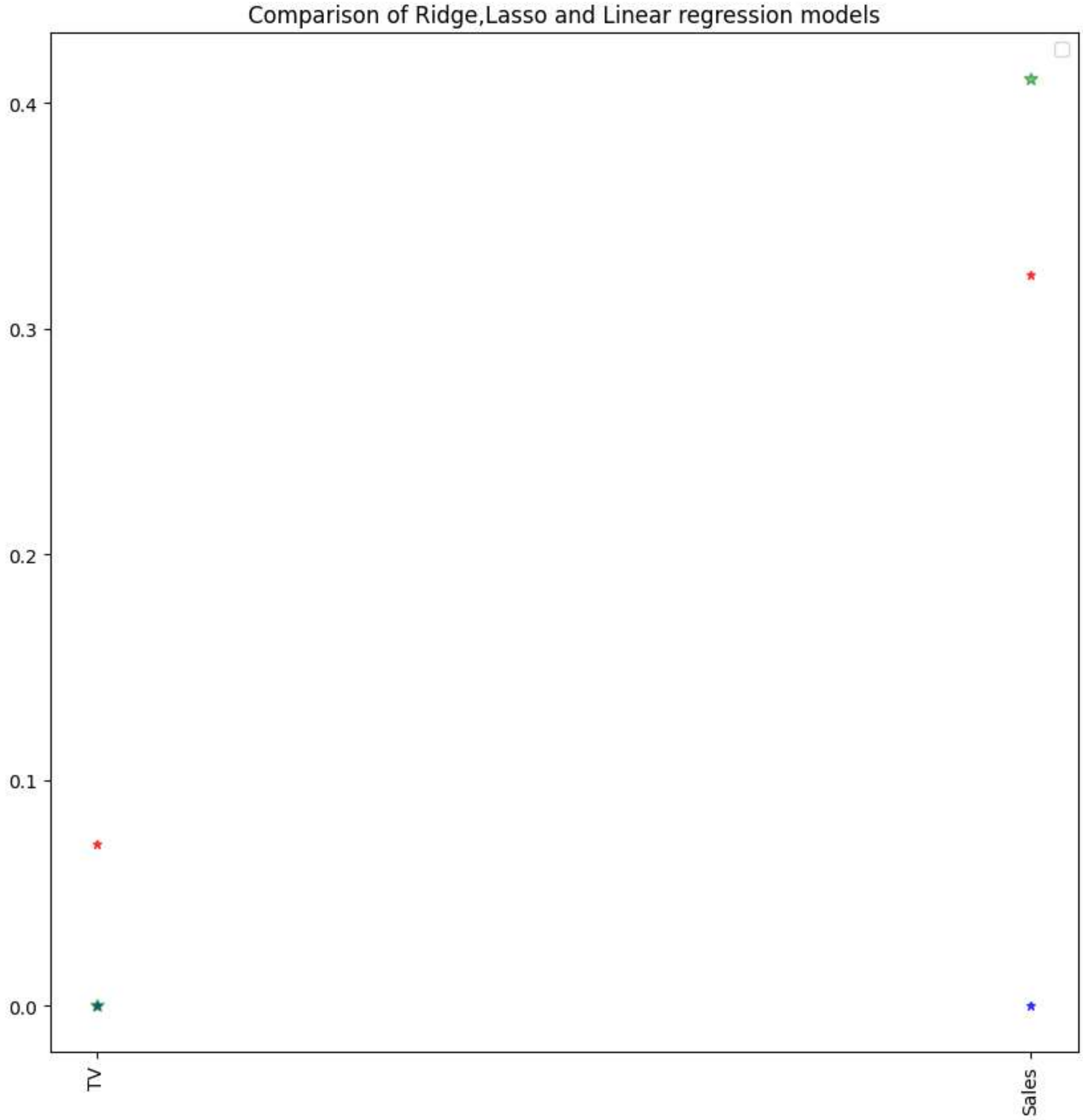
<Axes: >



In [43]:

```
#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(features,lassoReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5,colo
#add plot for linear model
plt.plot(features,regr.coef_,alpha=0.5,linestyle='none',marker='*',markersize=7,color='g
#rotate axis
plt.xticks(rotation=90)
plt.legend()
plt.title("Comparison of Ridge,Lasso and Linear regression models")
plt.show()
```

```
No artists with labels found to put in legend.  Note that artists whose la
bel start with an underscore are ignored when legend() is called with no a
rgument.
```



In [44]:

```
#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.        ]
2.026383919311004
Mean Squared Error on test set 0.5538818050142158
```