In [1]:
```
import numpy as np
import pandas as pd
```

In [2]:
```
df=pd.read_csv(r"C:\Users\USER\Downloads\Advertising.csv")
df
```

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]:
```
df.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]:
```
df.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]:
```
df.shape
```

Out[5]:
```
(200, 4)
```

In [6]:
```
df.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]:
```
df.info()
```

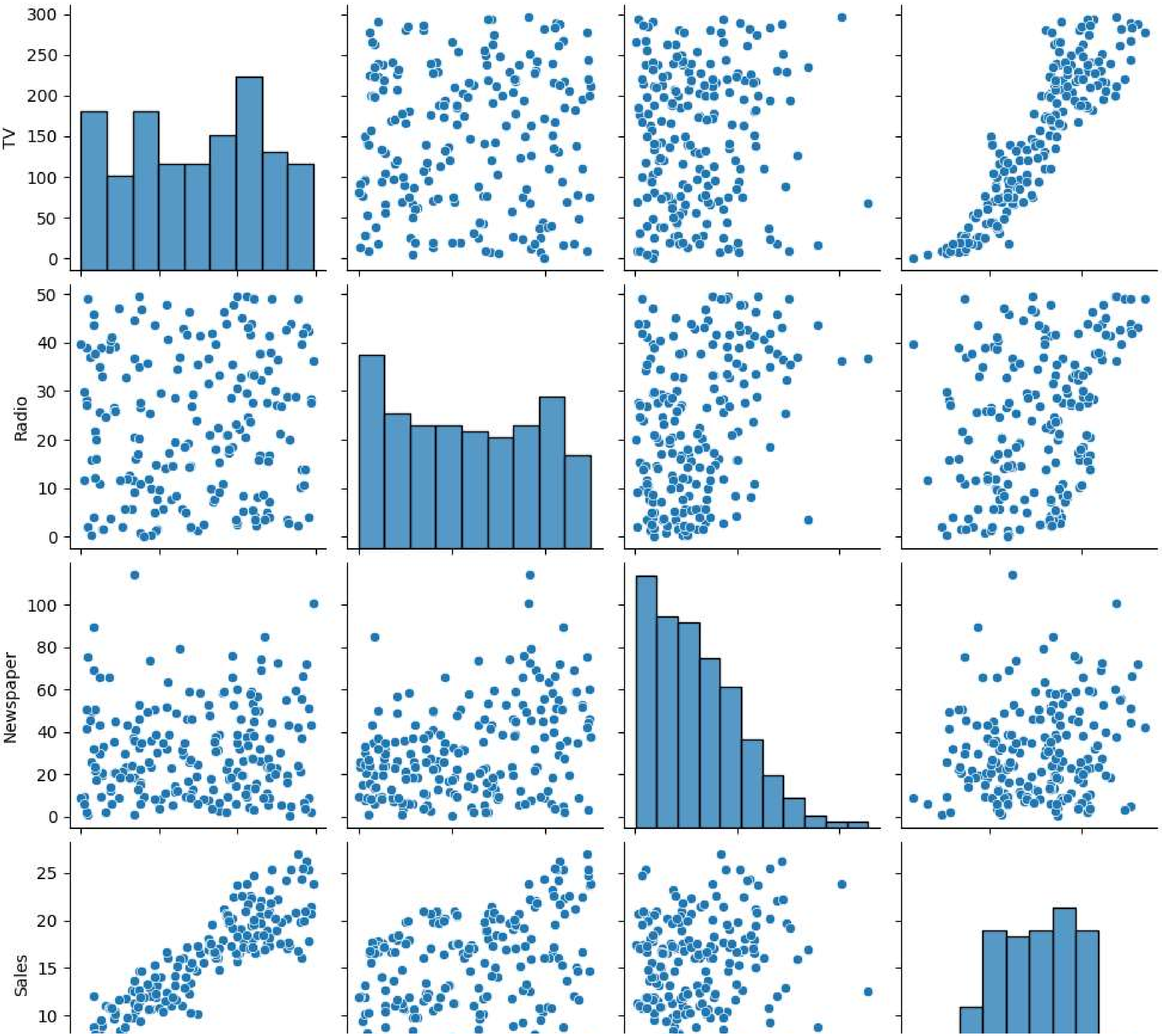
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   TV         200 non-null    float64
 1   Radio      200 non-null    float64
 2   Newspaper  200 non-null    float64
 3   Sales      200 non-null    float64
dtypes: float64(4)
memory usage: 6.4 KB
```

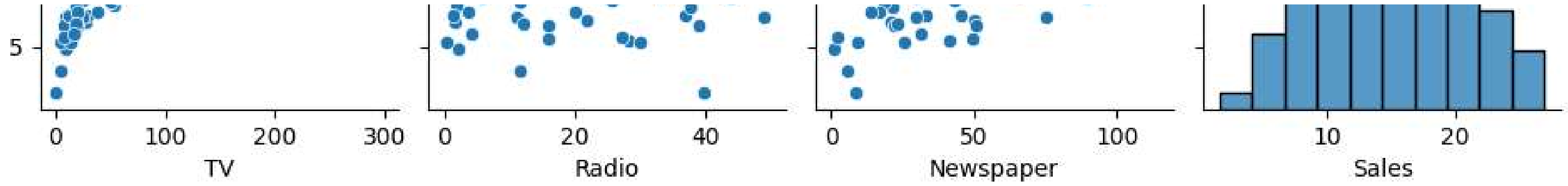
In [8]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

In [9]:
```
df.isna().any()
```

Out[9]:
```
TV           False
Radio        False
Newspaper    False
Sales        False
dtype: bool
```

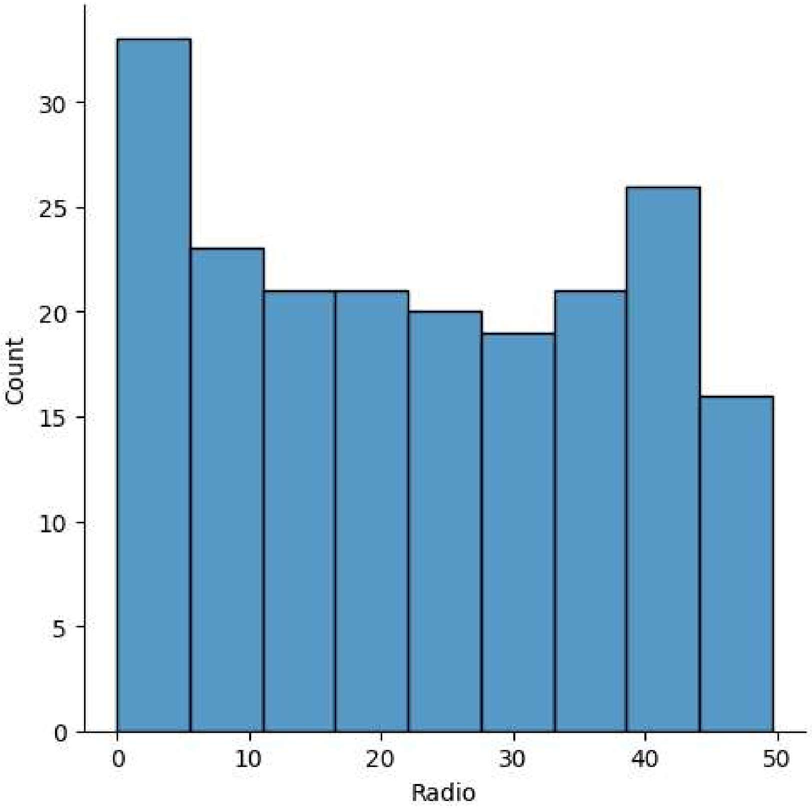
```
In [10]: sns.pairplot(df)

Out[10]: <seaborn.axisgrid.PairGrid at 0x1dadbe57d50>
```


300
250
200
150
100
50
0
TV
50
40
30
20
10
0
Radio
100
80
60
40
20
0
Newspaper
25
20
15
10
Sales



```
In [11]: sns.displot(df['Radio'])
```

```
Out[11]: <seaborn.axisgrid.FacetGrid at 0x1dae5bc9110>
```

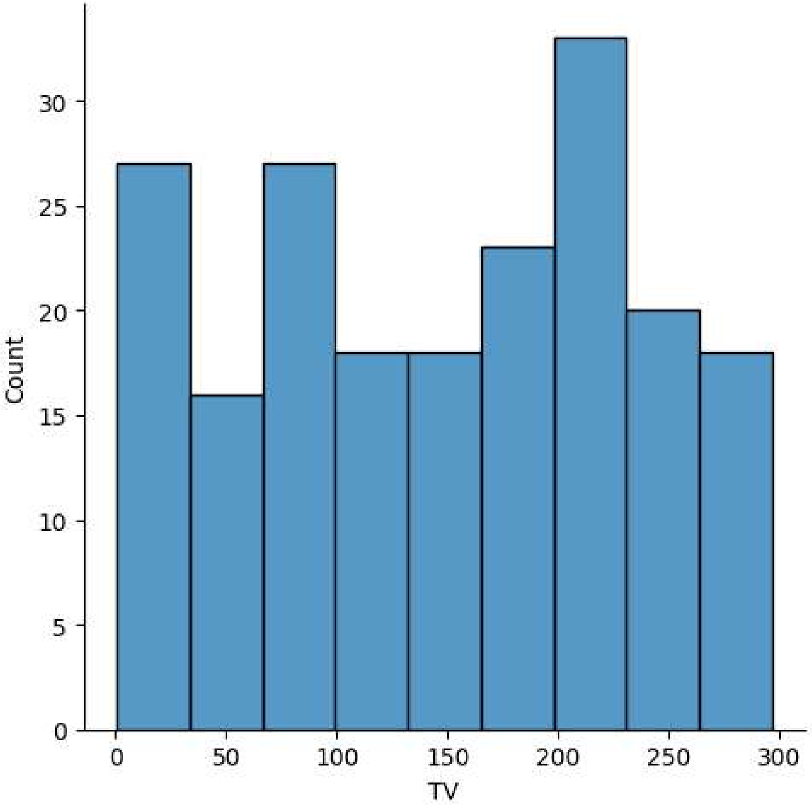


In [12]:

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

Out[12]:

```
<seaborn.axisgrid.FacetGrid at 0x1dae856e0d0>
```



In [13]:

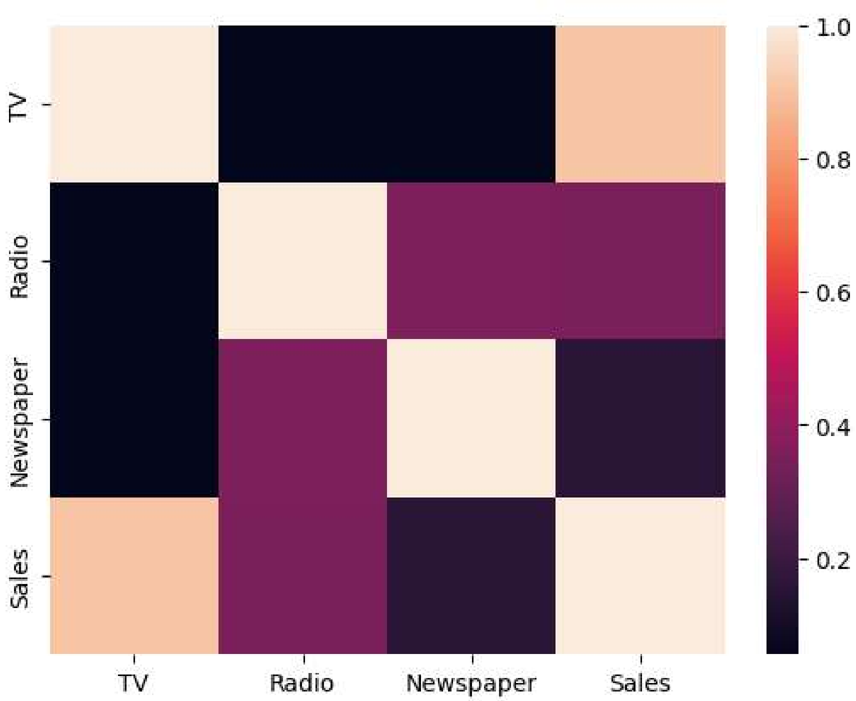
```
df.columns
```

Out[13]:

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

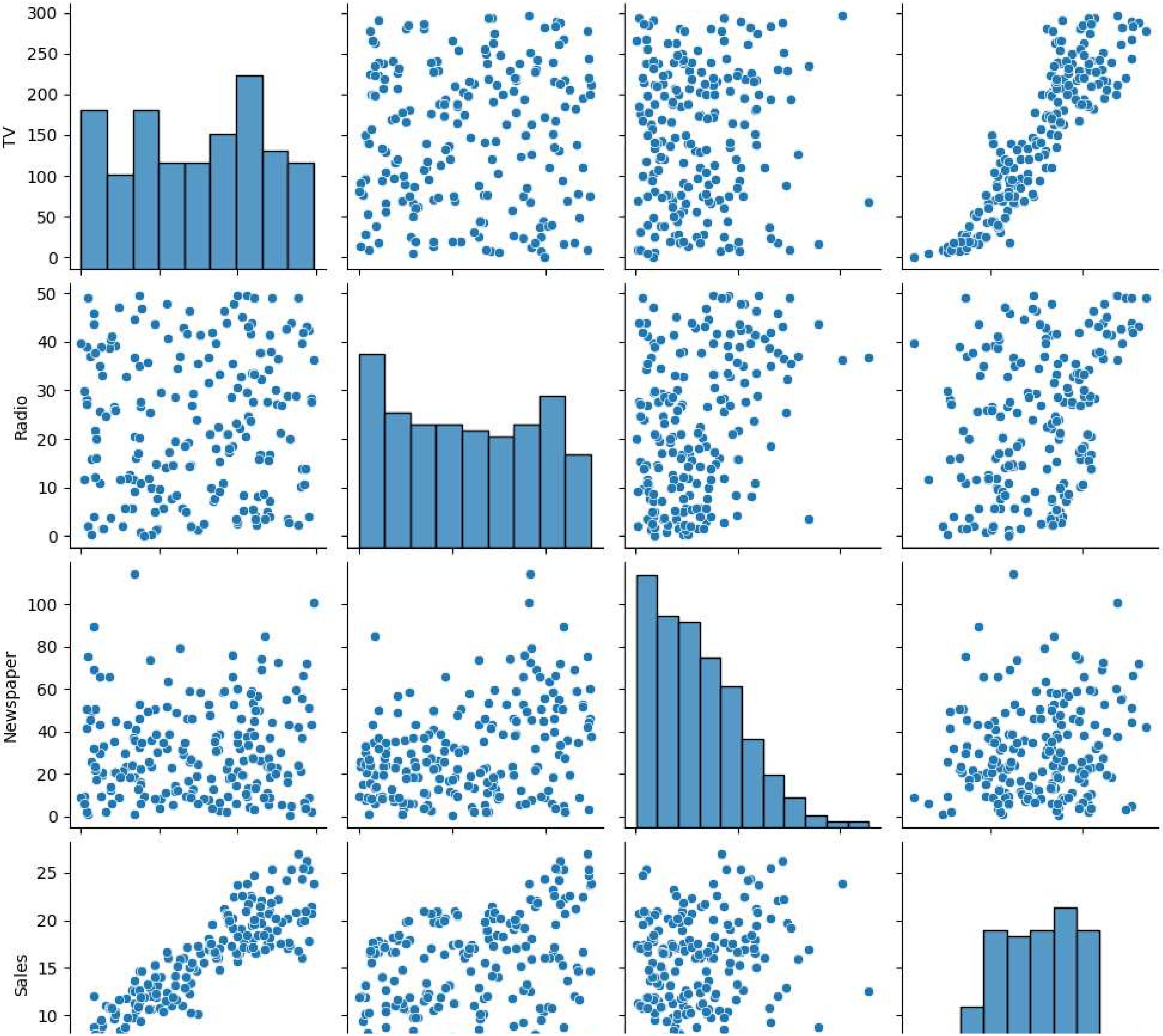
```
In [14]: sns.heatmap(df.corr())
```

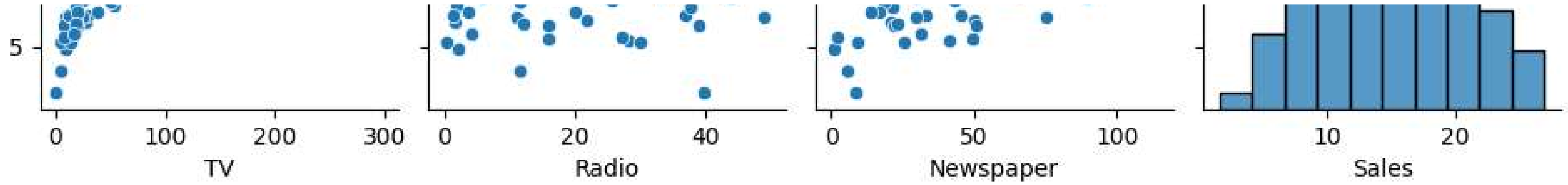
```
Out[14]: <Axes: >
```



In [15]:

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


TV
300
250
200
150
100
50
0
Radio
50
40
30
20
10
0
Newspaper
100
80
60
40
20
0
Sales
25
20
15
10



In [17]:

```
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)
```

In [18]:

```
from sklearn.preprocessing import StandardScaler
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_train 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_train is (60, 2)
```

In [19]:

```
from sklearn.linear_model import LinearRegression
lm=LinearRegression()
lm.fit(x_train,y_train)
```

Out[19]:

```
▾ LinearRegression
LinearRegression()
```

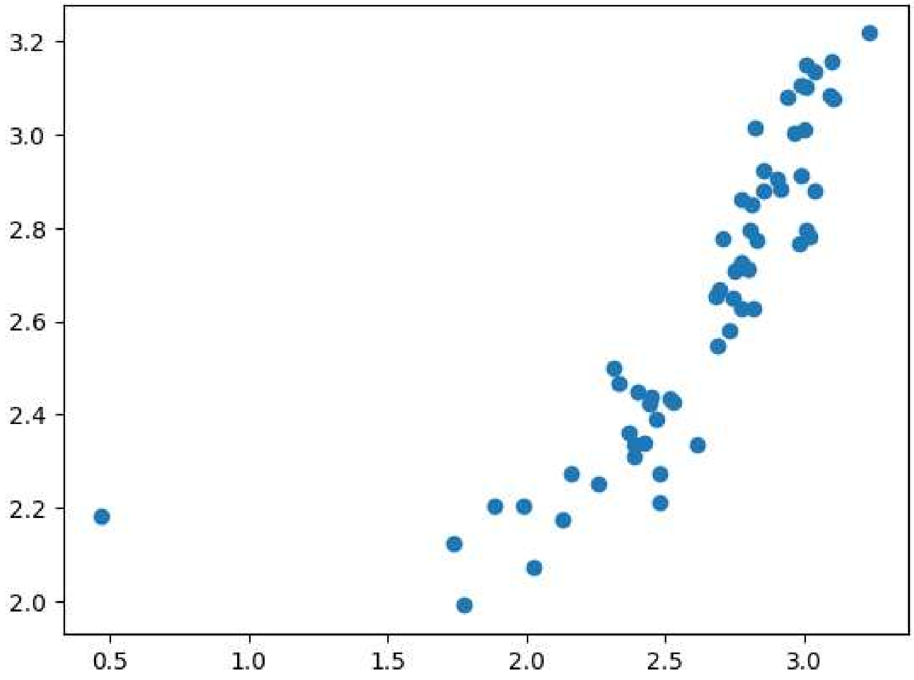
In [20]:

```
print(lm.intercept_)
```

```
2.649682499818669
```

```
In [21]: predictions=lm.predict(x_test)
         plt.scatter(y_test,predictions)
```

```
Out[21]: <matplotlib.collections.PathCollection at 0x1daebcd8290>
```

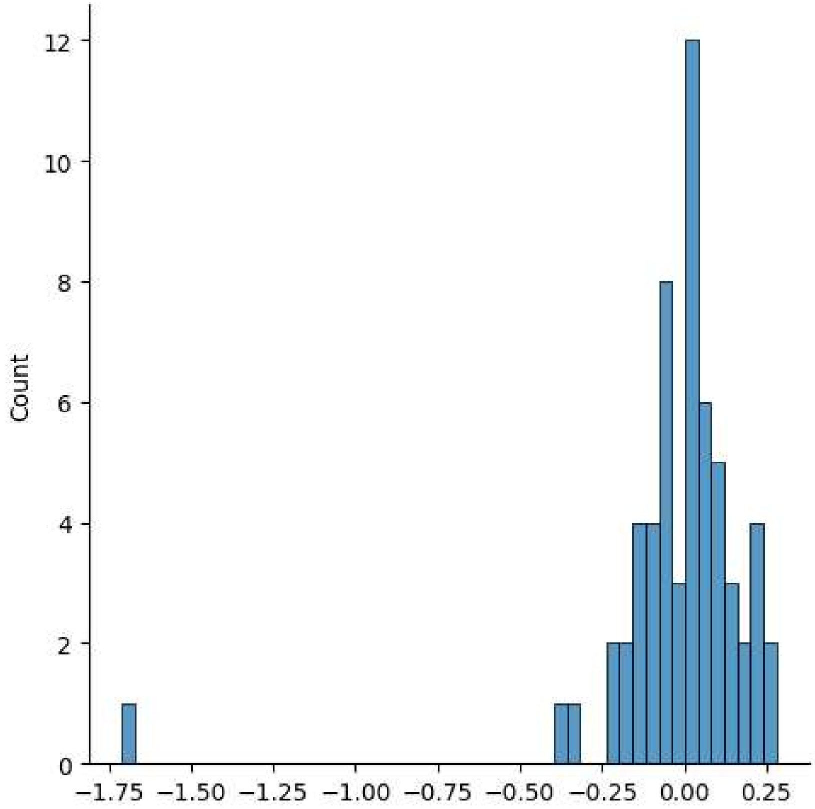


```
In [22]: from sklearn import metrics
         print('MAE:',metrics.mean_absolute_error(y_test,predictions))
         print('MSE:',metrics.mean_squared_error(y_test,predictions))
         print('RMSE:',np.sqrt(metrics.mean_absolute_error(y_test,predictions)))
```

```
MAE: 0.13137245926116148
MSE: 0.0667845950252303
RMSE: 0.3624533890877025
```

In [23]:
```
sns.displot((y_test-predictions),bins=50)
```

Out[23]:
```
<seaborn.axisgrid.FacetGrid at 0x1daeaca3b10>
```



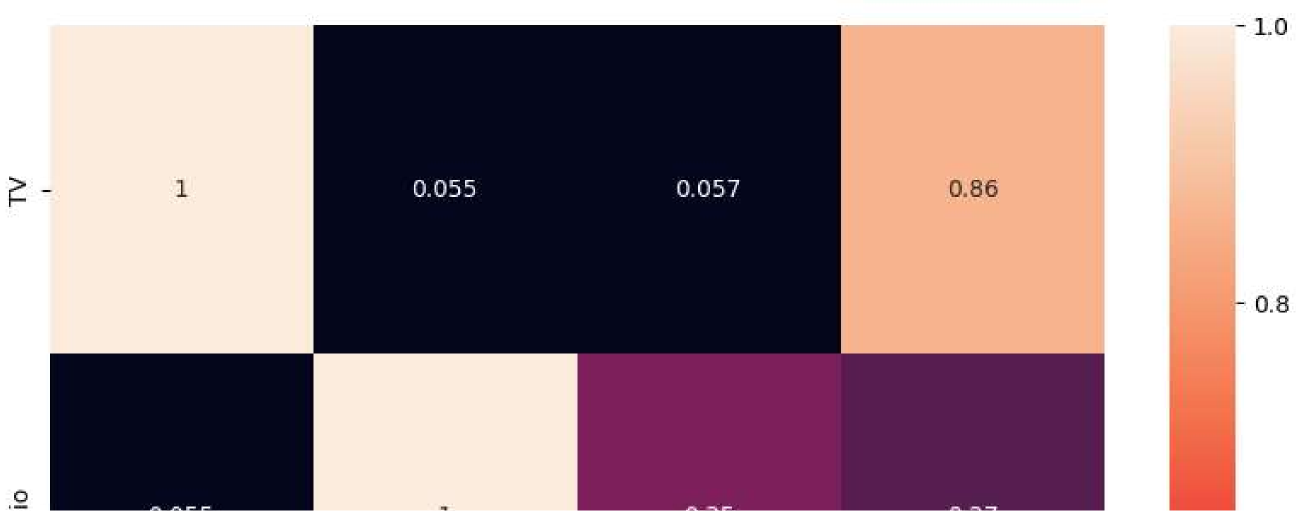
In [24]:
```
print(lm.coef_)
```

```
[0.35159618 0.10807217]
```

In [25]:
```
from sklearn.linear_model import Lasso,Ridge
```

In [26]:
```
plt.figure(figsize=(10,10))
sns.heatmap(df.corr(),annot=True)
```

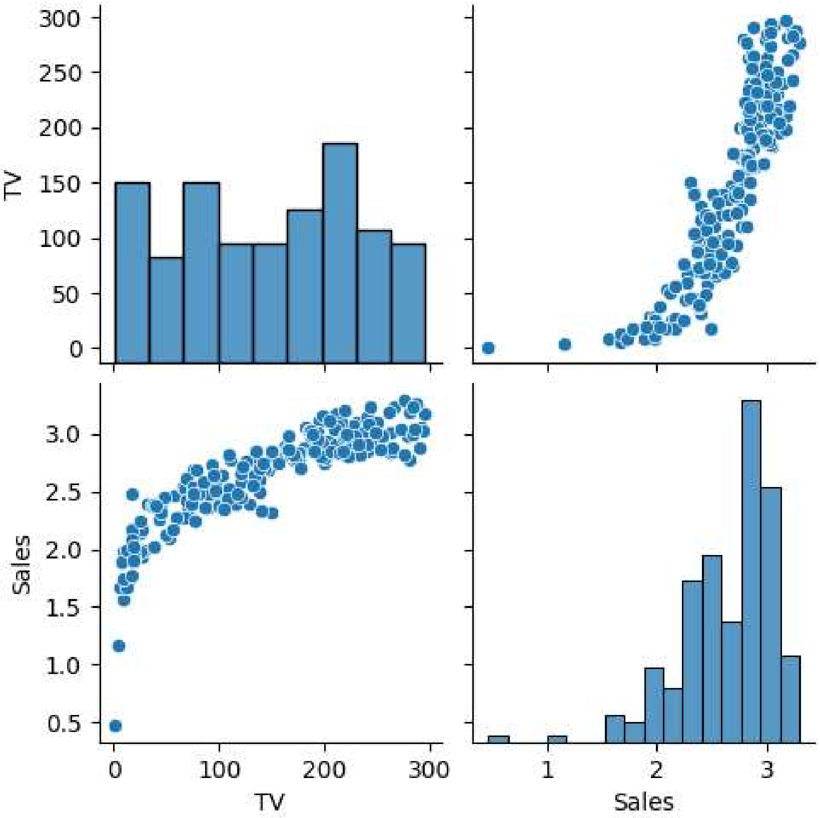
Out[26]:
```
<Axes: >
```



In [27]:
```
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 test score for ridge model is {}".format(test_score_ridge))
```

```

Ridge model:

The train score for ridge model is 0.8500055285406656
The test score for ridge model is 0.6534250193956134
```

In [28]:
```
df.drop(columns=["Radio","Newspaper"],inplace=True)
sns.pairplot(df)
df.Sales=np.log(df.Sales)
```



In [29]:
```
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 test score for lr model is {}".format(test_score_lr))
```

```

Linear Regression model:

The train score for lr model is 0.8534150366313791
The test score for lr model is 0.6654095499889279
```

In [30]:

```
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 test score for ridge model is {}".format(test_score_ridge))
```

```

Ridge model:

The train score for ridge model is 0.8500055285406656
The test score for ridge model is 0.6534250193956134
```

In [31]:

```
import numpy as np
import matplotlib.pyplot as plt
```

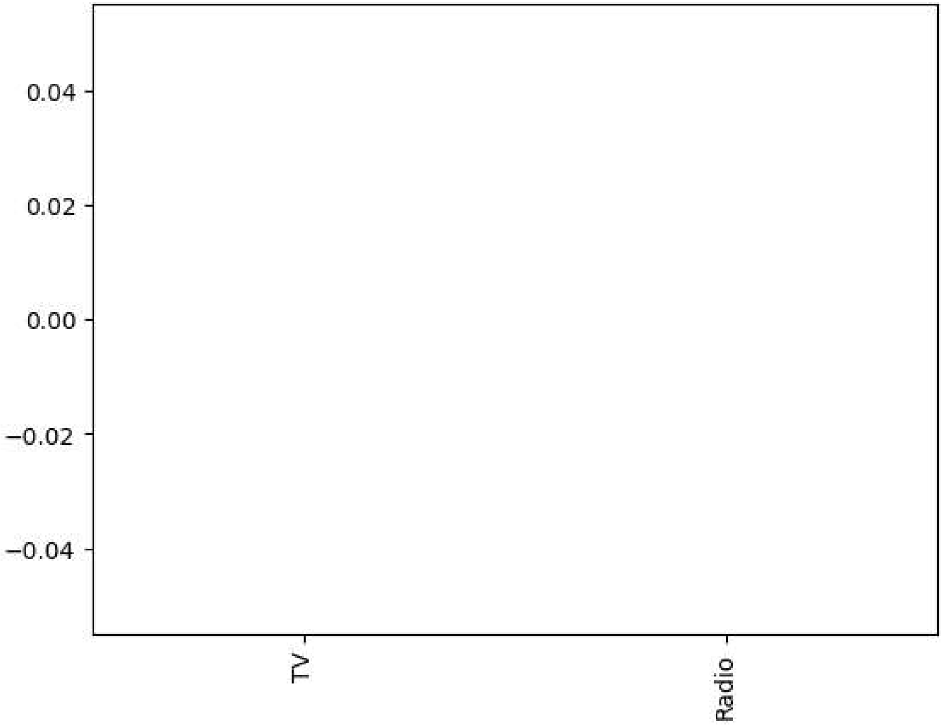
In [32]:

```
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 [33]:
```
pd.Series(lasso.coef_, features).sort_values(ascending = True).plot(kind = "bar")
```

Out[33]:
```
<Axes: >
```

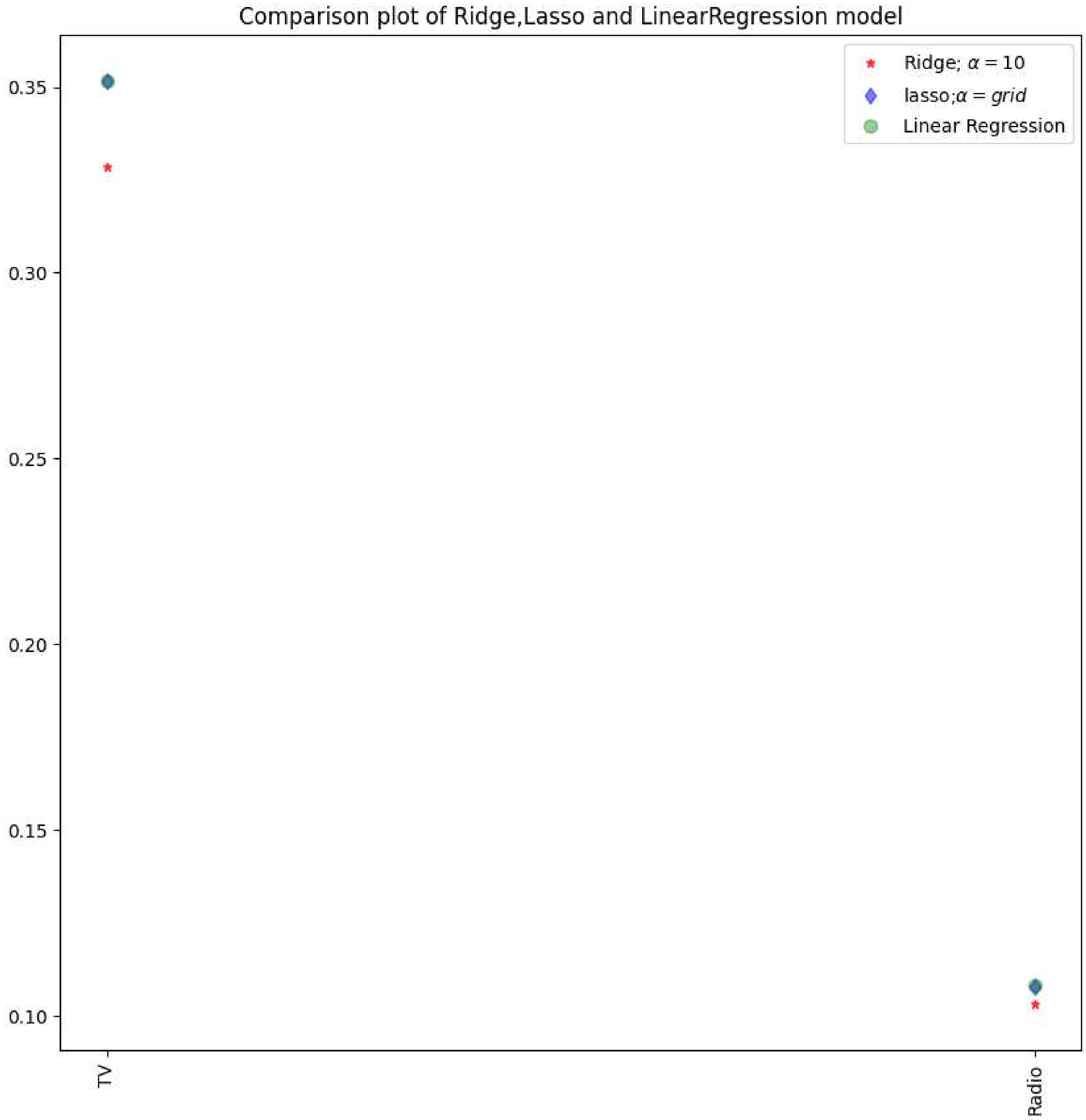


In [34]:
```
from sklearn.linear_model import LassoCV
lasso_cv=LassoCV(alphas=[0.0001,0.001,0.01,0.1,1,10],random_state=0).fit(x_train,y_train)
print(lasso_cv.score(x_train,y_train))
print(lasso_cv.score(x_test,y_test))
```

```
0.8534149297347071
0.6654001015086553
```

In [36]:

```
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=
plt.plot(lasso_cv.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color='blue',label=r'lasso;$\alpha=grid$')
plt.plot(features, lr.coef_, alpha=0.4, linestyle='none', marker='o', markersize=7, color='green',label='Linear Regression')
plt.xticks(rotation=90)
plt.legend()
plt.title('Comparison plot of Ridge,Lasso and LinearRegression model')
plt.show()
```


Comparison plot of Ridge,Lasso and LinearRegression model
Ridge; $\alpha = 10$
lasso;$\alpha = grid$
Linear Regression
0.35
0.30
0.25
0.20
0.15
0.10
TV
Radio

In [38]:

```
#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 0.8534146479788303
The train score for ridge model is 0.6653038057268583
```

In [48]:

```
from sklearn.linear_model import ElasticNet
regr=ElasticNet()
regr.fit(x_train,y_train)
print(regr.coef_)
print(regr.intercept_)
```

```
[0. 0.]
2.649682499818669
```

In [52]:

```
y_pred_elastic=regr.predict(x_train)
```

In [55]:

```
mean_squared_error=np.mean((y_pred_elastic- y_train)**2)
print("mean _squared_error",mean_squared_error)
```

```
mean _squared_error 0.16840246163748074
```

In [57]:

```
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=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)
```

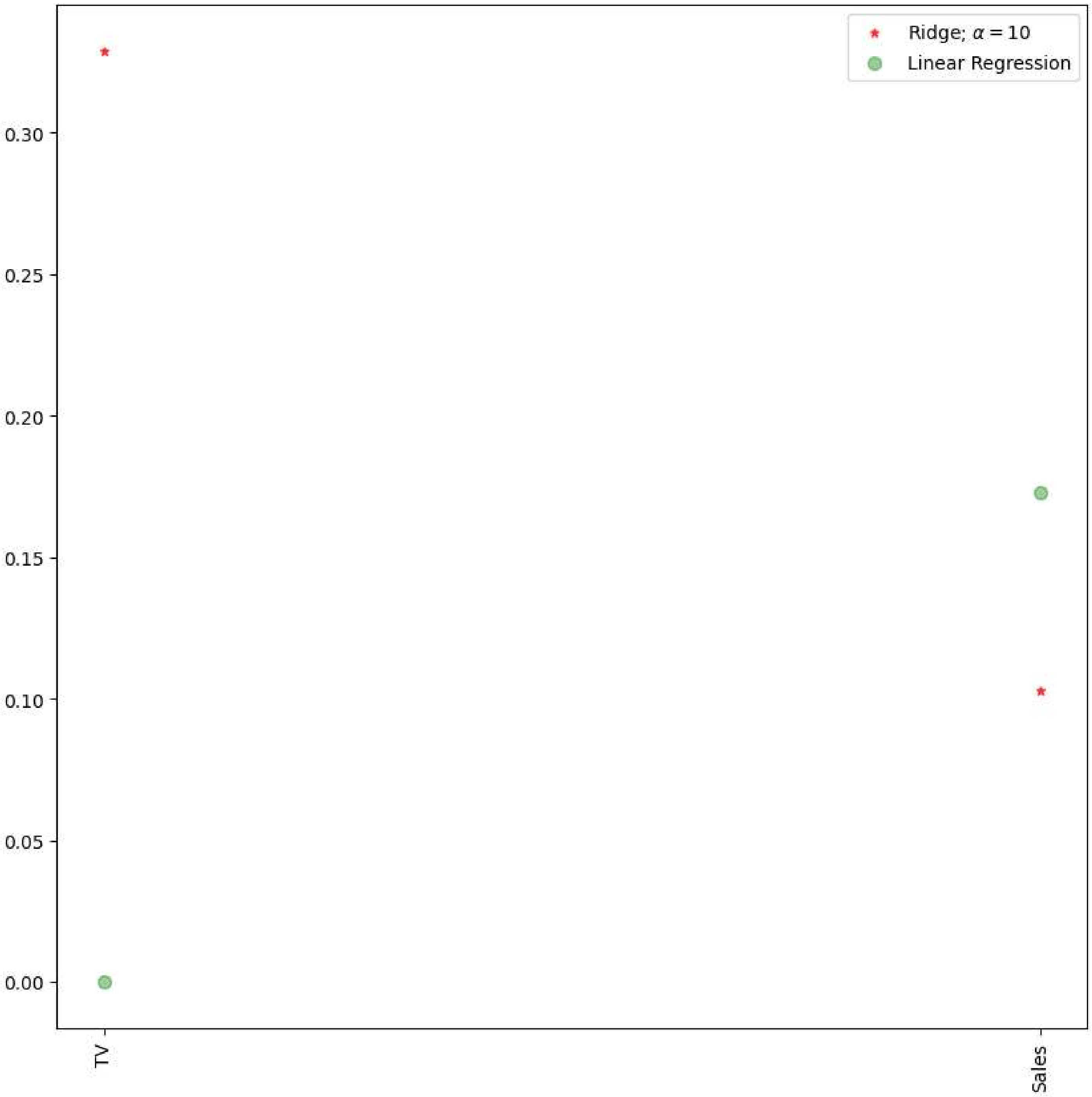
In [58]:

```
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 [64]:

```
lt.figure(figsize = (10, 10))
lt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5,color='red',label=r'Ridge; $\alpha = 10$',zorder=7)
plt.plot(rr100.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color='blue',label=r'Ridge; $\alpha = 100$')
lt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color='green',label='Linear Regression')
lt.xticks(rotation = 90)
lt.legend()
lt.show()
```


★ Ridge; $\alpha = 10$
● Linear Regression
0.30
0.25
0.20
0.15
0.10
0.05
0.00
TV
Sales

```
In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:
```