In [31]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import preprocessing,svm
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import Lasso,Ridge
```

In [32]:
```
df=pd.read_csv(r"C:\Users\DELL\Downloads\Advertising.csv")
df
```

Out[32]:

	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 [33]:
```
df.head(18)
```

Out[33]:

	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
5	8.7	48.9	75.0	7.2
6	57.5	32.8	23.5	11.8
7	120.2	19.6	11.6	13.2
8	8.6	2.1	1.0	4.8
9	199.8	2.6	21.2	15.6
10	66.1	5.8	24.2	12.6
11	214.7	24.0	4.0	17.4
12	23.8	35.1	65.9	9.2
13	97.5	7.6	7.2	13.7
14	204.1	32.9	46.0	19.0
15	195.4	47.7	52.9	22.4
16	67.8	36.6	114.0	12.5
17	281.4	39.6	55.8	24.4

In [34]:
```
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 [35]:
```
df.describe()
```

Out[35]:

	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 [36]:
```
df=df[['Sales','TV','Radio','Newspaper']]
df.columns=['sales','tv','radio','newspaper']
```

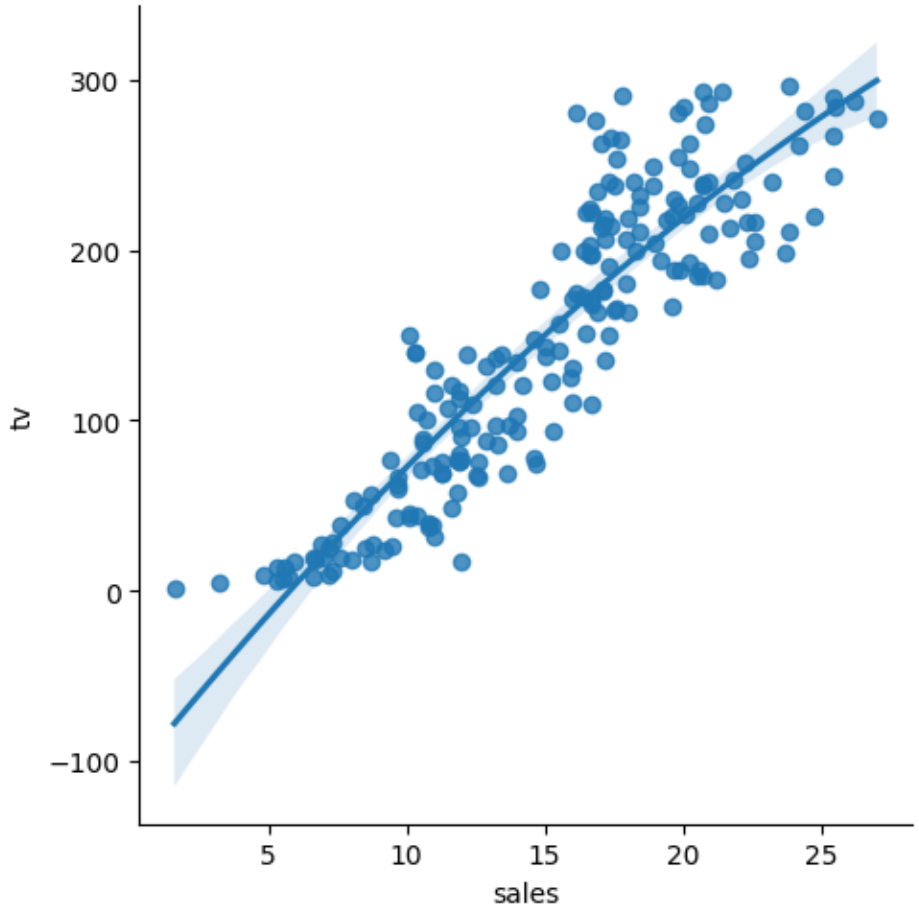
In [37]:
```
df.head(23)
```

Out[37]:

	sales	tv	radio	newspaper
0	22.1	230.1	37.8	69.2
1	10.4	44.5	39.3	45.1
2	12.0	17.2	45.9	69.3
3	16.5	151.5	41.3	58.5
4	17.9	180.8	10.8	58.4
5	7.2	8.7	48.9	75.0
6	11.8	57.5	32.8	23.5
7	13.2	120.2	19.6	11.6
8	4.8	8.6	2.1	1.0
9	15.6	199.8	2.6	21.2
10	12.6	66.1	5.8	24.2
11	17.4	214.7	24.0	4.0
12	9.2	23.8	35.1	65.9
13	13.7	97.5	7.6	7.2
14	19.0	204.1	32.9	46.0
15	22.4	195.4	47.7	52.9
16	12.5	67.8	36.6	114.0
17	24.4	281.4	39.6	55.8
18	11.3	69.2	20.5	18.3
19	14.6	147.3	23.9	19.1
20	18.0	218.4	27.7	53.4
21	17.5	237.4	5.1	23.5
22	5.6	13.2	15.9	49.6

In [38]:
```
sns.lmplot(x='sales',y='tv',data=df,order=2)
```

Out[38]:
```
<seaborn.axisgrid.FacetGrid at 0x21309151ad0>
```



In [39]:
```
df.describe()
```

Out[39]:

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

In [40]:
```
df.fillna(method='ffill',inplace=True)
```

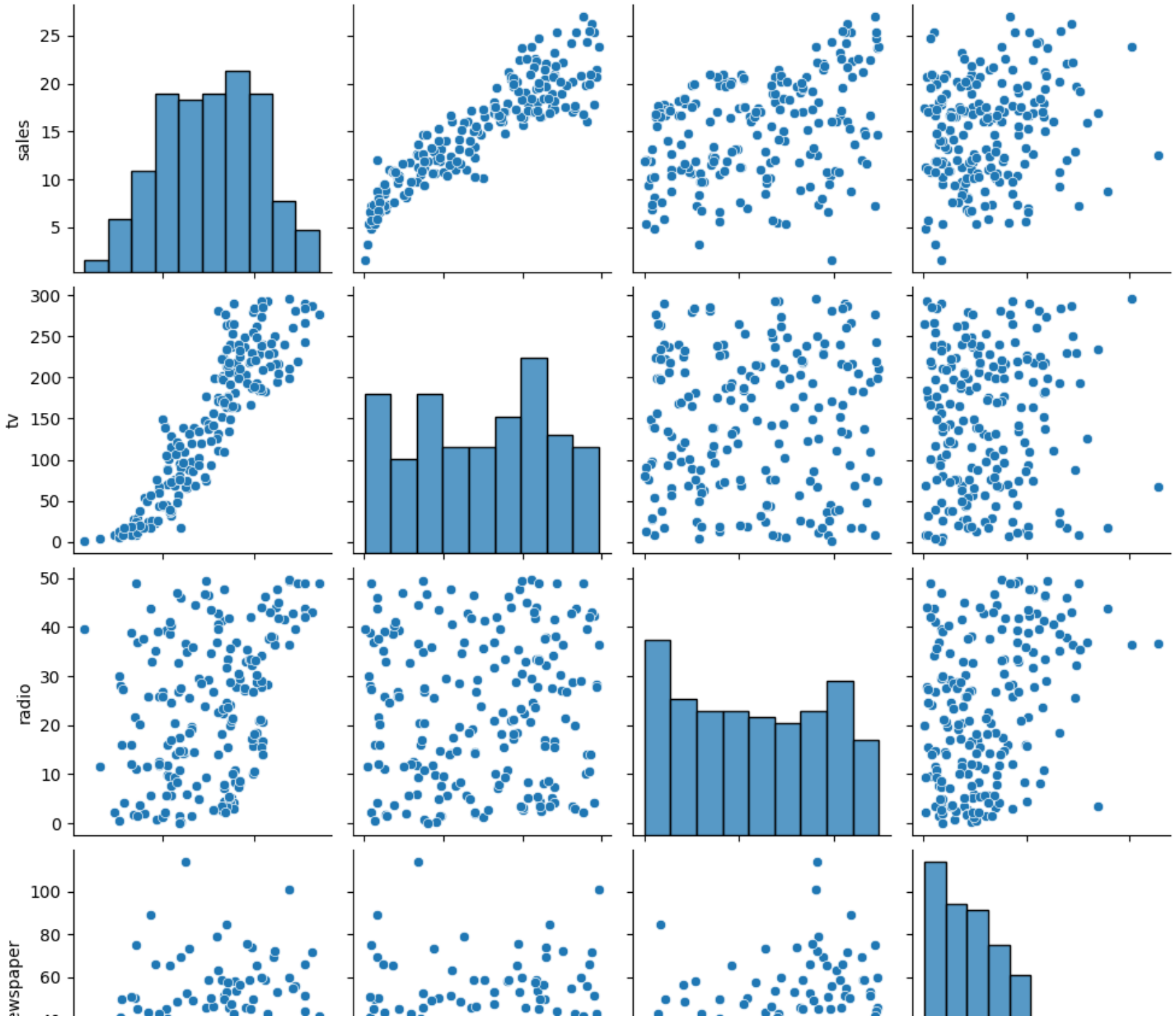
In [41]:
```
x=np.array(df['sales']).reshape(-1,1)
y=np.array(df['tv']).reshape(-1,1)
```

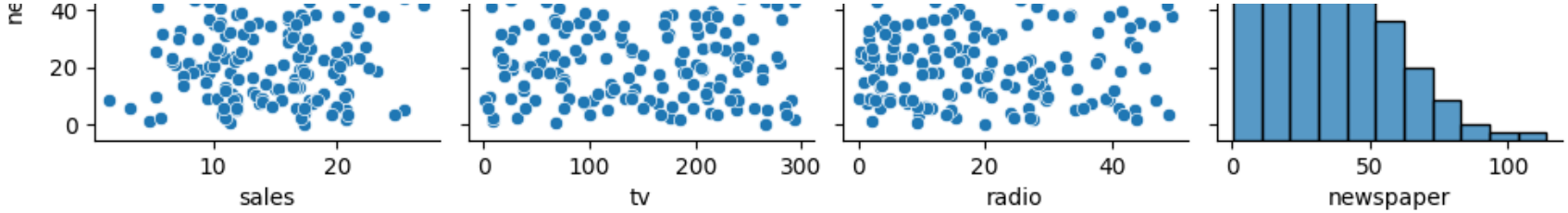
In [42]:
```
df.dropna(inplace=True)
```

In [43]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.5)
```

In [44]:
```
sns.pairplot(df)
```

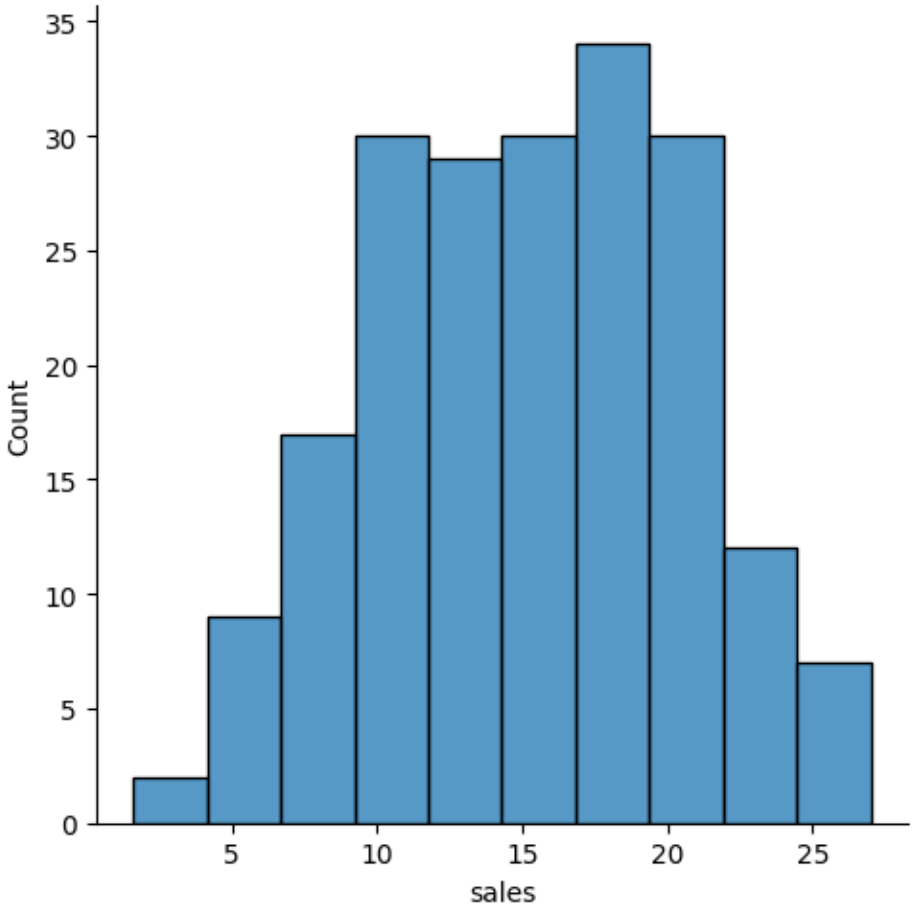
Out[44]:
```
<seaborn.axisgrid.PairGrid at 0x21305c59f10>
```


sales
tv
radio
ewspaper



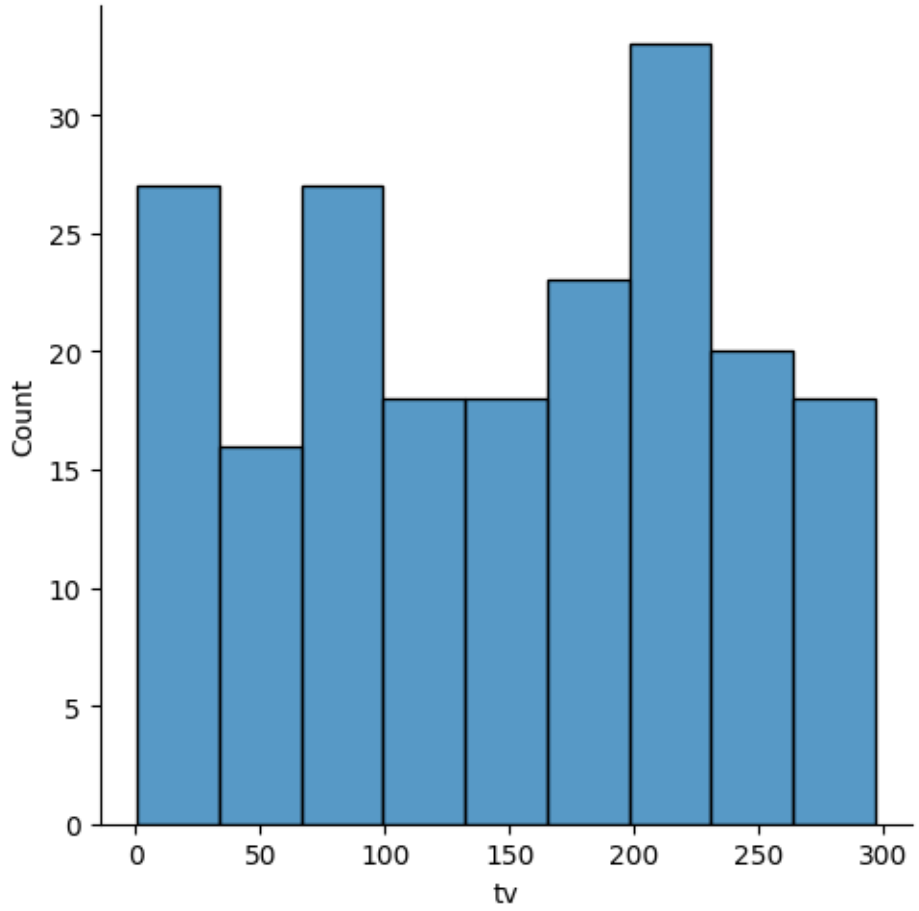
In [45]:
```
sns.displot(df['sales'])
```

Out[45]: <seaborn.axisgrid.FacetGrid at 0x21305c63650>



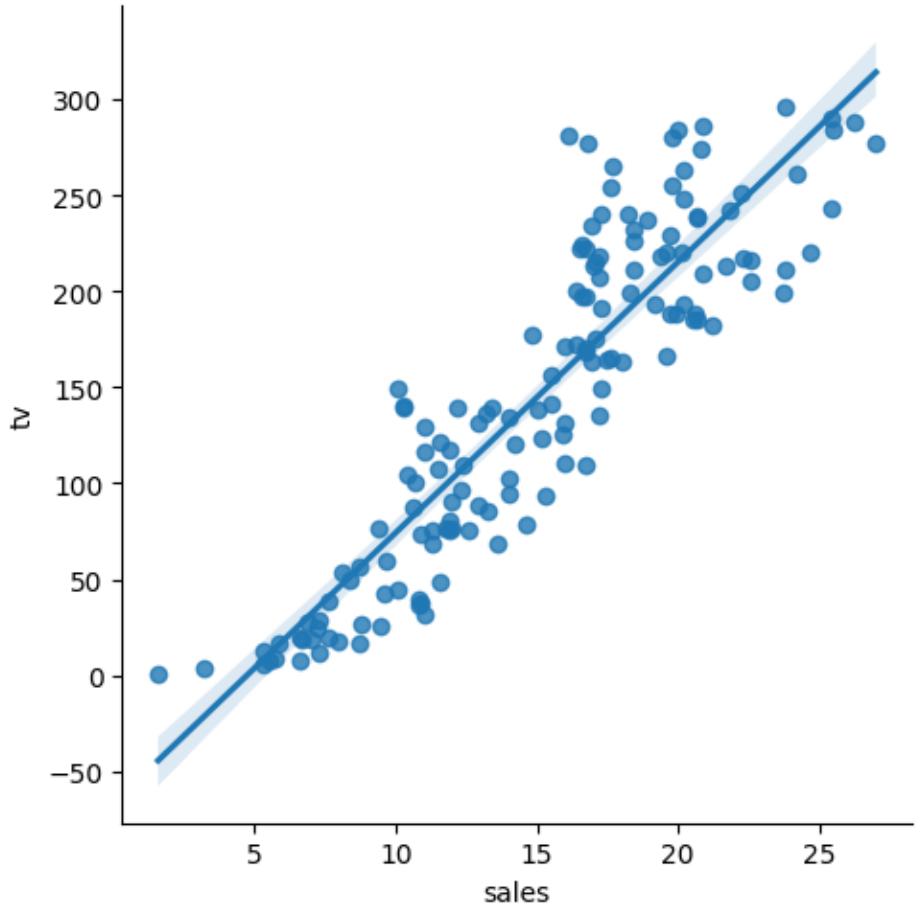
In [46]:
```
sns.displot(df['tv'])
```

Out[46]: <seaborn.axisgrid.FacetGrid at 0x2130c566390>



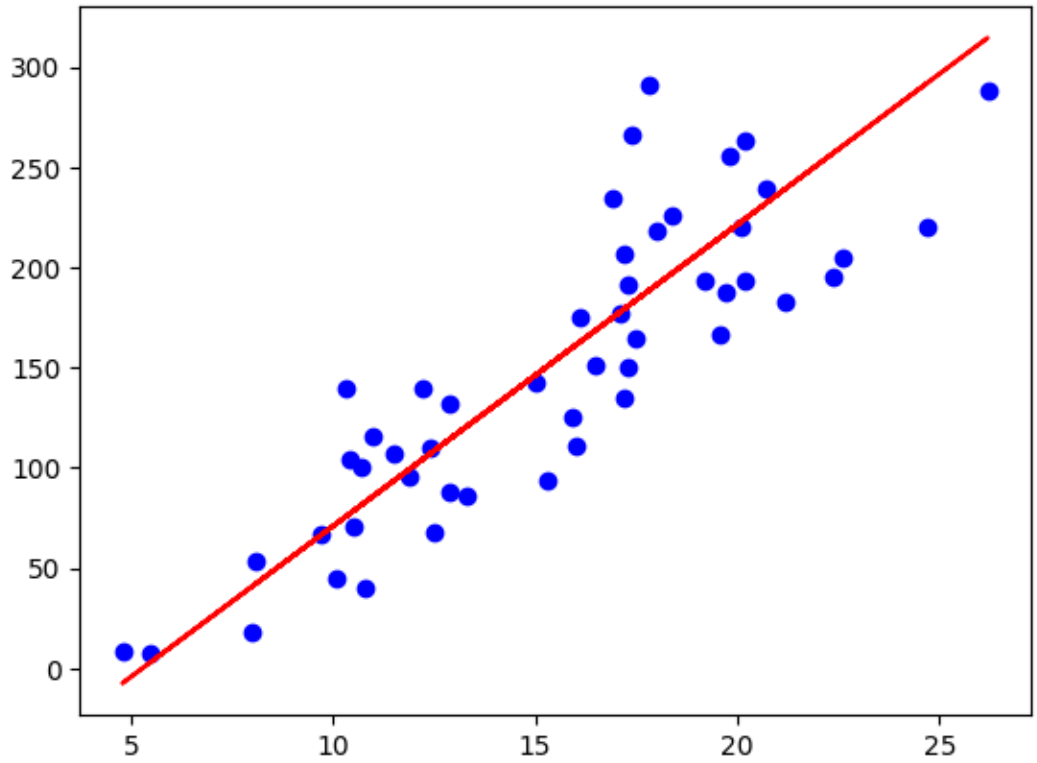
In [47]:
```
df500=df[:][50:500]
sns.lmplot(x='sales',y='tv',data=df500,order=1)
```

Out[47]: <seaborn.axisgrid.FacetGrid at 0x21305c7fe90>



In [48]:
```
df500.fillna(method='ffill',inplace=True)
x=np.array(df['sales']).reshape(-1,1)
y=np.array(df['tv']).reshape(-1,1)
df.dropna(inplace=True)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)
regr=LinearRegression()
regr.fit(x_train,y_train)
print("Regression:",regr.score(x_test,y_test))
y_pred=regr.predict(x_test)
plt.scatter(x_test,y_test,color='b')
plt.plot(x_test,y_pred,color='r')
plt.show()
```

```
Regression: 0.7276128589662572
```



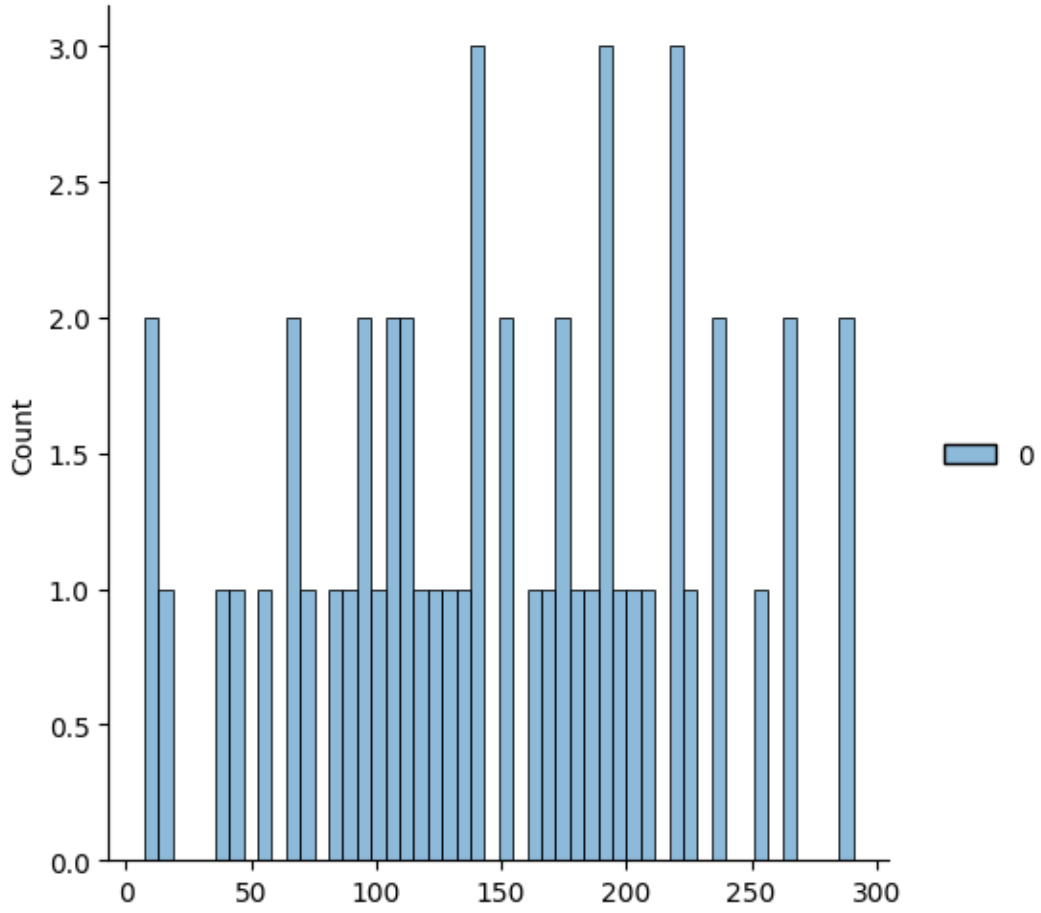
In [49]:
```
df.shape
```

Out[49]:
```
(200, 4)
```

In [50]:
```
df.isnull().sum()
```

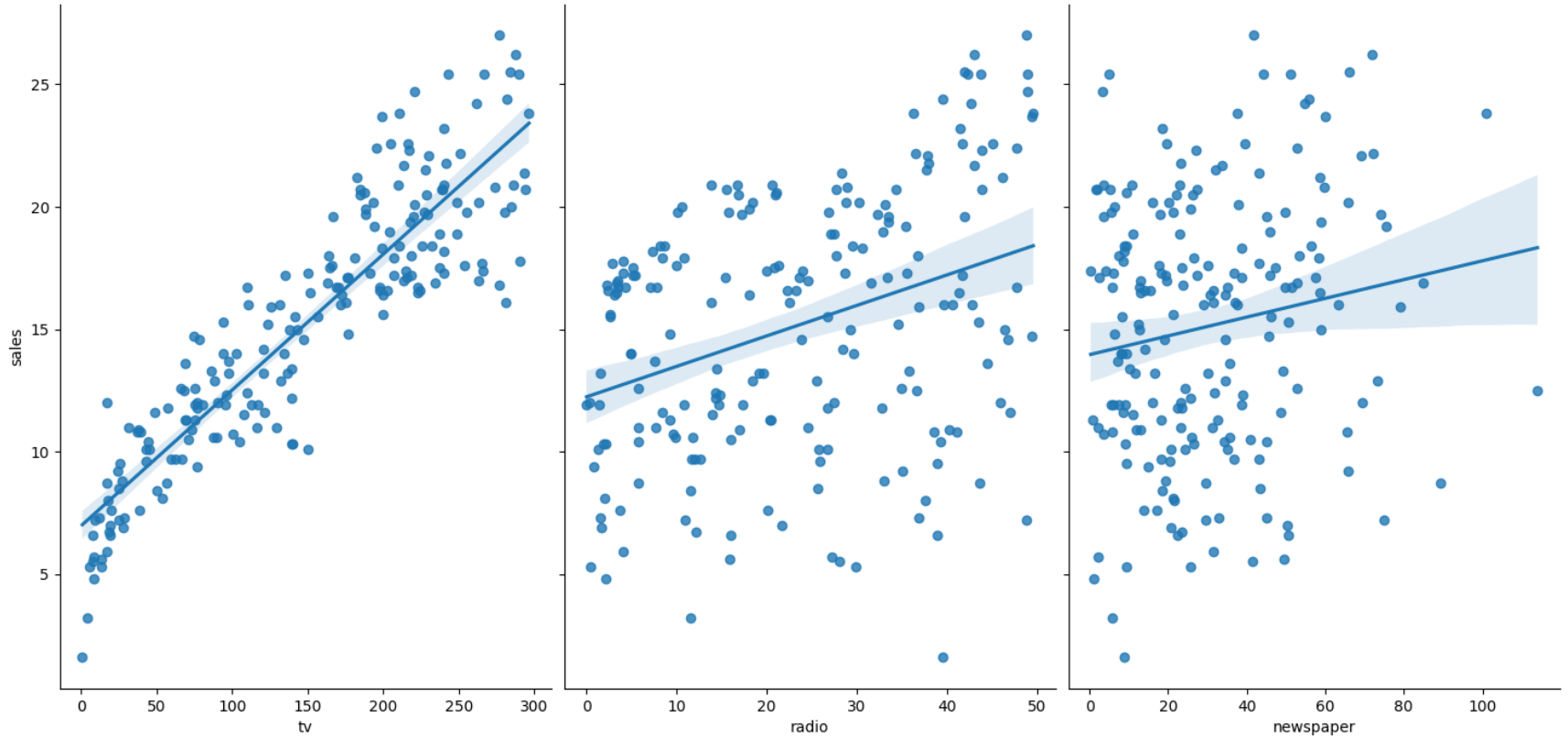
Out[50]:
```
sales        0
tv           0
radio        0
newspaper    0
dtype: int64
```

In [51]:
```
sns.displot((y_test),bins=50);
```



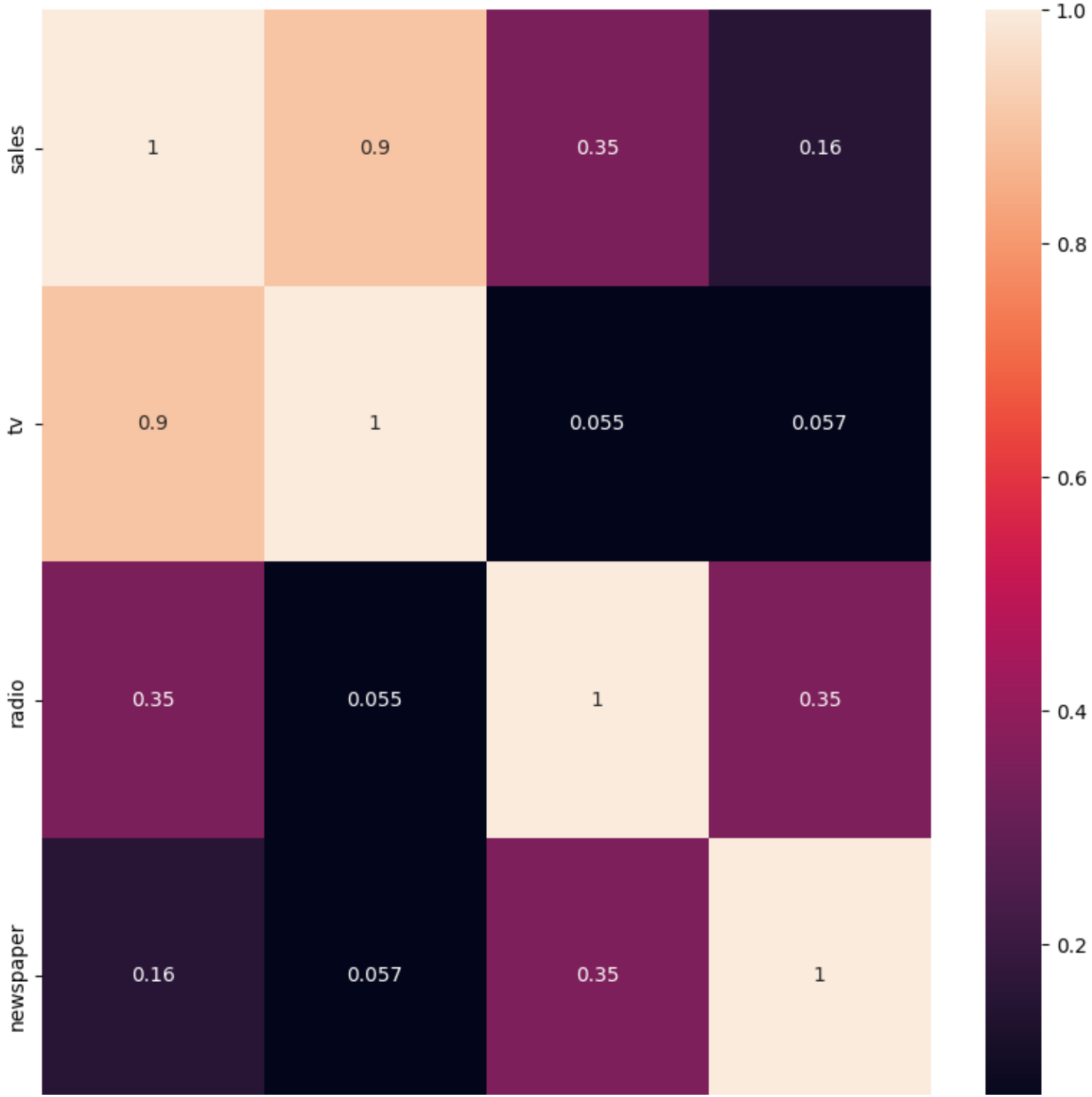
In [52]:
```
sns.pairplot(df,x_vars=['tv','radio','newspaper'],y_vars='sales',height=7,aspect=0.7,kind='reg')
```

Out[52]: <seaborn.axisgrid.PairGrid at 0x2130c511f10>



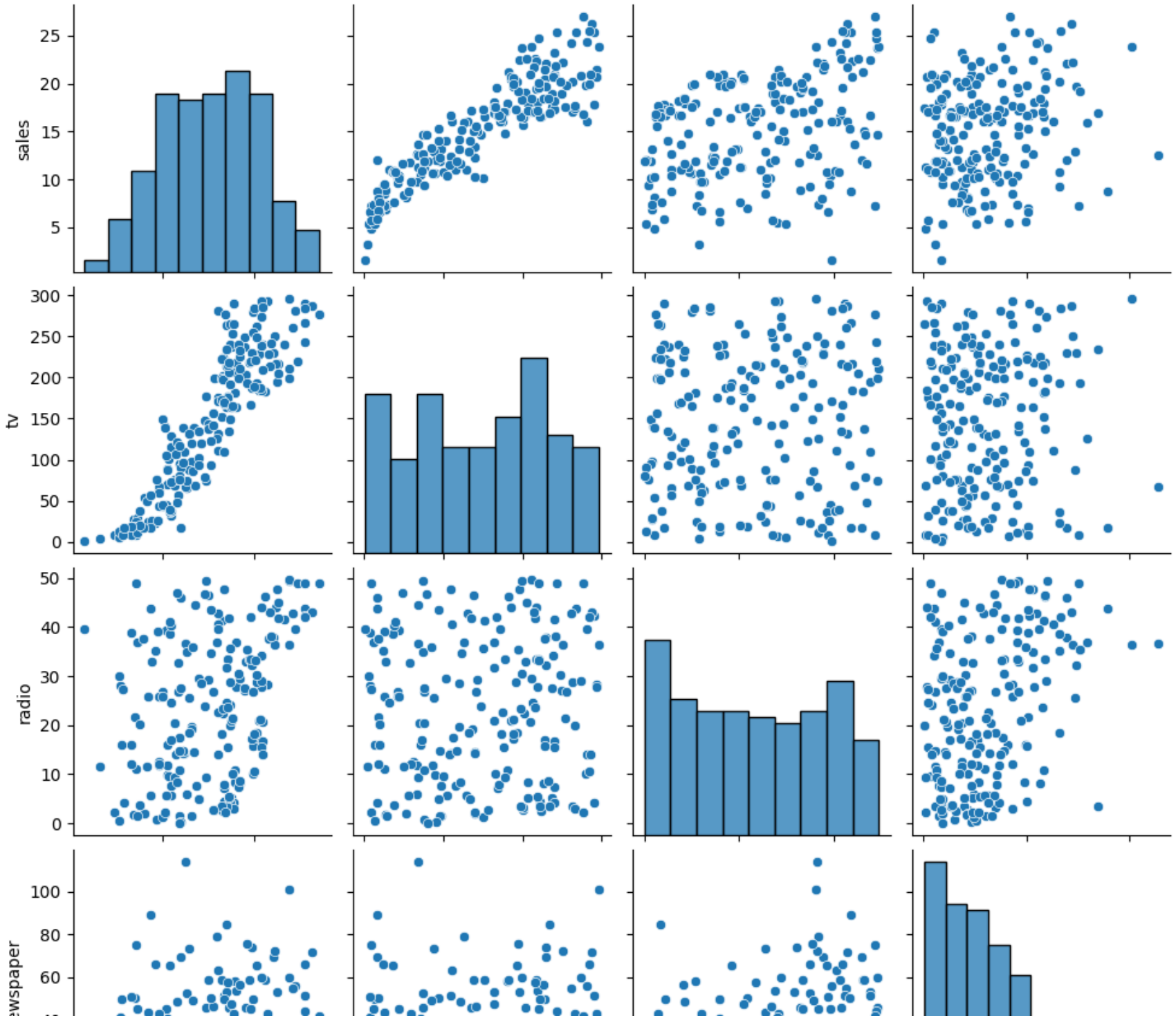
In [53]:
```
plt.figure(figsize=(10,10))
sns.heatmap(df.corr(),annot = True)
```

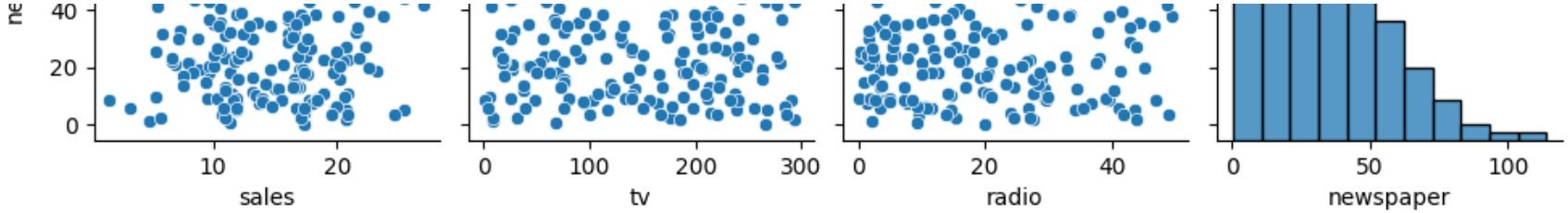
Out[53]:
```
<Axes: >
```


sales
tv
radio
newspaper
1
0.9
0.35
0.16
0.9
1
0.055
0.057
0.35
0.055
1
0.35
0.16
0.057
0.35
1
1.0
0.8
0.6
0.4
0.2

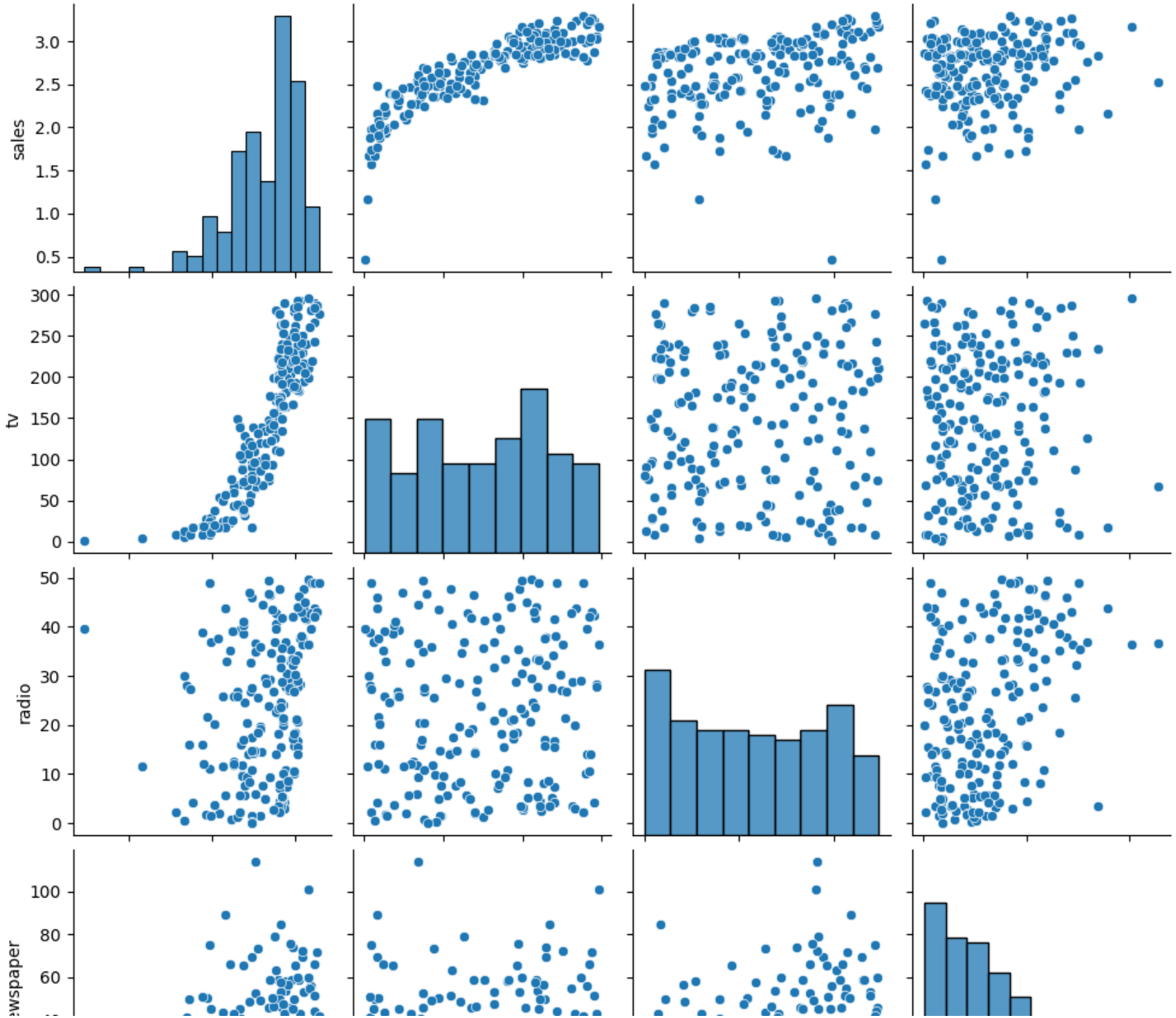

sales
tv
radio
newspaper

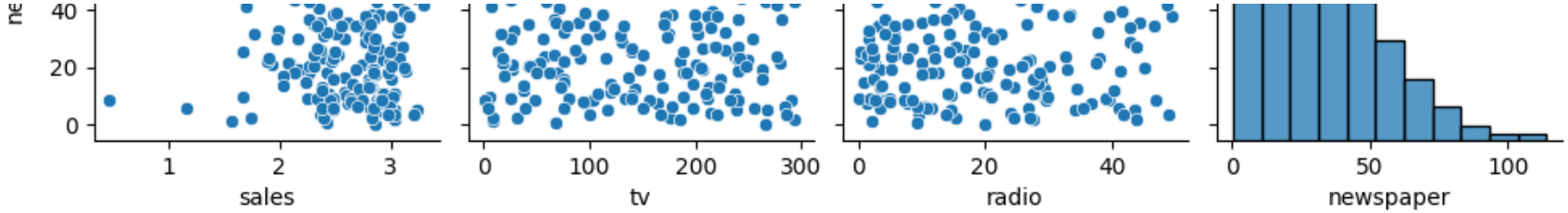
In [54]:
```
sns.pairplot(df)
df.sales = np.log(df.sales)
```


sales
tv
radio
ewspaper
25
20
15
10
5
300
250
200
150
100
50
0
50
40
30
20
10
0
100
80
60


40
20
0
10
20
sales
0
100
200
300
tv
0
20
40
radio
0
50
100
newspaper

In [55]:
```
sns.pairplot(df)
df.sales=np.log(df.sales)
```


sales
tv
radio
ewspaper
3.0
2.5
2.0
1.5
1.0
0.5
300
250
200
150
100
50
0
50
40
30
20
10
0
100
80
60



In [56]:
```
print(regr.score(x_test,y_test))
```

```
0.7276128589662572
```

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)
```

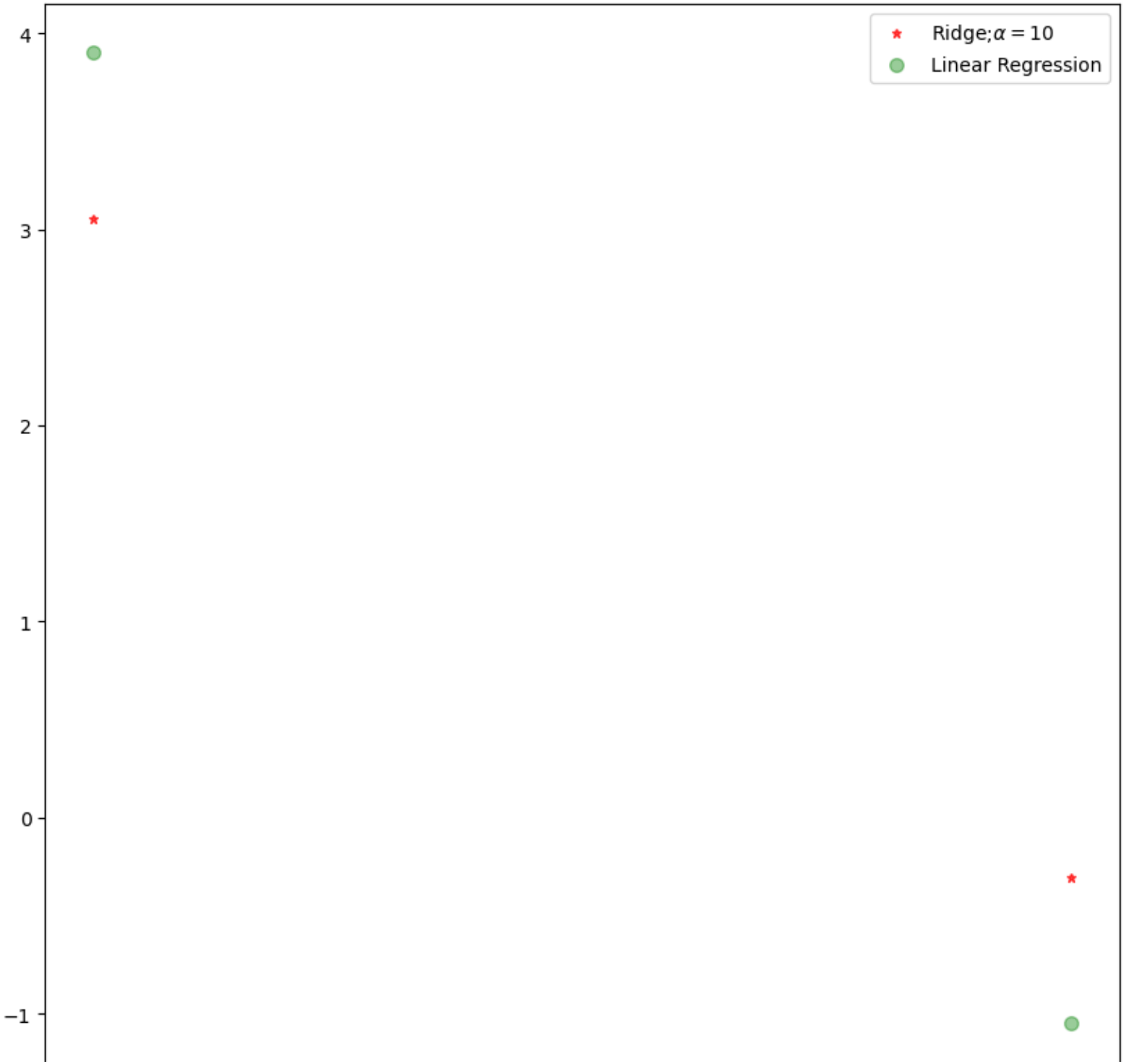
In [58]:
```
#model
lr=LinearRegression()
#fitmodel
lr.fit(X_train,y_train)
#actual
actual=y_test
train_score_lr=lr.score(X_train,y_train)
test_score_lr=lr.score(X_test,y_test)
print("\nLinear Regression Model:\n")
print("The train score for lr model is {}".format(train_score_lr))
print("The test score for lr model is {}".format(test_score_lr))
```

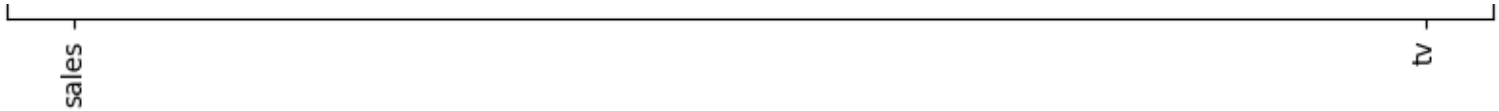
```
Linear Regression Model:

The train score for lr model is 0.02115661367780064
The test score for lr model is 0.02728680678747719
```

In [59]:
```
#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))
```

...

In [62]:
```
t.figure(figsize=(10,10))
.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5,color='red',label=r'Ridge;$\alpha=10$
.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color='green',label='Linear Regression')
.xticks(rotation=90)
.legend()
.show()
```


Ridge;$\alpha = 10$
Linear Regression
4
3
2
1
0
−1

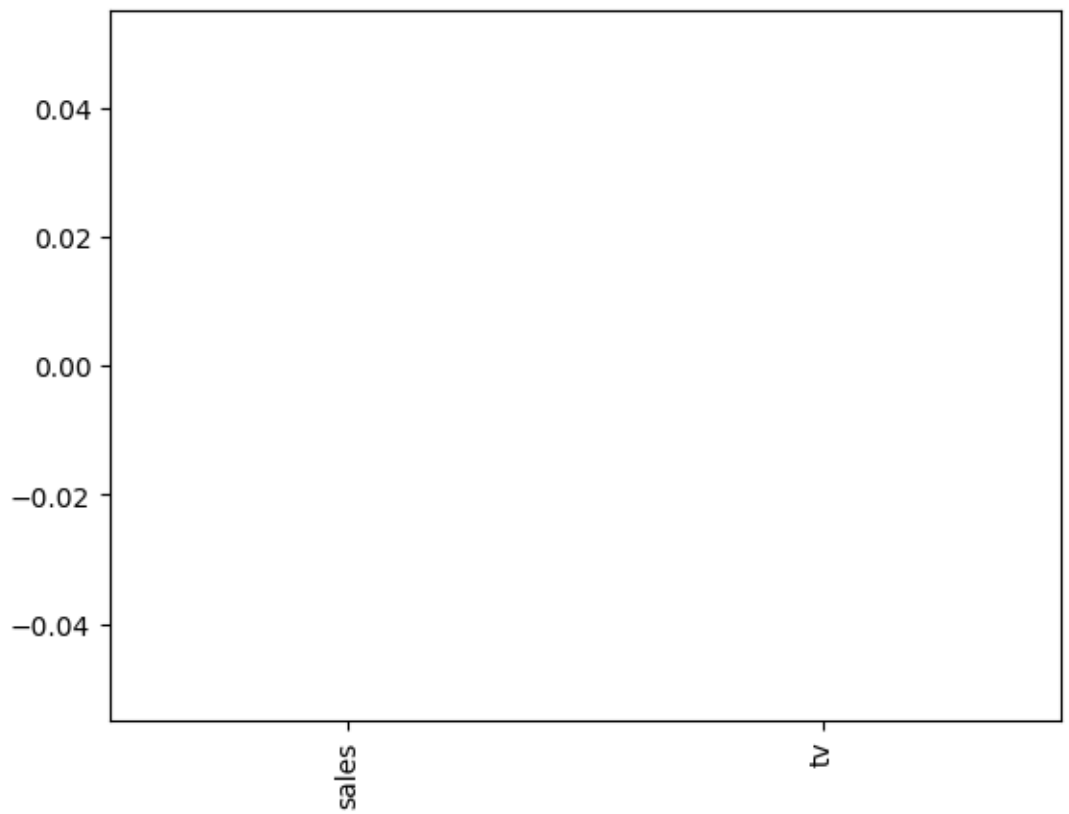


In [63]:
```
print("\nLasso model: \n")
lasso = Lasso(alpha=10)
lasso.fit(X_train,y_train)
train_score_ls =lasso.score(X_train,y_train)
test_score_ls =lasso.score(X_test,y_test)
print("The train score for ls model is {}".format(train_score_ls))
print("The test score for ls model is {}".format(test_score_ls))
```

```
Lasso model:

The train score for ls model is 0.0
The test score for ls model is -0.0003547334659412815
```

In [64]:
```
pd.Series(lasso.coef_,features).sort_values(ascending = True).plot(kind= "bar")
```

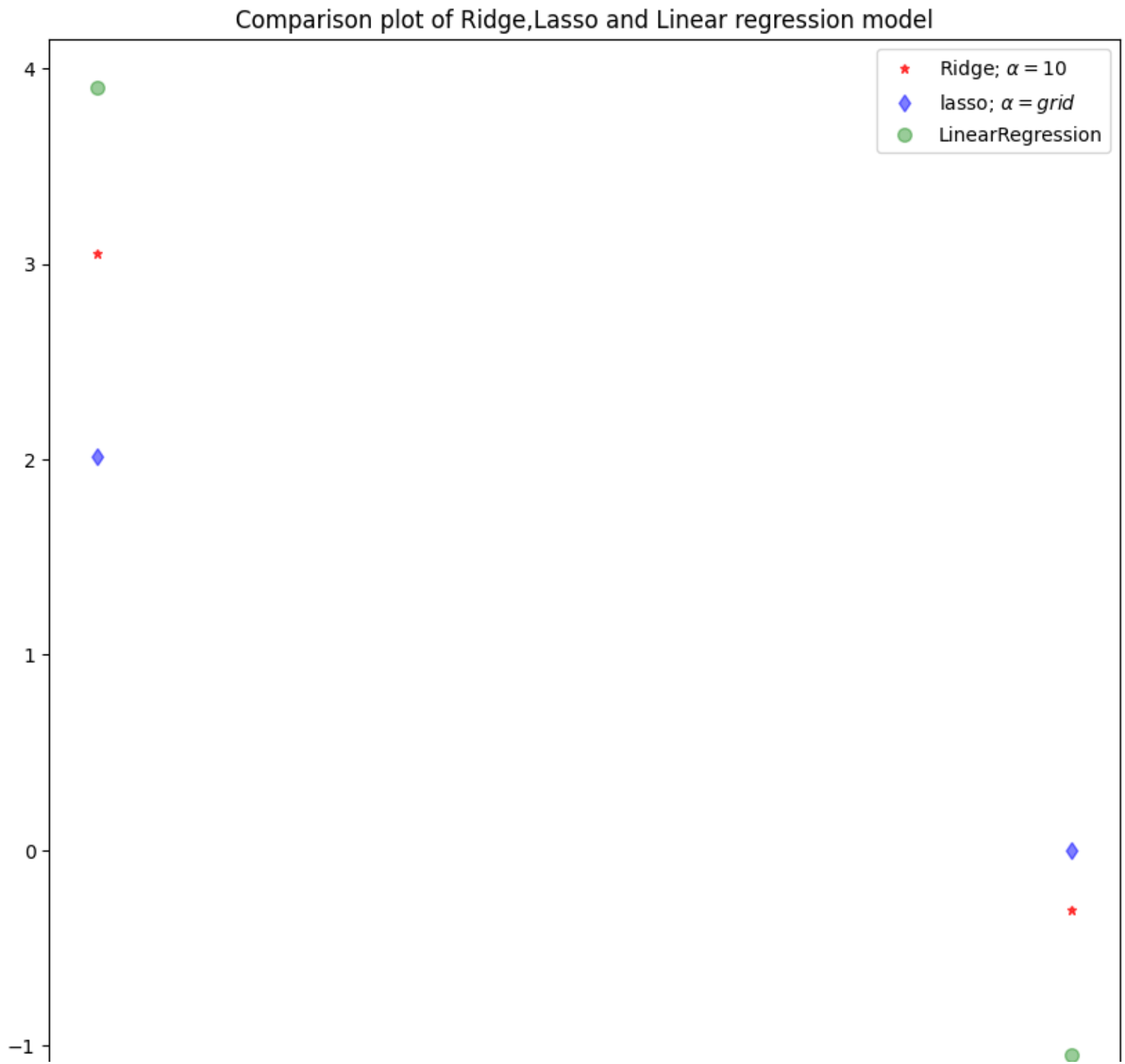
Out[64]: <Axes: >



In [65]:
```
#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))
```

```
0.018212978808147873
0.021362908672580283
```

In [67]:
```
t size
figure(figsize =(10,10))
 plot for ridge regression
plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5,color='red',label=r'Ridge; $\alpha=10$
 plot for lasso regression
plot(lasso_cv.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color='blue',label=r'lasso; $\alpha=grid$')
 plot for linear model
plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color='green',label='LinearRegression')
ate axis
xticks(rotation = 90)
legend()
title("Comparison plot of Ridge,Lasso and Linear regression model")
show()
```


Comparison plot of Ridge,Lasso and Linear regression model
Ridge; $\alpha = 10$
lasso; $\alpha = grid$
LinearRegression
4
3
2
1
0
−1

In [68]:
```
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 0.020702537937223098
The train score for ridge model is 0.023443477224617815
```

In [69]:
```
from sklearn.linear_model import ElasticNet
```

In [70]:
```
regr=ElasticNet()
regr.fit(X,y)
print(regr.coef_)
print(regr.intercept_)
```

```
[0.        0.0143007]
28.451189115205615
```

In [71]:
```
y_pred_elastic=regr.predict(X_train)
```

In [74]:
```
mean_squared_error=np.mean((y_pred_elastic-y_train)**2)
print("Mean Squared Error on test set",mean_squared_error)
```

```
Mean Squared Error on test set 447.09072538429047
```

In [ ]:
```

```

In [ ]: