In [94]:

```
import pandas as pd
import numpy as np
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 Ridge,RidgeCV,Lasso
```

In [95]:

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

Out[95]:

	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 [96]:
```
df.head()
```

Out[96]:

	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 [97]:
```
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 [98]:
```
df.tail(12)
```

Out[98]:

	TV	Radio	Newspaper	Sales
188	286.0	13.9	3.7	20.9
189	18.7	12.1	23.4	6.7
190	39.5	41.1	5.8	10.8
191	75.5	10.8	6.0	11.9
192	17.2	4.1	31.6	5.9
193	166.8	42.0	3.6	19.6
194	149.7	35.6	6.0	17.3
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 [99]:
```
df=df[['TV','Radio','Sales','Newspaper']]
df.columns=['tv','radio','sales','newspaper']
```

In [100]:
```
df.head(10)
```

Out[100]:

	tv	radio	sales	newspaper
0	230.1	37.8	22.1	69.2
1	44.5	39.3	10.4	45.1
2	17.2	45.9	12.0	69.3
3	151.5	41.3	16.5	58.5
4	180.8	10.8	17.9	58.4
5	8.7	48.9	7.2	75.0
6	57.5	32.8	11.8	23.5
7	120.2	19.6	13.2	11.6
8	8.6	2.1	4.8	1.0
9	199.8	2.6	15.6	21.2

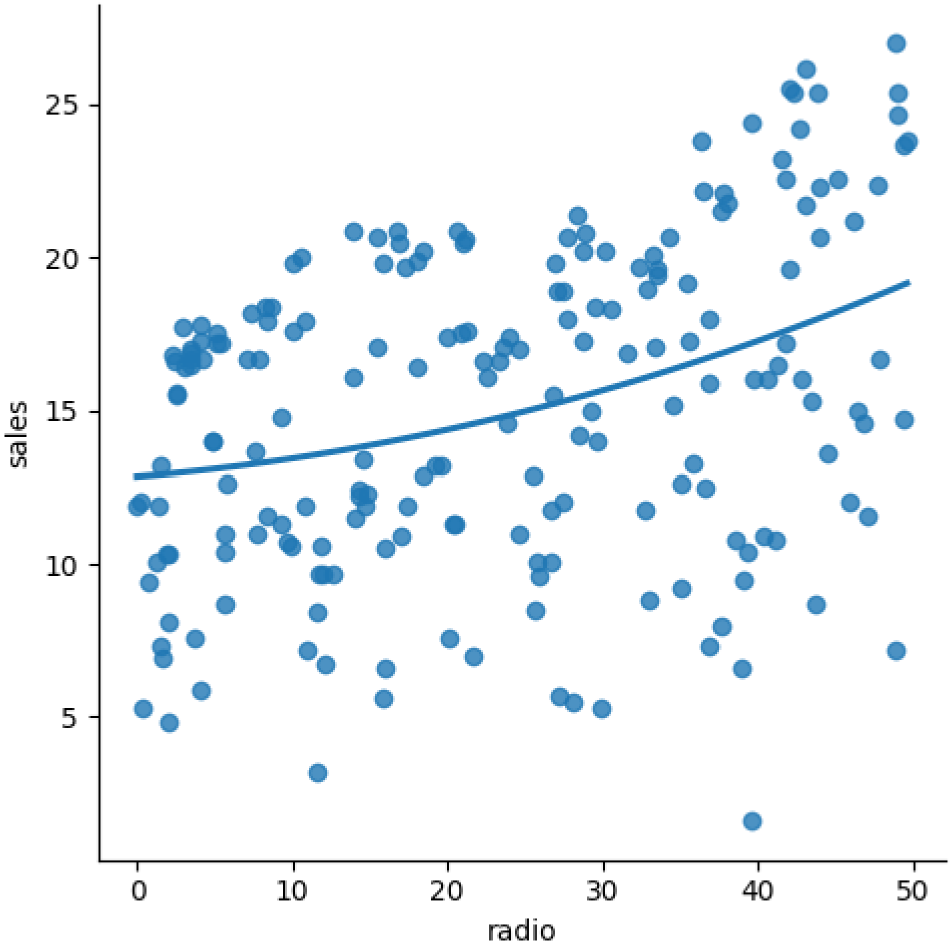
In [101]:
```
df.describe()
```

Out[101]:

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

```
In [102]: sns.lmplot(x="radio",y="sales",data=df,order=2,ci=None)

Out[102]: <seaborn.axisgrid.FacetGrid at 0x20e1f8c9b40>
```



```
In [103]: df.fillna(method='ffill',inplace=True)
```

In [104]:

```
x=np.array(df['radio']).reshape(-1,1)
y=np.array(df['sales']).reshape(-1,1)
df.dropna(inplace=True)
```

In [105]:

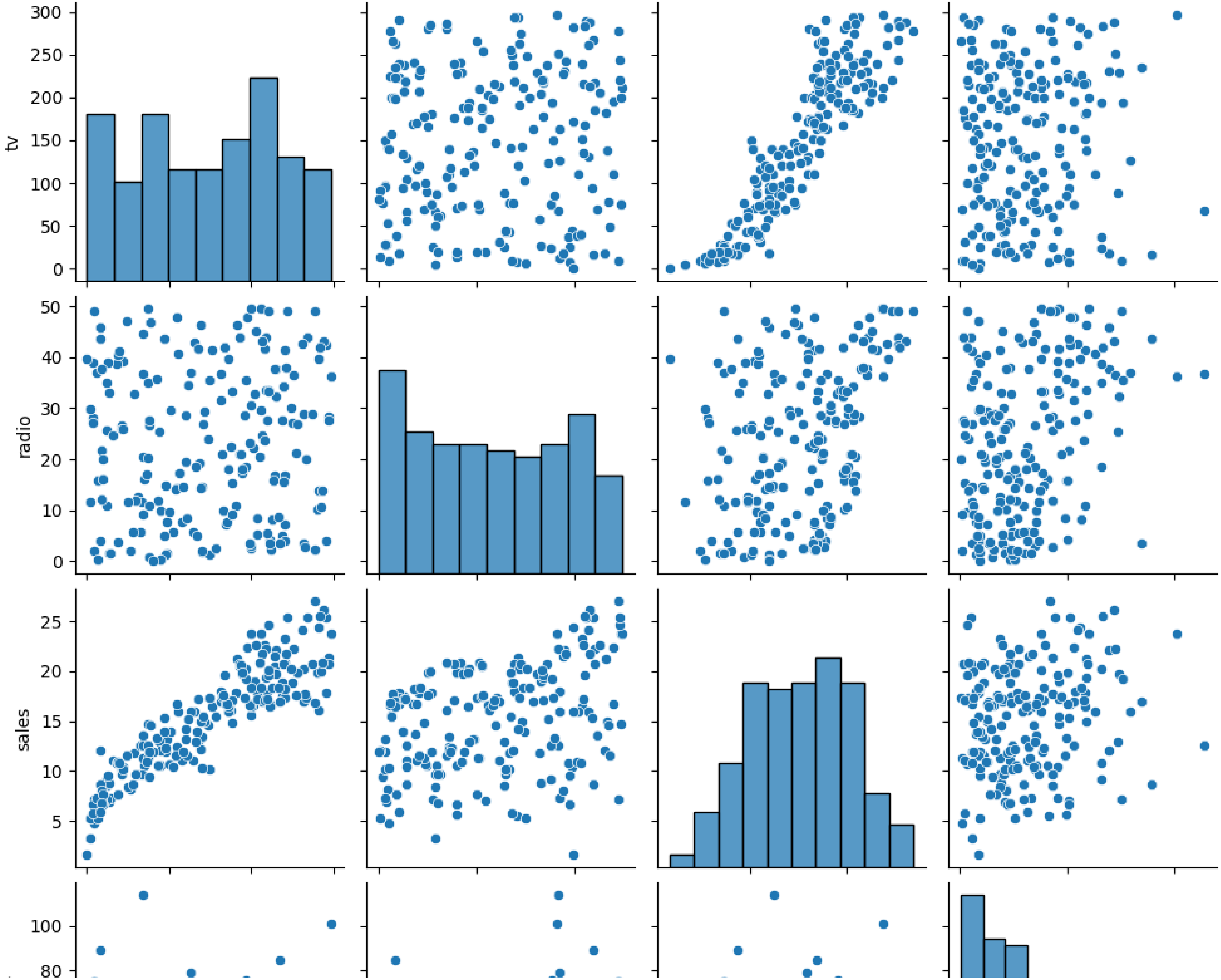
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.5)
```

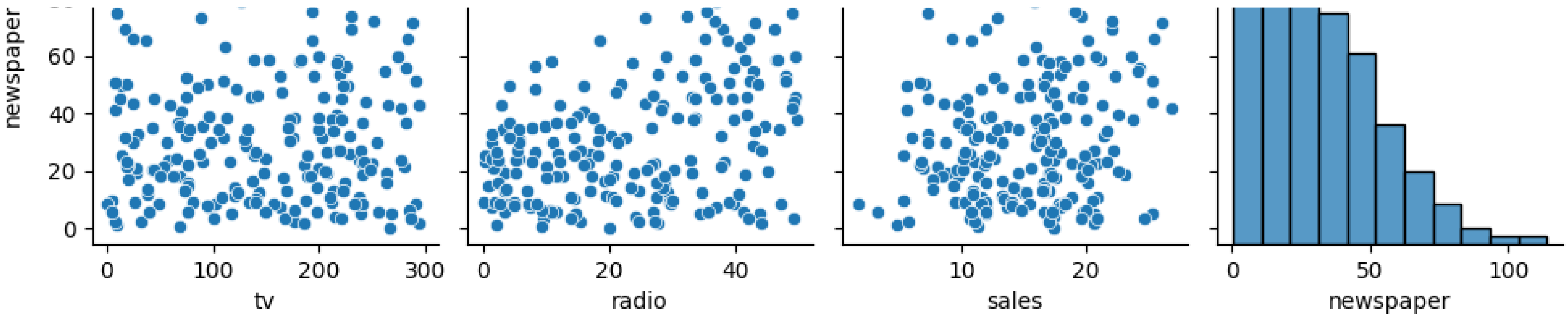
In [106]:

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

Out[106]:

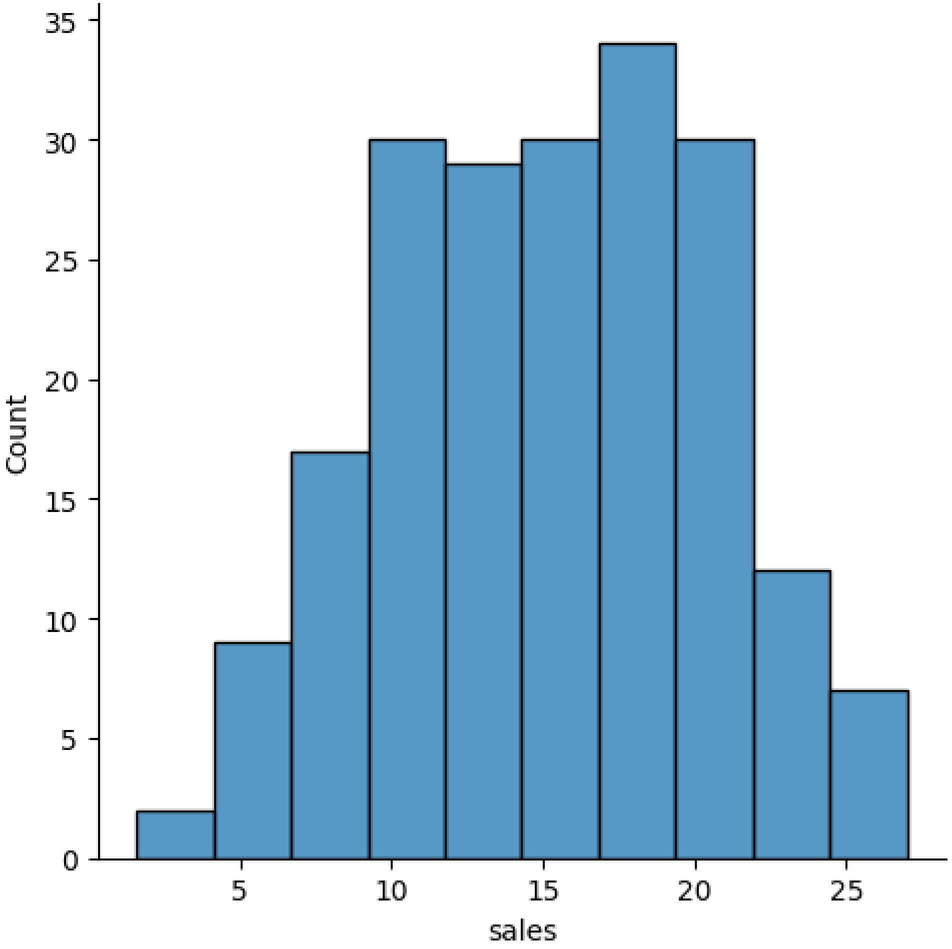
```
<seaborn.axisgrid.PairGrid at 0x20e1f917580>
```


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


newspaper
0
20
40
60
0
100
200
300
tv
0
20
40
radio
10
20
sales
0
50
100
newspaper

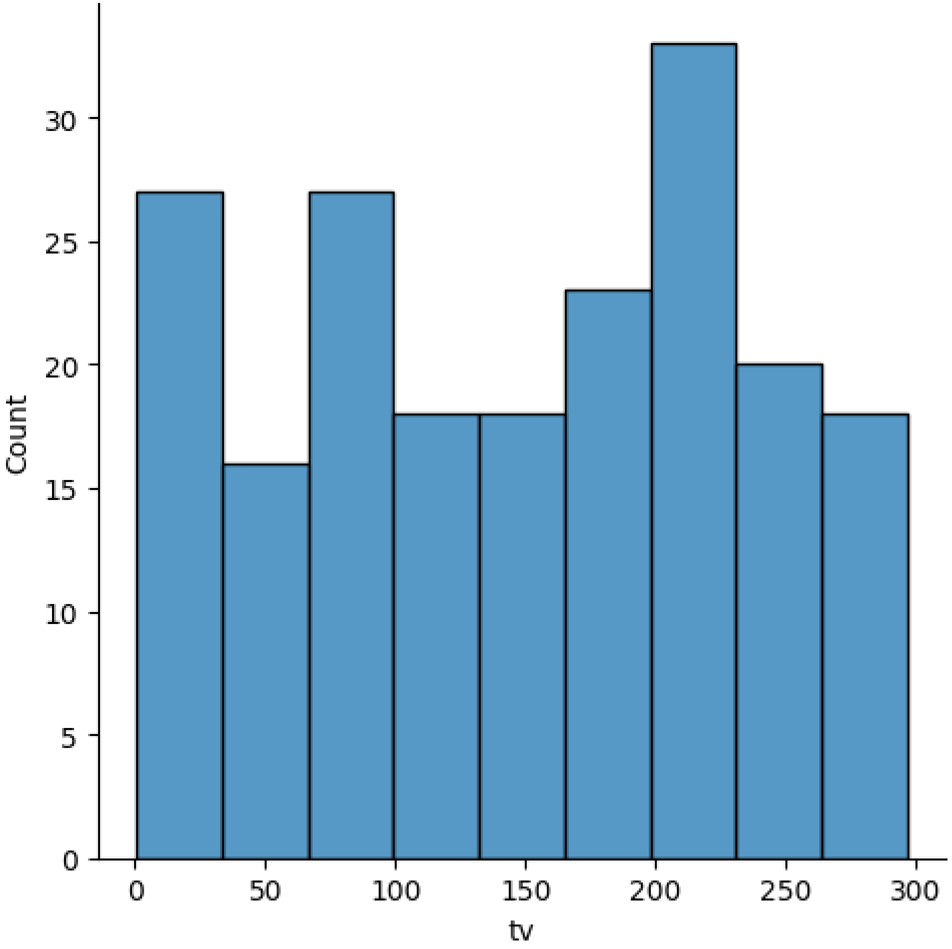
In [107]:
```
sns.displot(df['sales'])
```

Out[107]:
```
<seaborn.axisgrid.FacetGrid at 0x20e1f8c9990>
```



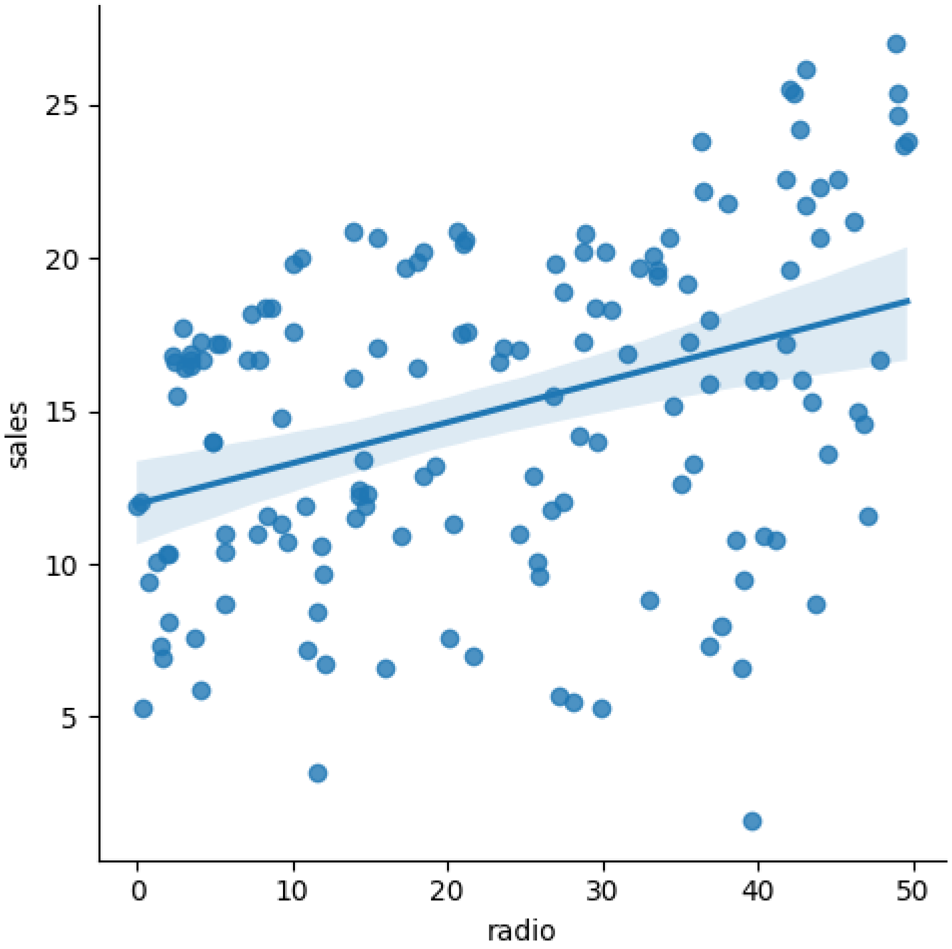
```
In [108]: sns.displot(df['tv'])
```

```
Out[108]: <seaborn.axisgrid.FacetGrid at 0x20e21a41a20>
```



```
In [109]: df500=df[:][50:500]
          sns.lmplot(x="radio",y="sales",data=df500,order=1)
```

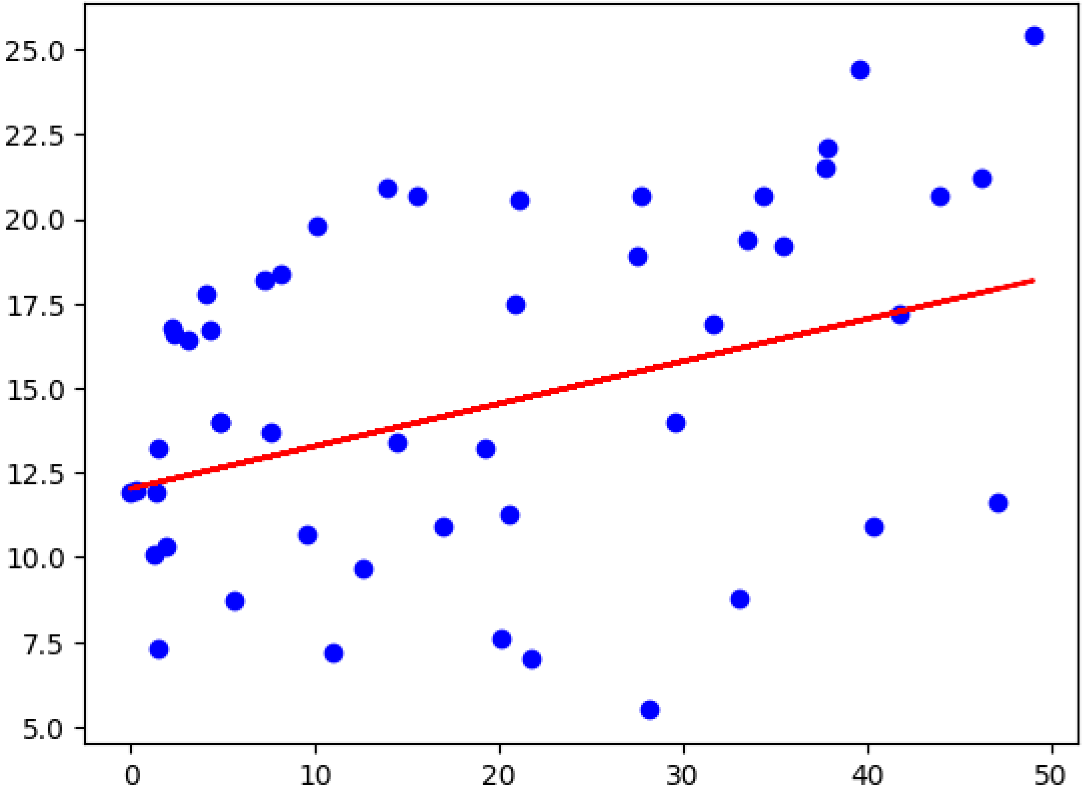
```
Out[109]: <seaborn.axisgrid.FacetGrid at 0x20e1f9177f0>
```



In [110]:

```
df500.fillna(method='ffill',inplace=True)
x=np.array(df['radio']).reshape(-1,1)
y=np.array(df['sales']).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: 0.14284532571622577
```



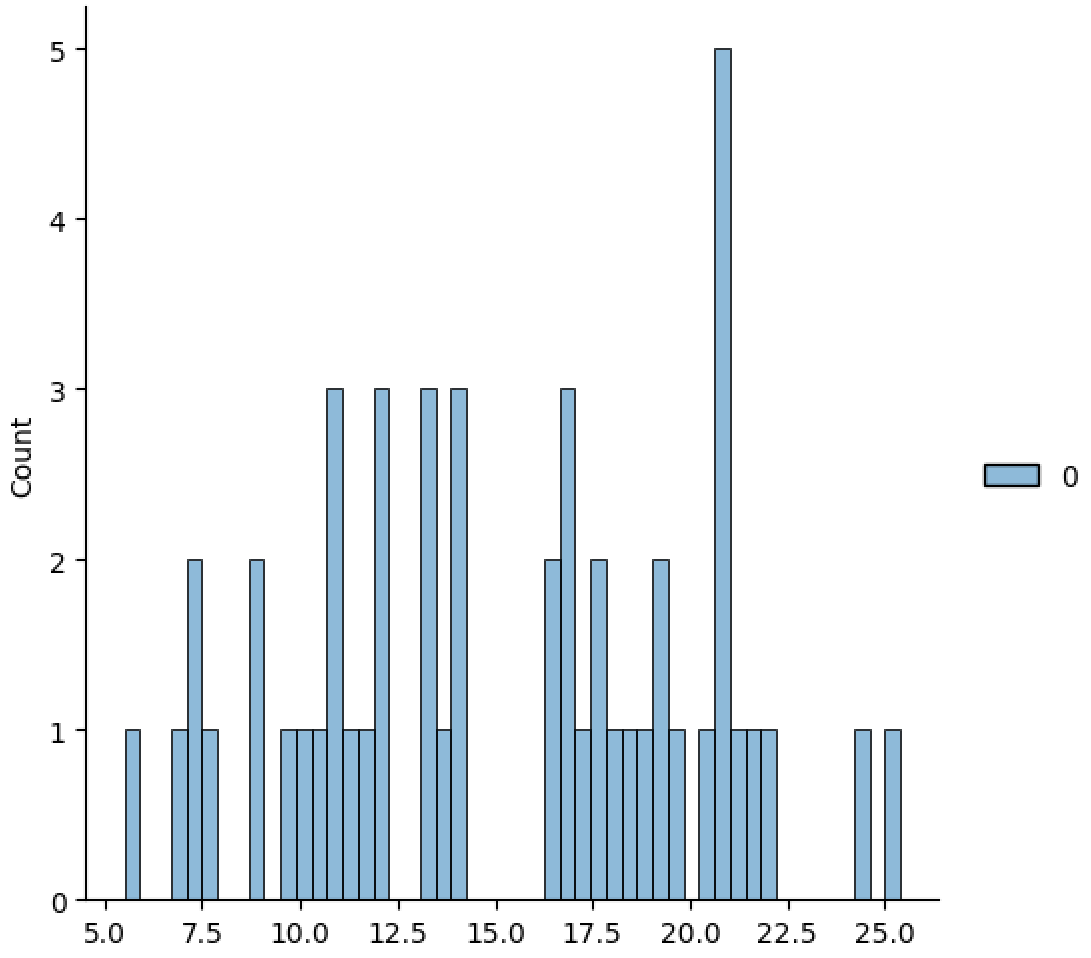
In [111]:
```
df.shape
```

Out[111]:
```
(200, 4)
```

In [112]:
```
df.isnull().sum()
```

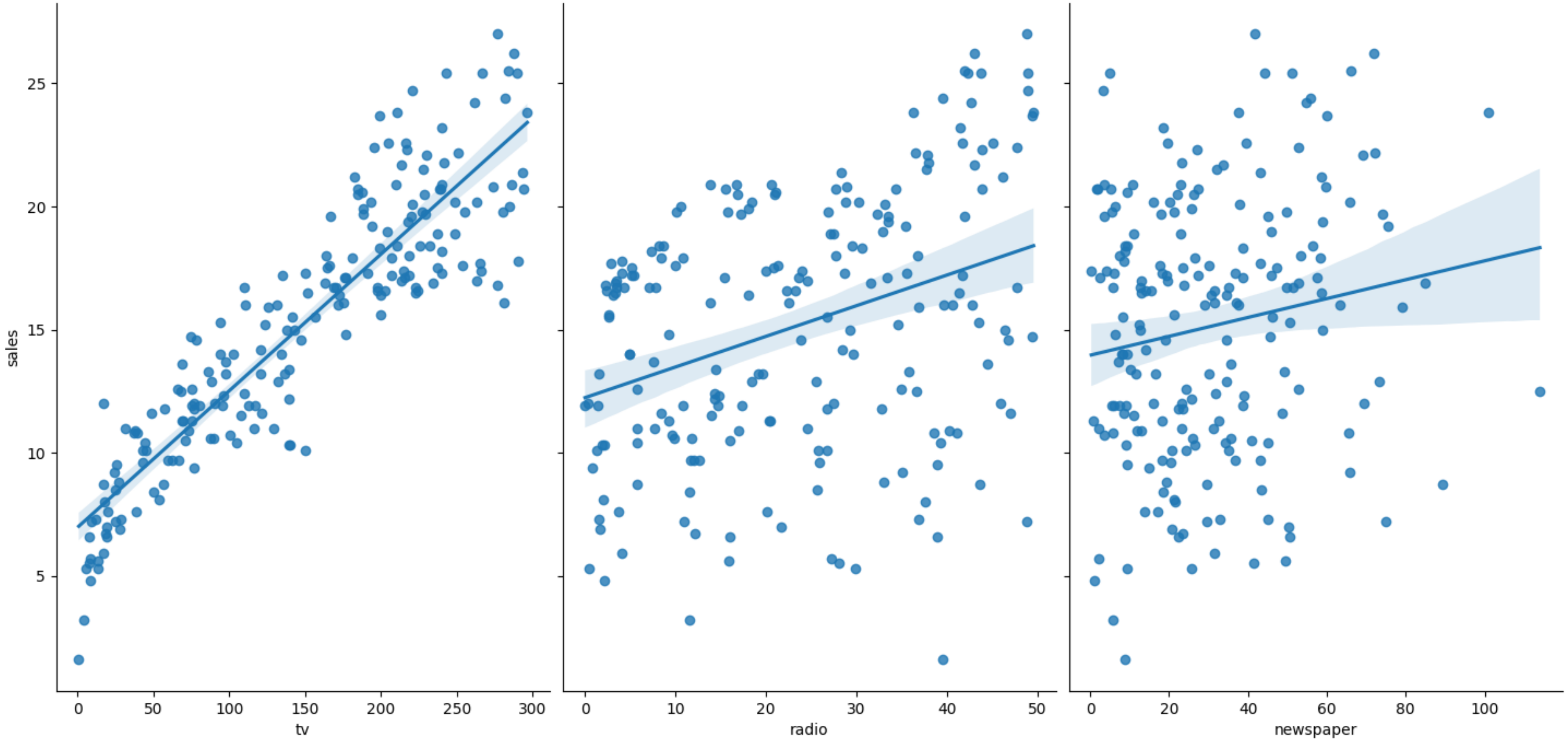
Out[112]:
```
tv           0
radio        0
sales        0
newspaper    0
dtype: int64
```

```
In [113]: sns.displot((y_test),bins=50);
```



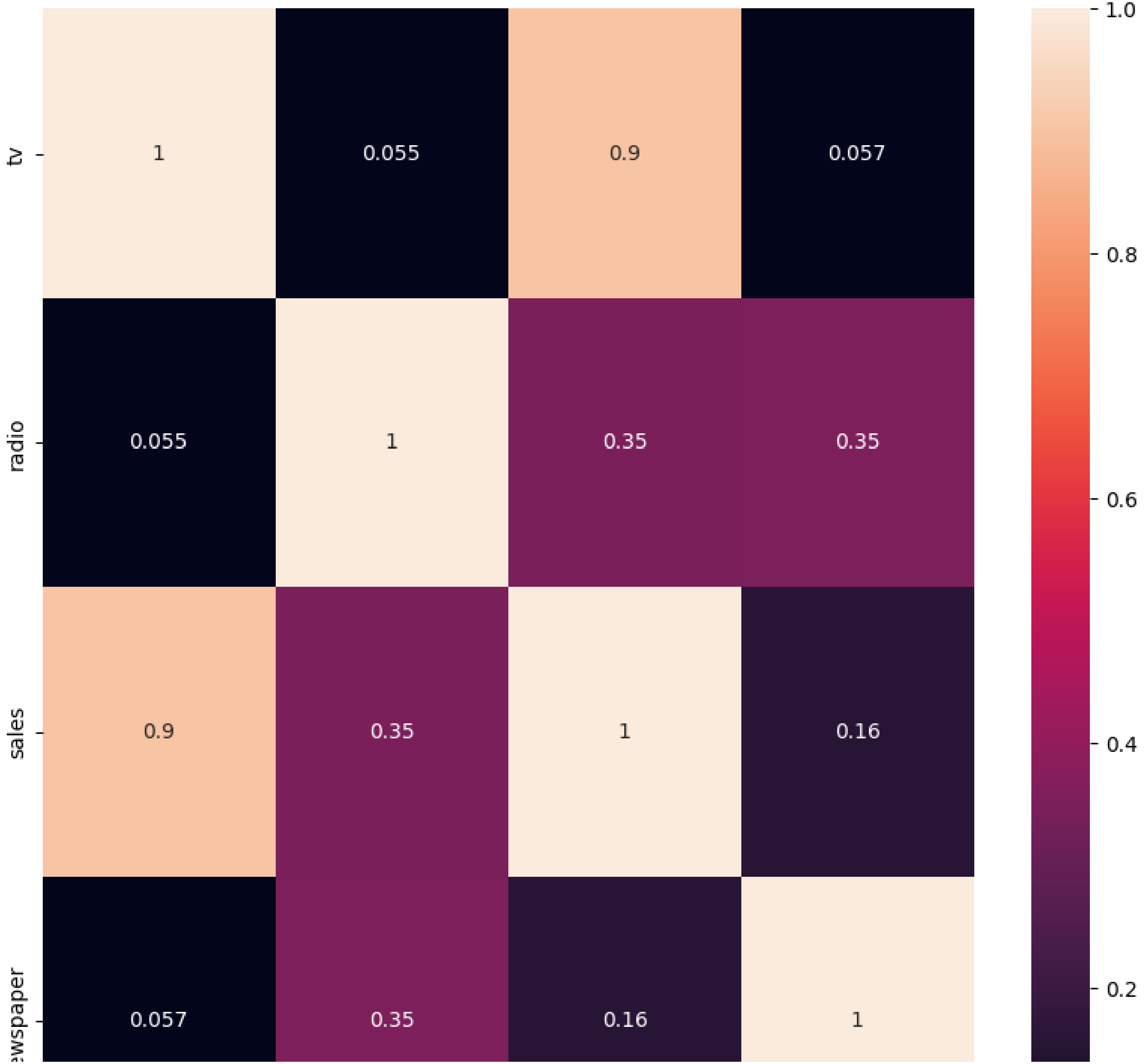
In [114]:
```
sns.pairplot(df,x_vars=['tv','radio','newspaper'],y_vars='sales',height=7,aspect=0.7,kind='reg')
```

Out[114]:
```
<seaborn.axisgrid.PairGrid at 0x20e1e7ea1d0>
```



In [115]:
```
plt.figure(figsize=(10,10))
sns.heatmap(df.corr(),annot=True)
```

Out[115]:
```
<Axes: >
```


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

In [116]:

```
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 [117]:

```
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 0.08820570968809427
The test score for lr model is 0.17667256011867194
```

In [118]:

```
ridgeReg = Ridge(alpha=10)
ridgeReg.fit(x_train,y_train)
#train and test scorefor 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))
```

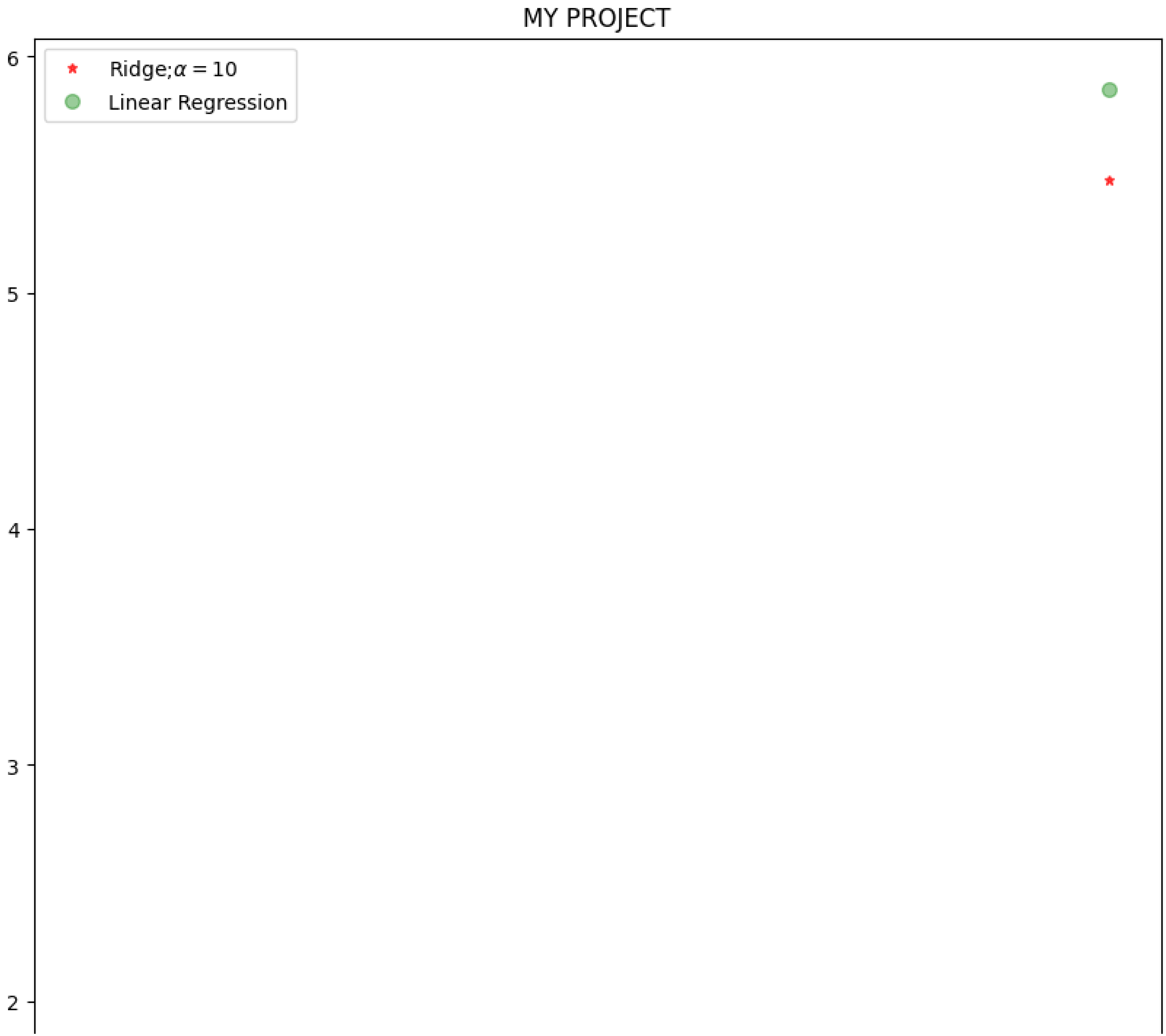
```

Ridge Model:

The train score for ridge model is 0.08784976325183447
The test score for ridge model is 0.1684728204438981
```

In [120]:

```
re(figsize=(10,10))
(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5,color='red',label=r'Ridge;$\alpha=10$'
t(lasso_cv.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color='blue',label=r'lasso;$\alpha=grid$')
(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color='green',label=r'Linear Regression')
ks(rotation=90)
nd()
e("MY PROJECT")
()
```


MY PROJECT
★ Ridge;$\alpha = 10$
● Linear Regression
6
5
4
3
2

In [121]:

```
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))
```

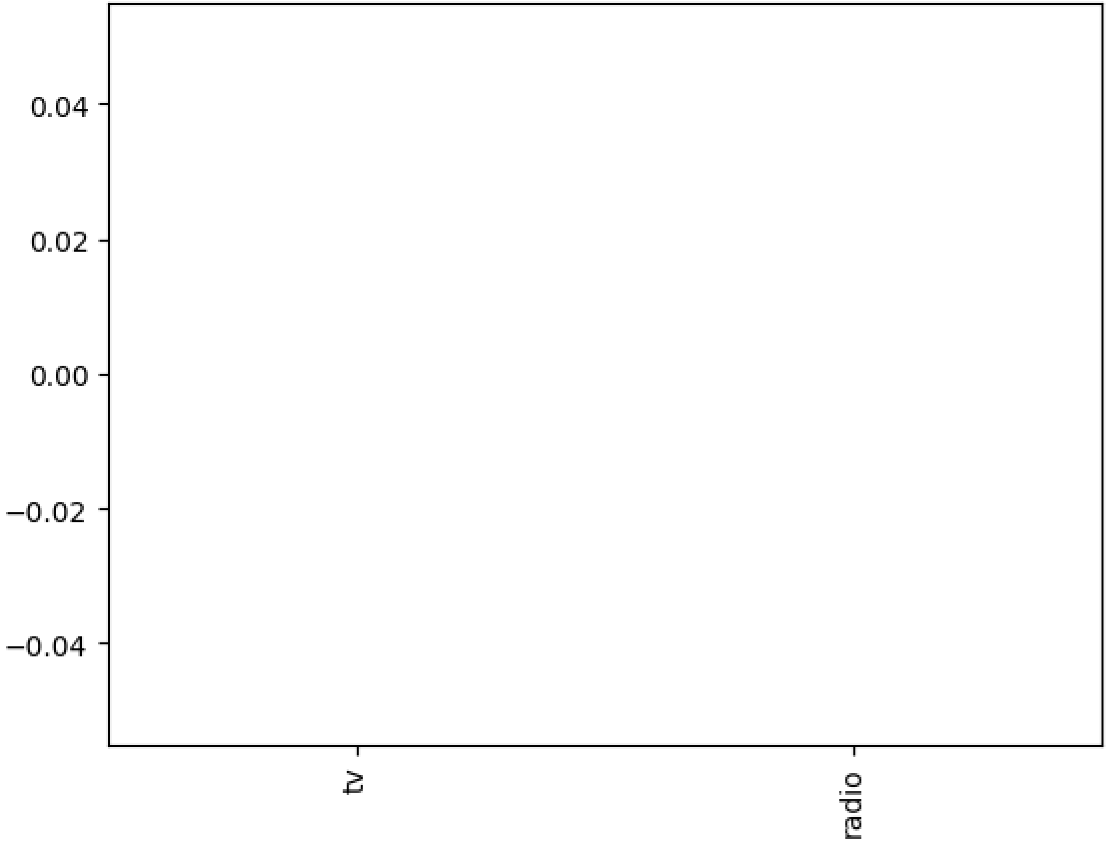
```

Lasso Model: 

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

```
In [122]: pd.Series(lasso.coef_,features).sort_values(ascending=True).plot(kind="bar")

Out[122]: <Axes: >
```

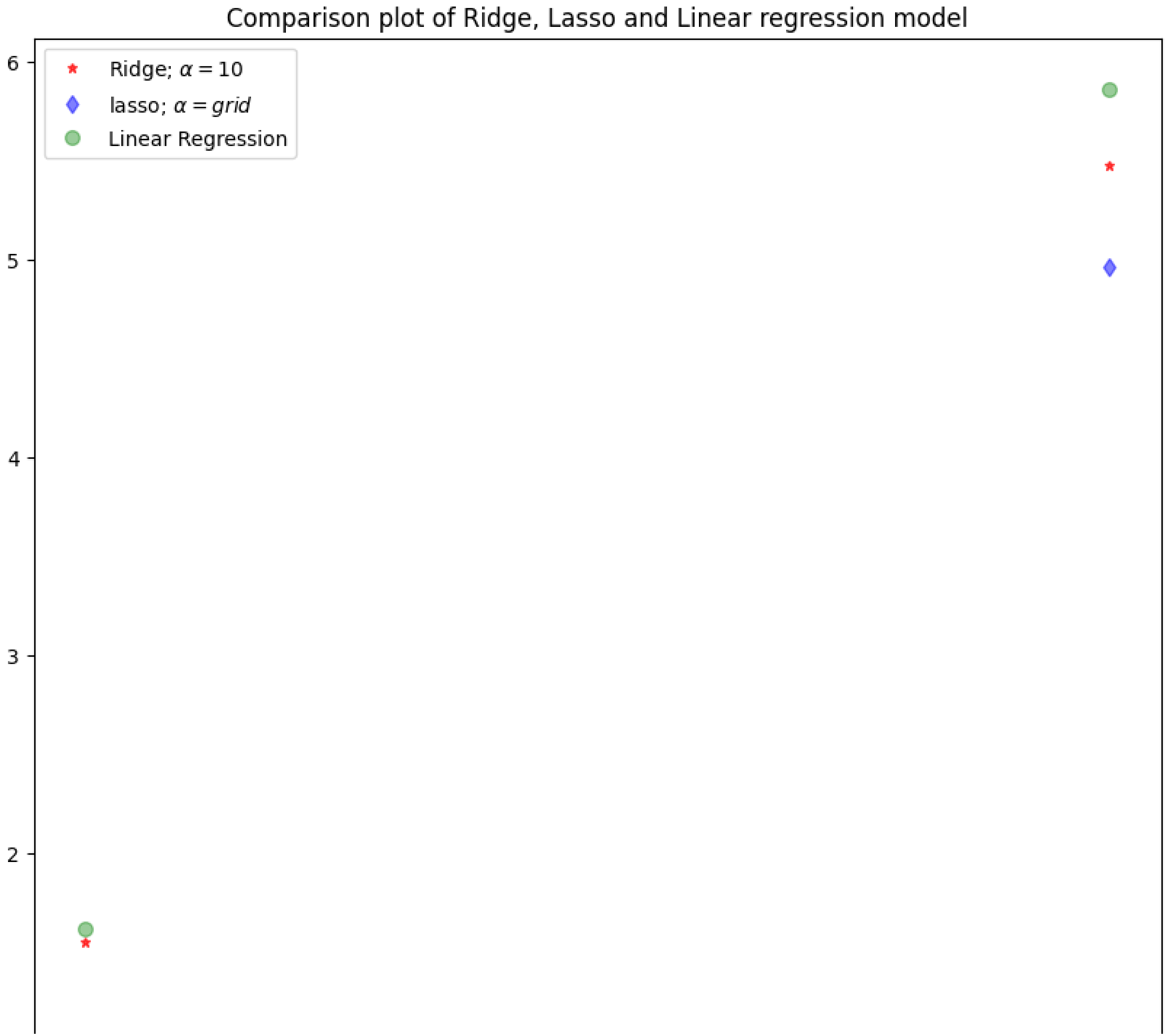


```
In [123]: from sklearn.linear_model import LassoCV
          lasso_cv=LassoCV(alphas=[0.0001,0.001,0.01,0.1,1,10],random_state=0).fit(x_train,y_train)
          print(lasso_cv.score(x_train,y_train))
          print(lasso_cv.score(x_test,y_test))

          0.08414370280677297
          0.16155807938746747
```

In [125]:

```
#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,color='red',label=r'Ridge; $\
#add plot for lasso regression
plt.plot(lasso_cv.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color='blue',label=r'lasso; $\alpha =
#add plot for linear model
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color='green',label='Linear Regress
#rotate axis
plt.xticks(rotation = 90)
plt.legend()
plt.title("Comparison plot of Ridge, Lasso and Linear regression model")
plt.show()
```


Comparison plot of Ridge, Lasso and Linear regression model
★ Ridge; $\alpha = 10$
♦ lasso; $\alpha = grid$
● Linear Regression
6
5
4
3
2

In [131]:

```
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 test score for ridge model is {}".format(ridge_cv.score(x_test,y_test)))
```

```
The train score for ridge model is 0.08784976325183436
The test score for ridge model is 0.16847282044389655
```

In [ ]: