In [1]:

```
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 [2]:

```
df=pd.read_csv(r"C:\Users\DELL E5490\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=df[['Sales','Radio','TV','Newspaper']]
df.columns=['sales','radio','tv','newspaper']
```

In [4]:
```
df.head()
```

Out[4]:

	sales	radio	tv	newspaper
0	22.1	37.8	230.1	69.2
1	10.4	39.3	44.5	45.1
2	12.0	45.9	17.2	69.3
3	16.5	41.3	151.5	58.5
4	17.9	10.8	180.8	58.4

In [5]:
```
df.describe()
```

Out[5]:

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

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

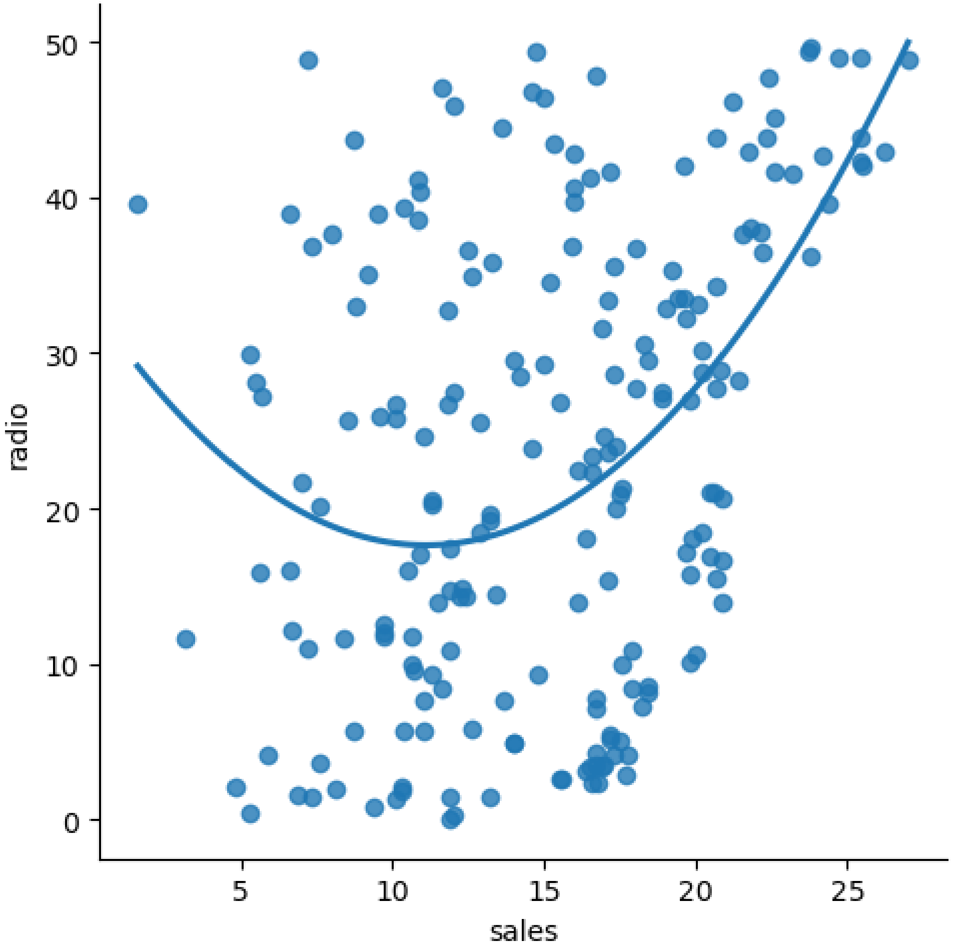
Out[6]:
```
     sales  radio     tv  newspaper
195    7.6    3.7   38.2       13.8
196   14.0    4.9   94.2        8.1
197   14.8    9.3  177.0        6.4
198   25.5   42.0  283.6       66.2
199   18.4    8.6  232.1        8.7
```

In [7]:

```
sns.lmplot(x="sales",y="radio",data=df,order=2,ci=None)
```

Out[7]:

```
<seaborn.axisgrid.FacetGrid at 0x264510f57e0>
```



In [8]:

```
df.info()
```

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

In [9]:

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

In [10]:

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.25)
```

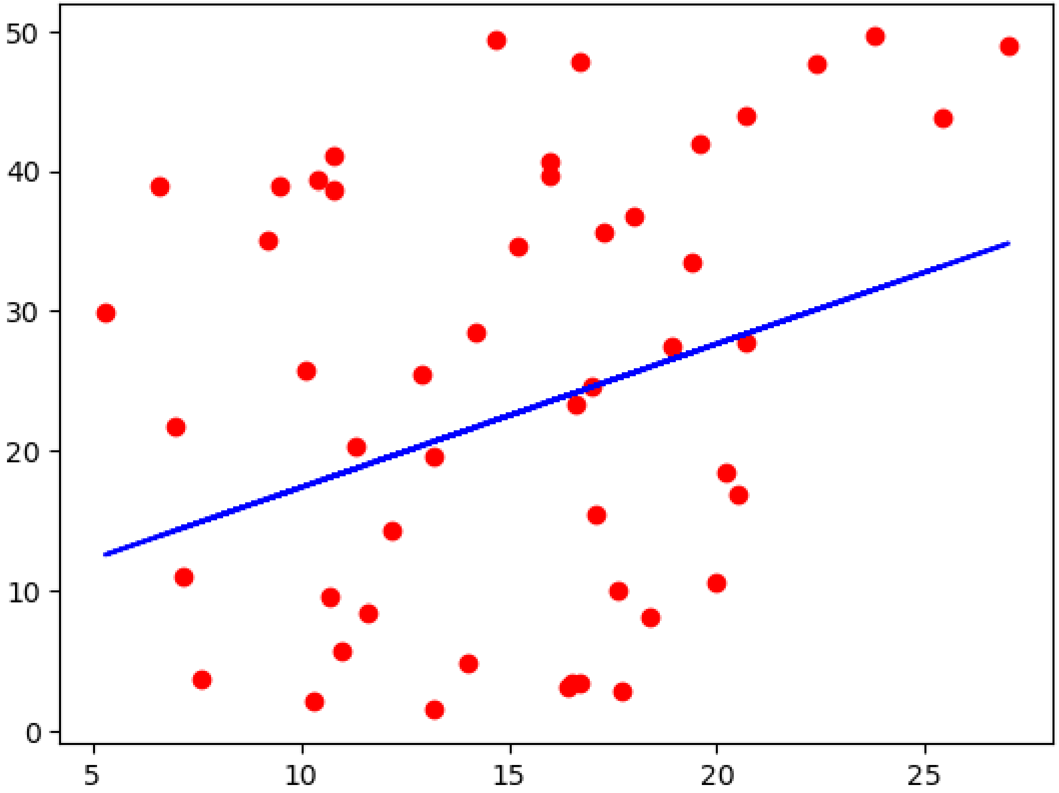
In [11]:

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

```
0.04288065646388528
```

In [12]:

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

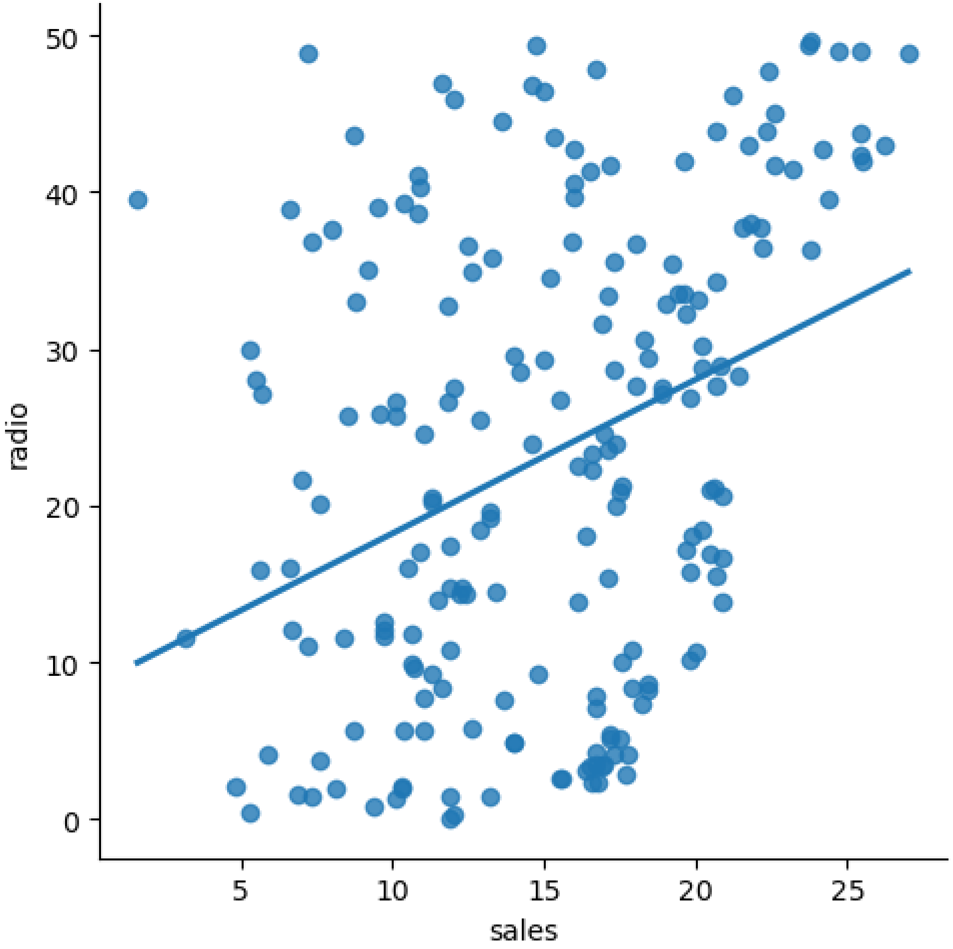


In [13]:

```
df500=df[:][:500]

sns.lmplot(x="sales",y="radio",data=df500,order=1,ci=None)
```

Out[13]:

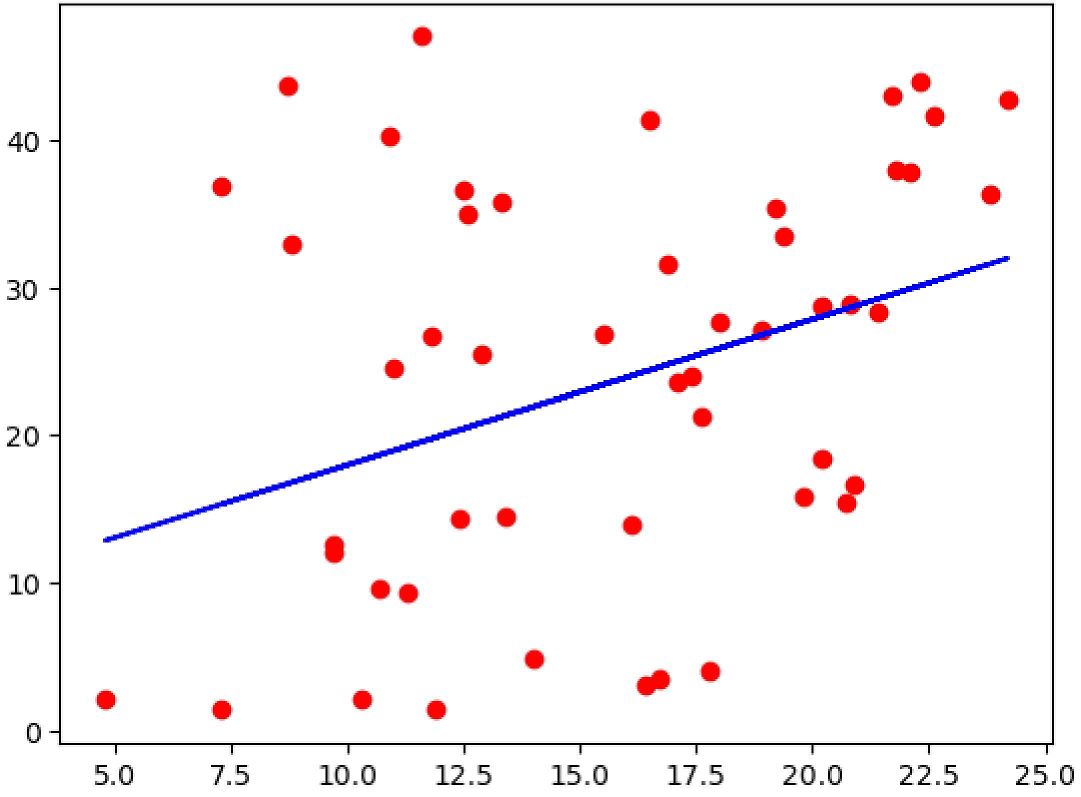
```
<seaborn.axisgrid.FacetGrid at 0x264519a5300>
```



In [ ]:

In [14]:
```
df500.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='r')
plt.plot(X_test,y_pred,color='b')
plt.show()
```

```
Regression: 0.11402599094377575
```



In [15]:

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
#train the model
model=LinearRegression()
model.fit(X_train,y_train)
y_pred=model.predict(X_test)
r2=r2_score(y_test,y_pred)
print("R2.score:",r2)
```

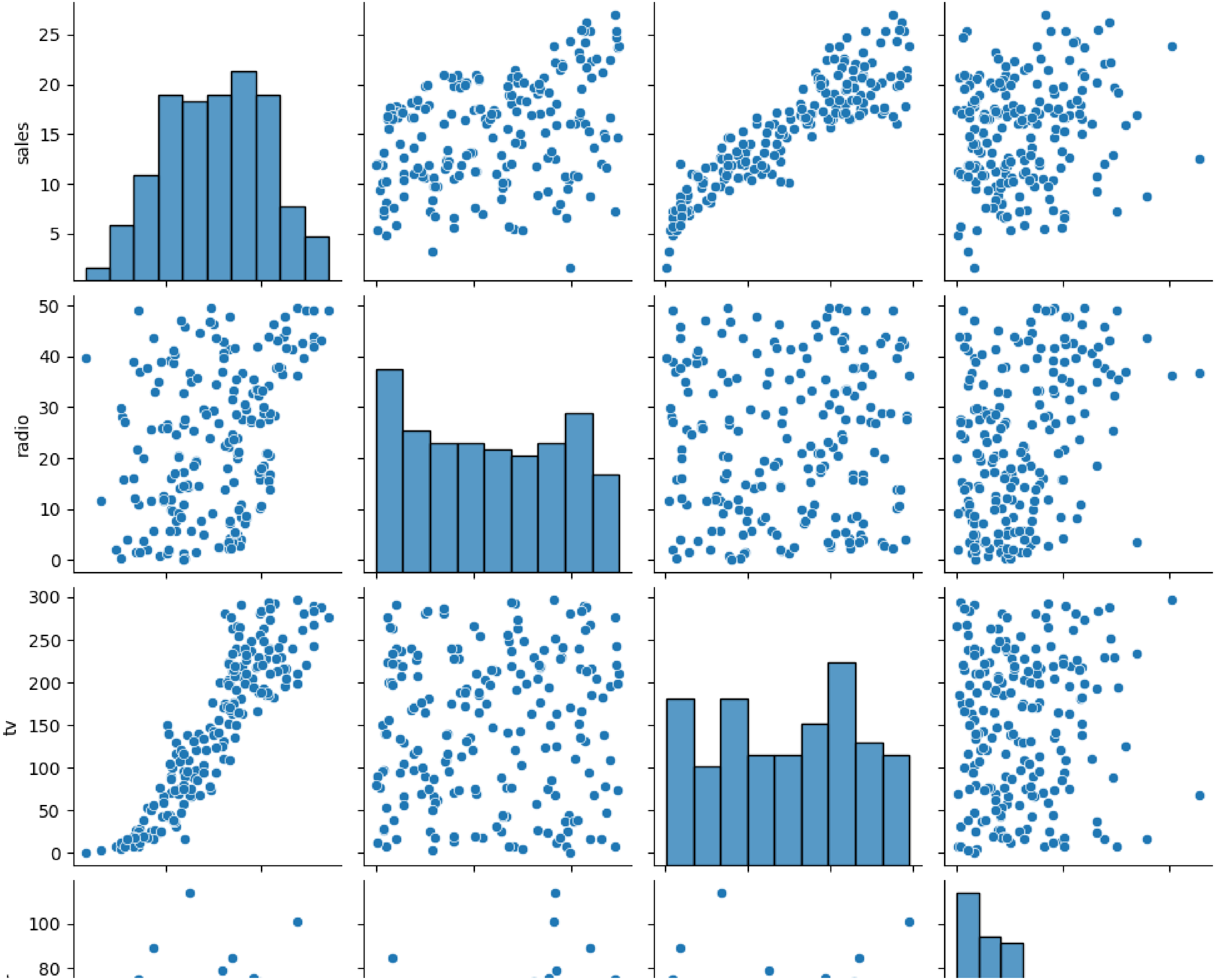
```
R2.score: 0.11402599094377575
```

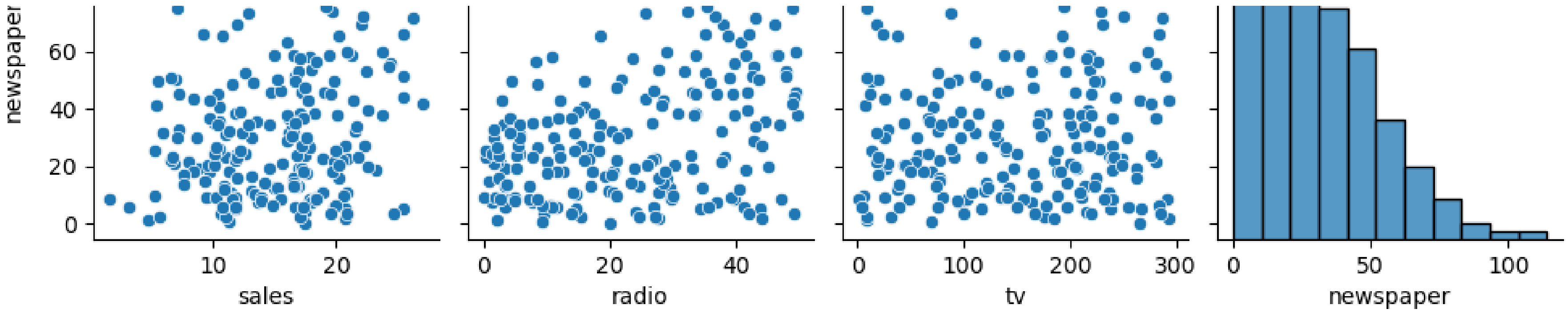
In [16]:

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

Out[16]:

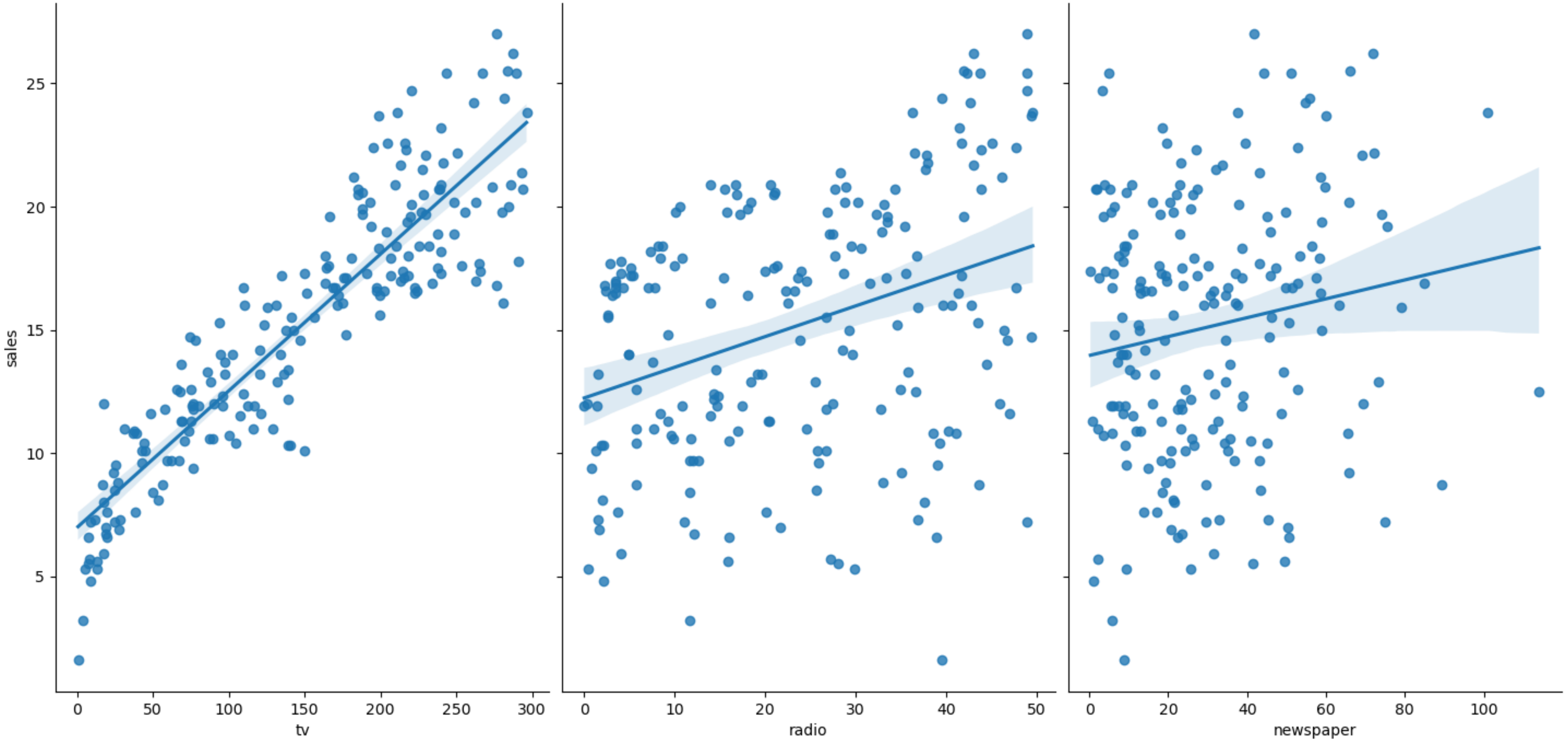
```
<seaborn.axisgrid.PairGrid at 0x264519f7760>
```


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



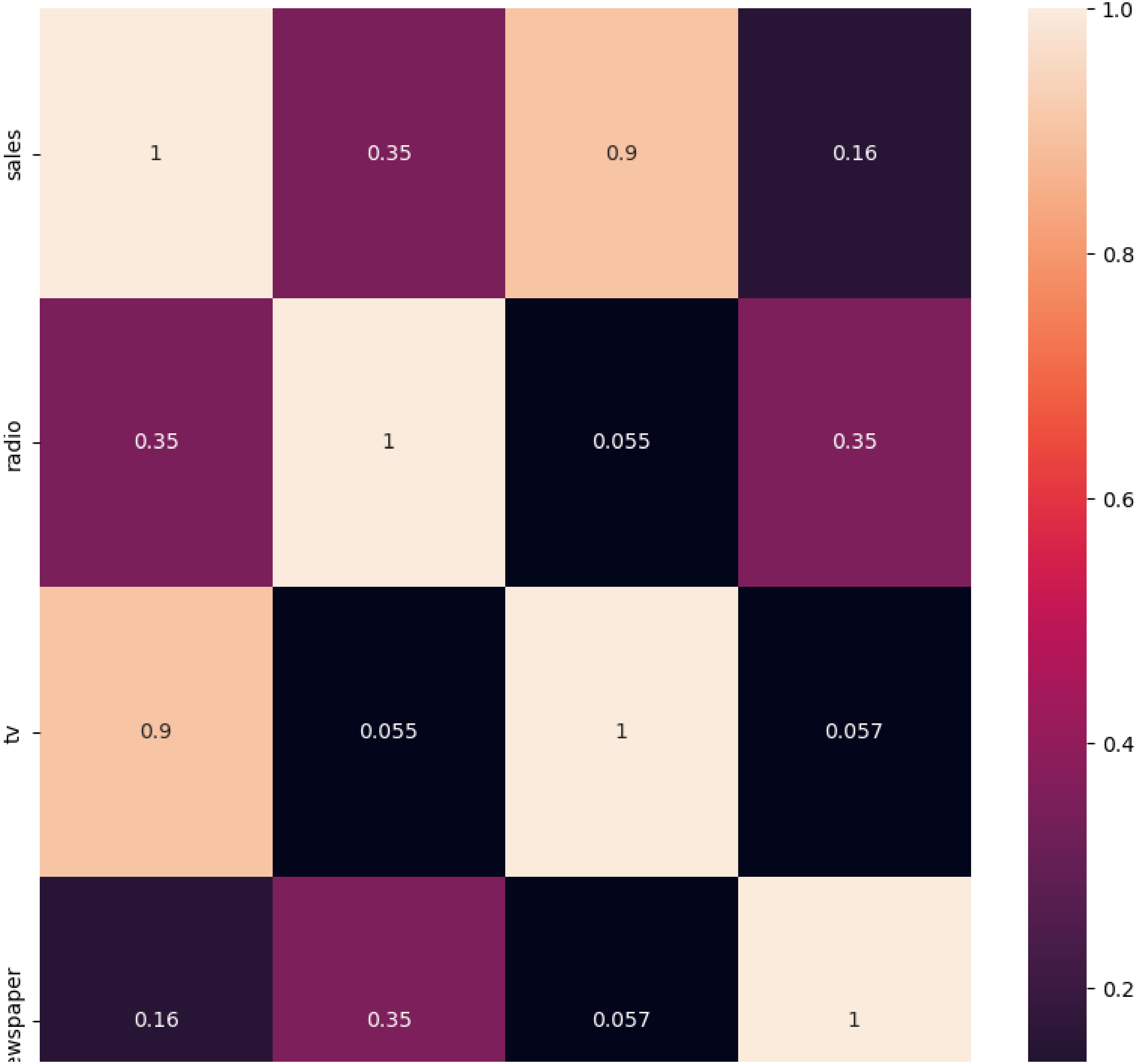
```
In [17]: sns.pairplot(df,x_vars=['tv','radio','newspaper'],y_vars='sales',height=7,aspect=0.7,kind='reg')
```

```
Out[17]: <seaborn.axisgrid.PairGrid at 0x26453636c20>
```



In [18]:
```
plt.figure(figsize=(10,10))
sns.heatmap(df.corr(),annot = True)
```

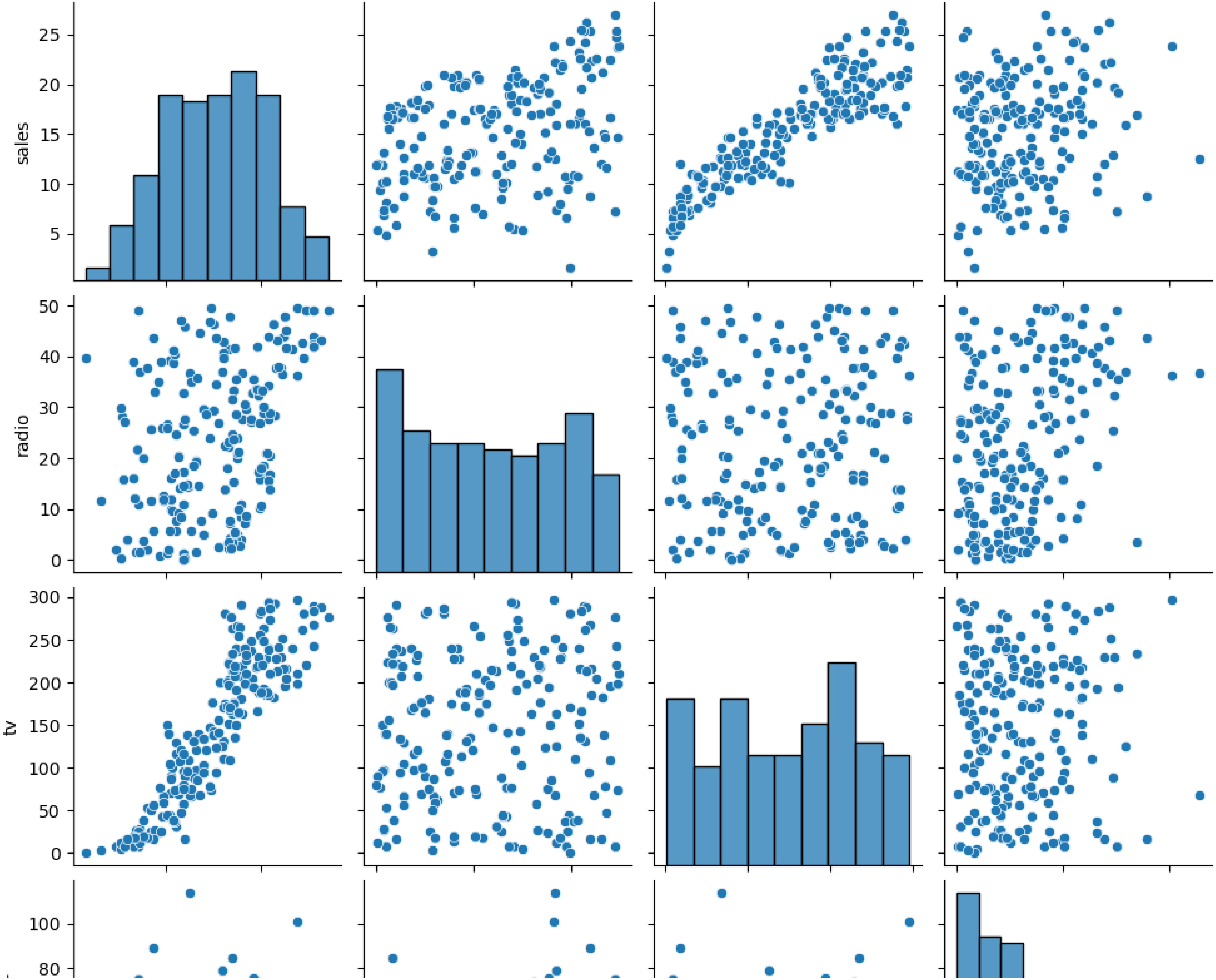
Out[18]:
```
<Axes: >
```

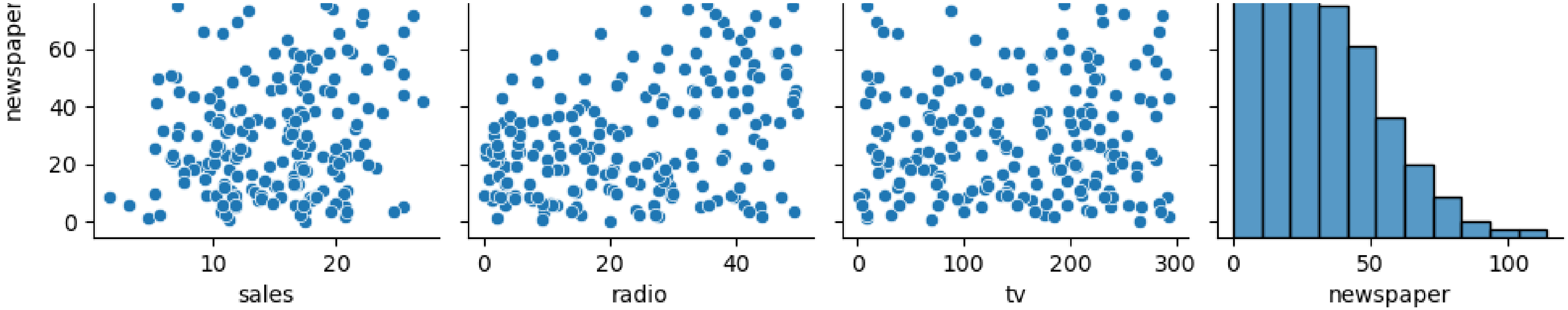

sales
radio
tv
ewspaper
1
0.35
0.9
0.16
0.35
1
0.055
0.35
0.9
0.055
1
0.057
0.16
0.35
0.057
1
1.0
0.8
0.6
0.4
0.2

ne
sales
radio
tv
newspaper

In [19]:

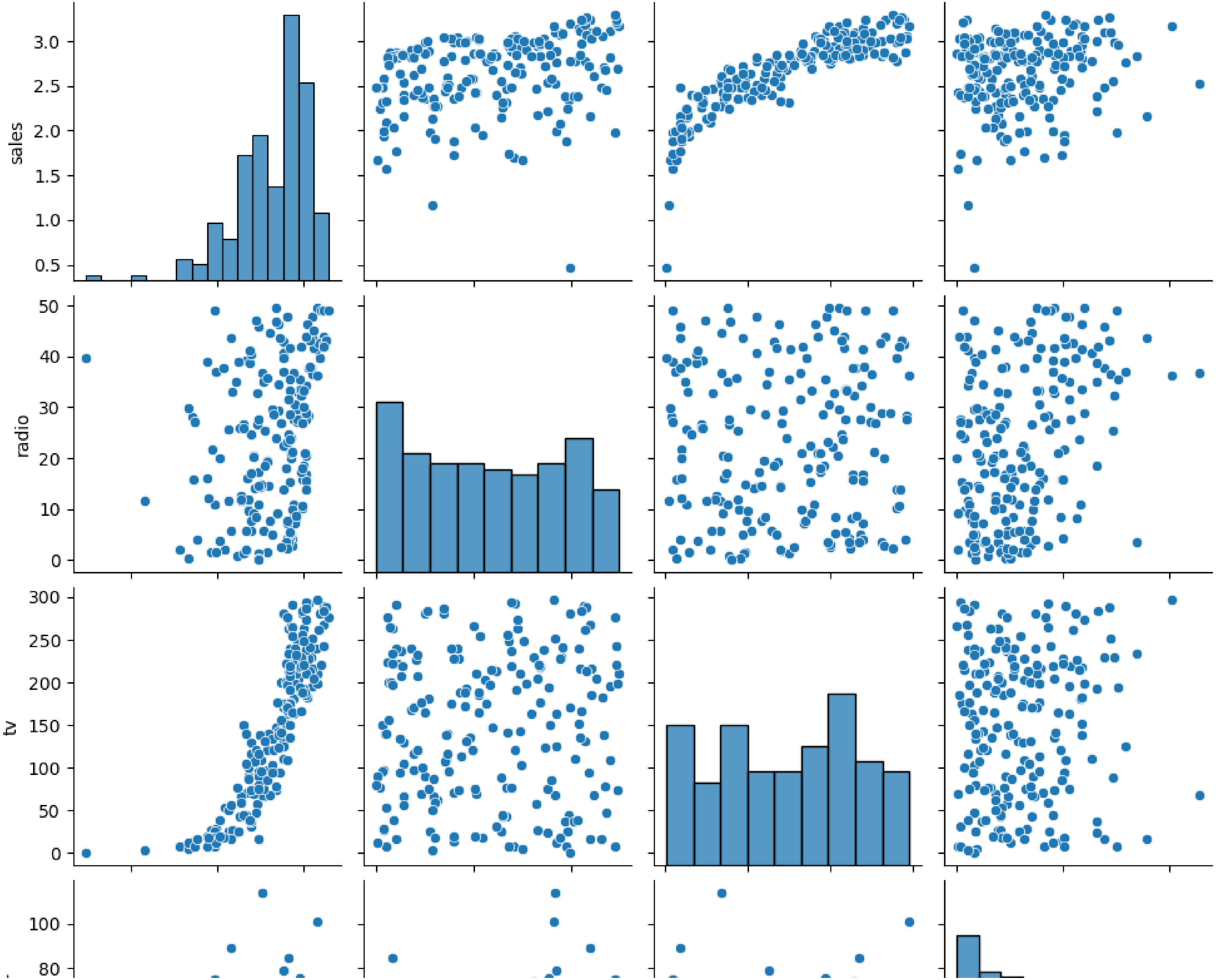
```

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

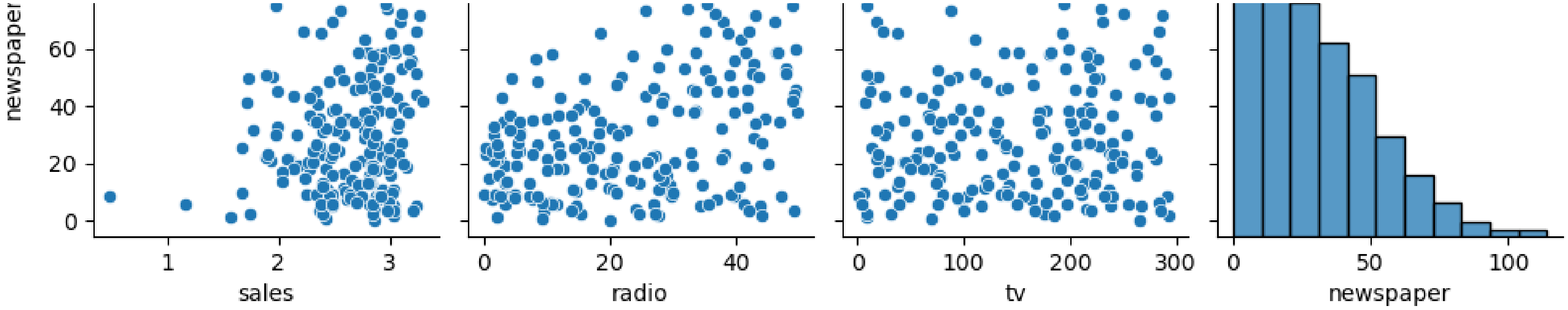

sales
25
20
15
10
5
radio
50
40
30
20
10
0
tv
300
250
200
150
100
50
0
100
80


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

In [20]:

```
#pairplot
sns.pairplot(df)
df.sales=np.log(df.sales)
```


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



In [21]:
```
print(regr.score(X_test,y_test))
```

```
0.11402599094377575
```

In [22]:
```
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 [23]:

```
#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 0.08495398635901708
The test score for lr model is 0.19849660346463094
```

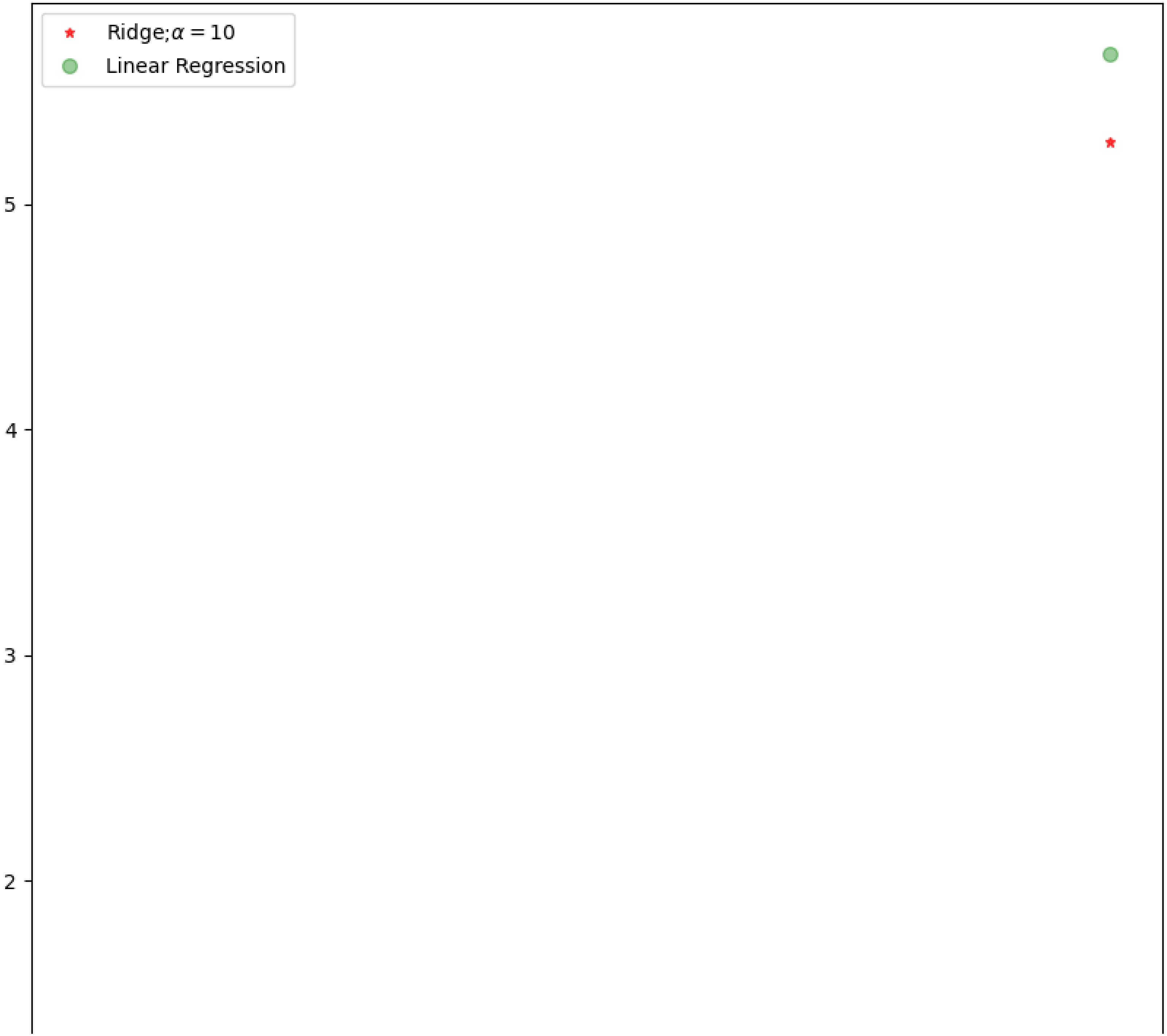
In [24]:

```
#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 0.08462956788927811
The test score for ridge model is 0.18990201599412504
```

In [25]:

```
plt.figure(figsize=(10,10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5,color='red',label=r'Ridge;$\alpha
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.show()
```


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

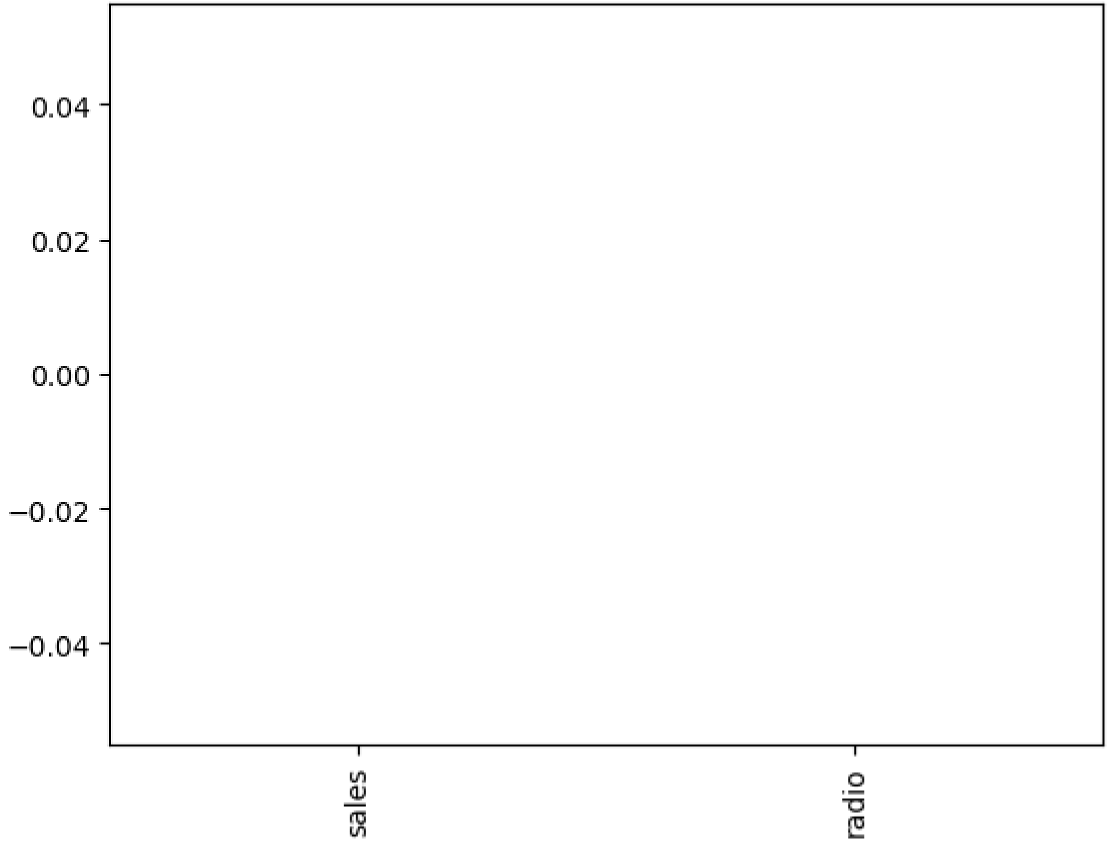
In [26]:

```
#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 -0.00035473346594128l5
```

In [27]:
```
pd.Series(lasso.coef_,features).sort_values(ascending = True).plot(kind= "bar")
```

Out[27]:
```
<Axes: >
```



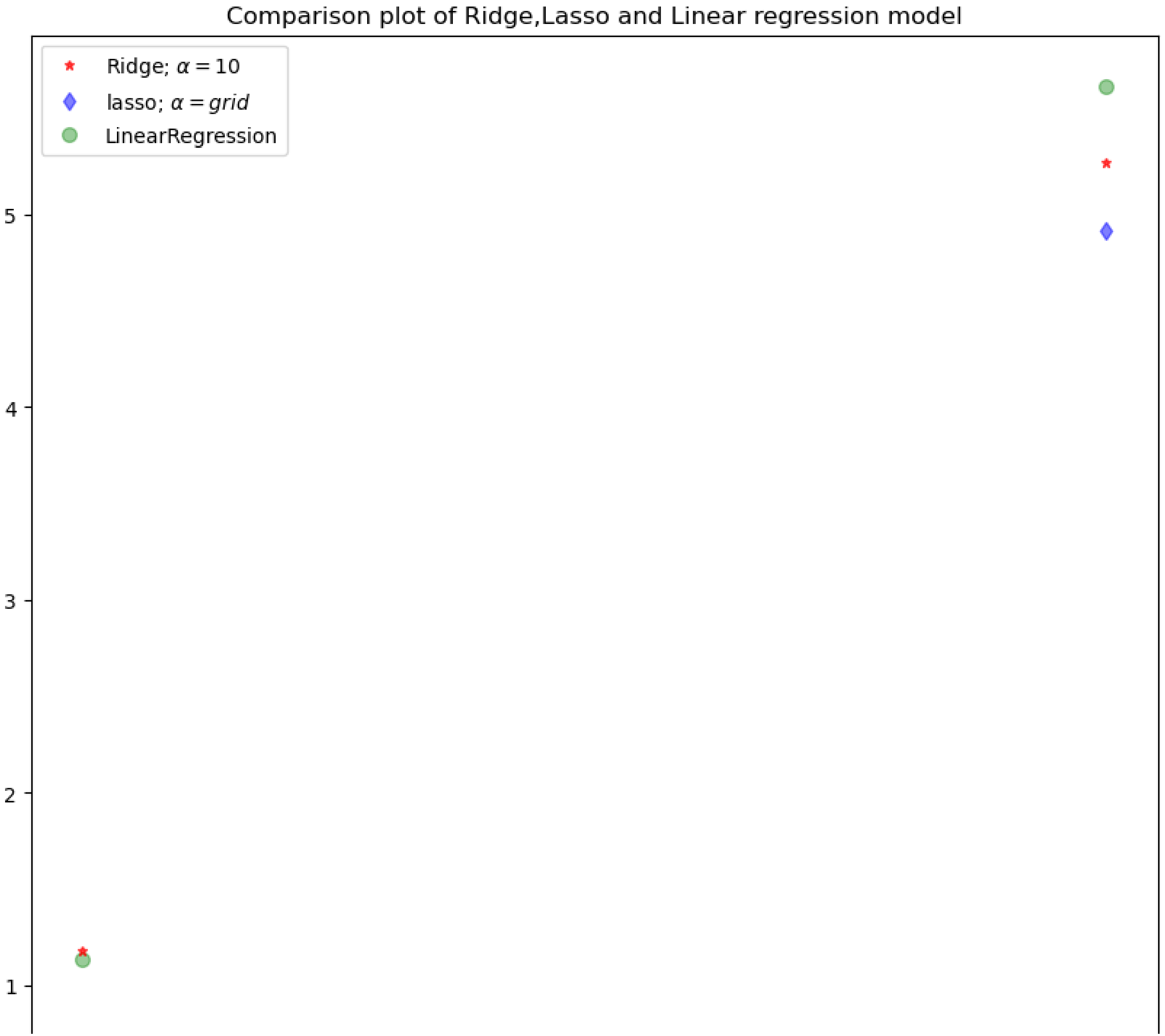
In [28]:

```
#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))
```

```
0.08156518528706347
0.1705627246981769
```

In [29]:

```
#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; $\alph
#add plot for lasso regression
plt.plot(lasso_cv.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color='blue',label=r'lasso; $\alpha=grid$'
#add plot for linear model
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color='green',label='LinearRegression')

#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; $\alpha = grid$
● LinearRegression
5
4
3
2
1

In [30]:
```
#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 0.08462956788927811
The train score for ridge model is 0.18990201599412537
```

In [31]:
```
from sklearn.linear_model import ElasticNet
```

In [37]:
```
regr=ElasticNet()
regr.fit(X,y)
print(regr.coef_)
print(regr.intercept_)
```

```
[0.        0.5159749]
18.55035993077493
```

In [33]:
```
y_pred_elastic=regr.predict(X_train)
```

In [36]:

```
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 578.2547454190322
```

In [ ]: