In [2]:
```
import pandas as pd
import numpy as np
```

In [3]:
```
df=pd.read_csv(r"C:\Users\SASIDHAR ROYAL\Downloads\Advertising.csv")
df
```

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
...	...	...	...	...
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 [4]:

```
df.head()
```

Out[4]:

	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 [5]:

```
df.tail()
```

Out[5]:

	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 [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.columns
```

Out[7]:
```
Index(['TV', 'Radio', 'Newspaper', 'Sales'], dtype='object')
```

In [8]:
```
df.shape
```

Out[8]:
```
(200, 4)
```

In [9]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

In [10]:
```
from sklearn import preprocessing,svm
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler
```

In [11]:
```
df=df[['TV','Radio','Newspaper','Sales']]
```

In [12]:

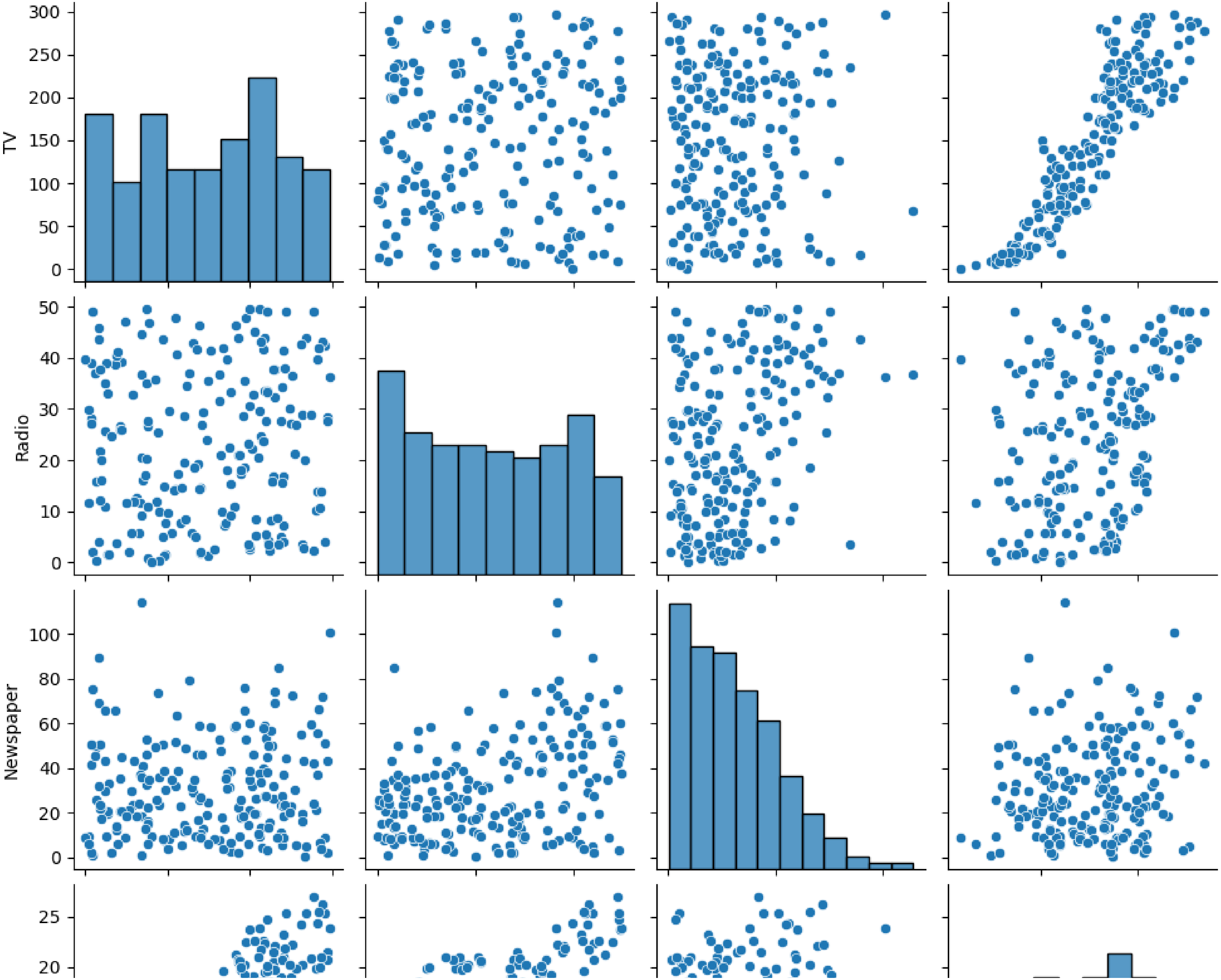
```
df.columns=['TV','Radio','Newspaper','Sales']
```

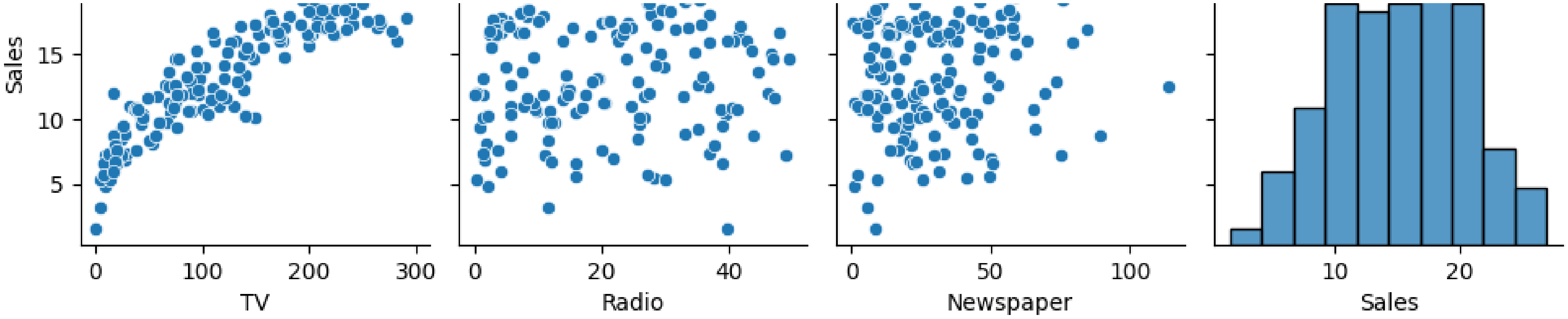
In [13]:

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

Out[13]:

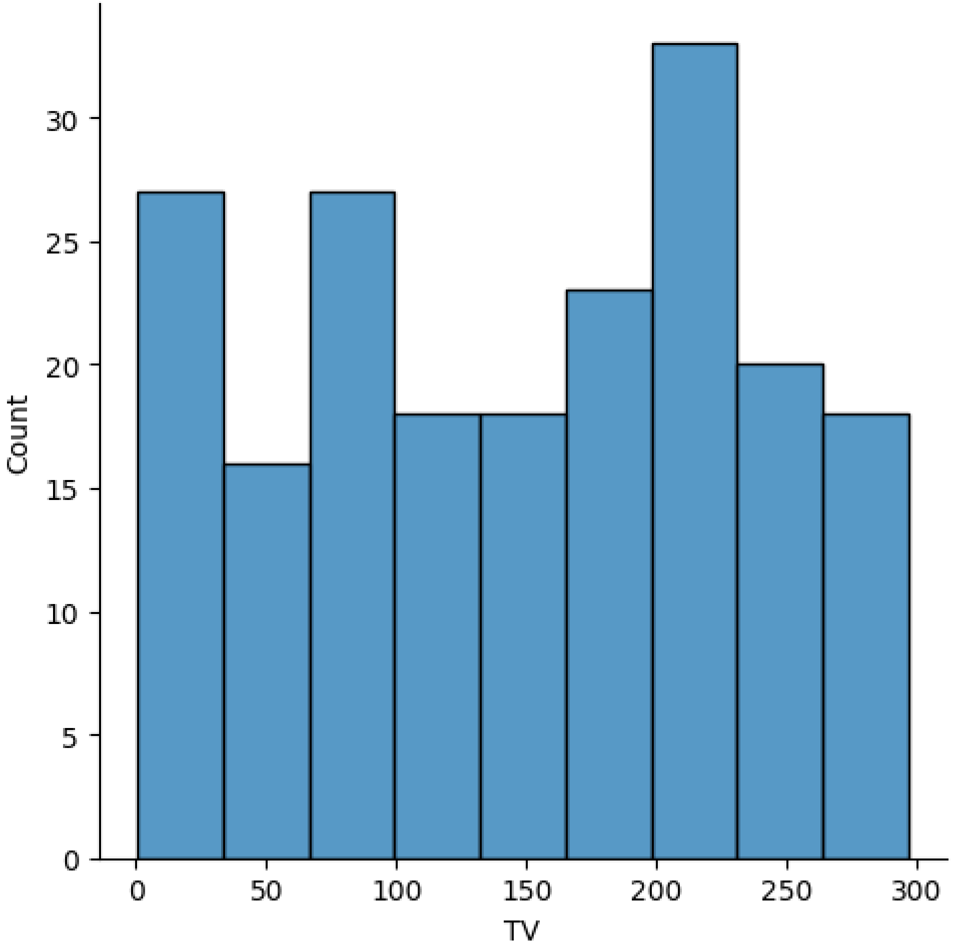
```
<seaborn.axisgrid.PairGrid at 0x27fb3f01950>
```


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


Sales
15
10
5
0
100
200
300
TV
0
20
40
Radio
0
50
100
Newspaper
10
20
Sales

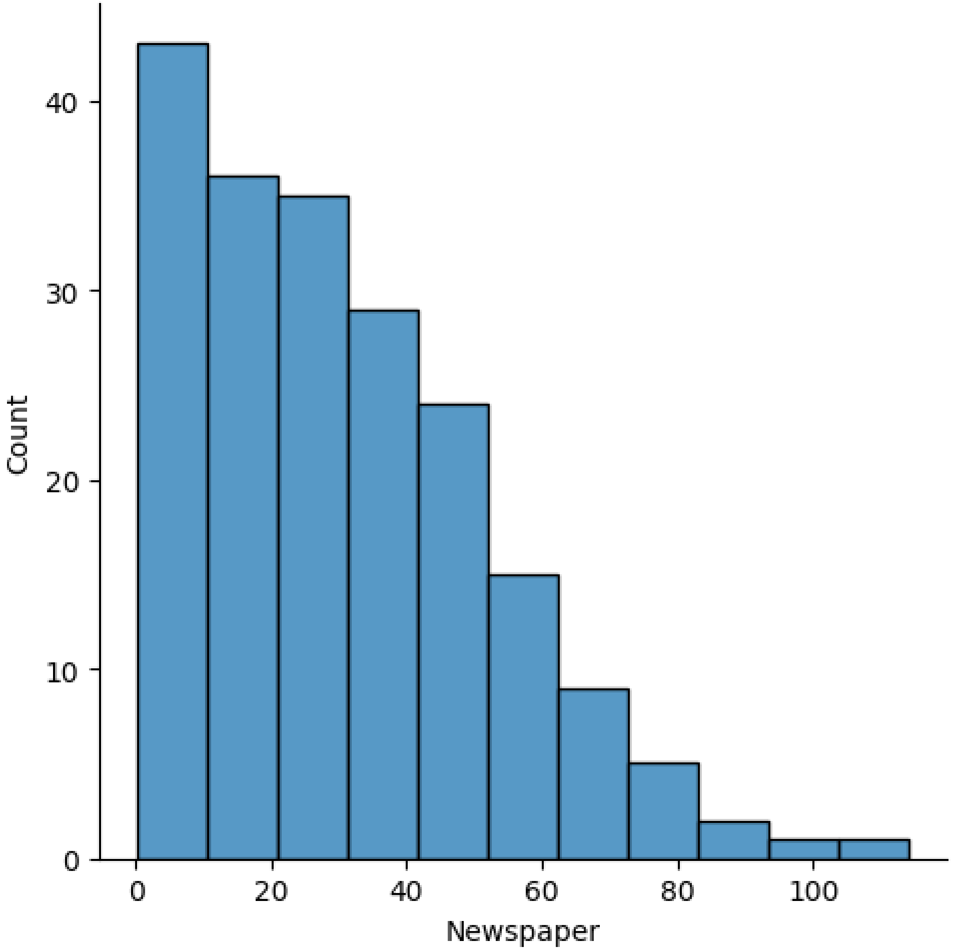
```
In [14]: sns.displot(df['TV'])
```

```
Out[14]: <seaborn.axisgrid.FacetGrid at 0x27fc92fce10>
```



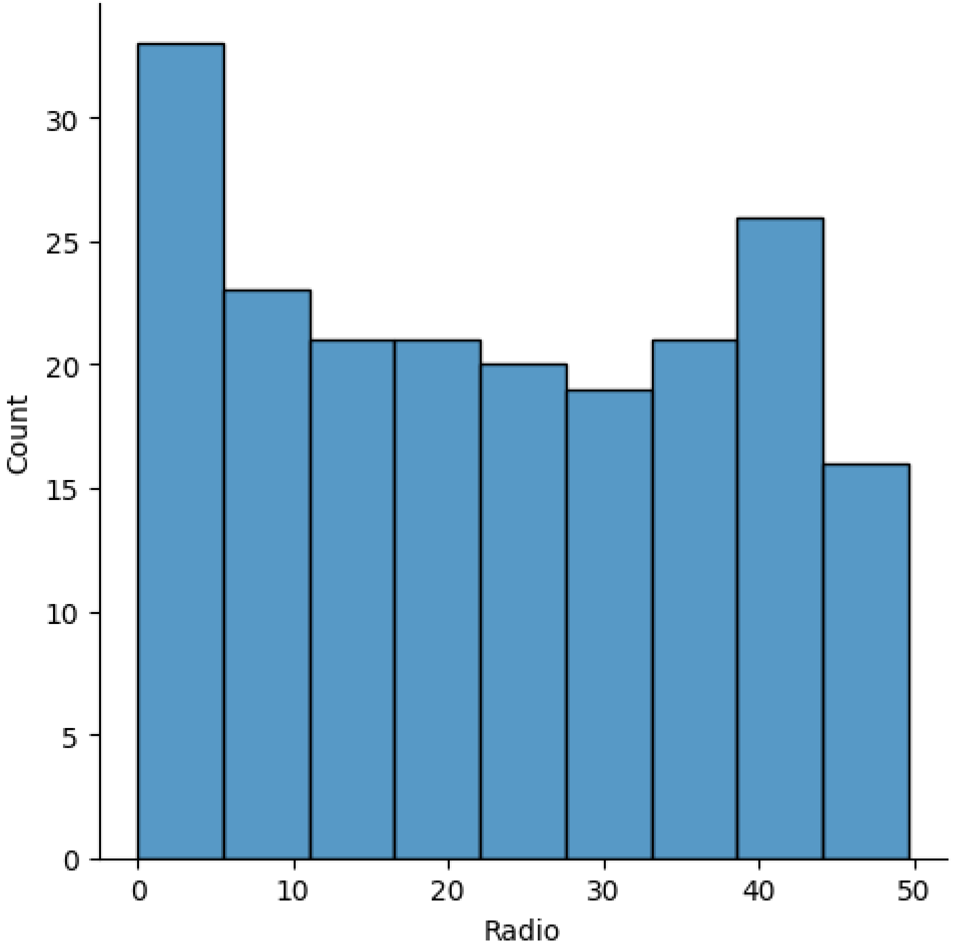
In [15]:
```
sns.displot(df['Newspaper'])
```

Out[15]:
```
<seaborn.axisgrid.FacetGrid at 0x27fc9308190>
```



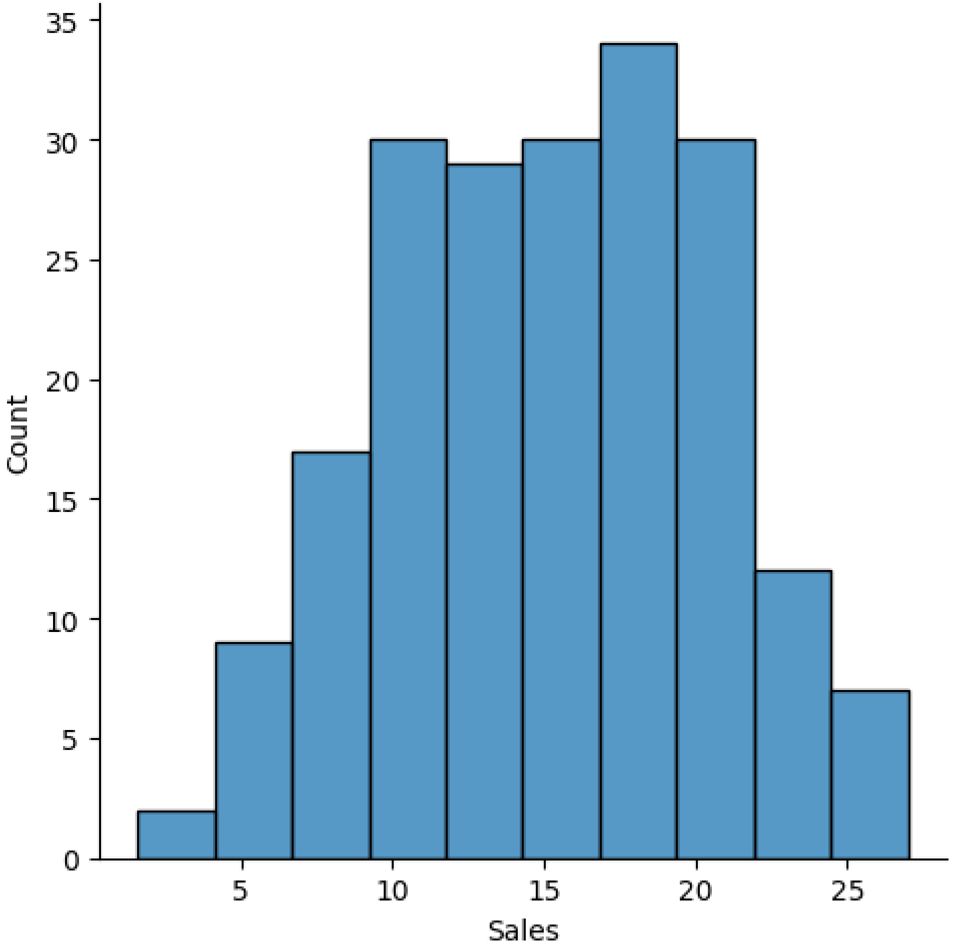
In [16]:
```
sns.displot(df['Radio'])
```

Out[16]:
```
<seaborn.axisgrid.FacetGrid at 0x27fc92fc810>
```



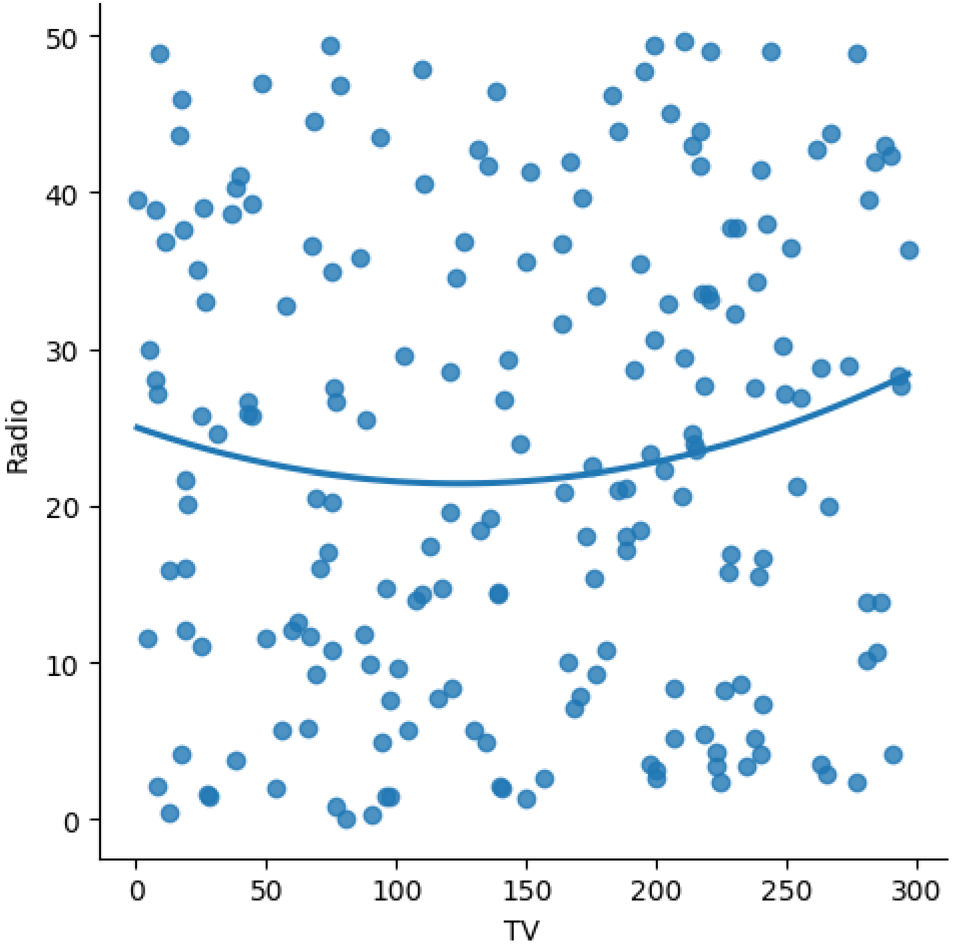
In [17]:
```
sns.displot(df['Sales'])
```

Out[17]:
```
<seaborn.axisgrid.FacetGrid at 0x27fca82ba90>
```



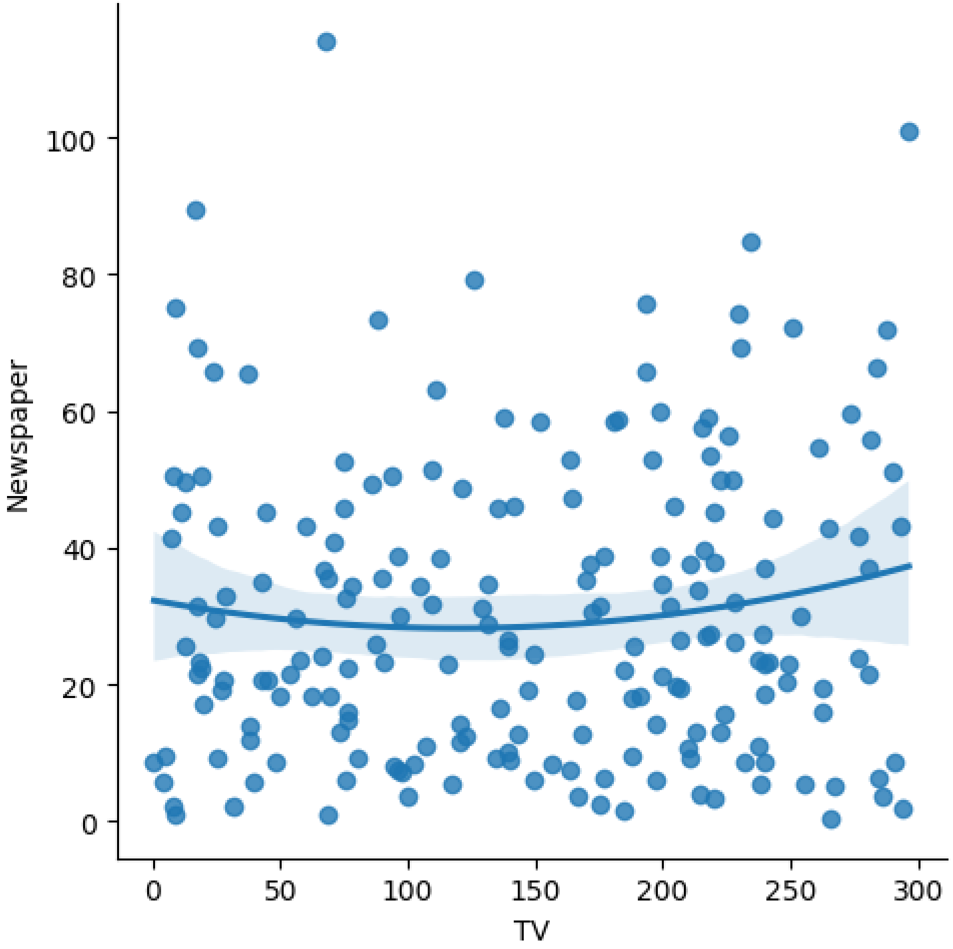
In [18]:
```
sns.lmplot(x="TV",y="Radio",data=df,order=2,ci=None)
```

Out[18]:
```
<seaborn.axisgrid.FacetGrid at 0x27fc936b250>
```



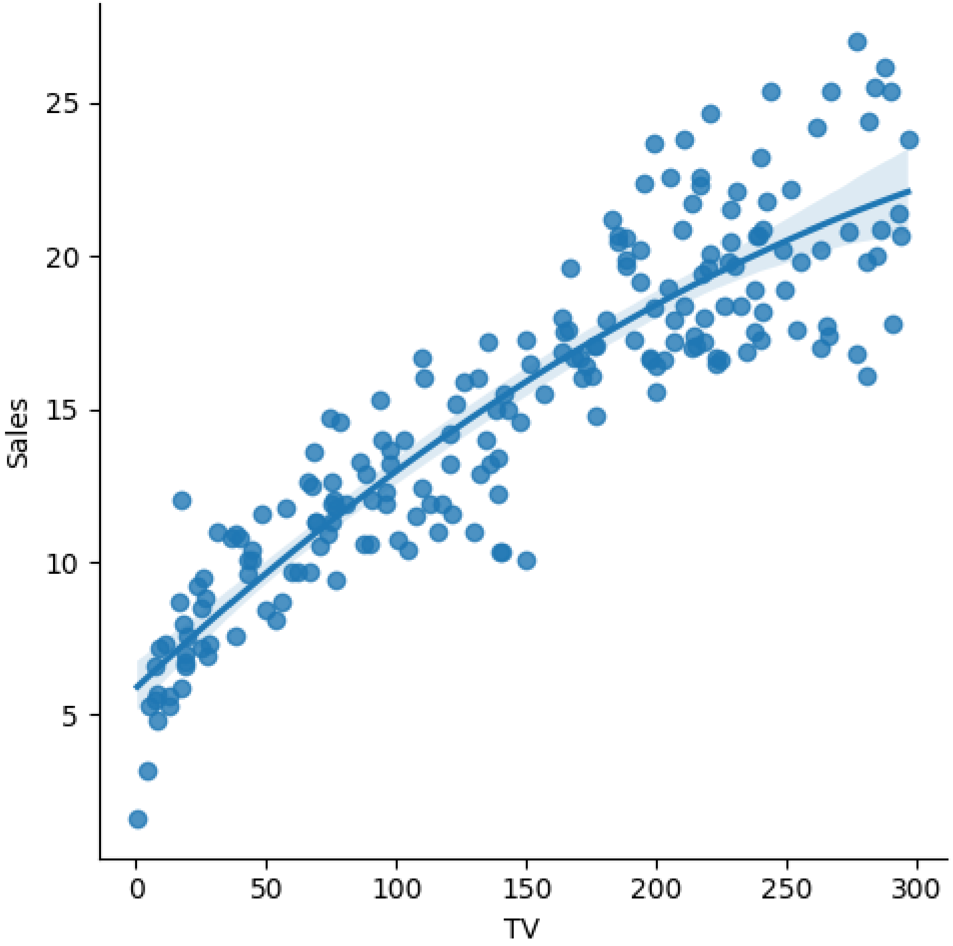
```
In [19]: sns.lmplot(x="TV",y="Newspaper",data=df,order=2)

Out[19]: <seaborn.axisgrid.FacetGrid at 0x27fca807c90>
```



In [20]:
```
sns.lmplot(x="TV",y="Sales",data=df,order=2)
```

Out[20]:
```
<seaborn.axisgrid.FacetGrid at 0x27fca98f450>
```



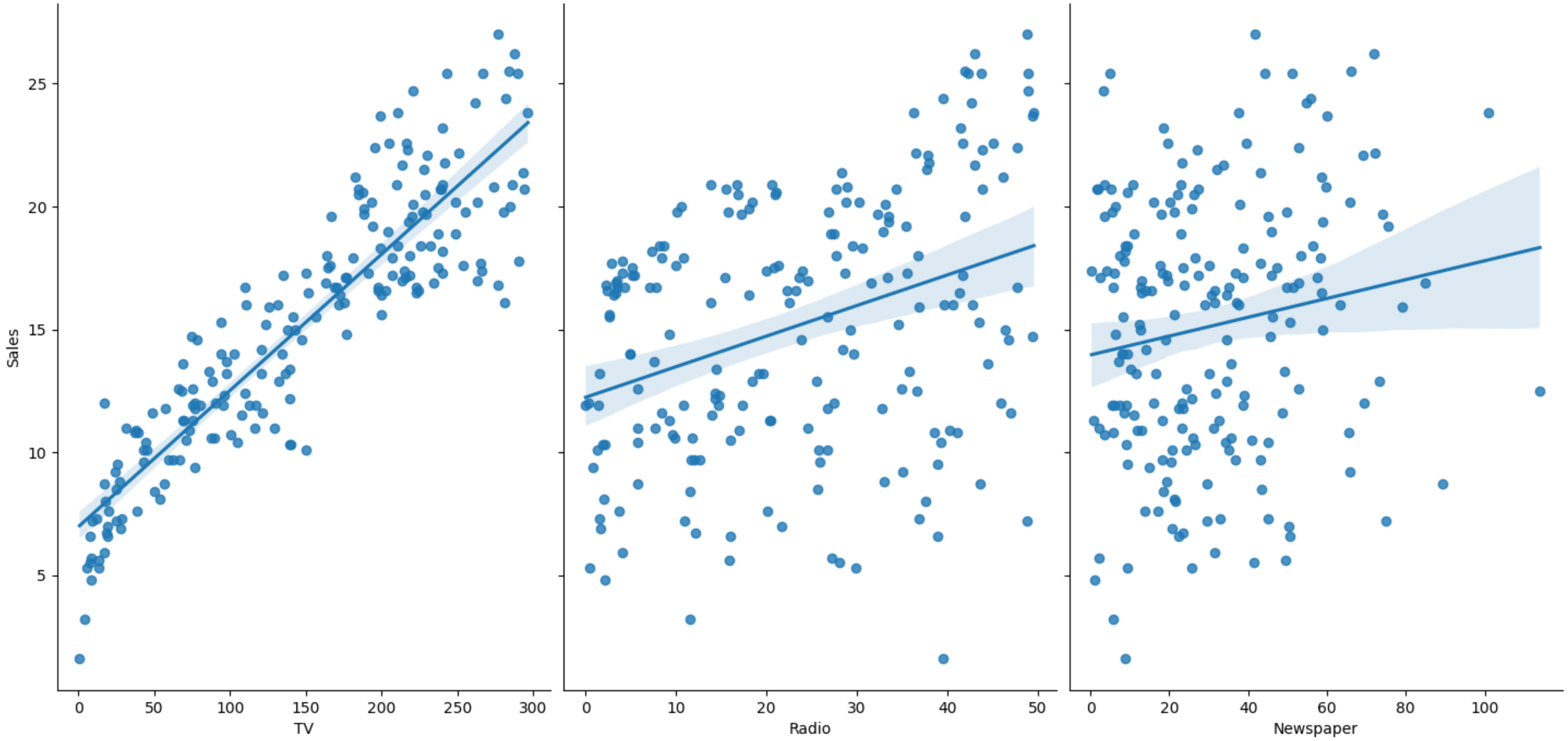
In [21]:
```
x=np.array(df['TV']).reshape(-1,1)
y=np.array(df['Radio']).reshape(-1,1)
```

In [22]:
```
df.dropna(inplace=True)
X_train,X_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
regr=LinearRegression()
regr.fit(X_train,y_train)
regr.fit(X_train,y_train)
```

Out[22]:
```
▾ LinearRegression
LinearRegression()
```

In [23]:
```
sns.pairplot(df,x_vars=['TV','Radio','Newspaper'],y_vars='Sales',height=7,aspect=0.7,kind='reg')
```

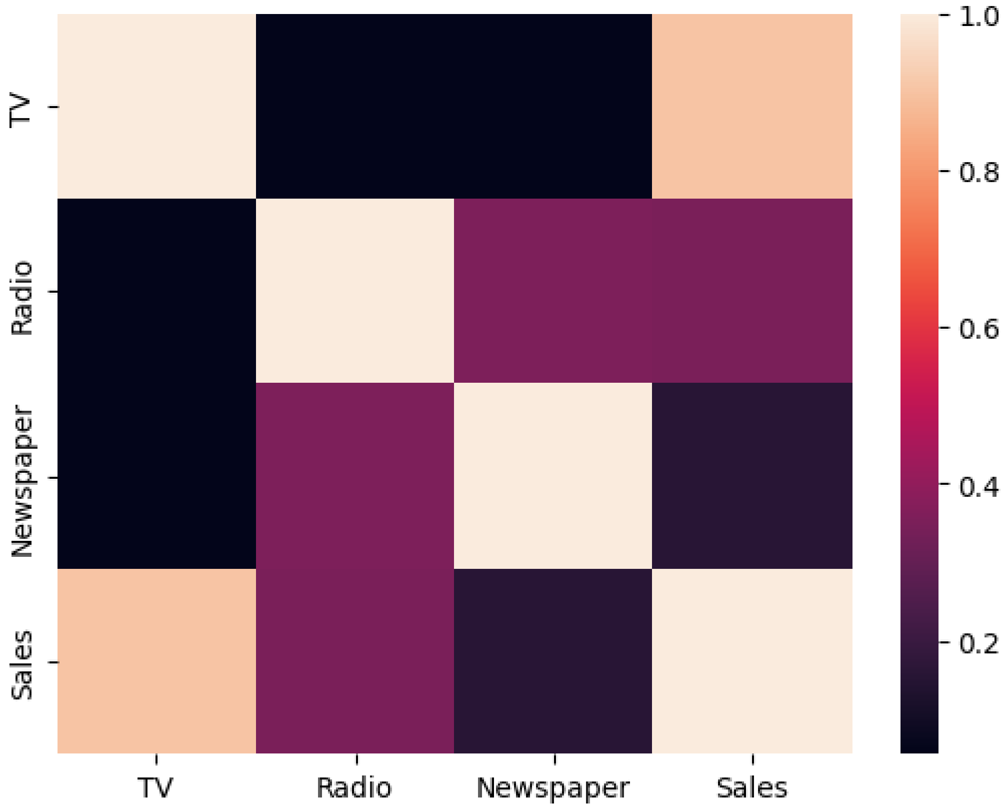
Out[23]:
```
<seaborn.axisgrid.PairGrid at 0x27fca92fd10>
```



```
In [24]: hk=df[['TV','Radio','Newspaper','Sales']]
```

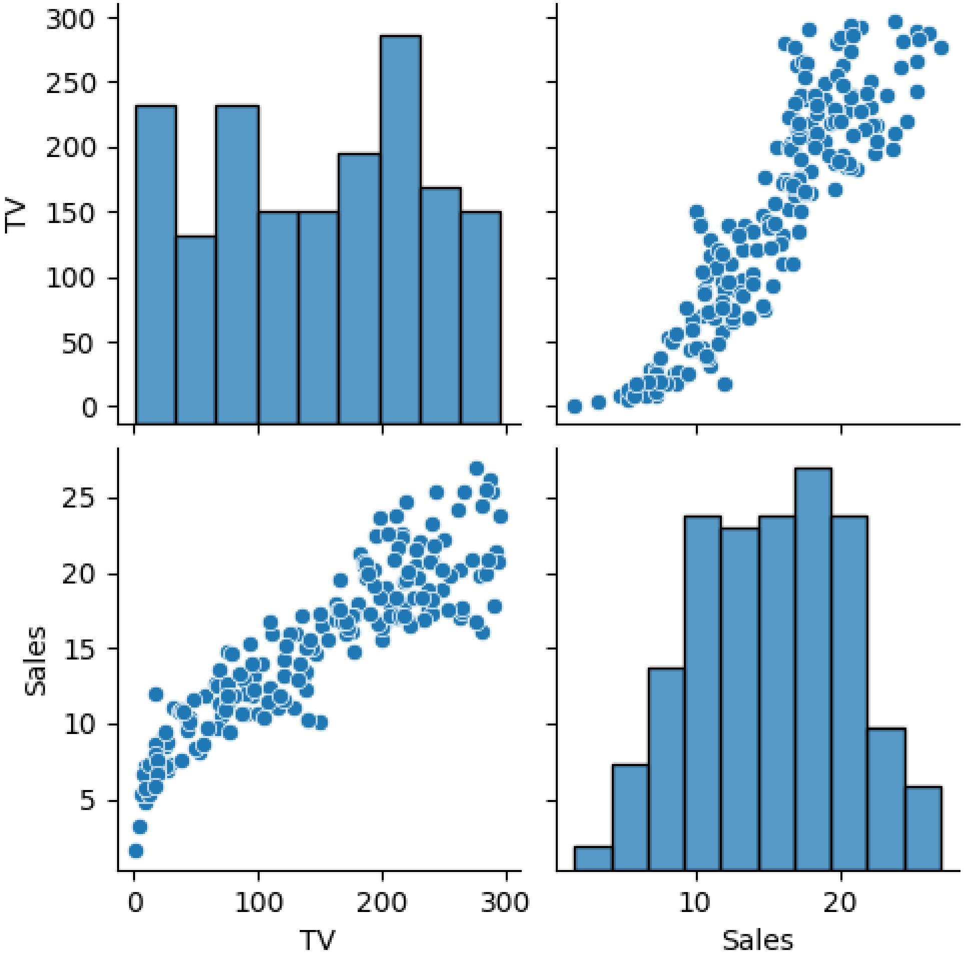
```
In [25]: sns.heatmap(hk.corr())
```

```
Out[25]: <Axes: >
```



In [26]:

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



In [27]:
```
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 [28]:
```
from sklearn.linear_model import Lasso,Ridge
```

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 train score lr model is {}".format(test_score_lr))
```

```

Linear Regression Model:

The train score for lr model is 1.0
The train score lr model is 1.0
```

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

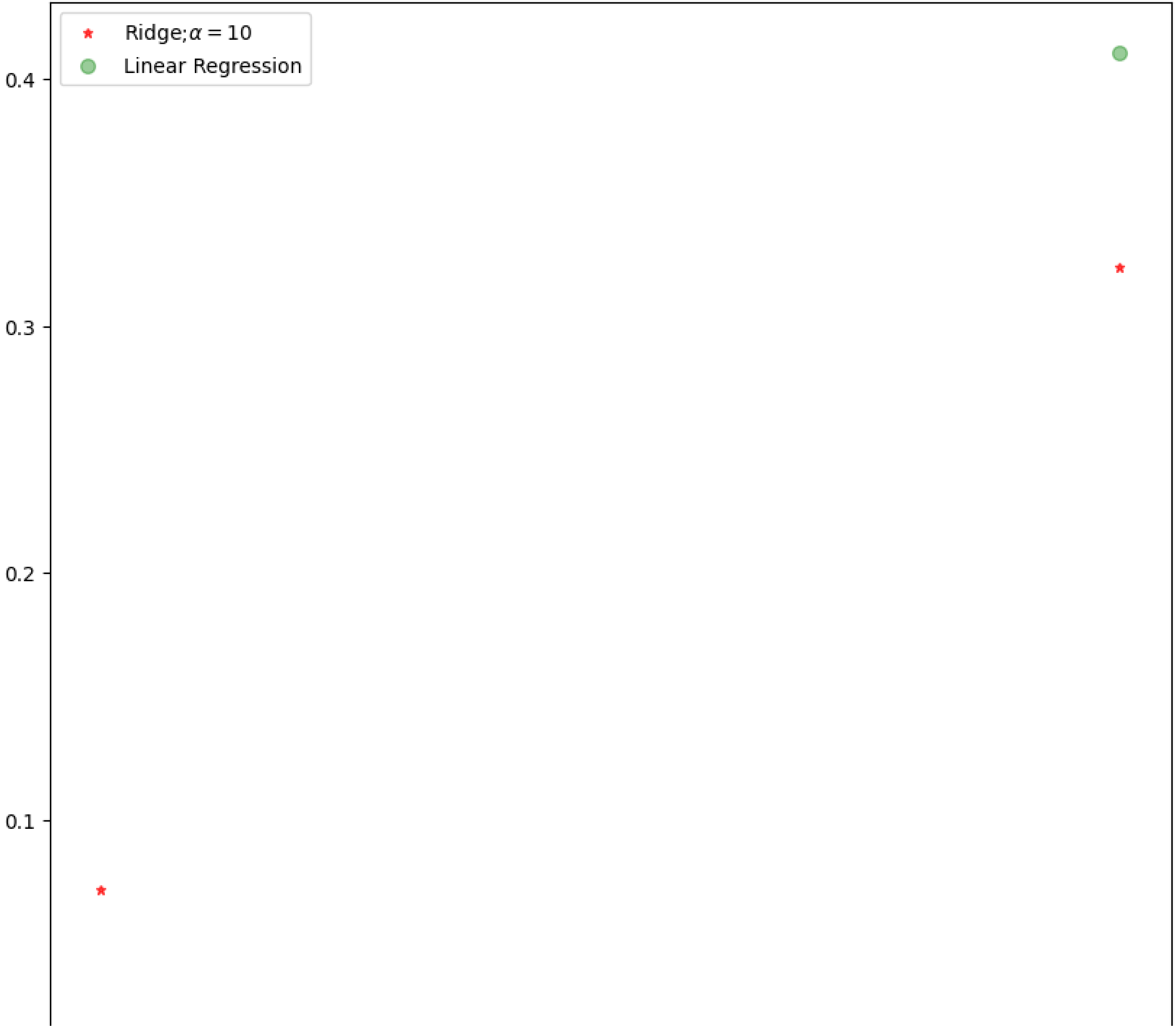
```

Ridge model\:

The train score for ridge model is 0.990287139194161
The train score for ridge model is 0.9844266285141221
```

In [31]:

```
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
0.4
0.3
0.2
0.1

In [32]:

```
lassoReg=Lasso(alpha=10)
lassoReg.fit(X_train,y_train)
train_score_lasso=lassoReg.score(X_train,y_train)
test_score_lasso=lassoReg.score(X_test,y_test)
print("\nRidge model\:\n")
print("The train score for lasso model is {}".format(train_score_ridge))
print("The train score for lasso model is {}".format(test_score_ridge))
```

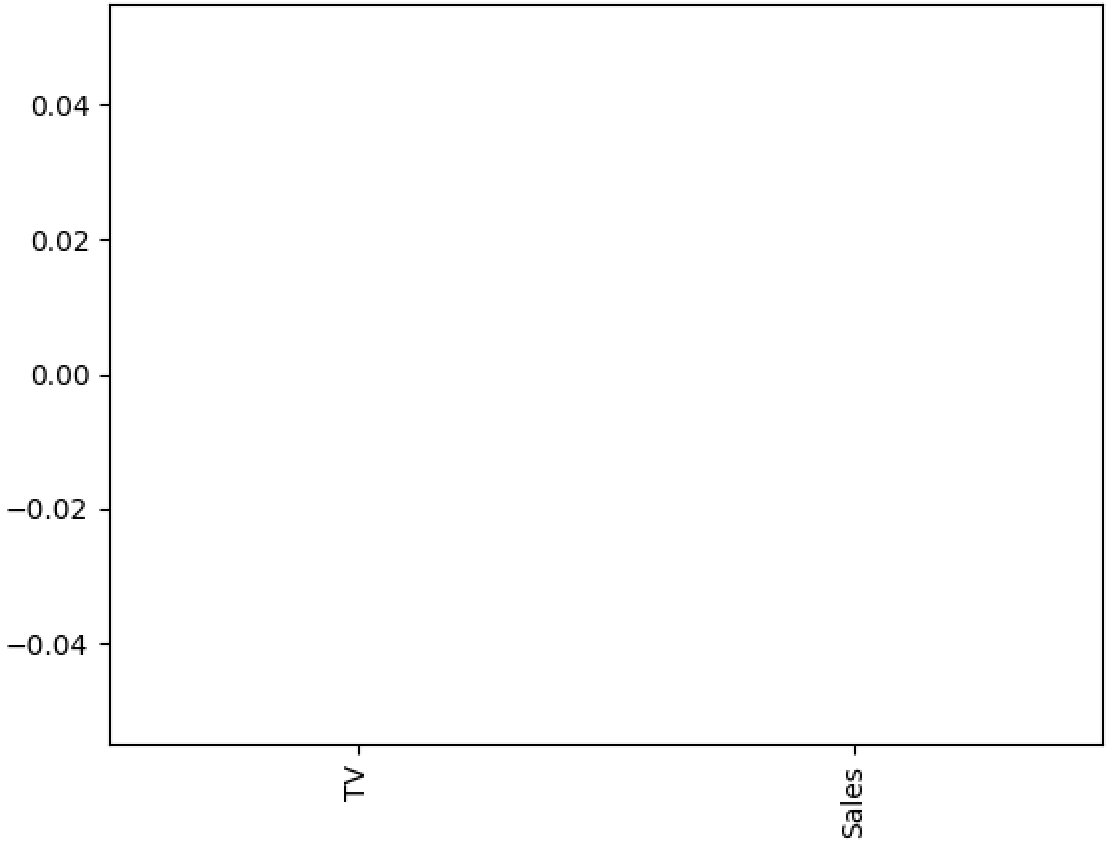
```

Ridge model\:

The train score for lasso model is 0.990287139194161
The train score for lasso model is 0.9844266285141221
```

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

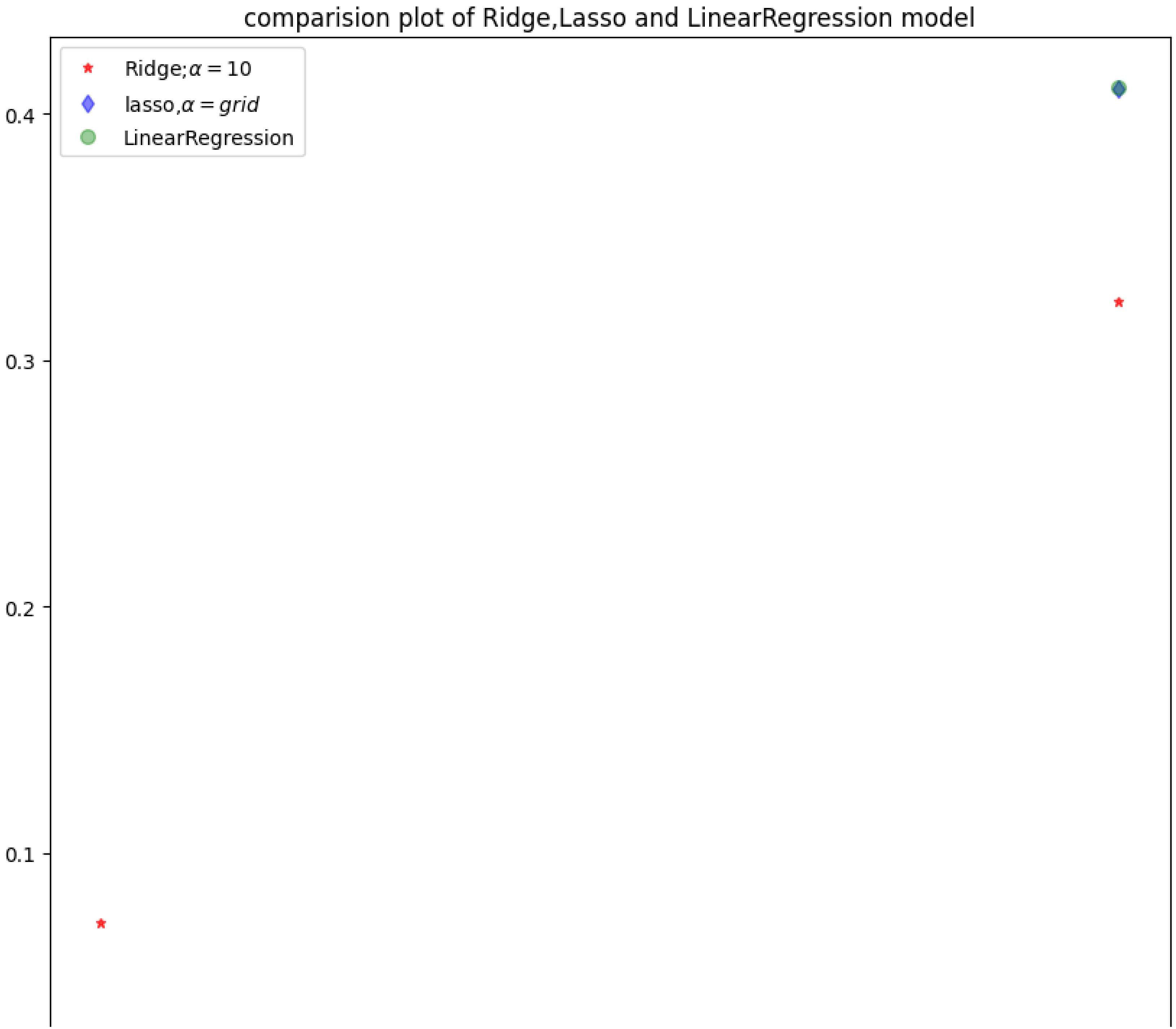
Out[33]:
```
<Axes: >
```



In [34]:
```
from sklearn.linear_model import LassoCV
lasso_CV=LassoCV(alphas=[0.0001,0.001,0.01,0.1,1,10]).fit(X_train,y_train)
print("The train score for lasso model is{}".format(lasso_CV.score(X_train,y_train)))
print("The test score for lasso model is{}".format(lasso_CV.score(X_test,y_test)))
```

```
The train score for lasso model is0.9999999343798134
The test score for lasso model is0.9999999152638072
```

In [35]:
```
plt.figure(figsize=(10,10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5,color='red',label=r'Ridge;$\alpha=
plt.plot(features,lasso_CV.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color='blue',label=r'lasso,$\alpha
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color='green',label='LinearRegression')
plt.xticks(rotation=90)
plt.legend()
plt.title("comparision plot of Ridge,Lasso and LinearRegression model")
plt.show()
```


comparision plot of Ridge,Lasso and LinearRegression model
Ridge;$\alpha = 10$
lasso,$\alpha = grid$
LinearRegression
0.4
0.3
0.2
0.1

In [36]:

```
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 is0.9999999999976281
The test score for ridge model is0.9999999999962489
```

In [48]:

```
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 the tset set",mean_squared_error)
```

```
[0.00417976 0.        ]
2.0263839193110043
mean Squared Error on the tset set 0.5538818050142152
```

In [ ]:

```

```