

Date:6-6-23__ADVERTISING USING LINEAR ,LASSO,RIDGE REGRESSION

In [13]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics
from sklearn import preprocessing,svm
```

In [14]:

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

Out[14]:

	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 [15]:

```
df.head()
```

Out[15]:

	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 [16]:

```
df.tail()
```

Out[16]:

	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 [17]:

```
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 [18]:

```
df.describe()
```

Out[18]:

	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 [19]:

```
df.shape
```

Out[19]:

(200, 4)

In [20]:

```
df.columns
```

Out[20]:

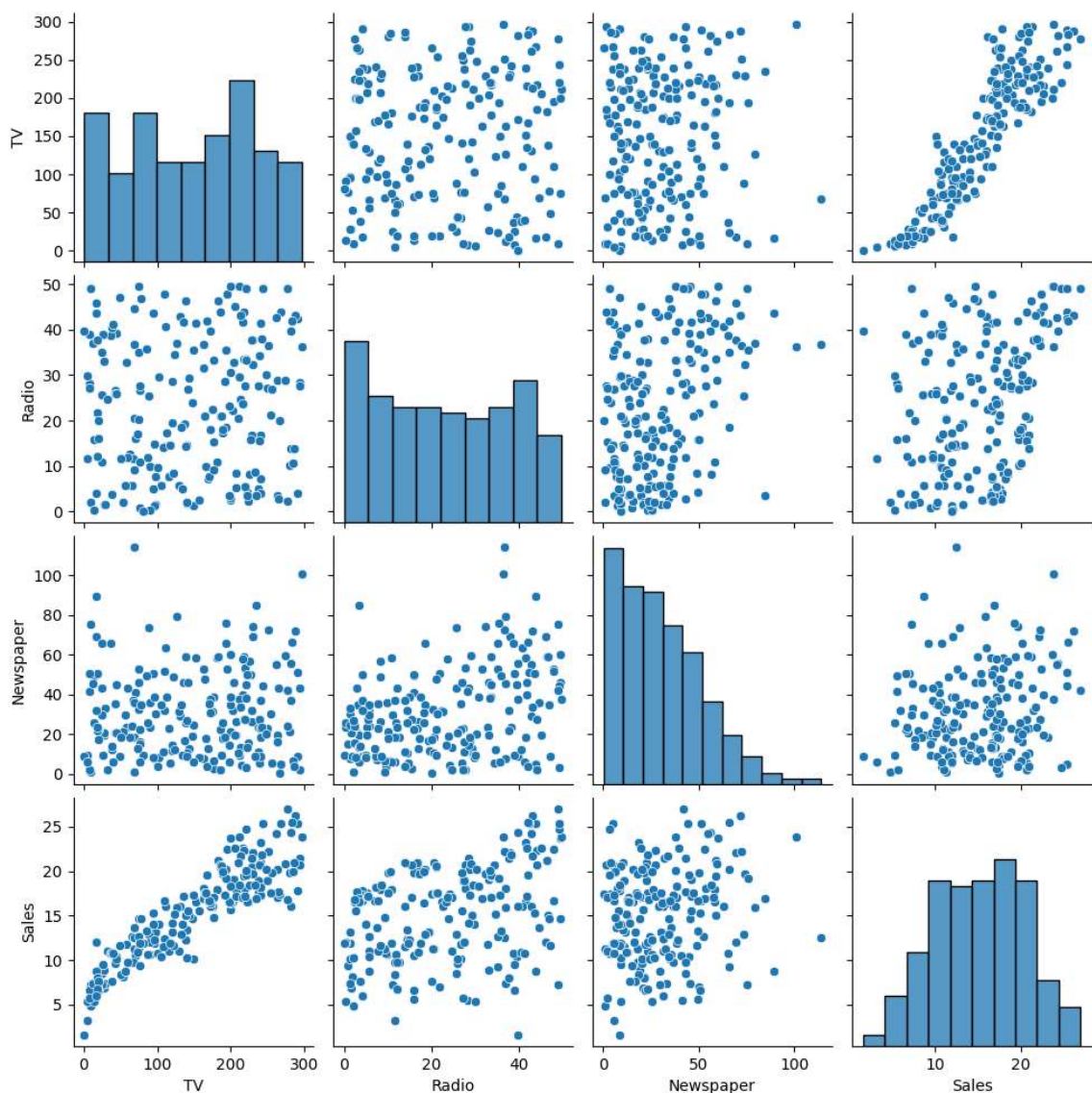
Index(['TV', 'Radio', 'Newspaper', 'Sales'], dtype='object')

In [21]:

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

Out[21]:

<seaborn.axisgrid.PairGrid at 0x1f65392a690>

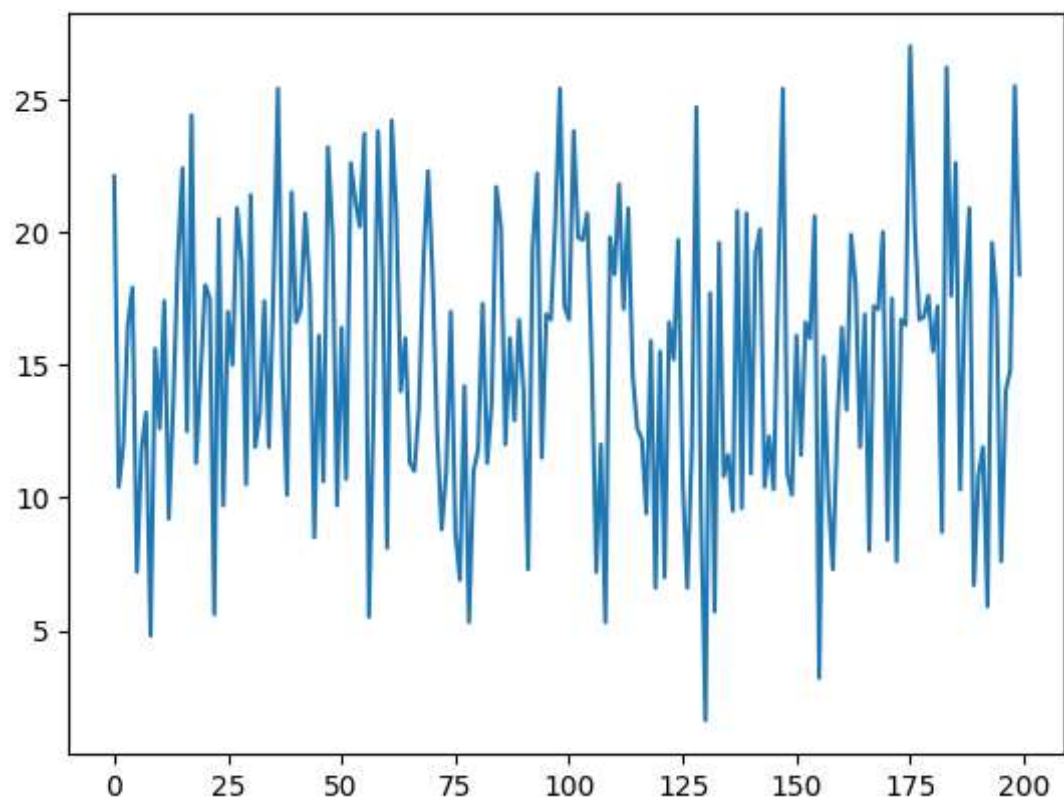


In [22]:

```
plt.plot(df['Sales'])
```

Out[22]:

[<matplotlib.lines.Line2D at 0x1f65740dd10>]

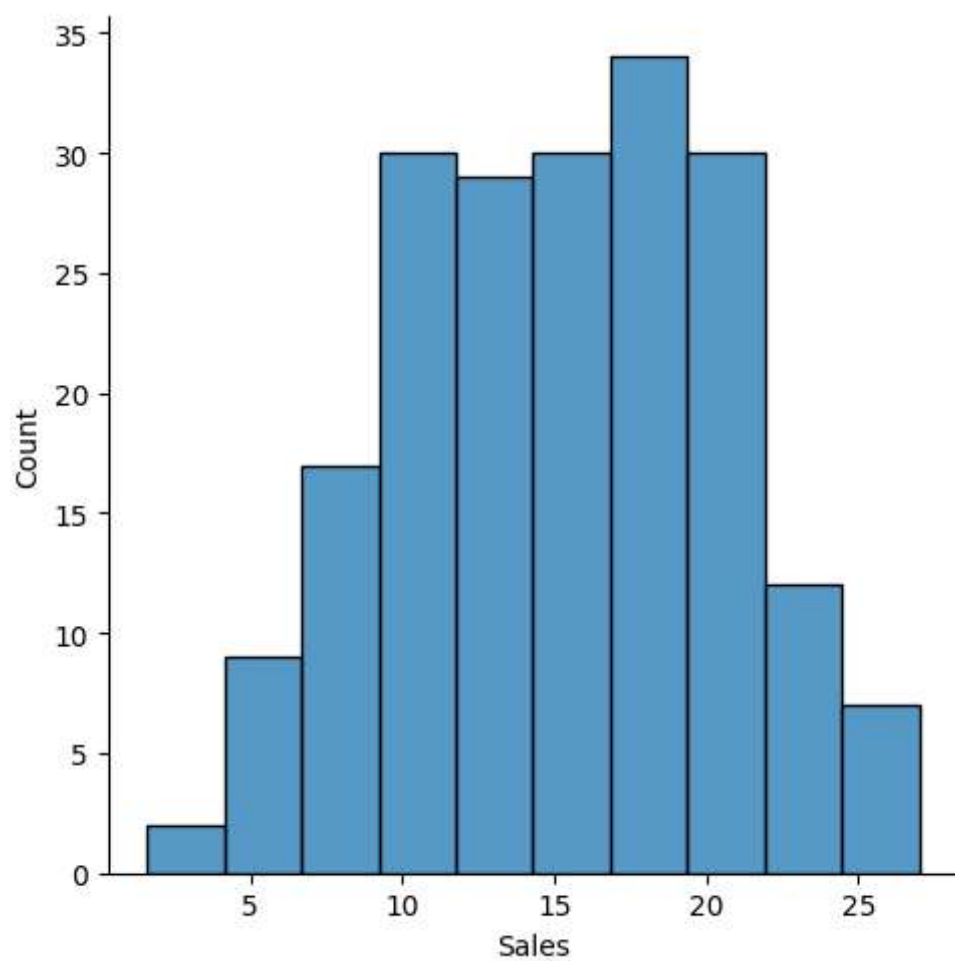


In [23]:

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

Out[23]:

<seaborn.axisgrid.FacetGrid at 0x1f657bef450>

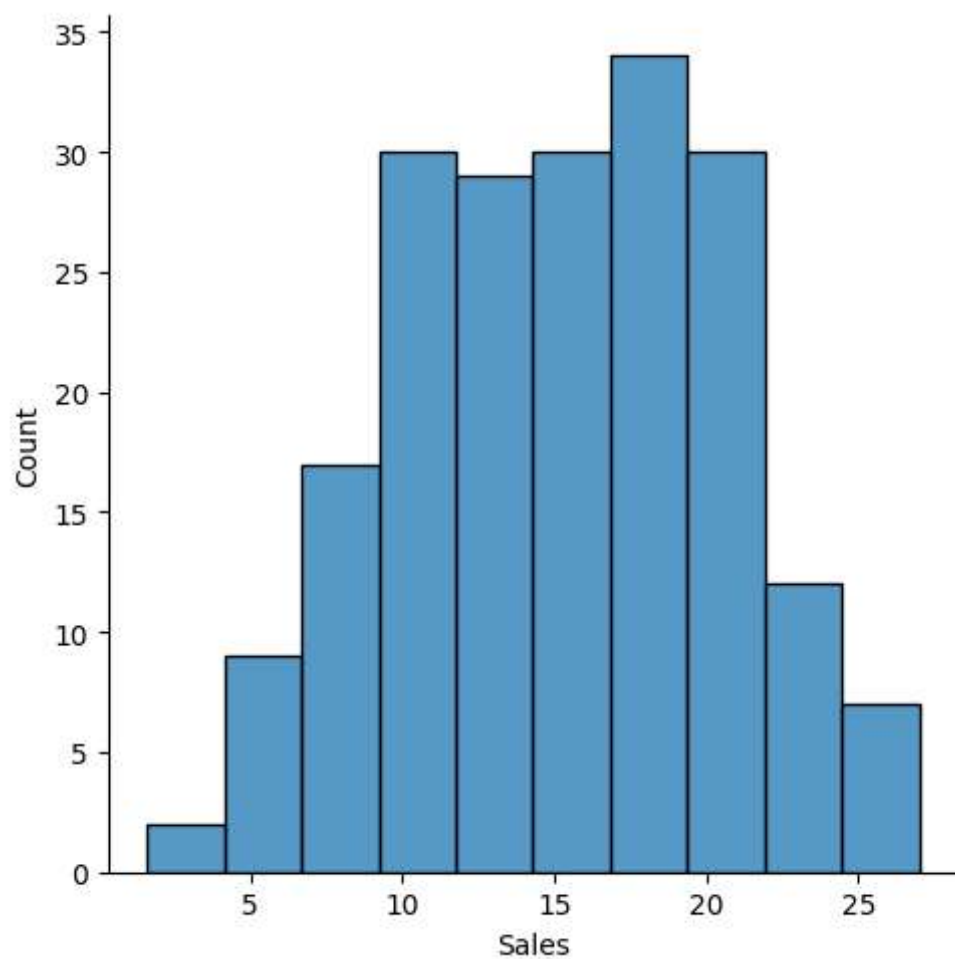


In [24]:

```
df1=df.iloc[:,0:4]  
sns.displot(df1['Sales'])
```

Out[24]:

<seaborn.axisgrid.FacetGrid at 0x1f6573a0610>

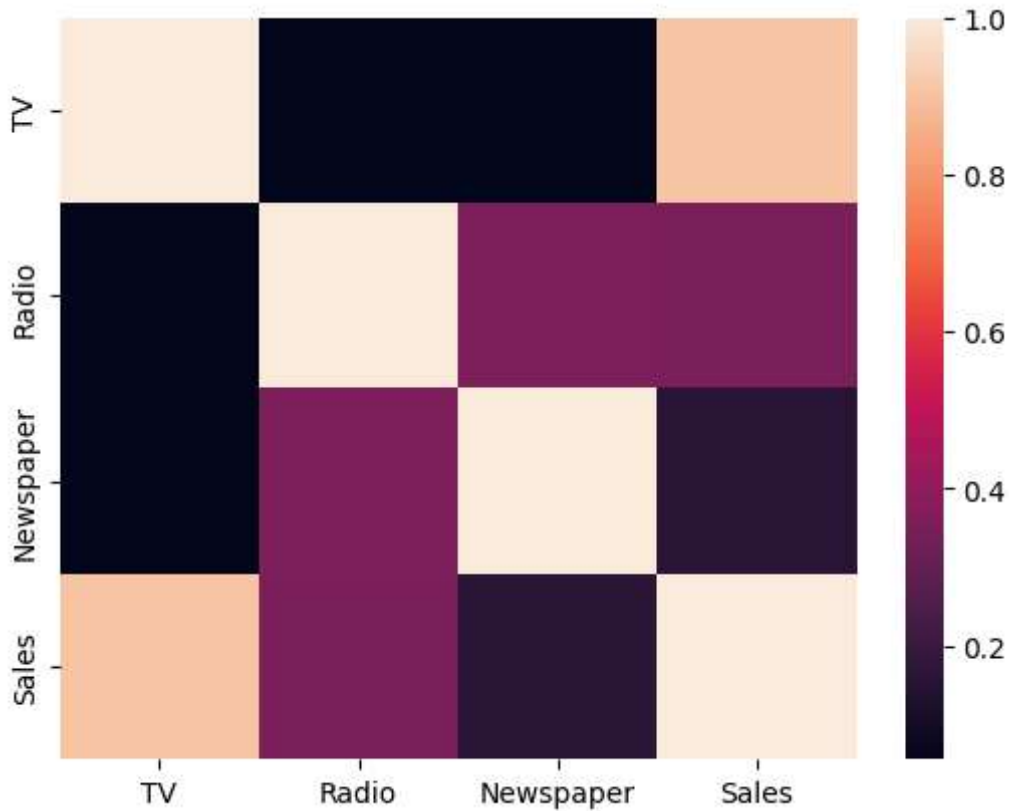


In [25]:

```
addf=df[['TV','Radio','Newspaper','Sales']]
sns.heatmap(addf.corr())
```

Out[25]:

<Axes: >



In [26]:

```
x=addf[['TV','Radio','Newspaper']]
y=df['Sales']
```

In [27]:

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

4.681232151484295

In [28]:

```
coeff_df=pd.DataFrame(lm.coef_,x.columns,columns=['coefficient'])  
coeff_df
```

Out[28]:

	coefficient
TV	0.054930
Radio	0.109558
Newspaper	-0.006194

In [29]:

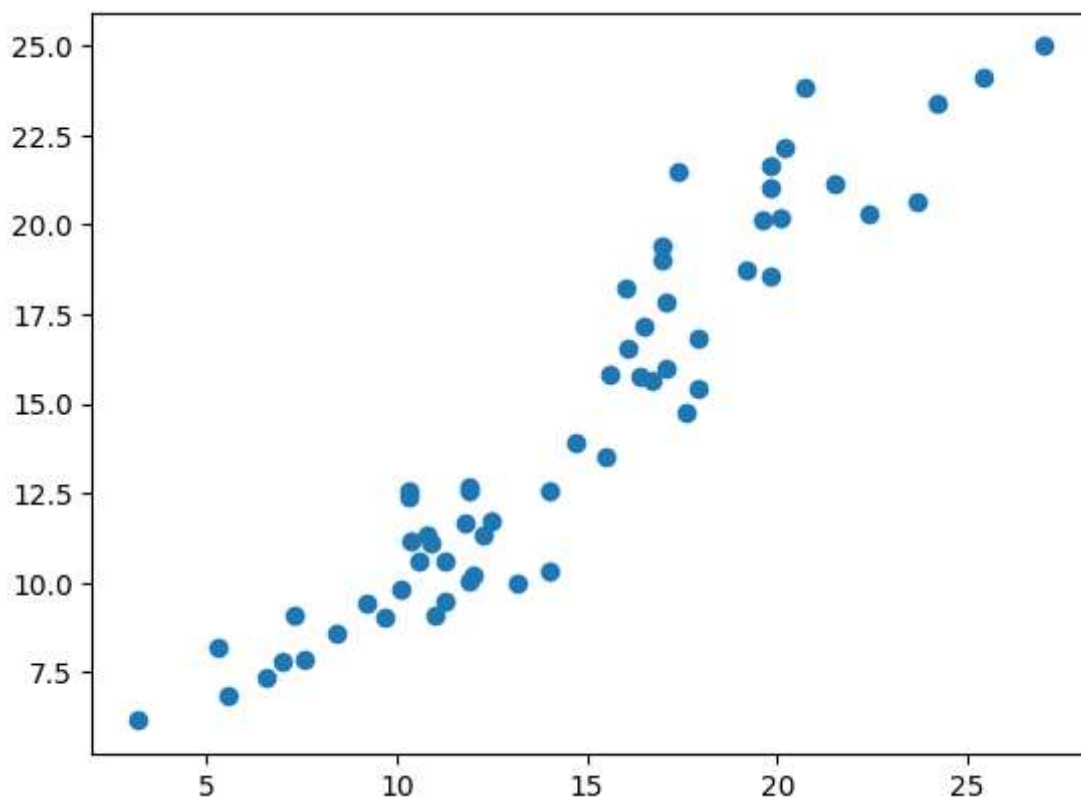
```
predictions=lm.predict(x_test)
```

In [30]:

```
plt.scatter(y_test,predictions)
```

Out[30]:

<matplotlib.collections.PathCollection at 0x1f657a85350>

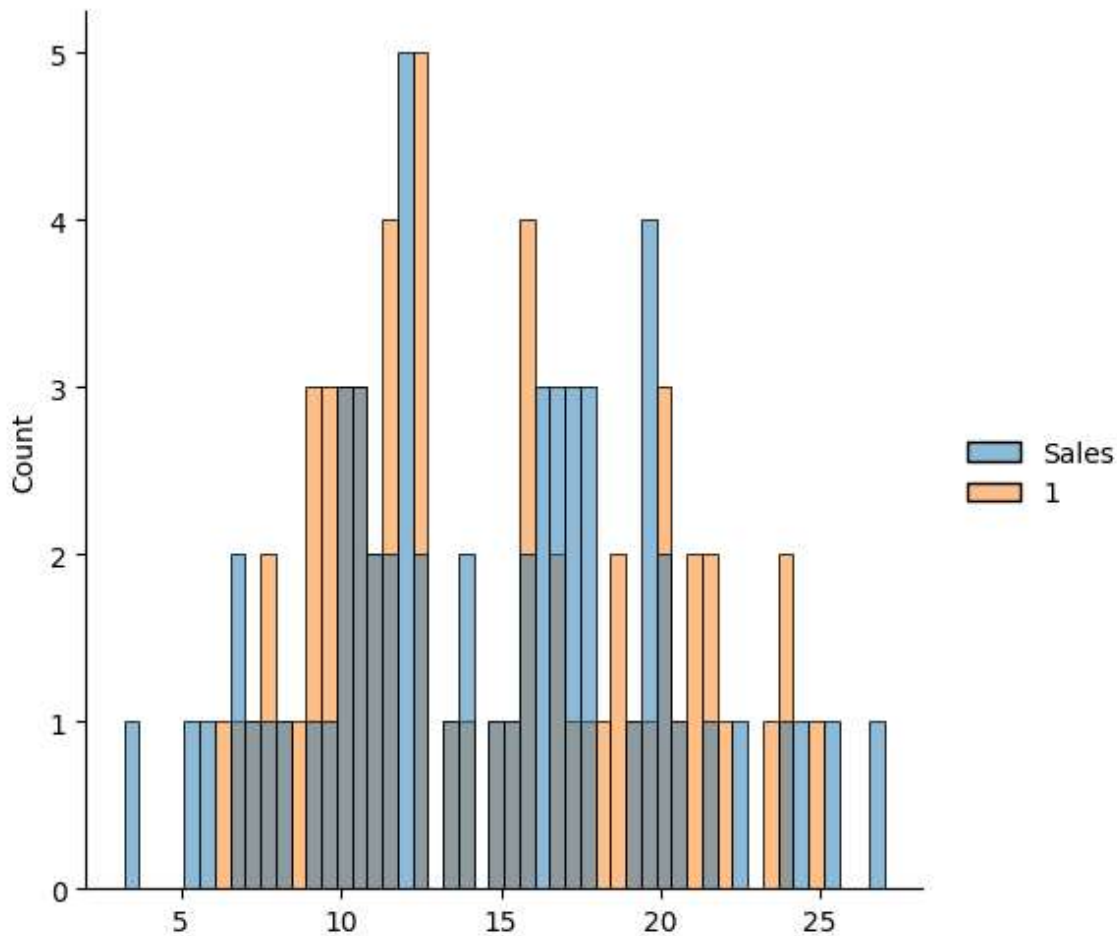


In [31]:

```
sns.displot((y_test,predictions),bins=50)#without semicolon
```

Out[31]:

<seaborn.axisgrid.FacetGrid at 0x1f657c798d0>



In [32]:

```
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_squared_error(y_test,predictions)))
```

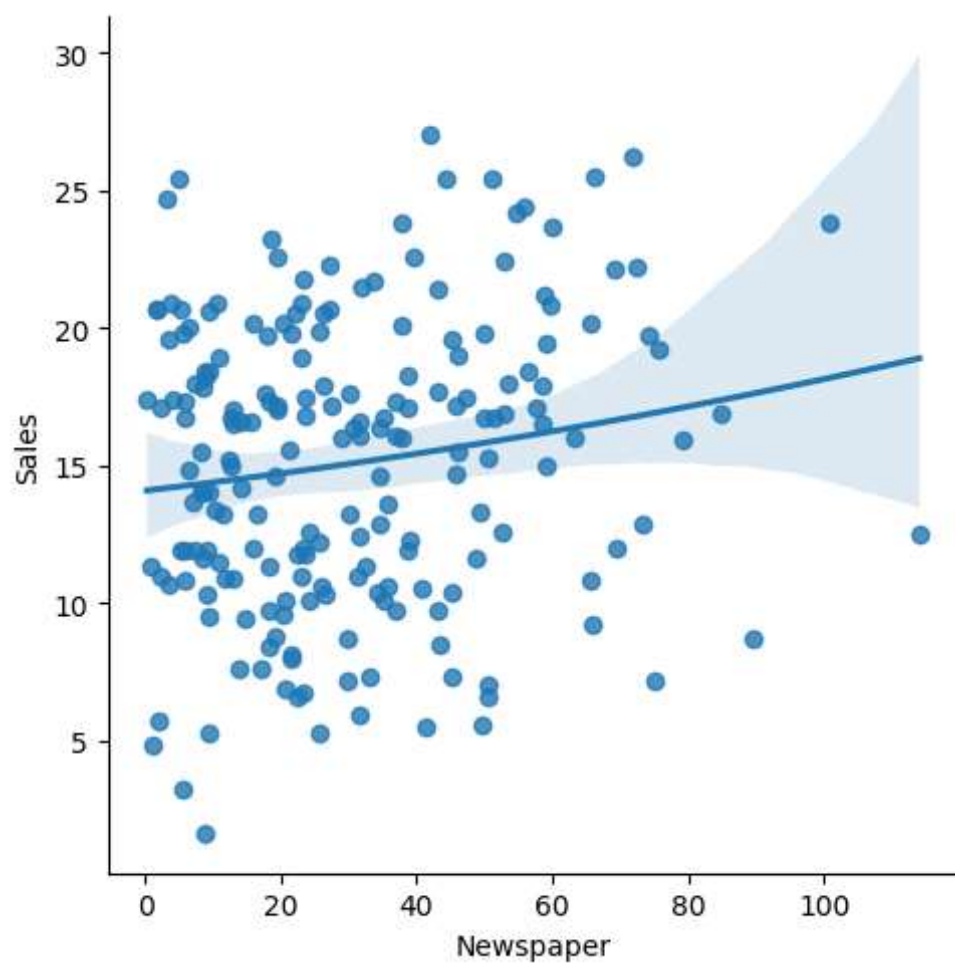
MAE: 1.3731200698367851
MSE: 2.8685706338964967
RMSE: 1.6936855180040056

In [33]:

```
sns.lmplot(x='Newspaper',y="Sales",data=df,order=2)
```

Out[33]:

<seaborn.axisgrid.FacetGrid at 0x1f657964390>

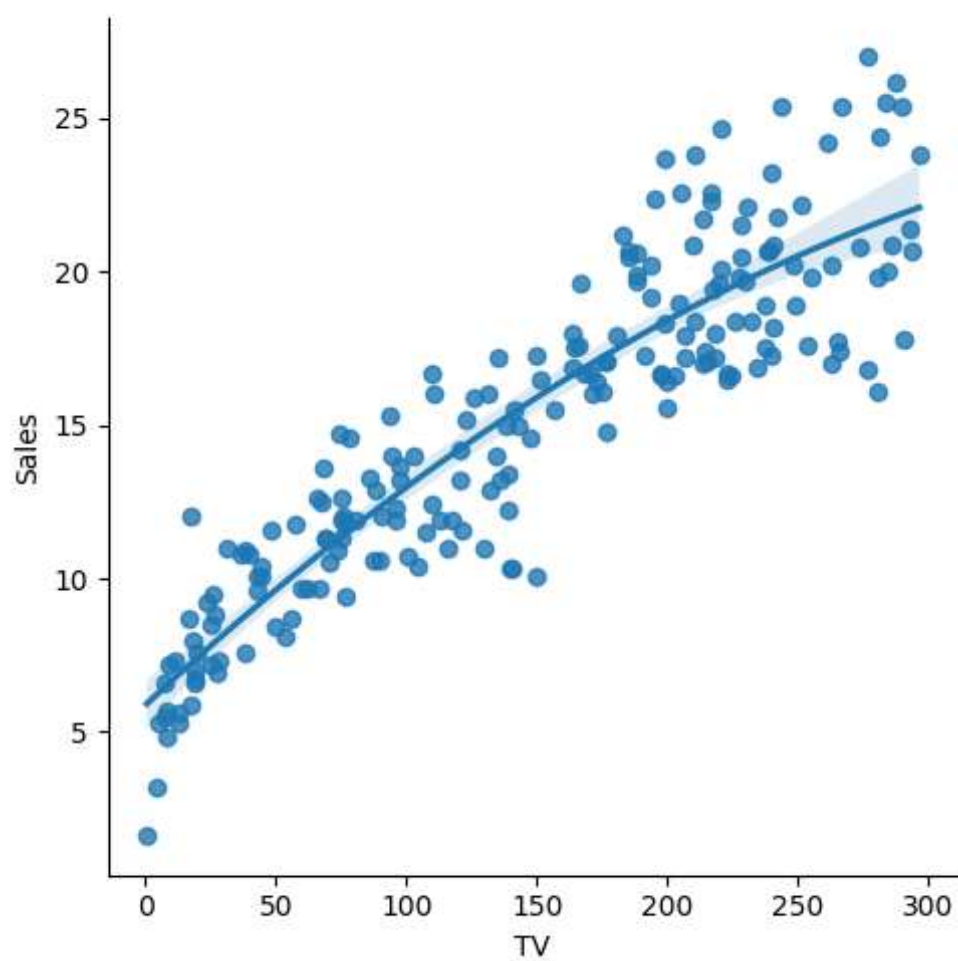


In [34]:

```
sns.lmplot(x='TV',y="Sales",data=df,order=2)
```

Out[34]:

<seaborn.axisgrid.FacetGrid at 0x1f65adeead0>

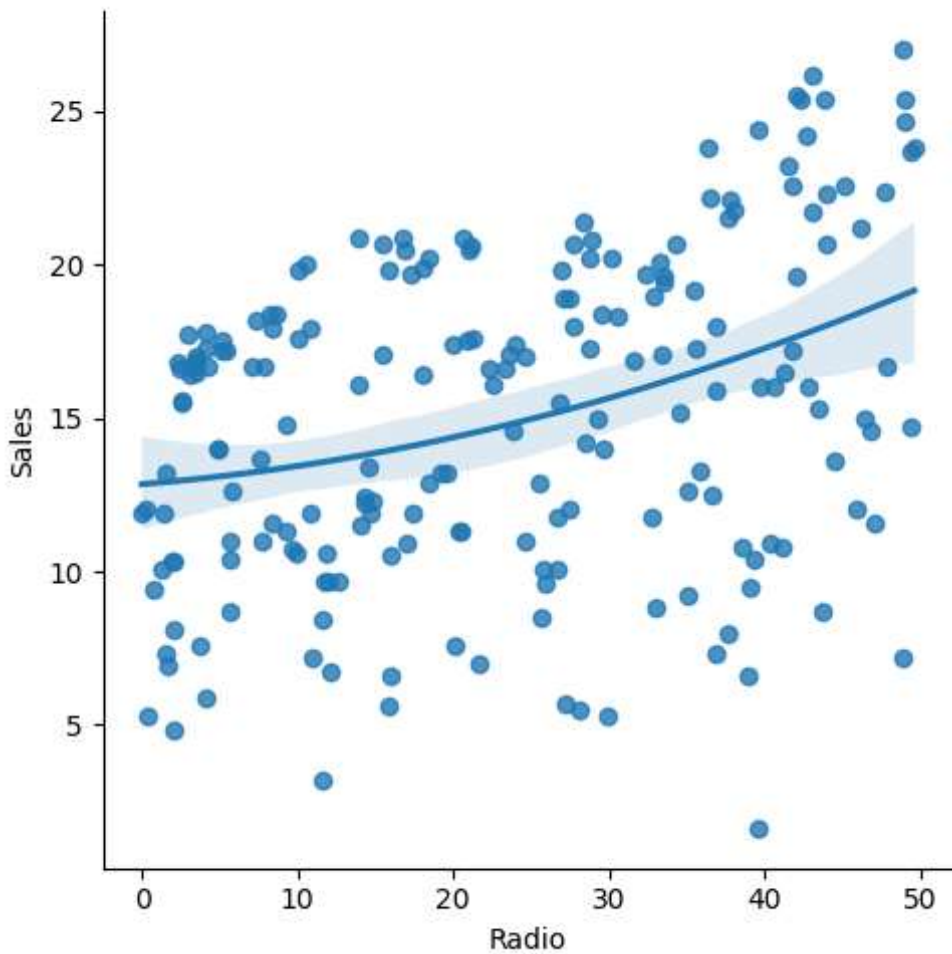


In [35]:

```
sns.lmplot(x='Radio',y="Sales",data=df,order=2)
```

Out[35]:

<seaborn.axisgrid.FacetGrid at 0x1f65ae66390>



In [36]:

```
df.fillna(method='ffill',inplace=True)
```

In [37]:

```
regr=LinearRegression()
```

In [38]:

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

In [39]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
regr.fit(x_train,y_train)
regr.fit(x_train,y_train)
```

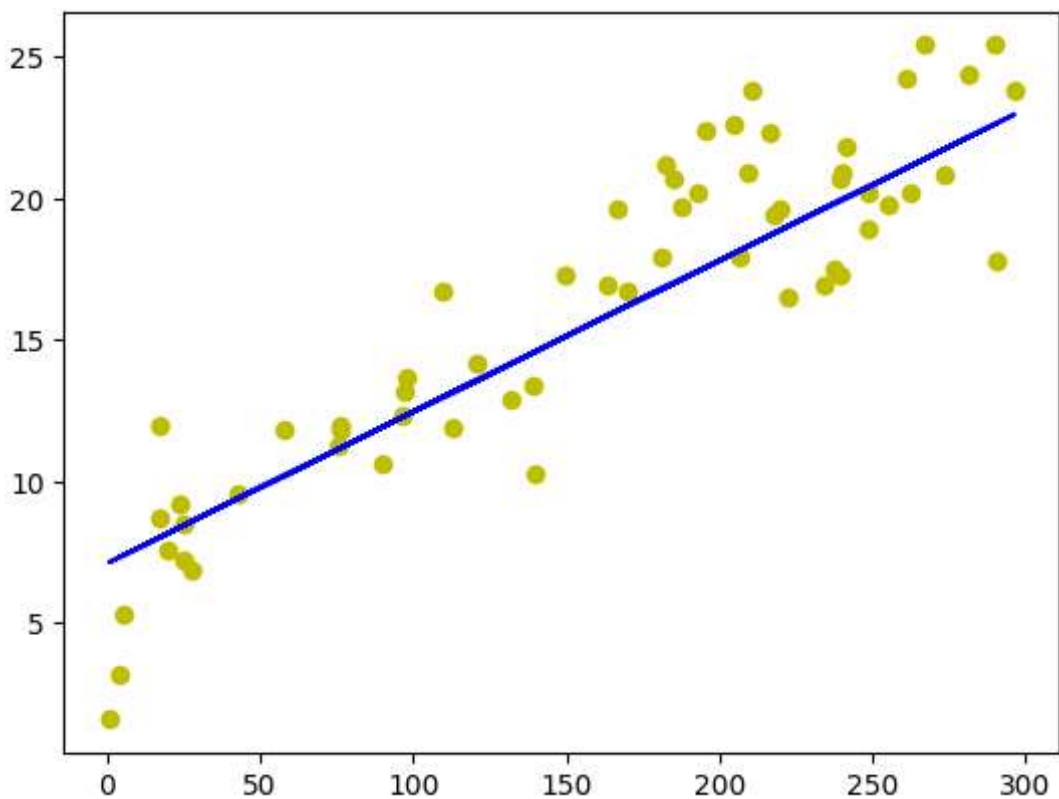
Out[39]:

▼ LinearRegression

LinearRegression()

In [40]:

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

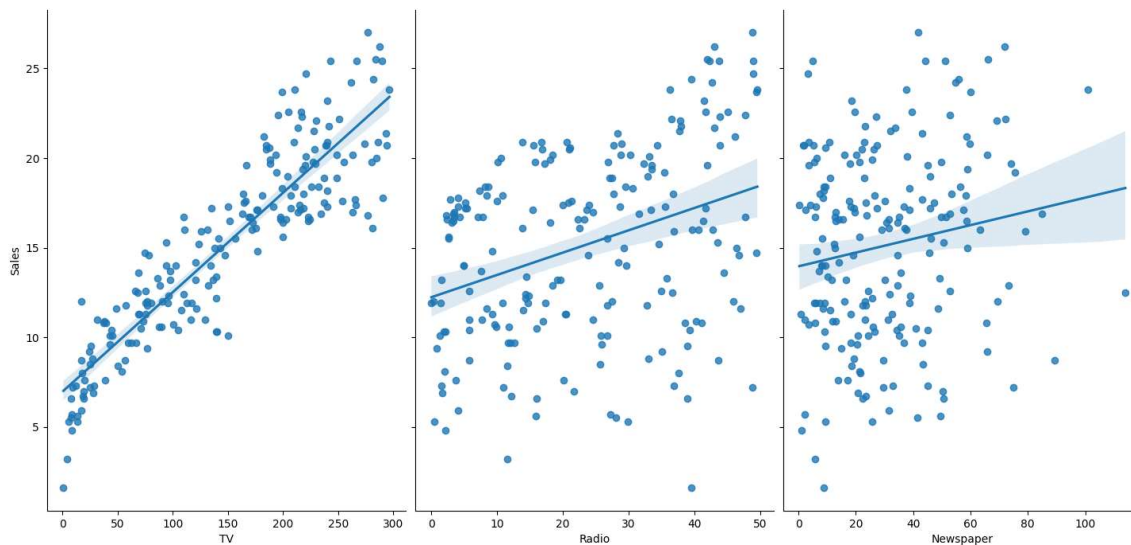


In [41]:

```
sns.pairplot(df,x_vars=['TV','Radio','Newspaper'],y_vars='Sales',height=7,aspect=0.7,kin
```

Out[41]:

```
<seaborn.axisgrid.PairGrid at 0x1f6579b39d0>
```



In [42]:

```
#accuracy
regr=LinearRegression()
regr.fit(x_train,y_train)
regr.fit(x_train,y_train)
print(regr.score(x_test,y_test))
```

```
0.8115360666424226
```

In [43]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge, RidgeCV, Lasso
from sklearn.preprocessing import StandardScaler
```

In [44]:

```
data=pd.read_csv(r"C:\Users\prajapath Arjun\Downloads\Advertising.csv")
data
```

Out[44]:

	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 [45]:

```
data.head()
```

Out[45]:

	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 [46]:

```
data.tail()
```

Out[46]:

	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 [47]:

```
plt.figure(figsize = (10, 10))  
sns.heatmap(data.corr(), annot = True)
```

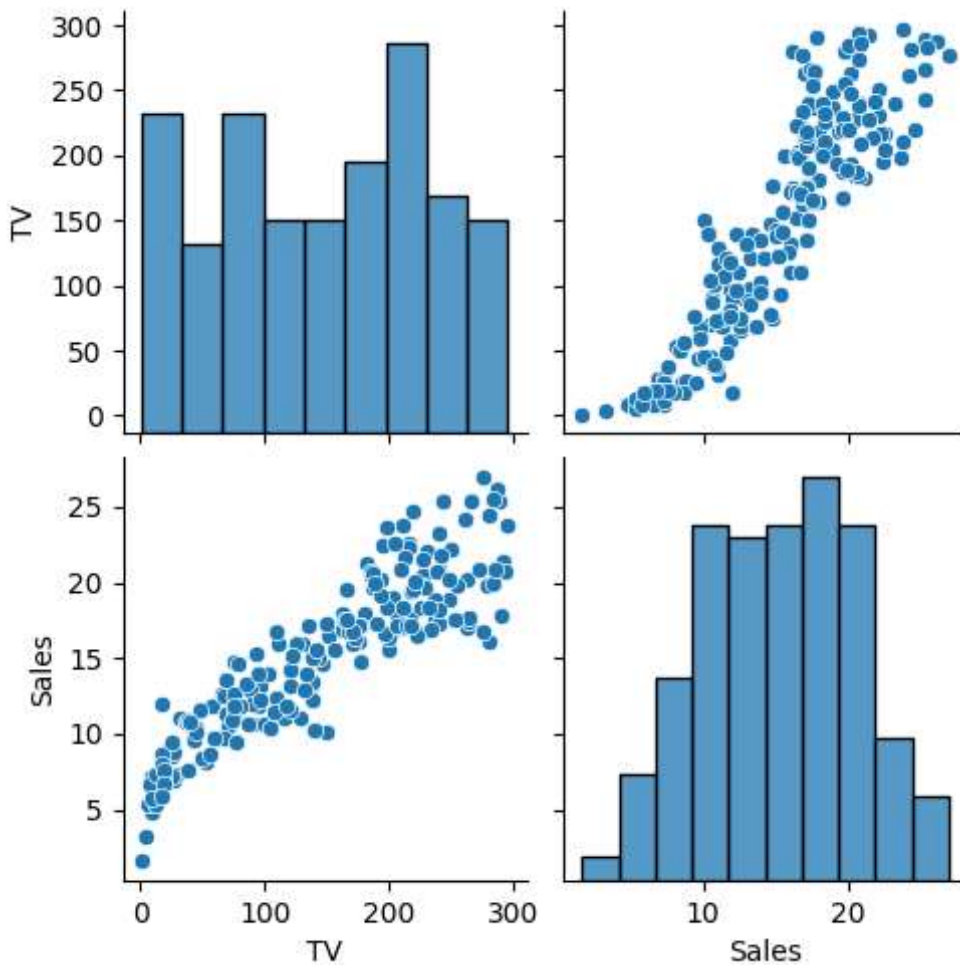
Out[47]:

<Axes: >



In [48]:

```
data.drop(columns = ["Radio", "Newspaper"], inplace = True)
#pairplot
sns.pairplot(data)
data.Sales = np.log(data.Sales)
```



In [49]:

```
features = data.columns[0:2]
target = data.columns[-1]
#X and y values
X = data[features].values
y = data[target].values
#split
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 [50]:

```
#Model
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 [51]:

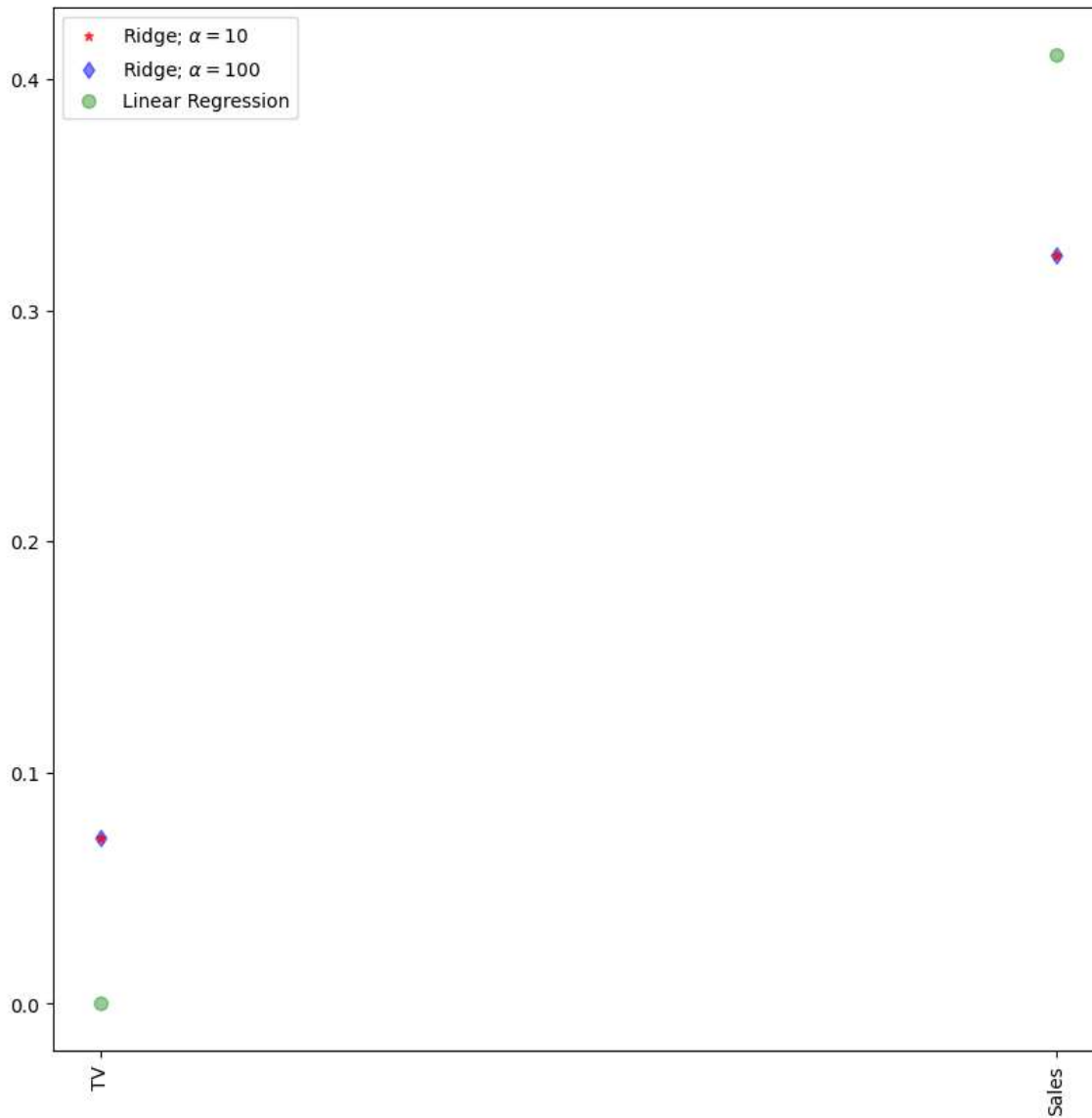
```
#Ridge Regression Model
ridgeReg = Ridge(alpha=10)
ridgeReg.fit(X_train,y_train)
#train and test scorefor ridge regression
train_score_ridge = ridgeReg.score(X_train, y_train)
test_score_ridge = ridgeReg.score(X_test, y_test)
print("\nRidge Model:\n")
print("The train score for ridge model is {}".format(train_score_ridge))
print("The test score for ridge model is {}".format(test_score_ridge))
```

Ridge Model:

The train score for ridge model is 0.9902871391941609
The test score for ridge model is 0.984426628514122

In [52]:

```
plt.figure(figsize = (10, 10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5,color='red',
plt.plot(ridgeReg.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color='blue',
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color='green')
plt.xticks(rotation = 90)
plt.legend()
plt.show()
```



In [53]:

```
#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

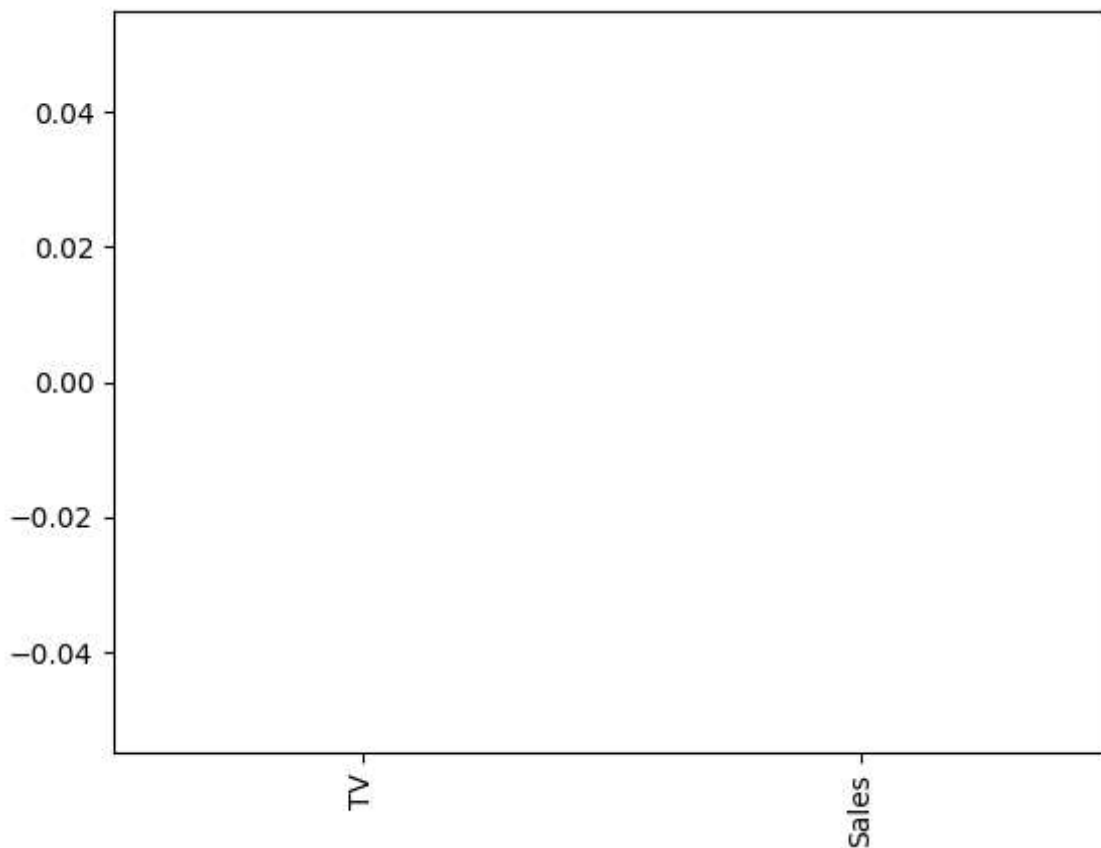
The test score for ls model is -0.0042092253233847465

In [54]:

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

Out[54]:

<Axes: >

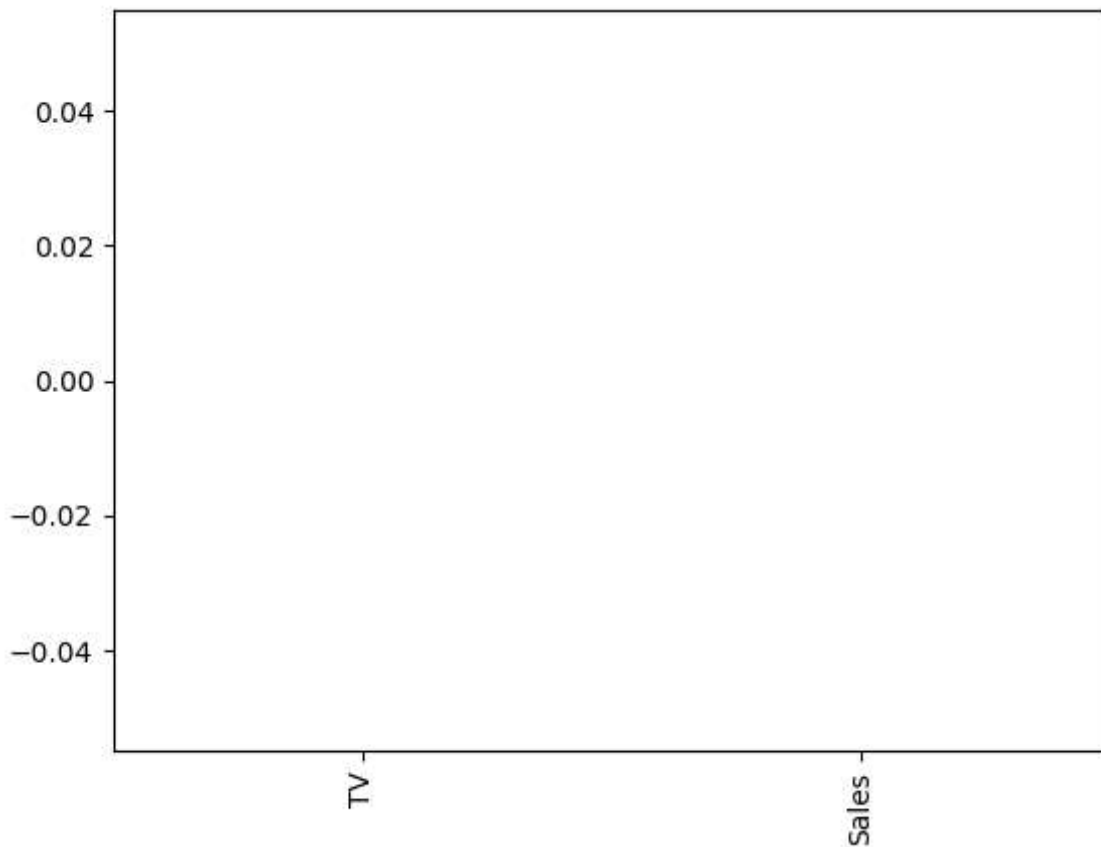


In [55]:

```
pd.Series(lasso.coef_, features).sort_values(ascending = True).plot(kind = "bar")  
#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))
```

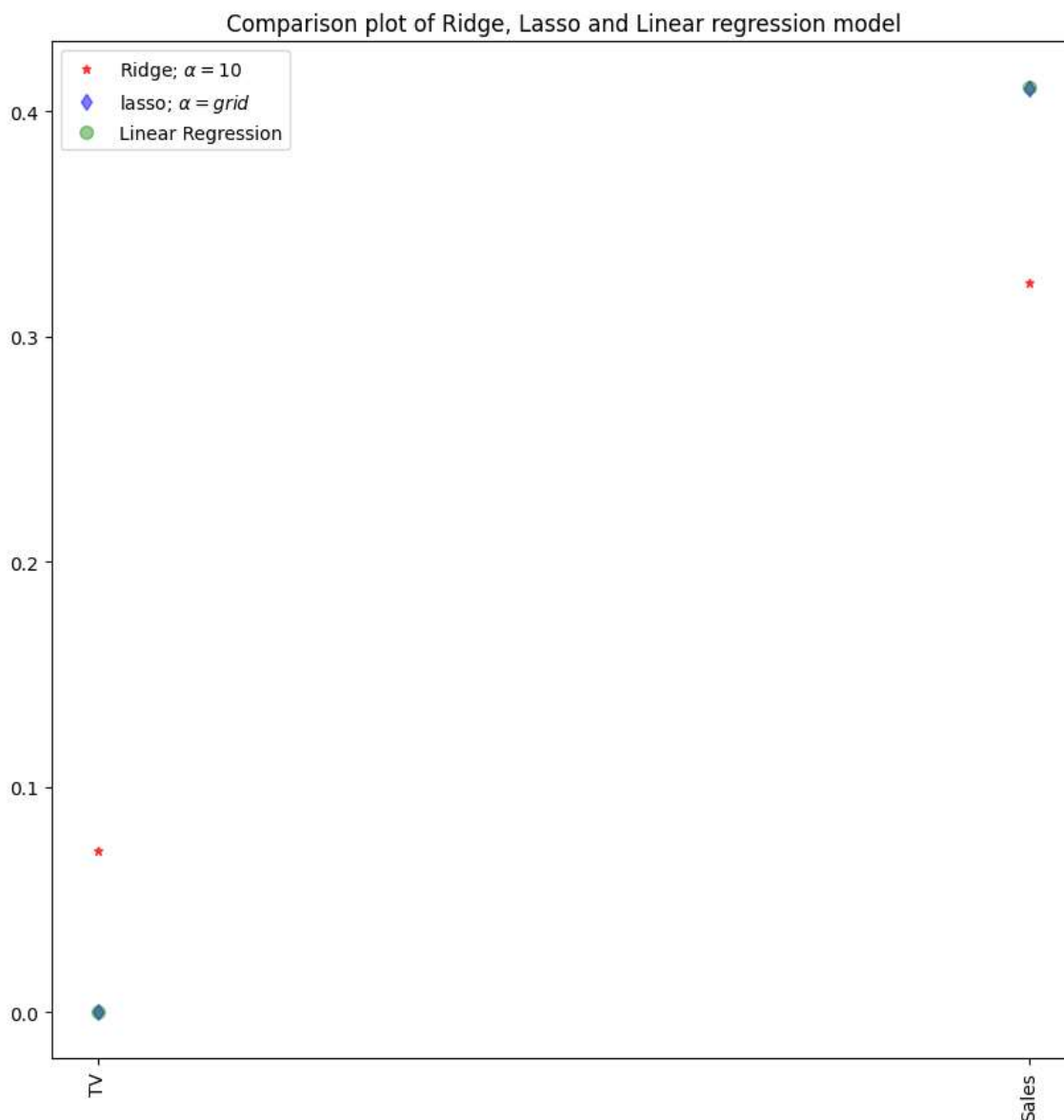
0.9999999343798134

0.9999999152638072



In [56]:

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



In [57]:

```
#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 0.999999999997627

The train score for ridge model is 0.9999999999962466