In [1]:

```python
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.linear_model import Lasso
from sklearn.linear_model import Ridge
```

In [2]:

```python
da=pd.read_csv(r"C:\Users\91955\Downloads\Advertising.csv")
da
```

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]:

```python
da.head()
```

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

In [4]:

```
1  da.tail()
```

Out[4]:

	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 [5]:

```
1  da.describe()
```

Out[5]:

	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 [6]:

```
1  da.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 [7]:

```
1  da.isna().any()
```

Out[7]:

```
TV           False
Radio        False
Newspaper    False
Sales        False
dtype: bool
```

In [8]:

```
1  da.shape
```

Out[8]:

```
(200, 4)
```

In [9]:

```
1  da.isnull().sum()
```

Out[9]:

```
TV           0
Radio        0
Newspaper    0
Sales        0
dtype: int64
```

In [10]:

```
1  plt.figure(figsize = (10, 10))
2  sns.heatmap(da.corr(), annot = True)
```
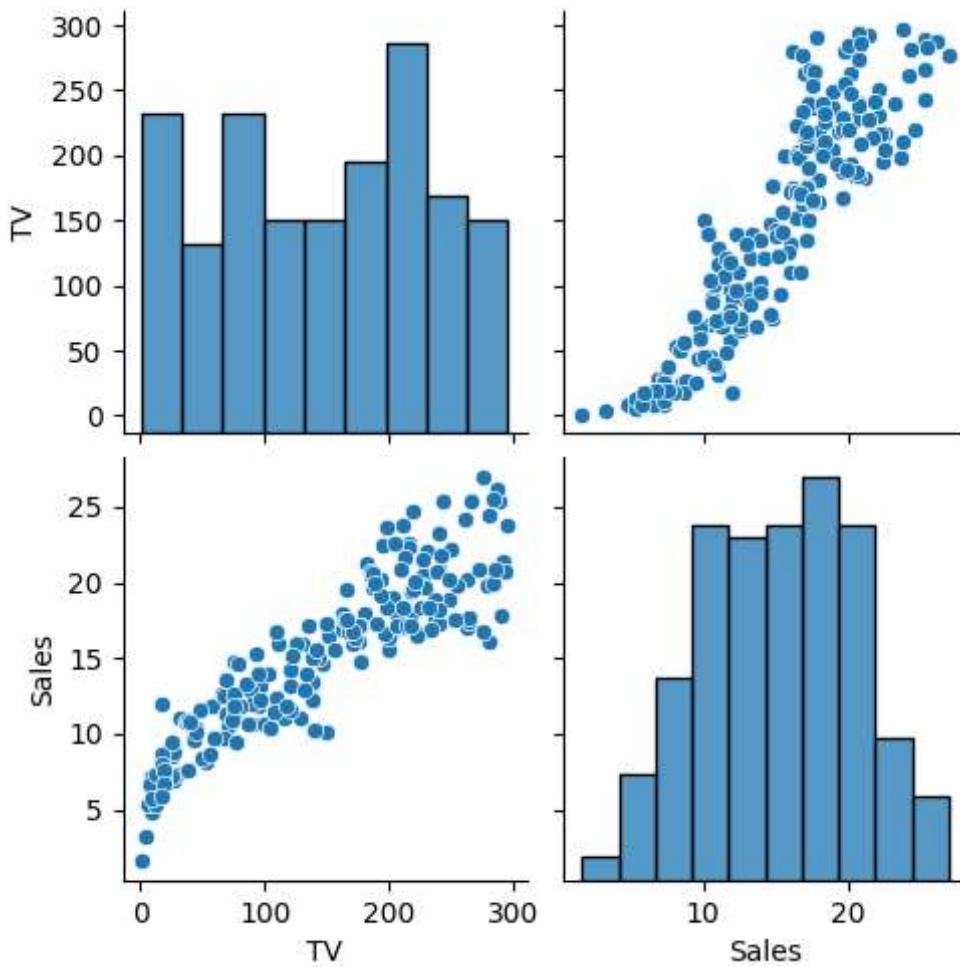
Out[10]:

<Axes: >

In [11]:

```
1  da.drop(columns = ['Radio','Newspaper'], inplace = True)
2  #pairplot
3  sns.pairplot(da)
4  da.Sales = np.log(da.Sales)
```


In [12]:

```
1  features = da.columns[0:2]
2  target = da.columns[-1]
3  #X and y values
4  X = da[features].values
5  y = da[target].values
6  #splot
7  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_stat
8  print("The dimension of X_train is {}".format(X_train.shape))
9  print("The dimension of X_test is {}".format(X_test.shape))
10 #Scale features
11 from sklearn.preprocessing import StandardScaler
12 scaler = StandardScaler()
13 X_train = scaler.fit_transform(X_train)
14 X_test = scaler.transform(X_test)
```

```
The dimension of X_train is (140, 2)
The dimension of X_test is (60, 2)
```

In [13]:

```python
#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:")
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 [14]:

```python
#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 0.9999999999962467
```

In [15]:

```python
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 0.990287139194161
The test score for ridge model is 0.9844266285141221
```
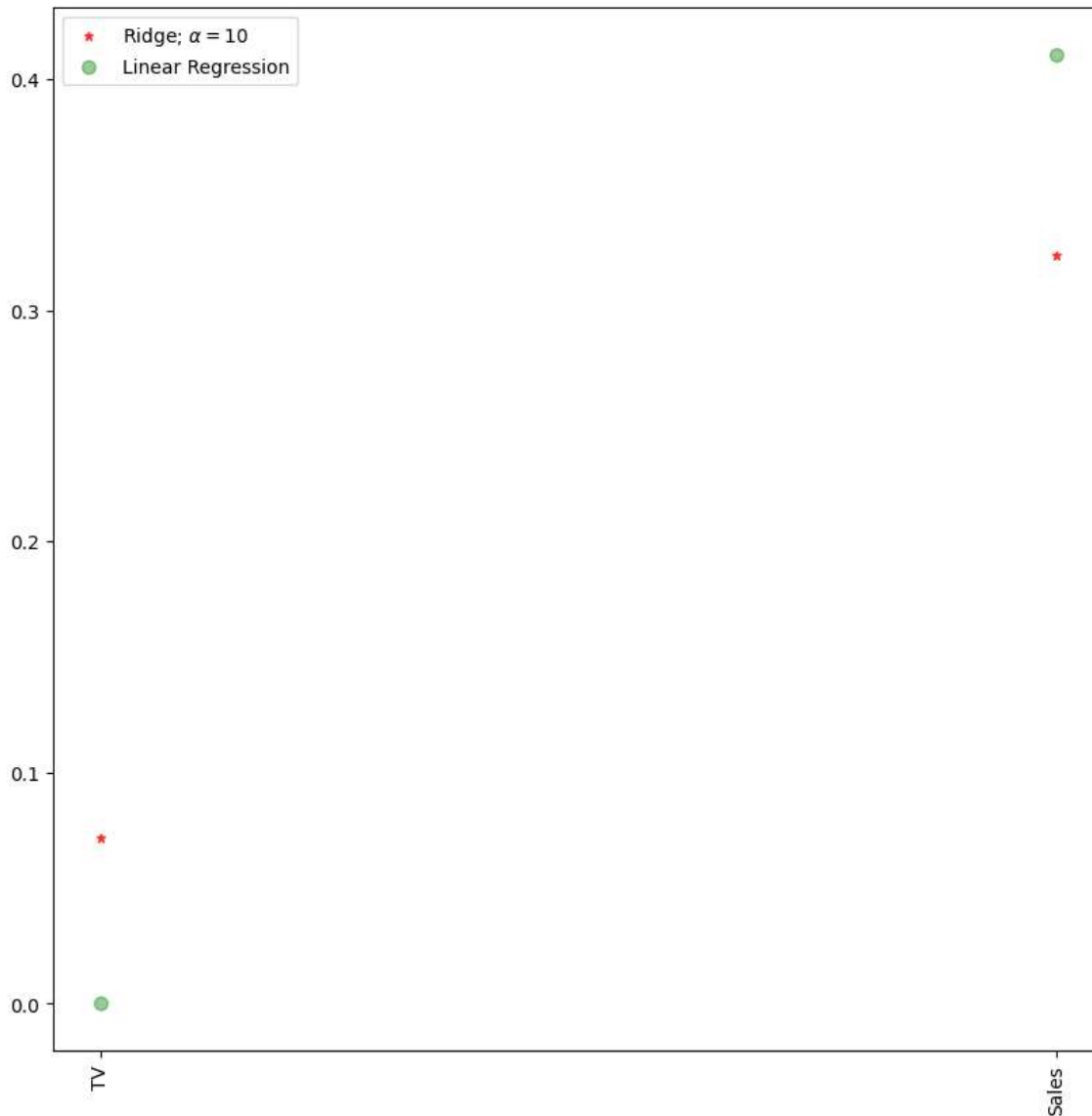
In [16]:

```
1  plt.figure(figsize = (10, 10))
2  plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5,
3
4  plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color=
5  plt.xticks(rotation = 90)
6  plt.legend()
7  plt.show()
```


In [17]:

```
1  #Using the linear CV model
2  from sklearn.linear_model import LassoCV
3  #Lasso Cross validation
4  lasso_cv = LassoCV(alphas = [0.0001, 0.001,0.01, 0.1, 1, 10], random_state=0).fit(X_
5  #score
6  print(lasso_cv.score(X_train, y_train))
7  print(lasso_cv.score(X_test, y_test))
```

```
0.999999343798134
0.999999152638072
```

In [18]:

```python
#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))
```
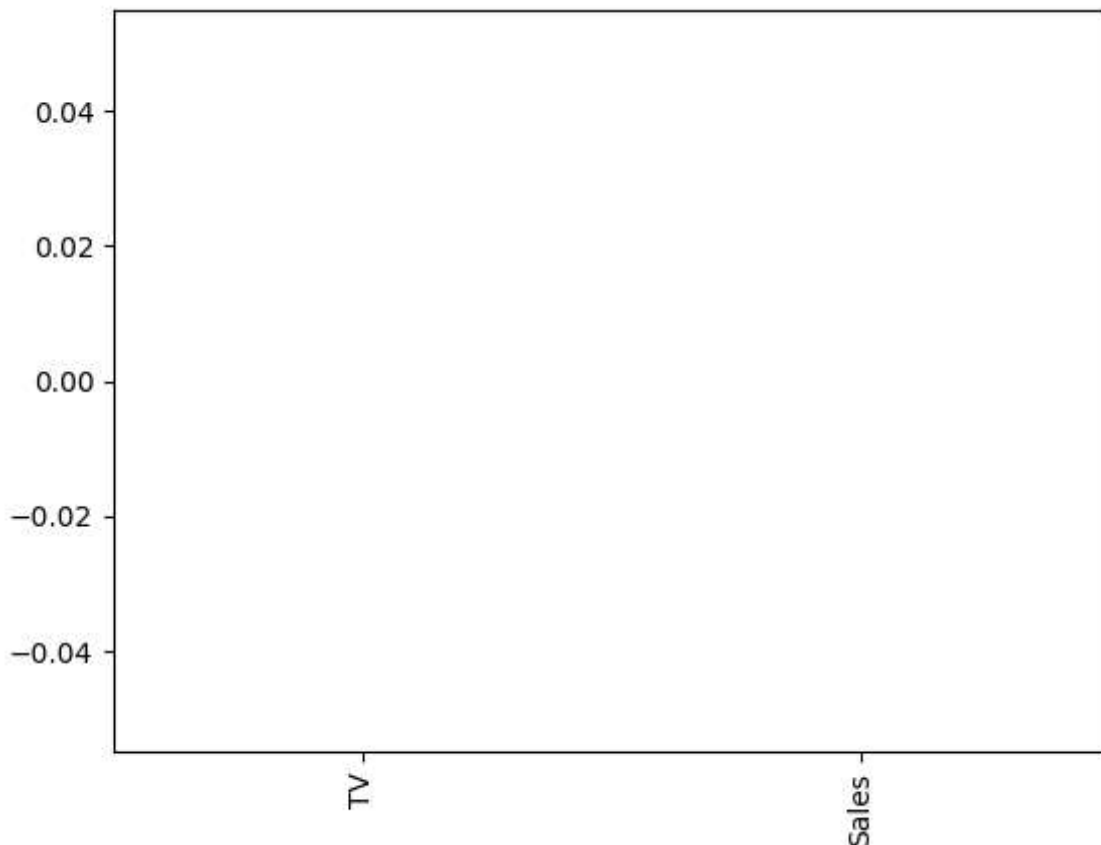
Lasso Model:

The train score for ls model is 0.0
The test score for ls model is -0.0042092253233847465

In [19]:

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

Out[19]:

<Axes: >

In [20]:

```python
#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,
#add plot for lasso regression
plt.plot(lasso_cv.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color='bl
#add plot for linear model
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color=
```

Out[20]:

[<matplotlib.lines.Line2D at 0x1ea95874a30>]
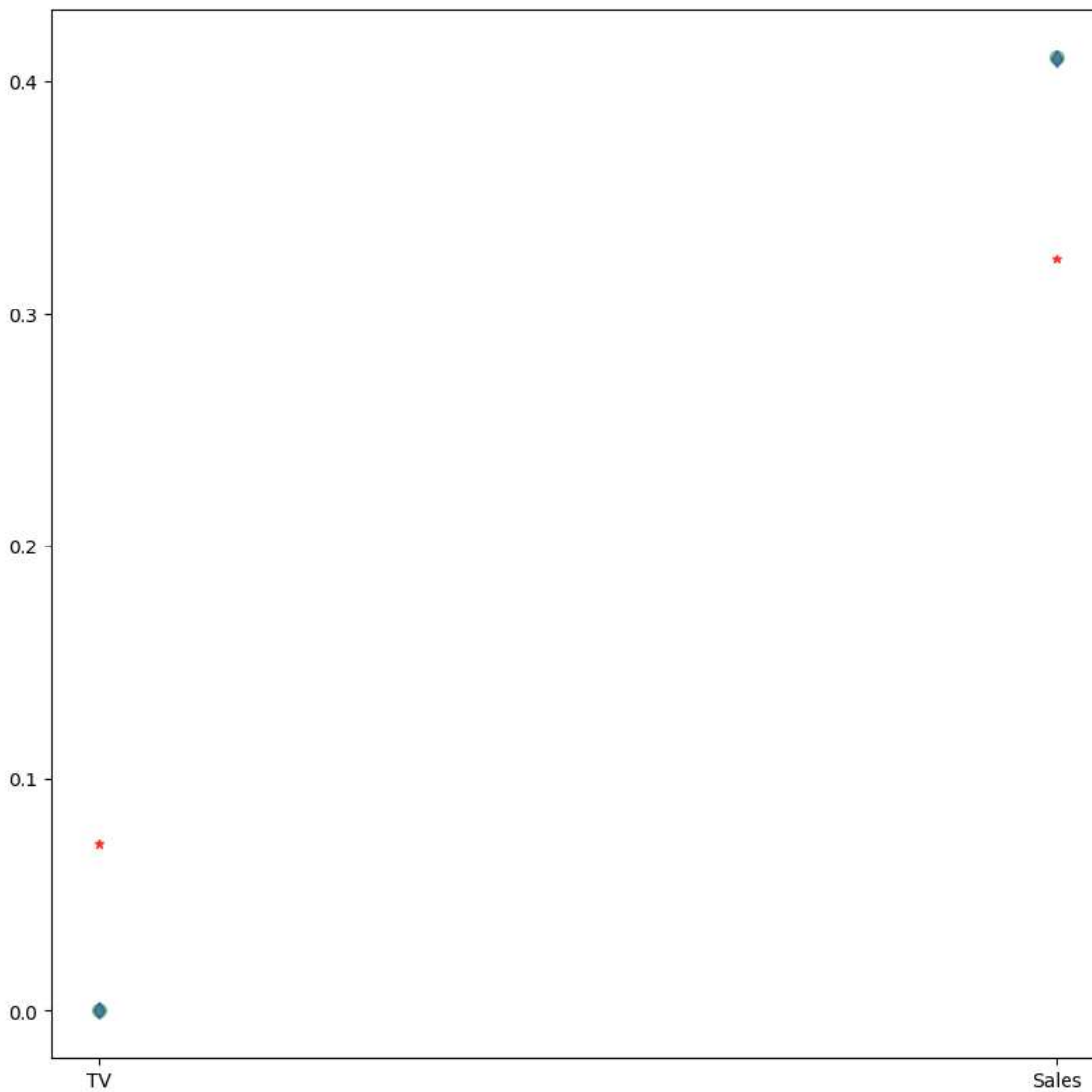
In [23]:

```python
from sklearn.linear_model import ElasticNet
regr=ElasticNet()
regr.fit(X,y)
print(regr.coef_)
print(regr.intercept_)

```

```
[0.00417976 0.          ]
2.026383919311004
```

In [22]:

```python
y_pred_elastic=regr.predict(X_train)
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 0.5538818050142158
```

In [ ]:

```python

```