In [4]:
```
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.linear_model import Ridge,RidgeCV,Lasso
from sklearn.preprocessing import StandardScaler
```

In [5]:
```
data=pd.read_csv(r"C:\Users\Lenovo\OneDrive\Desktop\Data Sets\Advertising.csv")
data
```

Out[5]:

	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 [6]:
```
data.head()
```

Out[6]:

	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 [7]:
```
data.tail()
```

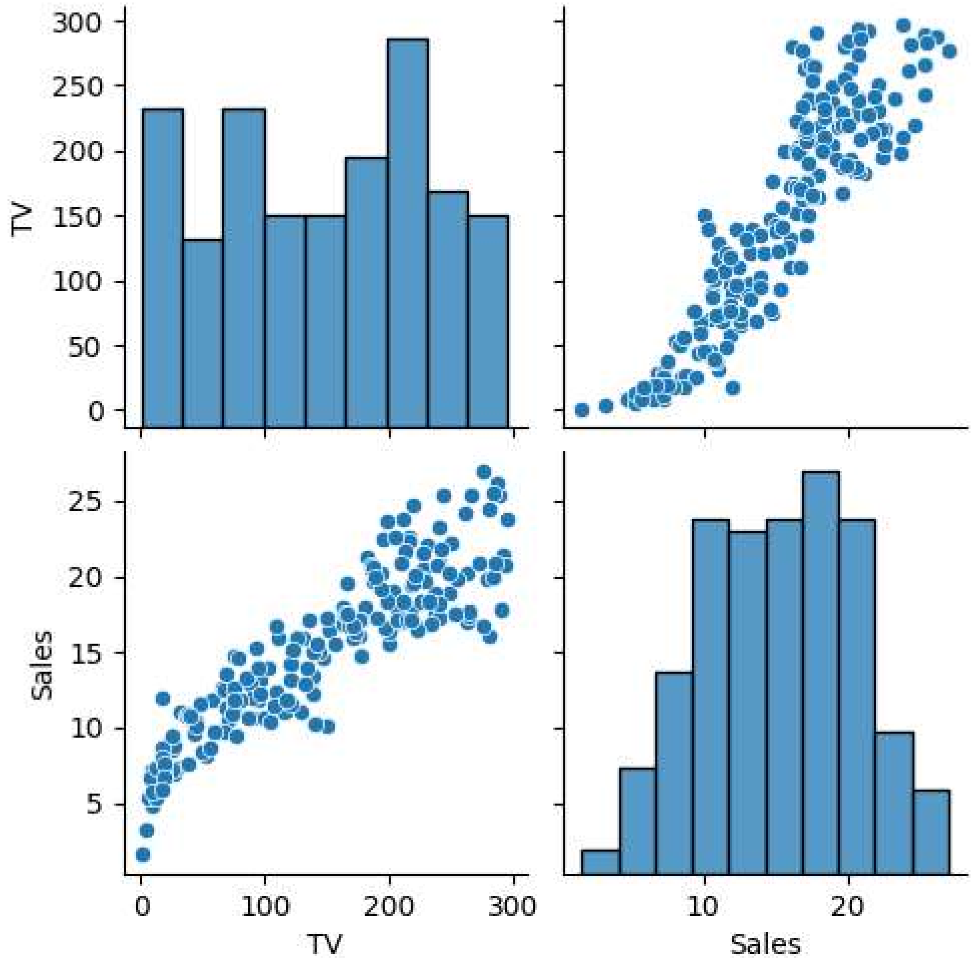
Out[7]:

	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 [8]:
```
plt.figure(figsize=(10,10))
sns.heatmap(data.corr(),annot=True)
```

Out[8]: <Axes: >

In [9]:
```
data.drop(columns=["Radio","Newspaper"],inplace=True)

#pairplot
sns.pairplot(data)

data.Sales=np.log(data.Sales)
```



In [10]:
```
features=data.columns[0:2]
target=data.columns[-1]

#X and y values
X=data[features].values
y=data[target].values

#splot
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=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 (183, 2)
The dimension of X test is (17, 2)
```

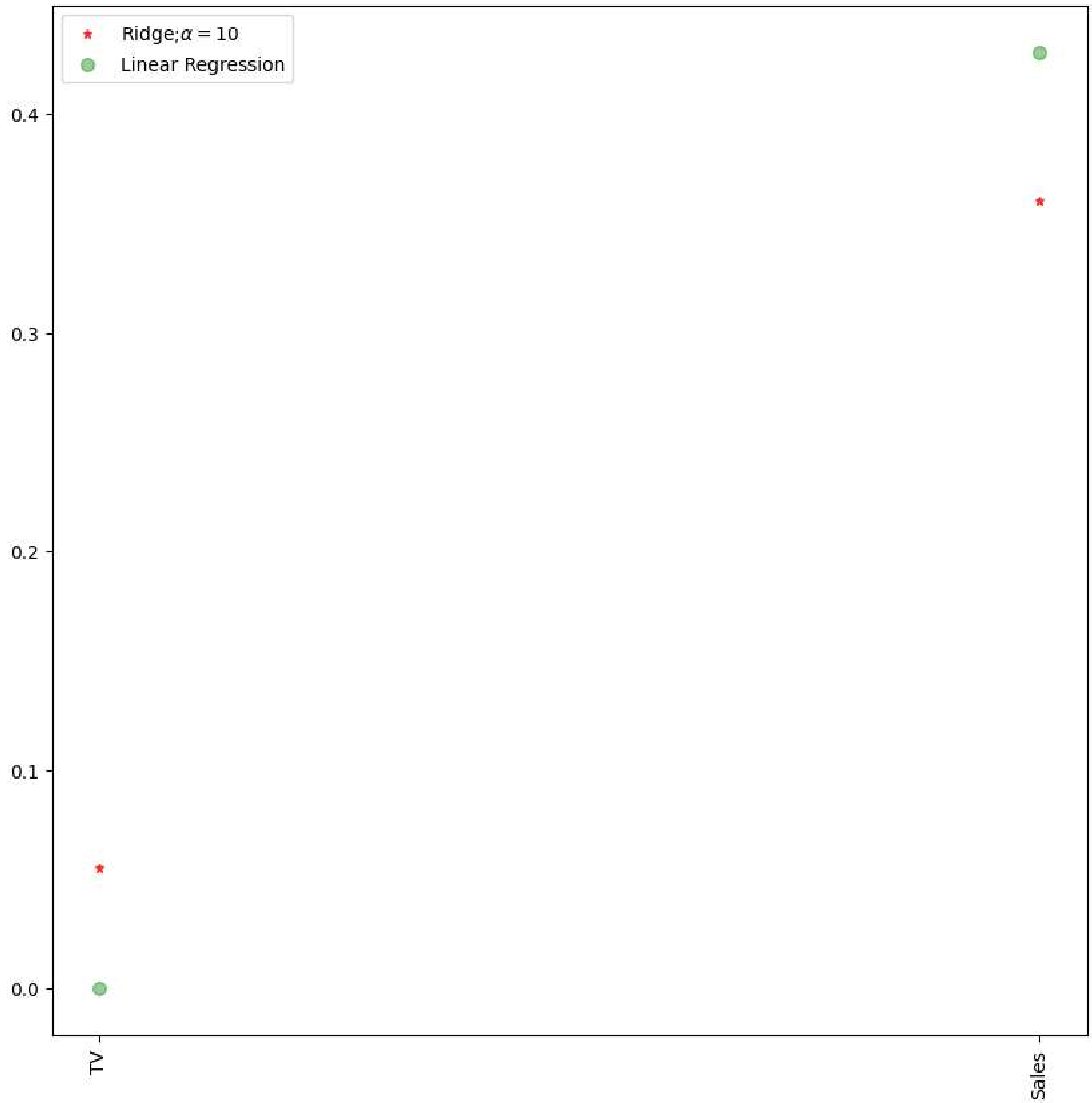
In [11]:
```
#model
lr=LinearRegression()

#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 [12]:
```
#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("\nLinear Regression Model:\n")
print("The train score for lr model is {}".format(train_score_ridge))
print("The test score for lr model is {}".format(test_score_ridge))
```

```

Linear Regression Model:

The train score for lr model is 0.9936432321621578
The test score for lr model is 0.992030334315756
```

In [13]:
```
plt.figure(figsize=(10,10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersi
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,c

plt.xticks(rotation=90)
plt.legend()
plt.show()
```

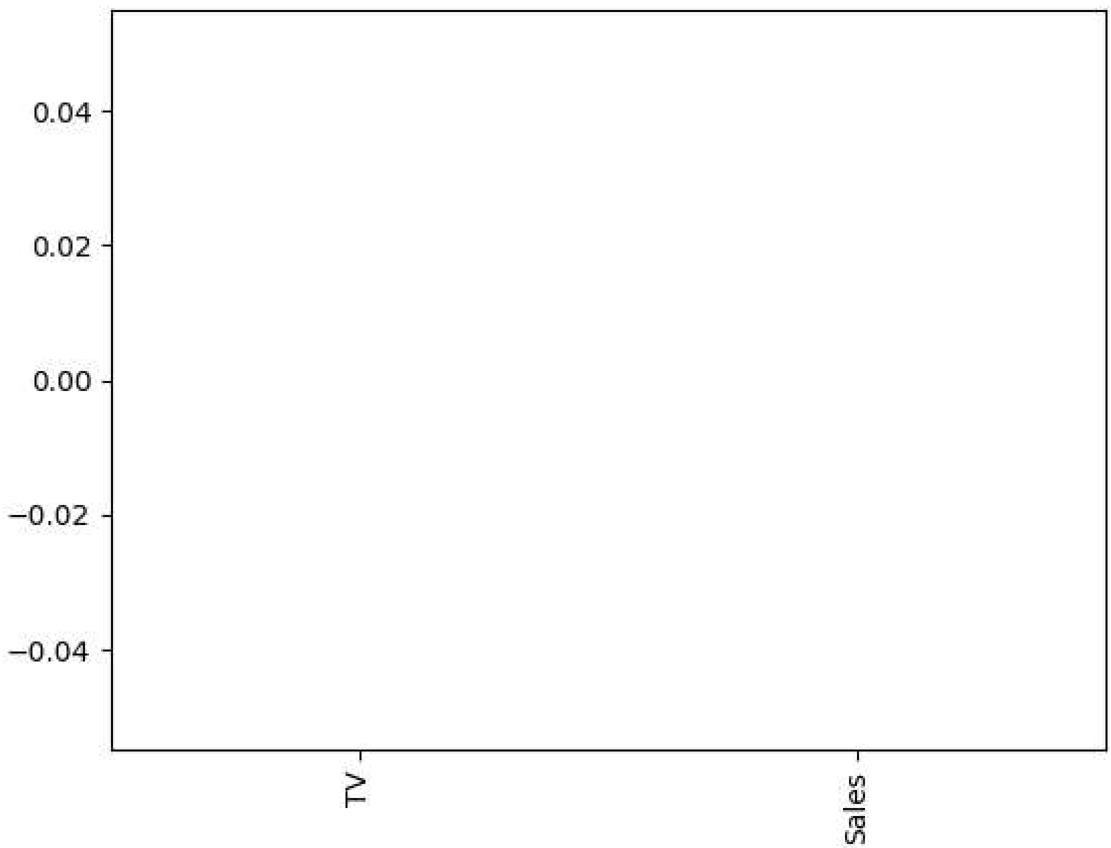


In [14]:
```
#lasso regression model
print("\nLasso Model: \n")
lasso=Lasso(alpha=10)
lasso.fit(X_train,y_train)
train_score_ls=lasso.score(X_train,y_train)
test_score_ls=lasso.score(X_test,y_test)

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 -0.026045348338436l9
```

In [15]:
```
pd.Series(lasso.coef_,features).sort_values(ascending=True).plot(kind="bar")
```

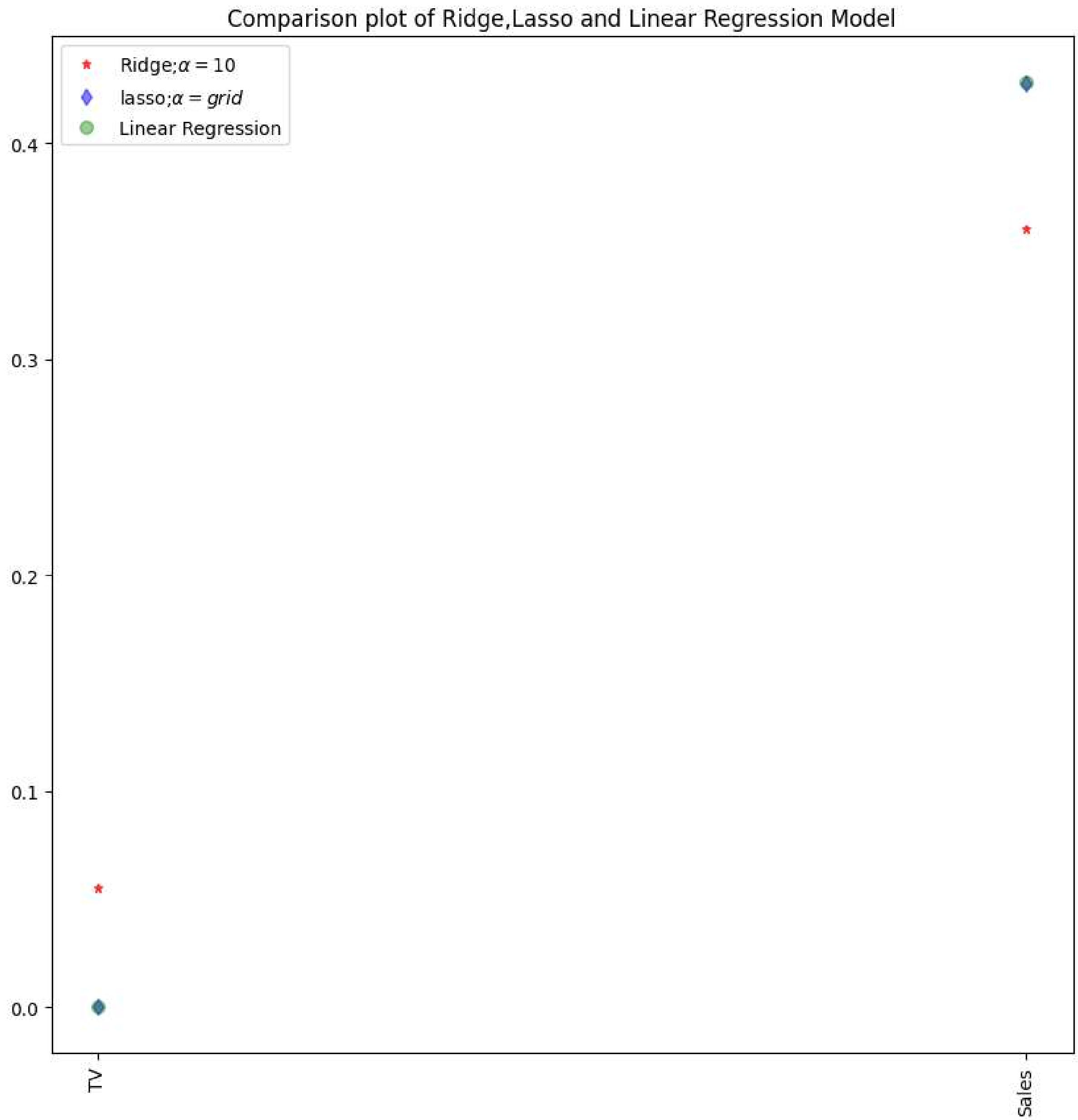
Out[15]: <Axes: >



In [16]:
```
#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_trai

#score
print(lasso_cv.score(X_train,y_train))
print(lasso_cv.score(X_test,y_test))
```

```
0.9999999412343775
0.9999999423739558
```

In [17]:
```
plt.figure(figsize=(10,10))

plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersi
plt.plot(lasso_cv.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,colo
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,c

plt.xticks(rotation=90)
plt.legend()
plt.title("Comparison plot of Ridge,Lasso and Linear Regression Model")
plt.show()
```



In [18]:
```
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_t
print("The train score for ridge model is {}".format(ridge_cv.score(X_test,y_te
```

```
The train score for ridge model is 0.9999999999988164
The train score for ridge model is 0.9999999999984447
```

In [19]:
```
from sklearn.linear_model import ElasticNet
regr=ElasticNet()
regr.fit(X,y)
print(regr.coef_)
print(regr.intercept_)
```

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

In [21]:
```
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 0.5520288898271857
```