In [26]:

```python
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 [27]:

```python
data=pd.read_csv(r"C:\Users\MY HOME\Downloads\Advertising.csv")
data
```

Out[27]:

	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 [28]:

```
data.head()
```

Out[28]:

	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 [29]:

```
data.tail()
```

Out[29]:

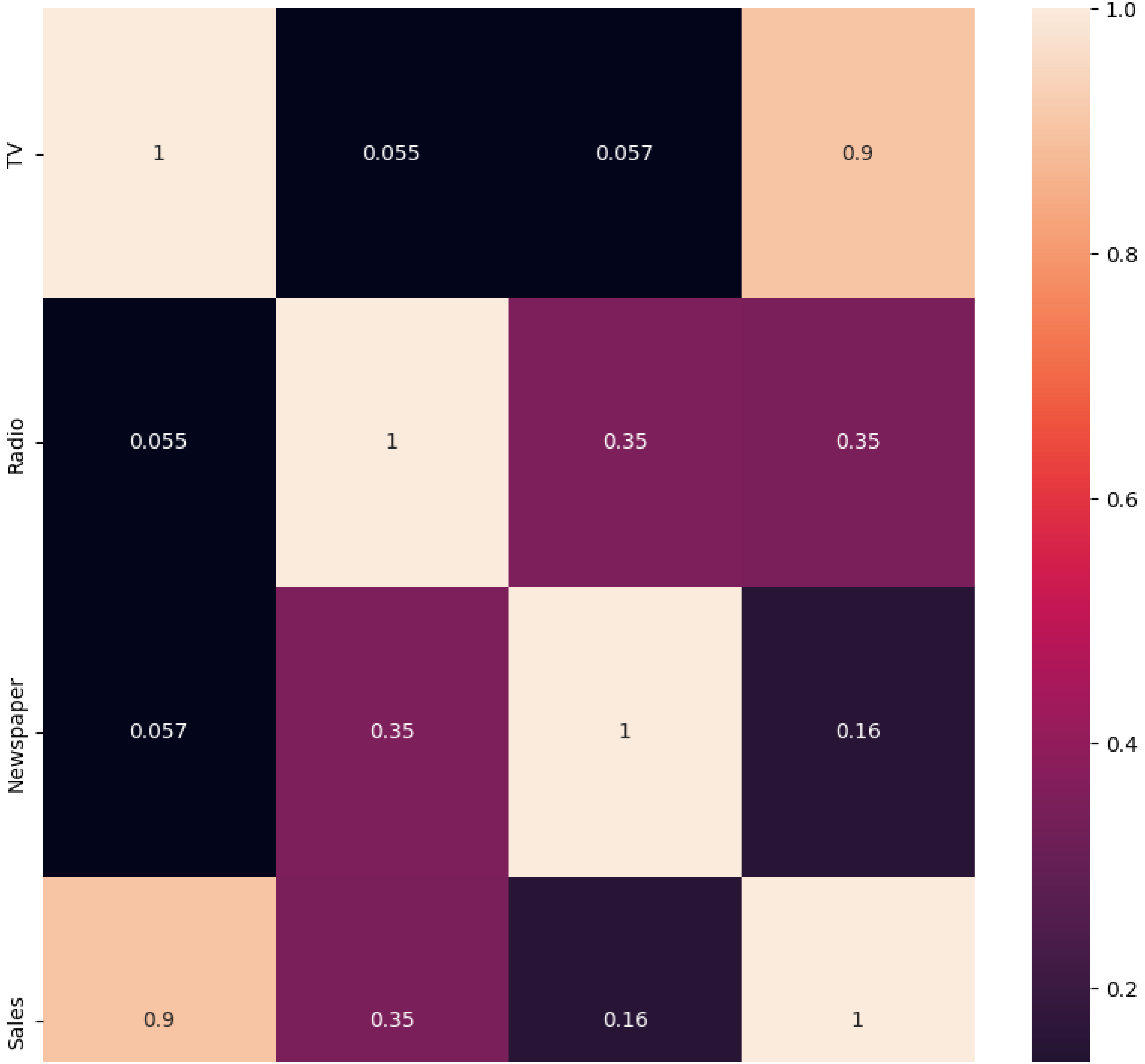
	TV	Radio	Newspaper	Sales
195	38.2	3.7	13.8	7.6
196	94.2	4.9	8.1	14.0
197	177.0	9.3	6.4	14.8
198	283.6	42.0	66.2	25.5
199	232.1	8.6	8.7	18.4

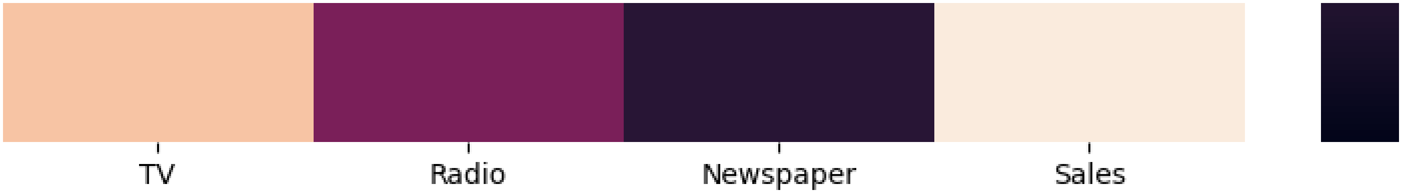
In [30]:

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

Out[30]:

```
<Axes: >
```


TV
Radio
Newspaper
Sales
1
0.055
0.057
0.9
0.055
1
0.35
0.35
0.057
0.35
1
0.16
0.9
0.35
0.16
1
1.0
0.8
0.6
0.4
0.2

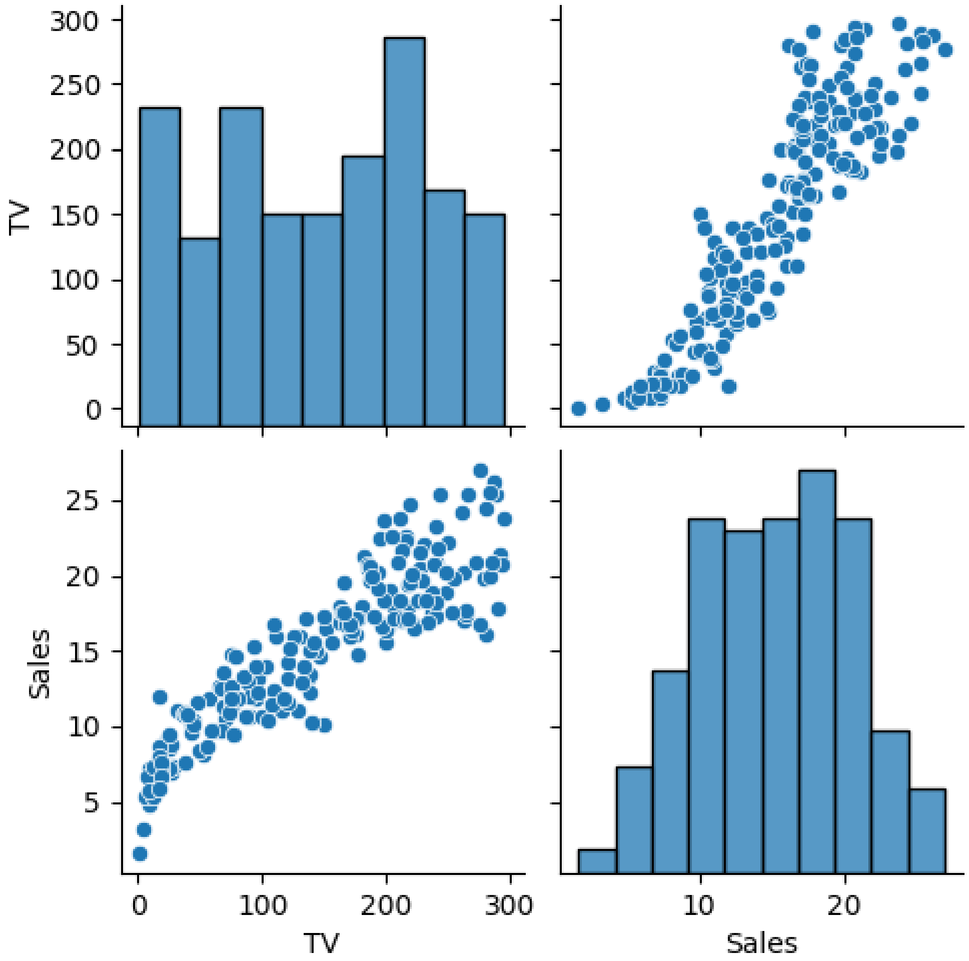

TV
Radio
Newspaper
Sales

In [31]:

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

#pairplot
sns.pairplot(data)


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



In [32]:

```
features=data.columns[0:2]
target=data.columns[-1]
#x and y values
x=data[features].values
y=data[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.fit_transform(x_test)
```

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

In [33]:

```
#model
lr=LinearRegression()
#fit model
lr.fit(x_train,y_train)
#prediction
actual=y_test
train_score_lr=lr.score(x_train,y_train)
test_score_lr=lr.score(x_test,y_test)
print("Linear Regression Model:")
print("The train score for lr model is {}".format(train_score_lr))
print("The train score for lr model is {}".format(test_score_lr))
```

```
Linear Regression Model:
The train score for lr model is 1.0
The train score for lr model is 1.0
```

In [34]:

```
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))
```

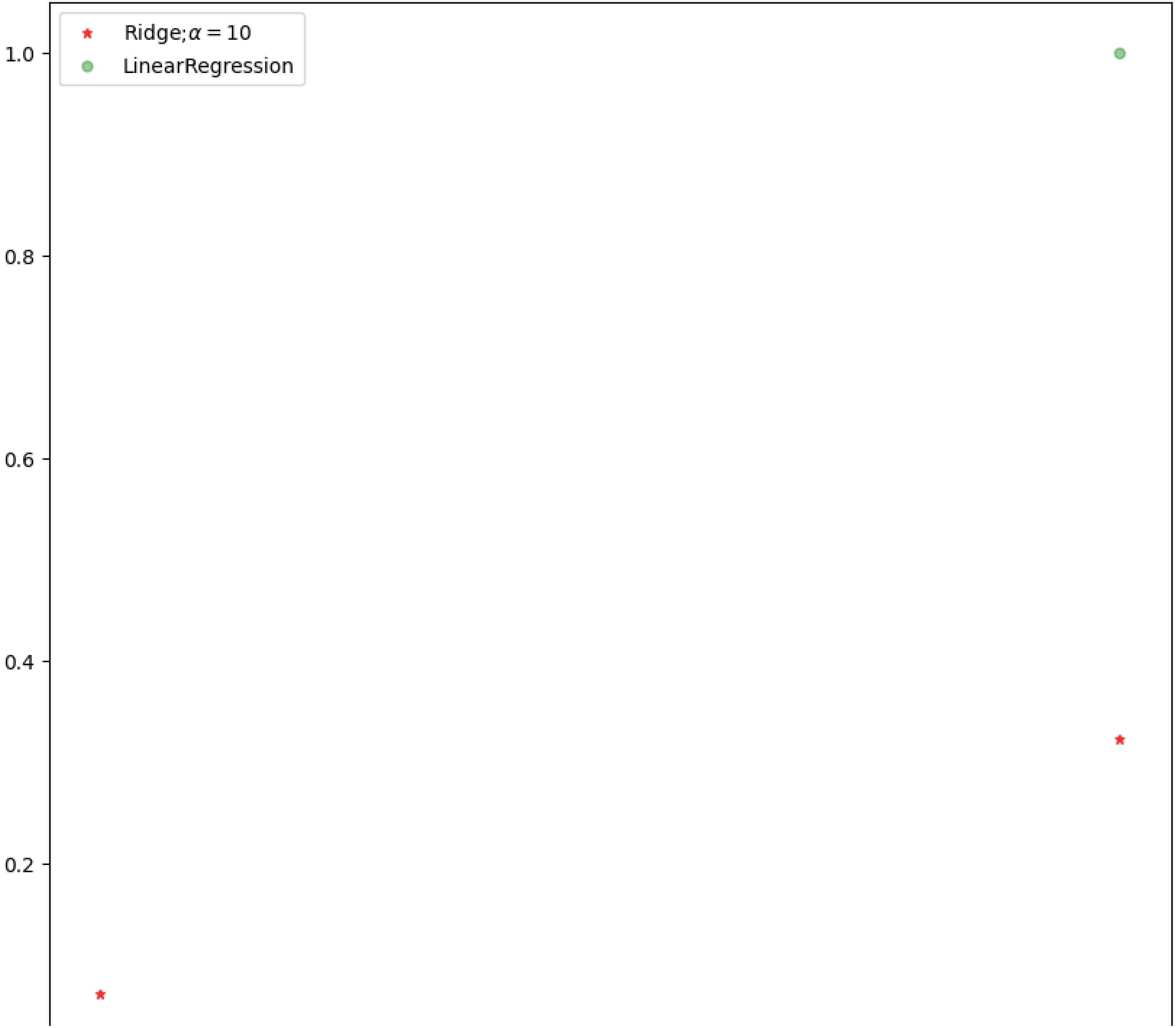
```

Ridge Model:

The train score for ridge model is 0.990287139194161
The test score for ridge model is 0.9665155846267538
```

In [48]:

```
plt.figure(figsize=(10,10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='None',marker='*',markersize=5,color='red',label=r'Ridge;$\
plt.plot(features,lr.coef_,alpha=0.4,linestyle='None',marker='o',markersize=5,color='green',label='LinearRegress
plt.xticks(rotation=90)
plt.legend()
plt.show()
```


Ridge; $\alpha = 10$
LinearRegression
1.0
0.8
0.6
0.4
0.2

In [39]:

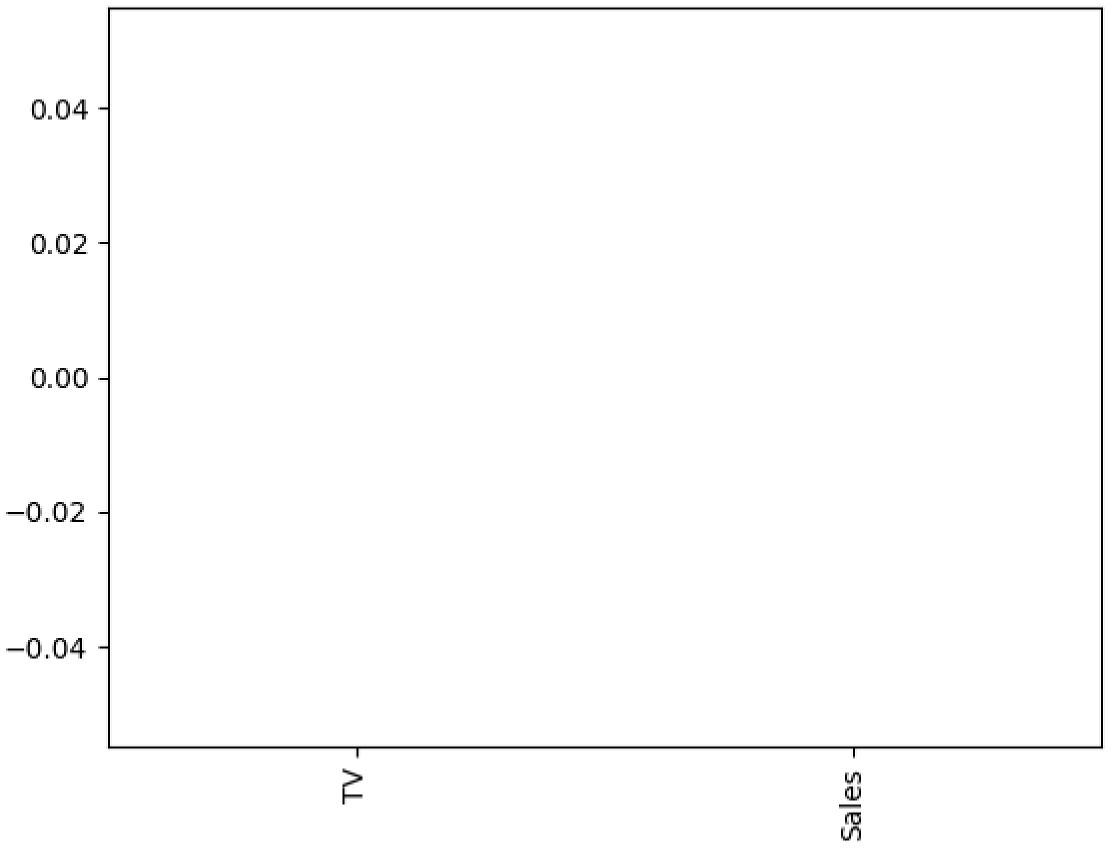
```
#Lasso regression model
print("\nLasso Model: \n")
lasso = Lasso(alpha = 10)
lasso.fit(X_train,y_train)
train_score_ls =lasso.score(X_train,y_train)
test_score_ls =lasso.score(X_test,y_test)
print("The train score for ls model is {}".format(train_score_ls))
print("The test score for ls model is {}".format(test_score_ls))
```

```
Lasso Model: 

The train score for ls model is 0.0
The test score for ls model is -0.004209225323384746
```

In [40]:
```
pd.Series(lasso.coef_, features).sort_values(ascending = True).plot(kind = "bar")
```

Out[40]:
```
<Axes: >
```



In [41]:

```
#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
0.9891149672489288
```

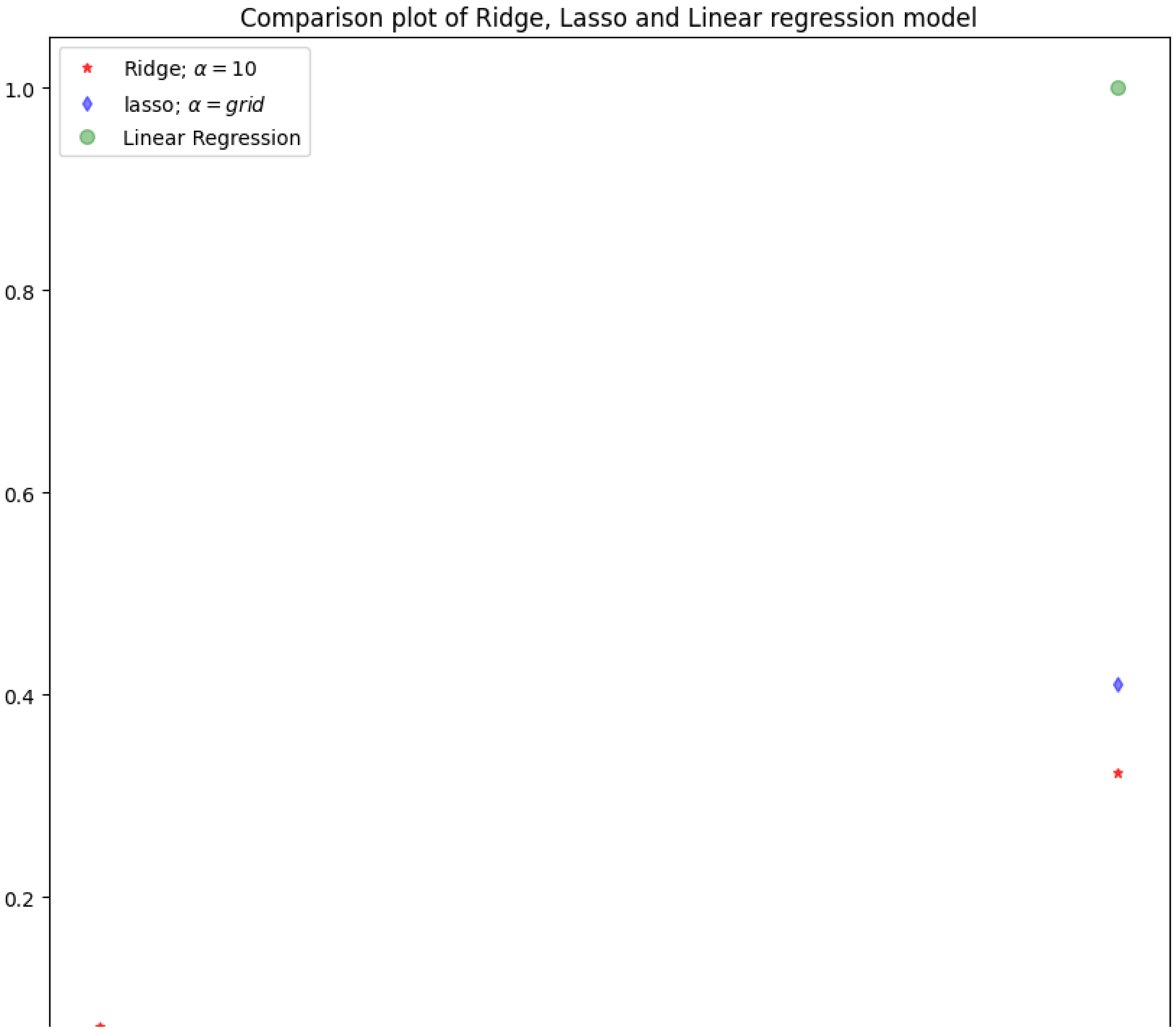
In [47]:

```
 #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; $

#add plot for lasso regression
plt.plot(lasso_cv.coef_,alpha=0.5,linestyle='none',marker='d',markersize=5,color='blue',label=r'lasso; $\alpha =

#add plot for linear model
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color='green',label='Linear Regres

#rotate ax
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; $\alpha = grid$
Linear Regression
1.0
0.8
0.6
0.4
0.2

In [43]:

```
#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 0.9891531024915332
```

ELASTICNET

In [44]:

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

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

In [45]:

```
y_pred_elastic=regr.predict(x_train)
```

In [46]:

```
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 0.036287050935513675
```