In [24]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge,RidgeCV,Lasso
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
```

In [25]:
```
df=pd.read_csv(r"C:\Users\yoshitha lakshmi\OneDrive\Desktop\python\Advertising.csv")
df
```

Out[25]:

	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 [26]:
```
1 df.head()
```

Out[26]:

	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 [27]:
```
1 df.tail()
```

Out[27]:

	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 [28]:
```
1 df.shape
```

Out[28]: (200, 4)

In [29]:
```
1 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 [30]:
```
1 df.describe()
```

Out[30]:

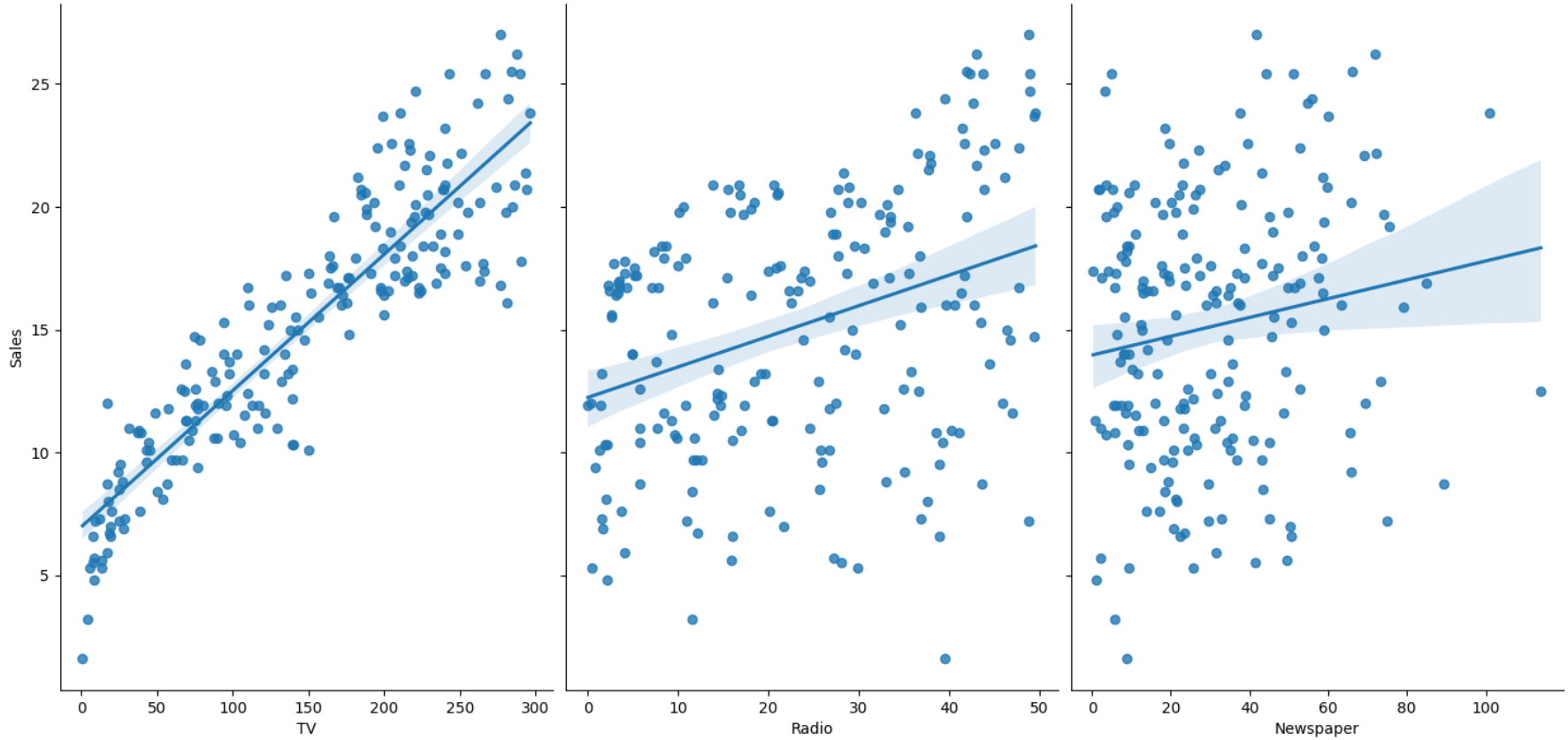
	TV	Radio	Newspaper	Sales
count	200.000000	200.000000	200.000000	200.000000
mean	147.042500	23.264000	30.554000	15.130500
std	85.854236	14.846809	21.778621	5.283892
min	0.700000	0.000000	0.300000	1.600000
25%	74.375000	9.975000	12.750000	11.000000
50%	149.750000	22.900000	25.750000	16.000000
75%	218.825000	36.525000	45.100000	19.050000
max	296.400000	49.600000	114.000000	27.000000

In [31]:
```
1 features = ['TV','Radio','Newspaper']
```

In [32]:
```
1 target = ['Sales']
```

In [33]:
```
1 sns.pairplot(df,x_vars=['TV','Radio','Newspaper'],y_vars=['Sales'],height=7,aspect=0.7,kind='reg')
```

Out[33]: <seaborn.axisgrid.PairGrid at 0x1acdfea2080>



In [34]:
```
1 from sklearn.model_selection import train_test_split
2 from sklearn.linear_model import LinearRegression
```

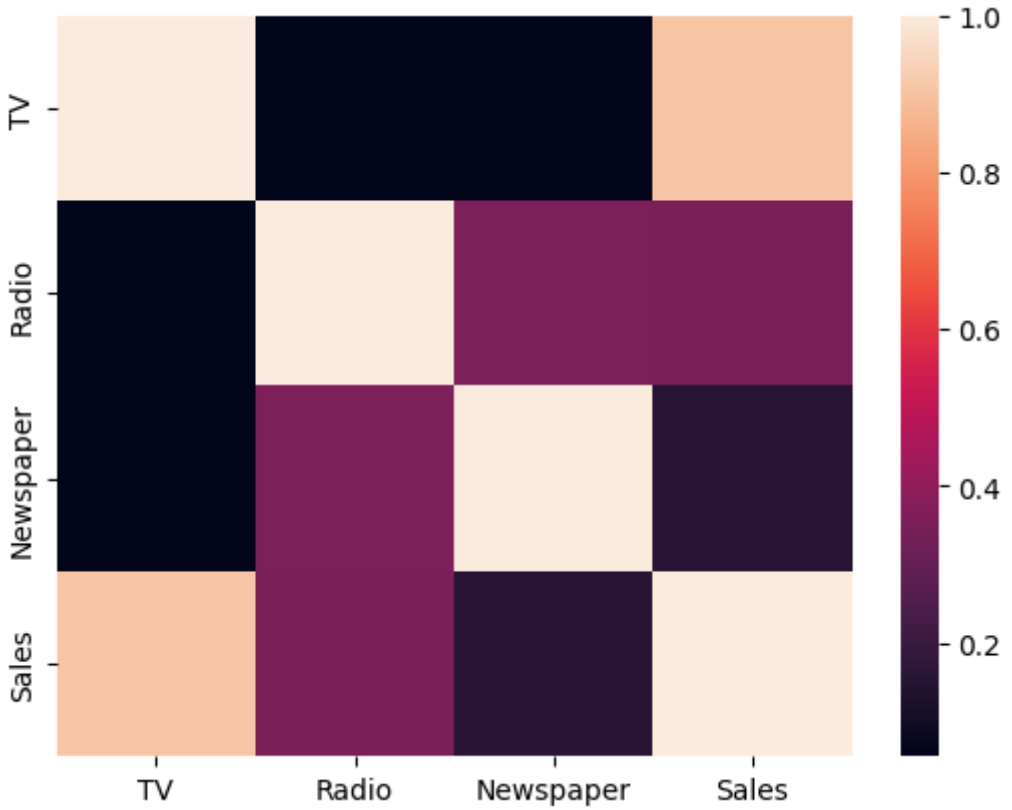
In [55]:
```
1 x=np.array(df[features])
2 y=np.array(df[target])
```

In [66]:
```
X_train,X_test,y_train,y_test = train_test_split(x,y,test_size=0.25)
regr = LinearRegression()
regr.fit(X_train,y_train)
print(regr.score(x_test, y_test))
```

```
-0.5824704202324338
```

In [67]:
```
sns.heatmap(df.corr())
```

Out[67]: <Axes: >



In [68]:
```
from sklearn.linear_model import LinearRegression
lm = LinearRegression()
lm.fit(X_train,y_train)
```

Out[68]:
```
LinearRegression()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [69]:
```
print(lm.intercept_)
```

```
4.660916410008371
```

In [70]:
```
df.columns
```

Out[70]:
```
Index(['TV', 'Radio', 'Newspaper', 'Sales'], dtype='object')
```

In [71]:
```
Addf=df[['TV', 'Radio', 'Newspaper', 'Sales']]
```

In [72]:
```
X = Addf[['TV', 'Radio', 'Newspaper']]
y = df['Sales']
```

In [73]:
```
from sklearn.linear_model import LinearRegression
lm = LinearRegression()
lm.fit(X_train,y_train)
```

Out[73]:
```
LinearRegression()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [74]:
```
coeff_df=pd.DataFrame(lm.coef_,X.columns,columns=['coefficient'])
coeff_df
```

Out[74]:

	coefficient
TV	0.054360
Radio	0.108862
Newspaper	-0.002042

In [75]:
```
predictions=lm.predict(x_test)
```

In [76]:
```
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: 5.609455830624045
MSE: 47.55186887354155
RMSE: 6.8957863129262895
```

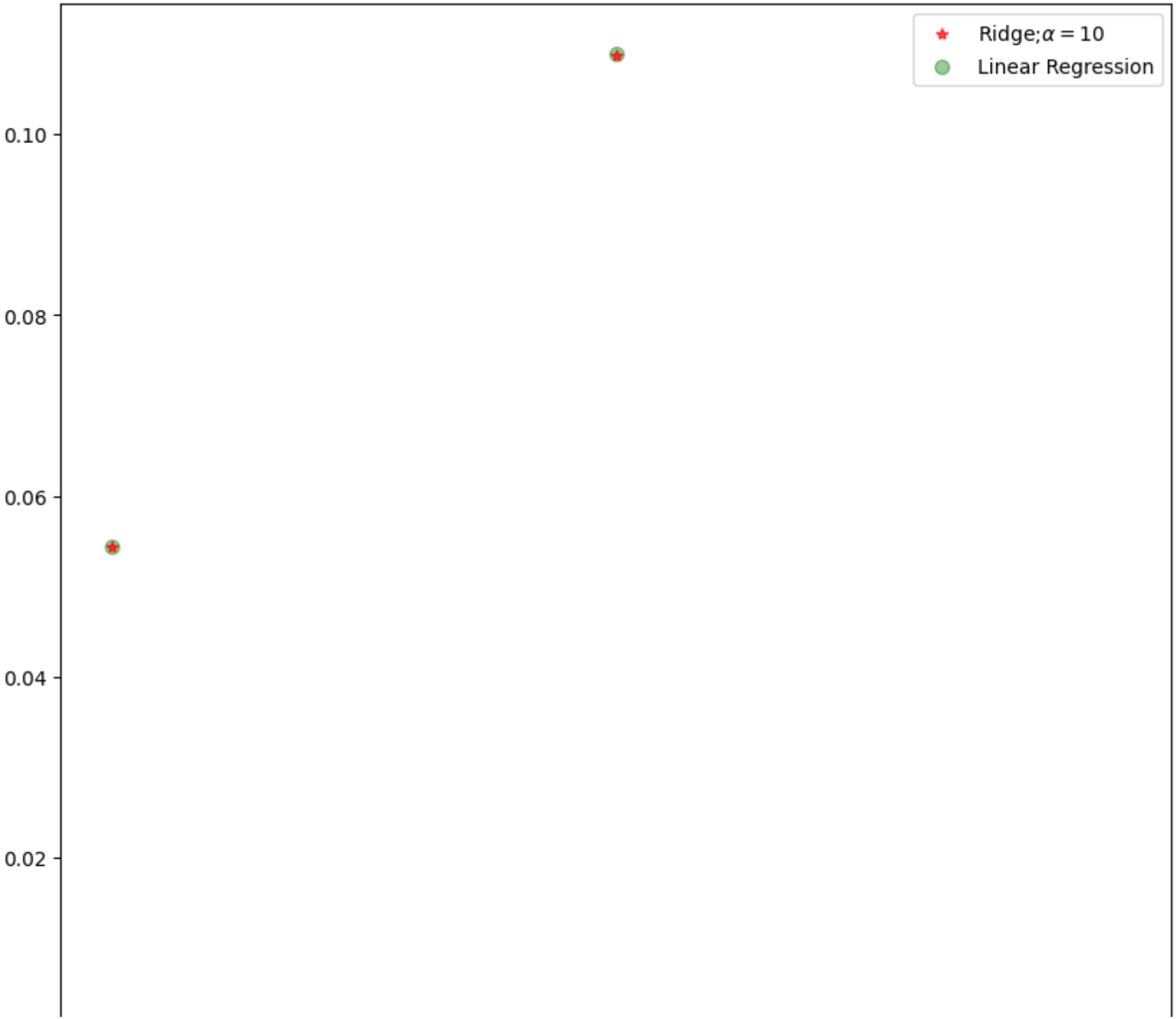
Ridge Regression Model

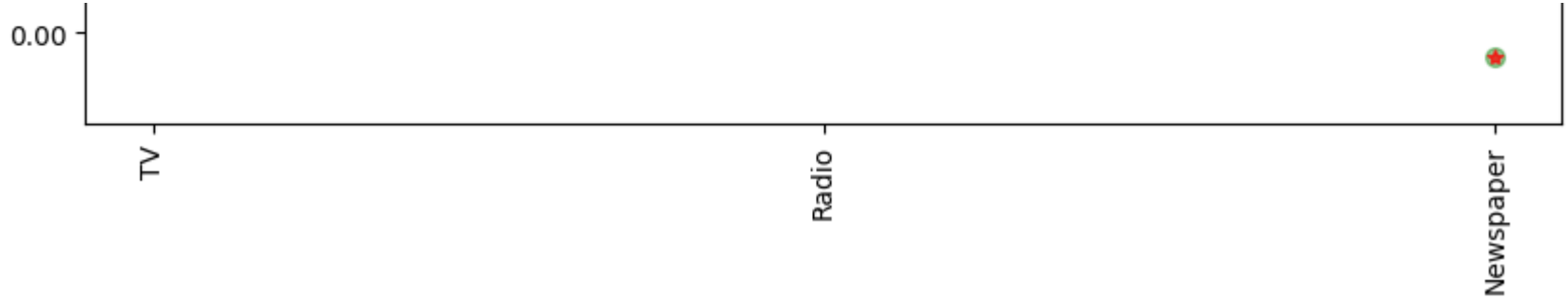
In [77]:
```
ridgeReg = Ridge(alpha=10)
ridgeReg.fit(X_train,y_train)
train_score_ridge = ridgeReg.score(X_train,y_train)
test_score_ridge = ridgeReg.score(x_test,y_test)

print('\nRidge Model\n')
print('Train score for ridge model is {}'.format(train_score_ridge))
print('Test score for ridge model is{}'.format(test_score_ridge))
```

```
Ridge Model

Train score for ridge model is 0.8954000079149985
Test score for ridge model is-0.5824696427173317
```

In [78]:

```
plt.figure(figsize = (10,10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=6,color='red',label=r'Ridge;$\
plt.plot(features,lm.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color='green',label='Linear Regres
plt.xticks(rotation=90)
plt.legend()
plt.show()
```


Ridge;$\alpha = 10$
Linear Regression
0.10
0.08
0.06
0.04
0.02



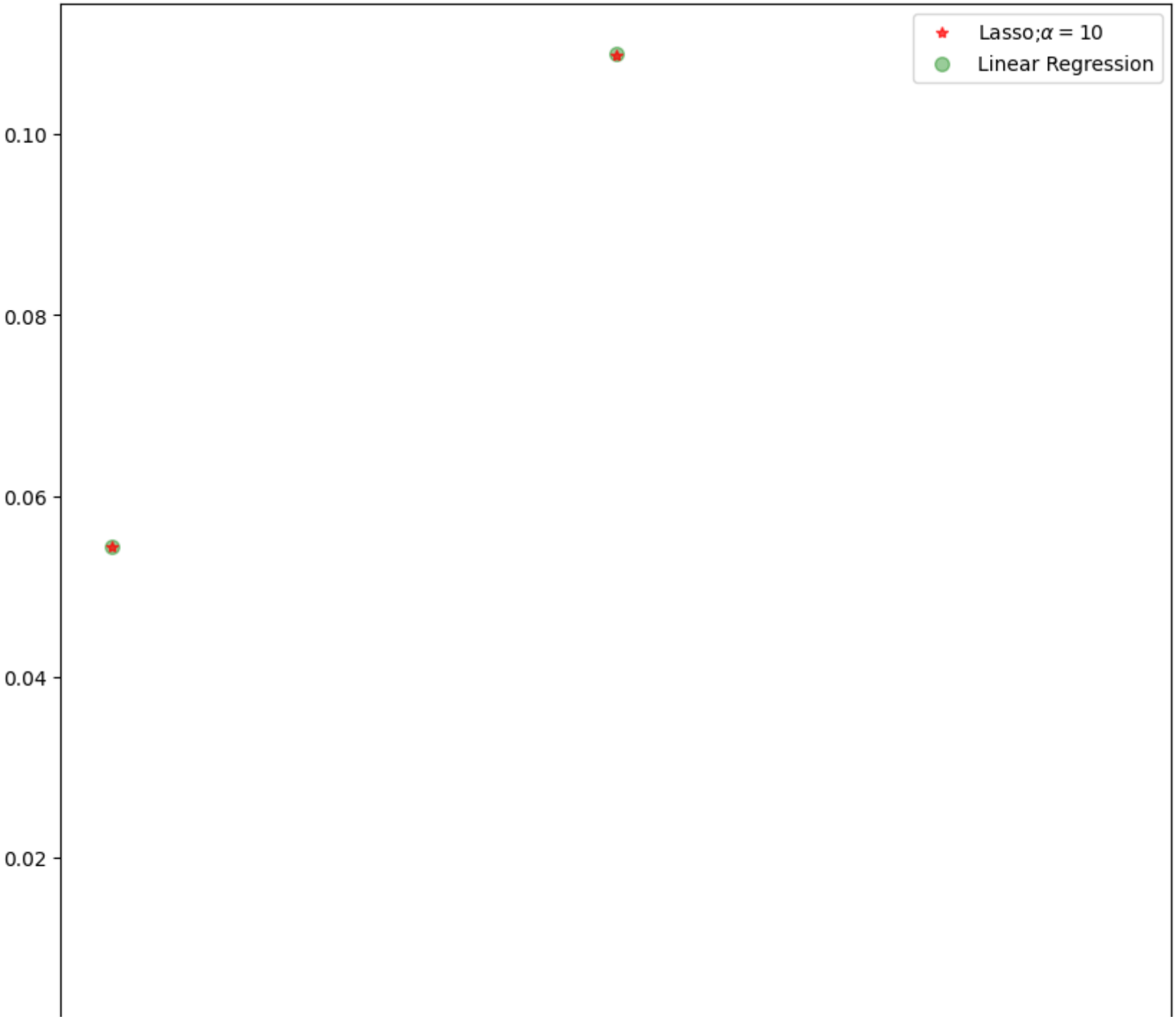
In [79]:

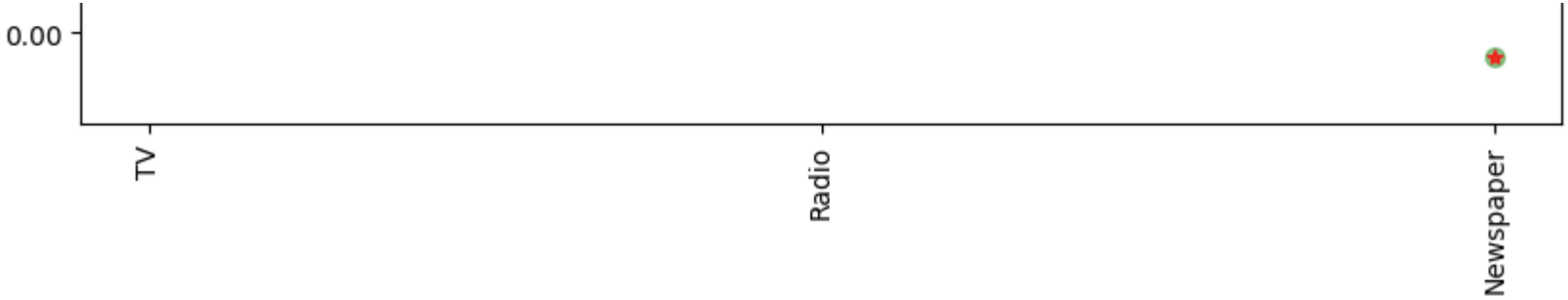
```
lassoReg = Lasso(alpha=10)
lassoReg.fit(X_train,y_train)
train_score_lasso = lassoReg.score(X_train,y_train)
test_score_lasso = lassoReg.score(x_test,y_test)

print('\nRidge Model\n')
print('Train score for lasso model is {}'.format(train_score_lasso))
print('Test score for lasso model is{}'.format(test_score_lasso))
```

```
Ridge Model

Train score for lasso model is 0.8789640031484611
Test score for lasso model is-0.5733638138042045
```

In [80]:

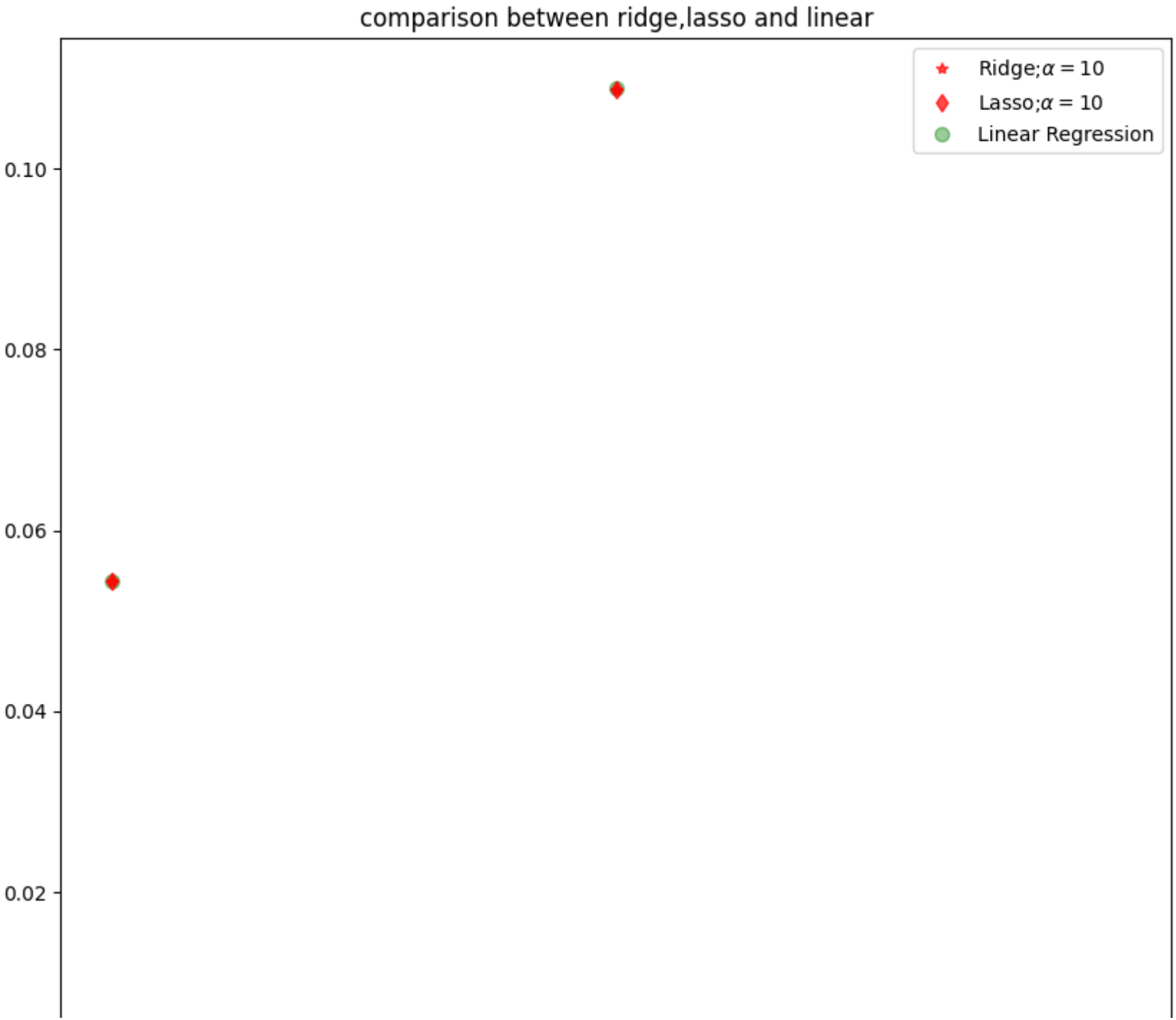
```
plt.figure(figsize = (10,10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=6,color='red',label=r'Lasso;$\
plt.plot(features,lm.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color='green',label='Linear Regres
plt.xticks(rotation=90)
plt.legend()
plt.show()
```

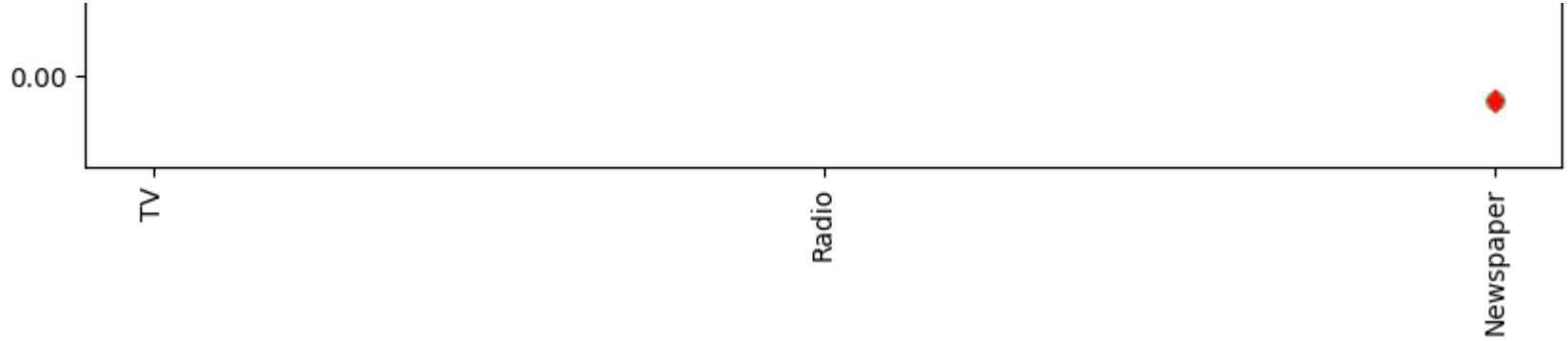

Lasso;α = 10
Linear Regression
0.10
0.08
0.06
0.04
0.02


0.00
TV
Radio
Newspaper

In [81]:

```
# Comparison between Ridge,Lasso and RidgeCV
plt.figure(figsize=(10,10))

plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=6,color='red',label=r'Ridge;$\
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='d',markersize=6,color='red',label=r'Lasso;$\
plt.plot(features,lm.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color='green',label='Linear Regres
plt.xticks(rotation=90)
plt.title('comparison between ridge,lasso and linear')
plt.legend()
plt.show()
```


comparison between ridge,lasso and linear
Ridge;$\alpha = 10$
Lasso;$\alpha = 10$
Linear Regression
0.10
0.08
0.06
0.04
0.02



In [82]:
```
# Linear CV model using Ridge
from sklearn.linear_model import RidgeCV
ridge_CV=RidgeCV(alphas=[0.0001,0.01,0.001,0.1]).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 is 0.8954000179269546
The test score for ridge model is -0.5824704108926089
```

In [83]:
```
# Linear CV model using Losso
from sklearn.linear_model import LassoCV
lasso_CV=LassoCV(alphas=[0.0001,0.01,0.001,0.1]).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 is 0.8954000179239859
The test score for lasso model is -0.5824703283838824
```

Elastic Net

In [87]:
```
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 test set',mean_squared_error)
```

```
[0.05440081 0.1046715  0.        ]
4.696191158087226
Mean squared error on test set 2.8150966541195057
```

In [ ]:
```

```