In [30]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
df=pd.read_csv(r"C:\Users\91756\Documents\python\Advertising.csv")
df
```

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]:
```
df.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]:
```
df.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]:
```
df.shape
```

Out[5]:
```
(200, 4)
```

In [6]:
```
df.describe()
```

Out[6]:

	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

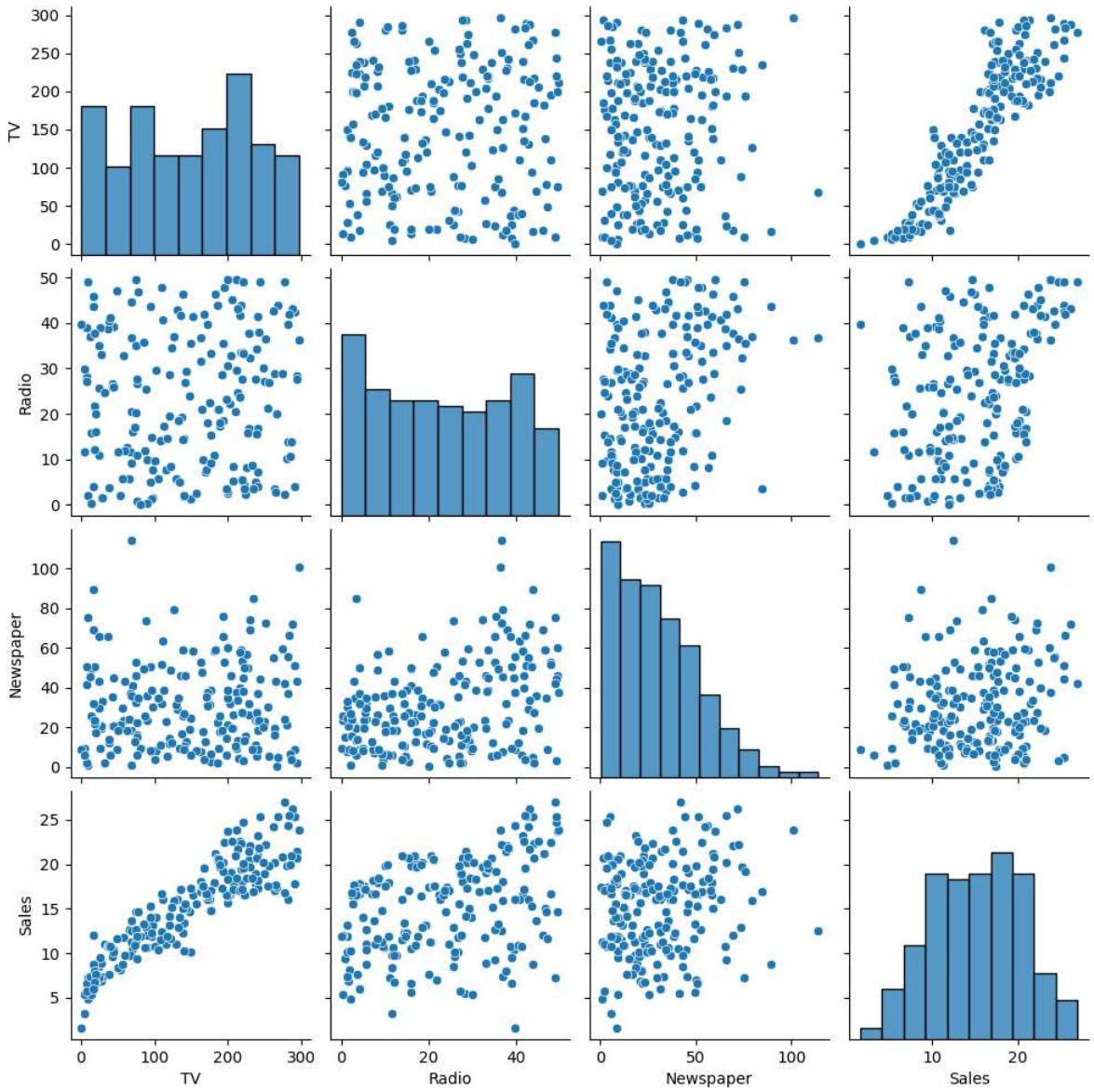
In [7]:
```
df.info()
```

```
<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 [8]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

In [9]:
```
sns.pairplot(df)
```

Out[9]: <seaborn.axisgrid.PairGrid at 0x22d9f352170>

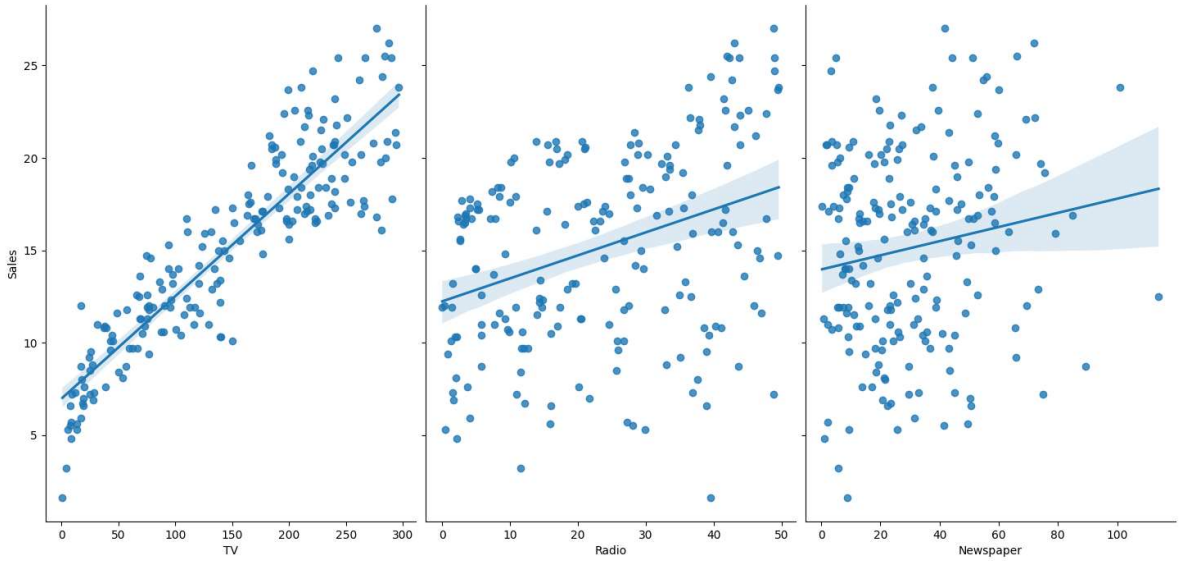


In [100]:
```
features =df.columns[0:3]
```

In [101]:
```
target = df.columns[-1]
```

In [102]:
```
sns.pairplot(df,x_vars=['TV','Radio','Newspaper'],y_vars='Sales',height=7,aspe
```

Out[102]: <seaborn.axisgrid.PairGrid at 0x22dc48368c0>



In [103]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

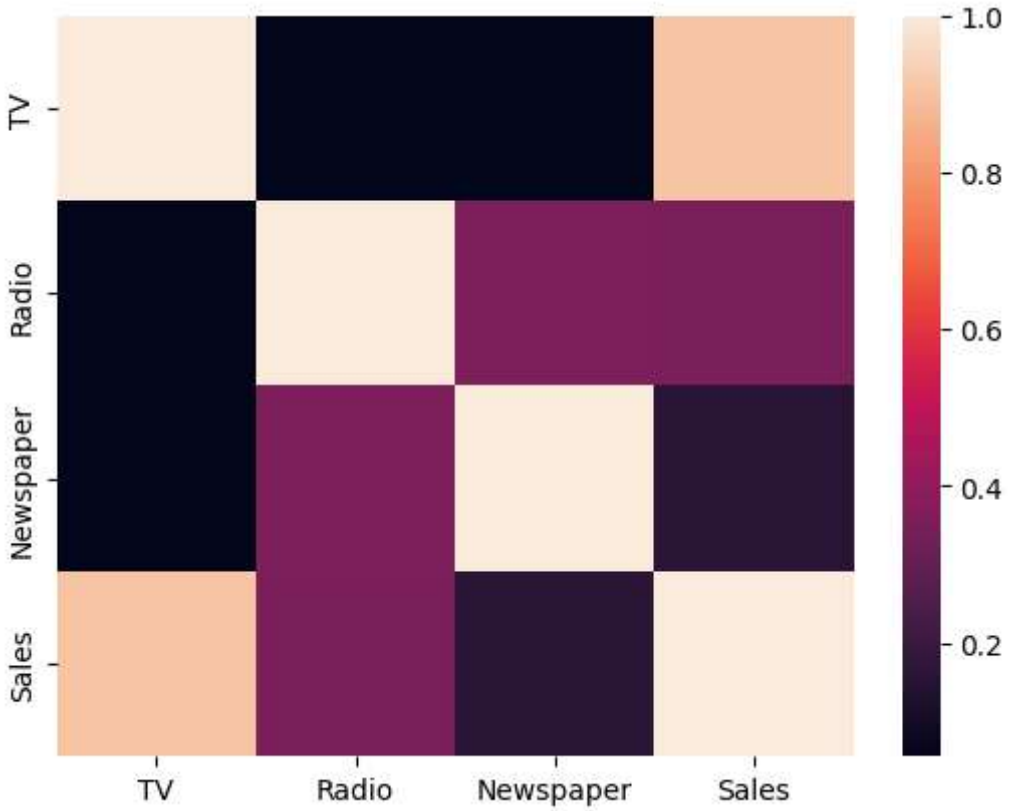
In [104]:
```
x=np.array(df[features])
y=np.array(df[target])
```

In [105]:
```
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))
```

```
0.8631814433474745
```

In [106]:
```
sns.heatmap(df.corr())
```

Out[106]: <Axes: >



In [107]:
```
from sklearn.linear_model import LinearRegression
lm=LinearRegression()
lm.fit(x_train,y_train)
```

Out[107]: 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 [108]:
```
print(lm.intercept_)
```

```
4.675527241458383
```

In [109]:
```
df.columns
```

Out[109]: Index(['TV', 'Radio', 'Newspaper', 'Sales'], dtype='object')

In [110]:
```
Addf=df[['TV','Radio','Newspaper','Sales']]
```

In [111]:
```
x=Addf[['TV','Radio','Newspaper']]
y=df['Sales']
```

In [112]:
```
from sklearn.linear_model import LinearRegression
lm=LinearRegression()
lm.fit(x_train,y_train)
```

Out[112]:
```
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 [113]:
```
print(lm.intercept_)
```

```
4.675527241458383
```

In [114]:
```
coeff_df = pd.DataFrame(lm.coef_)
coeff_df
```

Out[114]:

	0
0	0.056040
1	0.099342
2	-0.000878

In [115]:
```
predictions=lm.predict(x_test)
```

In [116]:
```
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: 1.2450972910731728
MSE: 2.9342791752483883
RMSE: 1.712973781249552
```

Ridge Regression Model

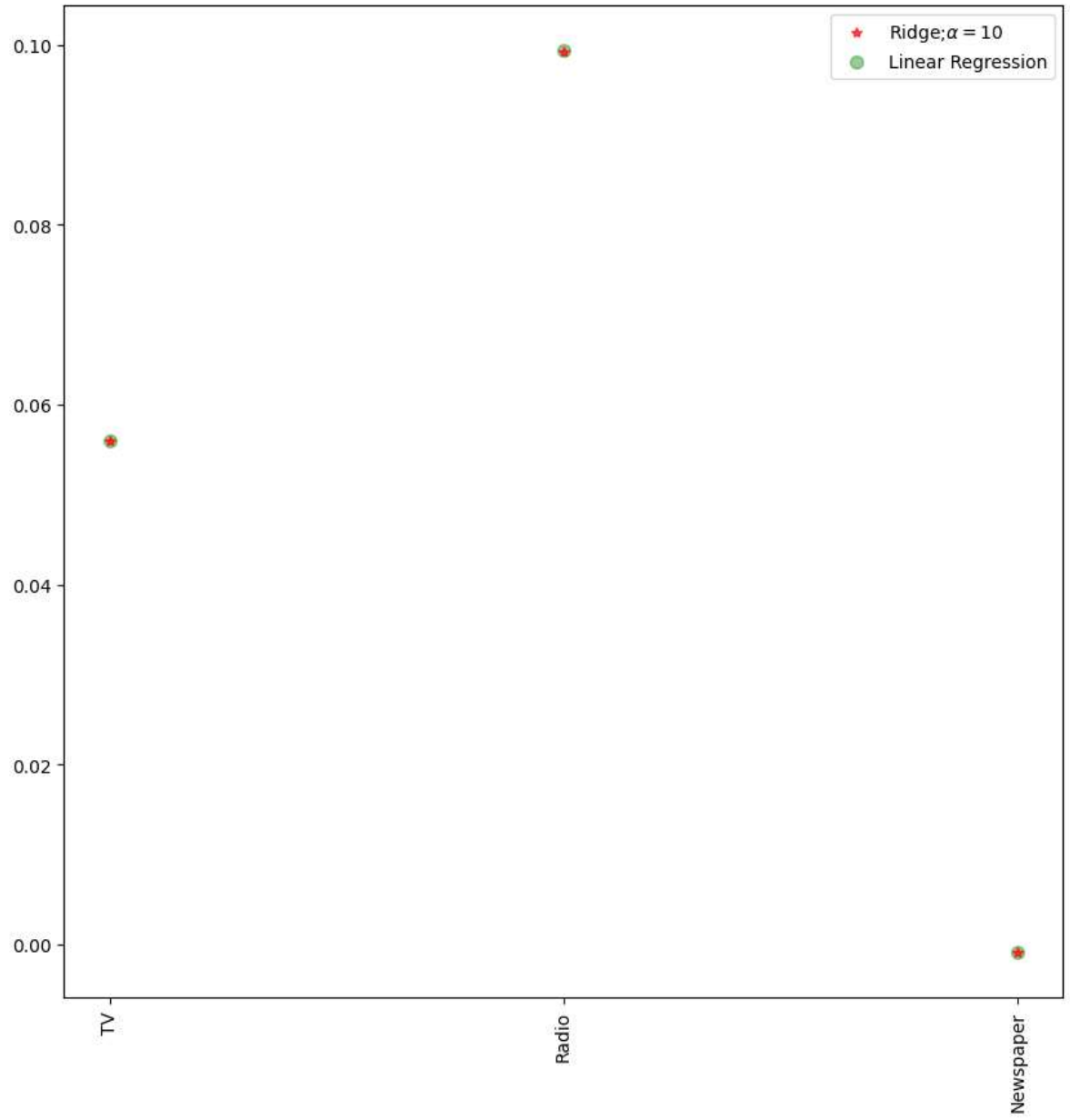
In [117]:
```
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge,RidgeCV,Lasso
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
```

In [118]:
```
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 0.9103297738689969
Test score for ridge model is 0.8631625911226438
```

In [119]:
```
plt.figure(figsize = (10,10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markers
plt.plot(features,lm.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,
plt.xticks(rotation=90)
plt.legend()
plt.show()
```

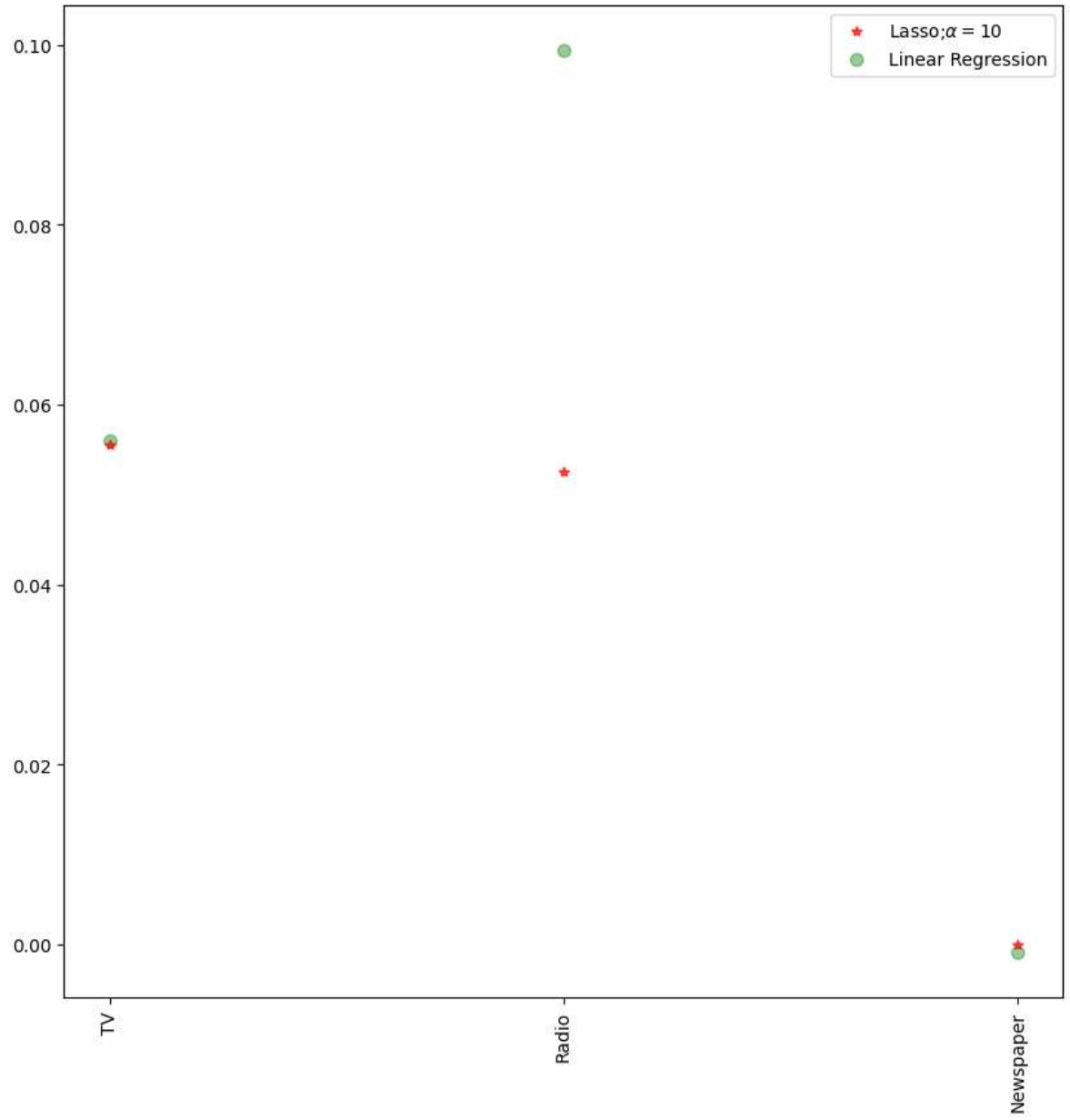


In [125]:
```
lassoReg = Lasso(alpha=10)
lassoReg.fit(x_train,y_train)
train_score_lasso = lassoReg.score(x_train,y_train)
test_score_lasso = lassoReg.score(x_test,y_test)
print('\nRidge Model\n')
print('Train score for lasso model is {}'.format(train_score_lasso))
print('Test score for lasso model is{}'.format(test_score_lasso))
```

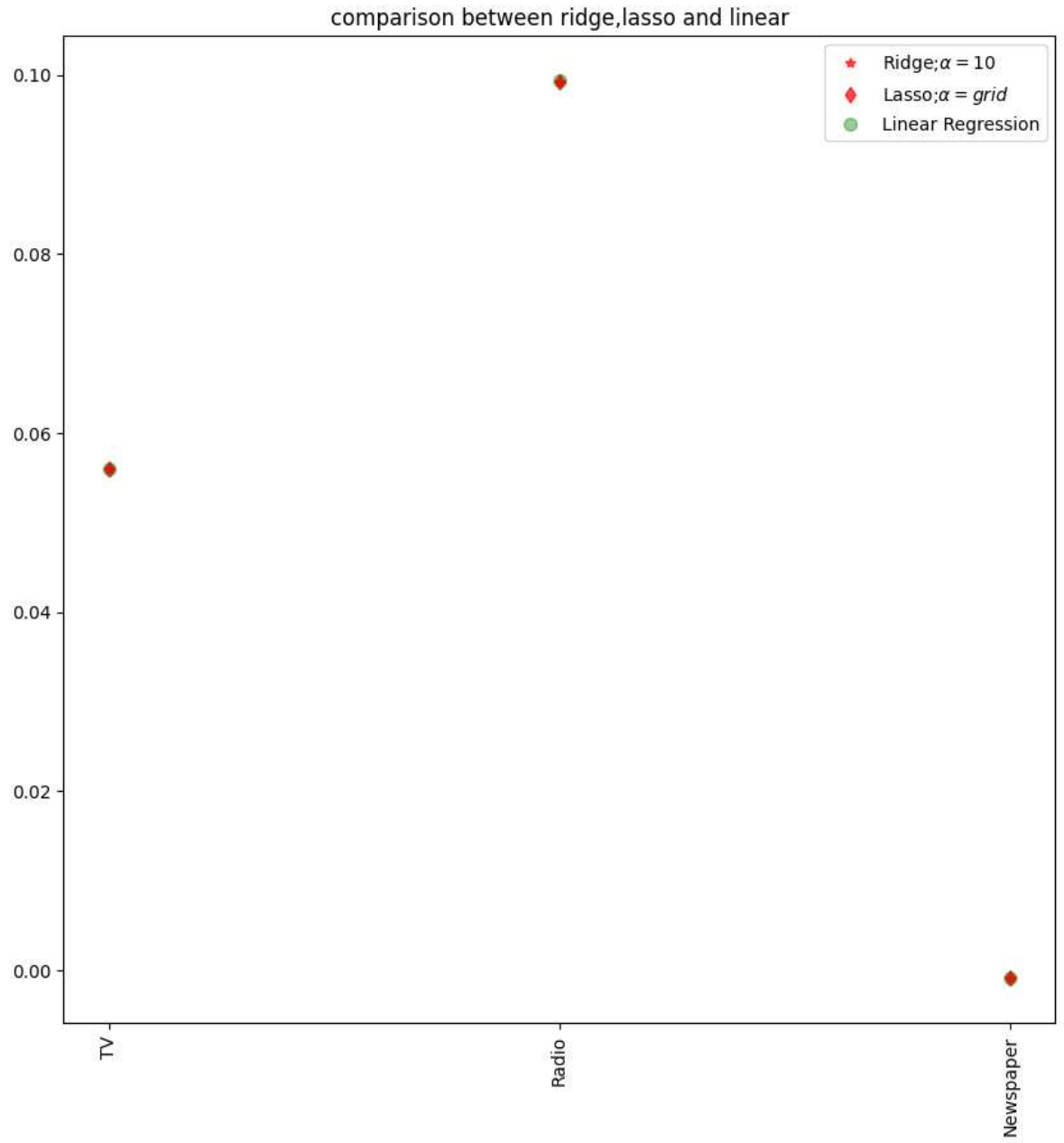
```
Ridge Model

Train score for lasso model is 0.8946969173105311
Test score for lasso model is0.8123990929003627
```

In [126]:
```
plt.figure(figsize = (10,10))
plt.plot(features,lassoReg.coef_,alpha=0.7,linestyle='none',marker='*',markers
plt.plot(features,lm.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,
plt.xticks(rotation=90)
plt.legend()
plt.show()
```



In [121]:
```
# Comparison between Ridge,Lasso and RidgeCV
plt.figure(figsize=(10,10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markers
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='d',markers
plt.plot(features,lm.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,
plt.xticks(rotation=90)
plt.title('comparison between ridge,lasso and linear')
plt.legend()
plt.show()
```



In [123]:
```
# 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_
print("The test score for ridge model is {}".format(ridge_CV.score(x_test,y_te
```

```
The train score for ridge model is 0.9103297817545735
The test score for ridge model is 0.863181256614648
```

In [124]:
```
# Linear CV model using Losso
from sklearn.linear_model import LassoCV
lasso_CV=L=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_
print("The test score for lasso model is {}".format(lasso_CV.score(x_test,y_te
```

```
The train score for lasso model is 0.9103264044782
The test score for lasso model is 0.8629561862241895
```

In [ ]:
```

```