In [2]:
```
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 Ridge,RidgeCV ,Lasso
from sklearn.preprocessing import StandardScaler
```

In [3]:
```
#(advertising) model
df=pd.read_csv(r"C:\Users\ubinl\Downloads\Advertising.csv")
df
```

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
...	...	...	...	...
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 [4]:
```
df.head()
```

Out[4]:

	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 [5]:
```
df.tail()
```

Out[5]:

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

Out[6]:

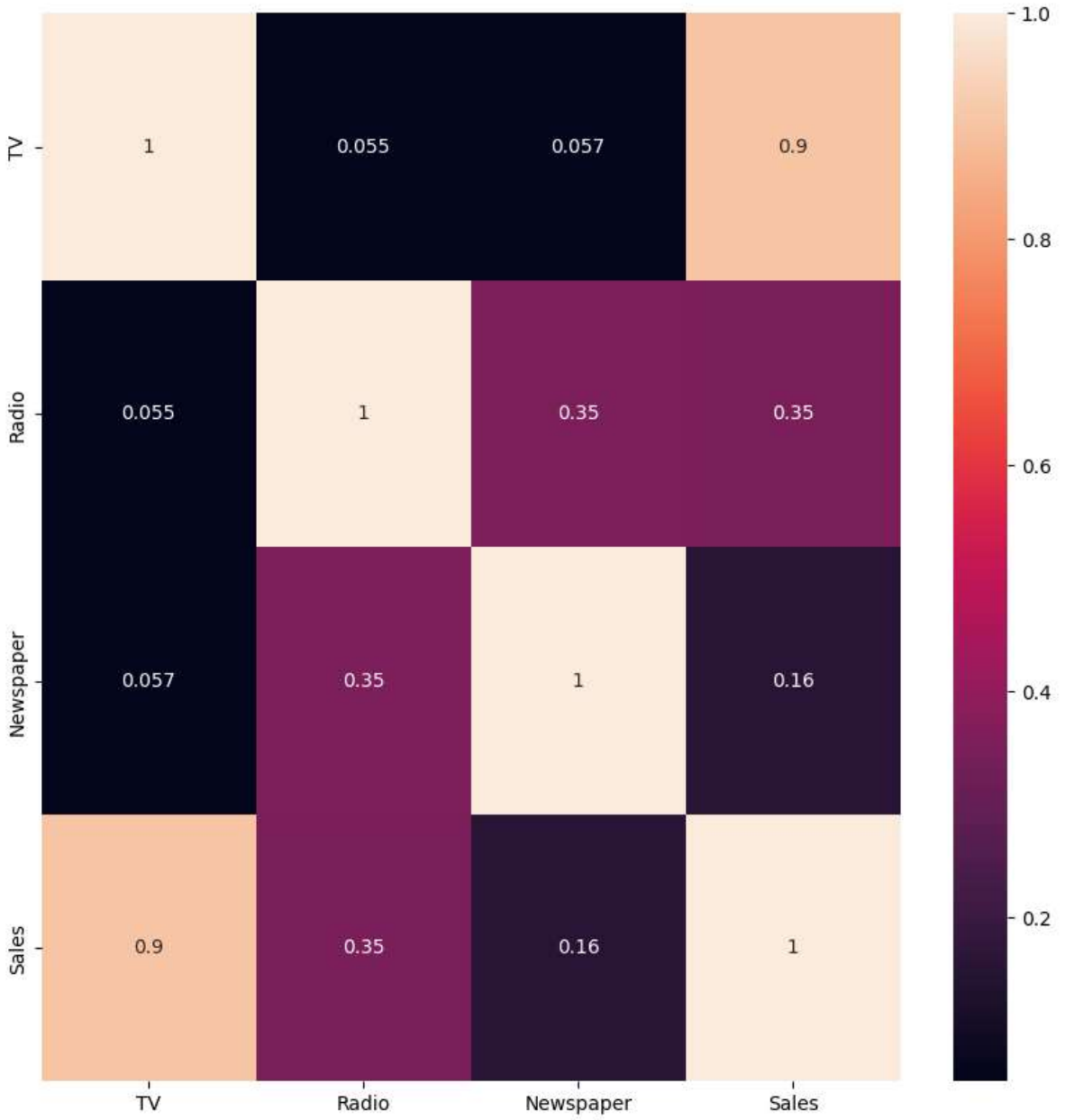
	TV	Radio	Newspaper	Sales
count	200.000000	200.000000	200.000000	200.000000
mean	147.042500	23.264000	30.554000	15.130500
std	85.854236	14.846809	21.778621	5.283892
min	0.700000	0.000000	0.300000	1.600000
25%	74.375000	9.975000	12.750000	11.000000
50%	149.750000	22.900000	25.750000	16.000000
75%	218.825000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

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

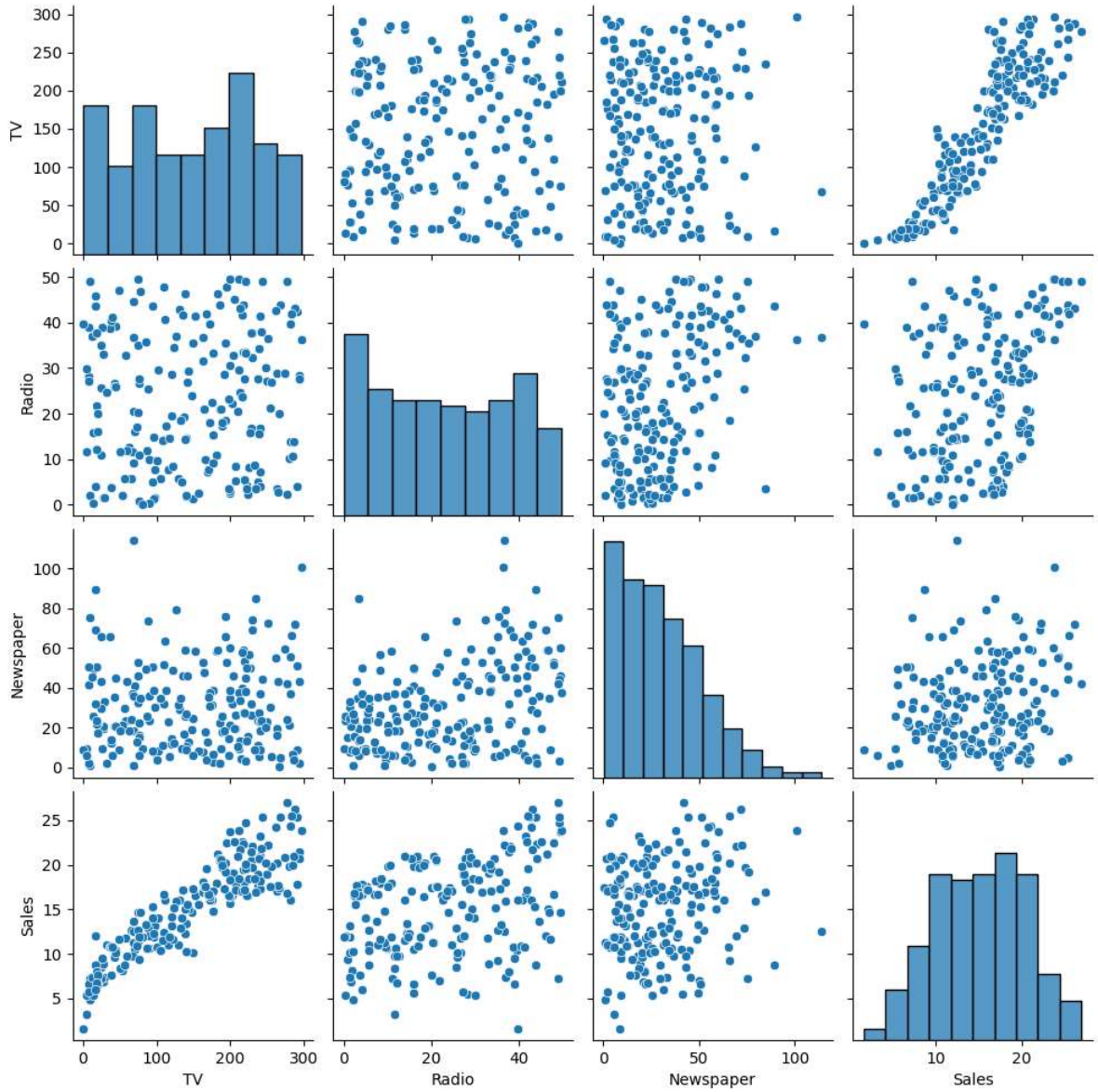
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   TV         200 non-null    float64
 1   Radio      200 non-null    float64
 2   Newspaper  200 non-null    float64
 3   Sales      200 non-null    float64
dtypes: float64(4)
memory usage: 6.4 KB
```

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

Out[8]: <Axes: >



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

sns.pairplot(df)
df.Sales = np.log(df.Sales)
```



In [10]:
```
features=df.columns[0:2]
target=df.columns[-1]
x=df[features].values
y=df[target].values
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=
```

In [11]:
```
print("The dimension of x_train is {}".format(x_train.shape))
print("The dimension of x_train is {}".format(x_test.shape))
```

```
The dimension of x_train is (140, 2)
The dimension of x_train is (60, 2)
```

In [12]:
```
scaler=StandardScaler()
x_train=scaler.fit_transform(x_train)
x_test=scaler.transform(x_test)
```

In [13]:
```
lr=LinearRegression()
lr.fit(x_train,y_train)
actual=y_test
train_score_lr=lr.score(x_train,y_train)
test_score_lr=lr.score(x_test,y_test)
```

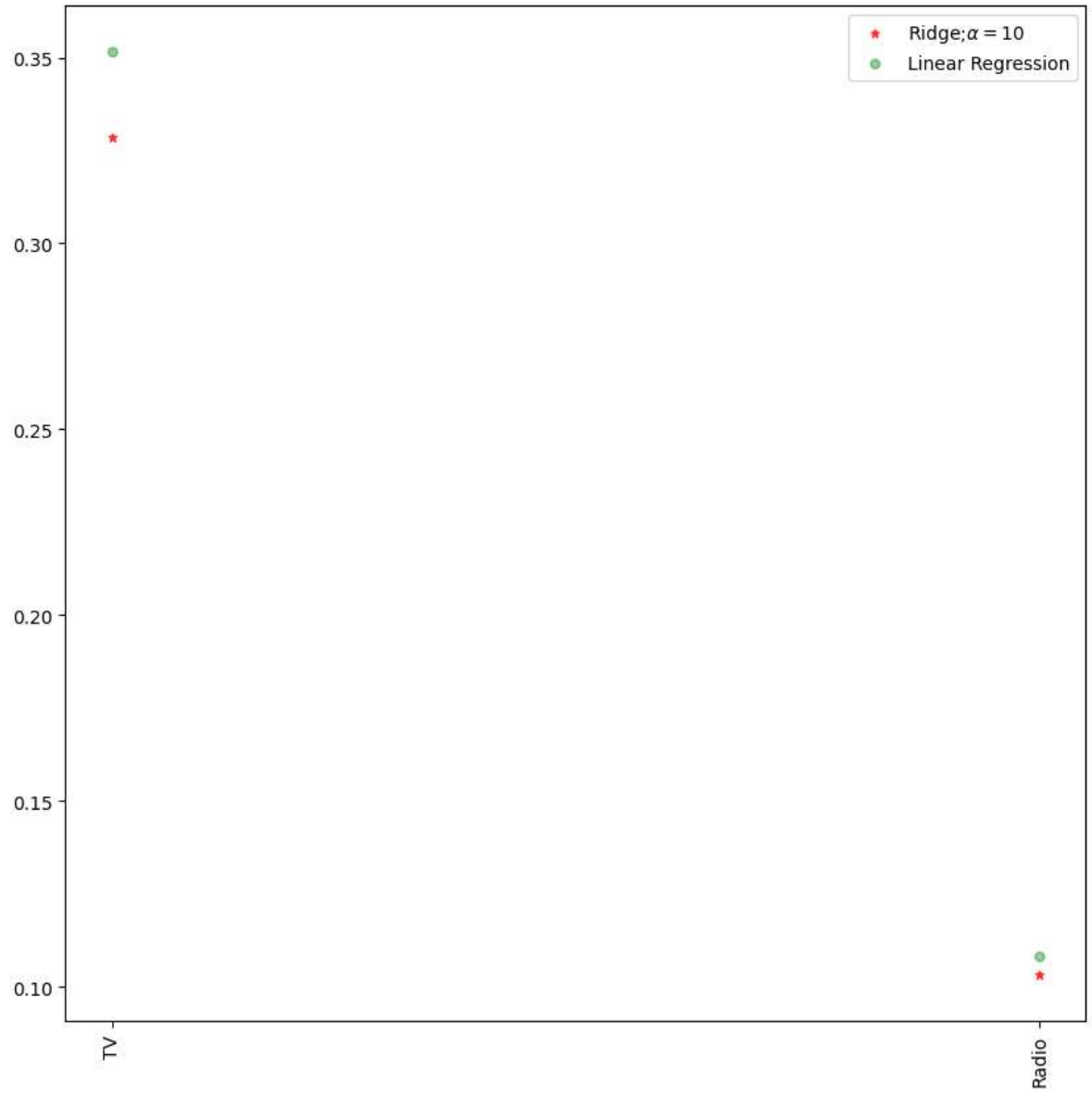
In [14]:
```
print("\nLinear Regression Model:\n")
print("The train score of lr model is {}".format(train_score_lr))
print("The test score of lr model is {}".format(test_score_lr))
```

```
Linear Regression Model:

The train score of lr model is 0.8534150366313791
The test score of lr model is 0.6654095499889279
```

In [15]:
```
r=Ridge(alpha=10)
r.fit(x_train,y_train)
train_score_ridge=r.score(x_train,y_train)
test_score_ridge=r.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 0.8500055285406656
The test score for ridge model is 0.6534250193956134
```

In [16]:
```
plt.figure(figsize=(10,10))
plt.plot(features,r.coef_,alpha=0.7,linestyle='None',marker='*',markersize=5,c
plt.plot(features,lr.coef_,alpha=0.4,linestyle='None',marker='o',markersize=5,
plt.xticks(rotation=90)
plt.legend()
plt.show()
```

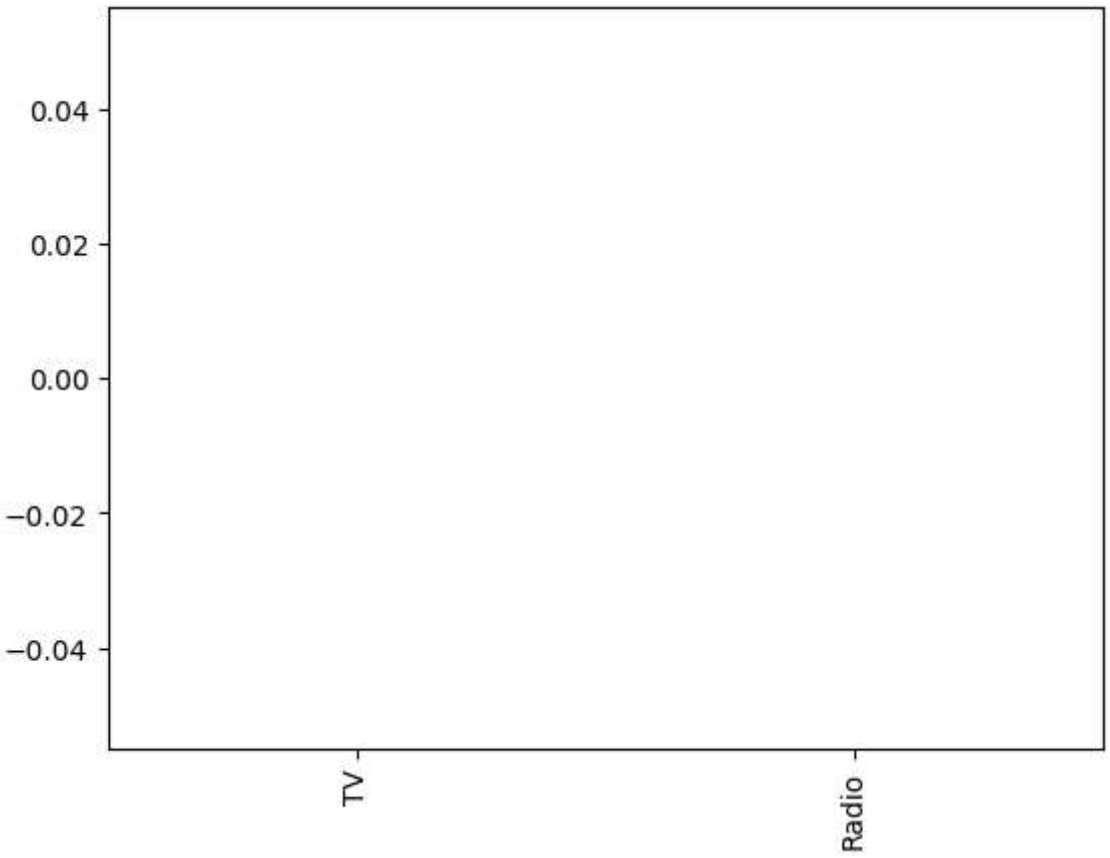


In [17]:
```
l=Lasso(alpha=10)
l.fit(x_train,y_train)
train_score_ls=l.score(x_train,y_train)
test_score_ls=l.score(x_test,y_test)
print("\nRidge model:\n")
print("The train score for ridge model is {}".format(train_score_ls))
print("The test score for ridge model is {}".format(test_score_ls))
```

```
Ridge model:

The train score for ridge model is 0.0
The test score for ridge model is -0.0042092253233847465
```

In [18]:
```
pd.Series(l.coef_,features).sort_values(ascending=True).plot(kind='bar')
```

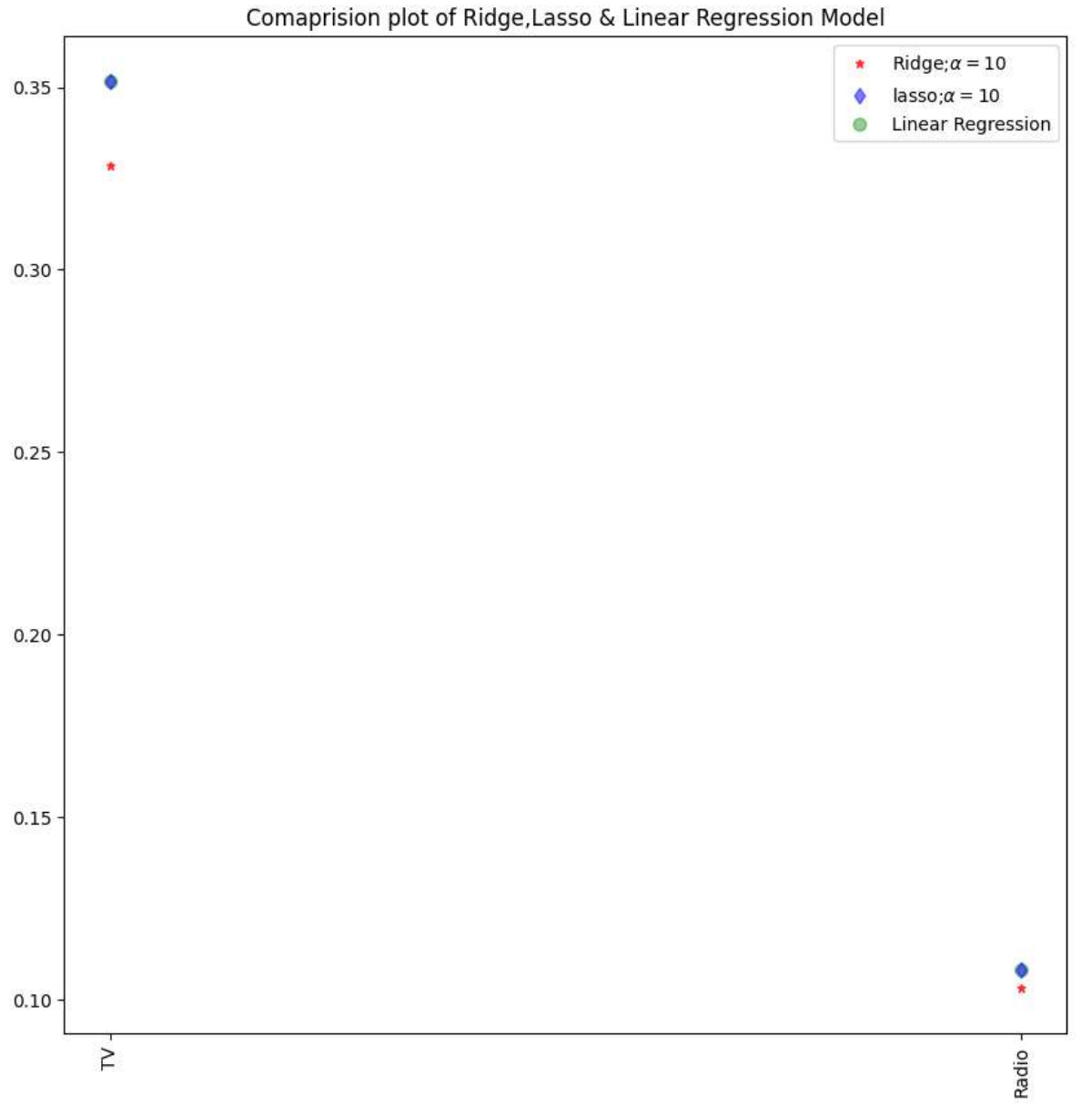
Out[18]: <Axes: >



In [19]:
```
from sklearn.linear_model import LassoCV
lc=LassoCV(alphas=[0.0001,0.001,0.01,0.1,0,1,10],random_state=0).fit(x_train,y
print(lc.score(x_train,y_train))
print(lc.score(x_test,y_test))
```

```
0.8534150366313791
0.6654095499889281

C:\Users\ubinl\AppData\Local\Programs\Python\Python311\Lib\site-packages\skle
arn\linear_model\_coordinate_descent.py:617: UserWarning: Coordinate descent
without L1 regularization may lead to unexpected results and is discouraged.
Set l1_ratio > 0 to add L1 regularization.
  model = cd_fast.enet_coordinate_descent_gram(
C:\Users\ubinl\AppData\Local\Programs\Python\Python311\Lib\site-packages\skle
arn\linear_model\_coordinate_descent.py:617: UserWarning: Coordinate descent
without L1 regularization may lead to unexpected results and is discouraged.
Set l1_ratio > 0 to add L1 regularization.
  model = cd_fast.enet_coordinate_descent_gram(
C:\Users\ubinl\AppData\Local\Programs\Python\Python311\Lib\site-packages\skle
arn\linear_model\_coordinate_descent.py:617: UserWarning: Coordinate descent
without L1 regularization may lead to unexpected results and is discouraged.
Set l1_ratio > 0 to add L1 regularization.
  model = cd_fast.enet_coordinate_descent_gram(
C:\Users\ubinl\AppData\Local\Programs\Python\Python311\Lib\site-packages\skle
arn\linear_model\_coordinate_descent.py:617: UserWarning: Coordinate descent
without L1 regularization may lead to unexpected results and is discouraged.
Set l1_ratio > 0 to add L1 regularization.
  model = cd_fast.enet_coordinate_descent_gram(
C:\Users\ubinl\AppData\Local\Programs\Python\Python311\Lib\site-packages\skle
arn\linear_model\_coordinate_descent.py:617: UserWarning: Coordinate descent
without L1 regularization may lead to unexpected results and is discouraged.
Set l1_ratio > 0 to add L1 regularization.
  model = cd_fast.enet_coordinate_descent_gram(
C:\Users\ubinl\AppData\Local\Programs\Python\Python311\Lib\site-packages\skle
arn\linear_model\_coordinate_descent.py:1712: UserWarning: With alpha=0, this
algorithm does not converge well. You are advised to use the LinearRegression
estimator
  model.fit(X, y)
C:\Users\ubinl\AppData\Local\Programs\Python\Python311\Lib\site-packages\skle
arn\linear_model\_coordinate_descent.py:631: UserWarning: Coordinate descent
with no regularization may lead to unexpected results and is discouraged.
  model = cd_fast.enet_coordinate_descent(
C:\Users\ubinl\AppData\Local\Programs\Python\Python311\Lib\site-packages\skle
arn\linear_model\_coordinate_descent.py:631: ConvergenceWarning: Objective di
d not converge. You might want to increase the number of iterations, check th
e scale of the features or consider increasing regularisation. Duality gap:
1.728e+00, tolerance: 2.358e-03 Linear regression models with null weight for
the l1 regularization term are more efficiently fitted using one of the solve
rs implemented in sklearn.linear_model.Ridge/RidgeCV instead.
  model = cd_fast.enet_coordinate_descent(
```

In [20]:
```
plt.figure(figsize=(10,10))
plt.plot(features,r.coef_,alpha=0.7,linestyle='None',marker='*',markersize=5,c
plt.plot(features,lc.coef_,alpha=0.5,linestyle='None',marker='d',markersize=6,
plt.plot(features,lr.coef_,alpha=0.4,linestyle='None',marker='o',markersize=7,
plt.xticks(rotation=90)
plt.title("Comaprision plot of Ridge,Lasso & Linear Regression Model")
plt.legend()
plt.show()
```



In [21]:
```
from sklearn.linear_model import RidgeCV
rc=RidgeCV(alphas=[0.0001,0.001,0.01,0.1,1,10]).fit(x_train,y_train)
print("\nRidge model:\n")
print("The train score for ridge model is {}".format(rc.score(x_train,y_train)
print("The test score for ridge model is {}".format(rc.score(x_test,y_test)))
```

```
Ridge model:

The train score for ridge model is 0.8534146479788303
The test score for ridge model is 0.66530380572686
```

In [22]:
```
#ELASTICNET
from sklearn.linear_model import ElasticNet
e=ElasticNet()
e.fit(x,y)
print(e.coef_)
print(e.intercept_)
```

```
[0.00414142 0.00404556]
1.9379057734206566
```

In [23]:
```
y_pred_elastic=e.predict(x_train)
```

In [24]:
```
mse=np.mean((y_pred_elastic-y_train)**2)
print("Mean squared error on test set",mse)
```

```
Mean squared error on test set 0.6708648977649432
```