In [3]:

```
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
```

In [4]:

```
df=pd.read_csv(r"C:\Users\hp\Downloads\Advertising.csv")
df
```

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
...	...	...	...	...
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 [5]:

```
df.describe()
```

Out[5]:

	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 [6]:

```
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 [7]:

```
df.isnull().any()
```

Out[7]:

```
TV           False
Radio        False
Newspaper    False
Sales        False
dtype: bool
```

In [8]:

```
df.head()
```

Out[8]:

	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 [9]:

```
df.tail()
```

Out[9]:

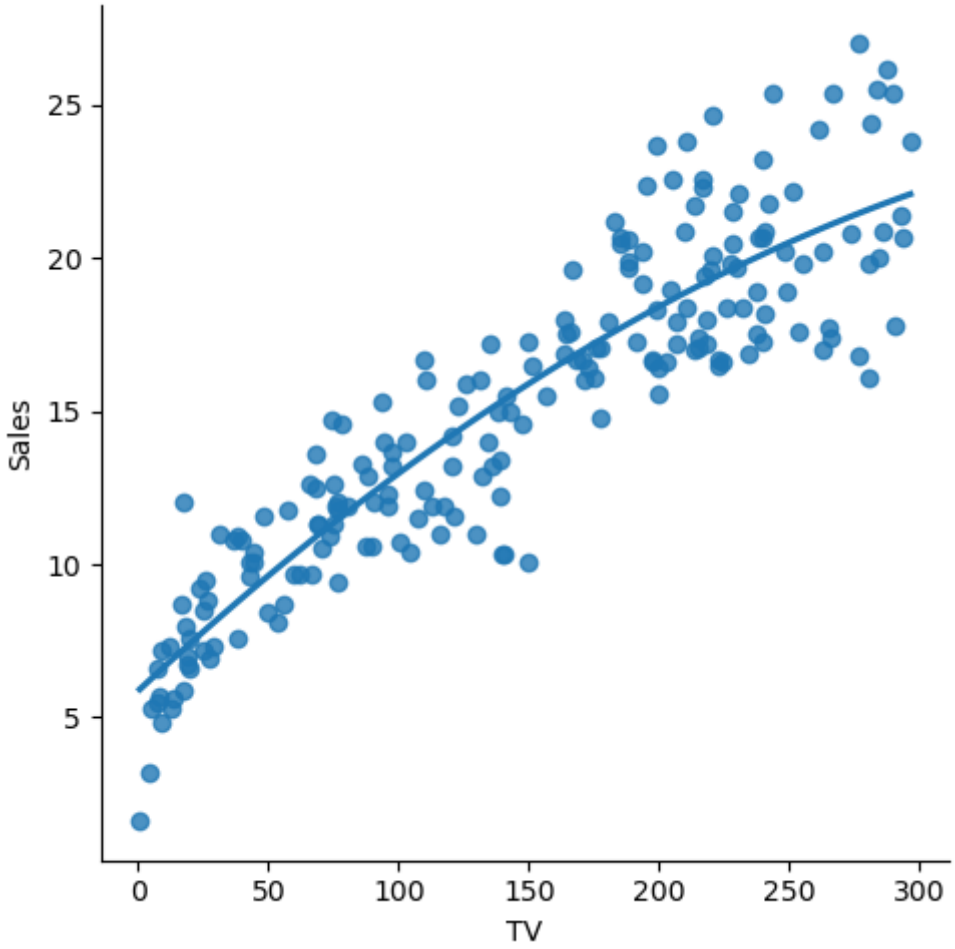
	TV	Radio	Newspaper	Sales
195	38.2	3.7	13.8	7.6
196	94.2	4.9	8.1	14.0
197	177.0	9.3	6.4	14.8
198	283.6	42.0	66.2	25.5
199	232.1	8.6	8.7	18.4

In [10]:

```
sns.lmplot(x="TV",y="Sales",data=df,order=2,ci=None)
```

Out[10]:

```
<seaborn.axisgrid.FacetGrid at 0x1ff98d7e770>
```



In [11]:

```
x=np.array(df['TV']).reshape(-1,1)
y=np.array(df['Sales']).reshape(-1,1)
```

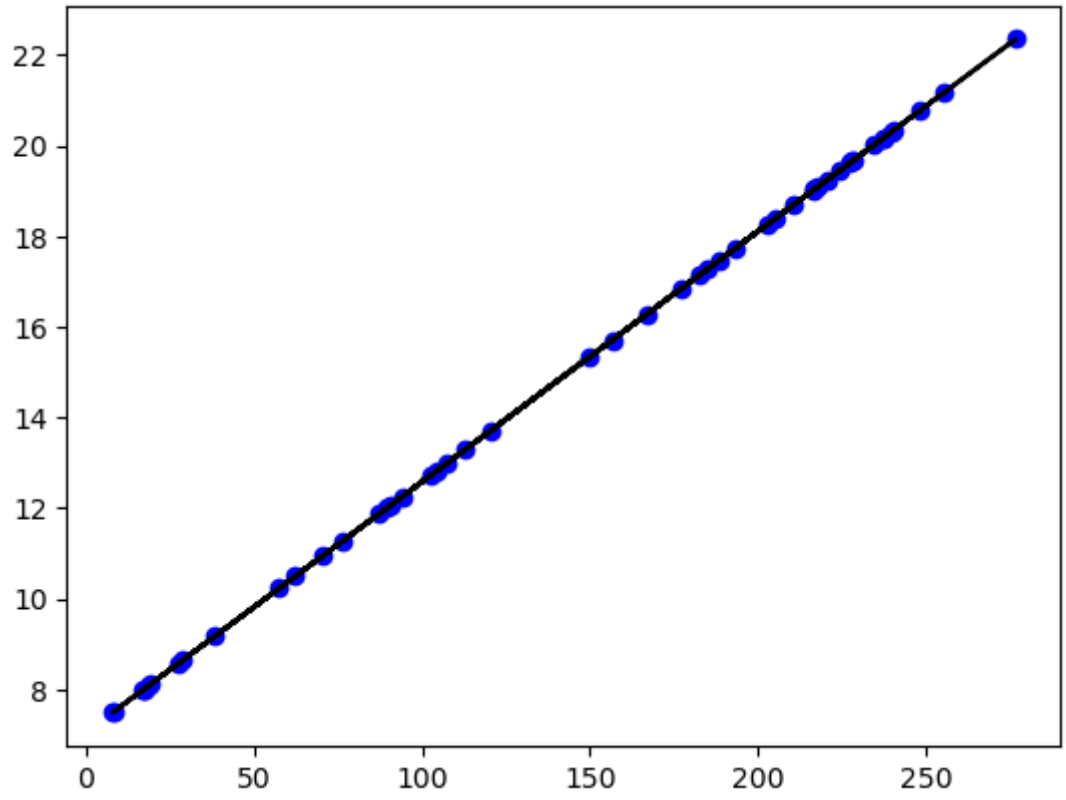
In [12]:

```
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))
```

```
0.809792104152377
```

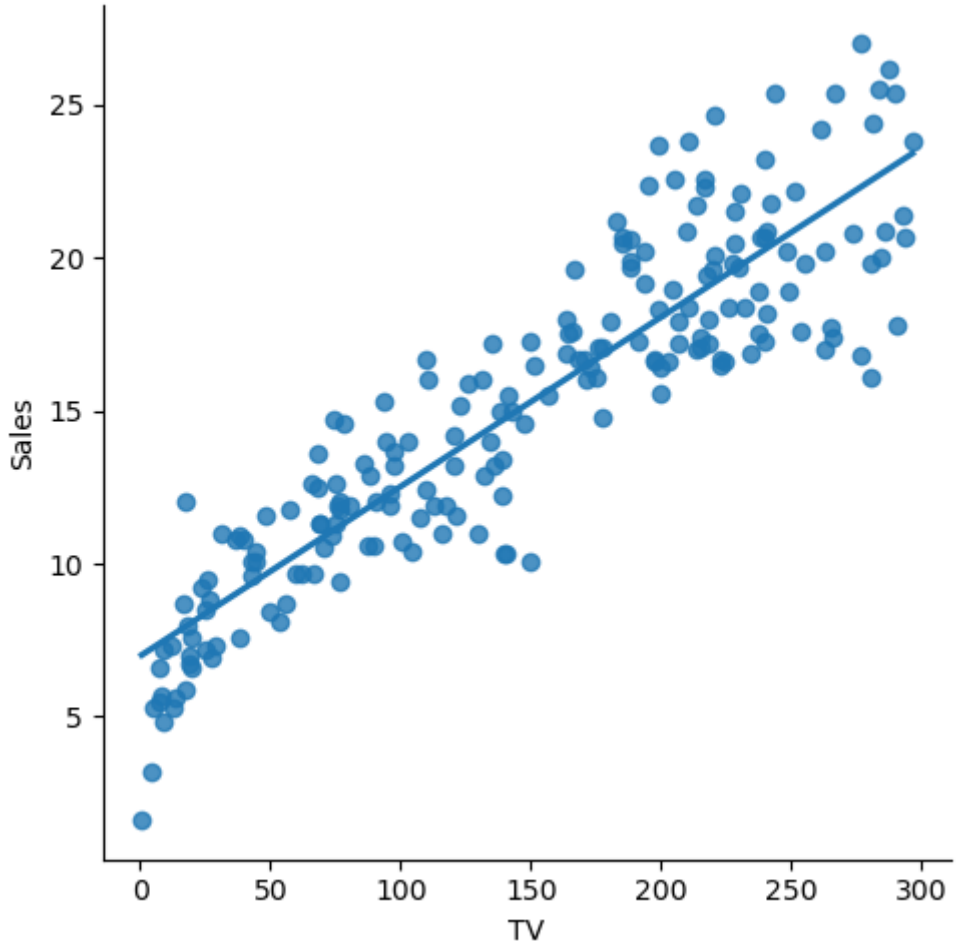
In [13]:

```
y_pred=regr.predict(x_test)
plt.scatter(x_test,y_pred,color='b')
plt.plot(x_test,y_pred,color='k')
plt.show()
```



In [14]:

```
df500=df[:][:500]
sns.lmplot(x="TV",y="Sales",data=df500,order=1,ci=None)
x=np.array(df500['TV']).reshape(-1,1)
y=np.array(df500['Sales']).reshape(-1,1)
df500.dropna(inplace=True)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)
```



In [15]:

```
regr=LinearRegression()
regr.fit(x_train,y_train)
print("Regression:",regr.score(x_test,y_test))
```

```
Regression: 0.8064318173784625
```

In [16]:

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
model=LinearRegression()
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
r2=r2_score(y_test,y_pred)
print("R2 Score:",r2)
```

```
R2 Score: 0.8064318173784625
```

In [17]:

```
features=df.iloc[:,0:4]
```

In [18]:

```
df.tail()
```

Out[18]:

	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 [19]:

```
from sklearn.linear_model import Lasso,Ridge
from sklearn.preprocessing import StandardScaler
```

In [20]:

```
features= df.columns[0:2]
target= df.columns[-1]
```

In [21]:

```
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=17)

print("The dimension of x_train is {}")
print("The dimension of x_test is {}")
scaler =StandardScaler()
x_train=scaler.fit_transform(x_train)
x_test=scaler.transform(x_test)
```

```
The dimension of x_train is {}
The dimension of x_test is {}
```

In [22]:

```
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)
```

In [23]:

```
print("\n Ridge 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))
```

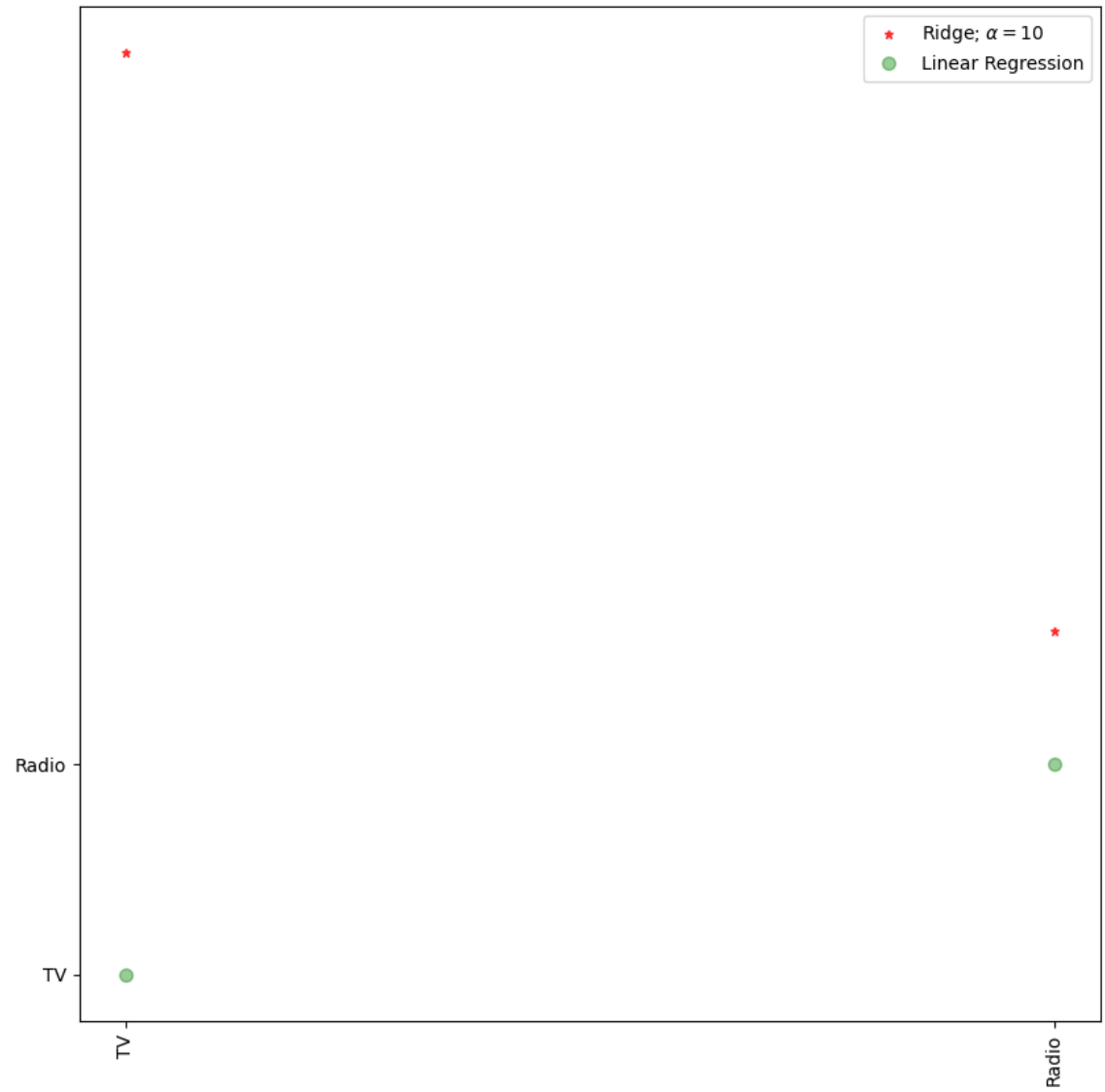
```
 Ridge Model:

the train score for ridge model is0.913873609964124
the test score for ridge model is0.8486032178989196
```

In [24]:

```
lr=LinearRegression()
```

In [25]:

```
plt.figure(figsize= (10,10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker="*",markersize=5,colo
plt.plot(features,alpha=0.4,linestyle='none',marker='o',markersize=7,color="green",label
plt.xticks(rotation = 90)
plt.legend()
plt.show()
```



In [26]:

```
print("\n Lasso 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))
```

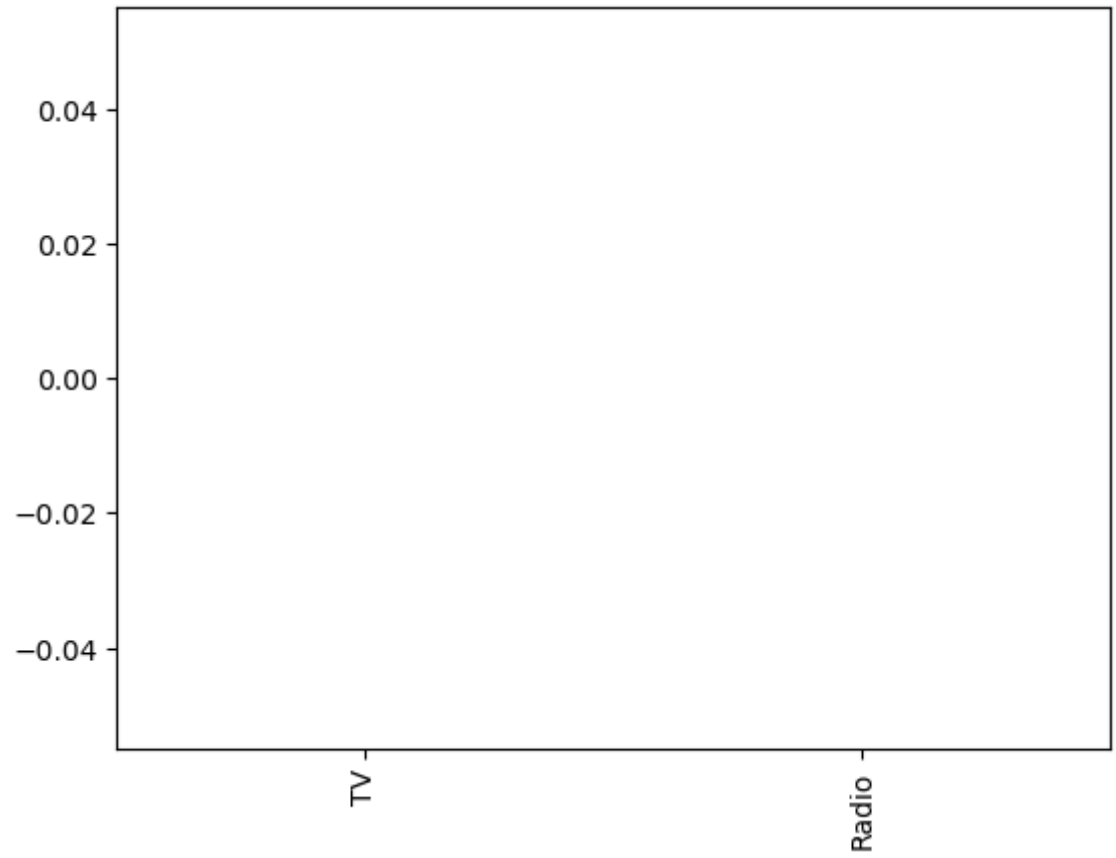
```
 Lasso Model:

The train score for ls model is 0.0
The test score for ls model is-0.0064111102763571015
```

In [27]:

```
pd.Series(lasso.coef_,features).sort_values(ascending=True).plot(kind="bar")
```

Out[27]:

```
<Axes: >
```



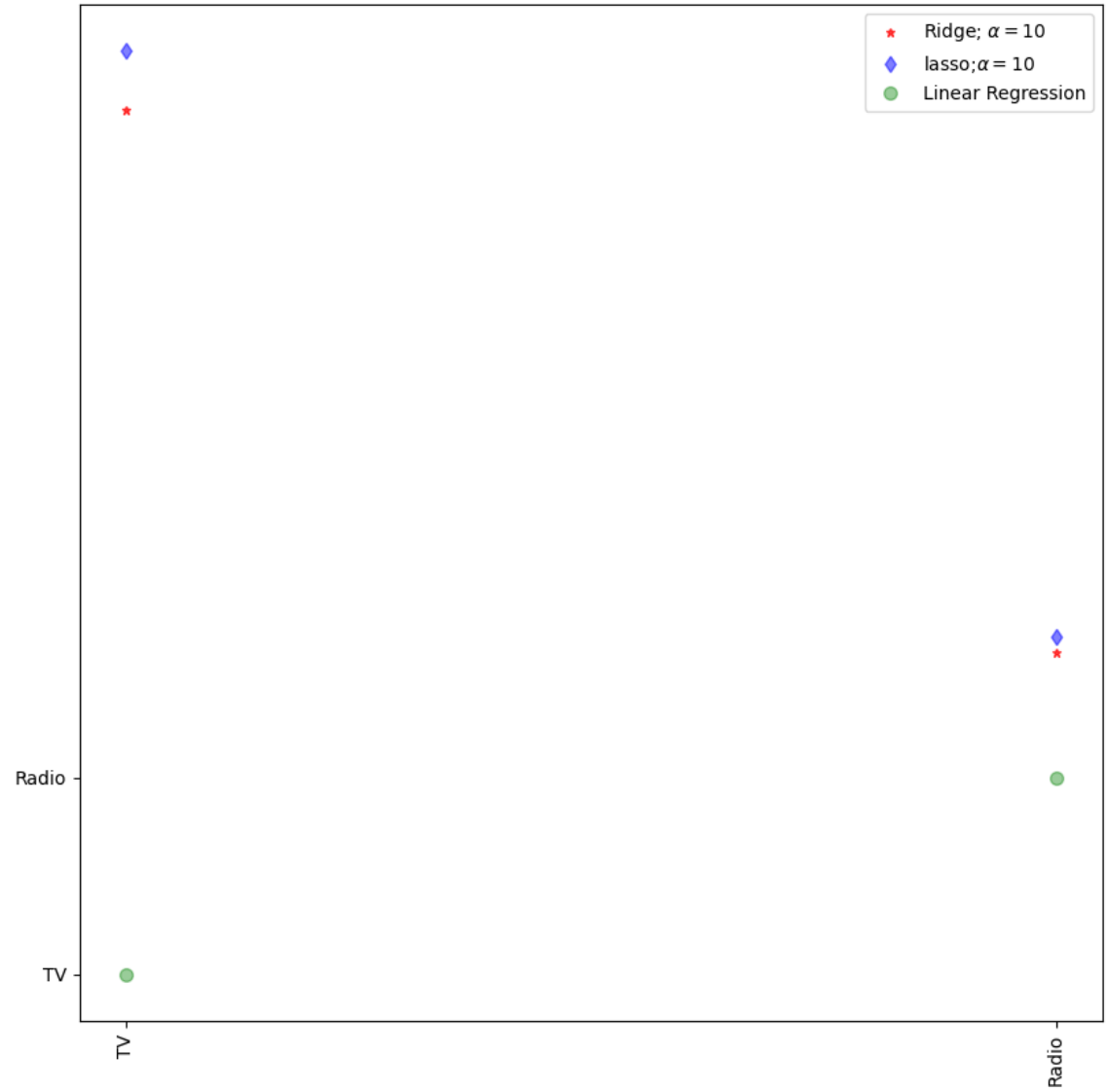
In [28]:

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

```
0.9174763126220602
0.8567421168446285
```

In [29]:

```
plt.figure(figsize= (10,10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker="*",markersize=5,colo
plt.plot(lasso_cv.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color='blue',
plt.plot(features,alpha=0.4,linestyle='none',marker='o',markersize=7,color="green",label
plt.xticks(rotation = 90)
plt.legend()
plt.show()
```



In [30]:

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

```
[0.05440081 0.1046715 ]
4.696191158087224
```

In [37]:

```
y_pred_elastic = regr.predict(x_train)
mean_squared_error=np.mean((y_pred_elastic - y_train)**2)
print(mean_squared_error)
```

```
139.54160889209743
```