ADVERSTING DATA SET

In [1]:
```
import numpy as np
import pandas as pd
```

In [2]:
```
df=pd.read_csv(r"C:\Users\manis\OneDrive\Pictures\Documents\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.shape
```

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

In [5]:
```
df.describe
```

Out[5]:
```
<bound method NDFrame.describe of        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 x 4 columns]>
```

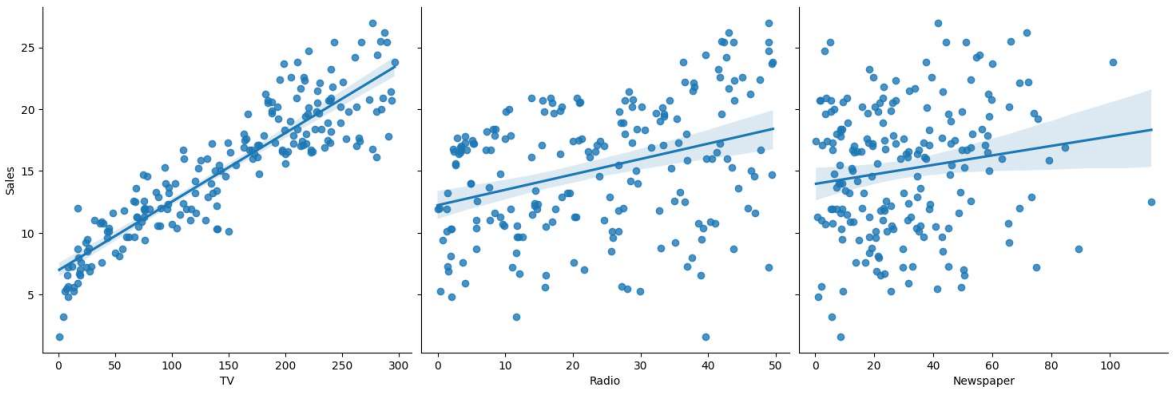
In [6]:
```
df.info()
```

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

In [8]:
```
sns.pairplot(df,x_vars=['TV','Radio','Newspaper'],y_vars='Sales',height=5,aspect
```

Out[8]:
```
<seaborn.axisgrid.PairGrid at 0x1d0455425d0>
```



In [9]:
```
features=df.columns[0:3]
```

In [10]:
```
target=df.columns[-1]
```

In [11]:
```
x=df[features].values
y=df[target].values
```

In [12]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.6)
```

In [13]:
```
from sklearn.linear_model import LinearRegression
```

In [14]:
```
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[14]:
```
▾ LinearRegression
LinearRegression()
```

In [15]:
```
coeff_df=pd.DataFrame(lr.coef_)
coeff_df
```

Out[15]:

	0
0	0.051954
1	0.088201
2	0.007692

In [16]:
```
predictions=lr.predict(x_test)
```

In [17]:
```
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: 1.1001388382182984
MSE: 2.0198049179188313
RMSE: 1.4211984090614622
```

RIDGE

In [18]:
```
from sklearn.linear_model import Ridge,Lasso
from sklearn import preprocessing
from sklearn.preprocessing import StandardScaler
```

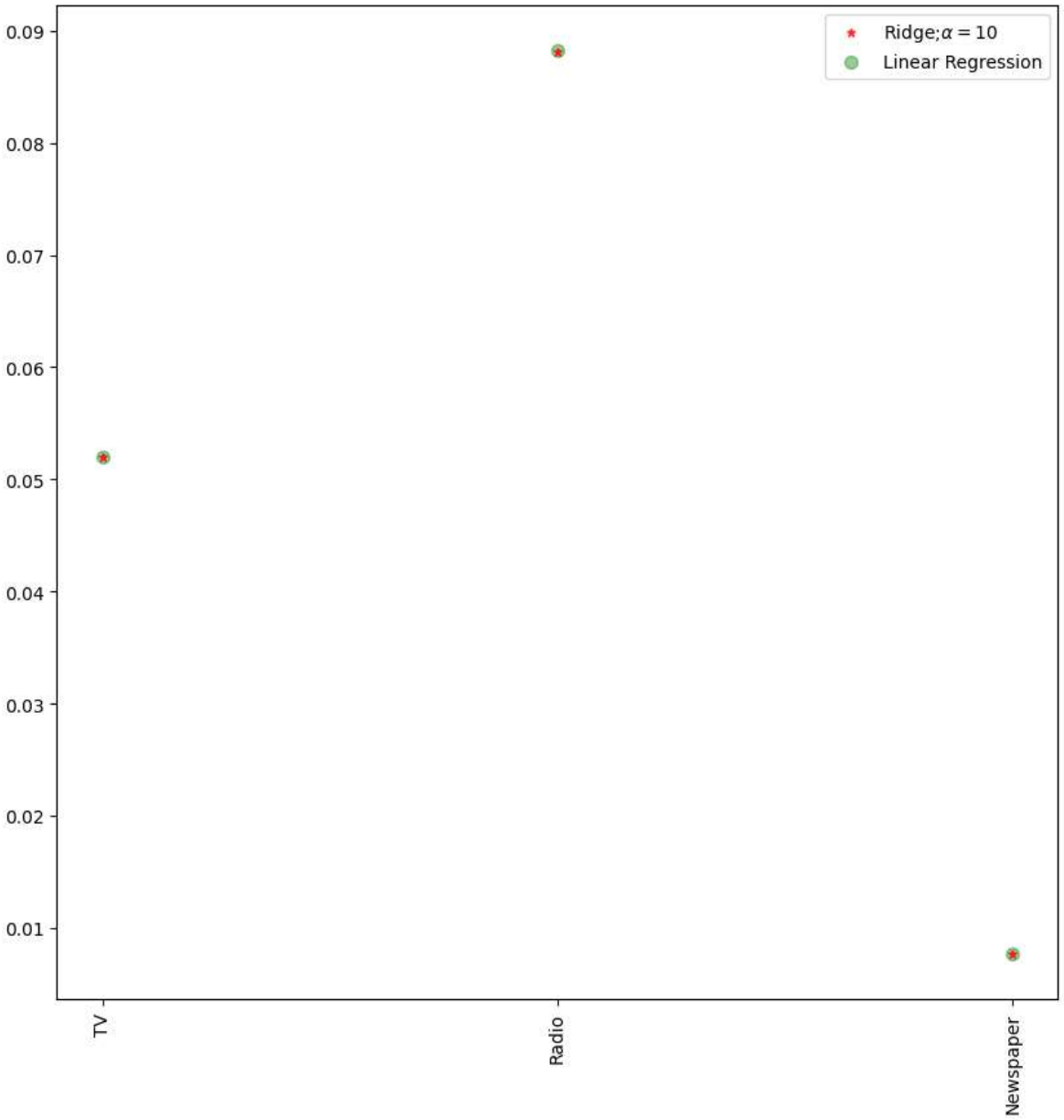
In [19]:
```
#ridge regression model
ridgeReg=Ridge(alpha=10)
ridgeReg.fit(x_train,y_train)
train_score_ridge=ridgeReg.score(x_train,y_train)
test_score_ridge=ridgeReg.score(x_test,y_test)
print("\nRidge Model\n")
print("Train Score for ridge model is",(train_score_ridge))
print("Test Score for ridge model is",(test_score_ridge))
```

```
Ridge Model

Train Score for ridge model is 0.8387910376366433
Test Score for ridge model is 0.9305874853065024
```

In [20]:
```
plt.figure(figsize=(10,10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersiz
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,cc
plt.xticks(rotation=90)
```

```
plt.legend()
plt.show()
```



In [21]:
```
from sklearn.linear_model import RidgeCV
```

In [22]:
```
ridge_CV=RidgeCV(alphas=[0.0001,0.001,0.01,0.1,1,10]).fit(x_train,y_train)
print("The Train score for ridge model is {}".format(ridge_CV.score(x_train,y_tr
print("The Test score for ridge model is {}".format(ridge_CV.score(x_test,y_test
```

```
The Train score for ridge model is 0.8387910376366433
The Test score for ridge model is 0.9305874853058705
```

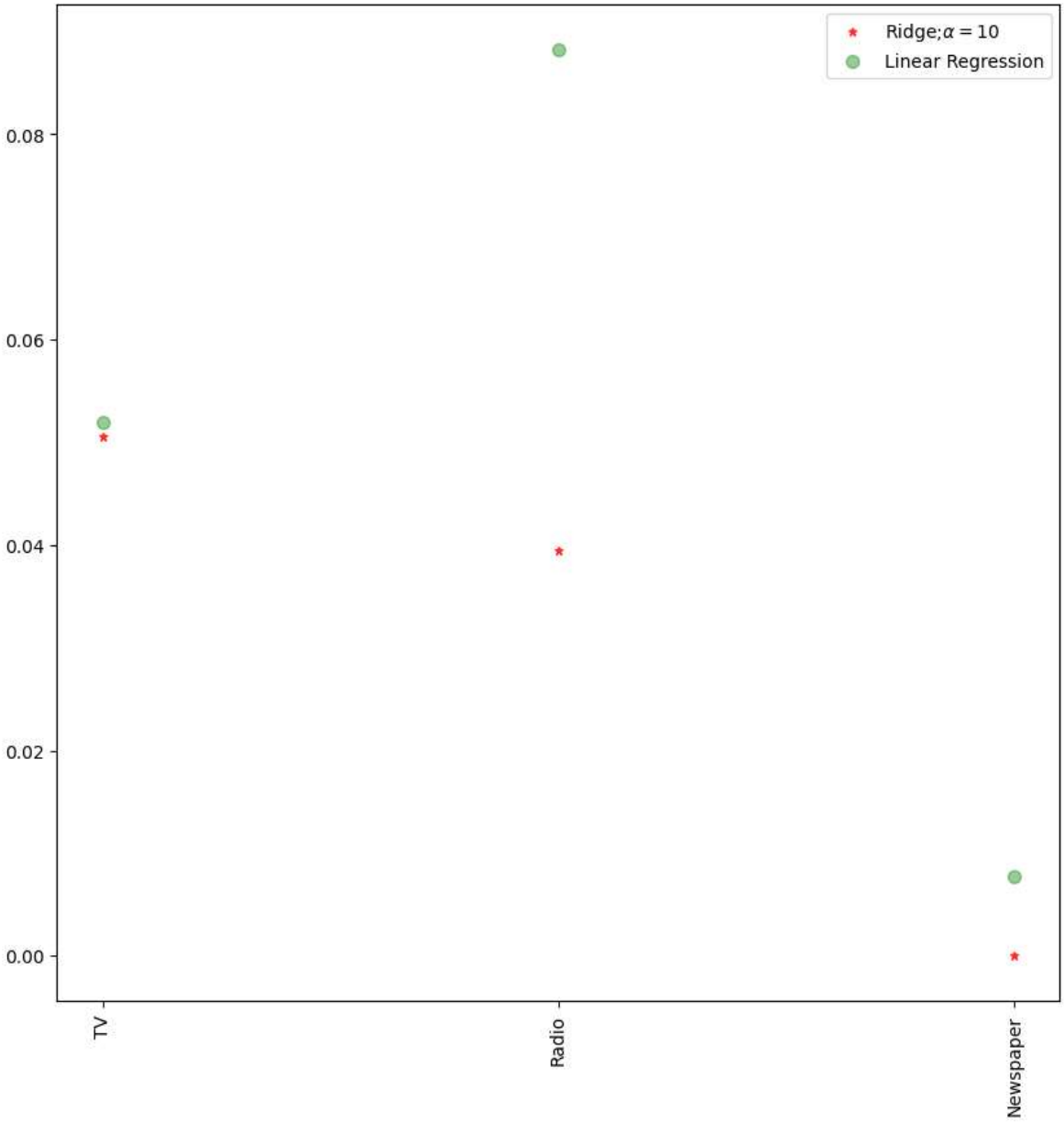
LASSO

In [23]:
```
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("\nLasso Model\n")
print("Train Score for lasso model is",(train_score_lasso))
print("Test Score for lasso model is",(test_score_lasso))
```

```
Lasso Model

Train Score for lasso model is 0.8163237225268702
Test Score for lasso model is 0.8803178322416207
```

In [24]:
```
plt.figure(figsize=(10,10))
plt.plot(features,lassoReg.coef_,alpha=0.7,linestyle='none',marker='*',markersiz
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,cc
plt.xticks(rotation=90)
plt.legend()
plt.show()
```



In [25]:
```
from sklearn.linear_model import LassoCV
```

In [26]:
```
lasso_CV=LassoCV(alphas=[0.0001,0.001,0.01,0.1,1,10]).fit(x_train,y_train)
print("The Train score for lasso model is {}".format(lasso_CV.score(x_train,y_tr
print("The Test score for lasso model is {}".format(lasso_CV.score(x_test,y_test
```

```
The Train score for lasso model is 0.8385568957893701
The Test score for lasso model is 0.9280821516893354
```

ELASTICNET

In [27]:
```
from sklearn.linear_model import ElasticNet
regr=ElasticNet()
regr.fit(x,y)
print(regr.coef_)
print(regr.intercept_)
```

```
[0.05440081 0.1046715  0.        ]
4.696191158087226
```

In [28]:
```
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)
```

```
Mean Squared Error on test set 4.184983297184837
```

In [ ]:
```

```