In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import warnings
warnings.simplefilter(action='ignore')
```

In [2]:
```
df=pd.read_csv(r"C:\Users\pappu\Downloads\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.describe()
```

Out[4]:

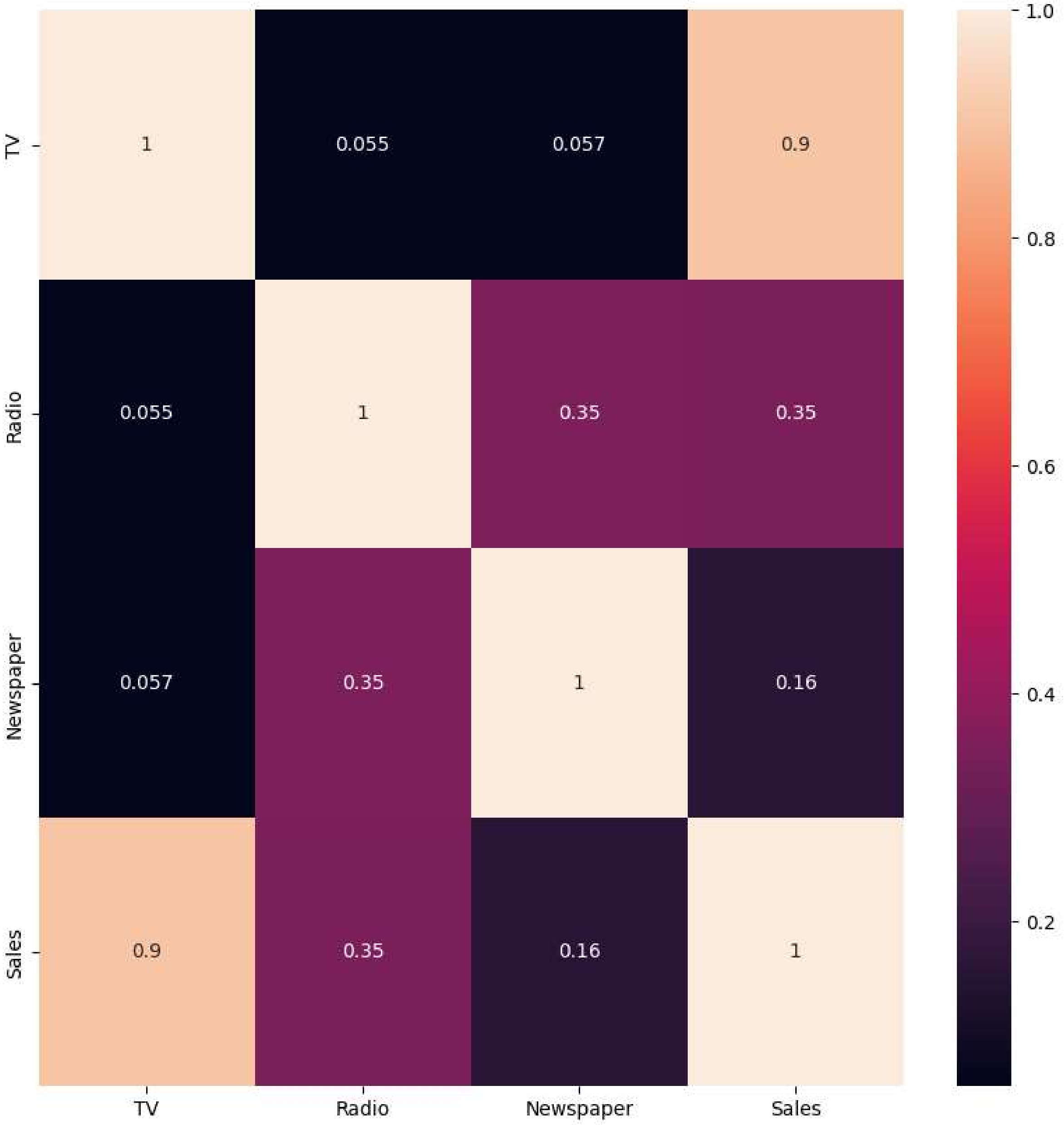
	TV	Radio	Newspaper	Sales
count	200.000000	200.000000	200.000000	200.000000
mean	147.042500	23.264000	30.554000	15.130500
std	85.854236	14.846809	21.778621	5.283892
min	0.700000	0.000000	0.300000	1.600000
25%	74.375000	9.975000	12.750000	11.000000
50%	149.750000	22.900000	25.750000	16.000000
75%	218.825000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

In [5]:
```
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
```

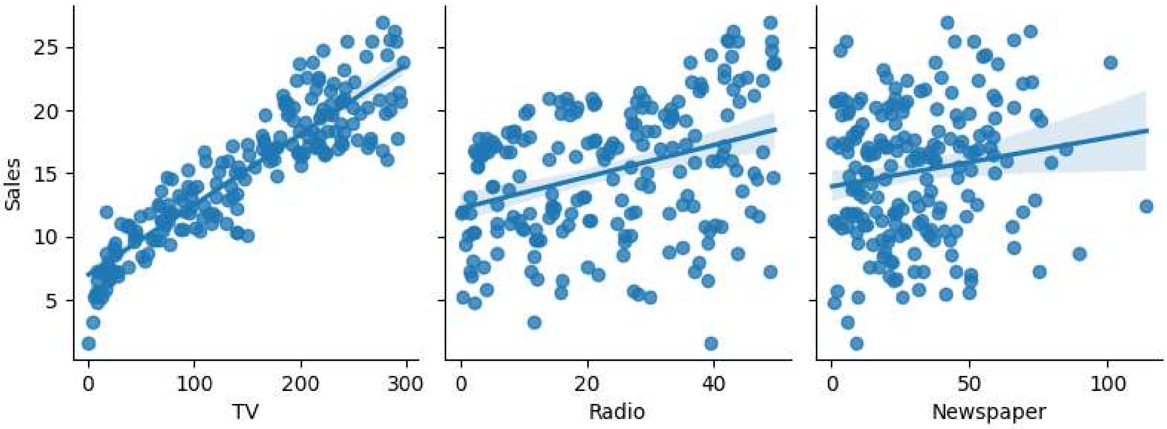
In [6]:
```
plt.figure(figsize=(10,10))
sns.heatmap(df.corr(),annot=True)
```

Out[6]:
```
<Axes: >
```

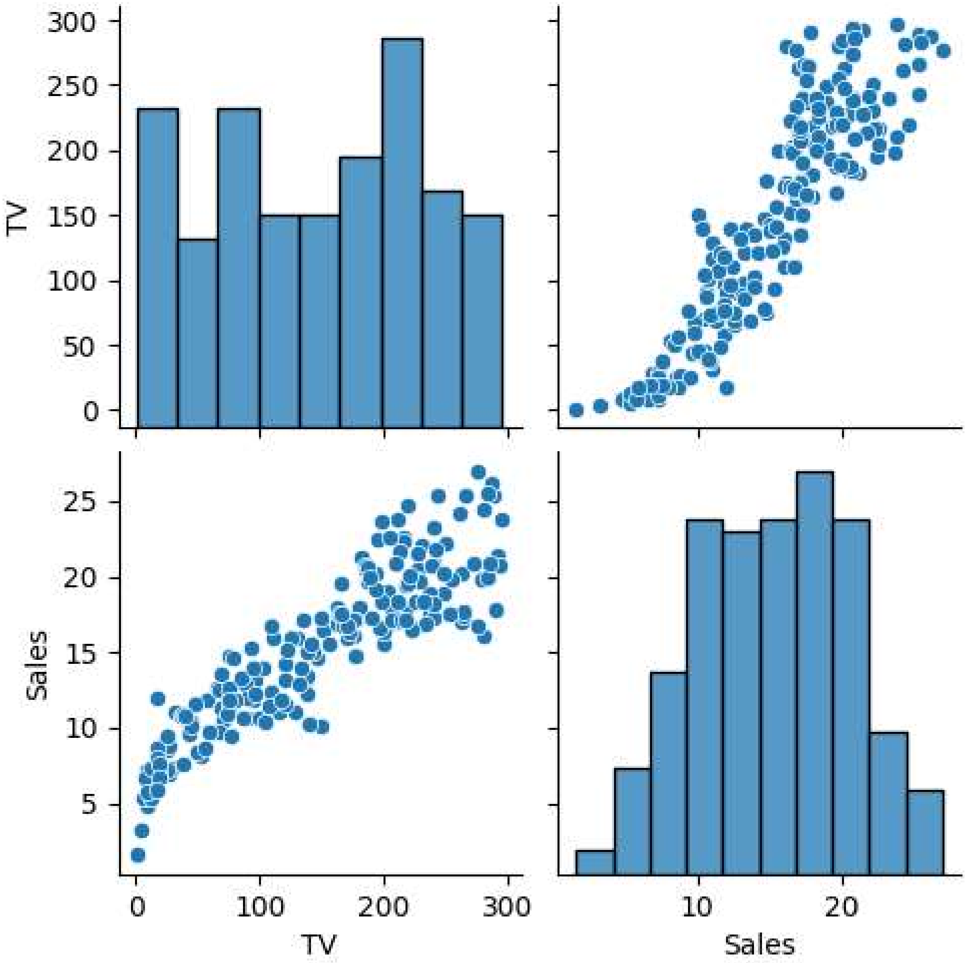


In [7]:
```
sns.pairplot(df,x_vars=["TV","Radio","Newspaper"],y_vars='Sales',height=3,aspe
```

Out[7]:
```
<seaborn.axisgrid.PairGrid at 0x146c8e8a4d0>
```



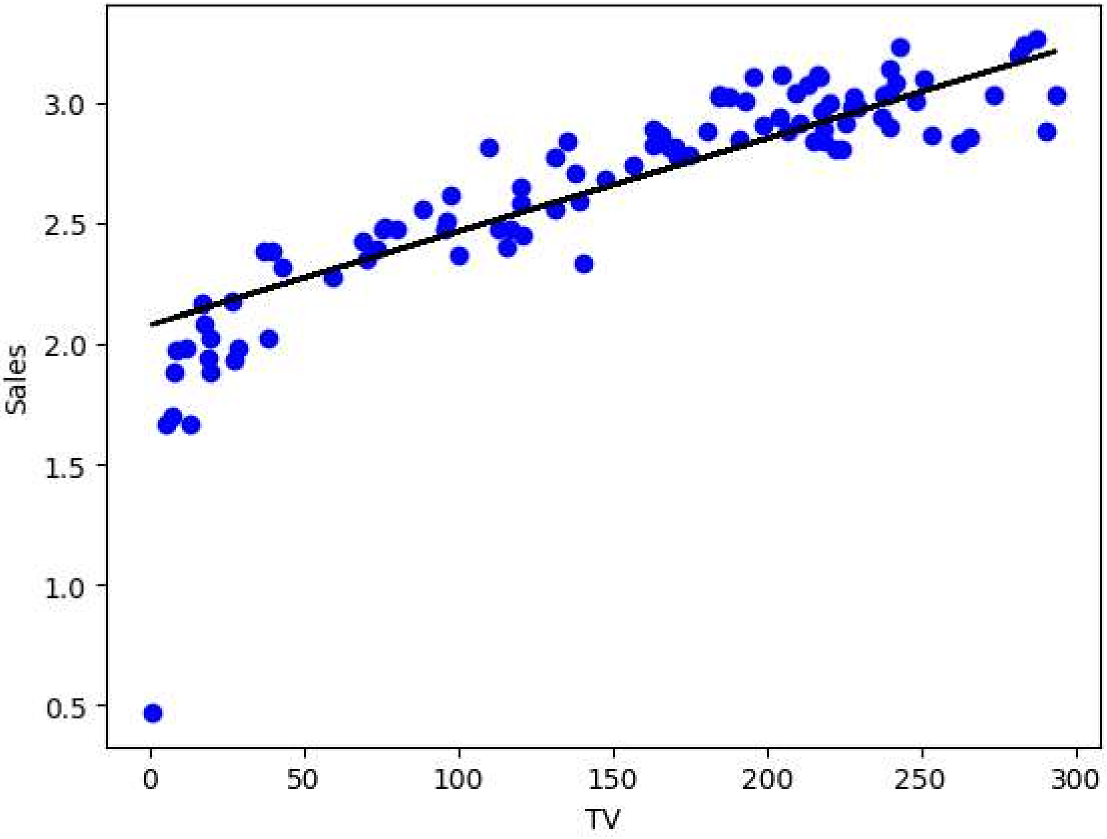
In [8]:
```
df.drop(columns = ["Radio", "Newspaper"], inplace = True)
sns.pairplot(df)
df.Sales = np.log(df.Sales)
```



```
In [9]: x=np.array(df['TV']).reshape(-1,1)
        y=np.array(df['Sales']).reshape(-1,1)
```

```
In [10]: x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.5)
         reg=LinearRegression()
         reg.fit(x_train,y_train)
         print(reg.score(x_test,y_test))

         0.7496231155715057
```

```
In [11]: y_pred=reg.predict(x_test)
         plt.scatter(x_test,y_test,color='b')
         plt.plot(x_test,y_pred,color='k')
         plt.xlabel('TV')
         plt.ylabel('Sales')
         plt.show()
```



Ridge Regression

```
In [12]: from sklearn.linear_model import Ridge,Lasso,RidgeCV
         from sklearn.preprocessing import StandardScaler
```

In [13]:

```
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("\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 is 0.7209580239837404
The test score for ridge model is 0.7496202217703465
```

In [14]:

```
features=['Tv']
target=['Sales']
```

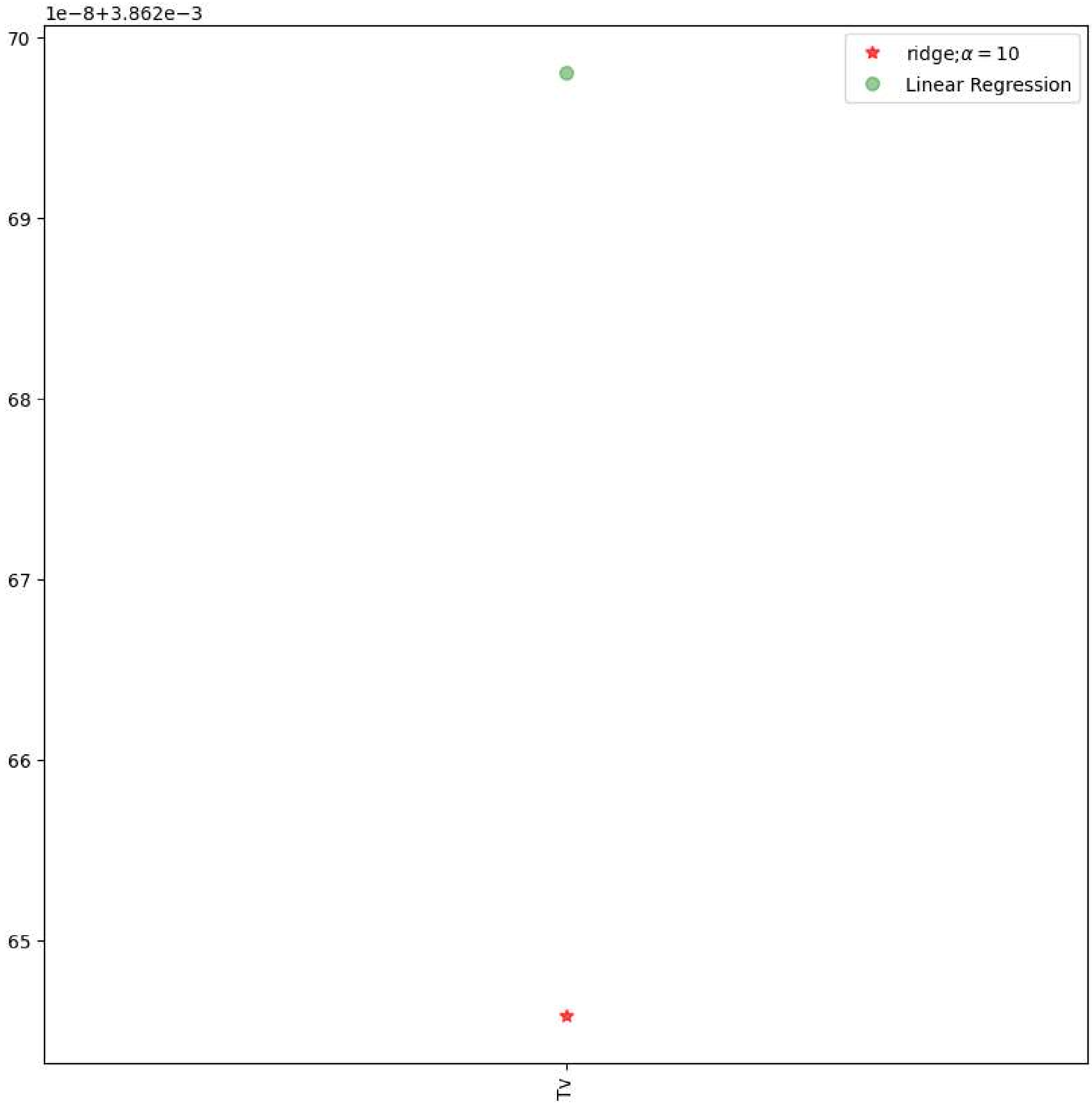
In [15]:

```
plt.figure(figsize=(10,10))
```

Out[15]:

```
<Figure size 1000x1000 with 0 Axes>

<Figure size 1000x1000 with 0 Axes>
```

In [16]:
```
plt.figure(figsize=(10,10))
plt.plot(features,ridgereg.coef_,alpha=0.7,linestyle='none',marker='*',markers
plt.plot(features,reg.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7
plt.xticks(rotation=90)
plt.legend()
plt.show()
```



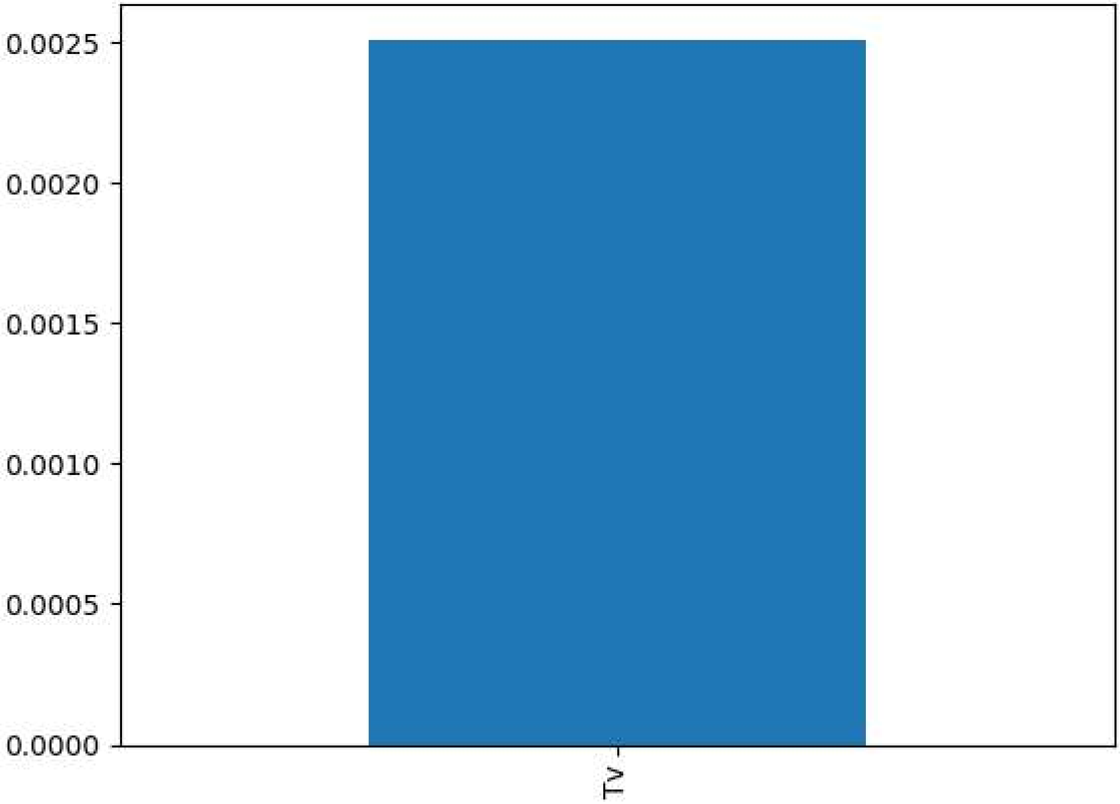
Lasso Regression

In [17]:
```
print("\n Lasso Model:\n")
lasso=Lasso(alpha=10)
lasso.fit(x_train,y_train)
train_score_Lasso=lasso.score(x_train,y_train)
test_score_Lasso=lasso.score(x_test,y_test)

print("The train score for lasso model is {}".format(train_score_Lasso))
print("The test score for lasso model is {}".format(test_score_Lasso))
```

```
 Lasso Model:

The train score for lasso model is 0.6325500243138178
The test score for lasso model is 0.6084953871294831
```

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

Out[18]:
```
<Axes: >
```



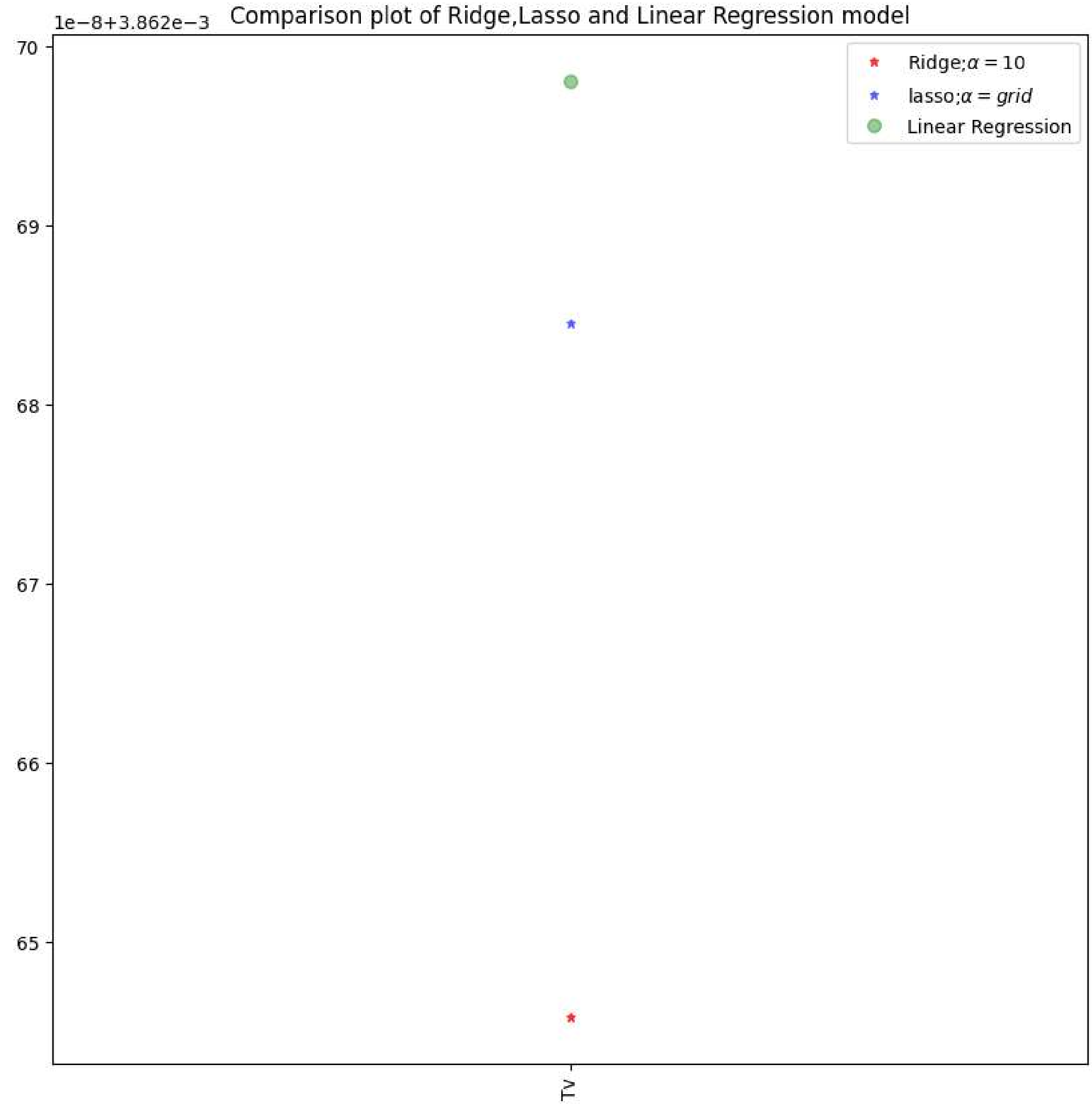
In [19]:
```
from sklearn.linear_model import LassoCV
```

In [20]:
```
lasso_cv=LassoCV(alphas=[0.0001,0.001,0.01,0.1,1,10],random_state=0).fit(x_tra
print(lasso_cv.score(x_train,y_train))
print(lasso_cv.score(x_test,y_test))
```

```
0.7209580241068045
0.7496223664145654
```

In [21]:

```
#Linear CV model
plt.figure(figsize=(10,10))
plt.plot(features,ridgereg.coef_,alpha=0.7,linestyle='none',marker='*',markers
plt.plot(lasso_cv.coef_,alpha=0.5,linestyle='none',marker='*',markersize=5,col
plt.plot(features,reg.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7
plt.xticks(rotation=90)
plt.legend()
plt.title('Comparison plot of Ridge,Lasso and Linear Regression model')
plt.show()
```



Elastic Net

In [22]:

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

```
[0.00417976]
[2.02638392]
```

In [23]:

```
y_pred_elastic=regr.predict(x_train)
```

In [24]:

```
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 0.28219900917896384
```

In [ ]: