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

In [2]:
```
data=pd.read_csv(r"C:\Users\MSI\Downloads\Advertising.csv")
data
```

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]:
```
data.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]:
```
data.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]:
```
data.shape
```

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

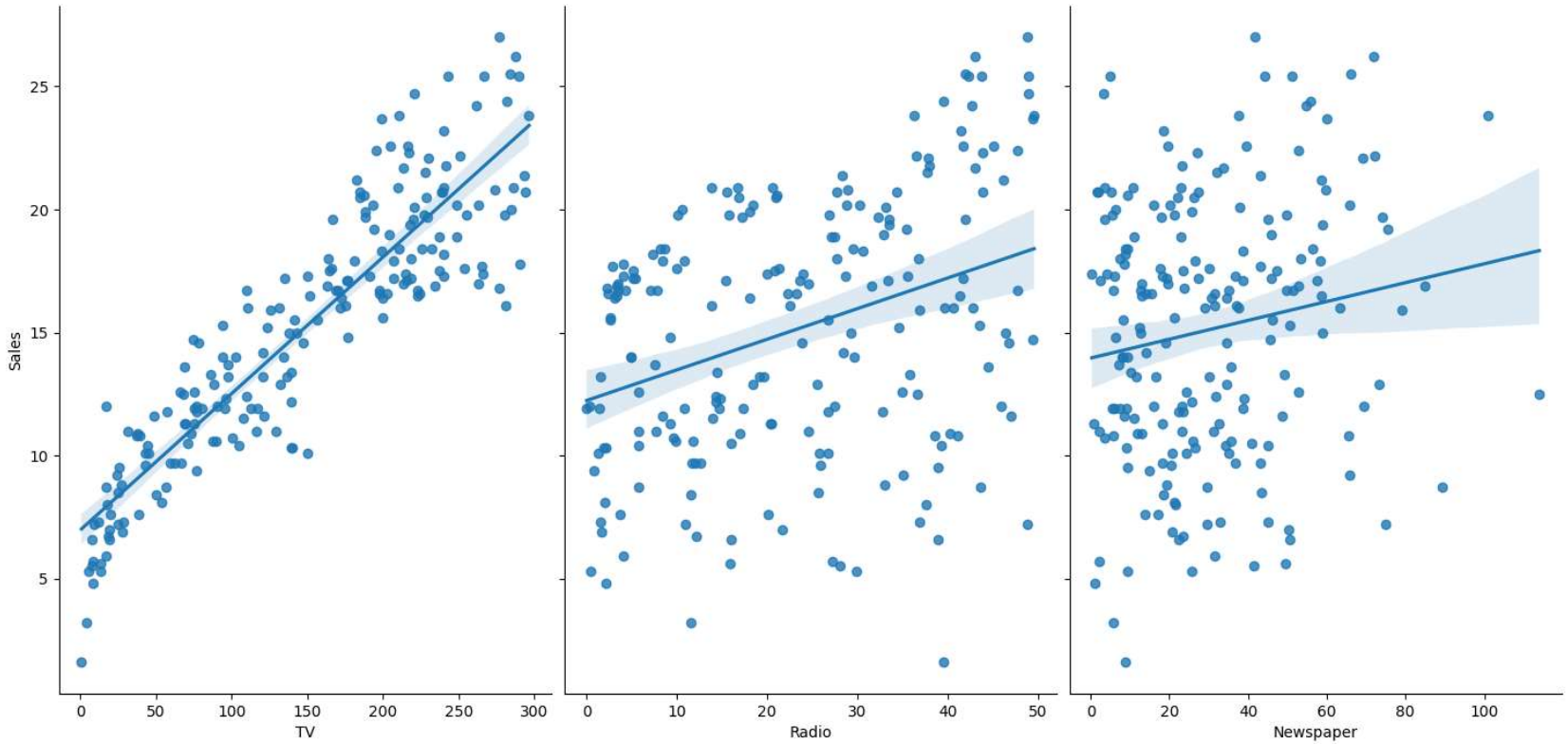
In [6]:
```
data.info()
```

```
<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(data,x_vars=['TV','Radio','Newspaper'],y_vars='Sales',height=7,aspect=0.7,kind='reg')
```

Out[8]:
```
<seaborn.axisgrid.PairGrid at 0x17035ff42d0>
```



In [9]:
```
feature=['TV','Radio','Newspaper']
```

In [10]:
```
x=data[feature]
```

In [11]:
```
x=data[['TV','Radio','Newspaper']]
```

In [12]:
```
x.head()
```

Out[12]:

	TV	Radio	Newspaper
0	230.1	37.8	69.2
1	44.5	39.3	45.1
2	17.2	45.9	69.3
3	151.5	41.3	58.5
4	180.8	10.8	58.4

In [13]:
```
print(type(x))
print(x.shape)
```

```
<class 'pandas.core.frame.DataFrame'>
(200, 3)
```

In [14]:
```
y=data['Sales']
y=data.Sales
y.head()
```

Out[14]:
```
0    22.1
1    10.4
2    12.0
3    16.5
4    17.9
Name: Sales, dtype: float64
```

In [15]:
```
print(type(y))
print(y.shape)
```

```
<class 'pandas.core.series.Series'>
(200,)
```

In [16]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=1)
```

In [17]:
```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(150, 3)
(50, 3)
(150,)
(50,)
```

In [18]:
```
from sklearn.linear_model import LinearRegression
linreg = LinearRegression()
linreg.fit(x_train,y_train)
```

Out[18]:
```
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 [19]:
```
print(linreg.intercept_)
print(linreg.coef_)
```

```
4.633808551125243
[0.05483762 0.10218027 0.00078783]
```

In [20]:
```
zip(feature, linreg.coef_)
```

Out[20]:
```
<zip at 0x1702b3fd4c0>
```

In [21]:
```
y_pred = linreg.predict(x_test)
```

In [22]:
```
true = [100, 50, 30, 20]
pred = [90, 50, 50, 30]
```

In [23]:
```
print((10 + 0 + 20 + 10) /4)
from sklearn import metrics
print(metrics.mean_absolute_error(true, pred))
```

```
10.0
10.0
```

In [24]:
```
import numpy as np
print((10**2 + 0**2 + 20**2 + 10**2) / 4)
print(metrics.mean_squared_error(true, pred))
```

```
150.0
150.0
```

In [25]:
```
import numpy as np
print(np.sqrt(((10**2 + 0**2 + 20**2 + 10**2) / 4)))
print(np.sqrt(metrics.mean_squared_error(true, pred)))
```

```
12.24744871391589
12.24744871391589
```

In [26]:
```
print(np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

```
1.509610929572584
```

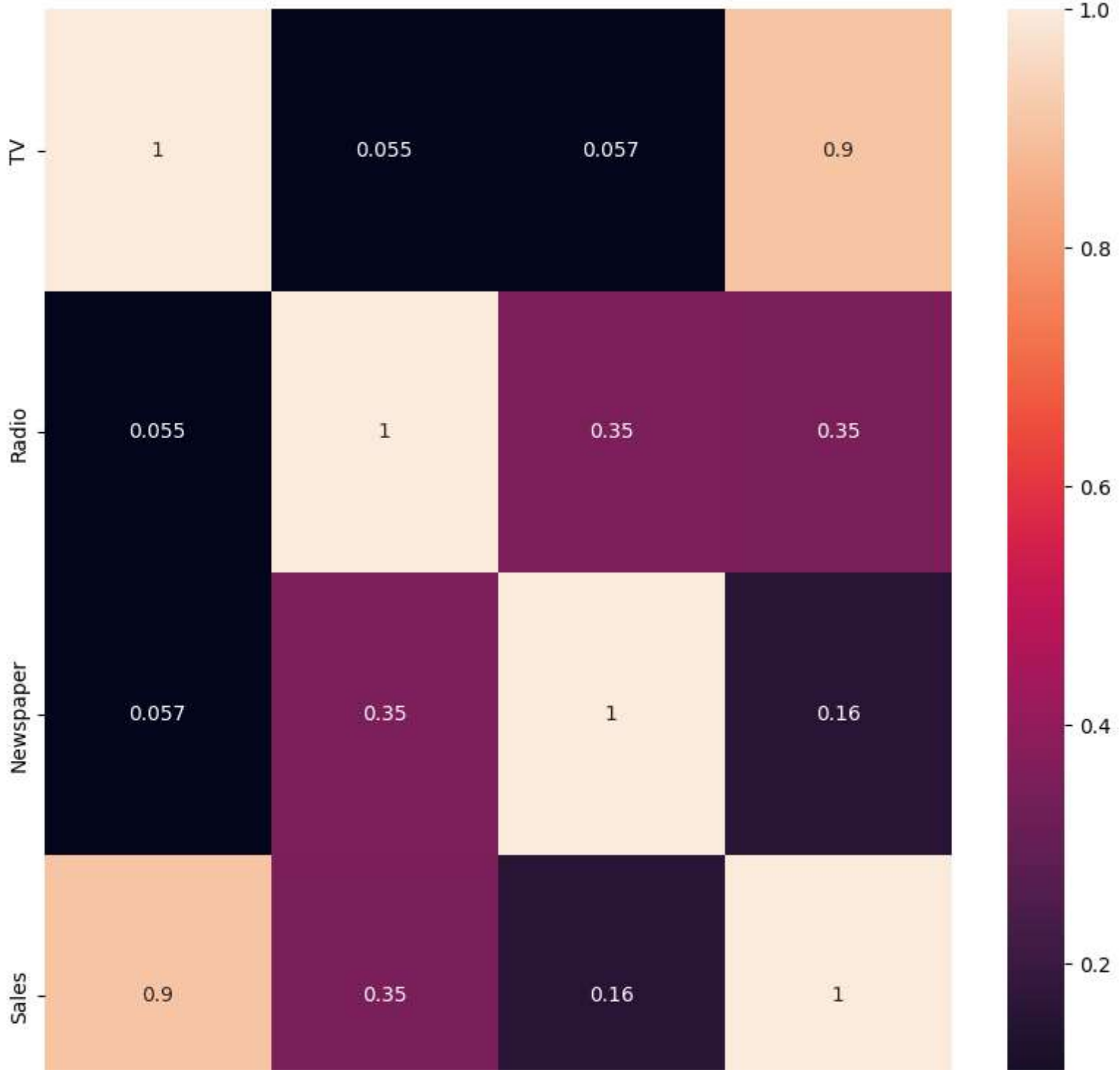
In [27]:
```
feature_cols = ['TV', 'Radio']
x=data[feature_cols]
y=data.Sales
x_train,x_test,y_train,y_test = train_test_split(x,y,random_state=1)
linreg.fit(x_train,y_train)
y_pred = linreg.predict(x_test)
print(np.sqrt(metrics.mean_squared_error(y_test,y_pred)))
```

```
1.5092481618667393
```

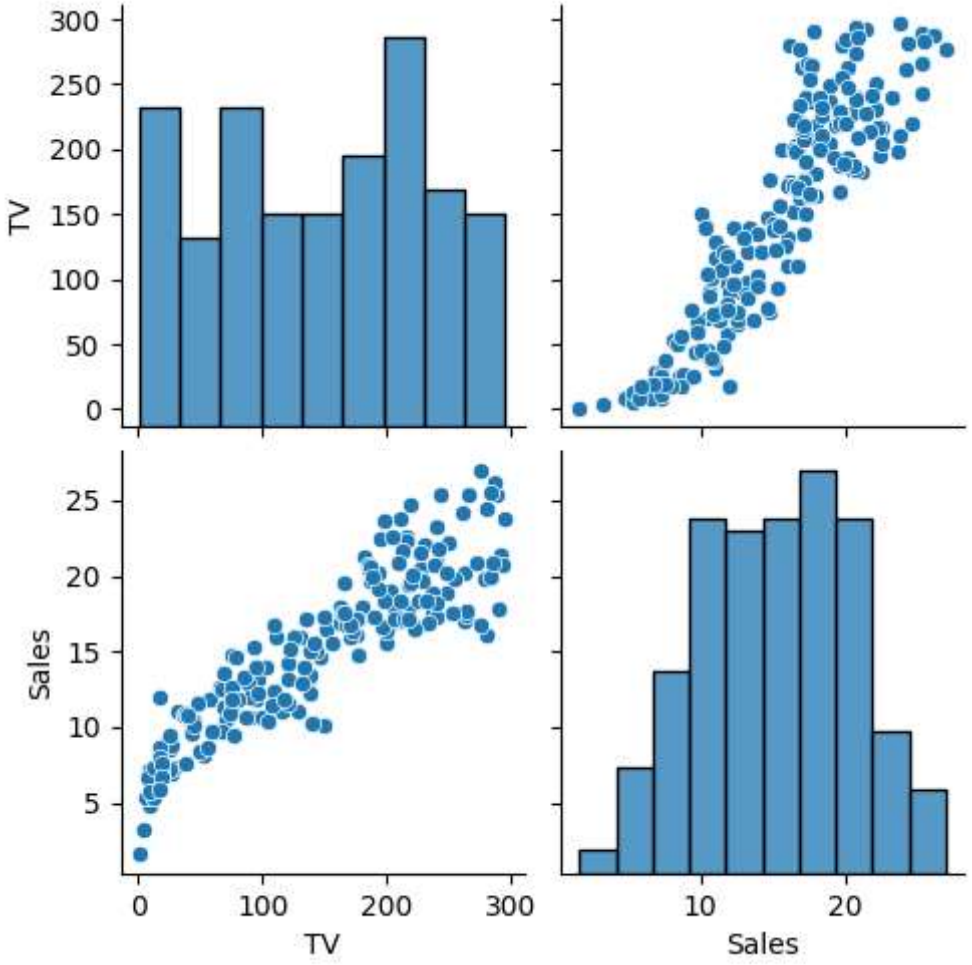
In [28]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge, RidgeCV, Lasso
from sklearn.preprocessing import StandardScaler
```

In [29]:
```
plt.figure(figsize = (10, 10))
sns.heatmap(data.corr(), annot = True)
```

Out[29]:
```
<Axes: >
```


TV
Radio
Newspaper
Sales
1
0.055
0.057
0.9
0.055
1
0.35
0.35
0.057
0.35
1
0.16
0.9
0.35
0.16
1
1.0
0.8
0.6
0.4
0.2

In [30]:
```
data.drop(columns = ["Radio", "Newspaper"],inplace = True)
sns.pairplot(data)
data.Sales = np.log(data.Sales)
```



In [31]:
```
features = data.columns[0:2]
target = data.columns[-1]
x=data[features].values
y=data[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 {}".format(x_train.shape))
print("The dimension of x_test is {}".format(x_test.shape))
scaler = StandardScaler()
x_train = scaler.fit_transform(x_train)
x_test = scaler.transform(x_test)
```

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

In [32]:
```
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)
print("\nLinear Regression Model:\n")
print("The train score for lr model is {}".format(train_score_lr))
print("The test score for lr model is {}".format(test_score_lr))
```

```
Linear Regression Model:

The train score for lr model is 1.0
The test score for lr model is 1.0
```

In [33]:
```
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("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 0.9902871391941609
The test score for ridge model is 0.984426628514122
```

In [34]:
```
plt.figure(figsize = (10, 10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5,color='red',label=r'Ridge
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color='green',label='LinearRegr
plt.xticks(rotation=90)
plt.legend()
plt.show()
```


Ridge; $\alpha = 10$
LinearRegression
0.4
0.3
0.2
0.1

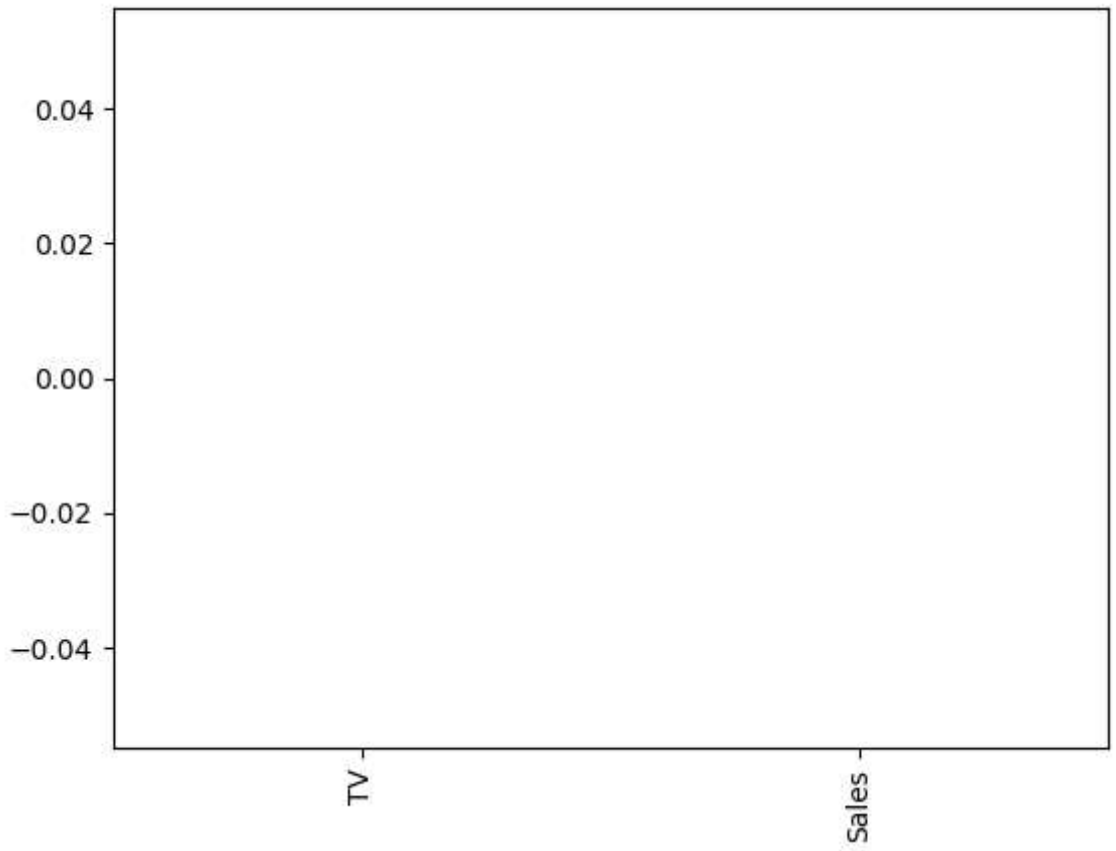
In [35]:
```
print("\nLasso Model: \n")
lasso = Lasso(alpha = 10)
lasso.fit(x_train,y_train)
train_score_ls =lasso.score(x_train,y_train)
test_score_ls =lasso.score(x_test,y_test)
print("The train score for ls model is {}".format(train_score_ls))
print("The test score for ls model is {}".format(test_score_ls))
```

```
Lasso Model:

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

In [36]:
```
pd.Series(lasso.coef_,features).sort_values(ascending = True).plot(kind = "bar")
```

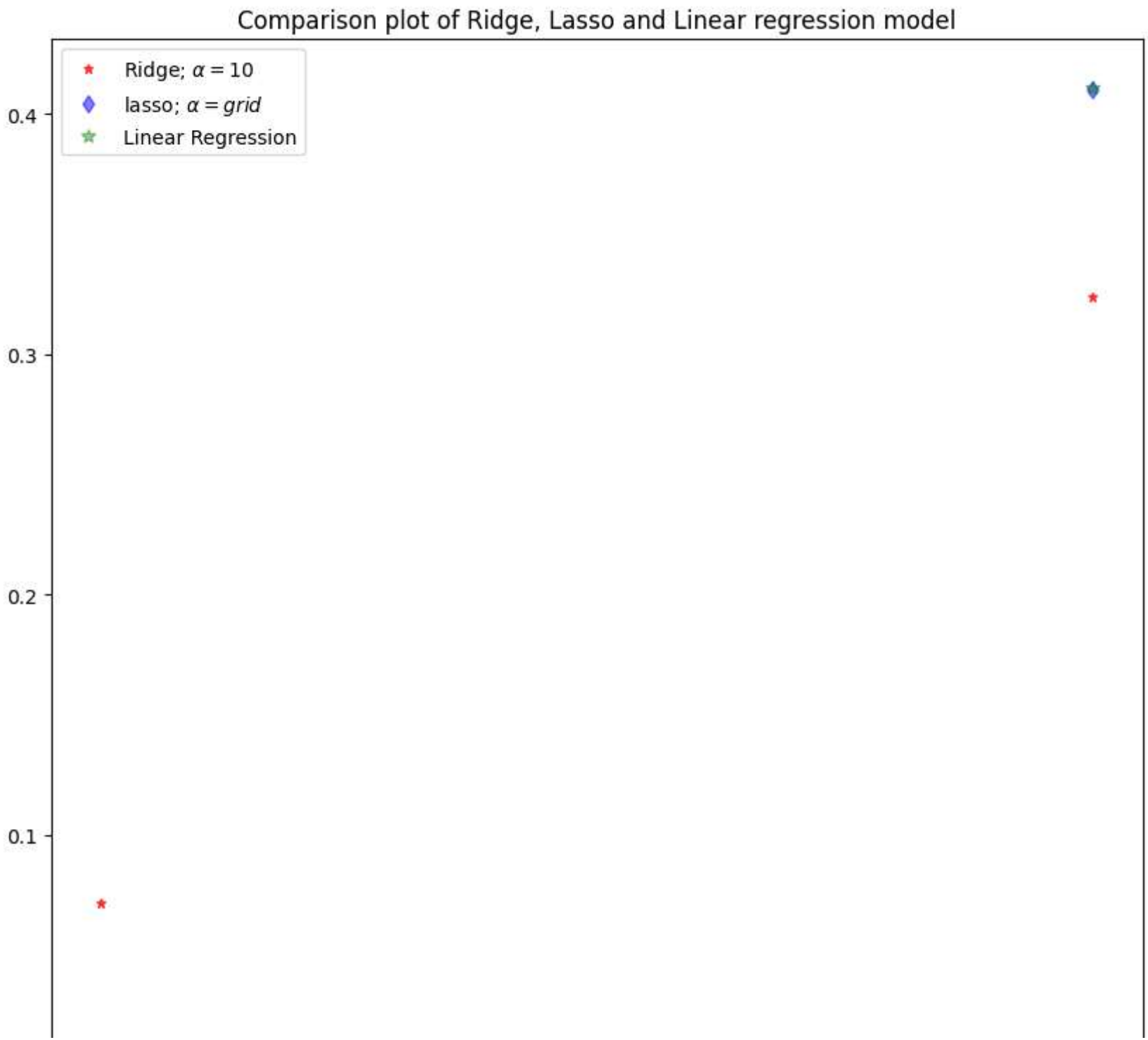
Out[36]: <Axes: >



In [37]:
```
from sklearn.linear_model import LassoCV
lasso_cv=LassoCV(alphas = [0.0001, 0.001, 0.01, 0.1, 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))
```

```
0.9999999343798134
0.9999999152638072
```

In [38]:
```
plt.figure(figsize=(10, 10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5,color='red',label=r'Ridge
plt.plot(lasso_cv.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color='blue',label=r'lasso; $\alph
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='*',markersize=7,color='green',label='Linear Reg
plt.xticks(rotation = 90)
plt.legend()
plt.title("Comparison plot of Ridge, Lasso and Linear regression model")
plt.show()
```


Comparison plot of Ridge, Lasso and Linear regression model
Ridge; $\alpha = 10$
lasso; $\alpha = grid$
Linear Regression
0.4
0.3
0.2
0.1

In [39]:
```
from sklearn.linear_model import RidgeCV
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_train)))
print("The test score for ridge model is {}".format(ridge_cv.score(x_test,y_test)))
```

```
The train score for ridge model is 0.999999999997627
The test score for ridge model is 0.9999999999962466
```

In [40]:
```
from sklearn.linear_model import ElasticNet
regr = ElasticNet()
regr.fit(x,y)
print(regr.coef_)
print(regr.intercept_)
```

```
[0.00417976 0.        ]
2.026383919311004
```

In [41]:
```
y_pred_elastic = regr.predict(x_train)
```

In [42]:
```
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 0.5538818050142158
```

In [ ]:
```

```