In [1]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn import preprocessing,svm
from sklearn.model_selection import train_test_split
```

In [40]:

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

Out[40]:

	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 [41]:

```
data.head()
```

Out[41]:

	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 [16]:

```
data.tail()
```

Out[16]:

	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 [17]:

```
data.describe
```

Out[17]:

```
<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 [18]:

```
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 [19]:

```
data.columns
```

Out[19]:

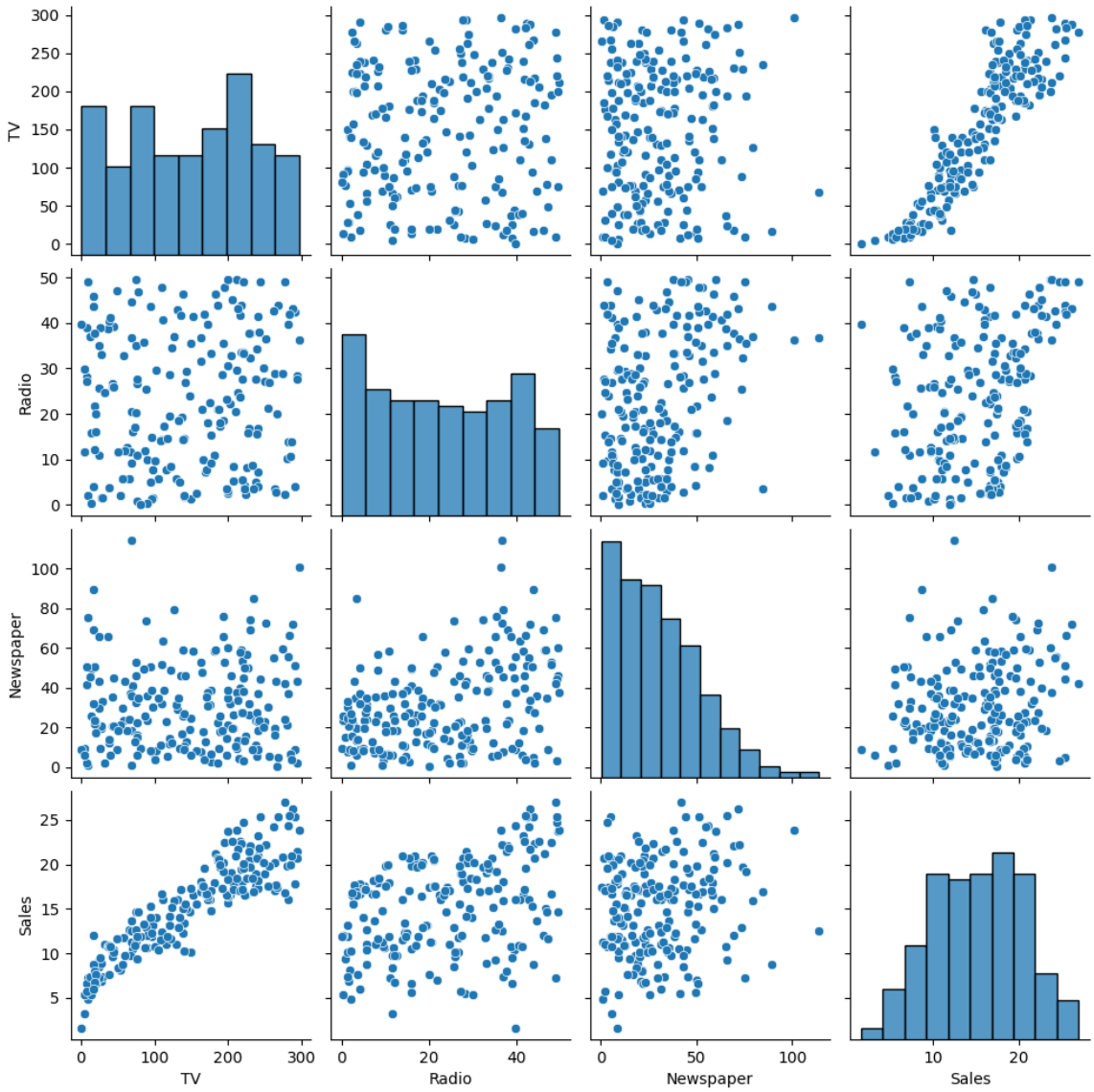
```
Index(['TV', 'Radio', 'Newspaper', 'Sales'], dtype='object')
```

In [20]:

```
sns.pairplot(data)
```

Out[20]:

```
<seaborn.axisgrid.PairGrid at 0x2143decab60>
```



In [21]:

```
data.isna().any()
```

Out[21]:

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

In [23]:

```
features=["TV","Radio","Newspaper"]
x=features
```

In [24]:

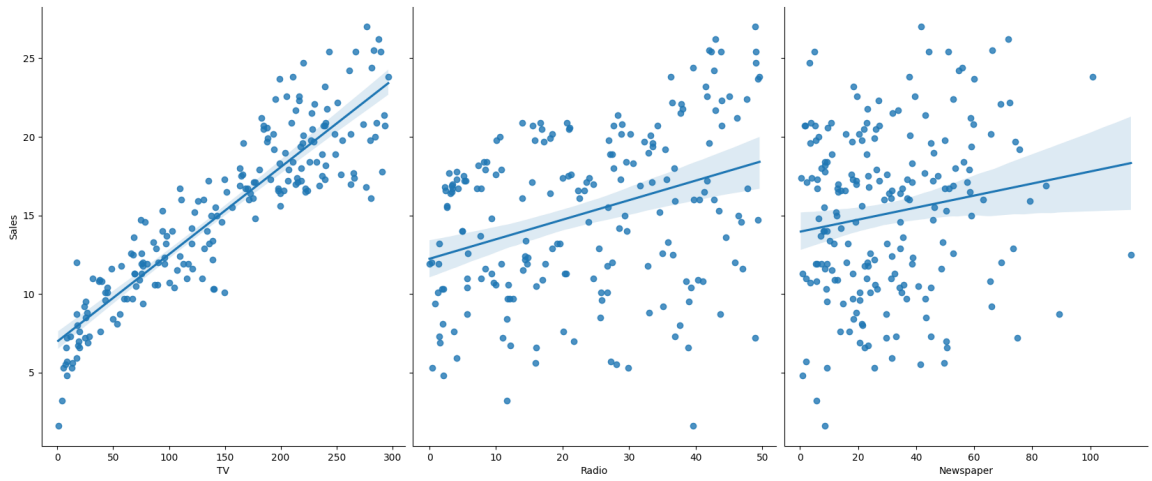
```
y=["Sales"]
```

In [25]:

```
sns.pairplot(data ,x_vars=['TV','Radio','Newspaper'],y_vars='Sales',height=7,aspect=0.8,k
```

Out[25]:

```
<seaborn.axisgrid.PairGrid at 0x2143fcebfa0>
```



In [26]:

```
x=data.iloc[:,0:3]
y=data.iloc[:,-1]
```

In [27]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)
```

In [28]:

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

```
0.9194718041406325
```

In [29]:

```
from sklearn.metrics import r2_score
```

In [30]:

```
model=LinearRegression()
model.fit(x_train,y_train)
```

Out[30]:

```
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 [31]:

```
y_pred=model.predict(x_test)
r2=r2_score(y_test,y_pred)
print("R2 score",r2)
```

```
R2 score 0.9194718041406325
```

In [32]:

```
print(model.intercept_)
```

```
4.746051241038177
```

In [33]:

```
print(regr.coef_)
```

```
[0.05368353 0.10664656 0.00260815]
```

In [34]:

```
features=data[["TV","Radio","Newspaper"]]
features
```

Out[34]:

	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
...	...	...	...
195	38.2	3.7	13.8
196	94.2	4.9	8.1
197	177.0	9.3	6.4
198	283.6	42.0	66.2
199	232.1	8.6	8.7

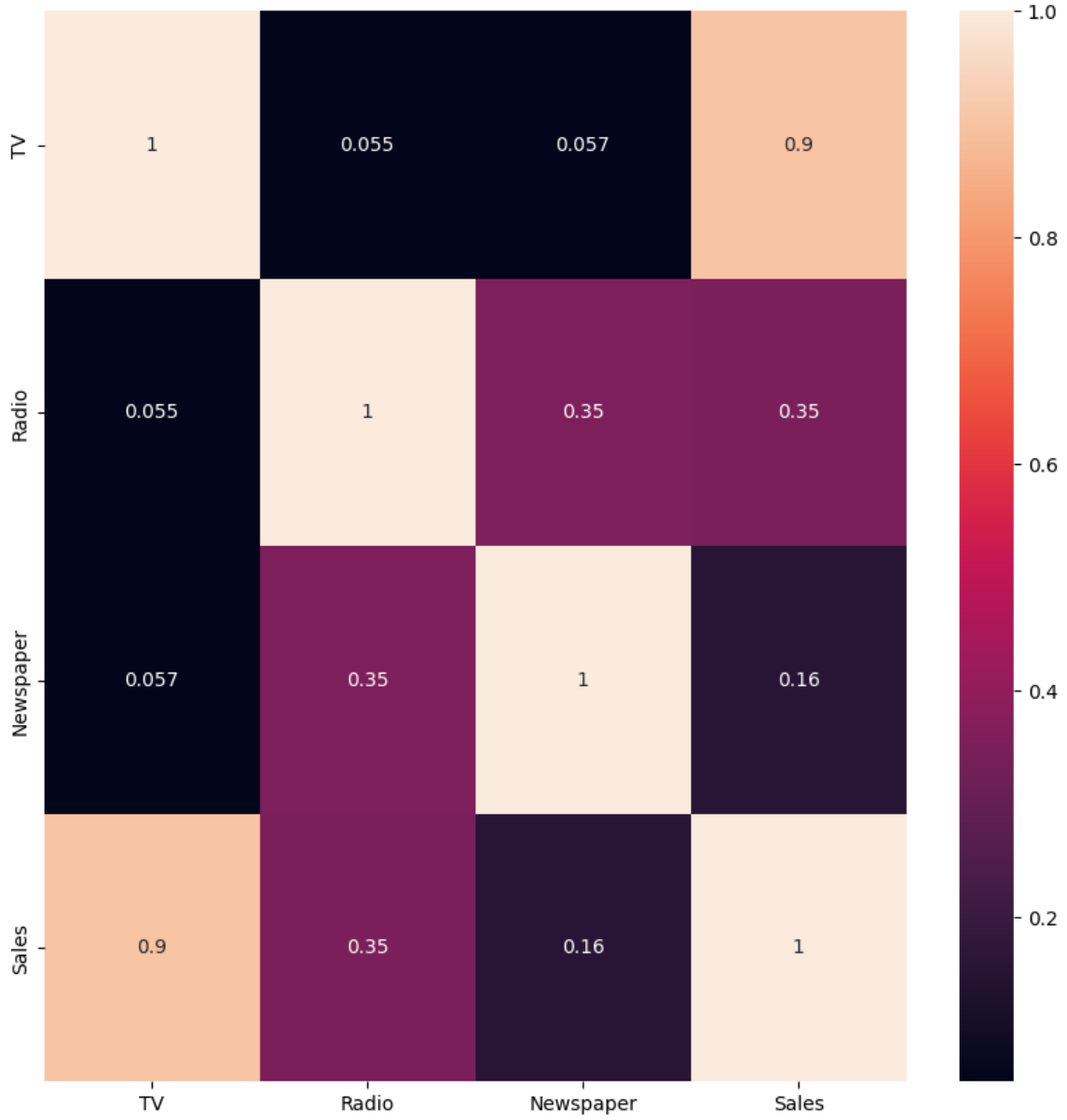
200 rows × 3 columns

In [35]:

```
plt.figure(figsize = (10, 10))
sns.heatmap(data.corr(), annot = True)
```

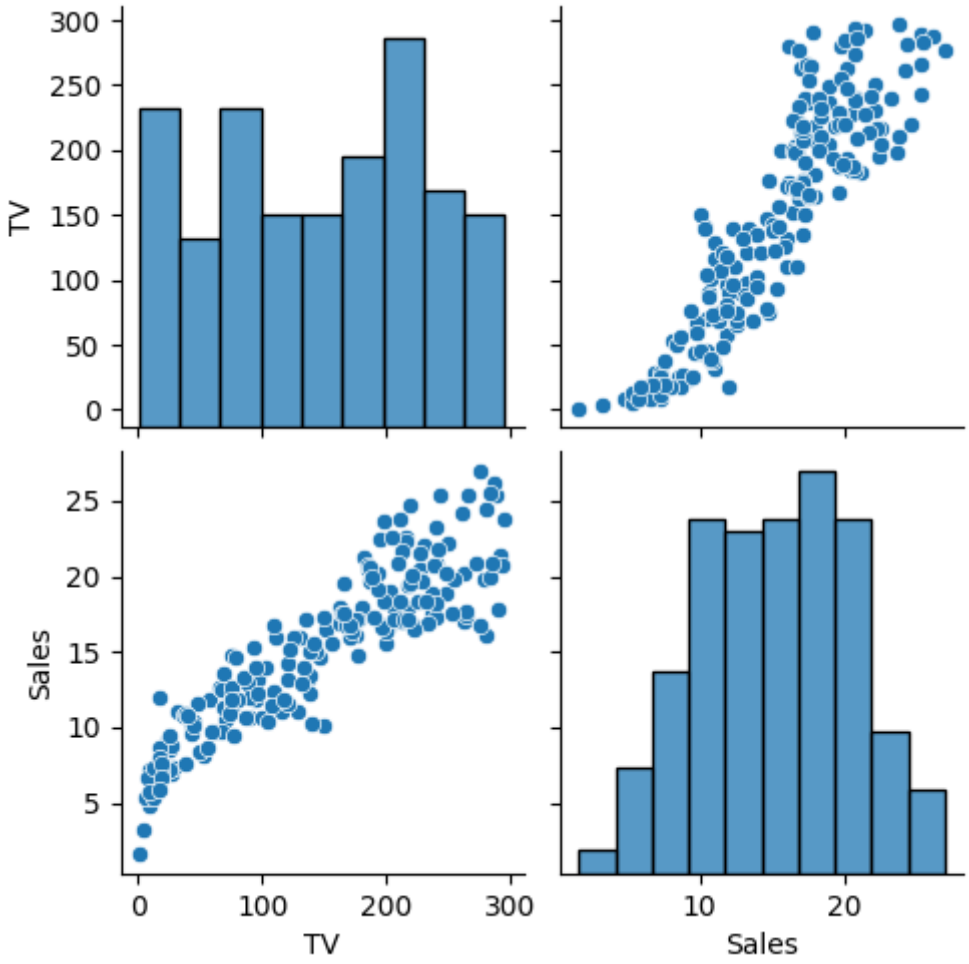
Out[35]:

```
<Axes: >
```



In [42]:

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



In [47]:

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

In [48]:

```
features = data.columns[0:2]
target = data.columns[-1]
#X and y values
X = data[features].values
y = data[target].values
#splot
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))
#Scale features
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 [49]:

```
#Model
lr = LinearRegression()
#Fit model
lr.fit(X_train, y_train)
#predict
#prediction = lr.predict(X_test)
#actual
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
```

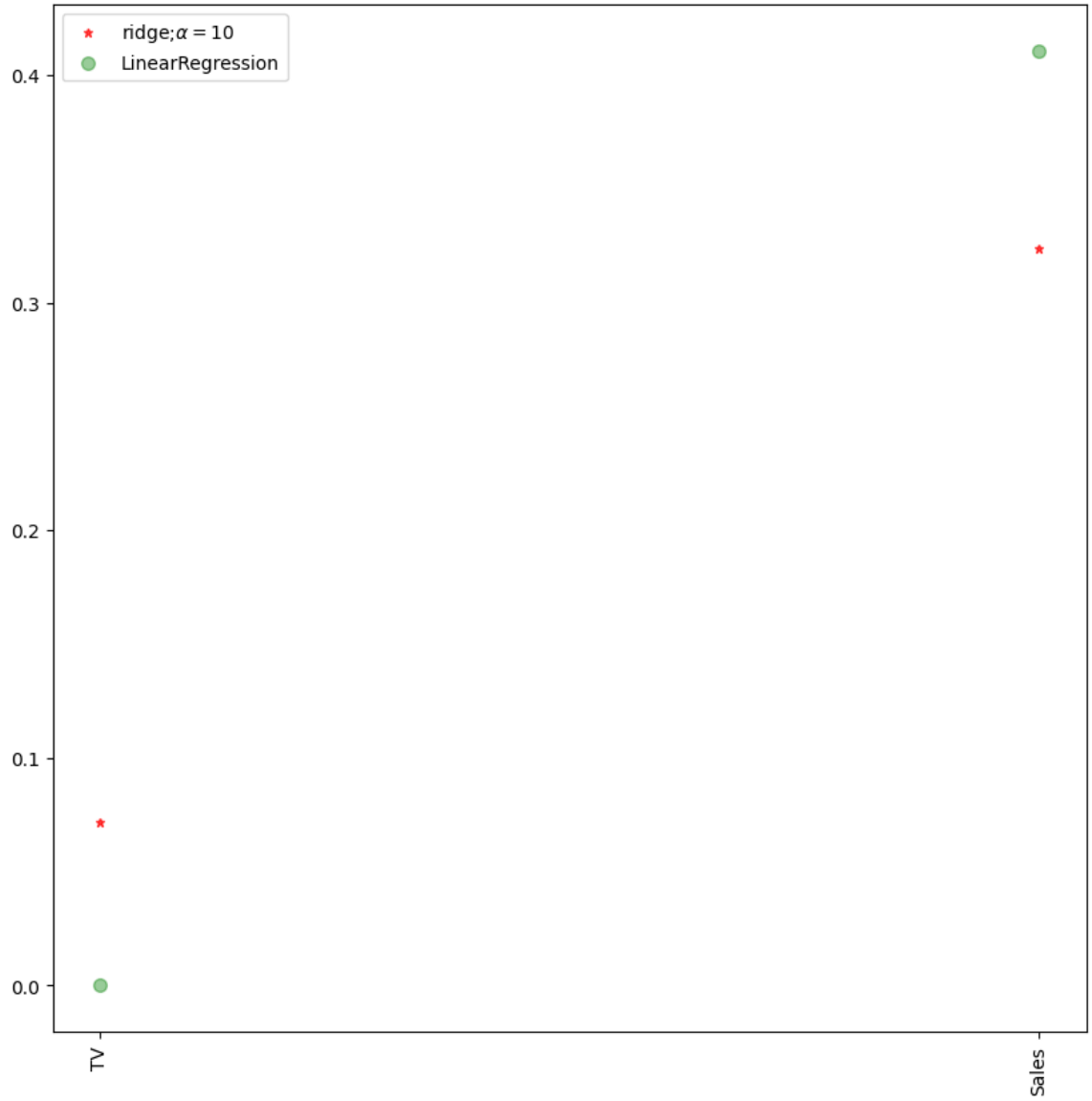
In [50]:

```
#Ridge Regression Model
ridgeReg = Ridge(alpha=10)
ridgeReg.fit(X_train,y_train)
#train and test scorefor ridge regression
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 0.990287139194161
The test score for ridge model is 0.9844266285141221
```

In [53]:

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



In [54]:

```
#Lasso regression model
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))
```

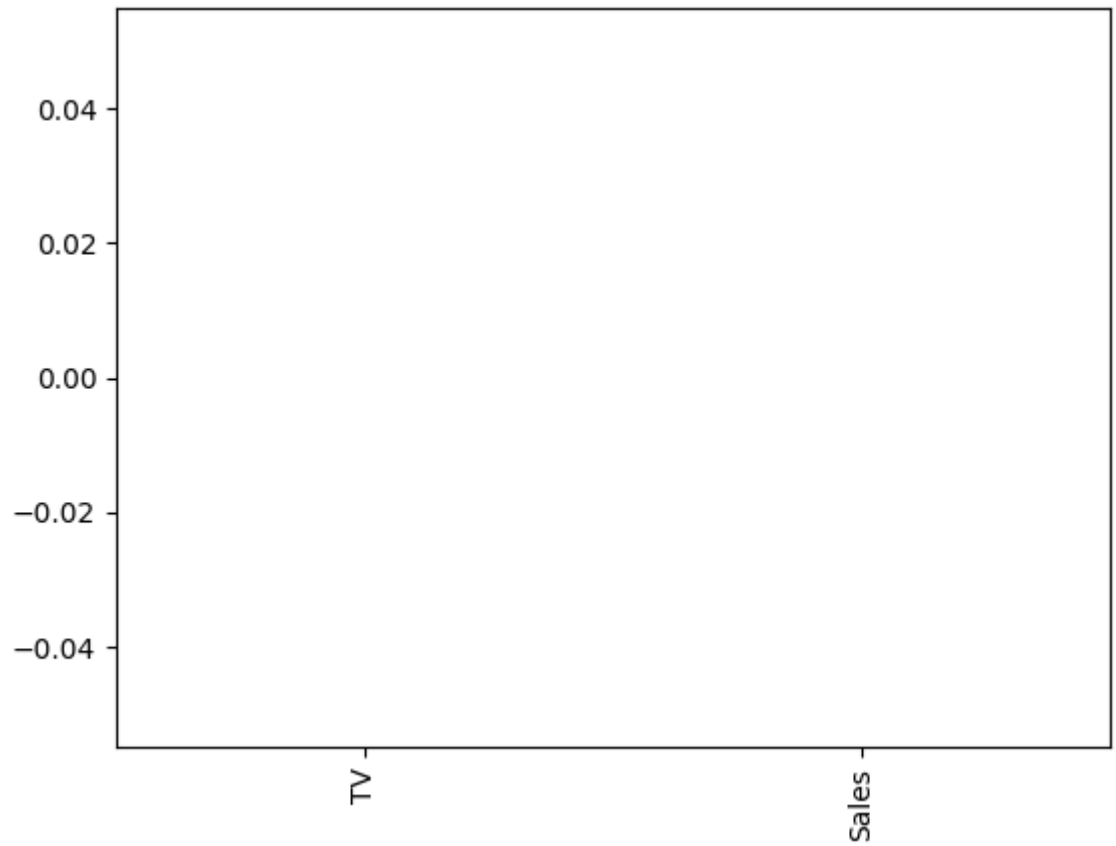
```
Lasso Model:

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

In [55]:

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

Out[55]:

```
<Axes: >
```



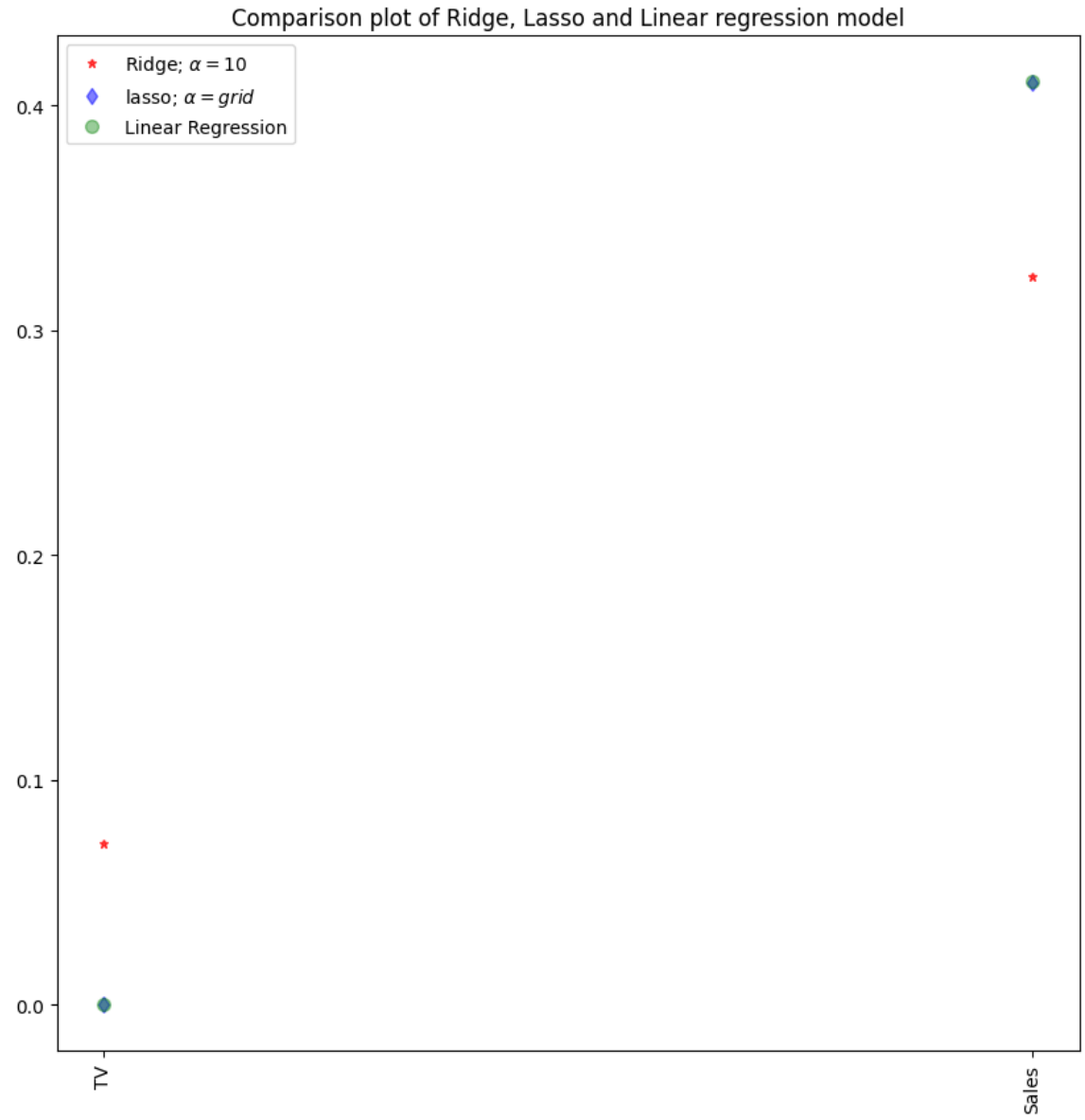
In [56]:

```
#Using the linear CV model
from sklearn.linear_model import LassoCV
#Lasso Cross validation
lasso_cv = LassoCV(alphas = [0.0001, 0.001,0.01, 0.1, 1, 10], random_state=0).fit(X_trair
#score
print(lasso_cv.score(X_train, y_train))
print(lasso_cv.score(X_test, y_test))
```

```
0.9999999343798134
0.9999999152638072
```

In [58]:

```
#plot size
plt.figure(figsize = (10, 10))
#add plot for ridge regression
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5,color
#add plot for lasso regression
plt.plot(lasso_cv.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color='blue',l
#add plot for linear model
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color='gree
#rotate axis
plt.xticks(rotation = 90)
plt.legend()
plt.title("Comparison plot of Ridge, Lasso and Linear regression model")
plt.show()
```



In [59]:

```
#Using the linear CV model
from sklearn.linear_model import RidgeCV
#Ridge Cross validation
ridge_cv = RidgeCV(alphas = [0.0001, 0.001,0.01, 0.1, 1, 10]).fit(X_train, y_train)
#score
print("The train score for ridge model is {}".format(ridge_cv.score(X_train, y_train)))
print("The train score for ridge model is {}".format(ridge_cv.score(X_test, y_test)))
```

```
The train score for ridge model is 0.999999999997627
The train score for ridge model is 0.9999999999962467
```

In [ ]: