In [1]:
```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
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 [2]:
```python
df = pd.read_csv(r"C:\Users\Sudheer\AppData\Local\Microsoft\Windows\INetCache\
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]:
```python
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]:
```python
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]:
```python
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]:
```python
df.columns
```

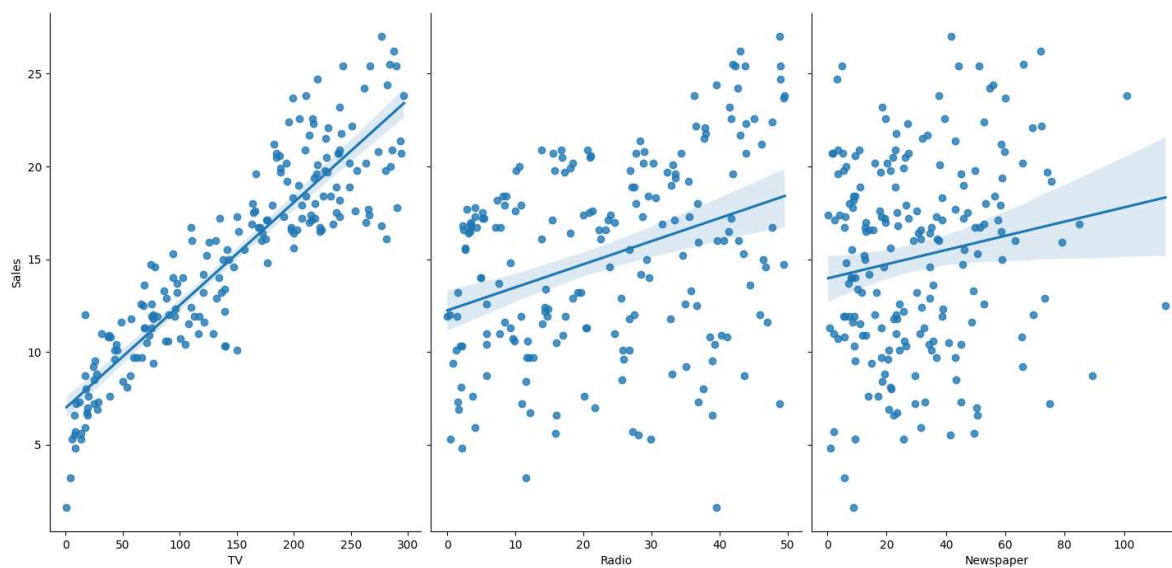Out[6]: Index(['TV', 'Radio', 'Newspaper', 'Sales'], dtype='object')

In [7]: `sns.pairplot(df)`

Out[7]: `<seaborn.axisgrid.PairGrid at 0x2267845d960>`
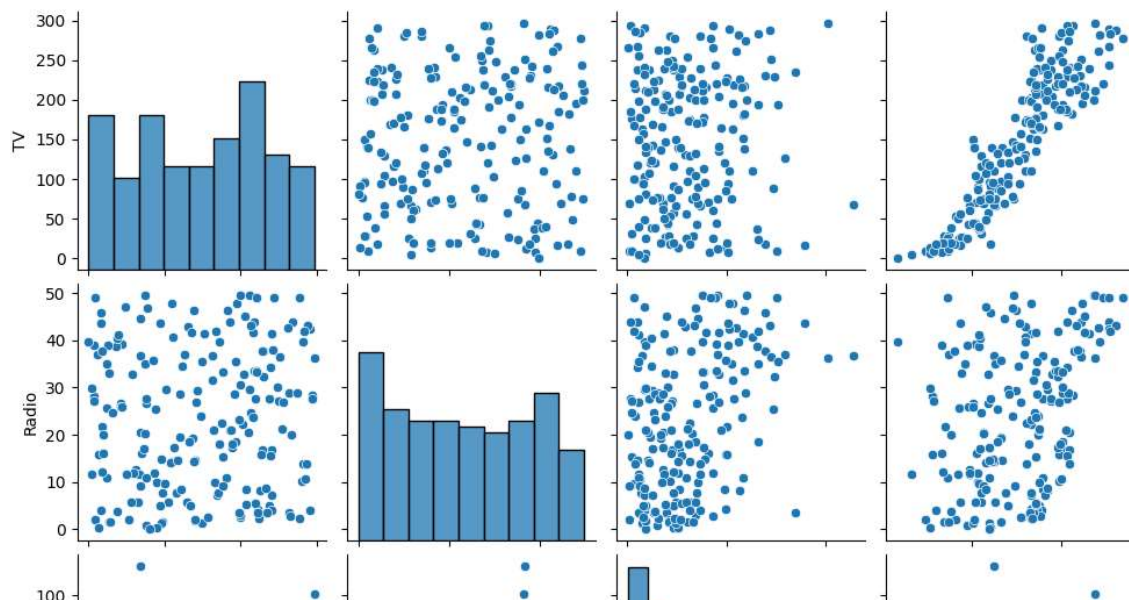
In [8]: `sns.pairplot(df,x_vars=["TV","Radio","Newspaper"],y_vars='Sales',height=7,aspe`

Out[8]: `<seaborn.axisgrid.PairGrid at 0x2267c1966b0>`

In [ ]:

In [9]:
```python
plt.figure(figsize = (10, 10))
sns.heatmap(df.corr(), annot = True)
```

Out[9]: <Axes: >

```
In [10]: features = df.columns[0:2]
         target = df.columns[-1]
         #X and y values
         X = df[features].values
         y = df[target].values
         #splot
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, rando
         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 [11]: #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 0.913873609964124
The test score for ridge model is 0.8486032178989196
```

```
In [12]: #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 0.9174763132374348
The test score for lr model is 0.8567416956374659
```

In [13]:
```python
plt.figure(figsize = (10, 10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markers
#plt.plot(lr100.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,
plt.xticks(rotation = 90)
plt.legend()
plt.show()
```
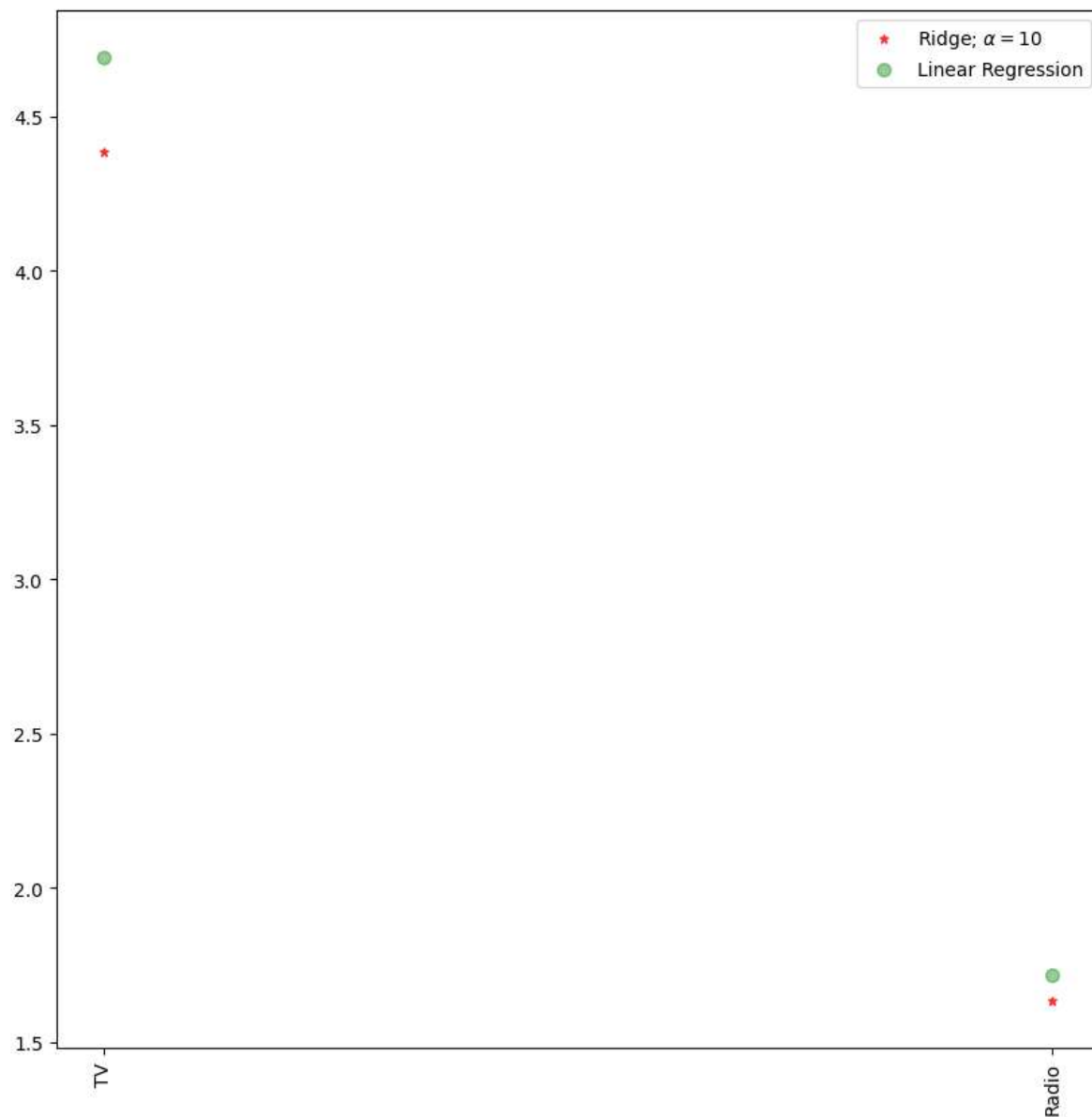
In [14]:
```python
#Lasso regression model
print("\nLasso Model: \n")
lasso = Lasso(alpha = 10)
lasso.fit(X_train,y_train)
train_score_ls =lasso.score(X_train,y_train)
test_score_ls =lasso.score(X_test,y_test)
print("The train score for ls model is {}".format(train_score_ls))
print("The test score for ls model is {}".format(test_score_ls))
```
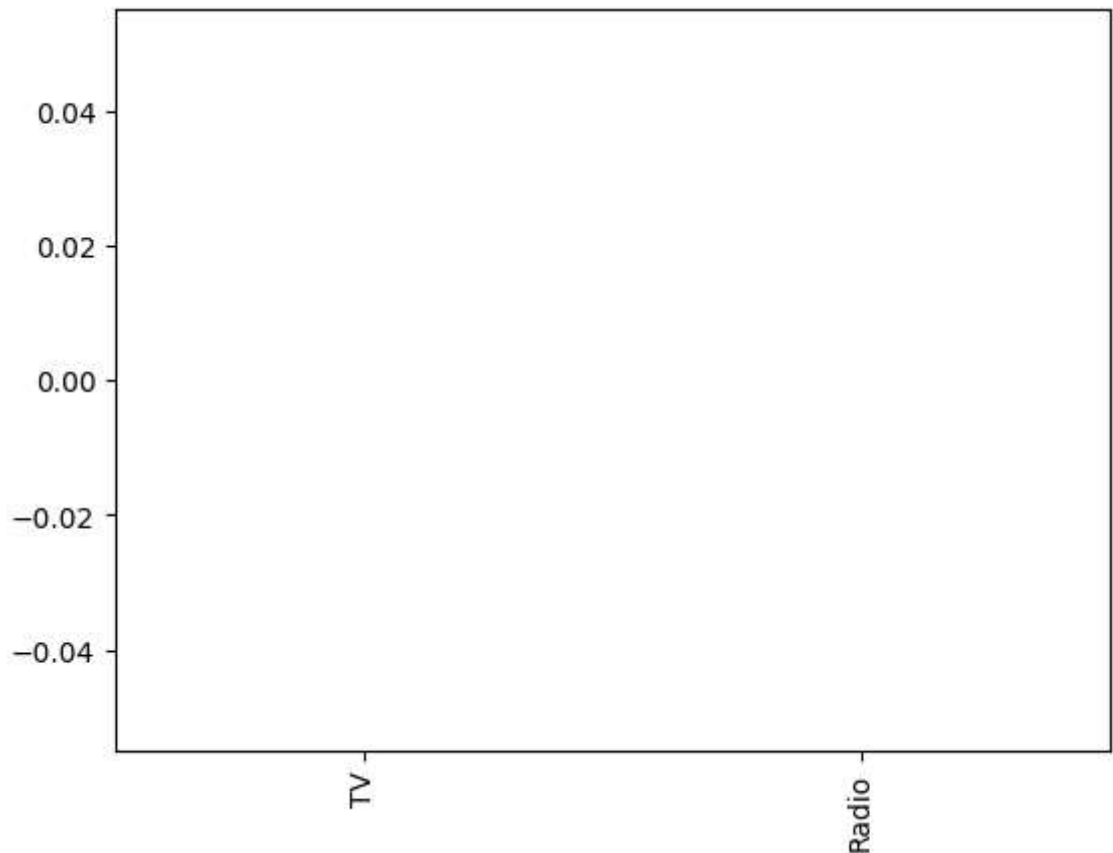
```
Lasso Model:

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

In [15]:
```python
pd.Series(lasso.coef_, features).sort_values(ascending = True).plot(kind = "ba
```
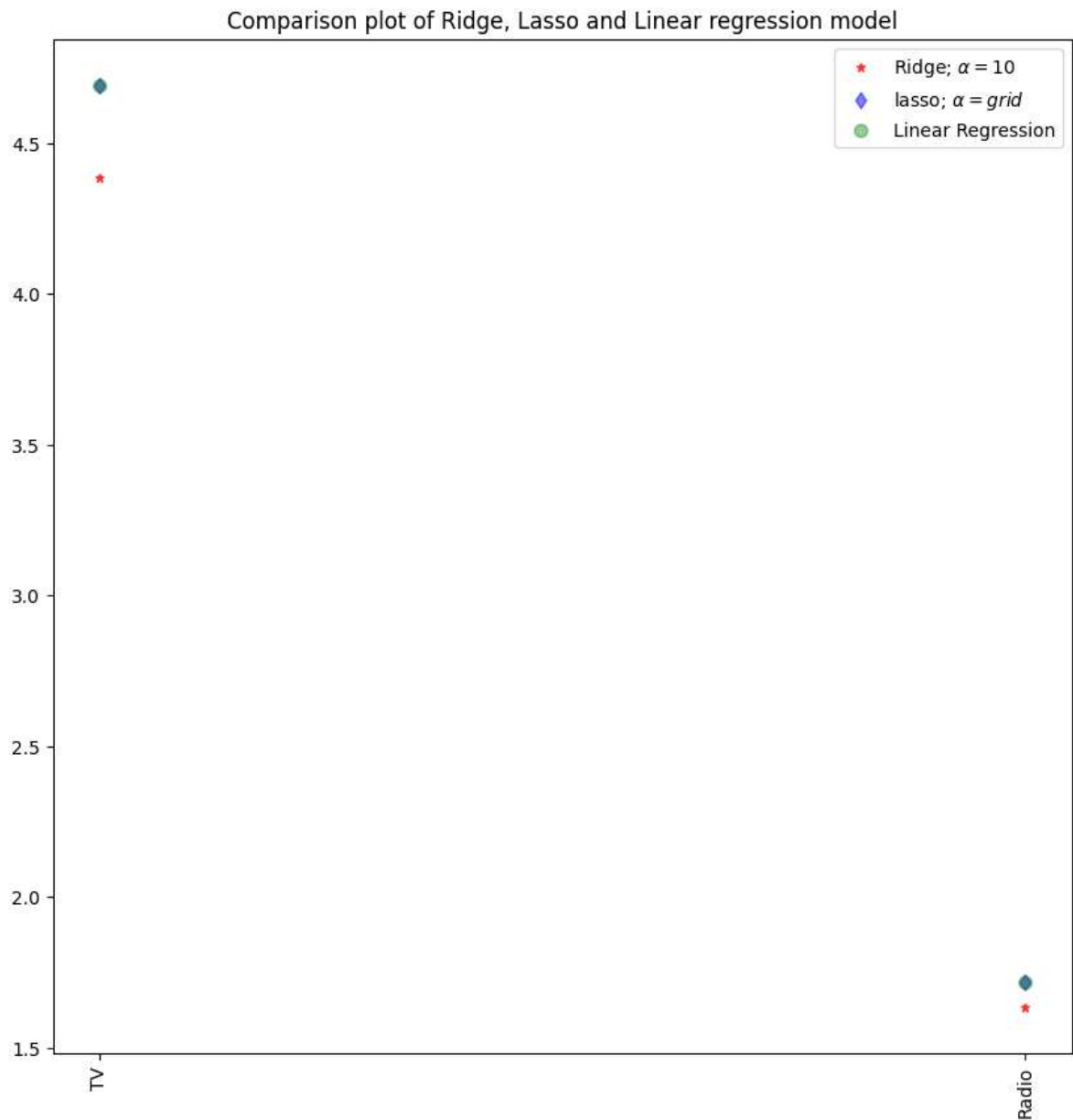
Out[15]: `<Axes: >`

In [16]:
```python
#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).
#score
print(lasso_cv.score(X_train, y_train))
print(lasso_cv.score(X_test, y_test))
```

```
0.9174763126220602
0.8567421168446285
```

```python
In [17]: #plot size
         plt.figure(figsize = (10, 10))
         #add plot for ridge regression
         plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markers
         #add plot for lasso regression
         plt.plot(lasso_cv.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,col
         #add plot for linear model
         plt.plot(features,lr.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()
```

In [18]:
```python
#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_t
#score
print("The train score for ridge model is {}".format(ridge_cv.score(X_train, y
print("The train score for ridge model is {}".format(ridge_cv.score(X_test, y_
```

```
The train score for ridge model is 0.9174759031352496
The train score for ridge model is 0.8566880002211078
```

In [ ]:

In [23]:

In [ ]:

In [ ]: