In [2]:
```
import numpy as np
import pandas as pd
```

In [3]:
```
df=pd.read_csv(r'C:\Users\HP\Downloads\Advertising.csv')
df
```

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
...	...	...	...	...
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

LINEAR REGRESSION

In [4]:
```
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import preprocessing,svm
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

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

Out[5]:
```
Index(['TV', 'Radio', 'Newspaper', 'Sales'], dtype='object')
```

In [6]:
```
df.head(10)
```

Out[6]:

	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
5	8.7	48.9	75.0	7.2
6	57.5	32.8	23.5	11.8
7	120.2	19.6	11.6	13.2
8	8.6	2.1	1.0	4.8
9	199.8	2.6	21.2	15.6

In [7]:
```
df=df[['TV','Sales']]
df.columns=['T','S']
```

In [8]:
```
df.describe()
```

Out[8]:

	T	S
count	200.000000	200.000000
mean	147.042500	15.130500
std	85.854236	5.283892
min	0.700000	1.600000
25%	74.375000	11.000000
50%	149.750000	16.000000
75%	218.825000	19.050000
max	296.400000	27.000000

In [9]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   T       200 non-null    float64
 1   S       200 non-null    float64
dtypes: float64(2)
memory usage: 3.2 KB
```

In [10]:
```
df.fillna(method="ffill",inplace=True)
```

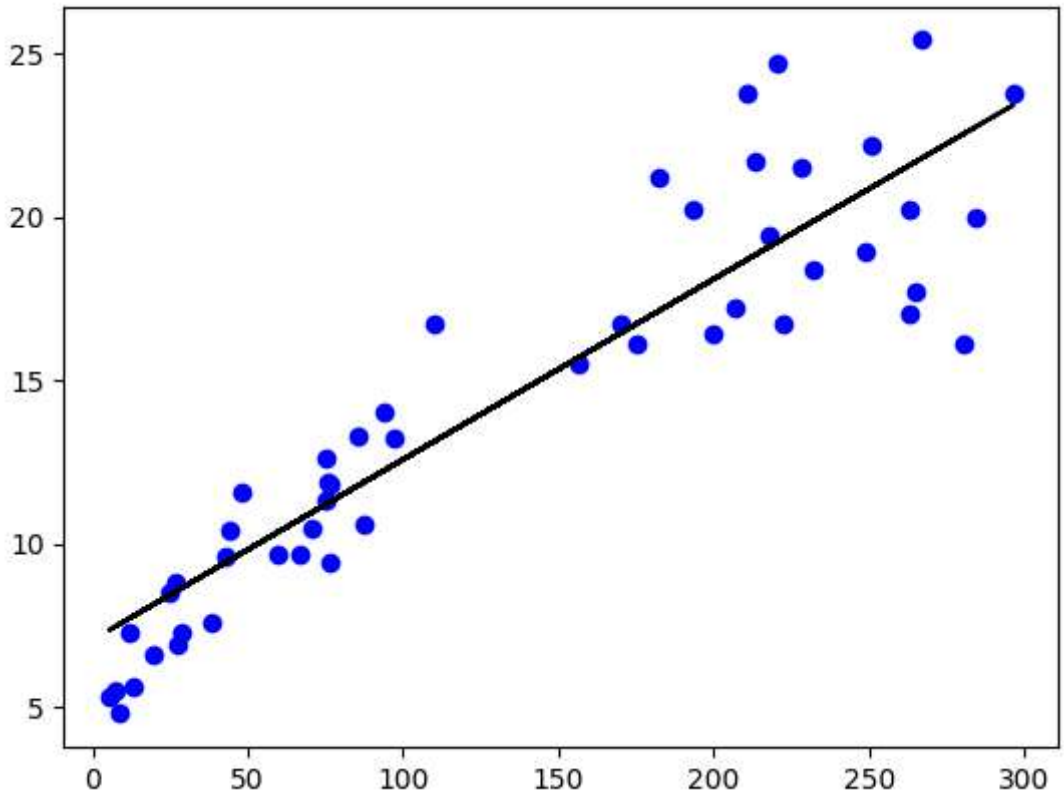
```
C:\Users\HP\AppData\Local\Temp\ipykernel_7508\1844562654.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df.fillna(method="ffill",inplace=True)
```

In [11]:
```
x=np.array(df['T']).reshape(-1,1)
y=np.array(df['S']).reshape(-1,1)
```

In [12]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)
regr=LinearRegression()
regr.fit(x_train,y_train)
print(regr.score(x_test,y_test))
```

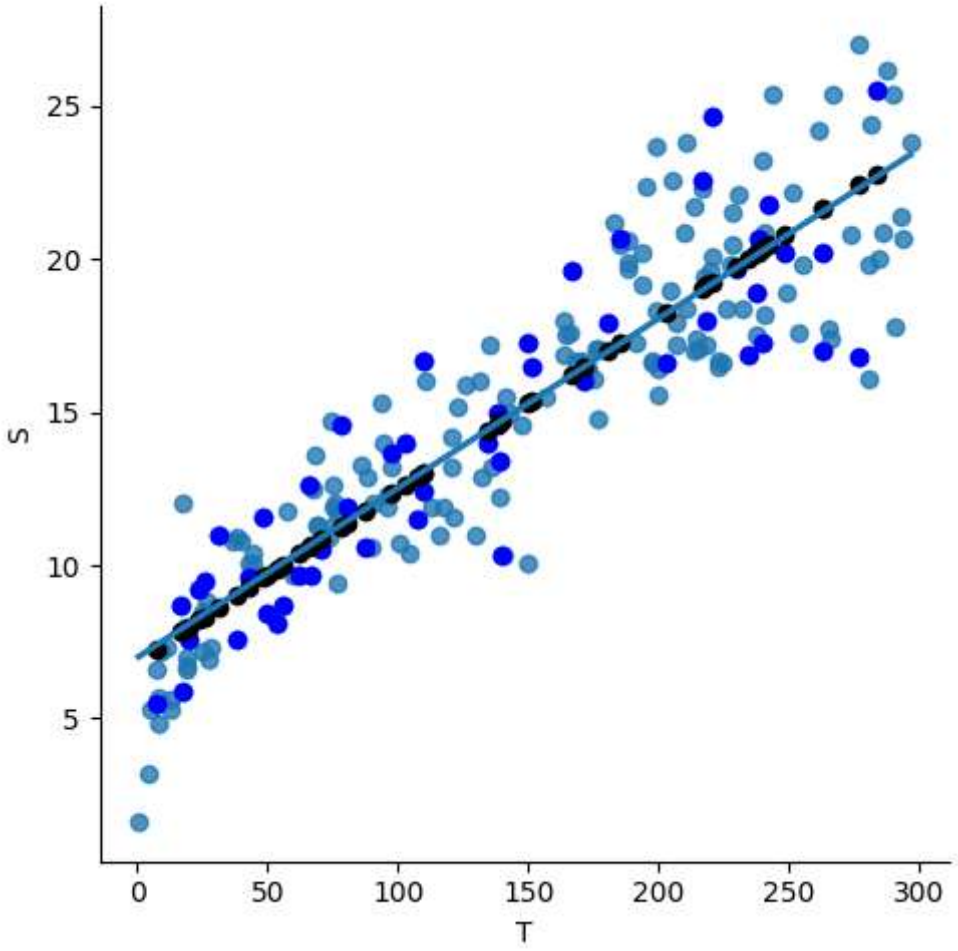
```
0.8354647174818481
```

In [13]:
```
y_pred=regr.predict(x_test)
plt.scatter(x_test,y_test,color='b')
plt.plot(x_test,y_pred,color='k')
plt.show()
```



In [14]:
```
df500=df[:][:500]
sns.lmplot(x="T",y="S",data=df500,order=1,ci=None)
df500.fillna(method='ffill',inplace=True)
x=np.array(df500['T']).reshape(-1,1)
y=np.array(df500['S']).reshape(-1,1)
df500.dropna(inplace=True)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)
regr=LinearRegression()
regr.fit(x_train,y_train)
print("Regression:",regr.score(x_test,y_test))
y_pred=regr.predict(x_test)
plt.scatter(x_test,y_test,color='b')
plt.scatter(x_test,y_pred,color='k')
plt.show()
```

```
Regression: 0.7995655902671848
```



In [15]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
model=LinearRegression()
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
r2=r2_score(y_test,y_pred)
print(r2)
```

```
0.7995655902671848
```

In [16]:
```
from sklearn.linear_model import Lasso,Ridge
import seaborn as sns
import matplotlib.pyplot as plt
```

In [17]:
```
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('The Train score is {}'.format(train_score_ridge))
print('The Test score is {}'.format(test_score_ridge))
```

```
The Train score is 0.8138285804088078
The Test score is 0.799566459773546
```

In [18]:
```
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 [19]:
```
df=pd.read_csv(r'C:\Users\HP\Downloads\Advertising.csv')
df
```

Out[19]:

	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 [20]:
```
df.head()
```

Out[20]:

	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

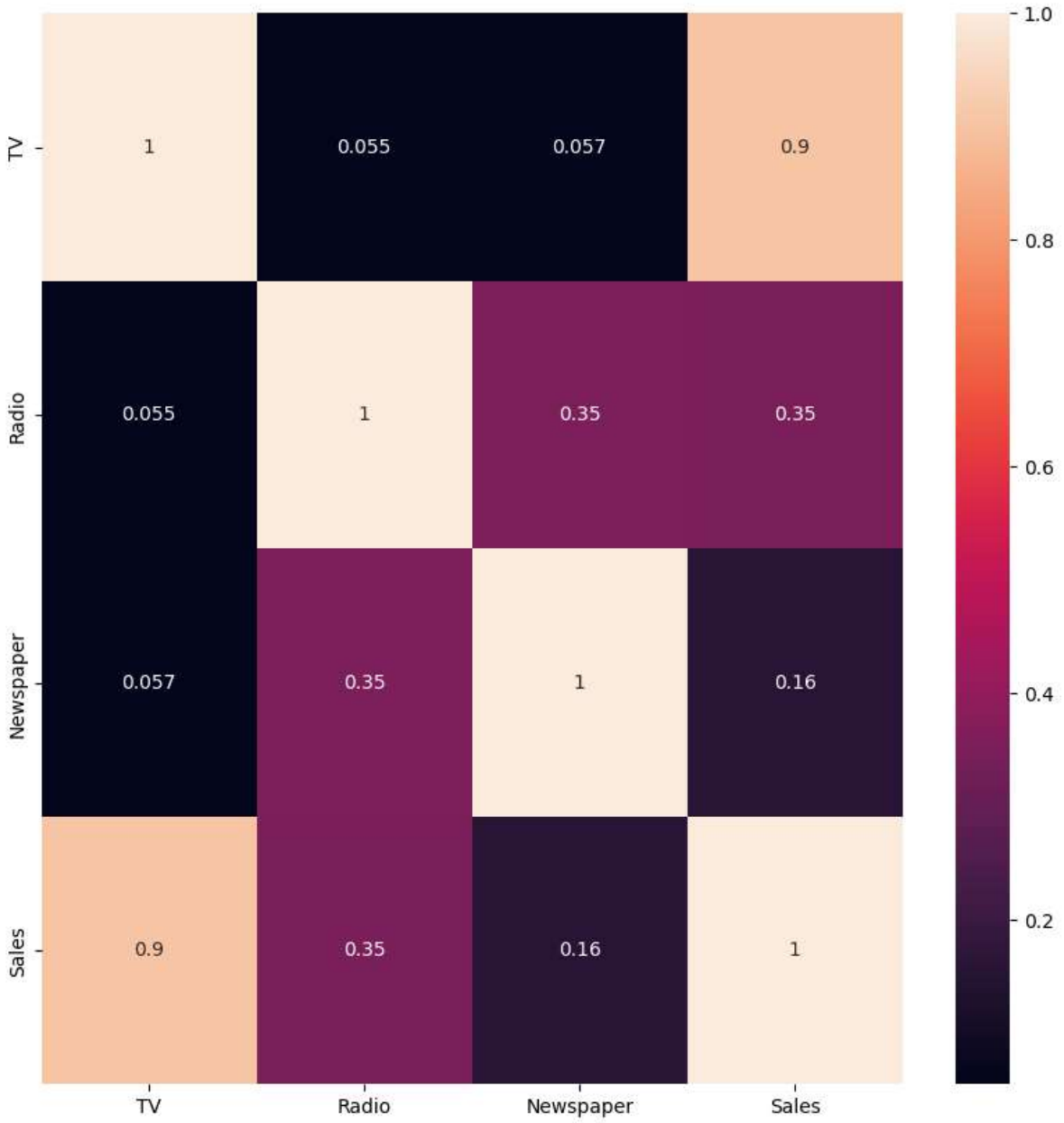
In [21]:
```
df.tail()
```

Out[21]:

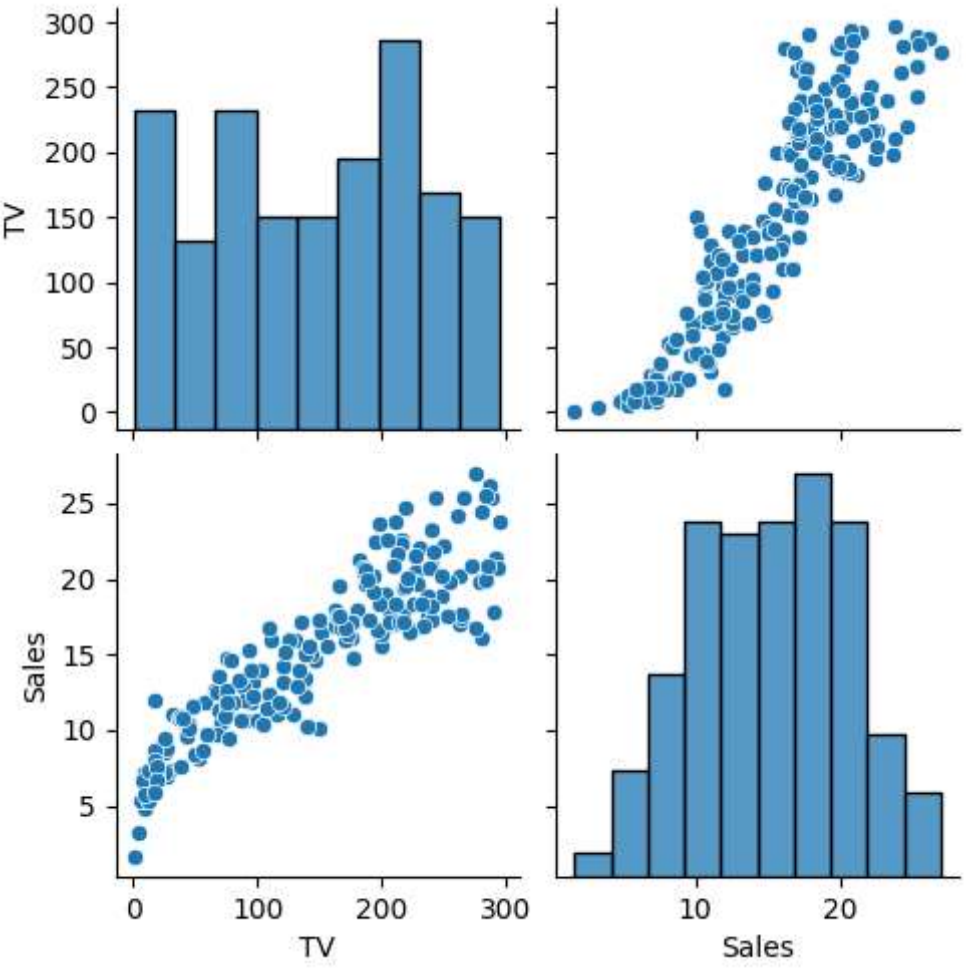
	TV	Radio	Newspaper	Sales
195	38.2	3.7	13.8	7.6
196	94.2	4.9	8.1	14.0
197	177.0	9.3	6.4	14.8
198	283.6	42.0	66.2	25.5
199	232.1	8.6	8.7	18.4

In [22]:
```
plt.figure(figsize = (10, 10))
sns.heatmap(df.corr(), annot = True)
```

Out[22]: <Axes: >



In [23]:
```
In [6]: df.drop(columns = ["Radio", "Newspaper"], inplace = True)
sns.pairplot(df)
df.Sales = np.log(df.Sales)
```



In [24]:
```
features = df.columns[0:2]
target = df.columns[-1]
X = df[features].values
y = df[target].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, randc
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)
```

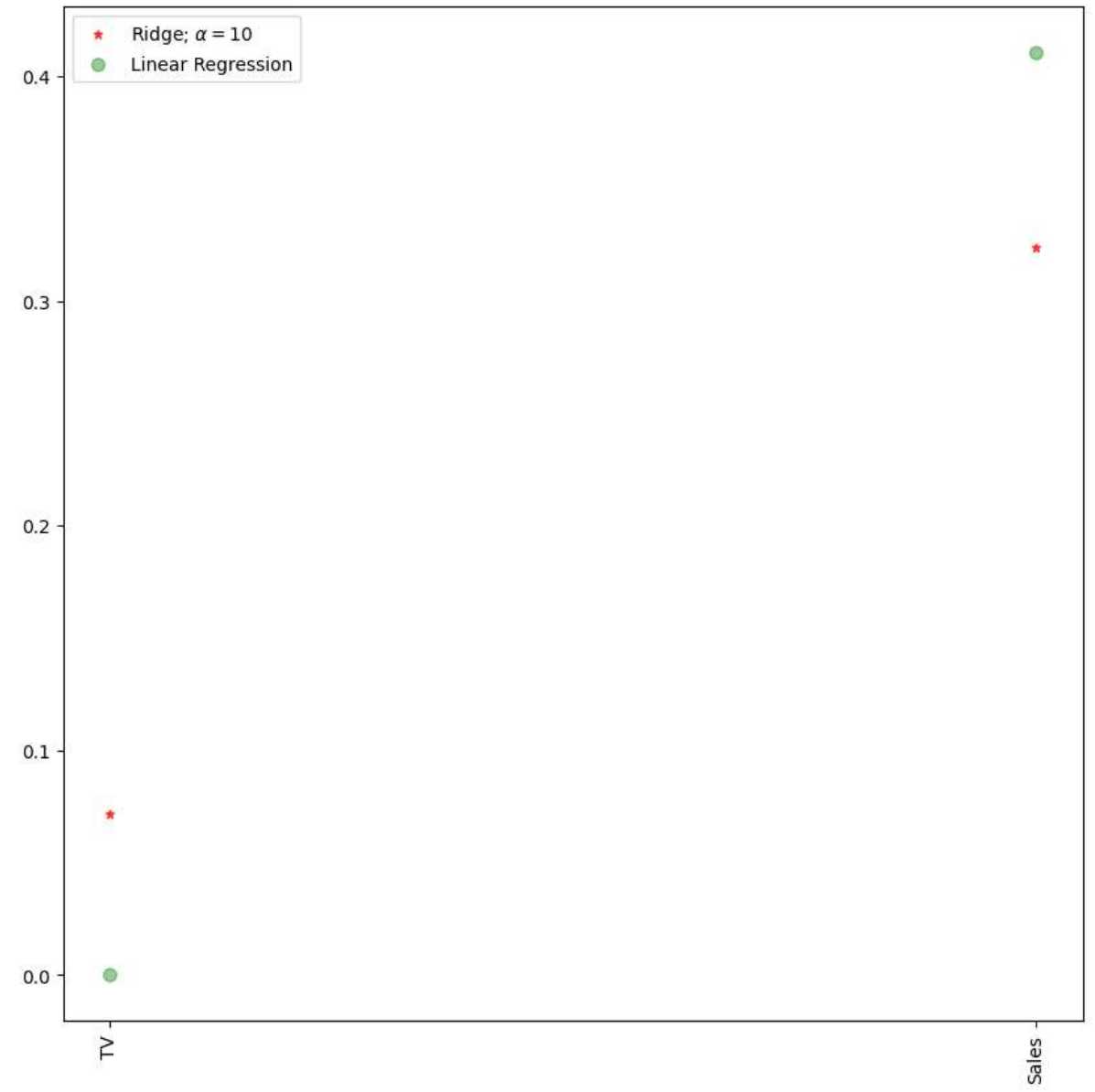
In [25]:
```
lr = LinearRegression()
lr.fit(X_train, y_train)
actual = y_test
train_score_lr = lr.score(X_train, y_train)
test_score_lr = lr.score(X_test, y_test)
print("\nLinear Regression Model:\n")
print("The train score for lr model is {}".format(train_score_lr))
print("The test score for lr model 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 [26]:
```
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 [45]:
```
plt.figure(figsize = (10, 10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markers
#plt.
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,
plt.xticks(rotation = 90)
plt.legend()
plt.show()
```

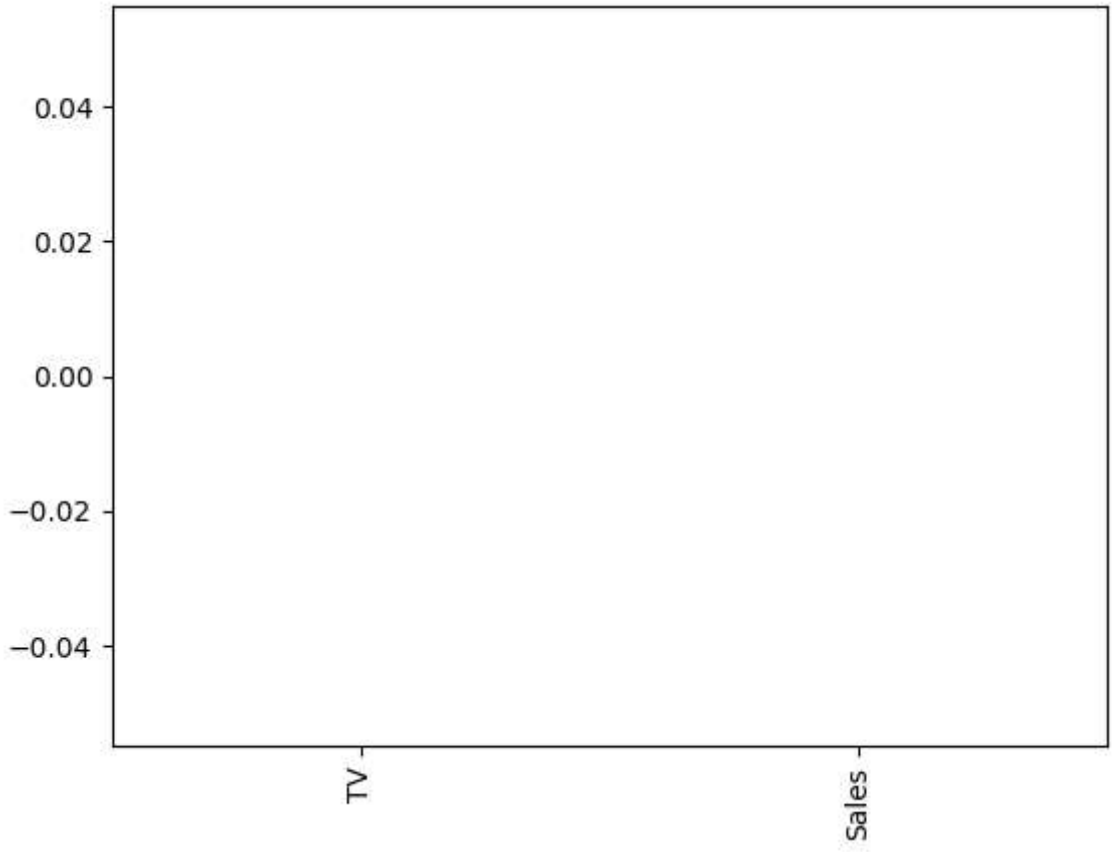


In [46]:
```
print("\nLasso Model: \n")
lasso = Lasso(alpha = 10)
lasso.fit(X_train,y_train)
train_score_ls =lasso.score(X_train,y_train)
test_score_ls =lasso.score(X_test,y_test)
print("The train score for ls model is {}".format(train_score_ls))
print("The test score for ls model is {}".format(test_score_ls))
```

```
Lasso Model:

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

In [47]:
```
pd.Series(lasso.coef_, features).sort_values(ascending = True).plot(kind = "ba
```

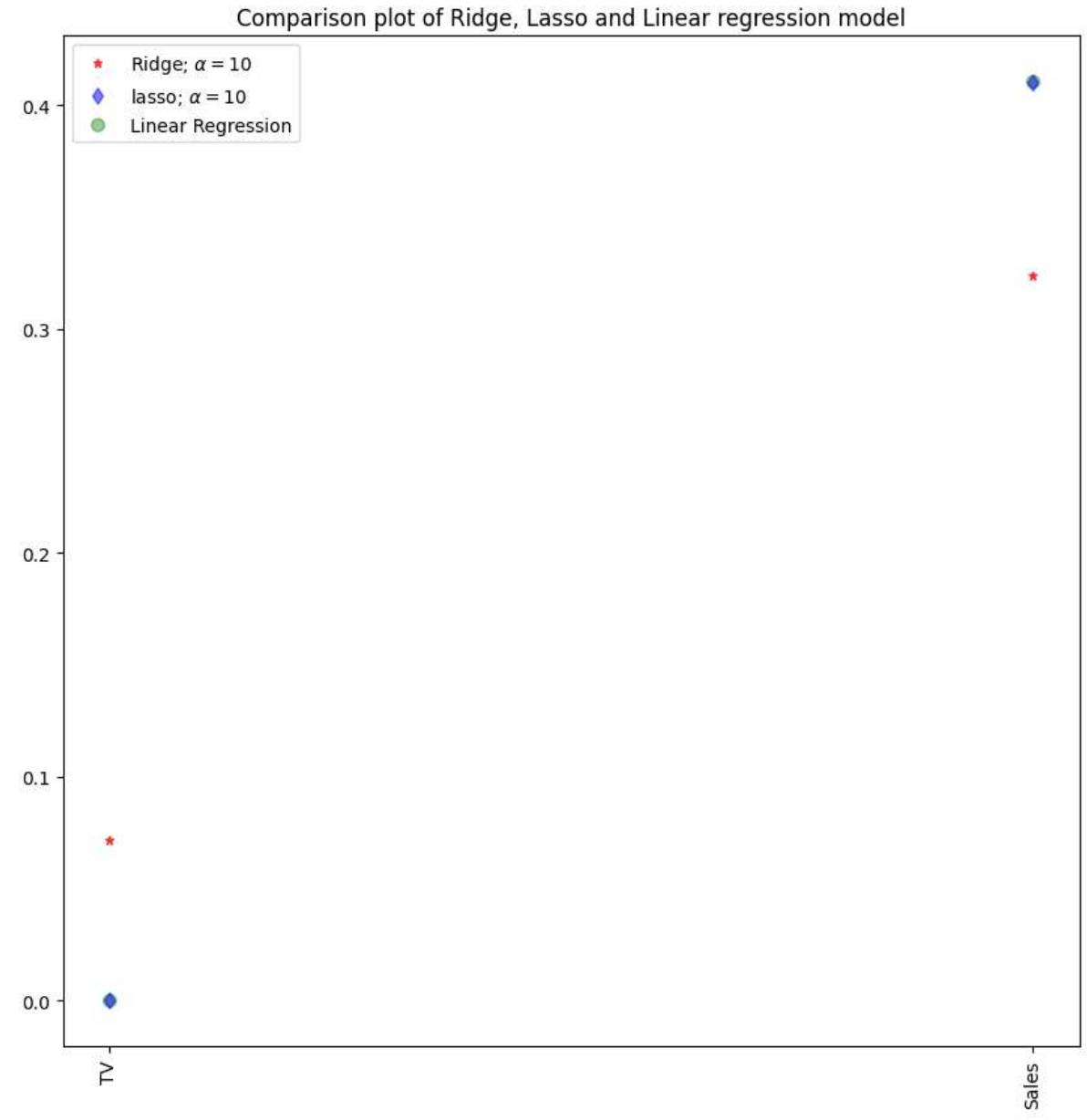
Out[47]: <Axes: >



In [48]:
```
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))
```

```
0.9999999343798134
0.9999999152638072
```

In [49]:
```
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='d',markersize=6,col
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 [50]:
```
from sklearn.linear_model import RidgeCV
ridge_cv = RidgeCV(alphas = [0.0001, 0.001,0.01, 0.1, 1, 10]).fit(X_train, y_t
print("The train score for ridge model is {}".format(ridge_cv.score(X_train, y
print("The test score for ridge model is {}".format(ridge_cv.score(X_test, y_t
```

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

ELASTIC NET

In [53]:
```
from sklearn.linear_model import ElasticNet
regr=ElasticNet()
regr.fit(X,y)
print(regr.coef_)
print(regr.intercept_)
```

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

In [54]:
```
y_pred_elastic=regr.predict(X_train)
```

In [55]:
```
mean_squared_error=np.mean((y_pred_elastic-y_train)**2)
print("MSE",mean_squared_error)
```

```
MSE 0.5538818050142158
```