In [1]:
```
import numpy as np
import pandas as pd
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 [2]:
```
df=pd.read_csv(r"C:\Users\Jayadeep\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.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]:
```
df.shape
```

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

In [6]:
```
df.describe
```

Out[6]:
```
<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 [7]:
```
df.info()
```

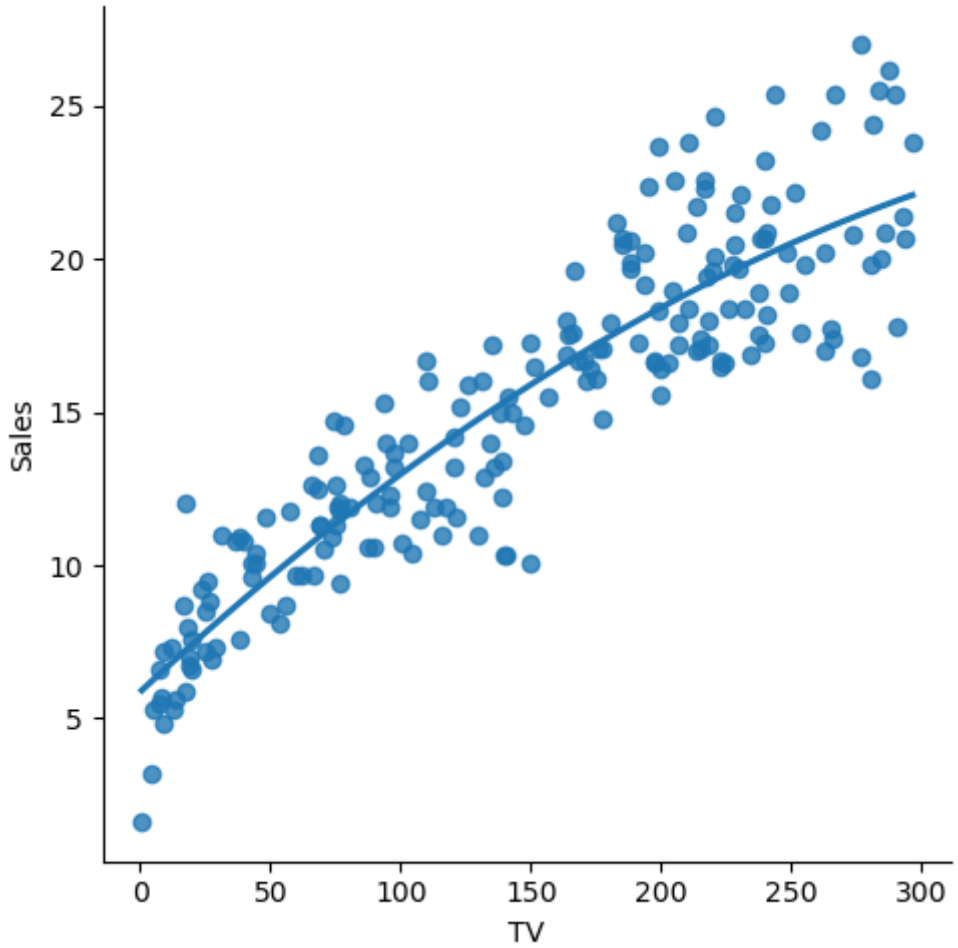
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   TV         200 non-null    float64
 1   Radio      200 non-null    float64
 2   Newspaper  200 non-null    float64
 3   Sales      200 non-null    float64
dtypes: float64(4)
memory usage: 6.4 KB
```

In [8]:
```
df.isna().any()
```

Out[8]:
```
TV           False
Radio        False
Newspaper    False
Sales        False
dtype: bool
```

In [9]:
```
sns.lmplot(x='TV',y='Sales',data=df,order=2,ci=None)
```

Out[9]: <seaborn.axisgrid.FacetGrid at 0x1f3184611f0>



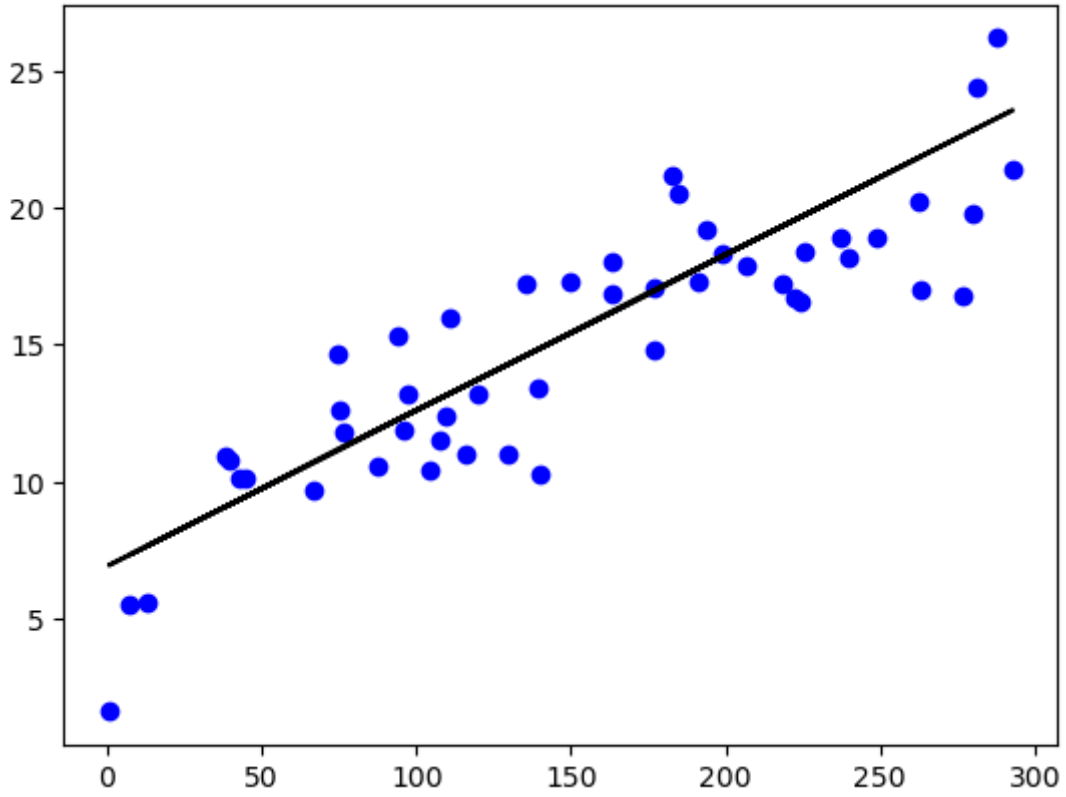
In [10]:
```
x=np.array(df['TV']).reshape(-1,1)
y=np.array(df['Sales']).reshape(-1,1)
```

In [11]:
```
df.dropna(inplace=True)
```

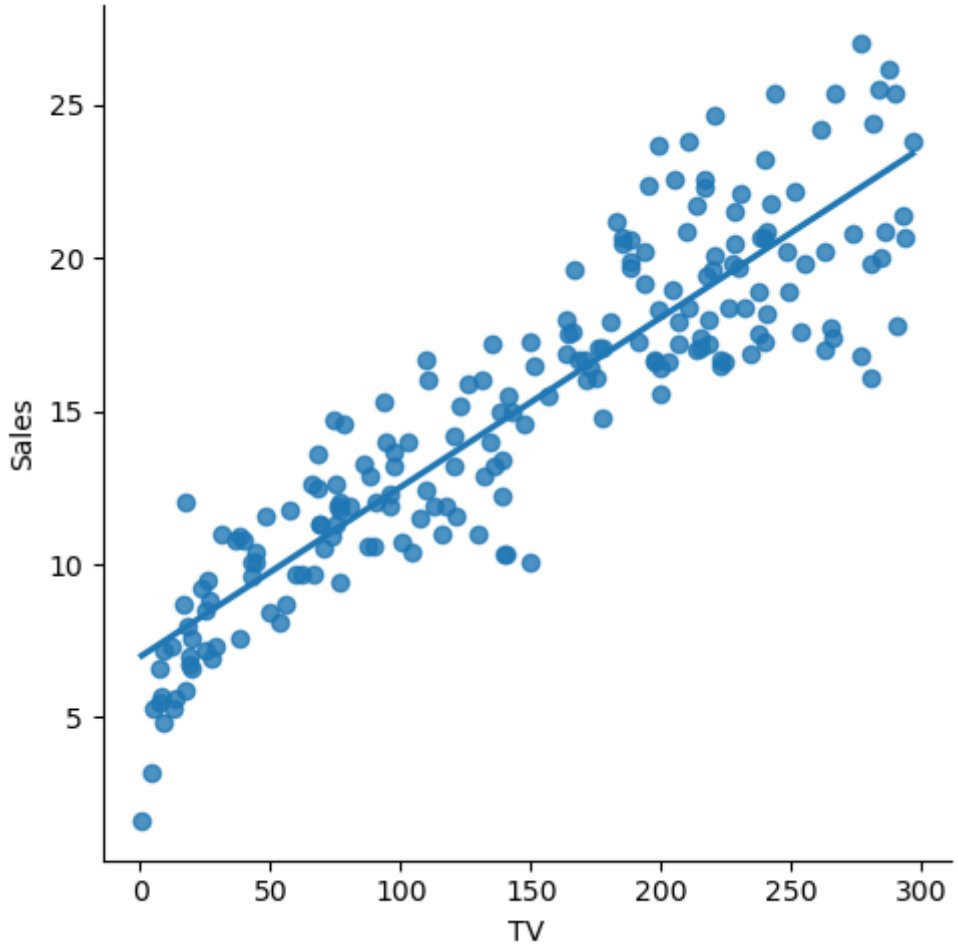
In [12]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)
#splitting data into train and test
regr=LinearRegression()
regr.fit(x_train,y_train)
print(regr.score(x_test,y_test))
```

```
0.7386521246956955
```

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="TV",y="Sales",data=df500,order=1,ci=None)
plt.show()
```



In [15]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
```

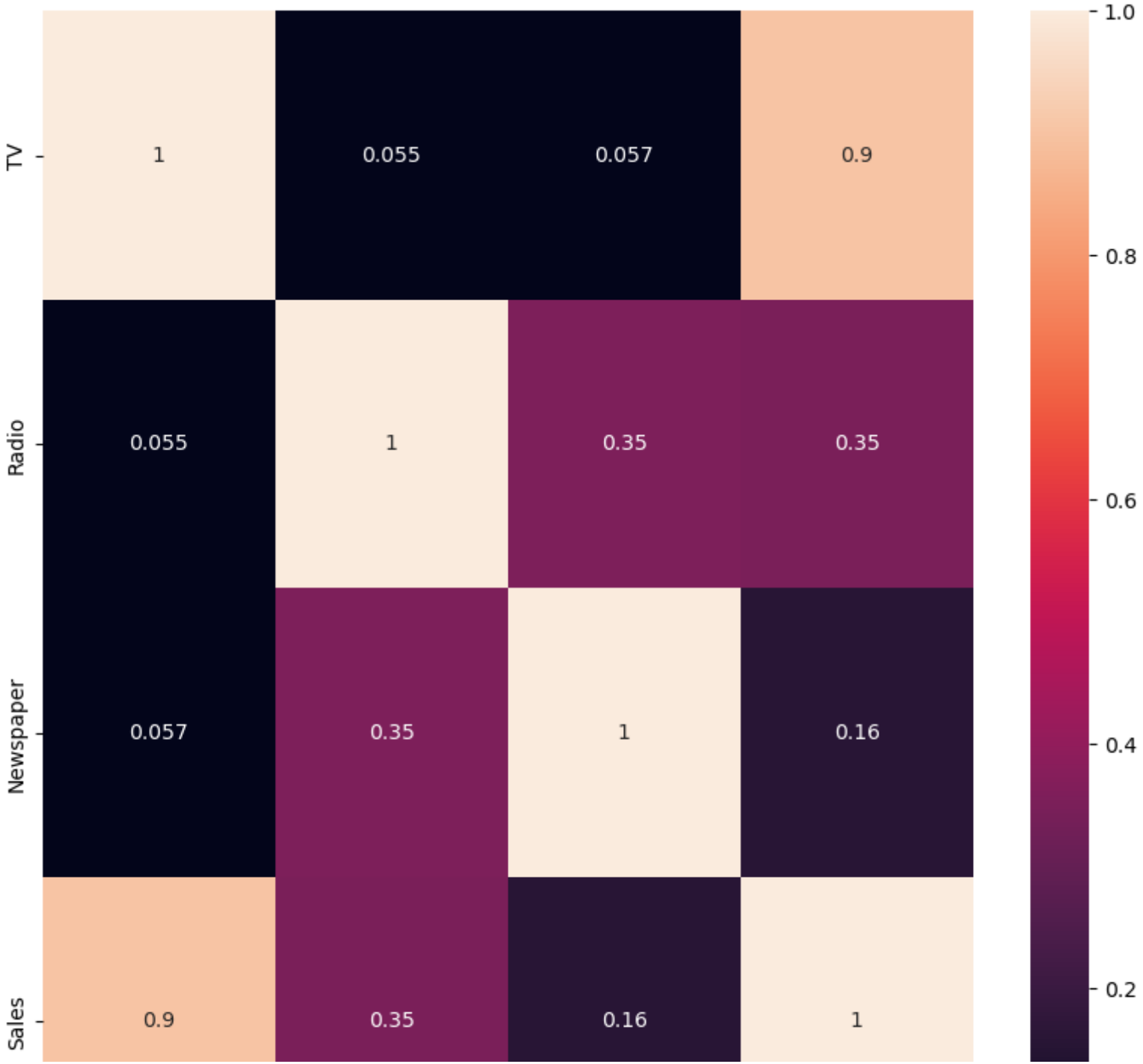
In [16]:
```
#train model
model=LinearRegression()
model.fit(x_train,y_train)
#Evaluation the model on the test set
y_pred=model.predict(x_test)
r2=r2_score(y_test,y_pred)
print("R2 score:",r2)
```

```
R2 score: 0.7386521246956955
```

RIDGE AND LASSO

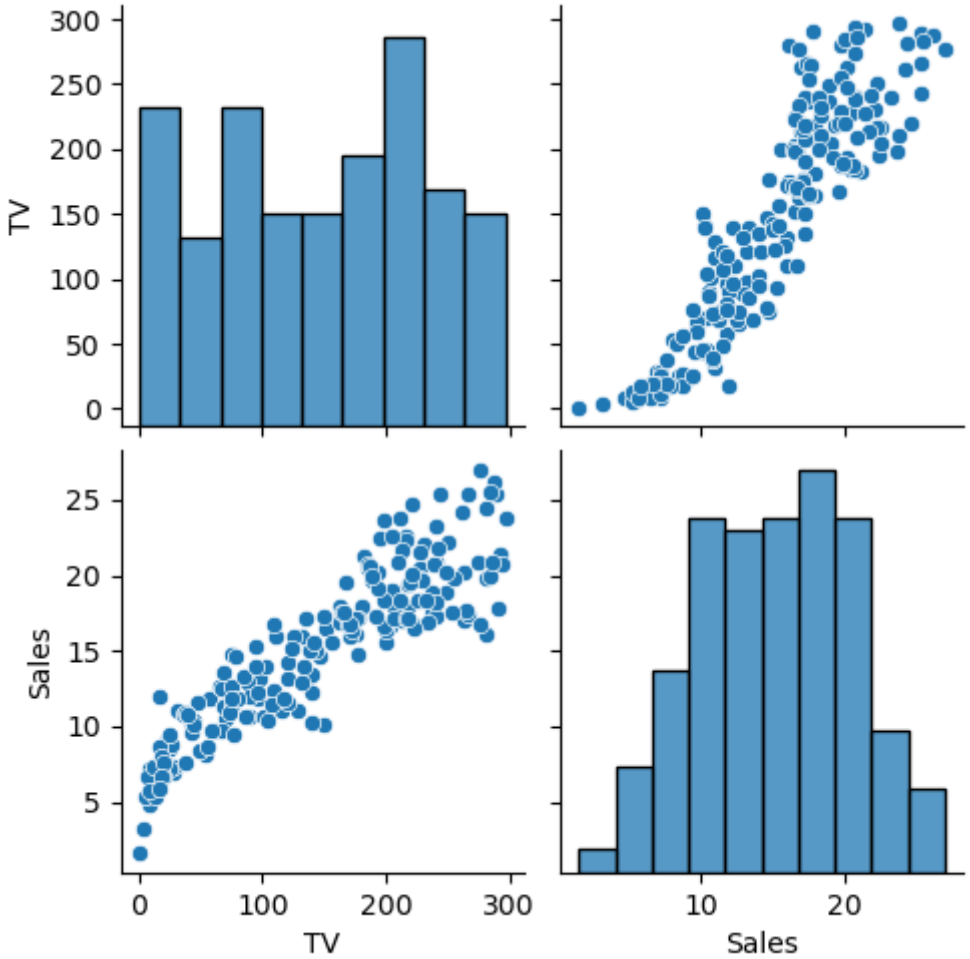
In [17]:
```
from sklearn.linear_model import Lasso,Ridge
from sklearn.preprocessing import StandardScaler
```

In [18]:
```
plt.figure(figsize = (10, 10))
sns.heatmap(df500.corr(), annot = True)
plt.show()
```


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


TV
Radio
Newspaper
Sales

In [19]:
```
df.drop(columns =['Radio','Newspaper'], inplace = True)
#pairplot
sns.pairplot(df)
df.Sales = np.log(df.Sales)
plt.show()
```



In [20]:
```
features=df.columns[0:2]
target=df.columns[-1]
```

In [21]:
```
#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, 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 [22]:
```
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 [23]:
```
#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))
```

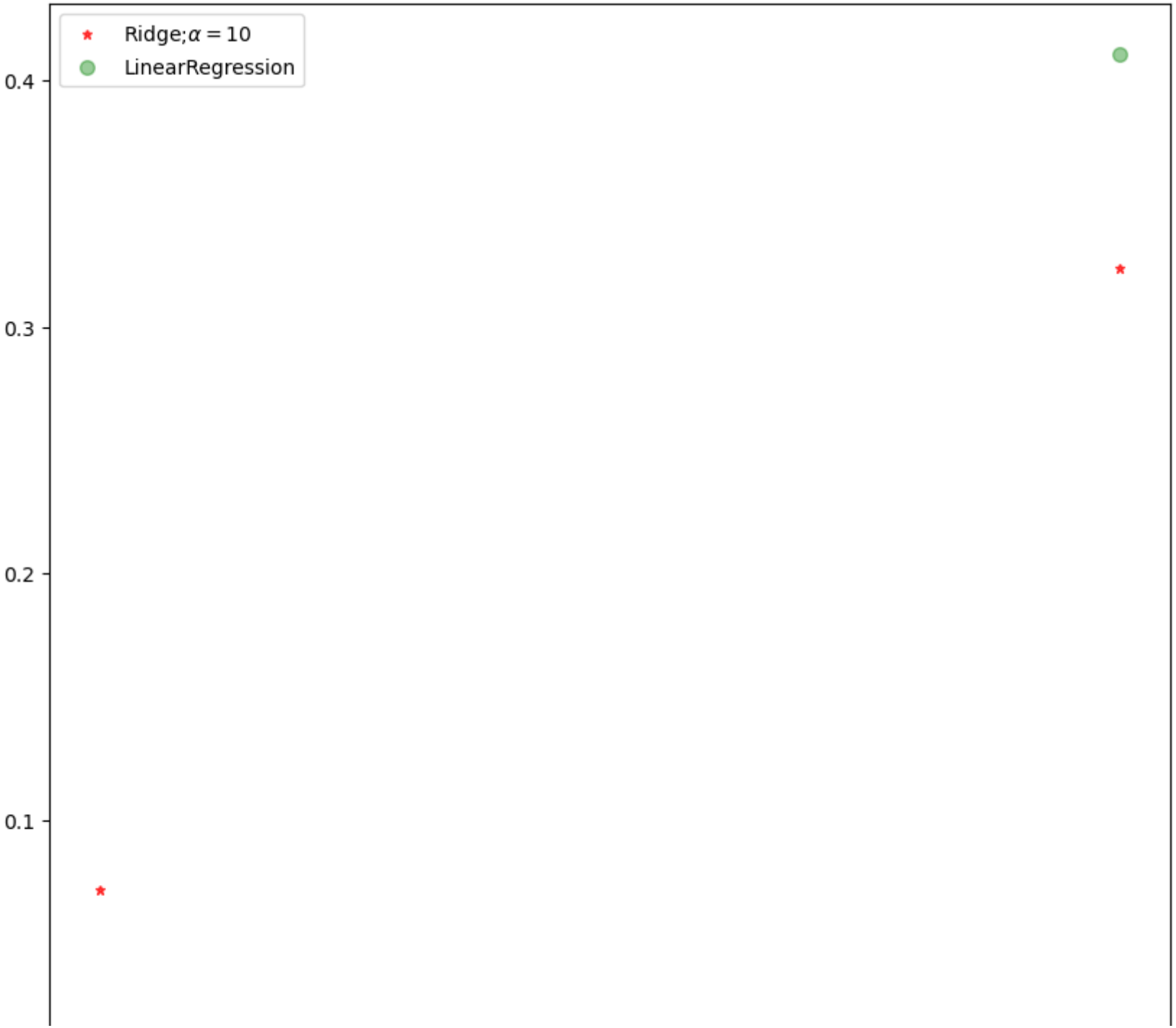
```
Ridge Model:

The train score for ridge model is 0.9902871391941607
The test score for ridge model is 0.9844266285141215
```

In [24]:
```
plt.figure(figsize=(10,10))
```

Out[24]:
```
<Figure size 1000x1000 with 0 Axes>
```

In [25]:
```
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker="*",markersize=5,color='red',label=r'Ridge;$\alpha=
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker="o",markersize=7,color='green',label='LinearRegression')
plt.xticks(rotation=90)
plt.legend()
plt.show()
```

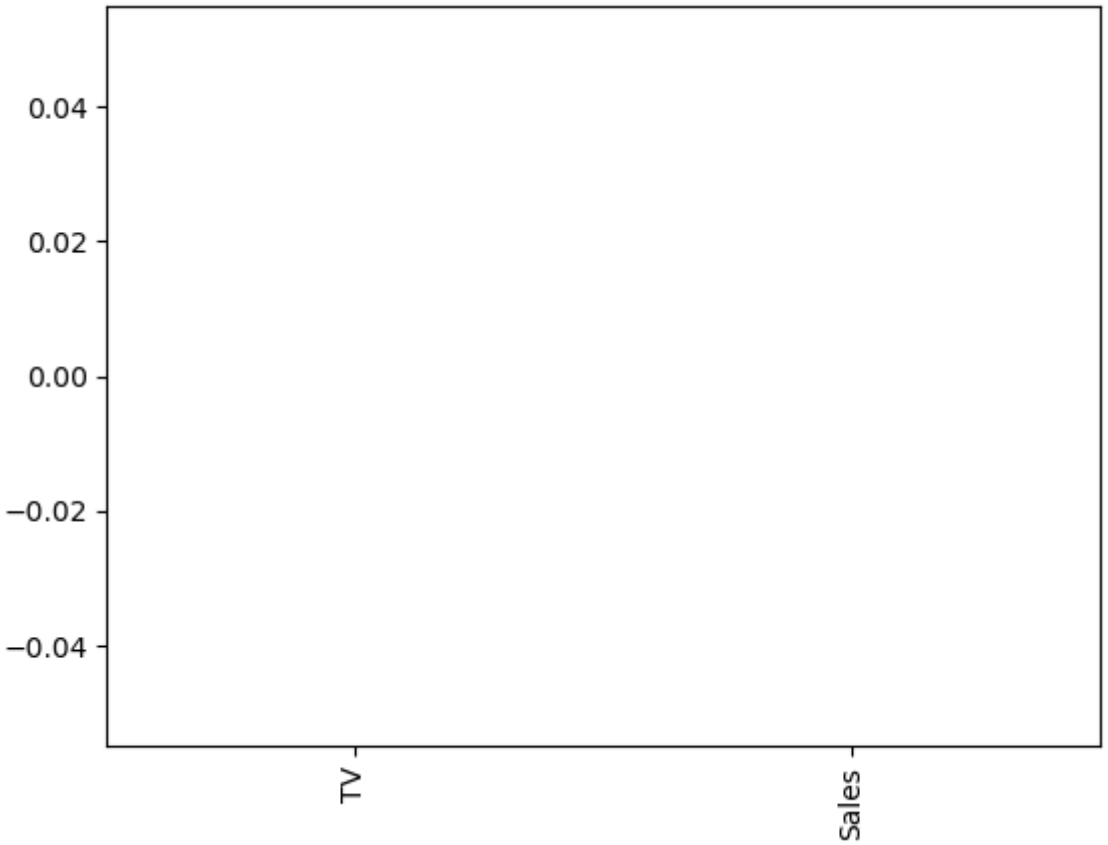

Ridge;$\alpha = 10$
LinearRegression
0.4
0.3
0.2
0.1

In [26]:
```
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 [27]:
```
pd.Series(lasso.coef_, features).sort_values(ascending = True).plot(kind = "bar")
plt.show()
```



In [28]:
```
#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_train, y_train)
#score
print(lasso_cv.score(X_train, y_train))
print(lasso_cv.score(X_test, y_test))
```

```
0.9999999343798134
0.9999999152638072
```

In [29]:
```
plt.figure(figsize = (10, 10))
#add plot for ridge regression
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5,color='red',label=r'Ridge; $\alpha
#add plot for lasso regression
plt.plot(lasso_cv.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color='blue',label=r'lasso; $\alpha = grid$
#add plot for linear model
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color='green',label='Linear Regression')
#rotate axis
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 [30]:
```
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 0.9999999999976276
The train score for ridge model is 0.9999999999962477
```

#Elastic

In [35]:
```
from sklearn.linear_model import ElasticNet
regr=ElasticNet()
regr.fit(X,y)
print(regr.coef_)
print(regr.intercept_)
```

```
[0.00417976 0.        ]
2.0263839193110043
```

In [36]:
```
y_pred_elastic=regr.predict(X_train)
```

In [37]:
```
mean_squared_error=np.mean((y_pred_elastic-y_train)**2)
print(mean_squared_error)
```

```
0.5538818050142152
```

In [ ]: