In [1]:

```
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]:

```
data=pd.read_csv(r"C:\Users\shaha\OneDrive\Desktop\Excel\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]:

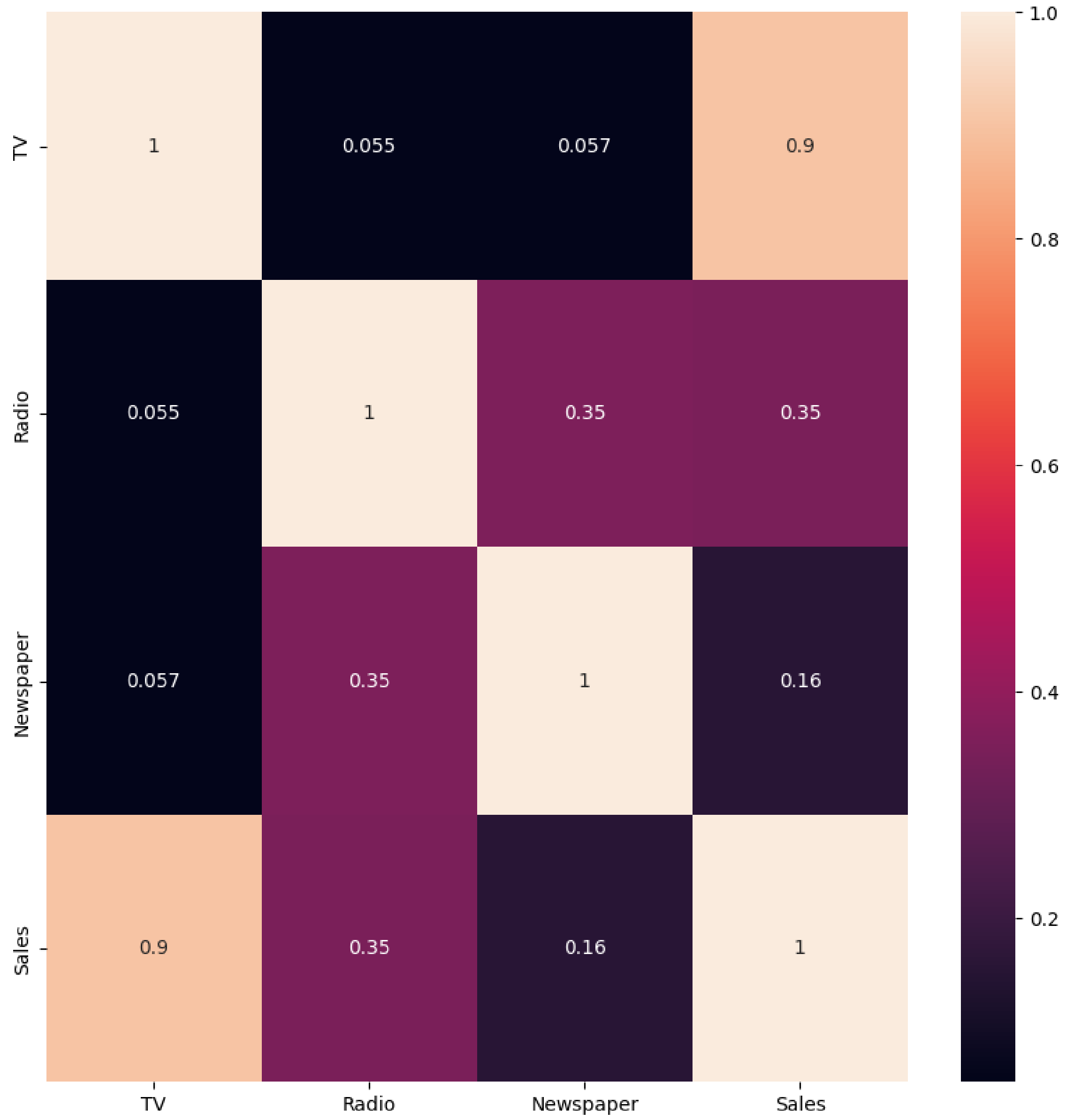
	TV	Radio	Newspaper	Sales
195	38.2	3.7	13.8	7.6
196	94.2	4.9	8.1	14.0
197	177.0	9.3	6.4	14.8
198	283.6	42.0	66.2	25.5
199	232.1	8.6	8.7	18.4

In [5]:

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

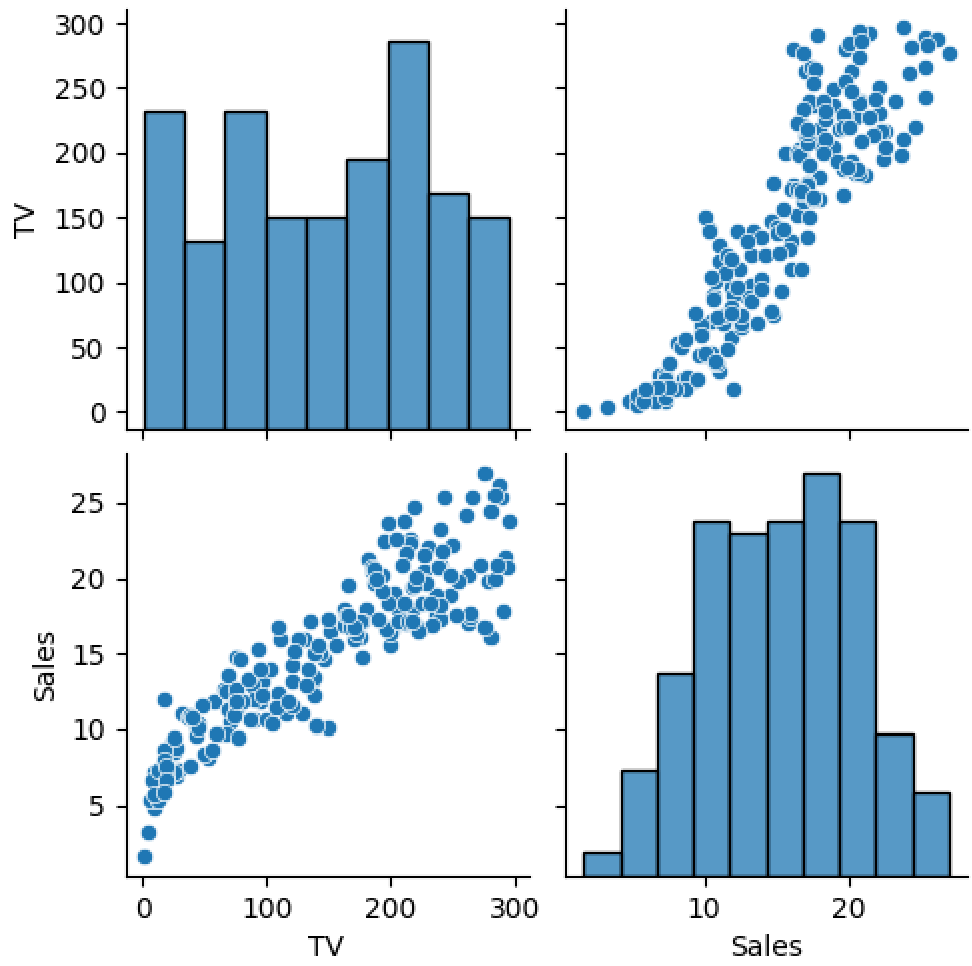
Out[5]:

```
<Axes: >
```



In [6]:

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



In [7]:

```
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 [8]:

```
#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
```

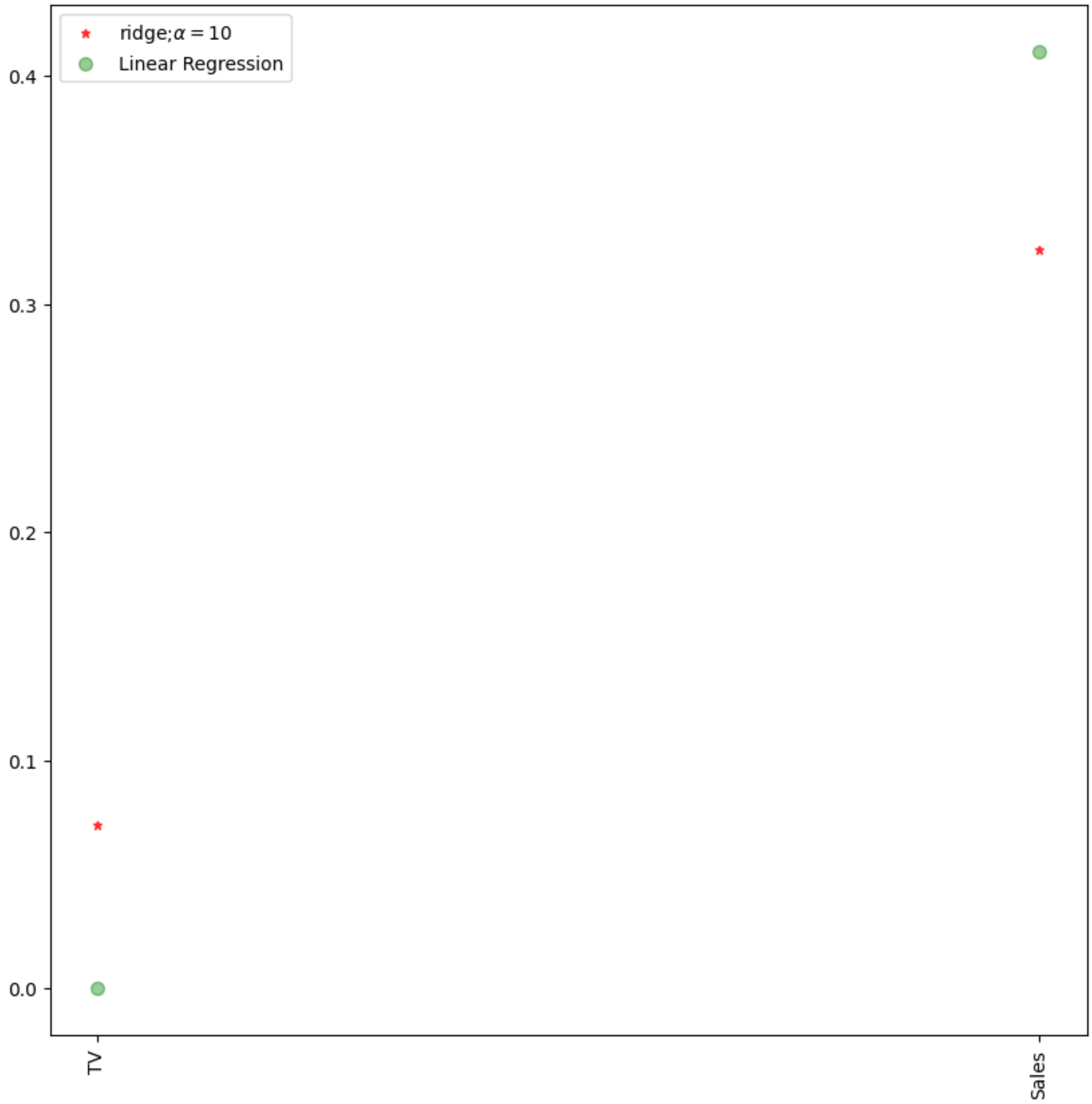
In [9]:

```
#Ridge Regression Model
ridgeReg = Ridge(alpha=10)

ridgeReg.fit(x_train,y_train)

#train and test score for 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 [10]:

```
plt.figure(figsize = (10,10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5,colo
         label=r'ridge;$\alpha=10$',zorder=7)
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color='gre
plt.xticks(rotation=90)
plt.legend()
plt.show()
```



In [14]:

```
#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))
```

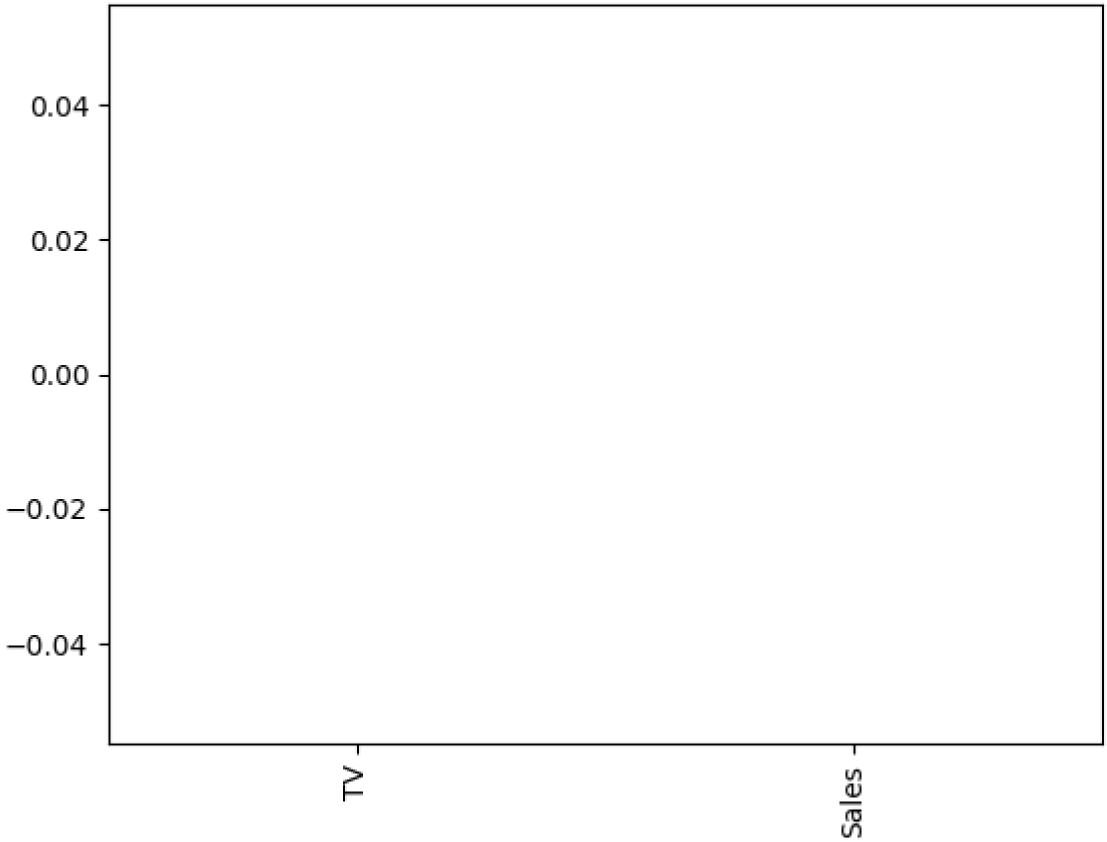
```
Lasso Model:

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

In [15]:

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

Out[15]:

```
<Axes: >
```



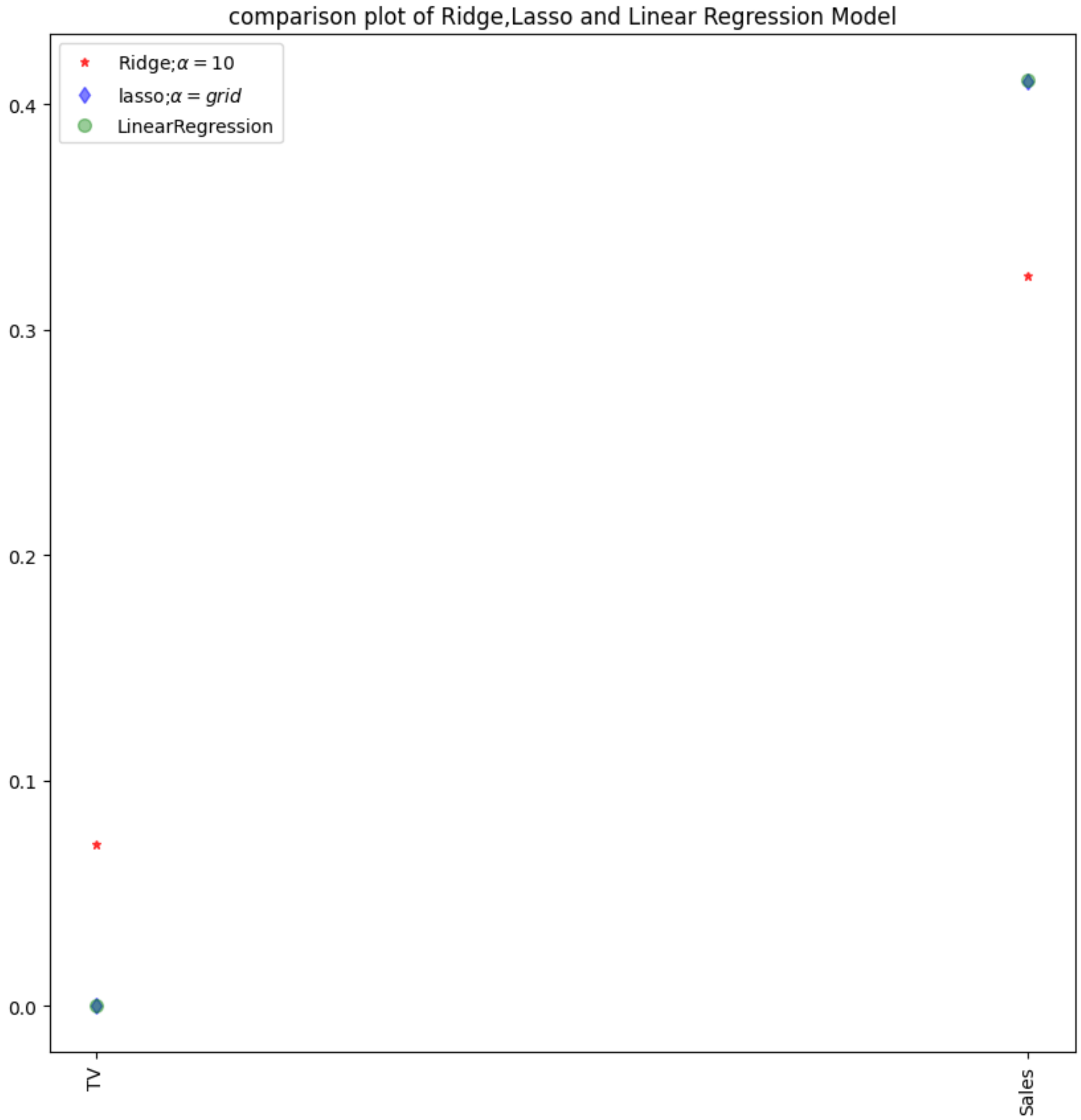
In [20]:

```
#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_t

#score
print(lasso_cv.score(x_train,y_train))
print(lasso_cv.score(x_test,y_test))
```

```
0.9999999343798134
0.9999999152638072
```

In [22]:

```
#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,colo
plt.plot(lasso_cv.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color='blue',
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker = 'o',markersize=7,color='g
plt.xticks(rotation=90)
plt.legend()
plt.title("comparison plot of Ridge,Lasso and Linear Regression Model")
plt.show()
```



In [25]:

```
#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 0.9999999999976281
The train score for ridge model is 0.9999999999962489
```

In [26]:

```
from sklearn.linear_model import ElasticNet
regr = ElasticNet()
regr.fit(x,y)
print(regr.coef_)
print(regr.intercept_)
```

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

In [27]:

```
y_pred_elastic = regr.predict(x_train)
```

In [29]:

```
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 0.5538818050142152
```