Ridge and Lasso Regression

Advertising dataset

In [19]:

```
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 [20]:

```
data=pd.read_csv(r"C:\Users\91949\Downloads\Advertising.csv")
data
```

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
...	...	...	...	...
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 [21]:

```
data.head()
```

Out[21]:

	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 [22]:

```
data.tail()
```

Out[22]:

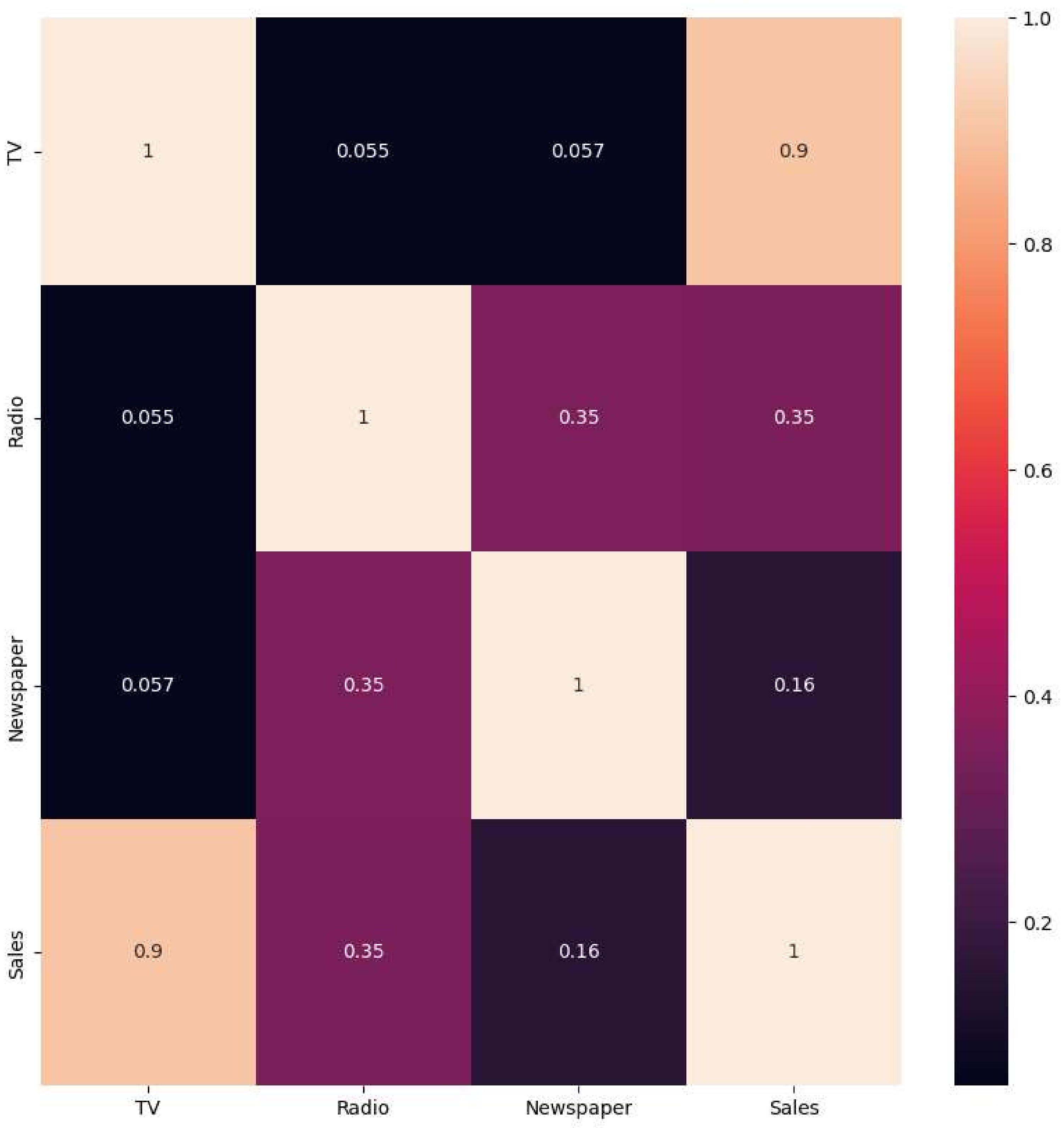
	TV	Radio	Newspaper	Sales
195	38.2	3.7	13.8	7.6
196	94.2	4.9	8.1	14.0
197	177.0	9.3	6.4	14.8
198	283.6	42.0	66.2	25.5
199	232.1	8.6	8.7	18.4

In [23]:

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

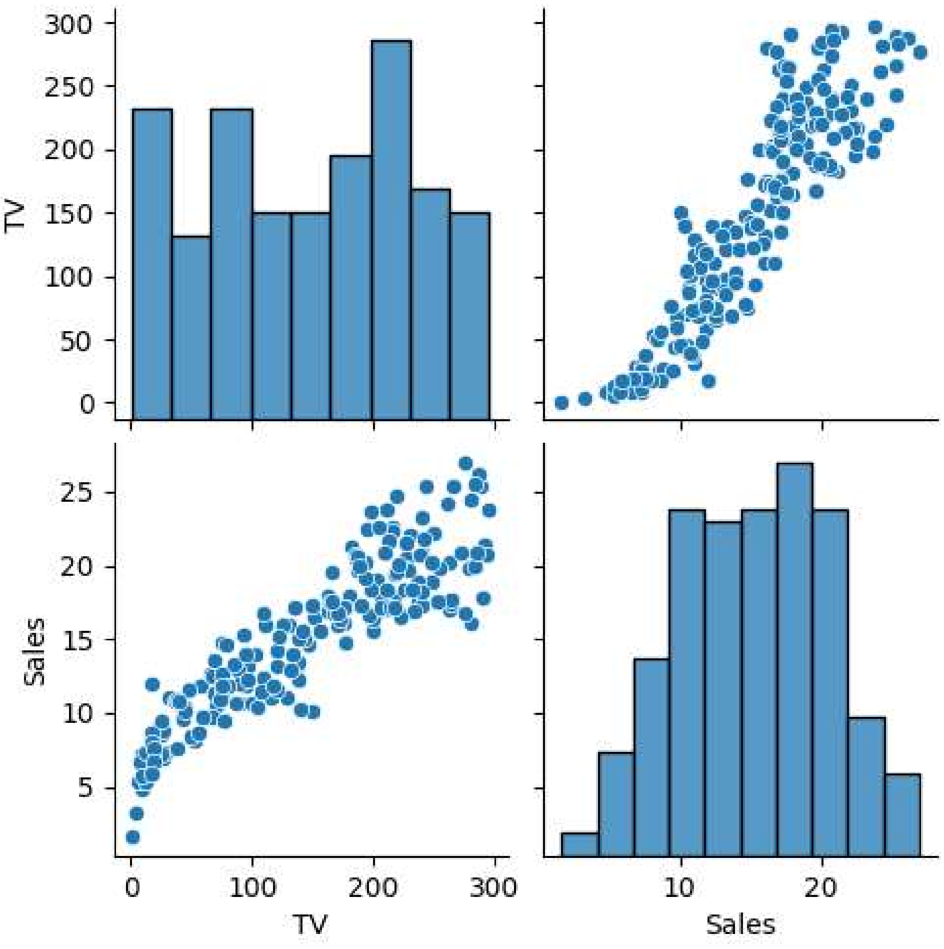
Out[23]:

```
<Axes: >
```



In [24]:

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



In [25]:

```
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 [26]:

```
#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
```

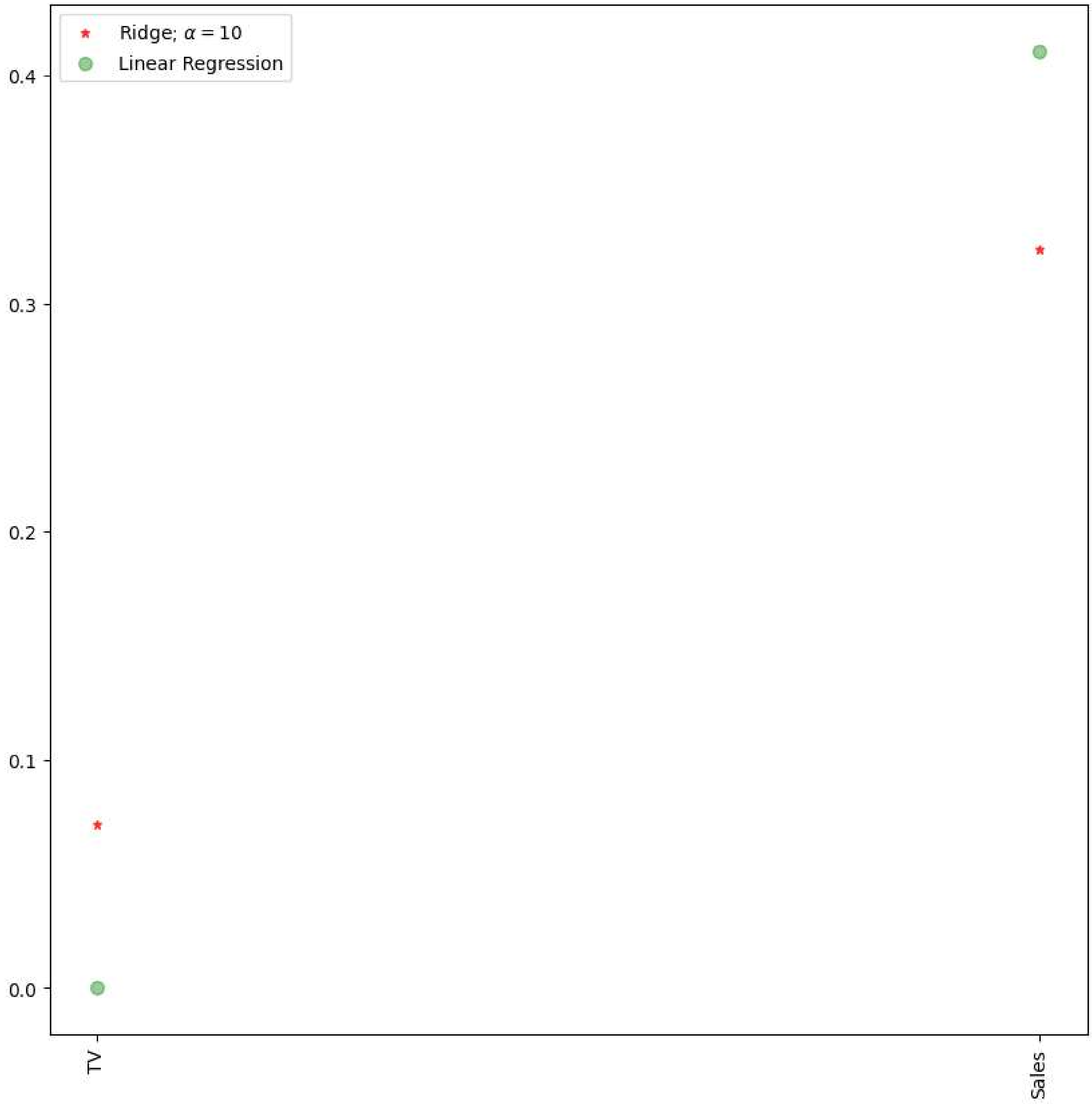
In [27]:

```
#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 [28]:

```
plt.figure(figsize = (10, 10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5,color='red',label=r'Ridge; $\alpha
#plt.plot(rr100.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color='blue',label=r'Ridge; $\alpha = 100$')
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color='green',label='Linear Regression')
plt.xticks(rotation = 90)
plt.legend()
plt.show()
```



In [29]:

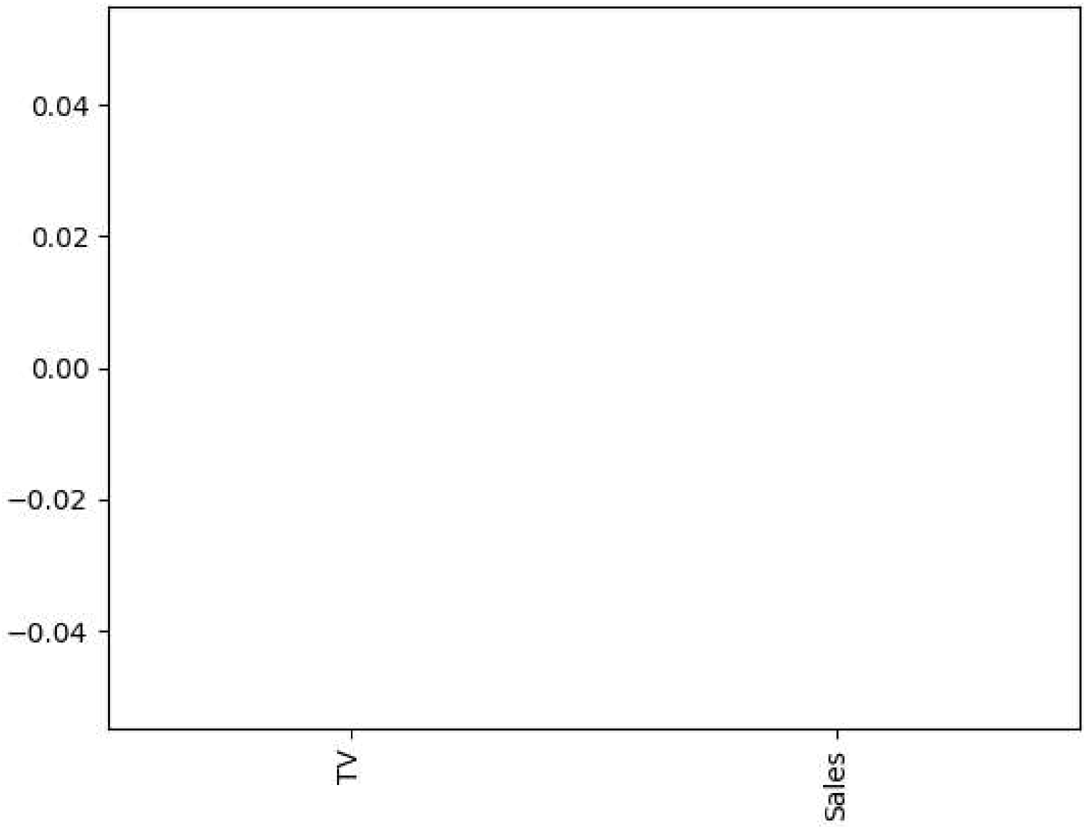
```
#Lasso regression model
print("\nLasso Model: \n")
lasso = Lasso(alpha = 10)
lasso.fit(X_train,y_train)
train_score_ls =lasso.score(X_train,y_train)
test_score_ls =lasso.score(X_test,y_test)
print("The train score for ls model is {}".format(train_score_ls))
print("The test score for ls model is {}".format(test_score_ls))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

```
Lasso Model: 

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

In [30]:

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

Out[30]: <Axes: >



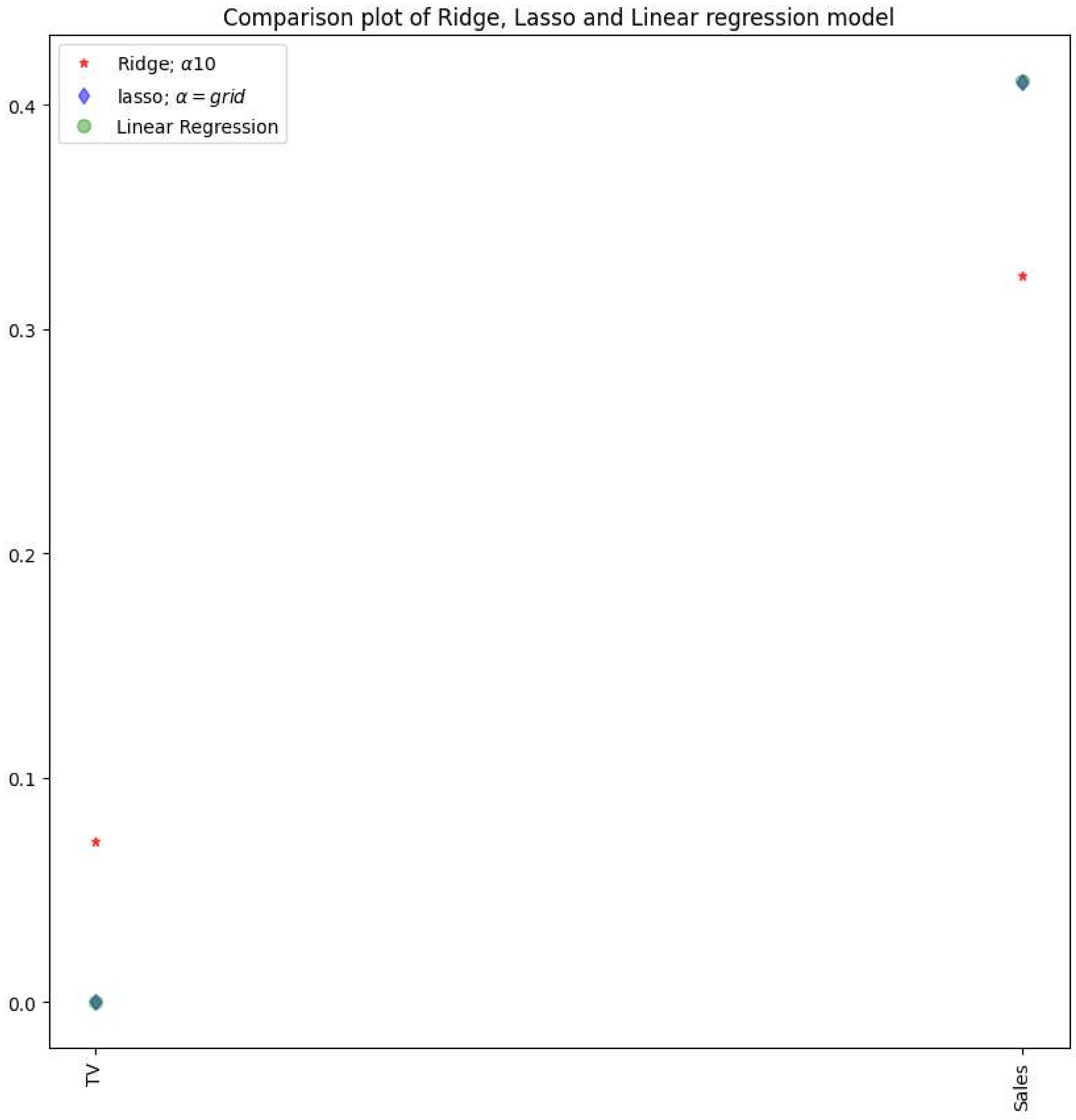
In [31]:

```
#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 [32]:

```
#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='red',label=r'Ridge; $\alpha1
#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()

```



In [33]:

```
#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
```

ELASTICNET

In [42]:

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

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

In [44]:

```python
y_pred_elastic=regr.predict(X_train)
```

In [45]:

```python
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 0.5538818050142158
```

vehicles dataset

In [27]:

```python
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 [28]:

```python
d=pd.read_csv(r"C:\Users\91949\Downloads\fiat500_VehicleSelection_Dataset.csv")
d
```

Out[28]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	price
0	1	lounge	51	882	25000	1	44.907242	8.611560	8900
1	2	pop	51	1186	32500	1	45.666359	12.241890	8800
2	3	sport	74	4658	142228	1	45.503300	11.417840	4200
3	4	lounge	51	2739	160000	1	40.633171	17.634609	6000
4	5	pop	73	3074	106880	1	41.903221	12.495650	5700
...	...	...	...	...	...	...	...	...	...
1533	1534	sport	51	3712	115280	1	45.069679	7.704920	5200
1534	1535	lounge	74	3835	112000	1	45.845692	8.666870	4600
1535	1536	pop	51	2223	60457	1	45.481541	9.413480	7500
1536	1537	lounge	51	2557	80750	1	45.000702	7.682270	5990
1537	1538	pop	51	1766	54276	1	40.323410	17.568270	7900

1538 rows × 9 columns

In [29]:

```python
d=d[['engine_power','age_in_days']]
d.columns=['ep','aid']
```

In [30]:

```python
d.describe()
```

Out[30]:

	ep	aid
count	1538.000000	1538.000000
mean	51.904421	1650.980494
std	3.988023	1289.522278
min	51.000000	366.000000
25%	51.000000	670.000000
50%	51.000000	1035.000000
75%	51.000000	2616.000000
max	77.000000	4658.000000

In [31]:

```python
d.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1538 entries, 0 to 1537
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   ep      1538 non-null   int64
 1   aid     1538 non-null   int64
dtypes: int64(2)
memory usage: 24.2 KB
```

In [32]:

```
d.fillna(method='ffill',inplace=True)
```

```
C:\Users\91949\AppData\Local\Temp\ipykernel_17204\2624214790.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-vi
ew-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  d.fillna(method='ffill',inplace=True)
```

In [33]:

```
x=np.array(d['ep']).reshape(-1,1)
y=np.array(d['aid']).reshape(-1,1)
```

In [34]:

```
d.dropna(inplace=True)
```

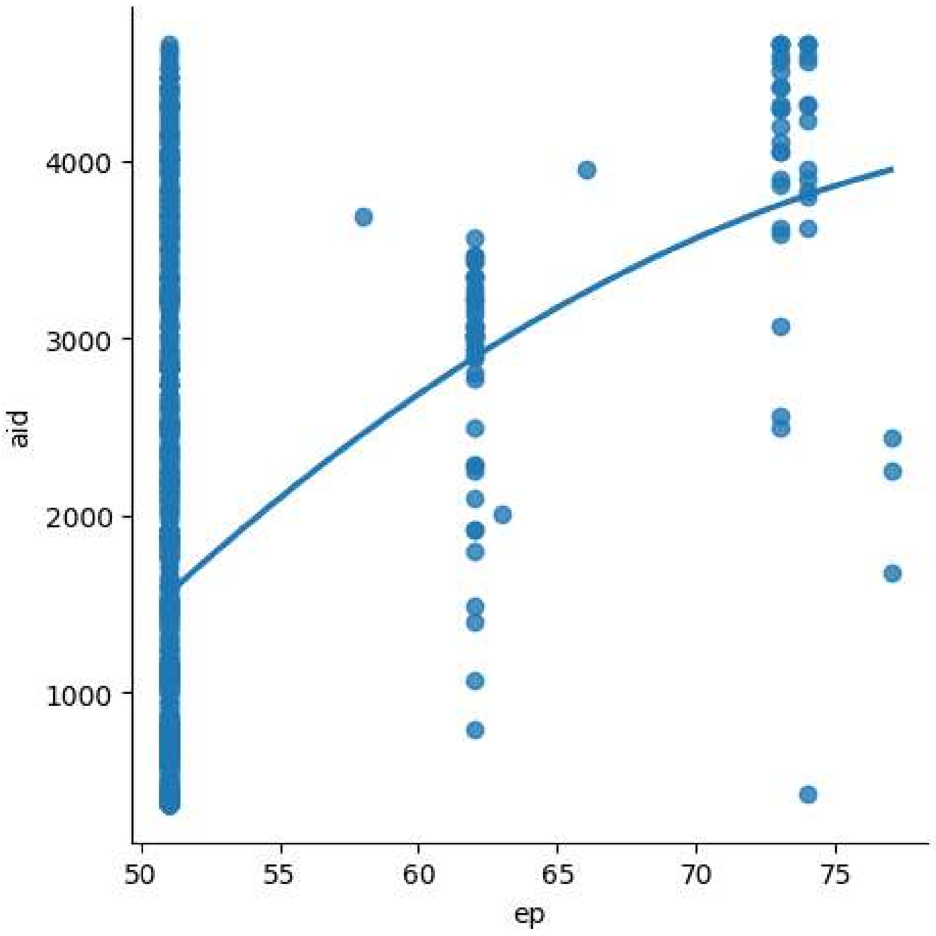
```
C:\Users\91949\AppData\Local\Temp\ipykernel_17204\1307611603.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-vi
ew-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  d.dropna(inplace=True)
```

In [35]:

```
sns.lmplot(x = "ep", y = "aid", data = d, order = 2, ci = None)
```

Out[35]:

```
<seaborn.axisgrid.FacetGrid at 0x240065c3510>
```



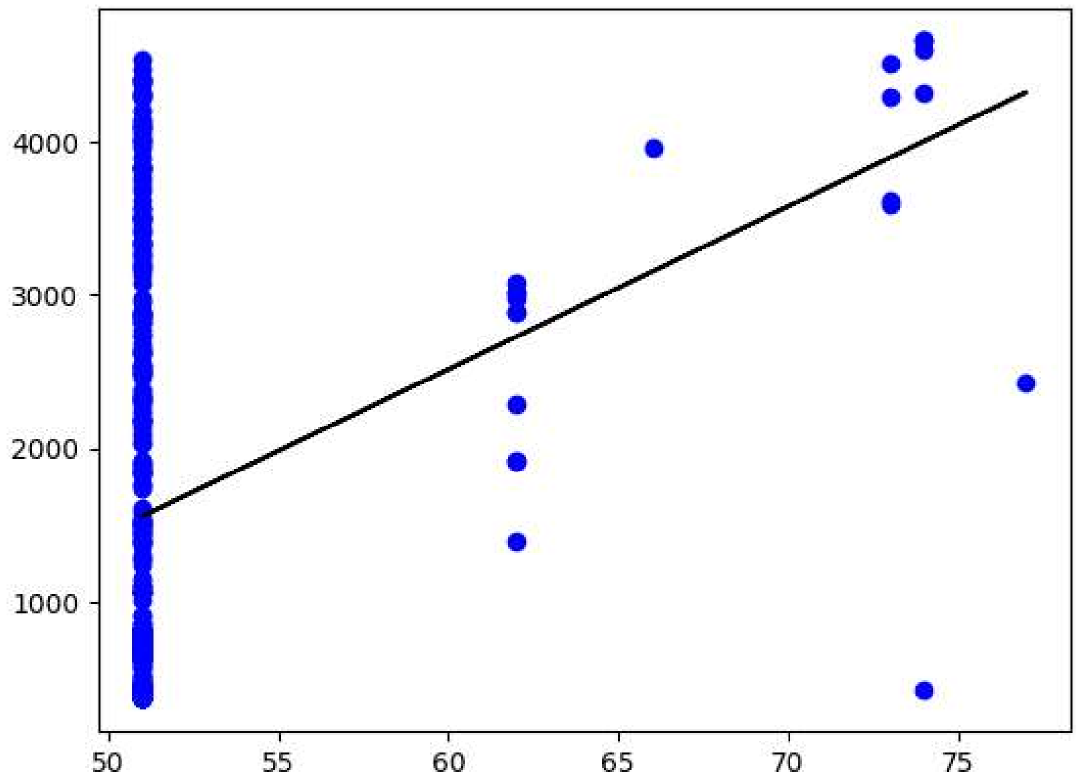
In [36]:

```
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))
```

```
0.08925408916467337
```

In [37]:

```
y_pred=regr.predict(x_test)
plt.scatter(x_test,y_test,color='b')
plt.plot(x_test,y_pred,color='k')
plt.show()
```

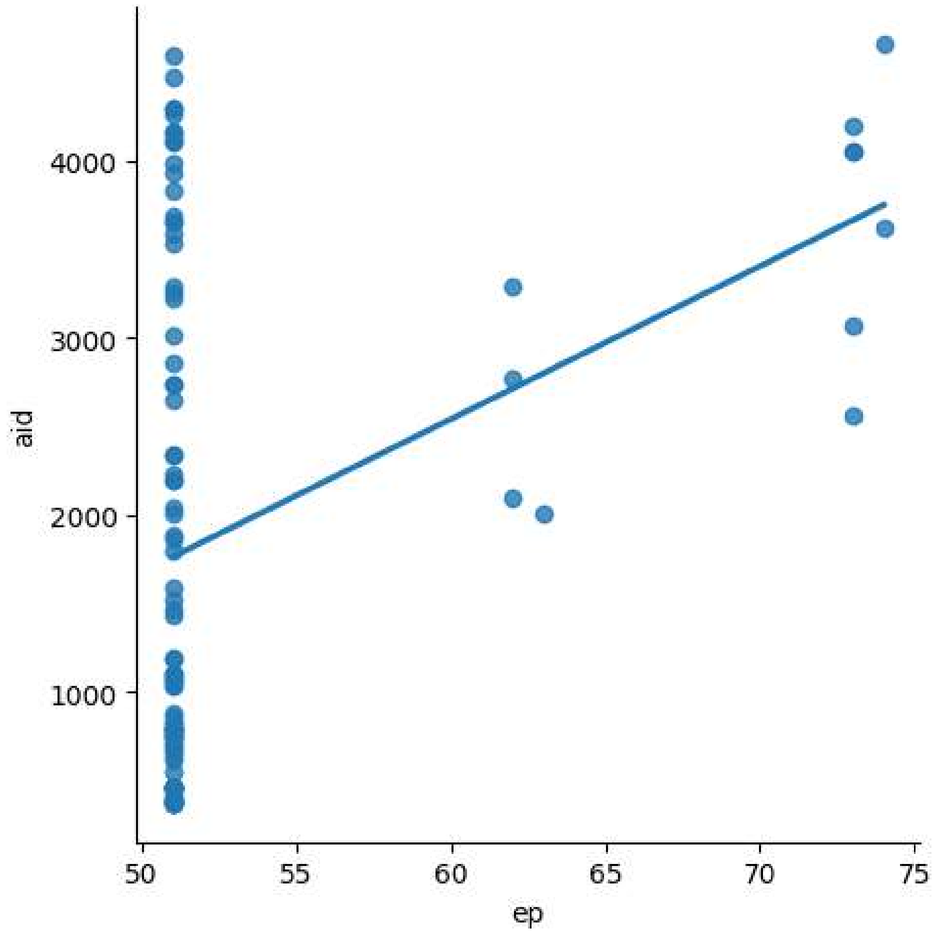


In [38]:

```
df100=d[:][:100]
sns.lmplot(x='ep',y='aid',data=df100,order=1,ci=None)
```

Out[38]:

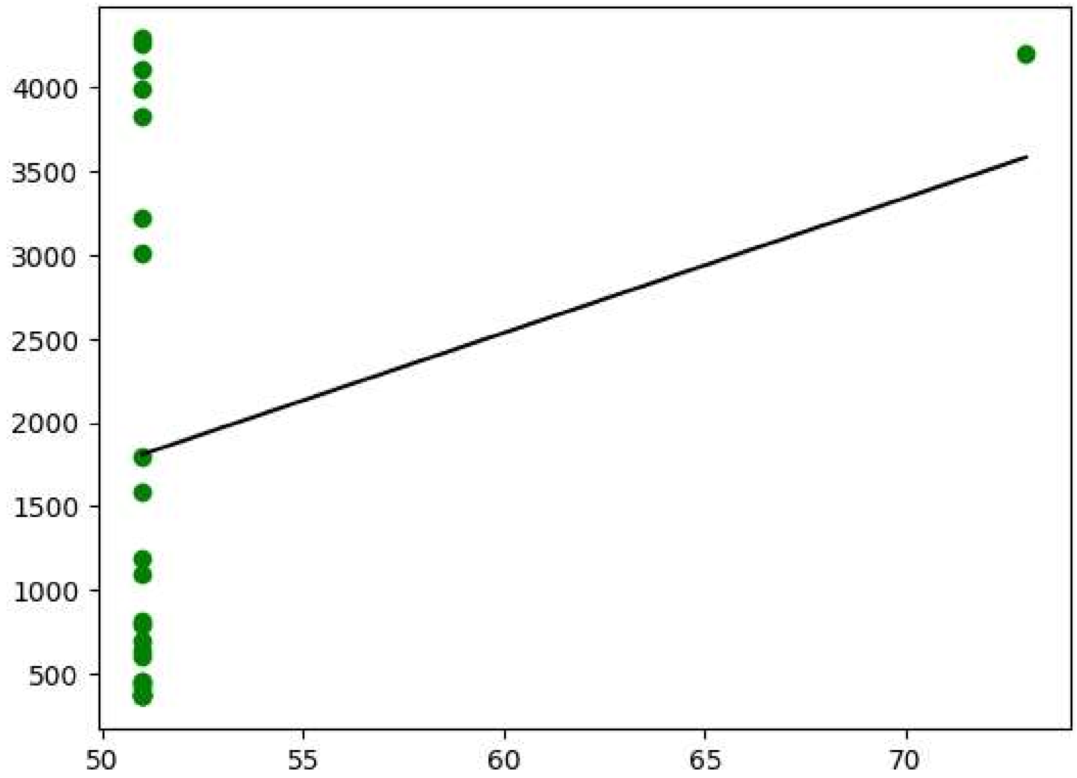
```
<seaborn.axisgrid.FacetGrid at 0x24007ee9750>
```



In [39]:

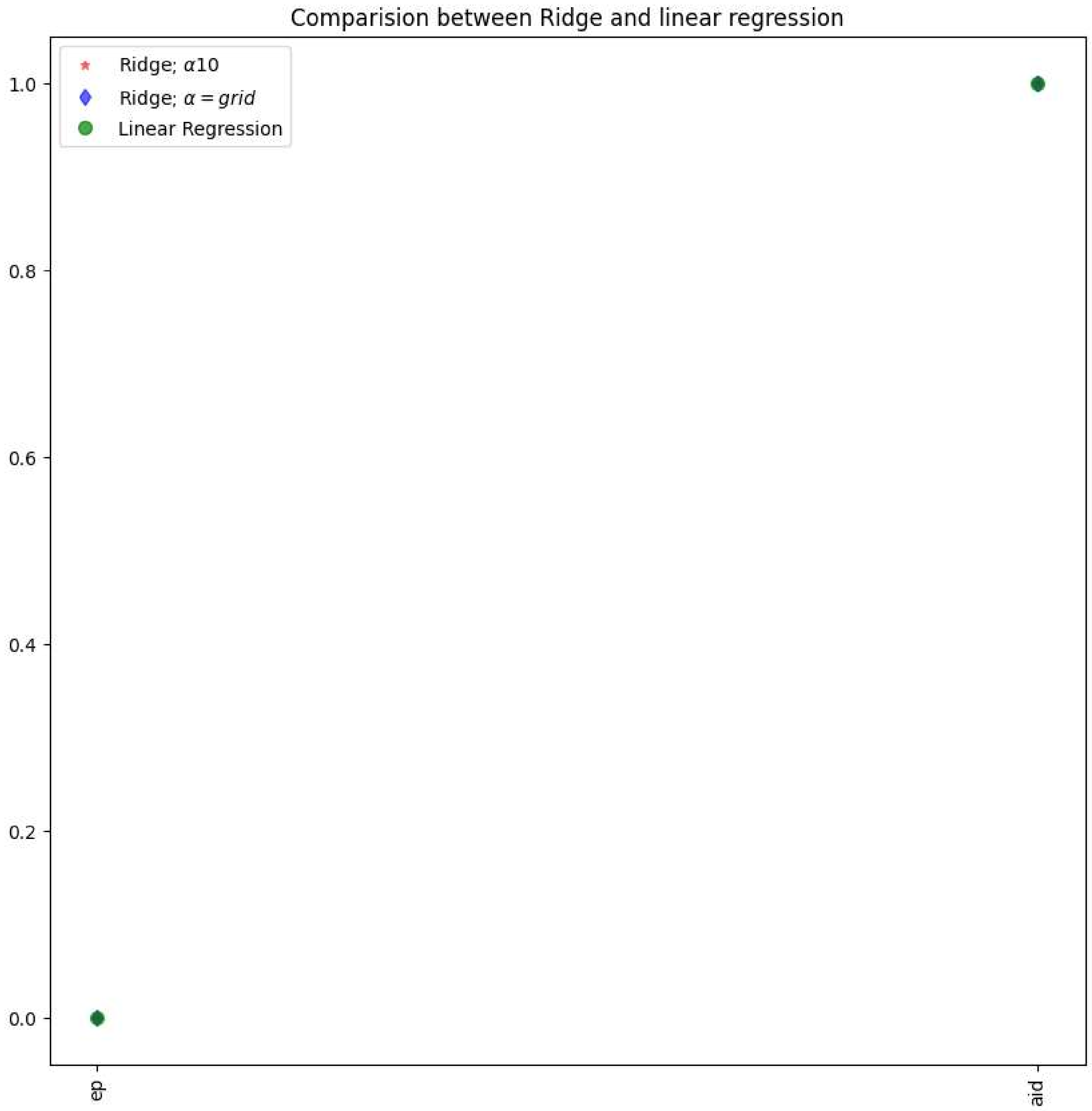
```
df100.fillna(method='ffill',inplace=True)
X=np.array(df100['ep']).reshape(-1,1)
y=np.array(df100['aid']).reshape(-1,1)
df100.dropna(inplace=True)
X_train,x_test,y_train,y_test=train_test_split(X,y,test_size=0.25)
regr=LinearRegression()
regr.fit(X_train,y_train)
print(regr.score(x_test,y_test))
print("Regression: ",regr.score(x_test,y_test))
y_pred=regr.predict(x_test)
plt.scatter(x_test,y_test,color='g')
plt.plot(x_test,y_pred,color='k')
plt.show()
```

```
0.08906920769617288
Regression:  0.08906920769617288
```



In [40]:

```
plt.figure(figsize=(10,10))
plt.plot(feature,ridge.coef_,alpha=0.5,marker='*',markersize=5,linestyle='None',color='red',label=r'Ridge; $\alpha10$')
plt.plot(feature,ridge.coef_,alpha=0.6,marker='d',markersize=6,linestyle='None',color='blue',label=r'Ridge; $\alpha = g
plt.plot(feature,lr.coef_,alpha=0.7,marker='o',markersize=7,color='green',linestyle='None',label='Linear Regression')
plt.xticks(rotation=90)
plt.title("Comparision between Ridge and linear regression")
plt.legend()
plt.show()
```



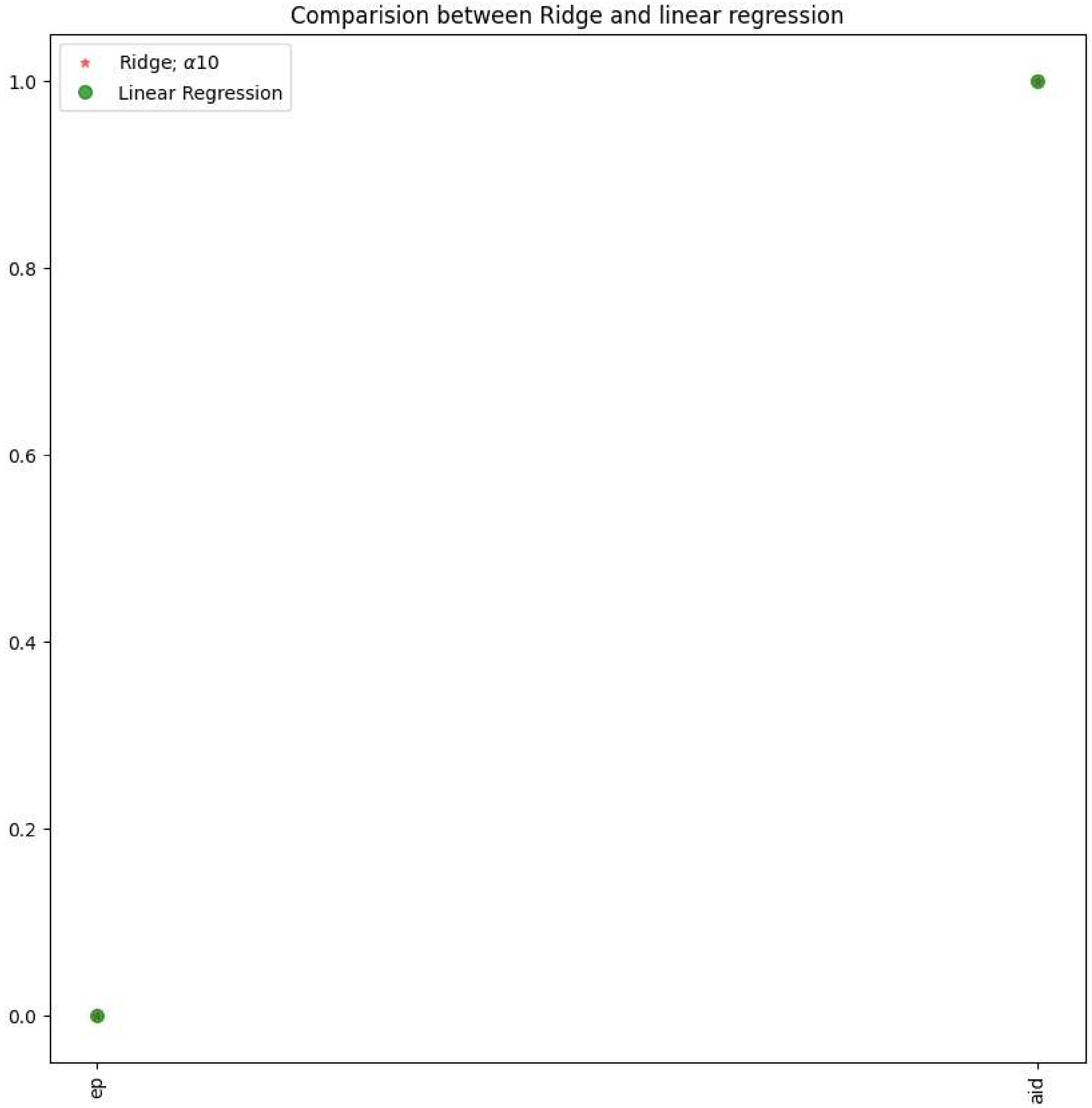
In [41]:

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
feature=d.columns[0:3]
target=d.columns[-1]
x=d[feature].values
y=d[target].values
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.25)
lr = LinearRegression()
lr.fit(x_train,y_train)
print(lr.score(x_test,y_test))
print(lr.score(x_train,y_train))
```

```
1.0
1.0
```

In [42]:

```
plt.figure(figsize=(10,10))
plt.plot(feature,ridge.coef_,alpha=0.5,marker='*',markersize=5,linestyle='None',color='red',label=r'Ridge; $\alpha10$')
plt.plot(feature,lr.coef_,alpha=0.7,marker='o',markersize=7,color='green',linestyle='None',label='Linear Regression')
plt.xticks(rotation=90)
plt.title("Comparision between Ridge and linear regression")
plt.legend()
plt.show()
```



In [43]:

```
from sklearn.linear_model import Ridge,RidgeCV,Lasso,LassoCV
ridge = Ridge(alpha=10)
ridge.fit(x_train,y_train)
train_score_ridge=ridge.score(x_train,y_train)
test_score_ridge=ridge.score(x_test,y_test)
print('\n Ridge method\n')
print('The score of ridge method is {}'.format(train_score_ridge))
print('The score of ridge method is {}'.format(test_score_ridge))
```

```

 Ridge method

The score of ridge method is 1.0
The score of ridge method is 1.0
```

In [44]:

```
lasso = Lasso(alpha=10)
lasso.fit(x_train,y_train)
train_score_lasso=lasso.score(x_train,y_train)
test_score_lasso=lasso.score(x_test,y_test)
print('\n Lasso method\n')
print('The score of lasso method is {}'.format(train_score_lasso))
print('The score of lasso method is {}'.format(test_score_lasso))
```

```

 Lasso method

The score of lasso method is 0.9999999999644131
The score of lasso method is 0.9999999999644129
```

In [45]:

```
lasso_cv= LassoCV(alphas=[0.2,0.03,0.004,0.0001,1,20]).fit(x_train,y_train)
train_score_lasso_cv=lasso_cv.score(x_train,y_train)
test_score_lasso_cv=lasso_cv.score(x_test,y_test)
print('\n LassoCV method\n')
print('The score of Lasso method is {}'.format(train_score_lasso_cv))
print('The score of Lasso method is {}'.format(test_score_lasso_cv))
```

```
 LassoCV method

The score of Lasso method is 1.0
The score of Lasso method is 1.0
```

In [46]:

```
ridge_cv=RidgeCV(alphas=[1,2.3,0.2,0.3,0.4,0.5,0.6]).fit(x_train,y_train)
print("\n RidgeCV Method\n")
print("The score of Ridge method is {}".format(ridge_cv.score(x_train,y_train)))
print("The score of Ridge method is {}".format(ridge_cv.score(x_test,y_test)))
```

```
 RidgeCV Method

The score of Ridge method is 0.9999999998399647
The score of Ridge method is 0.9999999998399633
```