Rain Fall disrict wise dataset

Problem statement: Analysing the monthly rainfall data and comparing with Annual data.

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df=pd.read_csv(r"C:\Users\yoshitha lakshmi\OneDrive\Desktop\python\Rainfall district wise.csv")
df
```

Out[2]:

	STATE_UT_NAME	DISTRICT	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC	ANNUAL	Jan-Feb	Ma M
0	ANDAMAN And NICOBAR ISLANDS	NICOBAR	107.3	57.9	65.2	117.0	358.5	295.5	285.0	271.9	354.8	326.0	315.2	250.9	2805.2	165.2	54(
1	ANDAMAN And NICOBAR ISLANDS	SOUTH ANDAMAN	43.7	26.0	18.6	90.5	374.4	457.2	421.3	423.1	455.6	301.2	275.8	128.3	3015.7	69.7	48:
2	ANDAMAN And NICOBAR ISLANDS	N & M ANDAMAN	32.7	15.9	8.6	53.4	343.6	503.3	465.4	460.9	454.8	276.1	198.6	100.0	2913.3	48.6	40
3	ARUNACHAL PRADESH	LOHIT	42.2	80.8	176.4	358.5	306.4	447.0	660.1	427.8	313.6	167.1	34.1	29.8	3043.8	123.0	84
4	ARUNACHAL PRADESH	EAST SIANG	33.3	79.5	105.9	216.5	323.0	738.3	990.9	711.2	568.0	206.9	29.5	31.7	4034.7	112.8	64
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
636	KERALA	IDUKKI	13.4	22.1	43.6	150.4	232.6	651.6	788.9	527.3	308.4	343.2	172.9	48.1	3302.5	35.5	42
637	KERALA	KASARGOD	2.3	1.0	8.4	46.9	217.6	999.6	1108.5	636.3	263.1	234.9	84.6	18.4	3621.6	3.3	27
638	KERALA	PATHANAMTHITTA	19.8	45.2	73.9	184.9	294.7	556.9	539.9	352.7	266.2	359.4	213.5	51.3	2958.4	65.0	55
639	KERALA	WAYANAD	4.8	8.3	17.5	83.3	174.6	698.1	1110.4	592.9	230.7	213.1	93.6	25.8	3253.1	13.1	27
640	LAKSHADWEEP	LAKSHADWEEP	20.8	14.7	11.8	48.9	171.7	330.2	287.7	217.5	163.1	157.1	117.7	58.8	1600.0	35.5	23

641 rows × 19 columns

In [3]:

```
df.head()
```

Out[3]:

	STATE_UT_NAME	DISTRICT	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC	ANNUAL	Jan-Feb	Mar-May	Jun-Sep
0	ANDAMAN And NICOBAR ISLANDS	NICOBAR	107.3	57.9	65.2	117.0	358.5	295.5	285.0	271.9	354.8	326.0	315.2	250.9	2805.2	165.2	540.7	1207.2
1	ANDAMAN And NICOBAR ISLANDS	SOUTH ANDAMAN	43.7	26.0	18.6	90.5	374.4	457.2	421.3	423.1	455.6	301.2	275.8	128.3	3015.7	69.7	483.5	1757.2
2	ANDAMAN And NICOBAR ISLANDS	N & M ANDAMAN	32.7	15.9	8.6	53.4	343.6	503.3	465.4	460.9	454.8	276.1	198.6	100.0	2913.3	48.6	405.6	1884.4
3	ARUNACHAL PRADESH	LOHIT	42.2	80.8	176.4	358.5	306.4	447.0	660.1	427.8	313.6	167.1	34.1	29.8	3043.8	123.0	841.3	1848.5
4	ARUNACHAL PRADESH	EAST SIANG	33.3	79.5	105.9	216.5	323.0	738.3	990.9	711.2	568.0	206.9	29.5	31.7	4034.7	112.8	645.4	3008.4

In [4]:

```
df.tail()
```

Out[4]:

	STATE_UT_NAME	DISTRICT	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC	ANNUAL	Jan-Feb	Mar-May
636	KERALA	IDUKKI	13.4	22.1	43.6	150.4	232.6	651.6	788.9	527.3	308.4	343.2	172.9	48.1	3302.5	35.5	426.6
637	KERALA	KASARGOD	2.3	1.0	8.4	46.9	217.6	999.6	1108.5	636.3	263.1	234.9	84.6	18.4	3621.6	3.3	272.9
638	KERALA	PATHANAMTHITTA	19.8	45.2	73.9	184.9	294.7	556.9	539.9	352.7	266.2	359.4	213.5	51.3	2958.4	65.0	553.5
639	KERALA	WAYANAD	4.8	8.3	17.5	83.3	174.6	698.1	1110.4	592.9	230.7	213.1	93.6	25.8	3253.1	13.1	275.4
640	LAKSHADWEEP	LAKSHADWEEP	20.8	14.7	11.8	48.9	171.7	330.2	287.7	217.5	163.1	157.1	117.7	58.8	1600.0	35.5	232.4

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

Out[5]: (641, 19)

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

Out[6]:

	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV
count	641.000000	641.000000	641.000000	641.000000	641.000000	641.000000	641.000000	641.000000	641.000000	641.000000	641.000000
mean	18.355070	20.984399	30.034789	45.543214	81.535101	196.007332	326.033697	291.152262	194.609048	90.446334	34.117473
std	21.082806	27.729596	45.451082	71.556279	111.960390	196.556284	221.364643	152.647325	99.830540	74.990685	59.371274
min	0.000000	0.000000	0.000000	0.000000	0.900000	3.800000	11.600000	14.100000	8.600000	3.100000	1.200000
25%	6.900000	7.000000	7.000000	5.000000	12.100000	68.800000	206.400000	194.600000	128.800000	34.300000	6.600000
50%	13.300000	12.300000	12.700000	15.100000	33.900000	131.900000	293.700000	284.800000	181.300000	62.600000	12.900000
75%	19.200000	24.100000	33.200000	48.300000	91.900000	226.600000	374.800000	358.100000	234.100000	130.200000	32.300000
max	144.500000	229.600000	367.900000	554.400000	733.700000	1476.200000	1820.900000	1522.100000	826.300000	517.700000	475.100000

In [7]:
```
df.info()
```

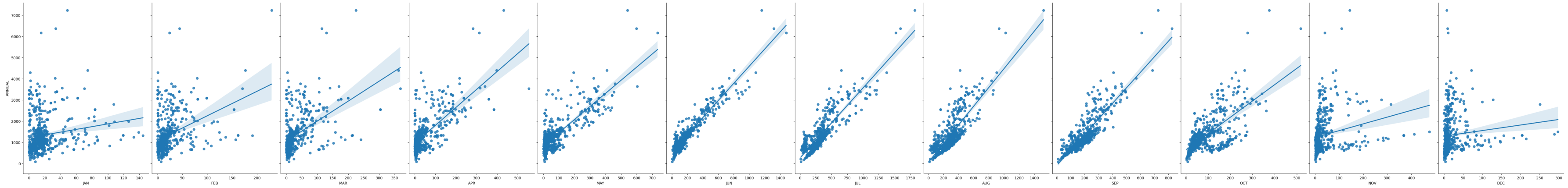
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 641 entries, 0 to 640
Data columns (total 19 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   STATE_UT_NAME  641 non-null    object 
 1   DISTRICT       641 non-null    object 
 2   JAN            641 non-null    float64
 3   FEB            641 non-null    float64
 4   MAR            641 non-null    float64
 5   APR            641 non-null    float64
 6   MAY            641 non-null    float64
 7   JUN            641 non-null    float64
 8   JUL            641 non-null    float64
 9   AUG            641 non-null    float64
 10  SEP            641 non-null    float64
 11  OCT            641 non-null    float64
 12  NOV            641 non-null    float64
 13  DEC            641 non-null    float64
 14  ANNUAL         641 non-null    float64
 15  Jan-Feb        641 non-null    float64
 16  Mar-May        641 non-null    float64
 17  Jun-Sep        641 non-null    float64
 18  Oct-Dec        641 non-null    float64
dtypes: float64(17), object(2)
memory usage: 95.3+ KB
```

In [15]:
```
features=df.columns[2:13]
```

In [16]:
```
target=df.columns[14]
```

In [17]:
```
FEB','MAR','APR','MAY','JUN','JUL','AUG','SEP','OCT','NOV','DEC'],y_vars=['ANNUAL'],height=7,aspect=0.7,kind='reg')
```

Out[17]: <seaborn.axisgrid.PairGrid at 0x1f097f97f40>



In [18]:
```
x=np.array(df[features])
y=np.array(df[target])
```

In [19]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
x_train,x_test,y_train,y_test=train_test_split(x,y,train_size=0.25)
lm=LinearRegression()
lm.fit(x_train,y_train)
print(lm.score(x_train,y_train))
```

```
0.9998981063634227
```

In [20]:
```
coeff_df=pd.DataFrame(lm.coef_)
coeff_df
```

Out[20]:

	0
0	1.547951
1	0.594910
2	1.233032
3	1.011594
4	0.962484
5	1.005697
6	1.001794
7	0.978258
8	1.065904
9	0.795722
10	1.644514

In [36]:
```
predictions=lm.predict(x_test)
```

In [25]:
```
from sklearn import metrics
print('MAE:',metrics.mean_absolute_error(y_test,predictions))
print('MSE:',metrics.mean_squared_error(y_test,predictions))
print('RMSE:',np.sqrt(metrics.mean_squared_error(y_test,predictions)))
```

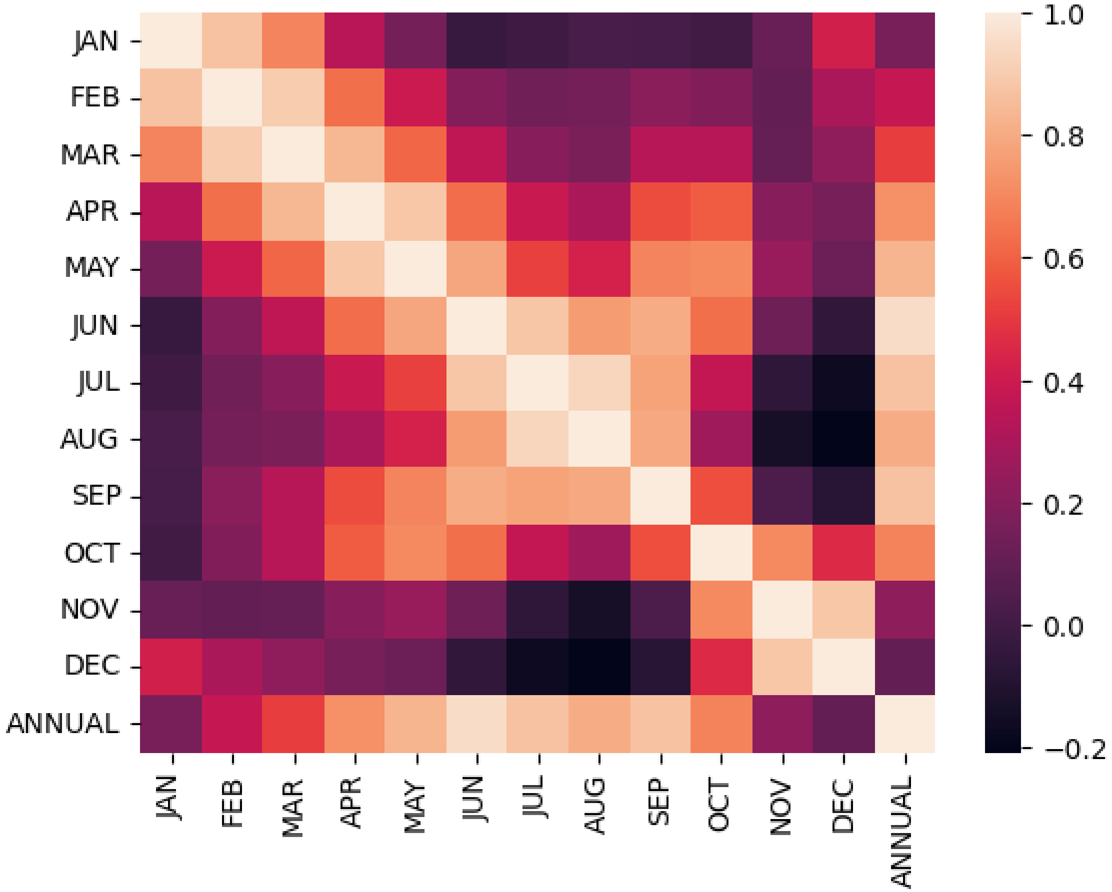
```
MAE: 5.915817136311949
MSE: 75.99669382142075
RMSE: 8.717608262672782
```

Exploratory Data Analysis

In [27]:
```
raindf=df[['JAN','FEB','MAR','APR','MAY','JUN','JUL','AUG','SEP','OCT','NOV','DEC','ANNUAL']]
```

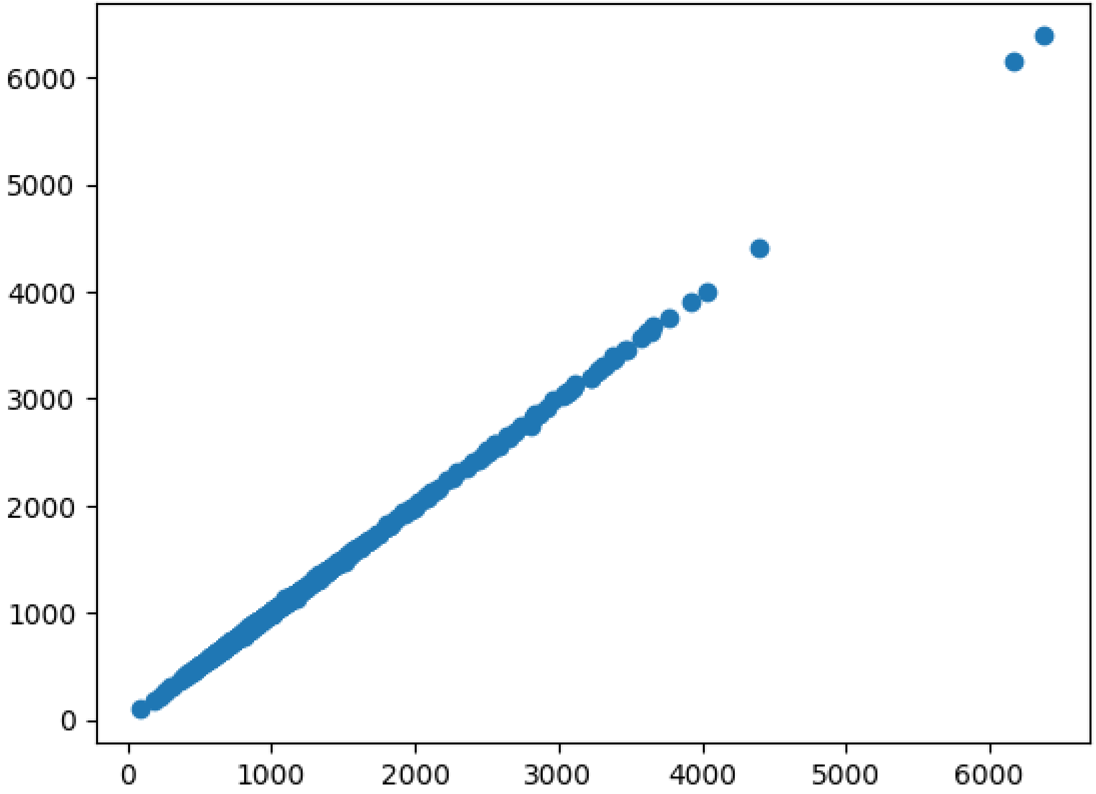
In [28]:
```
sns.heatmap(raindf.corr())
```

Out[28]: <Axes: >

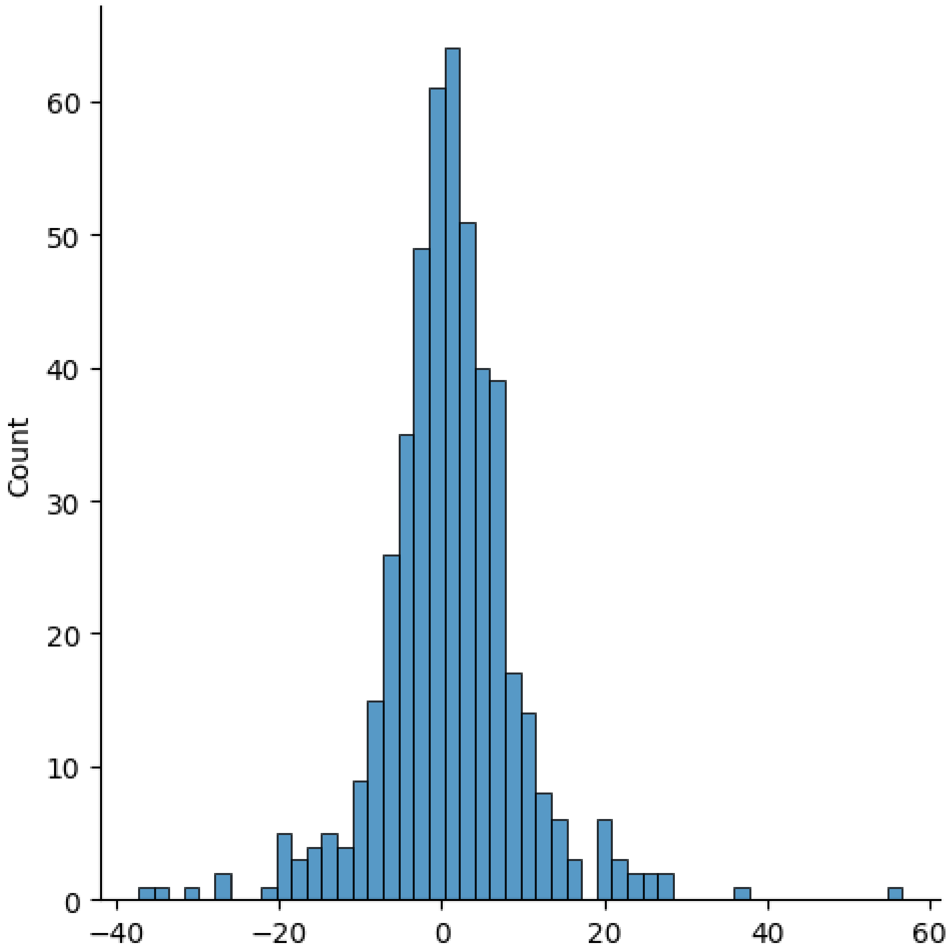


In [39]:
```
predictions=lm.predict(x_test)
plt.scatter(y_test,predictions)
```

Out[39]: <matplotlib.collections.PathCollection at 0x1f0b601a620>

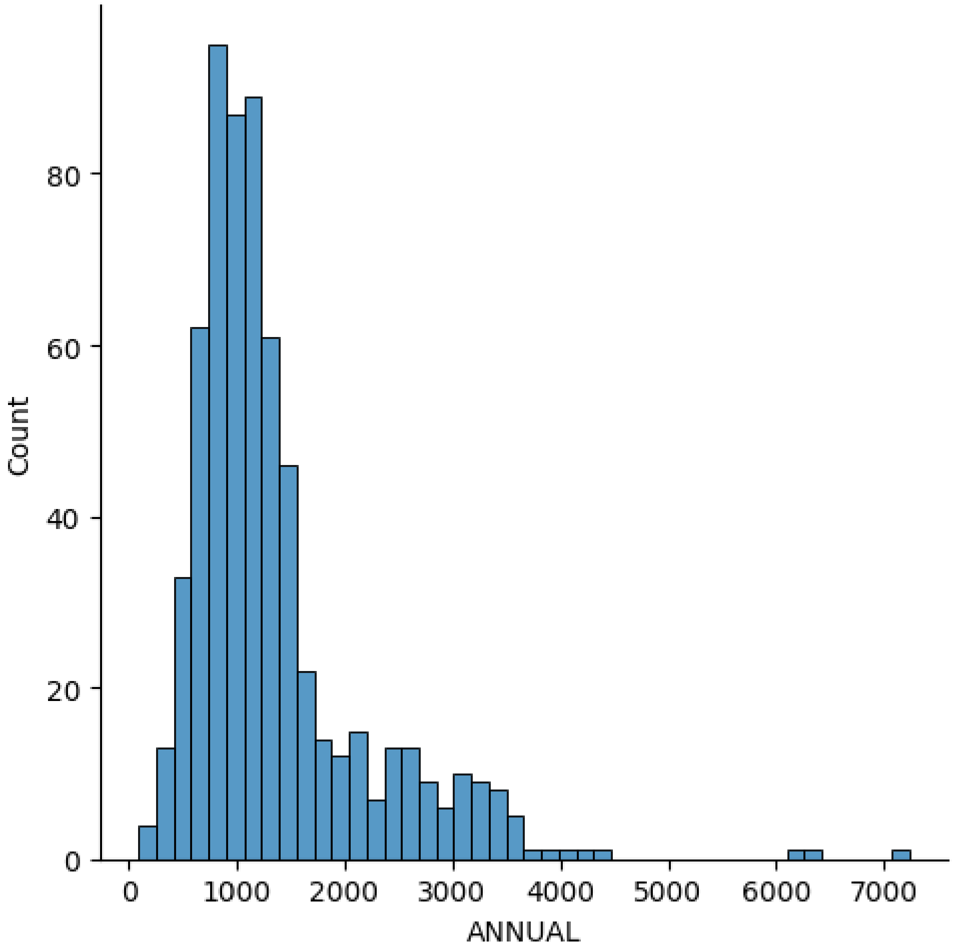


In [43]:
```
sns.displot((y_test-predictions),bins=50);
```



In [44]:
```
sns.displot(df['ANNUAL'])
```

Out[44]: <seaborn.axisgrid.FacetGrid at 0x1f0b5da4c10>

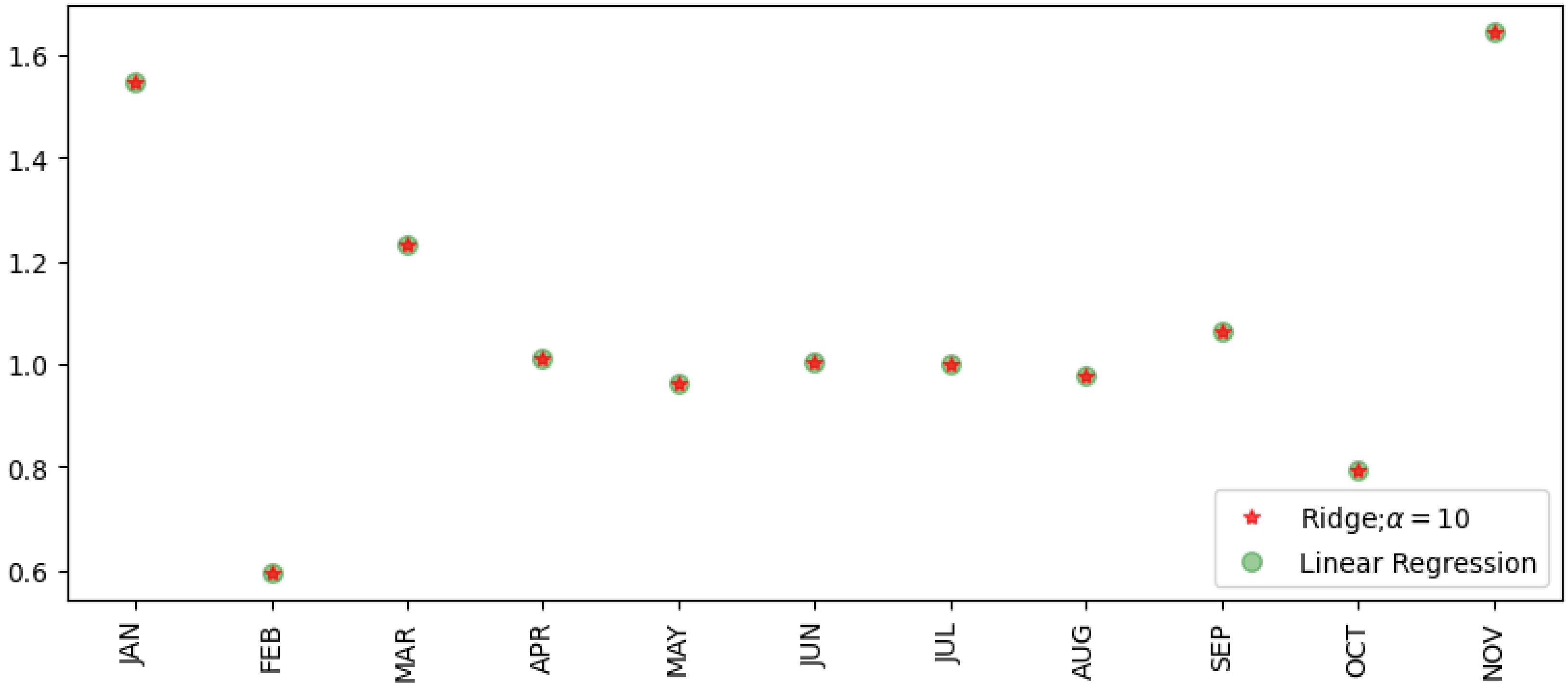


Ridge Regression Model

In [46]:
```
from sklearn.linear_model import Ridge,Lasso,RidgeCV
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('Train score for ridge model is {}'.format(train_score_ridge))
print('Test score for ridge model is {}'.format(test_score_ridge))
```

```
Ridge Model

Train score for ridge model is 0.9998981062781498
Test score for ridge model is 0.9998869925352645
```

In [52]:
```
plt.figure(figsize = (10,4))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=6,color='red',label=r'Ridge;$\
plt.plot(features,lm.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color='green',label='Linear Regres
plt.xticks(rotation=90)
plt.legend()
plt.show()
```

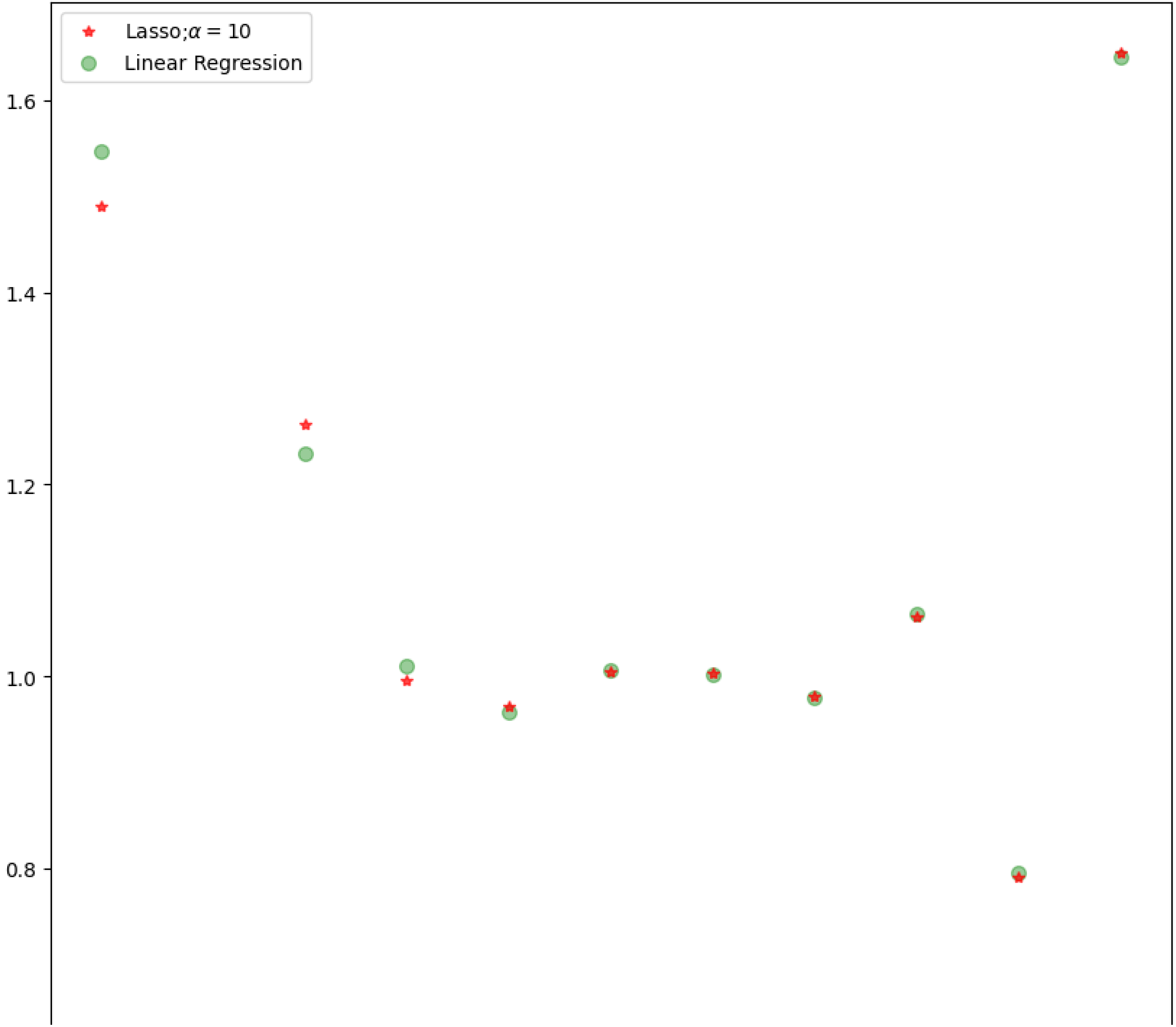


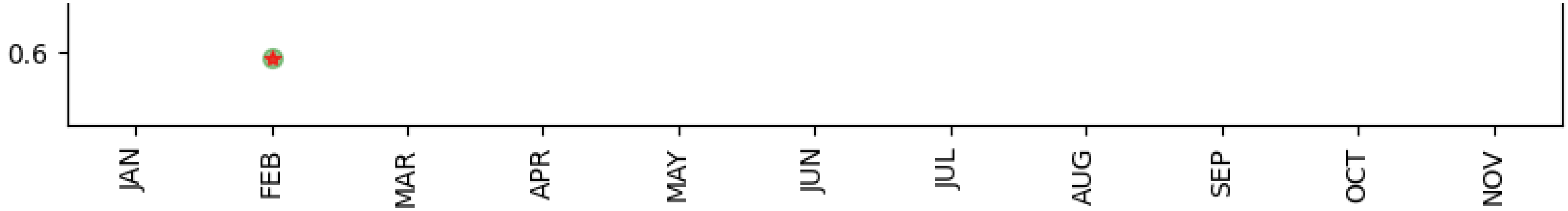
In [56]:

```
lassoReg = Lasso(alpha=10)
lassoReg.fit(x_train,y_train)
train_score_lasso = lassoReg.score(x_train,y_train)
test_score_lasso = lassoReg.score(x_test,y_test)

print('\nRidge Model\n')
print('Train score for lasso model is {}'.format(train_score_lasso))
print('Test score for lasso model is{}'.format(test_score_lasso))
```

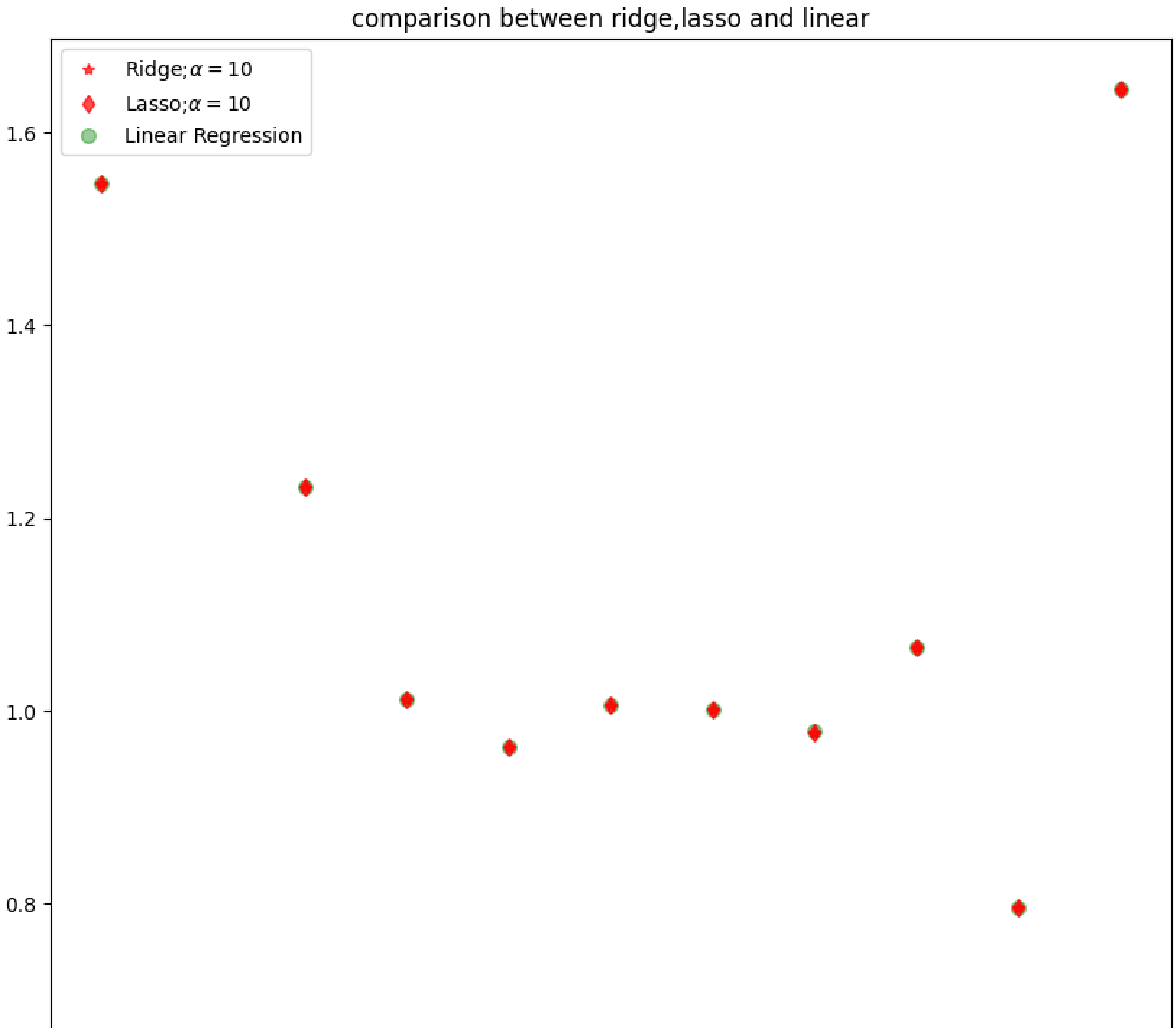
```
Ridge Model

Train score for lasso model is 0.9998976095395863
Test score for lasso model is0.999882011015853
```

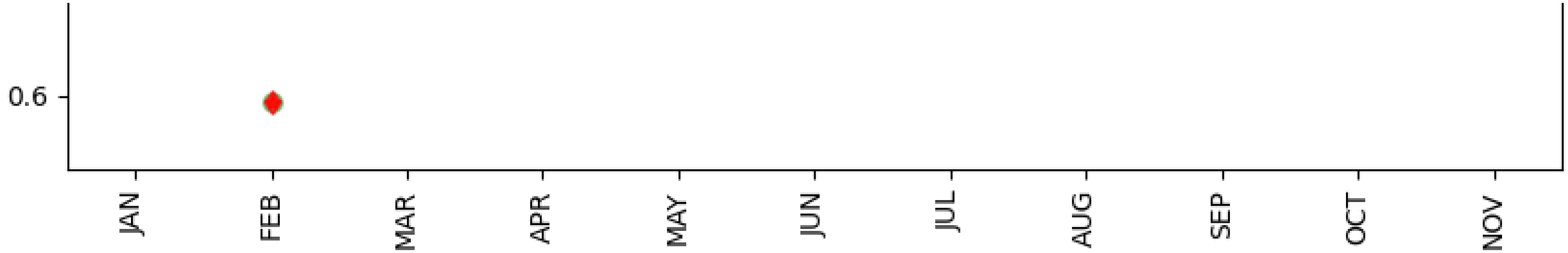
In [57]:
```
plt.figure(figsize = (10,10))
plt.plot(features,lassoReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=6,color='red',label=r'Lasso;$\
plt.plot(features,lm.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color='green',label='Linear Regres
plt.xticks(rotation=90)
plt.legend()
plt.show()
```


Lasso;α = 10
Linear Regression
1.6
1.4
1.2
1.0
0.8


0.6
JAN
FEB
MAR
APR
MAY
JUN
JUL
AUG
SEP
OCT
NOV

In [58]:
```
# Comparison between Ridge,Lasso and RidgeCV
plt.figure(figsize=(10,10))

plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=6,color='red',label=r'Ridge;$\
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='d',markersize=6,color='red',label=r'Lasso;$\
plt.plot(features,lm.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color='green',label='Linear Regres
plt.xticks(rotation=90)
plt.title('comparison between ridge,lasso and linear')
plt.legend()
plt.show()
```


comparison between ridge,lasso and linear
Ridge;$\alpha = 10$
Lasso;$\alpha = 10$
Linear Regression
1.6
1.4
1.2
1.0
0.8



In [62]:

```
# Linear CV model using Ridge
from sklearn.linear_model import RidgeCV
ridge_CV=RidgeCV(alphas=[0.0001,0.01,0.001,0.1]).fit(x_train,y_train)
print("The train score for ridge model is {}".format(ridge_CV.score(x_train,y_train)))
print("The test score for ridge model is {}".format(ridge_CV.score(x_test,y_test)))
```

```
The train score for ridge model is 0.9998981036461831
The test score for ridge model is 0.9998869496121741
```

In [63]:

```
# Linear CV model using Losso
from sklearn.linear_model import LassoCV
lasso_CV=LassoCV(alphas=[0.0001,0.01,0.001,0.1]).fit(x_train,y_train)
print("The train score for lasso model is {}".format(lasso_CV.score(x_train,y_train)))
print("The test score for lasso model is {}".format(lasso_CV.score(x_test,y_test)))
```

```
The train score for lasso model is 0.9998980995382691
The test score for lasso model is 0.9998872120915944
```

In [60]:

```
from sklearn.linear_model import ElasticNet
regr = ElasticNet()
regr.fit(x,y)
print(regr.coef_)
print(regr.intercept_)
y_pred_elastic = regr.predict(x_train)
mean_squared_error = np.mean((y_pred_elastic-y_train)**2)
print('Mean squared error on test set',mean_squared_error)
```

```
[1.59402227 0.77496303 1.07666284 1.00645481 0.98815802 1.00582305
 0.99931781 0.97704429 1.06450887 0.81537098 1.62088766]
-0.9346307788046033
Mean squared error on test set 88.18210015402676
```

In [65]:

```
regr.score(x_train,y_train)
```

Out[65]:

```
0.999888082161808
```

Conclusion

The Accuracy for LinearRegression model is 0.99 i.e, the data is fitted well where the accuracy for Ridge and Lasso and Elastic Net is also same.

Rainfall 1901-2015 dataset

Problem statement: Analysed the annual rainfall data for different years.

In [66]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [67]:
```
df=pd.read_csv(r"C:\Users\yoshitha lakshmi\OneDrive\Desktop\python\Rainfall 1901-2015.csv")
df
```

Out[67]:

	SUBDIVISION	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC	ANNUAL	Jan-Feb	Mar-May	Jun-Sep	
0	ANDAMAN & NICOBAR ISLANDS	1901	49.2	87.1	29.2	2.3	528.8	517.5	365.1	481.1	332.6	388.5	558.2	33.6	3373.2	136.3	560.3	1696.3	98
1	ANDAMAN & NICOBAR ISLANDS	1902	0.0	159.8	12.2	0.0	446.1	537.1	228.9	753.7	666.2	197.2	359.0	160.5	3520.7	159.8	458.3	2185.9	71
2	ANDAMAN & NICOBAR ISLANDS	1903	12.7	144.0	0.0	1.0	235.1	479.9	728.4	326.7	339.0	181.2	284.4	225.0	2957.4	156.7	236.1	1874.0	69
3	ANDAMAN & NICOBAR ISLANDS	1904	9.4	14.7	0.0	202.4	304.5	495.1	502.0	160.1	820.4	222.2	308.7	40.1	3079.6	24.1	506.9	1977.6	57
4	ANDAMAN & NICOBAR ISLANDS	1905	1.3	0.0	3.3	26.9	279.5	628.7	368.7	330.5	297.0	260.7	25.4	344.7	2566.7	1.3	309.7	1624.9	63
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
4111	LAKSHADWEEP	2011	5.1	2.8	3.1	85.9	107.2	153.6	350.2	254.0	255.2	117.4	184.3	14.9	1533.7	7.9	196.2	1013.0	31
4112	LAKSHADWEEP	2012	19.2	0.1	1.6	76.8	21.2	327.0	231.5	381.2	179.8	145.9	12.4	8.8	1405.5	19.3	99.6	1119.5	16
4113	LAKSHADWEEP	2013	26.2	34.4	37.5	5.3	88.3	426.2	296.4	154.4	180.0	72.8	78.1	26.7	1426.3	60.6	131.1	1057.0	17
4114	LAKSHADWEEP	2014	53.2	16.1	4.4	14.9	57.4	244.1	116.1	466.1	132.2	169.2	59.0	62.3	1395.0	69.3	76.7	958.5	29
4115	LAKSHADWEEP	2015	2.2	0.5	3.7	87.1	133.1	296.6	257.5	146.4	160.4	165.4	231.0	159.0	1642.9	2.7	223.9	860.9	55

4116 rows × 19 columns

In [68]:
```
df.head()
```

Out[68]:

	SUBDIVISION	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC	ANNUAL	Jan-Feb	Mar-May	Jun-Sep	Oct-Dec
0	ANDAMAN & NICOBAR ISLANDS	1901	49.2	87.1	29.2	2.3	528.8	517.5	365.1	481.1	332.6	388.5	558.2	33.6	3373.2	136.3	560.3	1696.3	980.3
1	ANDAMAN & NICOBAR ISLANDS	1902	0.0	159.8	12.2	0.0	446.1	537.1	228.9	753.7	666.2	197.2	359.0	160.5	3520.7	159.8	458.3	2185.9	716.7
2	ANDAMAN & NICOBAR ISLANDS	1903	12.7	144.0	0.0	1.0	235.1	479.9	728.4	326.7	339.0	181.2	284.4	225.0	2957.4	156.7	236.1	1874.0	690.6
3	ANDAMAN & NICOBAR ISLANDS	1904	9.4	14.7	0.0	202.4	304.5	495.1	502.0	160.1	820.4	222.2	308.7	40.1	3079.6	24.1	506.9	1977.6	571.0
4	ANDAMAN & NICOBAR ISLANDS	1905	1.3	0.0	3.3	26.9	279.5	628.7	368.7	330.5	297.0	260.7	25.4	344.7	2566.7	1.3	309.7	1624.9	630.8

In [69]:
```
df.tail()
```

Out[69]:

	SUBDIVISION	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC	ANNUAL	Jan-Feb	Mar-May	Jun-Sep	Oct-Dec
4111	LAKSHADWEEP	2011	5.1	2.8	3.1	85.9	107.2	153.6	350.2	254.0	255.2	117.4	184.3	14.9	1533.7	7.9	196.2	1013.0	316.6
4112	LAKSHADWEEP	2012	19.2	0.1	1.6	76.8	21.2	327.0	231.5	381.2	179.8	145.9	12.4	8.8	1405.5	19.3	99.6	1119.5	167.1
4113	LAKSHADWEEP	2013	26.2	34.4	37.5	5.3	88.3	426.2	296.4	154.4	180.0	72.8	78.1	26.7	1426.3	60.6	131.1	1057.0	177.6
4114	LAKSHADWEEP	2014	53.2	16.1	4.4	14.9	57.4	244.1	116.1	466.1	132.2	169.2	59.0	62.3	1395.0	69.3	76.7	958.5	290.5
4115	LAKSHADWEEP	2015	2.2	0.5	3.7	87.1	133.1	296.6	257.5	146.4	160.4	165.4	231.0	159.0	1642.9	2.7	223.9	860.9	555.4

In [70]:
```
df.describe()
```

Out[70]:

	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	
count	4116.000000	4112.000000	4113.000000	4110.000000	4112.000000	4113.000000	4111.000000	4109.000000	4112.000000	4110.000000	4109.
mean	1958.218659	18.957320	21.805325	27.359197	43.127432	85.745417	230.234444	347.214334	290.263497	197.361922	95.
std	33.140898	33.585371	35.909488	46.959424	67.831168	123.234904	234.710758	269.539667	188.770477	135.408345	99.
min	1901.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.400000	0.000000	0.000000	0.100000	0.
25%	1930.000000	0.600000	0.600000	1.000000	3.000000	8.600000	70.350000	175.600000	155.975000	100.525000	14.
50%	1958.000000	6.000000	6.700000	7.800000	15.700000	36.600000	138.700000	284.800000	259.400000	173.900000	65.
75%	1987.000000	22.200000	26.800000	31.300000	49.950000	97.200000	305.150000	418.400000	377.800000	265.800000	148.
max	2015.000000	583.700000	403.500000	605.600000	595.100000	1168.600000	1609.900000	2362.800000	1664.600000	1222.000000	948.

In [71]:
```
df.shape
```

Out[71]: (4116, 19)

In [72]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4116 entries, 0 to 4115
Data columns (total 19 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   SUBDIVISION  4116 non-null   object 
 1   YEAR         4116 non-null   int64  
 2   JAN          4112 non-null   float64
 3   FEB          4113 non-null   float64
 4   MAR          4110 non-null   float64
 5   APR          4112 non-null   float64
 6   MAY          4113 non-null   float64
 7   JUN          4111 non-null   float64
 8   JUL          4109 non-null   float64
 9   AUG          4112 non-null   float64
 10  SEP          4110 non-null   float64
 11  OCT          4109 non-null   float64
 12  NOV          4105 non-null   float64
 13  DEC          4106 non-null   float64
 14  ANNUAL       4090 non-null   float64
 15  Jan-Feb      4110 non-null   float64
 16  Mar-May      4107 non-null   float64
 17  Jun-Sep      4106 non-null   float64
 18  Oct-Dec      4103 non-null   float64
dtypes: float64(17), int64(1), object(1)
memory usage: 611.1+ KB
```

In [93]:
```
df.fillna(method='ffill', inplace=True)
```

In [94]:
```
features=df.columns[1:13]
```

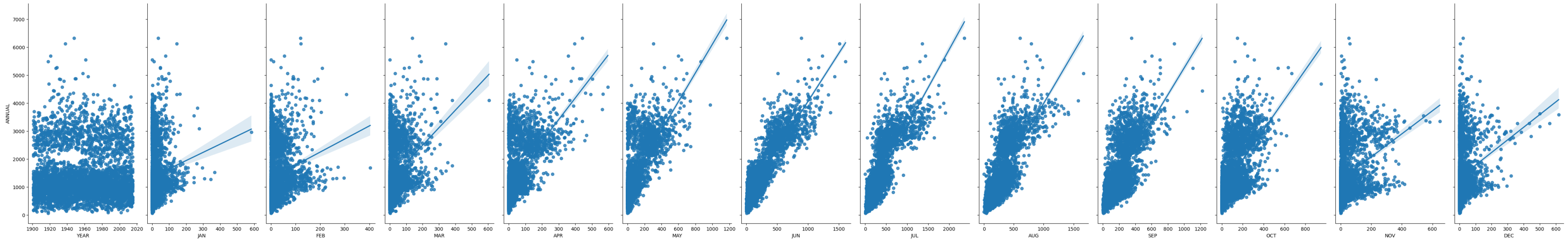
In [95]:
```
target=df.columns[14]
```

In [96]:
```
df.columns
```

Out[96]:
```
Index(['SUBDIVISION', 'YEAR', 'JAN', 'FEB', 'MAR', 'APR', 'MAY', 'JUN', 'JUL',
       'AUG', 'SEP', 'OCT', 'NOV', 'DEC', 'ANNUAL', 'Jan-Feb', 'Mar-May',
       'Jun-Sep', 'Oct-Dec'],
      dtype='object')
```

In [97]:
```
sns.pairplot(df,x_vars=['YEAR', 'JAN', 'FEB', 'MAR', 'APR', 'MAY', 'JUN', 'JUL',
       'AUG', 'SEP', 'OCT', 'NOV', 'DEC'],y_vars=['ANNUAL'],height=7,aspect=0.5,kind='reg')
```

Out[97]:
```
<seaborn.axisgrid.PairGrid at 0x1f0be4e8940>
```



In [98]:
```
x=np.array(df[features])
y=np.array(df[target])
```

In [99]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
x_train,x_test,y_train,y_test=train_test_split(x,y,train_size=0.25)
lm=LinearRegression()
lm.fit(x_train,y_train)
print(lm.score(x_train,y_train))
```

```
0.9916135364757804
```

In [102]:
```
coeff_df=pd.DataFrame(lm.coef_)
coeff_df
```

Out[102]:

	0
0	-0.036257
1	1.094004
2	0.792353
3	1.285093
4	1.018243
5	0.956323
6	0.995912
7	1.017815
8	0.917773
9	1.033492
10	1.061315
11	1.282726

In [103]:
```
print(lm.intercept_)
```

```
82.37961923825105
```

In [104]:
```
predictions=lm.predict(x_test)
```

In [105]:
```
from sklearn import metrics
print('MAE:',metrics.mean_absolute_error(y_test,predictions))
print('MSE:',metrics.mean_squared_error(y_test,predictions))
print('RMSE:',np.sqrt(metrics.mean_squared_error(y_test,predictions)))
```

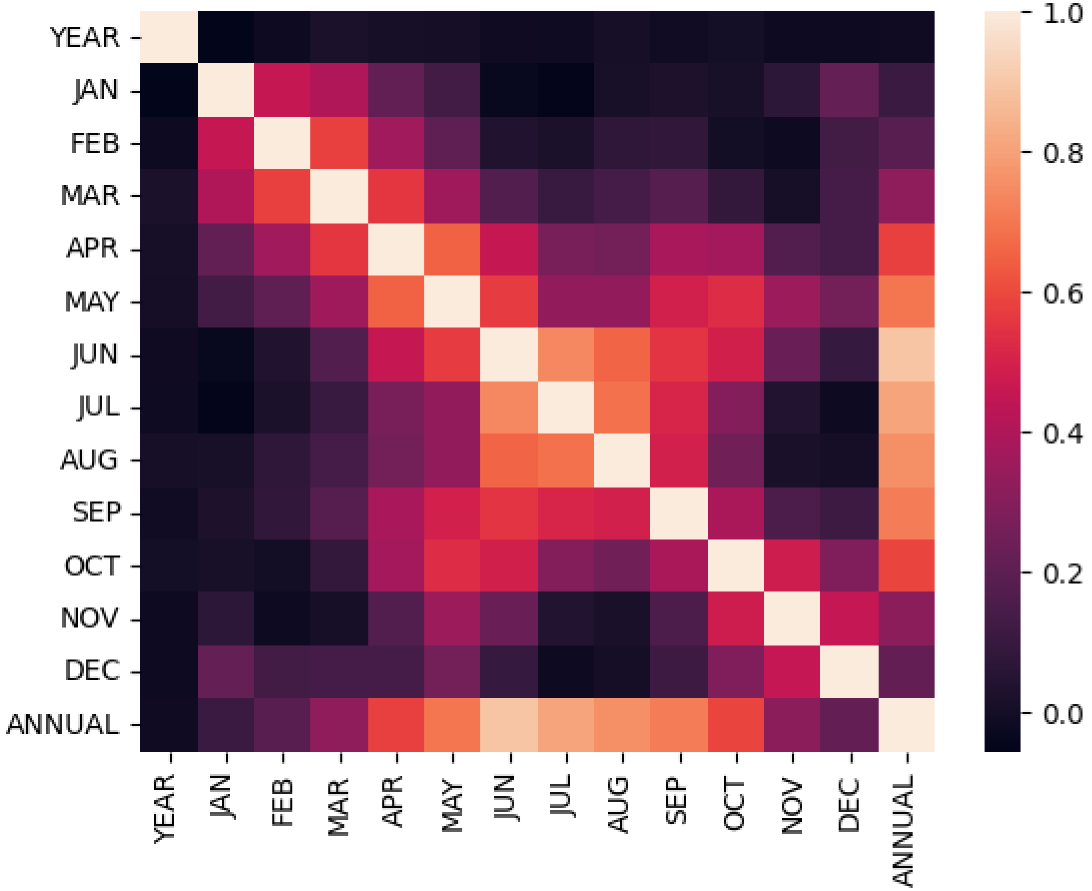
```
MAE: 25.590560026817133
MSE: 3777.721275095668
RMSE: 61.46317007034105
```

Exploratory Data Analysis

In [106]:
```
Raindf=df[['YEAR', 'JAN', 'FEB', 'MAR', 'APR', 'MAY', 'JUN', 'JUL',
       'AUG', 'SEP', 'OCT', 'NOV', 'DEC', 'ANNUAL']]
```

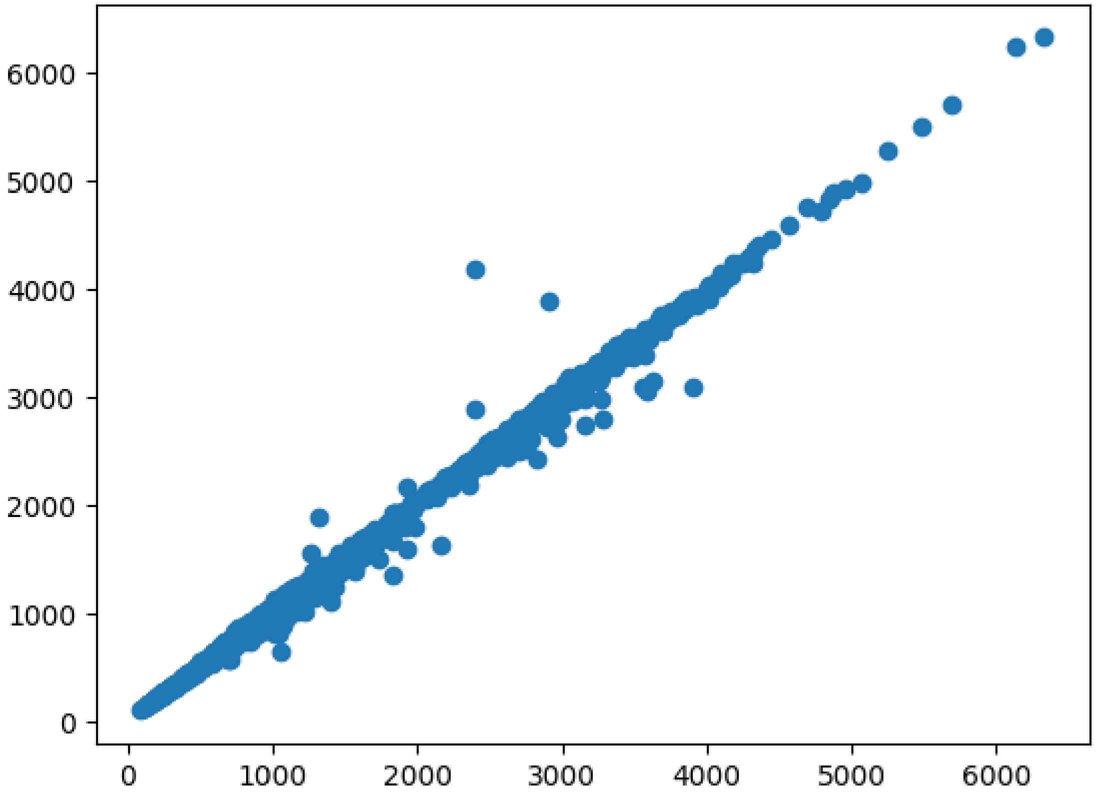
In [107]:
```
sns.heatmap(Raindf.corr())
```

Out[107]: <Axes: >



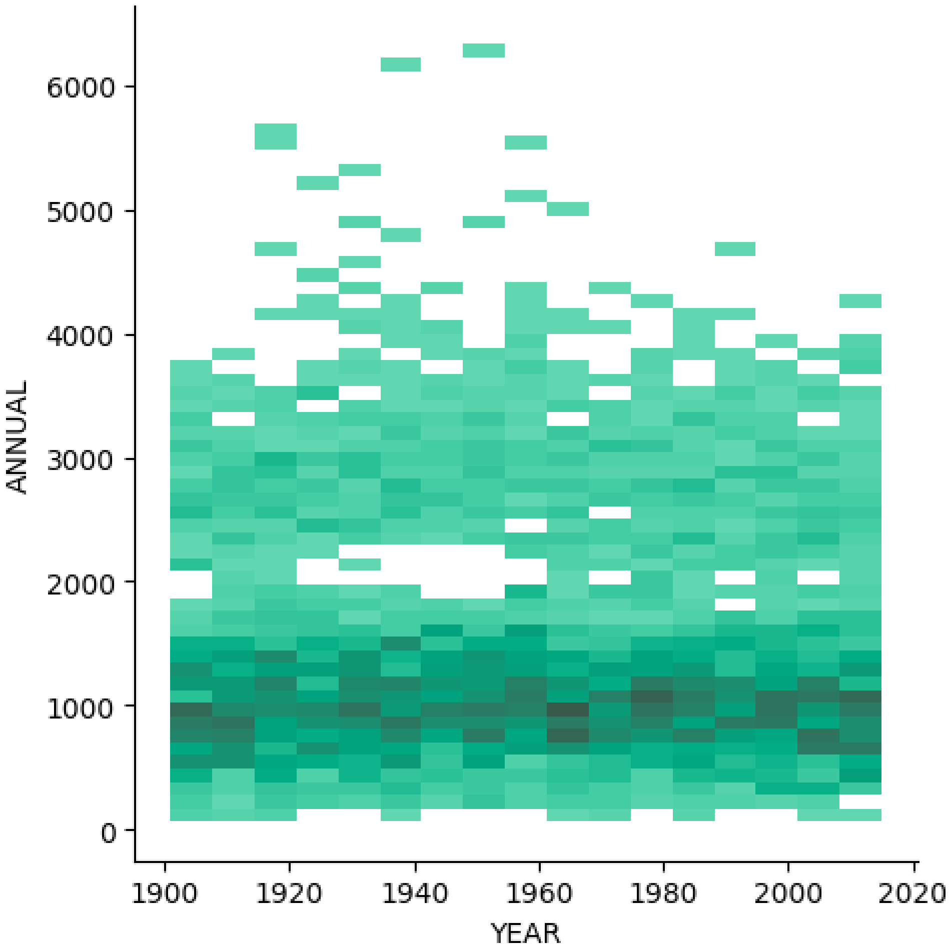
In [108]:
```
predictions=lm.predict(x_test)
plt.scatter(y_test,predictions)

```

Out[108]: <matplotlib.collections.PathCollection at 0x1f094e8e740>



In [119]:

```
sns.displot(x='YEAR',y='ANNUAL',data=df,color='aquamarine')
plt.show()
```

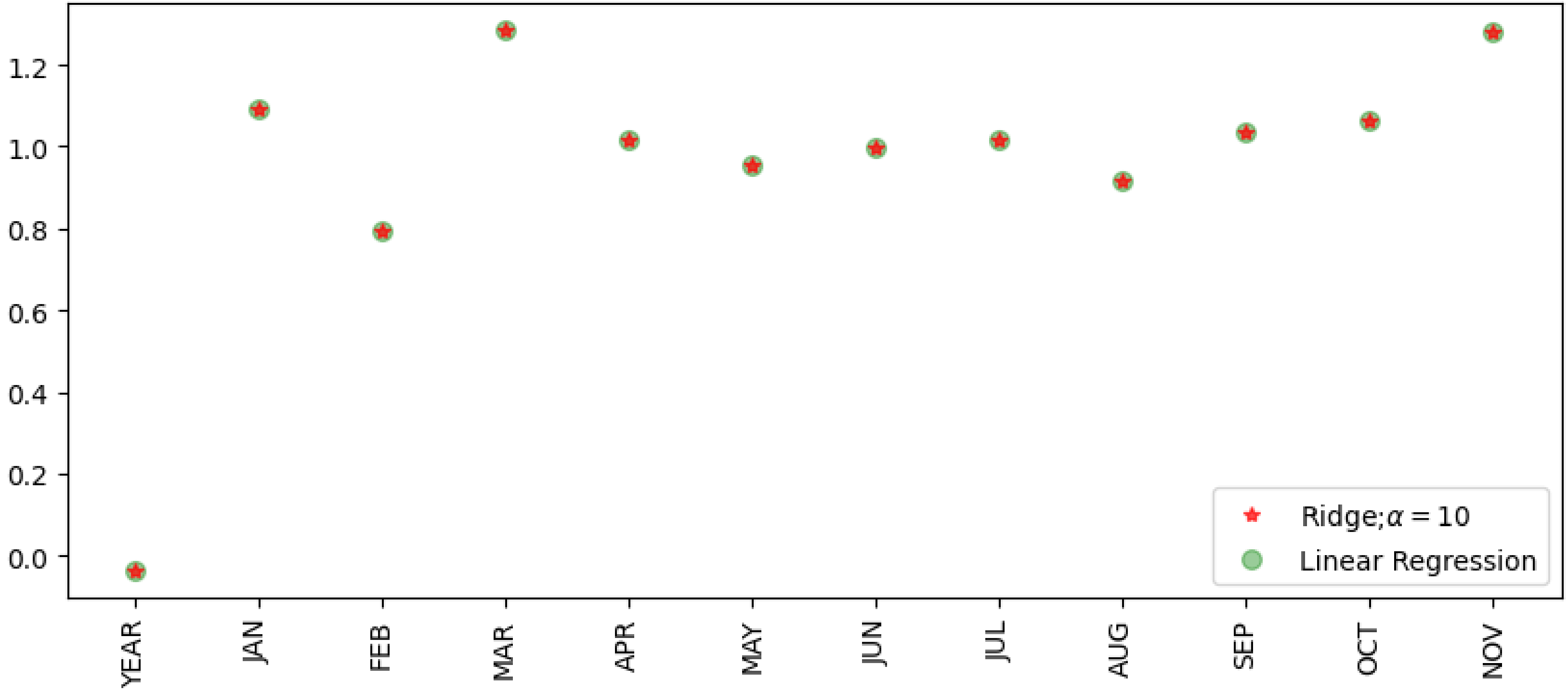


In [120]:

```
from sklearn.linear_model import Ridge,Lasso,RidgeCV
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('Train score for ridge model is {}'.format(train_score_ridge))
print('Test score for ridge model is {}'.format(test_score_ridge))

```

```
Ridge Model

Train score for ridge model is 0.9916135364755825
Test score for ridge model is 0.9954207572738636
```

In [121]:
```
plt.figure(figsize = (10,4))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=6,color='red',label=r'Ridge;$\
plt.plot(features,lm.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color='green',label='Linear Regres
plt.xticks(rotation=90)
plt.legend()
plt.show()
```



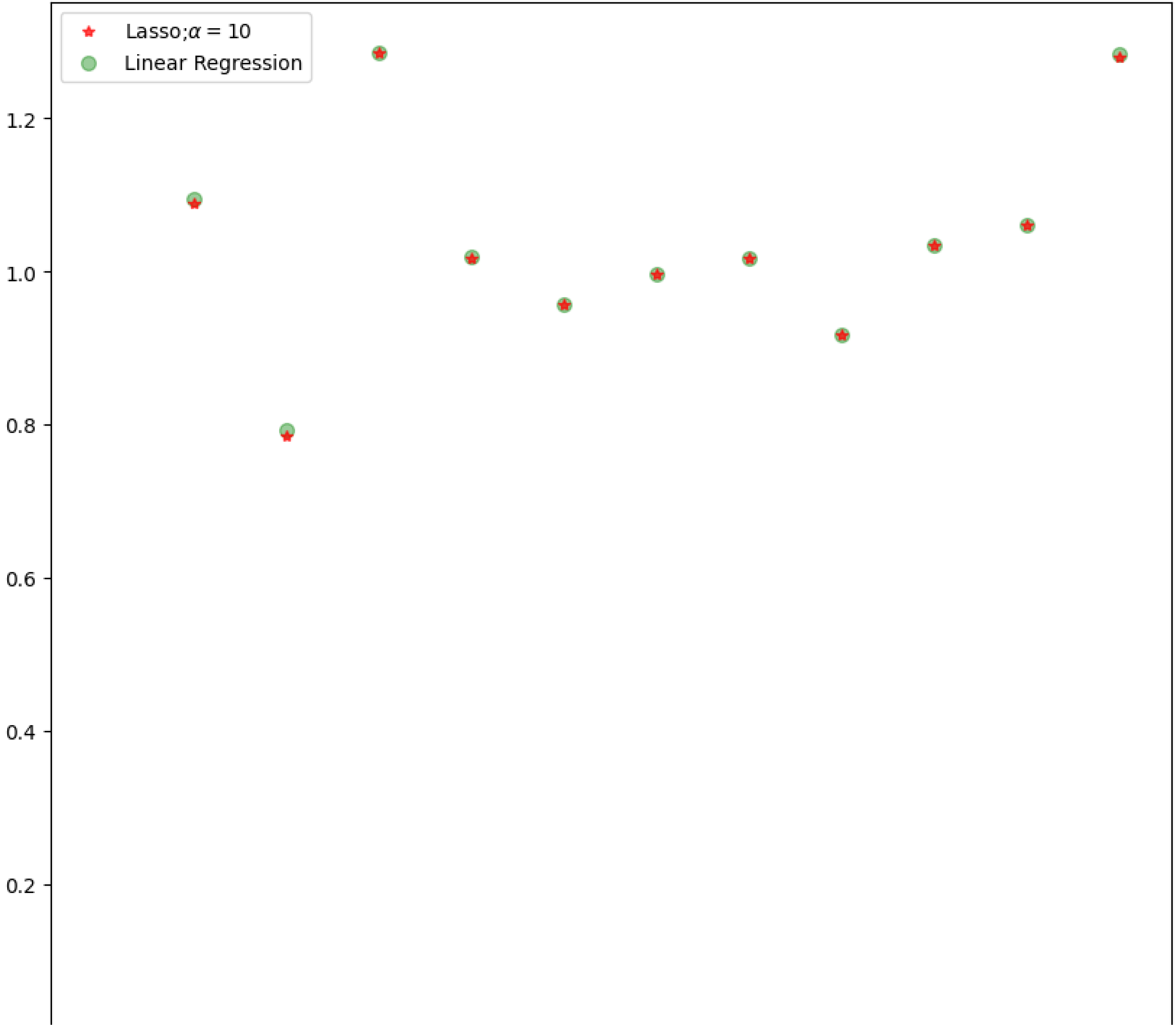
In [122]:

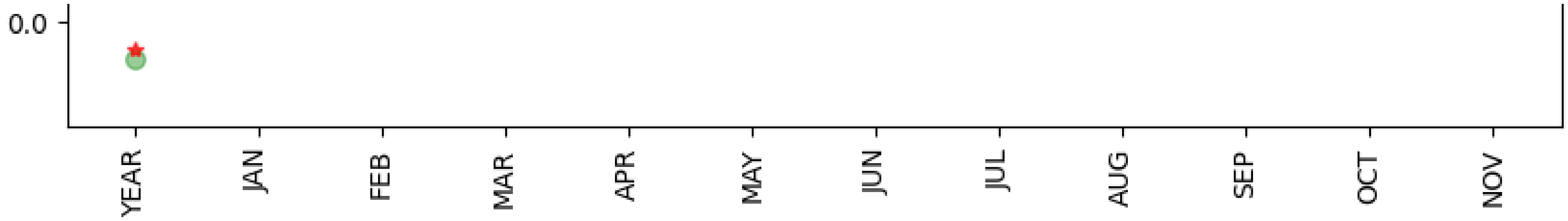
```
lassoReg = Lasso(alpha=10)
lassoReg.fit(x_train,y_train)
train_score_lasso = lassoReg.score(x_train,y_train)
test_score_lasso = lassoReg.score(x_test,y_test)

print('\nRidge Model\n')
print('Train score for lasso model is {}'.format(train_score_lasso))
print('Test score for lasso model is{}'.format(test_score_lasso))
```

```
Ridge Model

Train score for lasso model is 0.9916132450486558
Test score for lasso model is0.9954172471874398
```

In [123]:

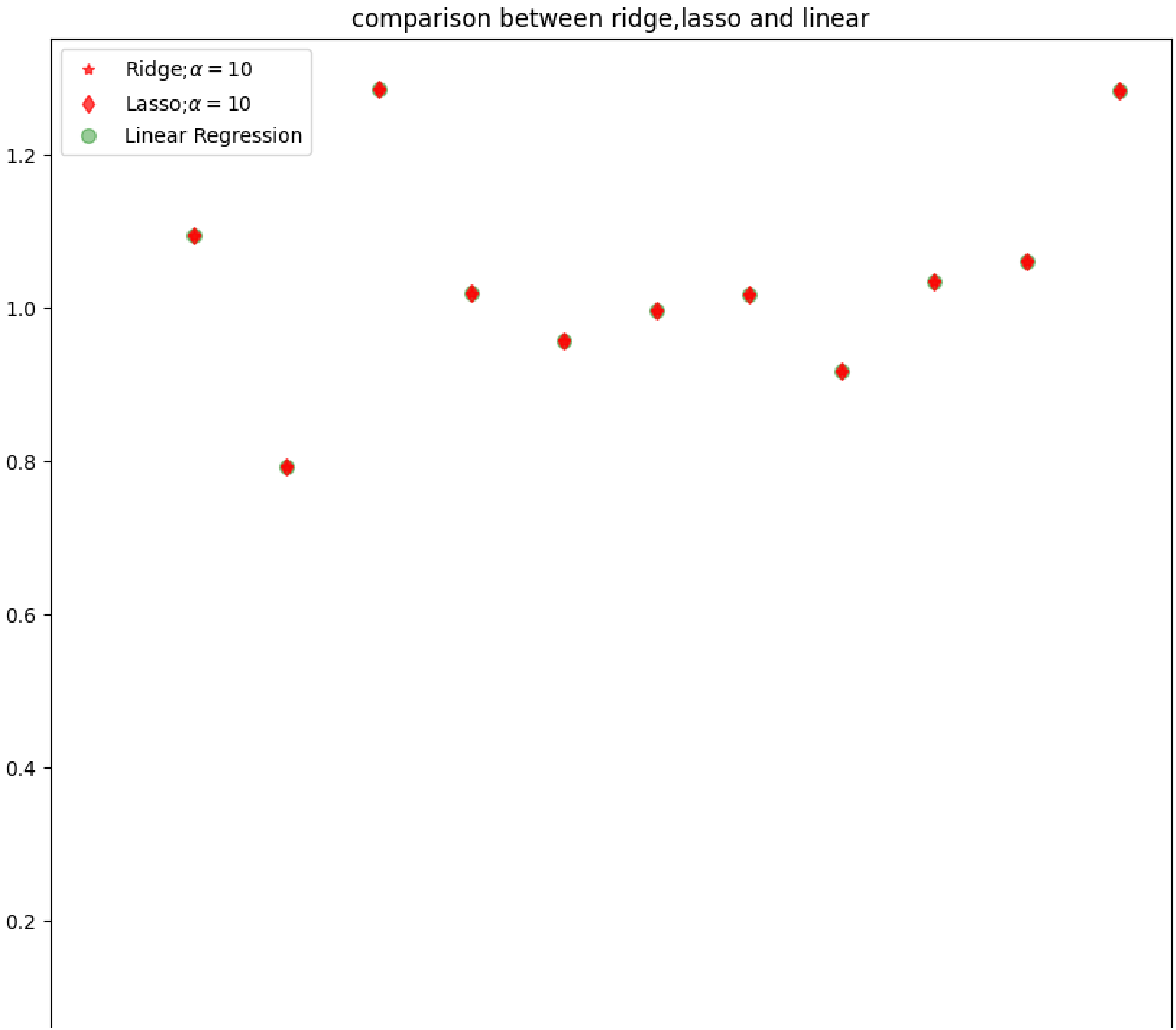
```
plt.figure(figsize = (10,10))
plt.plot(features,lassoReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=6,color='red',label=r'Lasso;$\
plt.plot(features,lm.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color='green',label='Linear Regres
plt.xticks(rotation=90)
plt.legend()
plt.show()
```

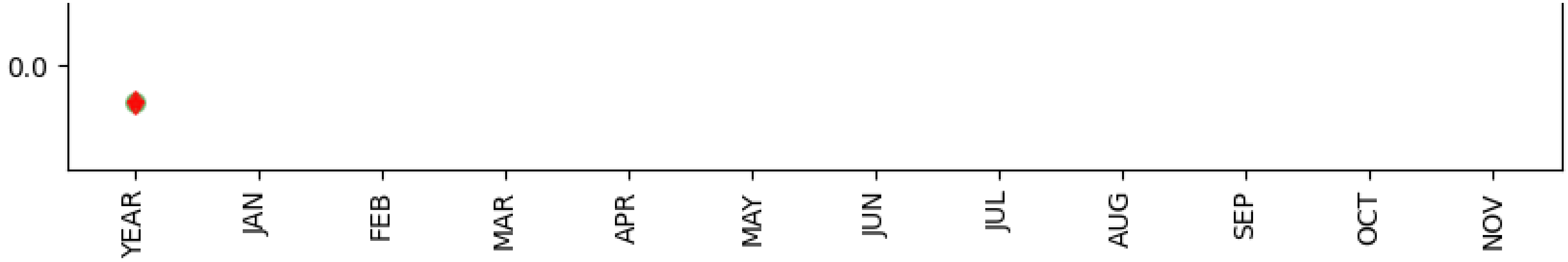

Lasso;α = 10
Linear Regression
1.2
1.0
0.8
0.6
0.4
0.2


0.0
YEAR
JAN
FEB
MAR
APR
MAY
JUN
JUL
AUG
SEP
OCT
NOV

In [124]:

```
plt.figure(figsize=(10,10))

plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=6,color='red',label=r'Ridge;$\
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='d',markersize=6,color='red',label=r'Lasso;$\
plt.plot(features,lm.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color='green',label='Linear Regres
plt.xticks(rotation=90)
plt.title('comparison between ridge,lasso and linear')
plt.legend()
plt.show()
```


comparison between ridge,lasso and linear
Ridge; $\alpha = 10$
Lasso; $\alpha = 10$
Linear Regression
1.2
1.0
0.8
0.6
0.4
0.2



In [125]:

```
# Linear CV model using Ridge
from sklearn.linear_model import RidgeCV
ridge_CV=RidgeCV(alphas=[0.0001,0.01,0.001,0.1]).fit(x_train,y_train)
print("The train score for ridge model is {}".format(ridge_CV.score(x_train,y_train)))
print("The test score for ridge model is {}".format(ridge_CV.score(x_test,y_test)))
```

```
The train score for ridge model is 0.99161353647277774
The test score for ridge model is 0.9954207508330051
```

In [126]:

```
# Linear CV model using Losso
from sklearn.linear_model import LassoCV
lasso_CV=LassoCV(alphas=[0.0001,0.01,0.001,0.1]).fit(x_train,y_train)
print("The train score for lasso model is {}".format(lasso_CV.score(x_train,y_train)))
print("The test score for lasso model is {}".format(lasso_CV.score(x_test,y_test)))
```

```
The train score for lasso model is 0.99161353644460524
The test score for lasso model is 0.9954207194518694
```

In [127]:
```
from sklearn.linear_model import ElasticNet
regr = ElasticNet()
regr.fit(x,y)
print(regr.coef_)
print(regr.intercept_)
y_pred_elastic = regr.predict(x_train)
mean_squared_error = np.mean((y_pred_elastic-y_train)**2)
print('Mean squared error on test set',mean_squared_error)
```

```
[0.00452878 1.21391301 1.00790437 1.04920427 0.96127995 1.00091858
 0.99101193 0.99152702 0.98668101 1.02070562 1.04790013 1.26485708]
-4.503727353915792
Mean squared error on test set 7054.045634195742
```

In [128]:
```
regr.score(x_train,y_train)
```

Out[128]:
```
0.9913834208596437
```

Conclusion

The accuracy is same for all Linear,Ridge,Lasso Regressions where Elastic Net has less accuracy compared to them.