PROBLEM STATEMENT:- TO PREDICT THE RAINFALL BASED ON VARIOUS FEATURES OF THEDATASET

In [ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

READ THE DATASET

In [2]:
```
df=pd.read_csv(r"C:\Users\pucha\Downloads\rainfall in india 1901-2015.csv")
df
```

Out[2]:

	SUBDIVISION	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC	ANNUAL	Jan-Feb	Mar-May	Jun-Sep	O D
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

DATA CLEANING AND PREPROCESSING

In [63]:
```
df.head()
```

Out[63]:

	JAN	FEB	MAR	APR	DEC
0	49.2	87.1	29.2	2.3	33.6
1	0.0	159.8	12.2	0.0	160.5
2	12.7	144.0	0.0	1.0	225.0
3	9.4	14.7	0.0	202.4	40.1
4	1.3	0.0	3.3	26.9	344.7

In [64]:
```
df.tail()
```

Out[64]:

	JAN	FEB	MAR	APR	DEC
4111	5.1	2.8	3.1	85.9	14.9
4112	19.2	0.1	1.6	76.8	8.8
4113	26.2	34.4	37.5	5.3	26.7
4114	53.2	16.1	4.4	14.9	62.3
4115	2.2	0.5	3.7	87.1	159.0

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

Out[5]:
```
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 [6]:
```
df.shape
```

Out[6]:
```
(4116, 19)
```

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

Out[7]:

	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	4
mean	1958.218659	18.957320	21.805325	27.359197	43.127432	85.745417	230.234444	347.214334	290.263497	197.361922	
std	33.140898	33.585371	35.909488	46.959424	67.831168	123.234904	234.710758	269.539667	188.770477	135.408345	
min	1901.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.400000	0.000000	0.000000	0.100000	
25%	1930.000000	0.600000	0.600000	1.000000	3.000000	8.600000	70.350000	175.600000	155.975000	100.525000	
50%	1958.000000	6.000000	6.700000	7.800000	15.700000	36.600000	138.700000	284.800000	259.400000	173.900000	
75%	1987.000000	22.200000	26.800000	31.300000	49.950000	97.200000	305.150000	418.400000	377.800000	265.800000	
max	2015.000000	583.700000	403.500000	605.600000	595.100000	1168.600000	1609.900000	2362.800000	1664.600000	1222.000000	

In [8]:
```
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
```

TO FIND THE NULL VALUES

In [65]:
```
df.isnull().sum()
```

Out[65]:
```
JAN    0
FEB    0
MAR    0
APR    0
DEC    0
dtype: int64
```

In [66]:
```
df.fillna(method="ffill",inplace=True)
```

```
C:\Users\pucha\AppData\Local\Temp\ipykernel_11648\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 [67]:
```
df.isnull().sum()
```

Out[67]:
```
JAN    0
FEB    0
MAR    0
APR    0
DEC    0
dtype: int64
```

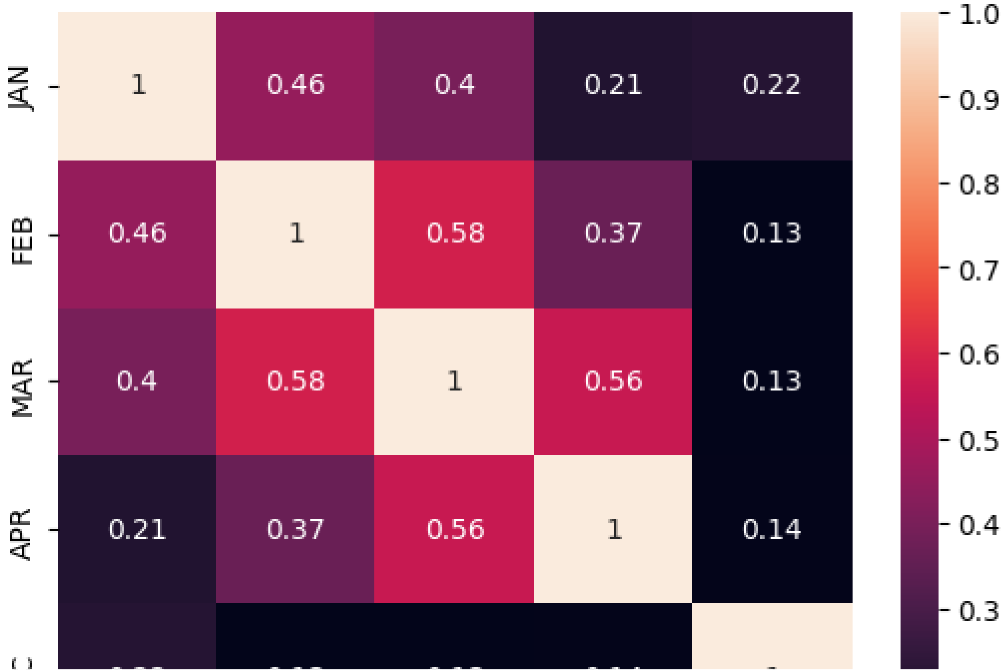
TO COUNT THE VALUES

In [12]:
```
df['YEAR'].value_counts()
```

Out[12]:
```
YEAR
1963    36
2002    36
1976    36
1975    36
1974    36
        ..
1915    35
1918    35
1954    35
1955    35
1909    34
Name: count, Length: 115, dtype: int64
```

EXPLORATARY DATA ANALYSIS:-

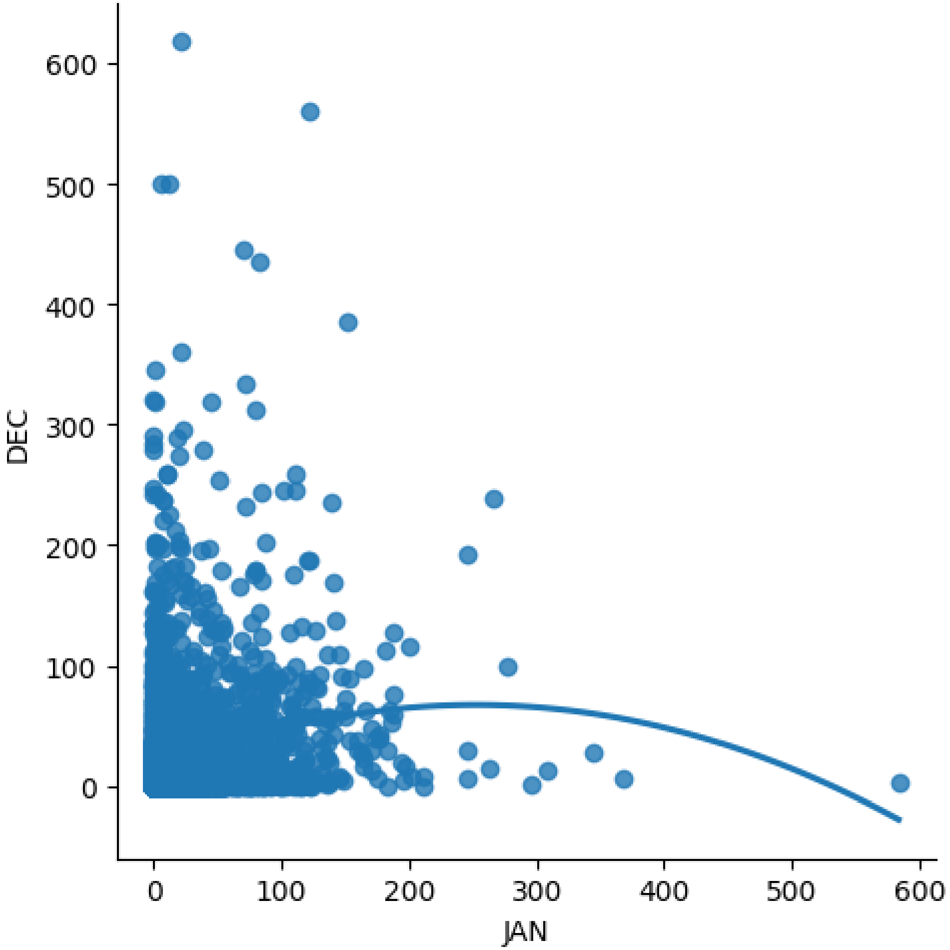
In [68]:
```
df=df[['JAN','FEB','MAR','APR','DEC']]
sns.heatmap(df.corr(),annot=True)
plt.show()
```



In [14]:
```
df.columns
```

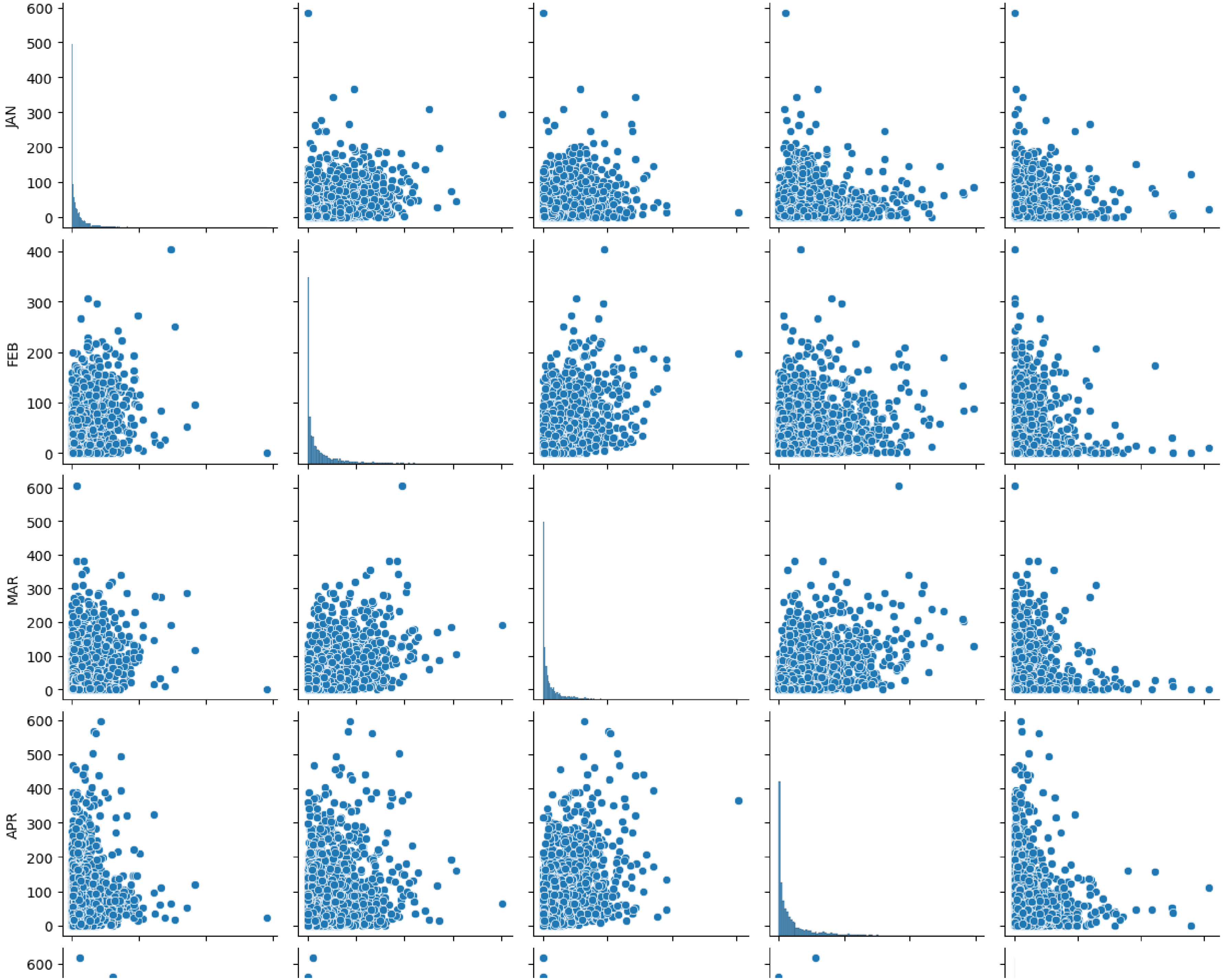
Out[14]:
```
Index(['JAN', 'FEB', 'MAR', 'APR', 'DEC'], dtype='object')
```

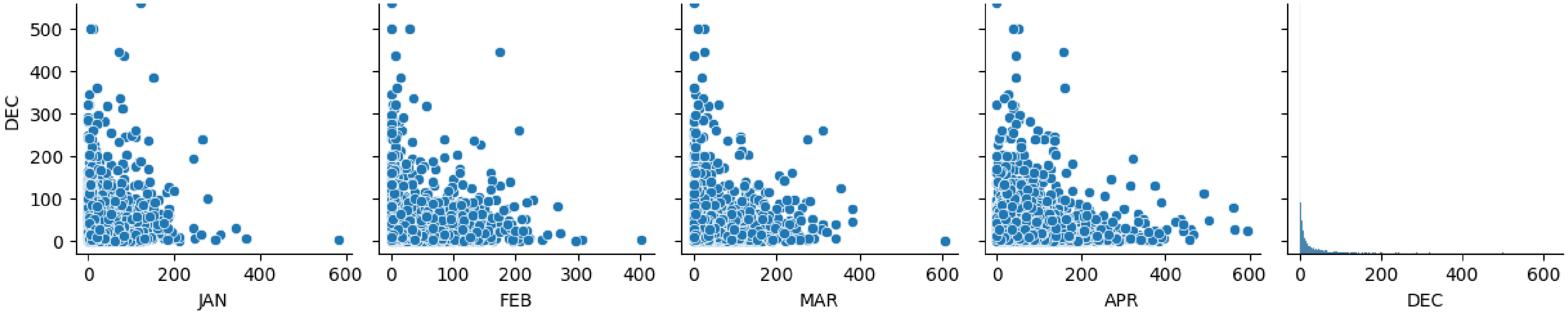
In [15]:
```
sns.lmplot(x='JAN',y='DEC',order=2,data=df,ci=None)
plt.show()
```



In [16]:
```
sns.pairplot(df)
```

Out[16]:
```
<seaborn.axisgrid.PairGrid at 0x266c4057370>
```


JAN
FEB
MAR
APR
600
500
400
300
200
100
0



In [23]:
```
x=df[['JAN']]
y=df['FEB']
```

In [24]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.33,random_state=42)
```

LINEAR REGRESSION:-

In [25]:
```
from sklearn.linear_model import LinearRegression
reg=LinearRegression()
reg.fit(x_train,y_train)
print(reg.intercept_)
coeff_=pd.DataFrame(reg.coef_,x.columns,columns=['coefficient'])
coeff_
```

```
12.044171565057082
```

Out[25]:

	coefficient
JAN	0.493537

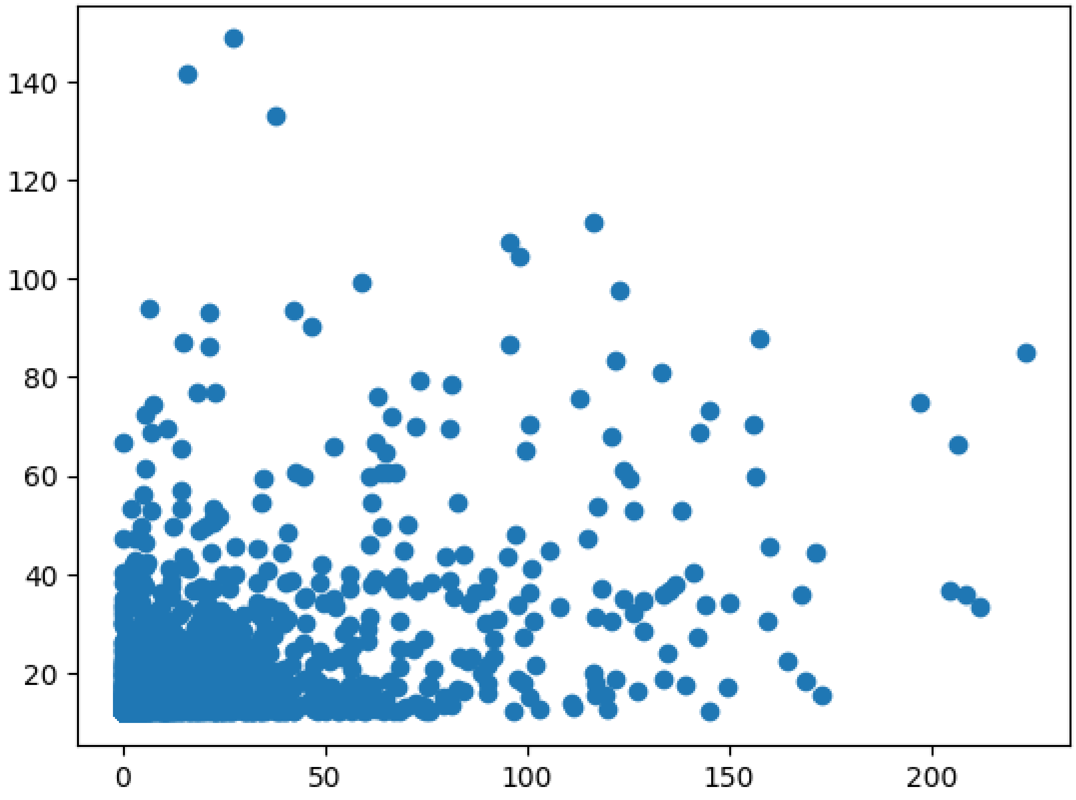
In [26]:
```
score=reg.score(x_test,y_test)
print(score)
```

```
0.18220657419532882
```

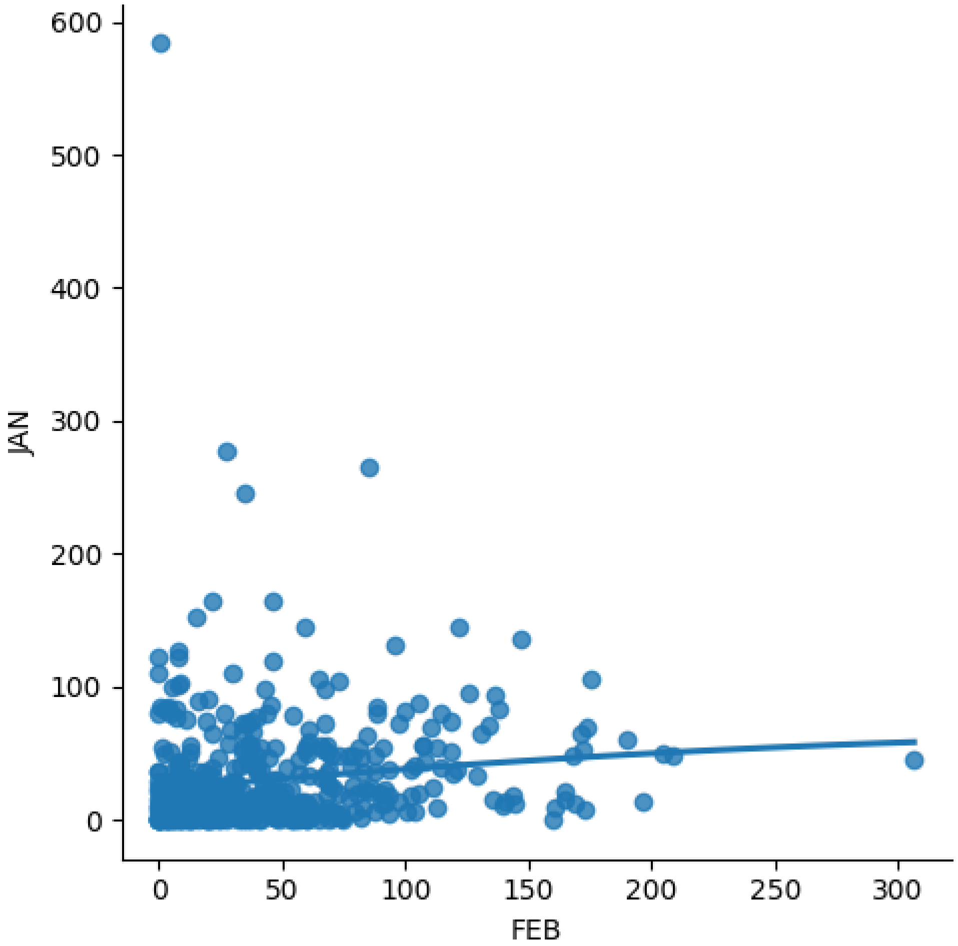
In [27]:
```
predictions=reg.predict(x_test)
```

In [28]:
```
plt.scatter(y_test,predictions)
```

Out[28]: <matplotlib.collections.PathCollection at 0x266c8bbaa10>



In [29]:
```
df500=df[:][:500]
sns.lmplot(x="FEB",y="JAN",order=2,ci=None,data=df500)
plt.show()
```

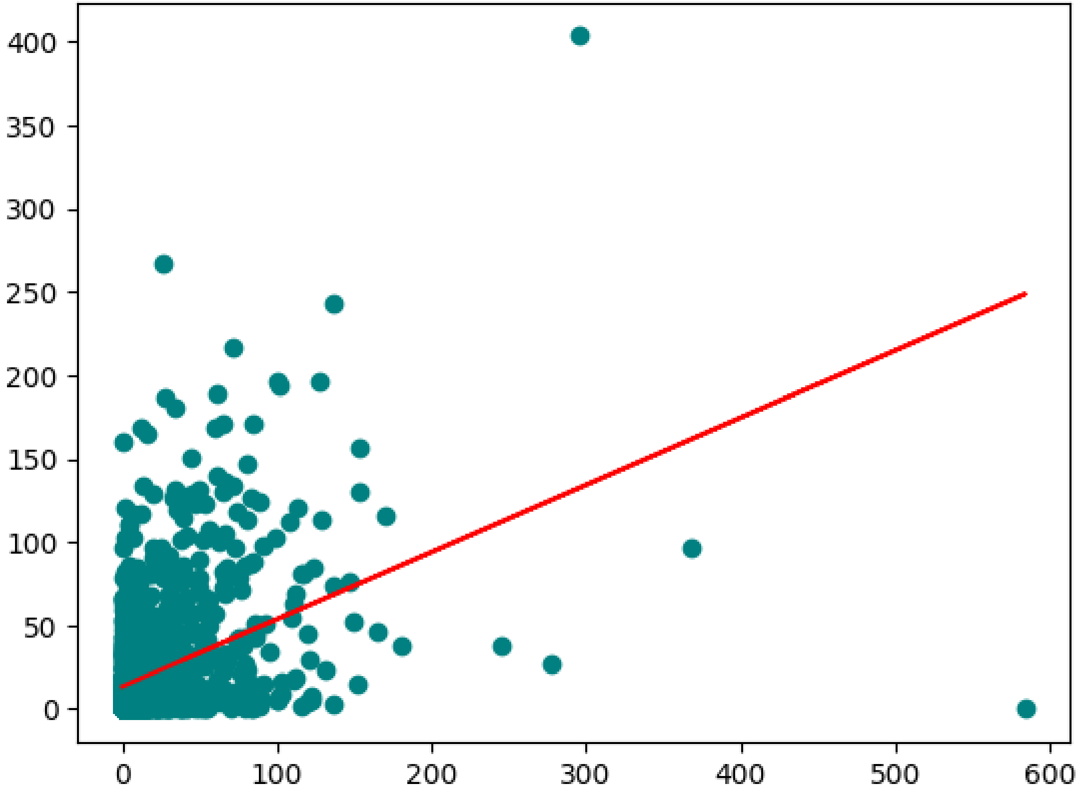


In [30]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.33)
reg.fit(x_train,y_train)
reg.fit(x_test,y_test)
```

Out[30]: LinearRegression()

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [33]:
```
y_pred=reg.predict(x_test)
plt.scatter(x_test,y_test,color='teal')
plt.plot(x_test,y_pred,color='red')
plt.show()
```



In [34]:
```
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 Score:",r2)
```

```
R2 Score: 0.14370148770973235
```

RIDGE MODEL:

In [69]:
```
from sklearn.linear_model import Lasso,Ridge
from sklearn.preprocessing import StandardScaler
```

In [70]:
```
features= df.columns[0:1]
target= df.columns[-5]
```

In [71]:
```
x=np.array(df['JAN']).reshape(-1,1)
y=np.array(df['FEB']).reshape(-1,2)
```

In [52]:
```
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,random_state=17)
```

In [53]:
```
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)
```

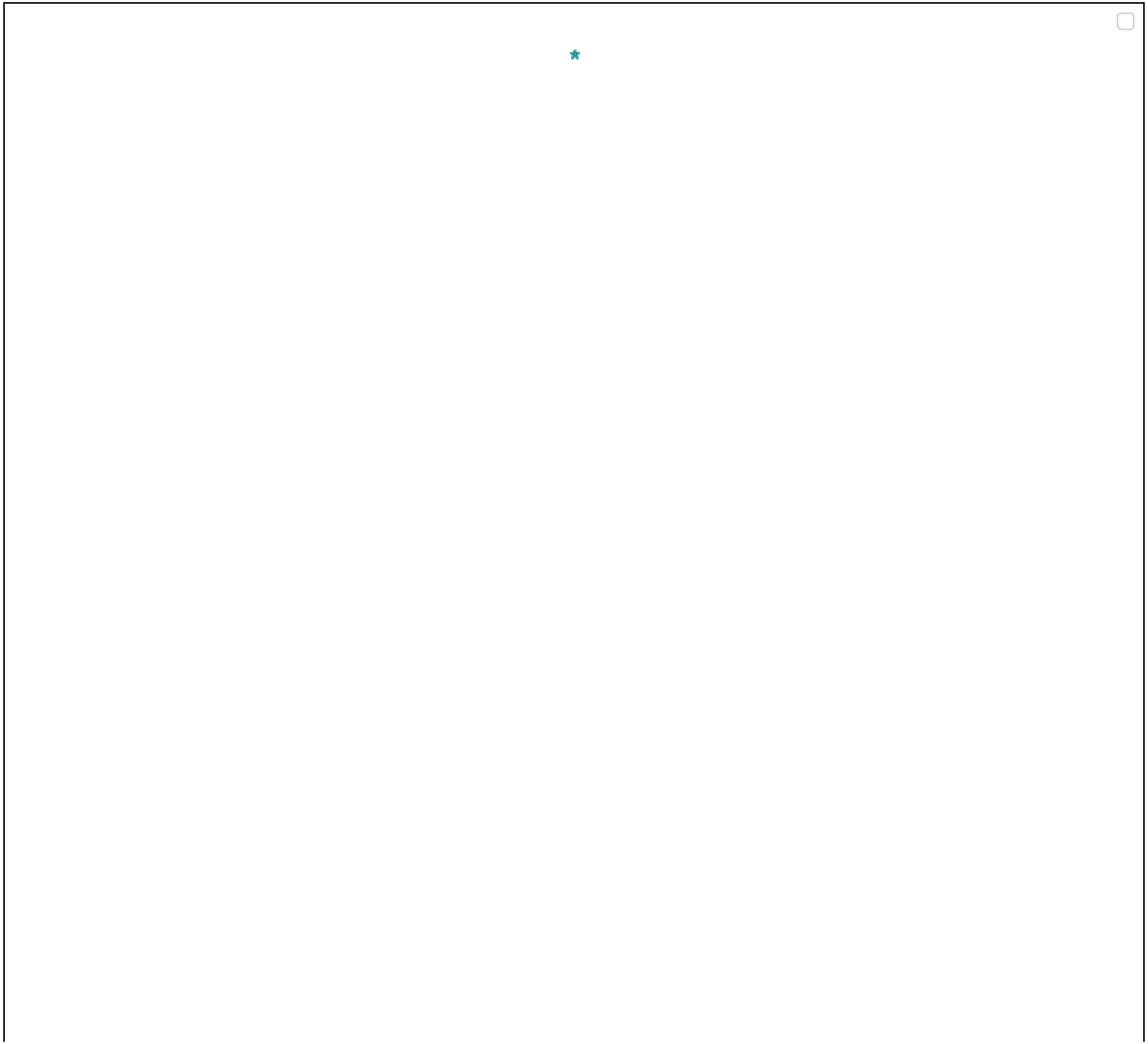
In [54]:
```
print("\n Ridge Model:\n")
print("the train score for ridge model is{}".format(train_score_ridge))
print("the test score for ridge model is{}".format(test_score_ridge))
```

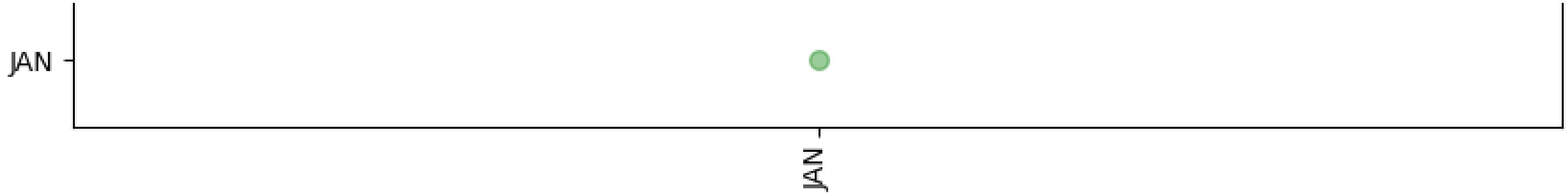
```
 Ridge Model:

the train score for ridge model is0.9999999999904551
the test score for ridge model is0.9999999999904435
```

In [55]:
```
lr=LinearRegression()
```

In [56]:
```
plt.figure(figsize= (10,10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker="*",markersize=5,color="teal")
plt.plot(features,alpha=0.4,linestyle='none',marker='o',markersize=7,color="green")
plt.xticks(rotation = 90)
plt.legend()
plt.show()
```

```
No artists with labels found to put in legend.  Note that artists whose label start with an underscore are ignor
ed when legend() is called with no argument.
```





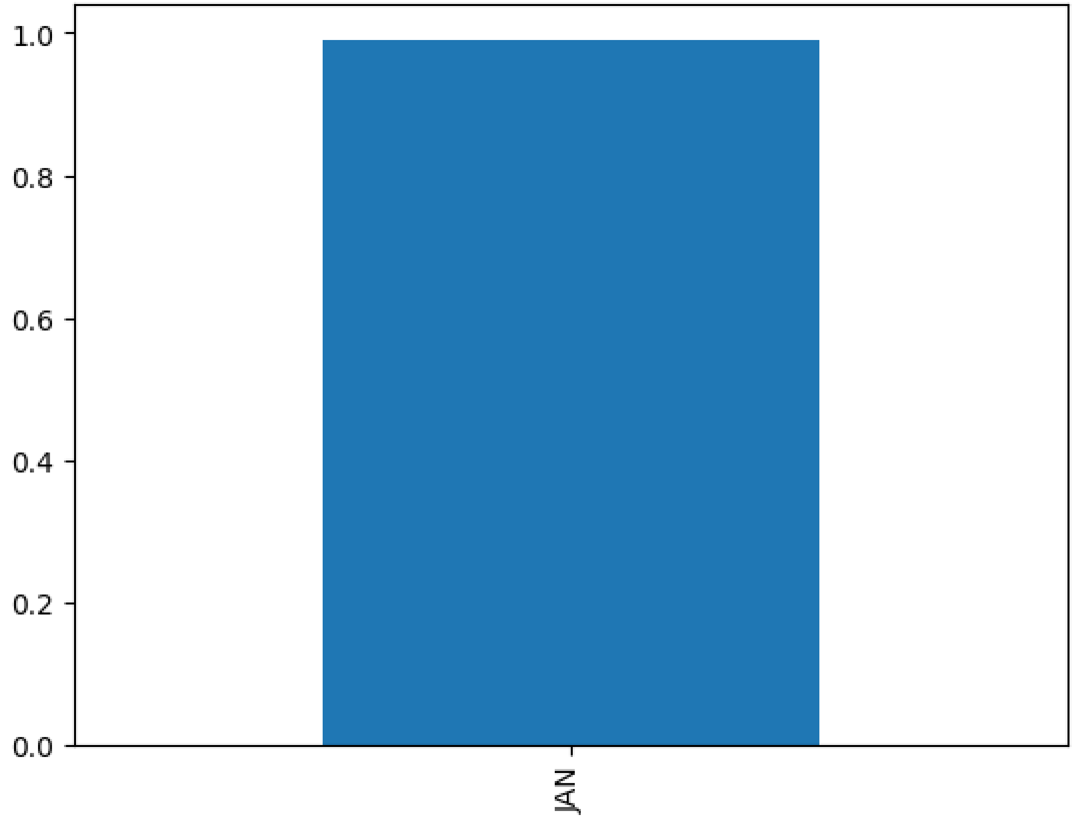
LASSO MODEL

In [57]:
```
print("\n Lasso 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 0.9999207747038827
The test score for ls model is0.9999206791315255
```

In [58]:
```
pd.Series(lasso.coef_,features).sort_values(ascending=True).plot(kind="bar")
```

Out[58]: <Axes: >

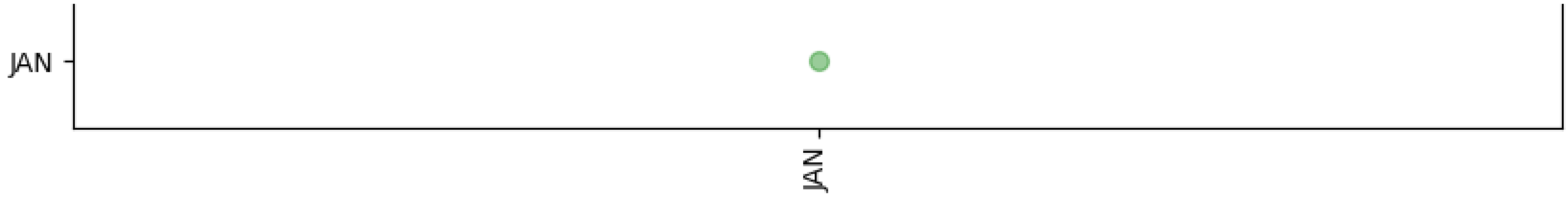


In [59]:
```
from sklearn.linear_model import LassoCV
lasso_cv=LassoCV(alphas=[0.0001,0.001,0.01,1,10],random_state=0).fit(x_train,y_train)
print(lasso_cv.score(x_train,y_train))
print(lasso_cv.score(x_test,y_test))
```

```
0.9999999999999921
0.9999999999999921
```

In [60]:
```
plt.figure(figsize= (10,10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker="*",markersize=5,color="yellow")
plt.plot(lasso_cv.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color='blue')
plt.plot(features,alpha=0.4,linestyle='none',marker='o',markersize=7,color="green")
plt.xticks(rotation = 90)
plt.legend()
plt.show()
```

```
No artists with labels found to put in legend.  Note that artists whose label start with an underscore are ignor
ed when legend() is called with no argument.
```



ELASTIC NET:-

In [61]:
```
from sklearn.linear_model import ElasticNet
eln=ElasticNet()
eln.fit(x,y)
print(eln.coef_)
print(eln.intercept_)
print(eln.score(x,y))
```

```
[0.99911315]
0.016812222871418925
0.999999213497588
```

In [62]:
```
y_pred_elastic = reg.predict(x_train)
mean_squared_error=np.mean((y_pred_elastic - y_train)**2)
print(mean_squared_error)
```

```
403.6352771780373

C:\Users\pucha\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\base.py:439: UserWarning: X doe
s not have valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```

CONCLUSION:-

THE SCORE OF LINEAR REGRESSION IS :- 0.1793580786264921

THE SCORE OF RIDGE MODEL IS :- 0.99999999998833

THE SCORE OF LASSO MODEL IS :- 0.999999999999992

THE SCORE OF ELASTIC NET IS :- 0.9999992160905338

AMONG ALL MODELS LASSO YEILD HIGHEST ACCURACY.SO,WE PREFER LASSO MODEL FOR THIS DATA SET

In [ ]: