

```
In [1]: import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn import preprocessing, svm
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import seaborn as sns
```

```
In [2]: df=pd.read_csv(r"C:\Users\DELL\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
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
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
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
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
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
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
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
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
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
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
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

4116 rows × 19 columns



In [3]: `df.head()`

Out[3]:

	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	1696.1
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
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
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
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

In [4]: `df.tail()`

Out[4]:

	SUBDIVISION	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC	ANNUAL	Jan-Feb	Mar-May	Jun-Sep
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	107.4
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	111.7
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	109.6
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	99.6
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	86.6

```
In [5]: df.isnull().any()
```

```
Out[5]: SUBDIVISION    False  
YEAR                False  
JAN                  True  
FEB                  True  
MAR                  True  
APR                  True  
MAY                  True  
JUN                  True  
JUL                  True  
AUG                  True  
SEP                  True  
OCT                  True  
NOV                  True  
DEC                  True  
ANNUAL               True  
Jan-Feb              True  
Mar-May              True  
Jun-Sep              True  
Oct-Dec              True  
dtype: bool
```

```
In [6]: df.fillna(method='ffill',inplace=True)
```

In [7]: `df.isnull().sum()`

```
Out[7]: SUBDIVISION    0
YEAR                0
JAN                 0
FEB                 0
MAR                 0
APR                 0
MAY                 0
JUN                 0
JUL                 0
AUG                 0
SEP                 0
OCT                 0
NOV                 0
DEC                 0
ANNUAL              0
Jan-Feb             0
Mar-May             0
Jun-Sep             0
Oct-Dec             0
dtype: int64
```

In [8]: `df.describe()`

```
Out[8]:
```

	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP
<b>count</b>	4116.000000	4116.000000	4116.000000	4116.000000	4116.000000	4116.000000	4116.000000	4116.000000	4116.000000	4116.000000
<b>mean</b>	1958.218659	18.957240	21.823251	27.415379	43.160641	85.788994	230.567979	347.177235	290.239796	197.524
<b>std</b>	33.140898	33.576192	35.922602	47.045473	67.816588	123.220150	234.896056	269.321089	188.785639	135.509
<b>min</b>	1901.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.400000	0.000000	0.000000	0.100
<b>25%</b>	1930.000000	0.600000	0.600000	1.000000	3.000000	8.600000	70.475000	175.900000	155.850000	100.575
<b>50%</b>	1958.000000	6.000000	6.700000	7.900000	15.700000	36.700000	138.900000	284.800000	259.400000	174.000
<b>75%</b>	1987.000000	22.200000	26.800000	31.400000	50.125000	97.400000	306.150000	418.325000	377.800000	266.225
<b>max</b>	2015.000000	583.700000	403.500000	605.600000	595.100000	1168.600000	1609.900000	2362.800000	1664.600000	1222.000

In [9]: 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                   4116 non-null   float64
3   FEB                   4116 non-null   float64
4   MAR                   4116 non-null   float64
5   APR                   4116 non-null   float64
6   MAY                   4116 non-null   float64
7   JUN                   4116 non-null   float64
8   JUL                   4116 non-null   float64
9   AUG                   4116 non-null   float64
10  SEP                   4116 non-null   float64
11  OCT                   4116 non-null   float64
12  NOV                   4116 non-null   float64
13  DEC                   4116 non-null   float64
14  ANNUAL                4116 non-null   float64
15  Jan-Feb               4116 non-null   float64
16  Mar-May               4116 non-null   float64
17  Jun-Sep               4116 non-null   float64
18  Oct-Dec               4116 non-null   float64
dtypes: float64(17), int64(1), object(1)
memory usage: 611.1+ KB
```

In [10]: df.columns

```
Out[10]: 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 [11]: df.shape

```
Out[11]: (4116, 19)
```

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

```
Out[12]: ANNUAL
790.5      4
770.3      4
1836.2     4
1024.6     4
1926.5     3
..
443.9      1
689.0      1
605.2      1
509.7      1
1642.9     1
Name: count, Length: 3712, dtype: int64
```

```
In [13]: df['Jan-Feb'].value_counts()
```

```
Out[13]: Jan-Feb
0.0      238
0.1       80
0.2       52
0.3       38
0.4       32
...
23.3      1
95.2      1
76.9      1
66.5      1
69.3      1
Name: count, Length: 1220, dtype: int64
```

```
In [14]: df['Mar-May'].value_counts()
```

```
Out[14]: Mar-May
0.0      29
0.1      13
0.3      11
8.3      11
11.5     10
..
246.3     1
248.1     1
151.3     1
249.5     1
223.9     1
Name: count, Length: 2262, dtype: int64
```

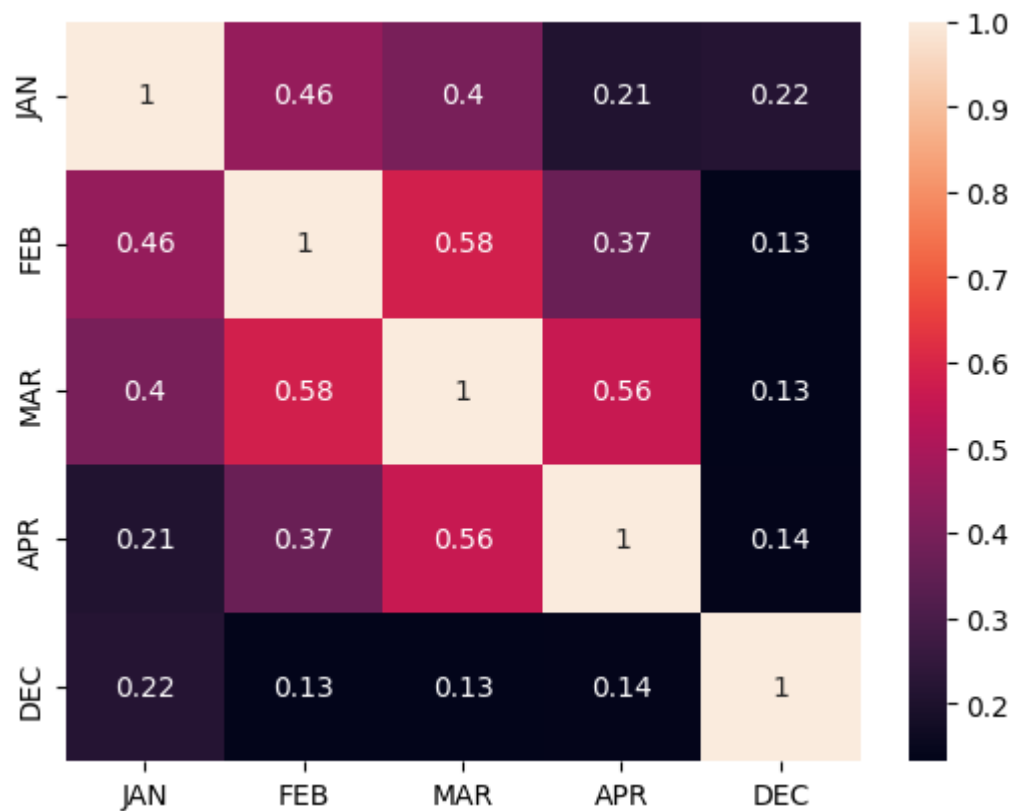
```
In [15]: df['Oct-Dec'].value_counts()
```

```
Out[15]: Oct-Dec
0.0      16
0.1      15
0.5      13
0.6      12
0.7      11
..
191.5     1
124.5     1
139.1     1
41.5      1
555.4     1
Name: count, Length: 2389, dtype: int64
```

EXPLORATORY DATA ANALYSIS:-



```
In [16]: df=df[['JAN', 'FEB', 'MAR', 'APR', 'DEC']]  
sns.heatmap(df.corr(),annot=True)  
plt.show()
```



```
In [17]: df.columns
```

```
Out[17]: Index(['JAN', 'FEB', 'MAR', 'APR', 'DEC'], dtype='object')
```

```
In [18]: x=df[["FEB"]]  
y=df[["JAN"]]
```

```
In [19]: from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(x,y,test_size=0.33,random_state=42)
```

```
In [20]: 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_
```

9.650666612303553

```
Out[20]:
```

	coefficient
FEB	0.442278

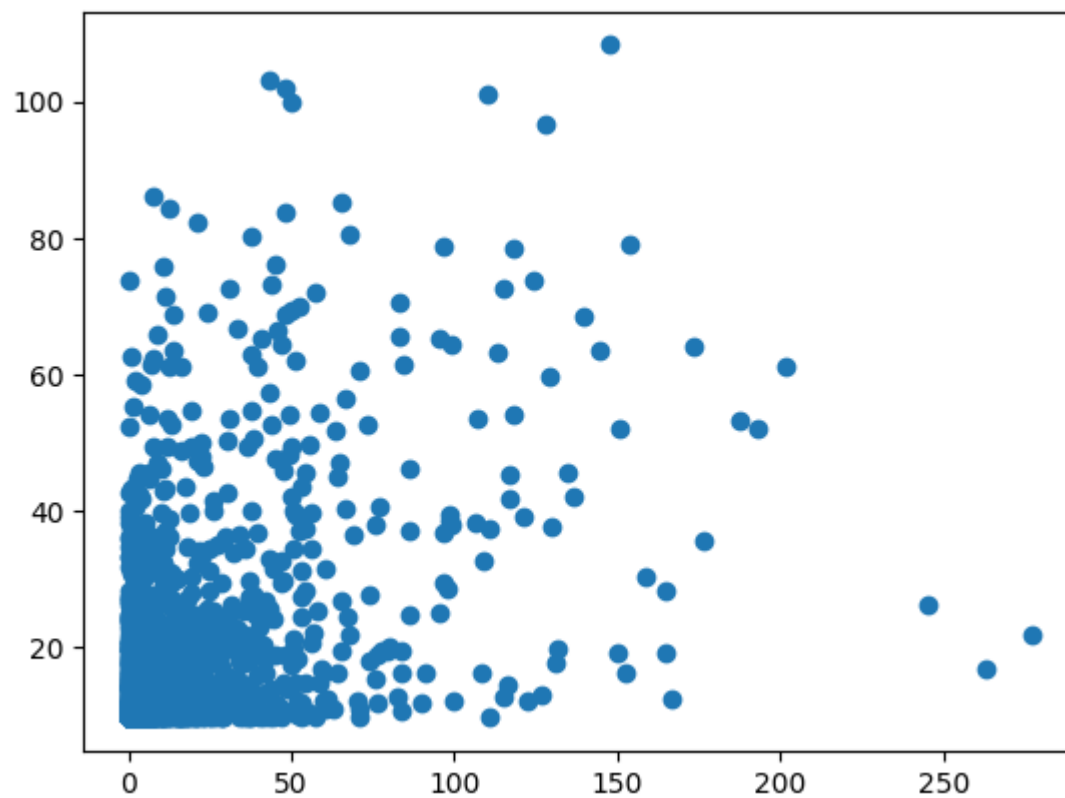
```
In [21]: score=reg.score(X_test,y_test)
print(score)
```

0.1793580786264921

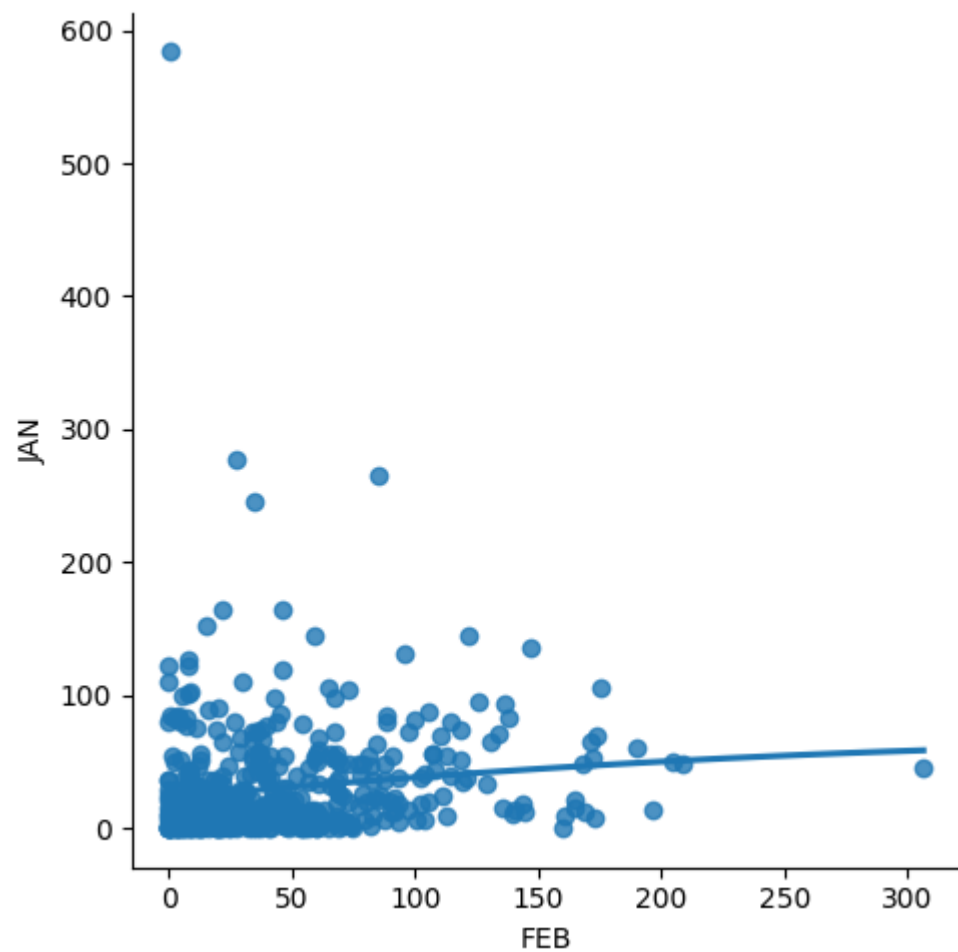
```
In [22]: predictions=reg.predict(X_test)
```

```
In [23]: plt.scatter(y_test,predictions)
```

```
Out[23]: <matplotlib.collections.PathCollection at 0x2429875b3d0>
```



```
In [24]: df500=df[:][:500]  
sns.lmplot(x="FEB",y="JAN",order=2,ci=None,data=df500)  
plt.show()
```

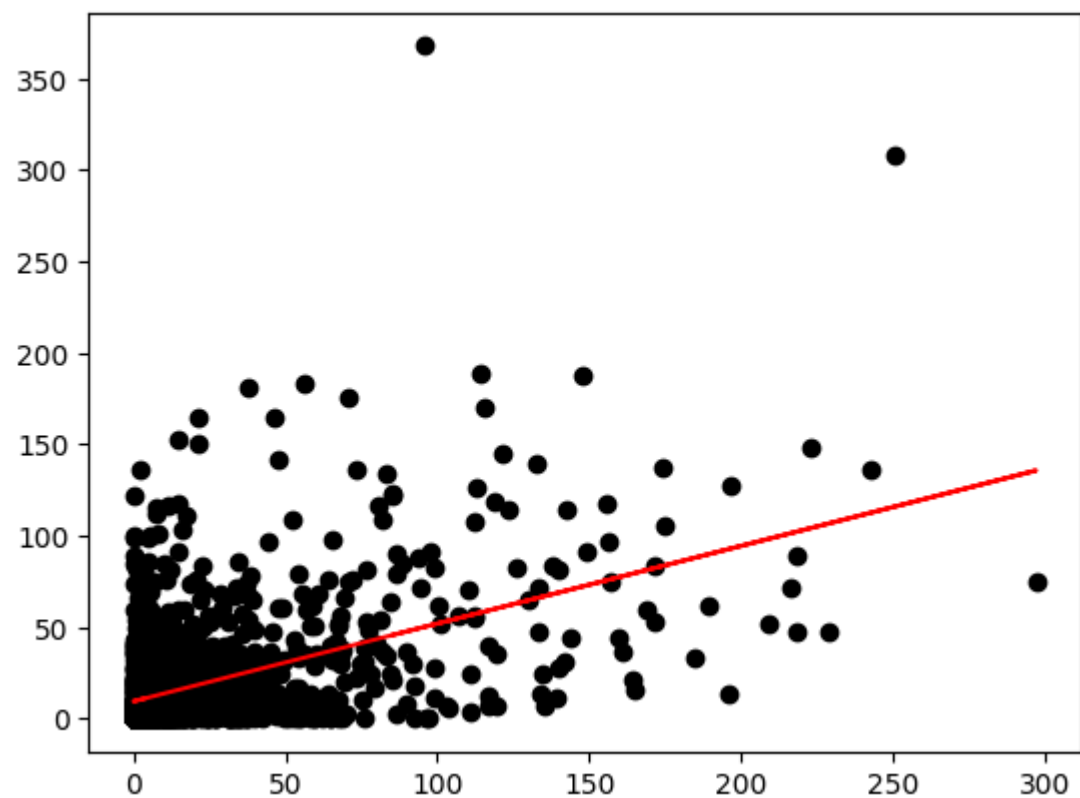


```
In [25]: X_train,X_test,y_train,y_test=train_test_split(x,y,test_size=0.33)
reg.fit(X_train,y_train)
reg.fit(X_test,y_test)
```

Out[25]: 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 [26]: y_pred=reg.predict(X_test)
plt.scatter(X_test,y_test,color='black')
plt.plot(X_test,y_pred,color='red')
plt.show()
```



```
In [27]: 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: 0.24027079060270906

RIDGE MODEL:-

```
In [28]: from sklearn.linear_model import Lasso,Ridge
from sklearn.preprocessing import StandardScaler
```

```
In [42]: features= df.columns[0:1]
target= df.columns[-5]
```

```
In [43]: x=np.array(df['JAN']).reshape(-1,1)
y=np.array(df['FEB']).reshape(-1,2)
```

```
In [44]: 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 [45]: 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 [46]: 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 is0.999999999904551  
the test score for ridge model is0.999999999904435

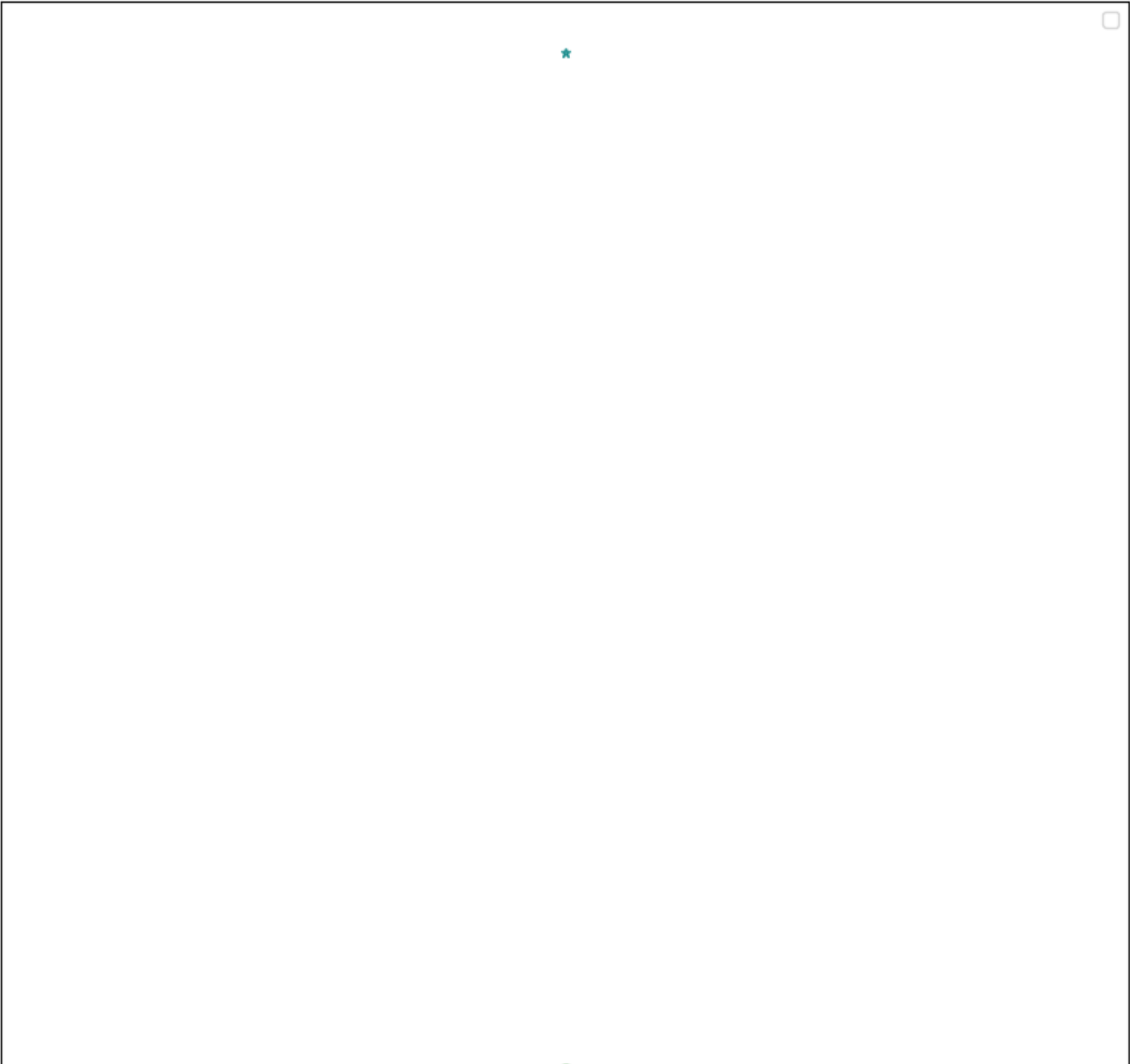
```
In [47]: lr=LinearRegression()
```

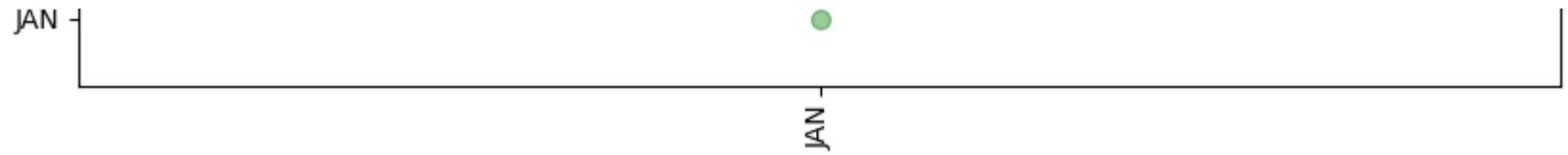
```
In [48]: plt.figure(figsize= (10,10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker="*",markersize=5,color="teal")
plt.plot(features,alpha=0.4,linestyle='none',marker='o',markersize=7,color="green")
plt.xticks(rotation = 90)
plt.legend()
plt.show()
```

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









### LASSO MODEL:-

```
In [49]: 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))
```

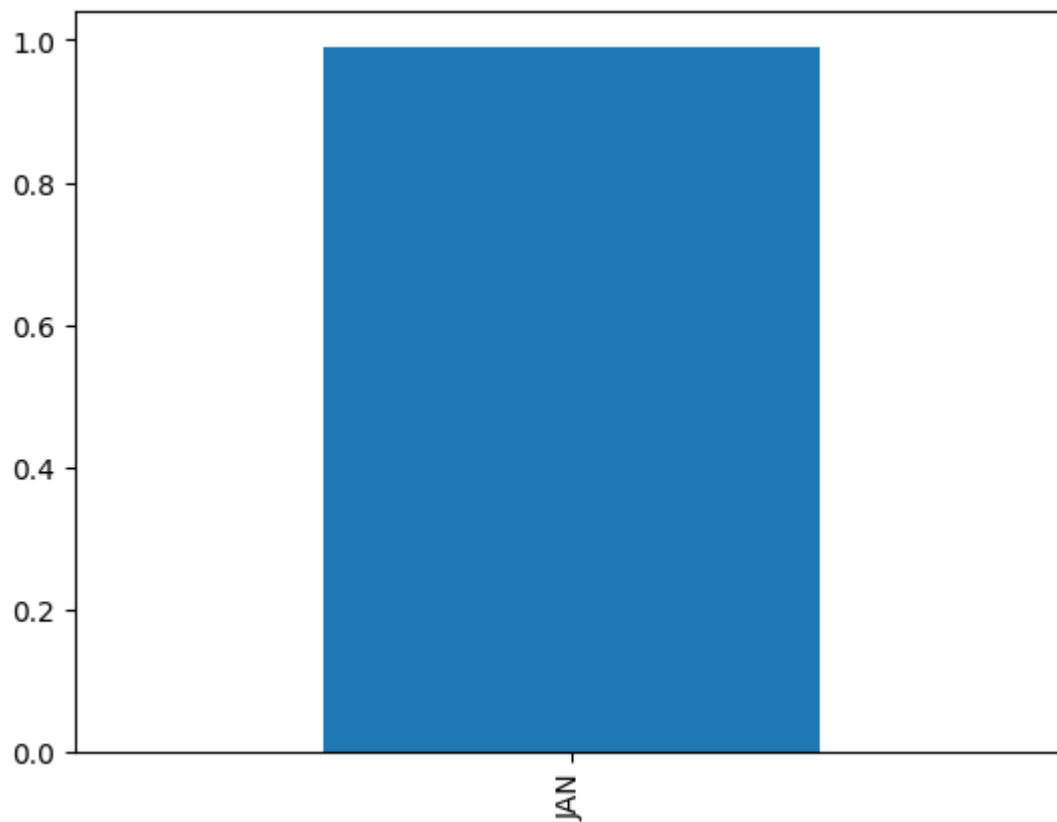
Lasso Model:

The train score for ls model is 0.9999207747038827

The test score for ls model is0.9999206791315255

```
In [50]: pd.Series(lasso.coef_,features).sort_values(ascending=True).plot(kind="bar")
```

```
Out[50]: <Axes: >
```



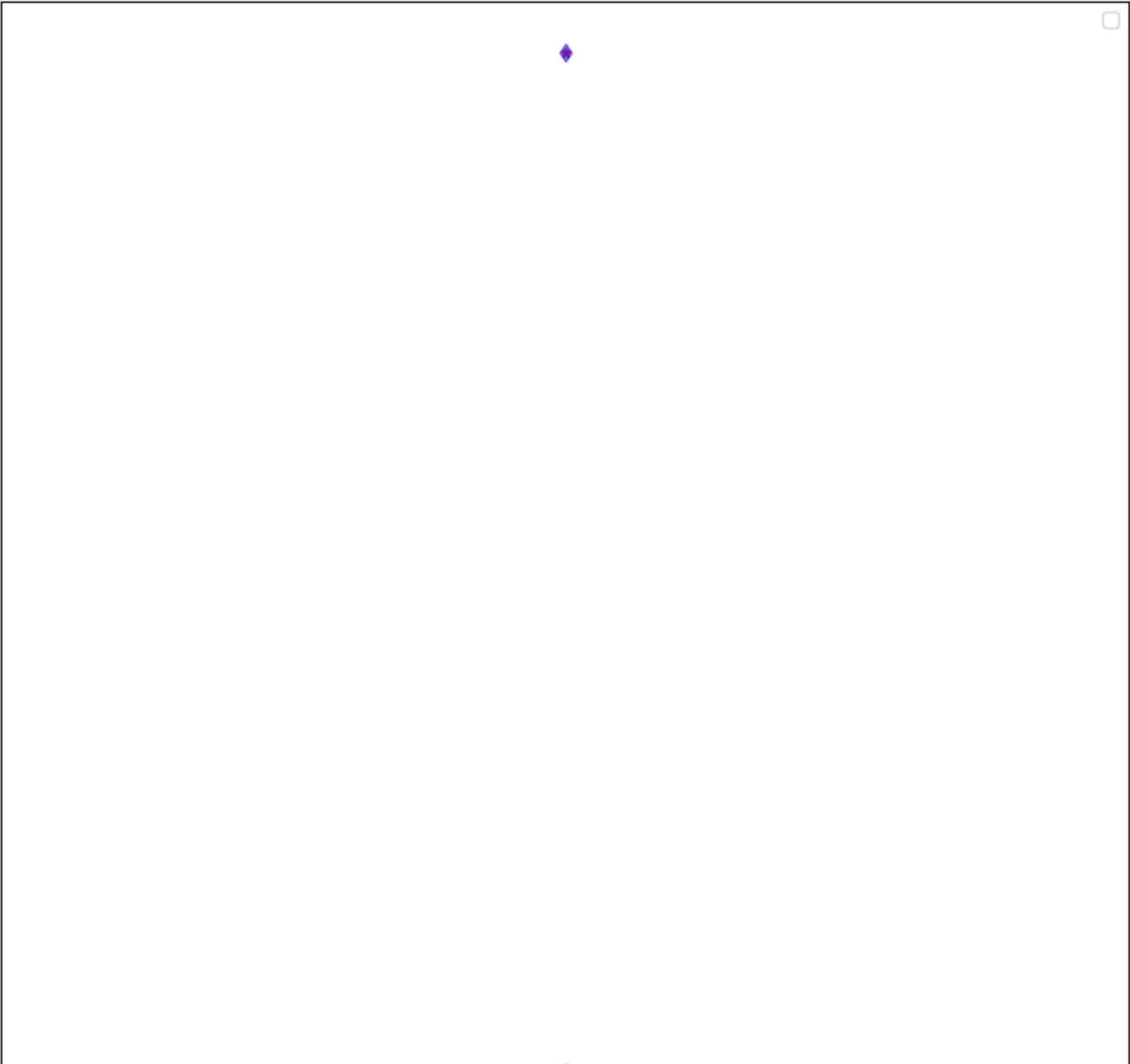
```
In [51]: from sklearn.linear_model import LassoCV
lasso_cv=LassoCV(alphas=[0.0001,0.001,0.01,1,10],random_state=0).fit(x_train,y_train)
print(lasso_cv.score(x_train,y_train))
print(lasso_cv.score(x_test,y_test))
```

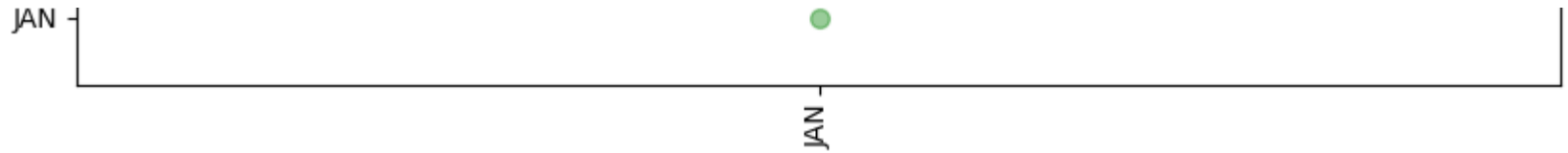
```
0.9999999999999921
0.9999999999999921
```

```
In [52]: plt.figure(figsize= (10,10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker="*",markersize=5,color='red')
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 ignored when legend() is called with no argument.







```
In [53]: from sklearn.linear_model import ElasticNet
elnet=ElasticNet()
elnet.fit(x,y)
print(elnet.coef_)
print(elnet.intercept_)
print(elnet.score(x,y))
```

```
[0.99911315]
0.01681222287140116
0.9999992134975881
```

```
In [54]: y_pred_elastic = reg.predict(x_train)
mean_squared_error=np.mean((y_pred_elastic - y_train)**2)
print(mean_squared_error)
```

```
374.33713051898116
```

C:\Users\DELL\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\base.py:439: UserWarning: X does not have valid feature names, but LinearRegression was fitted with feature names  
warnings.warn(

ELASTIC NET:-

```
In [56]: from sklearn.linear_model import ElasticNet
elnet=ElasticNet()
elnet.fit(x,y)
print(elnet.coef_)
print(elnet.intercept_)
print(elnet.score(x,y))
```

```
[0.99911315]
0.01681222287140116
0.9999992134975881
```



```
In [58]: y_pred_elastic = reg.predict(x_train)
mean_squared_error=np.mean((y_pred_elastic - y_train)**2)
print(mean_squared_error)
```

374.33713051898116

C:\Users\DELL\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\base.py:439: UserWarning: X does 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 :- 0.99999999999992  
THE SCORE OF ELASTIC NET IS :- 0.9999992160905338  
*\*AMONG ALL MODELS LASSO YEILD HIGHEST ACCURACY.SO,WE PREFER LASSO MODEL FO  
R THIS DATA SET\**

In [ ]: