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

Linear Regression

Data Collection:

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\91903\Downloads\rainfall in india 1901-2015.csv")
df
```

Out[2]:

	SUBDIVISION	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT
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
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
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
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
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
...	...	...	...	...	...	...	...	...	...	...	...	...
4111	LAKSHADWEEP	2011	5.1	2.8	3.1	85.9	107.2	153.6	350.2	254.0	255.2	117.4
4112	LAKSHADWEEP	2012	19.2	0.1	1.6	76.8	21.2	327.0	231.5	381.2	179.8	145.9
4113	LAKSHADWEEP	2013	26.2	34.4	37.5	5.3	88.3	426.2	296.4	154.4	180.0	72.8
4114	LAKSHADWEEP	2014	53.2	16.1	4.4	14.9	57.4	244.1	116.1	466.1	132.2	169.2
4115	LAKSHADWEEP	2015	2.2	0.5	3.7	87.1	133.1	296.6	257.5	146.4	160.4	165.4

4116 rows × 19 columns

Data Cleaning and preprocessing:

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

Out[3]:

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

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

Out[4]:

	SUBDIVISION	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	
4111	LAKSHADWEEP	2011	5.1	2.8	3.1	85.9	107.2	153.6	350.2	254.0	255.2	117.4	1
4112	LAKSHADWEEP	2012	19.2	0.1	1.6	76.8	21.2	327.0	231.5	381.2	179.8	145.9	
4113	LAKSHADWEEP	2013	26.2	34.4	37.5	5.3	88.3	426.2	296.4	154.4	180.0	72.8	
4114	LAKSHADWEEP	2014	53.2	16.1	4.4	14.9	57.4	244.1	116.1	466.1	132.2	169.2	
4115	LAKSHADWEEP	2015	2.2	0.5	3.7	87.1	133.1	296.6	257.5	146.4	160.4	165.4	2

In [5]:
```
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 [6]:
```
df.describe()
```

Out[6]:

	YEAR	JAN	FEB	MAR	APR	MAY	JU
count	4116.000000	4112.000000	4113.000000	4110.000000	4112.000000	4113.000000	4111.0000
mean	1958.218659	18.957320	21.805325	27.359197	43.127432	85.745417	230.2344
std	33.140898	33.585371	35.909488	46.959424	67.831168	123.234904	234.7107
min	1901.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.4000
25%	1930.000000	0.600000	0.600000	1.000000	3.000000	8.600000	70.3500
50%	1958.000000	6.000000	6.700000	7.800000	15.700000	36.600000	138.7000
75%	1987.000000	22.200000	26.800000	31.300000	49.950000	97.200000	305.1500
max	2015.000000	583.700000	403.500000	605.600000	595.100000	1168.600000	1609.9000

In [8]:
```
df.isna().any()
```

Out[8]:
```
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 [9]:
```
df.fillna(method='ffill',inplace=True)
```

In [10]:
```
df.isnull().sum()
```

Out[10]:
```
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 [11]:
```
df.describe()
```

Out[11]:

Y	JUN	JUL	AUG	SEP	OCT	NOV	DEC	
0	4116.000000	4116.000000	4116.000000	4116.000000	4116.000000	4116.000000	4116.000000	41
4	230.567979	347.177235	290.239796	197.524781	95.724198	40.081997	19.042225	14
0	234.896056	269.321089	188.785639	135.509037	99.689878	68.851397	42.655830	9
0	0.400000	0.000000	0.000000	0.100000	0.000000	0.000000	0.000000	
0	70.475000	175.900000	155.850000	100.575000	14.600000	0.700000	0.100000	8
0	138.900000	284.800000	259.400000	174.000000	65.750000	9.700000	3.100000	11
0	306.150000	418.325000	377.800000	266.225000	148.600000	46.325000	17.600000	16
0	1609.900000	2362.800000	1664.600000	1222.000000	948.300000	648.900000	617.500000	63

In [12]:
```
df.columns
```

Out[12]:
```
Index(['SUBDIVISION', 'YEAR', 'JAN', 'FEB', 'MAR', 'APR', 'MAY', 'JUN', 'JU
L',
       'AUG', 'SEP', 'OCT', 'NOV', 'DEC', 'ANNUAL', 'Jan-Feb', 'Mar-May',
       'Jun-Sep', 'Oct-Dec'],
      dtype='object')
```

In [13]:
```
df.shape
```

Out[13]:
```
(4116, 19)
```

In [14]:
```
df['Jan-Feb'].value_counts()
```

Out[14]:
```
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 [15]:
```
df['ANNUAL'].value_counts()
```

Out[15]:
```
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 [16]:
```
df['Mar-May'].value_counts()
```

Out[16]:
```
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 [17]:
```
df['Jun-Sep'].value_counts()
```

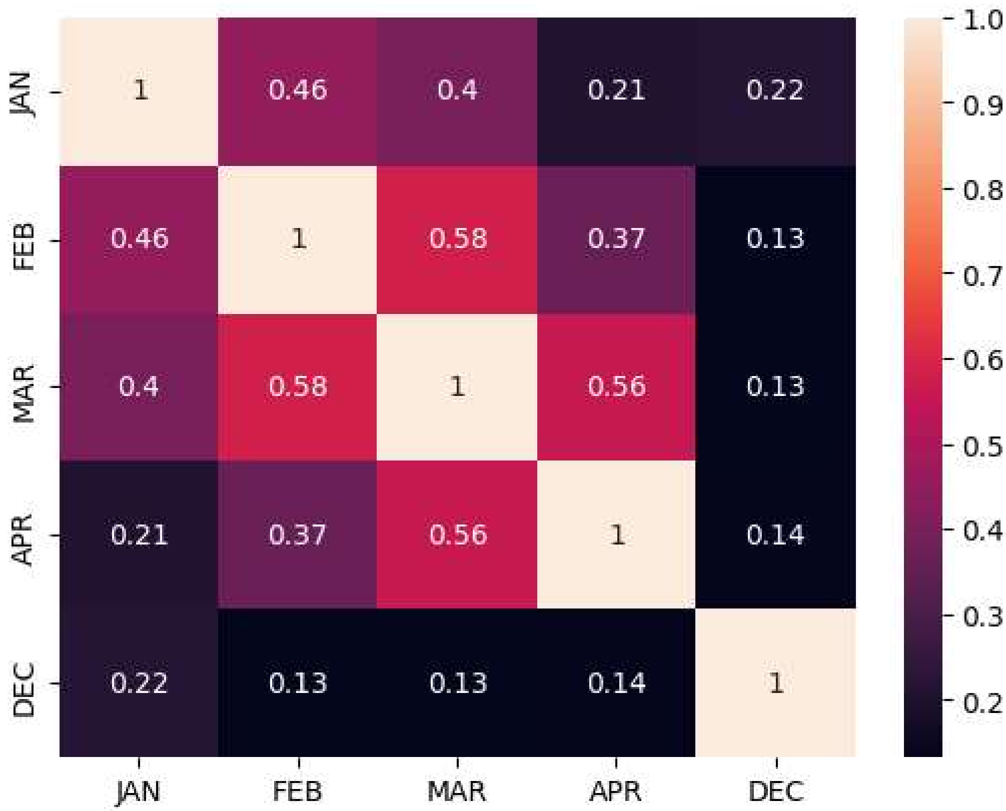
Out[17]:
```
Jun-Sep
434.3     4
334.8     4
573.8     4
613.3     4
1082.3    3
         ..
301.6     1
380.9     1
409.3     1
229.4     1
958.5     1
Name: count, Length: 3683, dtype: int64
```

In [18]:
```
df['Oct-Dec'].value_counts()
```

Out[18]:
```
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 [19]:
```
df=df[['JAN','FEB','MAR','APR','DEC']]
sns.heatmap(df.corr(),annot=True)
plt.show()
```



In [20]:
```
df.columns
```

Out[20]:
```
Index(['JAN', 'FEB', 'MAR', 'APR', 'DEC'], dtype='object')
```

In [21]:
```
x=df[["FEB"]]
y=df["JAN"]
```

In [22]:
```
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
```

In [23]:
```
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[23]:

	coefficient
FEB	0.442278

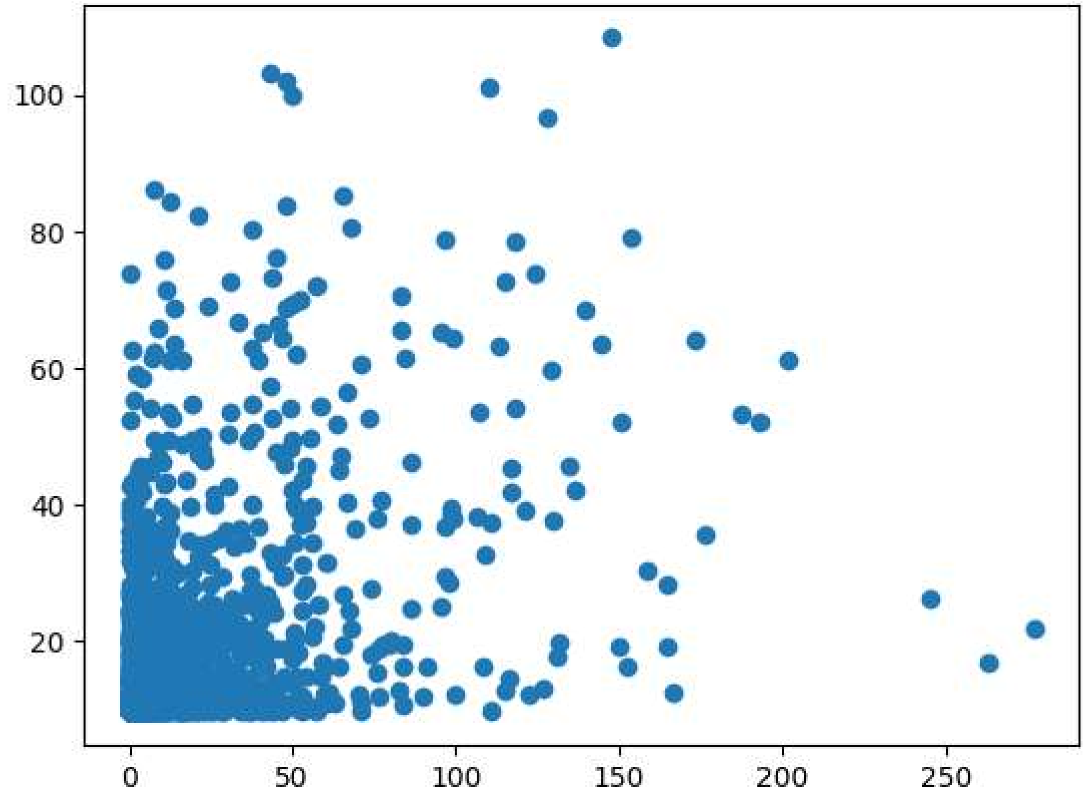
In [24]:
```
score=reg.score(X_test,y_test)
print(score)
```

```
0.1793580786264921
```

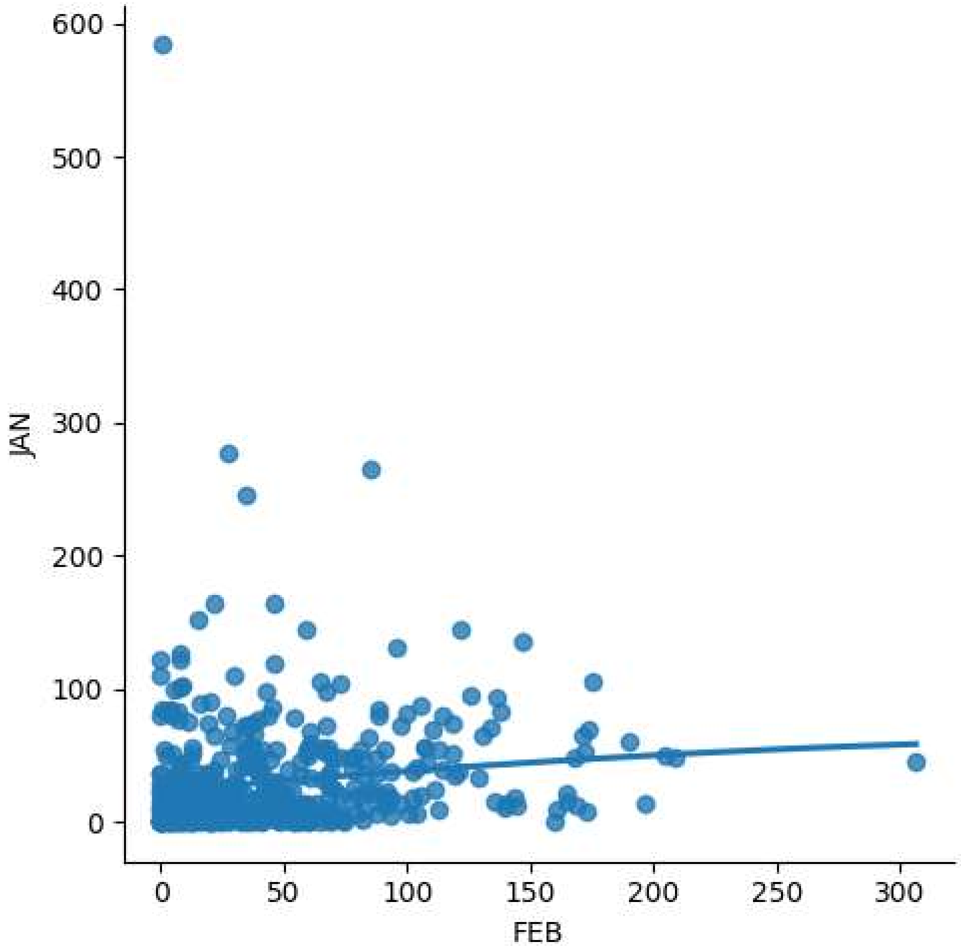
In [25]:
```
predictions=reg.predict(X_test)
```

In [26]:
```
plt.scatter(y_test,predictions)
```

Out[26]: <matplotlib.collections.PathCollection at 0x2f43cf33d30>



In [27]:
```
df500=df[:][:500]
sns.lmplot(x="FEB",y="JAN",order=2,ci=None,data=df500)
plt.show()
```

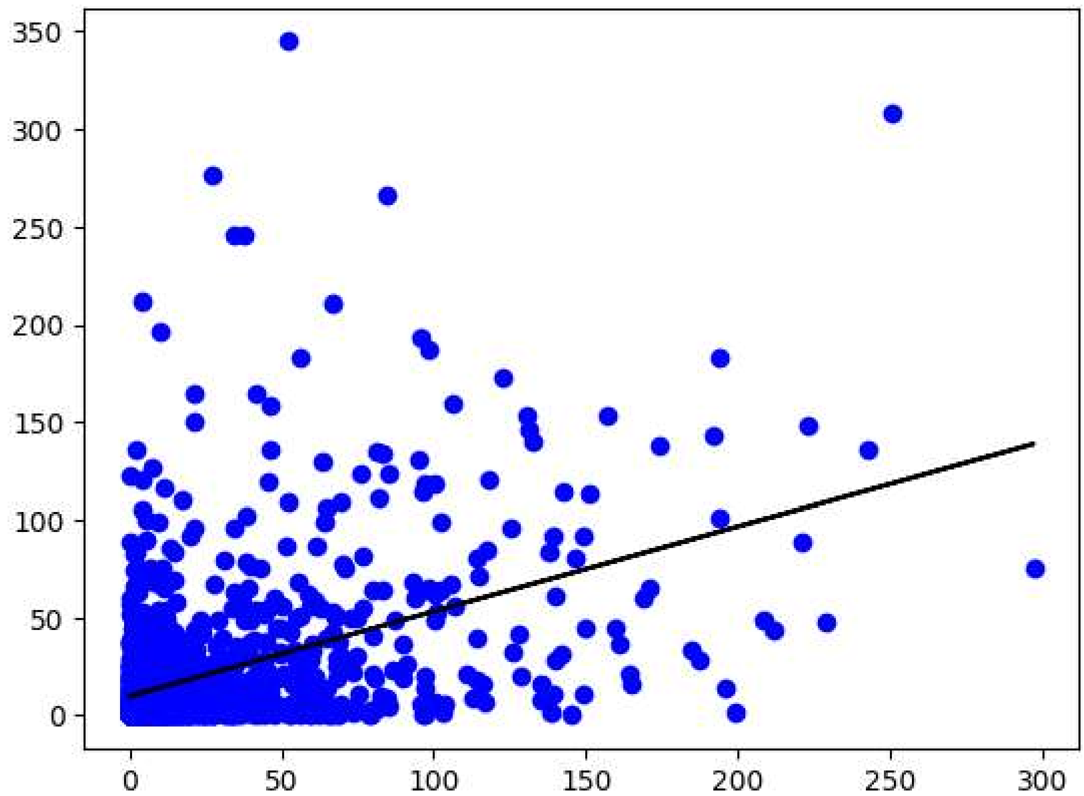


In [28]:
```
X_train,X_test,y_train,y_test=train_test_split(x,y,test_size=0.33)
reg.fit(X_train,y_train)
reg.fit(X_test,y_test)
```

Out[28]: 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 [29]:
```
y_pred=reg.predict(X_test)
plt.scatter(X_test,y_test,color='blue')
plt.plot(X_test,y_pred,color='black')
plt.show()
```



In [30]:
```
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: 0.21084431570038997
```

Ridge Regression

In [31]:
```
from sklearn.linear_model import Lasso,Ridge
from sklearn.preprocessing import StandardScaler
```

In [32]:
```
features= df.columns[0:5]
target= df.columns[-5]
```

In [33]:
```
x=np.array(df['JAN']).reshape(-1,1)
y=np.array(df['FEB']).reshape(-1,2)
```

In [34]:
```
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=
```

In [35]:
```
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)
```

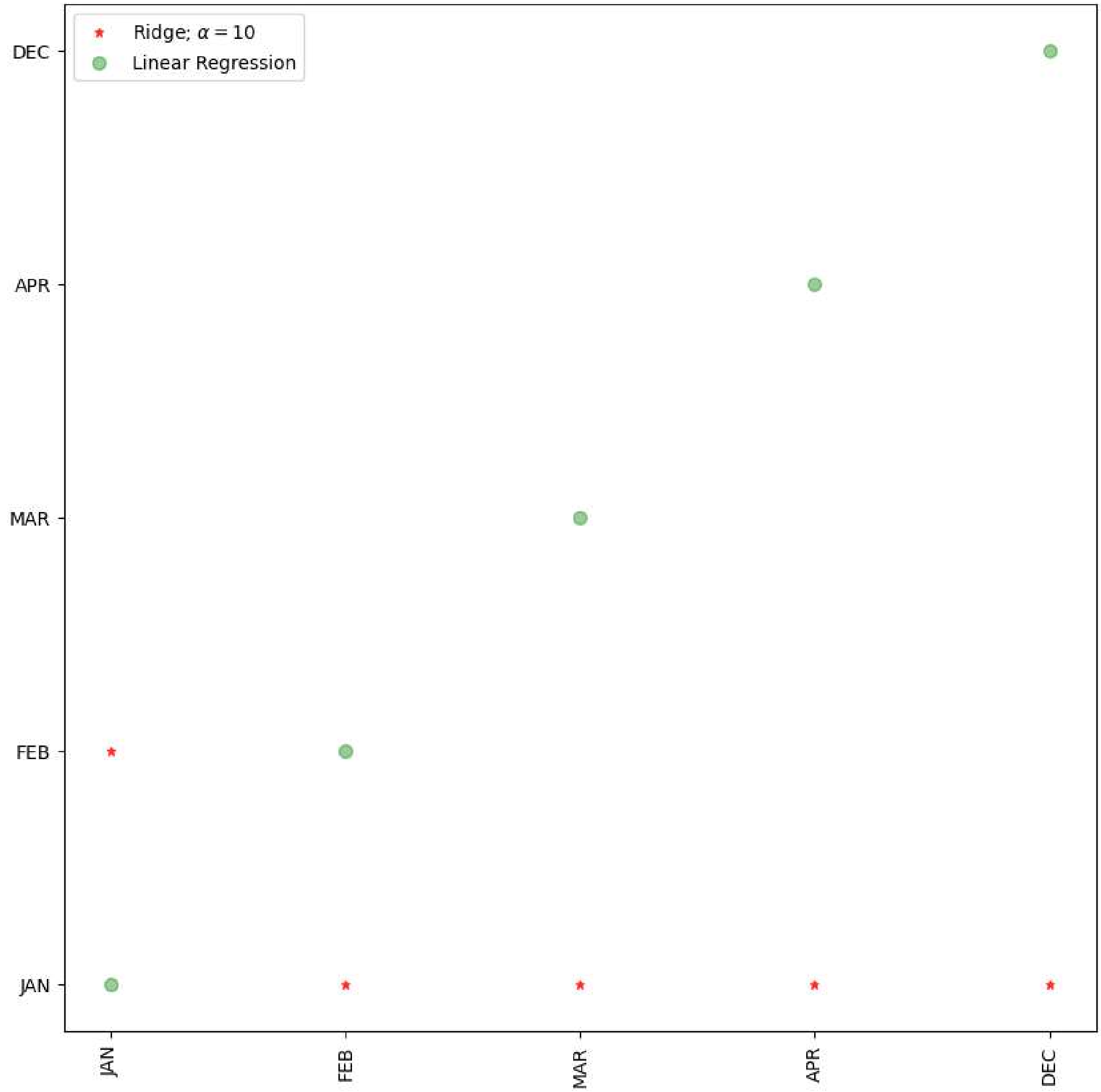
In [36]:
```
print("\n Ridge Model:\n")
print("the train score for ridge model is{}".format(train_score_ridge))
print("the test score for ridge model is{}".format(test_score_ridge))
```

```
 Ridge Model:

the train score for ridge model is0.9999999999874192
the test score for ridge model is0.99999999998833
```

In [37]:
```
lr=LinearRegression()
```

In [39]:
```
plt.figure(figsize= (10,10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker="*",markers
plt.plot(features,alpha=0.4,linestyle='none',marker='o',markersize=7,color="gr
plt.xticks(rotation = 90)
plt.legend()
plt.show()
```



Lasso regression

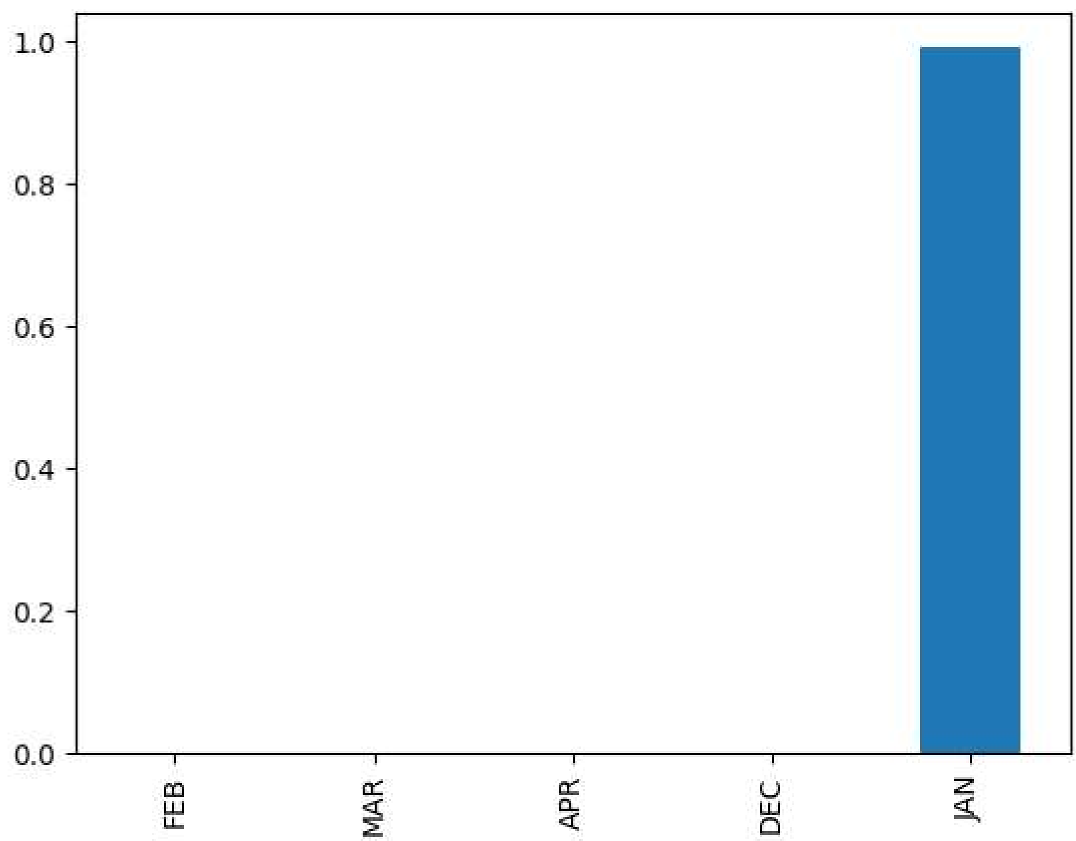
In [40]:
```
print("\n Lasso Model:\n")
lasso=Lasso(alpha=10)
lasso.fit(x_train,y_train)
train_score_ls=lasso.score(x_train,y_train)
test_score_ls=lasso.score(x_test,y_test)
print("The train score for ls model is {}".format(train_score_ls))
print("The test score for ls model is{}".format(test_score_ls))
```

```
 Lasso Model:

The train score for ls model is 0.9999207747038827
The test score for ls model is0.9999206791315255
```

In [41]:
```
pd.Series(lasso.coef_,features).sort_values(ascending=True).plot(kind="bar")
```

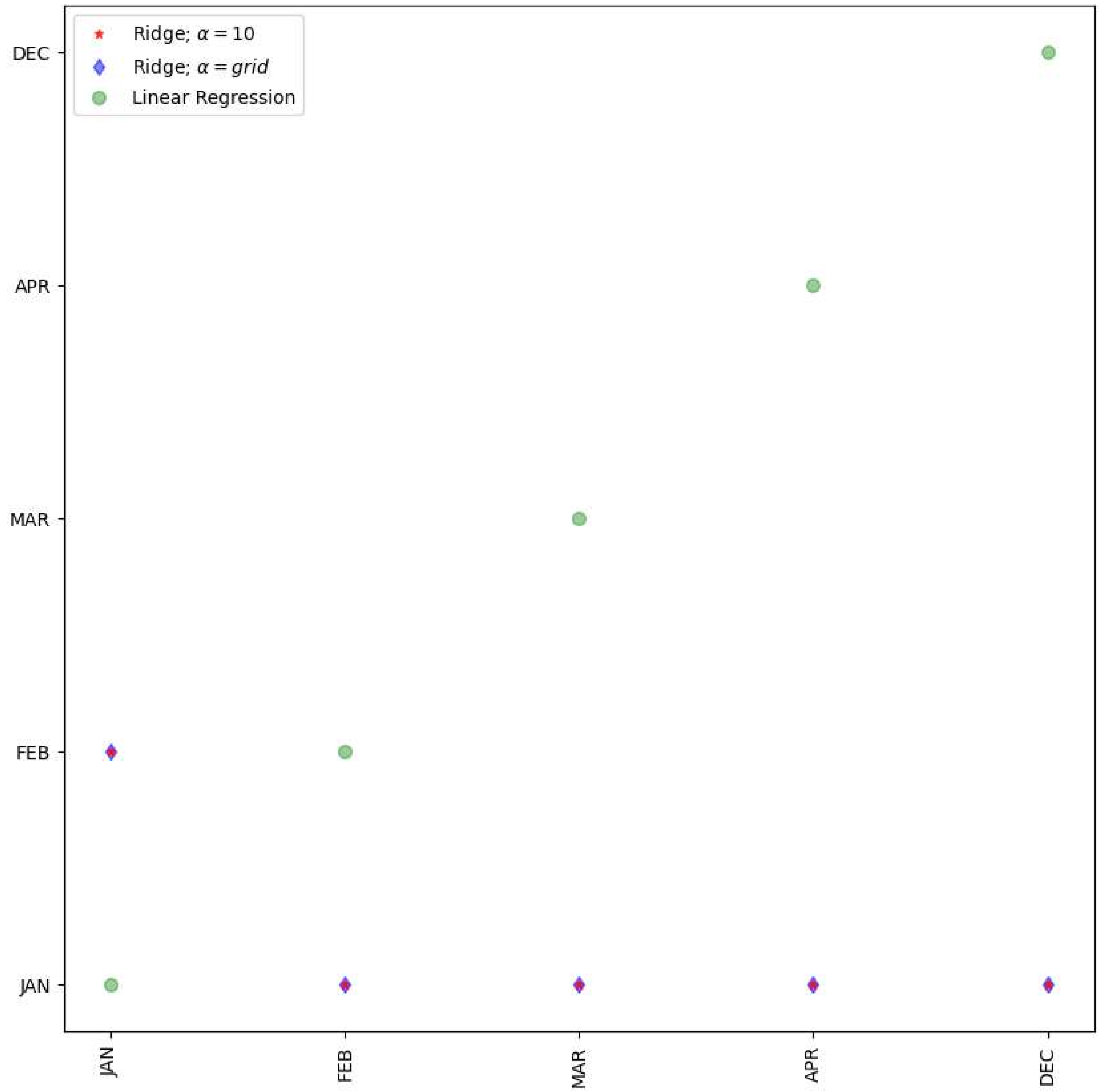
Out[41]: <Axes: >



In [42]:
```
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
print(lasso_cv.score(x_train,y_train))
print(lasso_cv.score(x_test,y_test))
```

```
0.9999999999999921
0.9999999999999921
```

In [44]:
```
plt.figure(figsize= (10,10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker="*",markers
plt.plot(lasso_cv.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,col
plt.plot(features,alpha=0.4,linestyle='none',marker='o',markersize=7,color="gr
plt.xticks(rotation = 90)
plt.legend()
plt.show()
```



ElasticNet Regression

In [45]:
```
from sklearn.linear_model import ElasticNet
regr=ElasticNet()
regr.fit(x,y)
print(regr.coef_)
print(regr.intercept_)
print(regr.score(x,y))
```

```
[9.99098574e-01 0.00000000e+00 3.02728910e-05 0.00000000e+00
 0.00000000e+00]
0.016258606966612632
0.9999992160905338
```

In [46]:
```
y_pred_elastic = regr.predict(x_train)
mean_squared_error=np.mean((y_pred_elastic - y_train)**2)
print(mean_squared_error)
```

```
0.0008816302333951303
```

Conclusion:

```
            For the given dataset,we have performed linear regression,ri
dge regression,lasso regression,elastic regression.
                 Linear Regression: -0.0001948420411366225
                 Ridge Regression: 0.9999999999897634
                 Lasso Regression: 0.9999999999999921
                 Elastic Net Regression: 0.9999992133148984
          Among all the models we observed that Elastic Net Regression g
ot highest accuracy.
```

In [ ]: