Mini-project 3:

PROBLEM STATEMENT:To predict the Rainfall based on various features of the dataset ¶

Import Libraries:

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

Out[2]:

	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
...	...	...	...	...	...	...	...	...	...	...	...	...	...
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
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
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
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
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

4116 rows × 19 columns

Data Cleaning & preprocessing:

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

Out[3]:

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

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

Out[4]:

	SUBDIVISION	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	
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	
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	
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	
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	
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	1

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

Out[5]:
```
(4116, 19)
```

In [6]:
```
df.columns
```

Out[6]:
```
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 [7]:
```
df.describe()
```

Out[7]:

	YEAR	JAN	FEB	MAR	APR	MAY	JUN	
count	4116.000000	4112.000000	4113.000000	4110.000000	4112.000000	4113.000000	4111.000000	410
mean	1958.218659	18.957320	21.805325	27.359197	43.127432	85.745417	230.234444	34
std	33.140898	33.585371	35.909488	46.959424	67.831168	123.234904	234.710758	26
min	1901.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.400000	
25%	1930.000000	0.600000	0.600000	1.000000	3.000000	8.600000	70.350000	17
50%	1958.000000	6.000000	6.700000	7.800000	15.700000	36.600000	138.700000	28
75%	1987.000000	22.200000	26.800000	31.300000	49.950000	97.200000	305.150000	41
max	2015.000000	583.700000	403.500000	605.600000	595.100000	1168.600000	1609.900000	236

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
```

Checking any null values.

In [9]:
```
df.isnull().sum()
```

Out[9]:
```
SUBDIVISION     0
YEAR            0
JAN             4
FEB             3
MAR             6
APR             4
MAY             3
JUN             5
JUL             7
AUG             4
SEP             6
OCT             7
NOV            11
DEC            10
ANNUAL         26
Jan-Feb         6
Mar-May         9
Jun-Sep        10
Oct-Dec        13
dtype: int64
```

In [10]:
```
df.dropna(inplace=True)
```

In [11]:
```
df.isnull().sum()
```

Out[11]:
```
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 [12]:
```
df["ANNUAL"].value_counts()
```

Out[12]:
```
ANNUAL
1024.6    4
770.3     4
790.5     4
1353.8    3
1138.2    3
         ..
419.8     1
428.9     1
527.8     1
322.9     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
        ...
66.5       1
80.9       1
26.4       1
102.5      1
69.3       1
Name: count, Length: 1211, 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
2.9      10
         ..
165.6     1
246.3     1
248.1     1
151.3     1
223.9     1
Name: count, Length: 2248, dtype: int64
```

In [15]:
```
df["Jun-Sep"].value_counts()
```

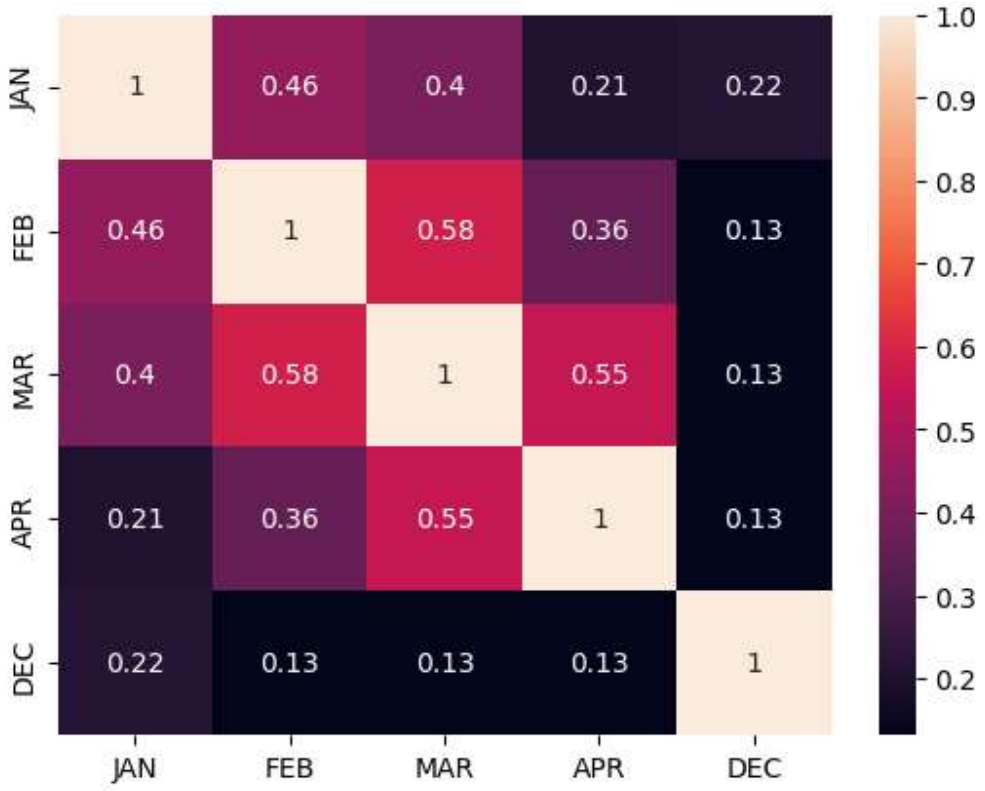
Out[15]:
```
Jun-Sep
334.8    4
434.3    4
573.8    4
613.3    4
403.9    3
        ..
897.7    1
301.6    1
380.9    1
409.3    1
958.5    1
Name: count, Length: 3670, dtype: int64
```

In [16]:
```
df["Oct-Dec"].value_counts()
```

Out[16]:
```
Oct-Dec
0.0      16
0.1      15
0.5      13
0.6      12
0.7      11
         ..
124.5     1
139.1     1
41.5      1
95.4      1
555.4     1
Name: count, Length: 2378, dtype: int64
```

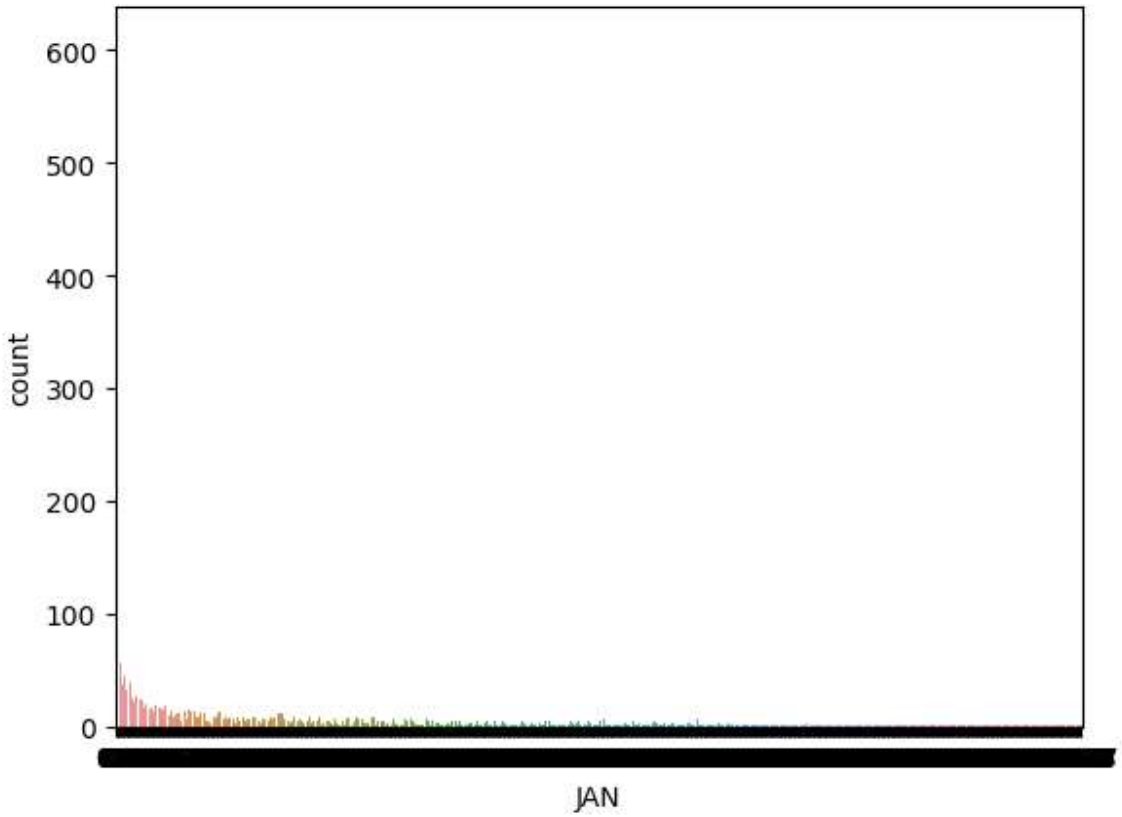
Data Visualisation:

In [17]:
```
df=df[['JAN','FEB','MAR','APR','DEC']]
sns.heatmap(df.corr(),annot=True)
plt.show()
```



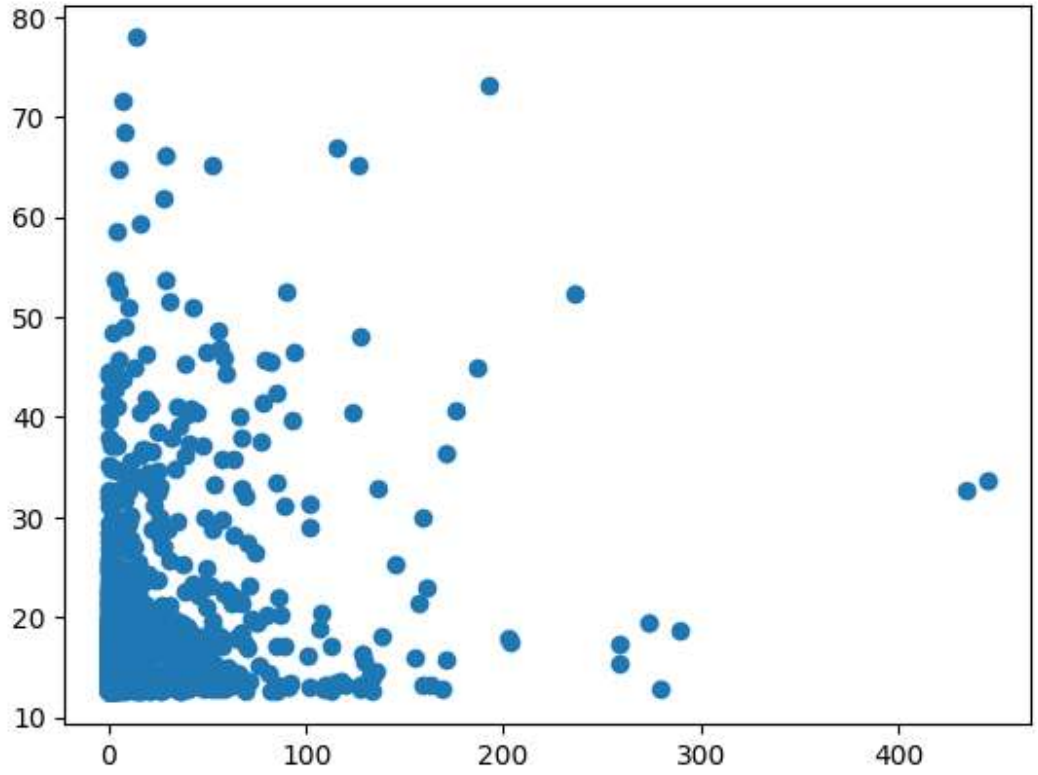
In [18]:
```
sns.countplot(x="JAN",data=df)
```

Out[18]:
```
<Axes: xlabel='JAN', ylabel='count'>
```



In [64]:
```
predictions=regr.predict(x_test)
plt.scatter(y_test,predictions)
```

Out[64]:
```
<matplotlib.collections.PathCollection at 0x17f336e5e50>
```



In [38]:
```
x=df[['MAR']]
y=df['JAN']
```

LINEAR REGRESSION:

In [39]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=101)
```

In [40]:
```
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)
```

Out[40]:
```
LinearRegression
LinearRegression()
```

In [41]:
```
print(regr.intercept_)
coeff_=pd.DataFrame(regr.coef_,x.columns,columns=['coefficient'])
coeff_
```

```
11.032992188591763
```

Out[41]:

	coefficient
MAR	0.306187

In [42]:
```
score=regr.score(x_test,y_test)
print(score)
```

```
0.11835017078035837
```

RIDGE:

In [43]:
```
from sklearn.linear_model import Lasso,RidgeCV,Ridge
from sklearn.preprocessing import StandardScaler
```

In [44]:
```
feature=df.columns[0:3]
target=df.columns[-1]
```

In [45]:
```
x=np.array(df['JAN']).reshape(-1,1)
y=np.array(df['FEB']).reshape(-1,2)
```

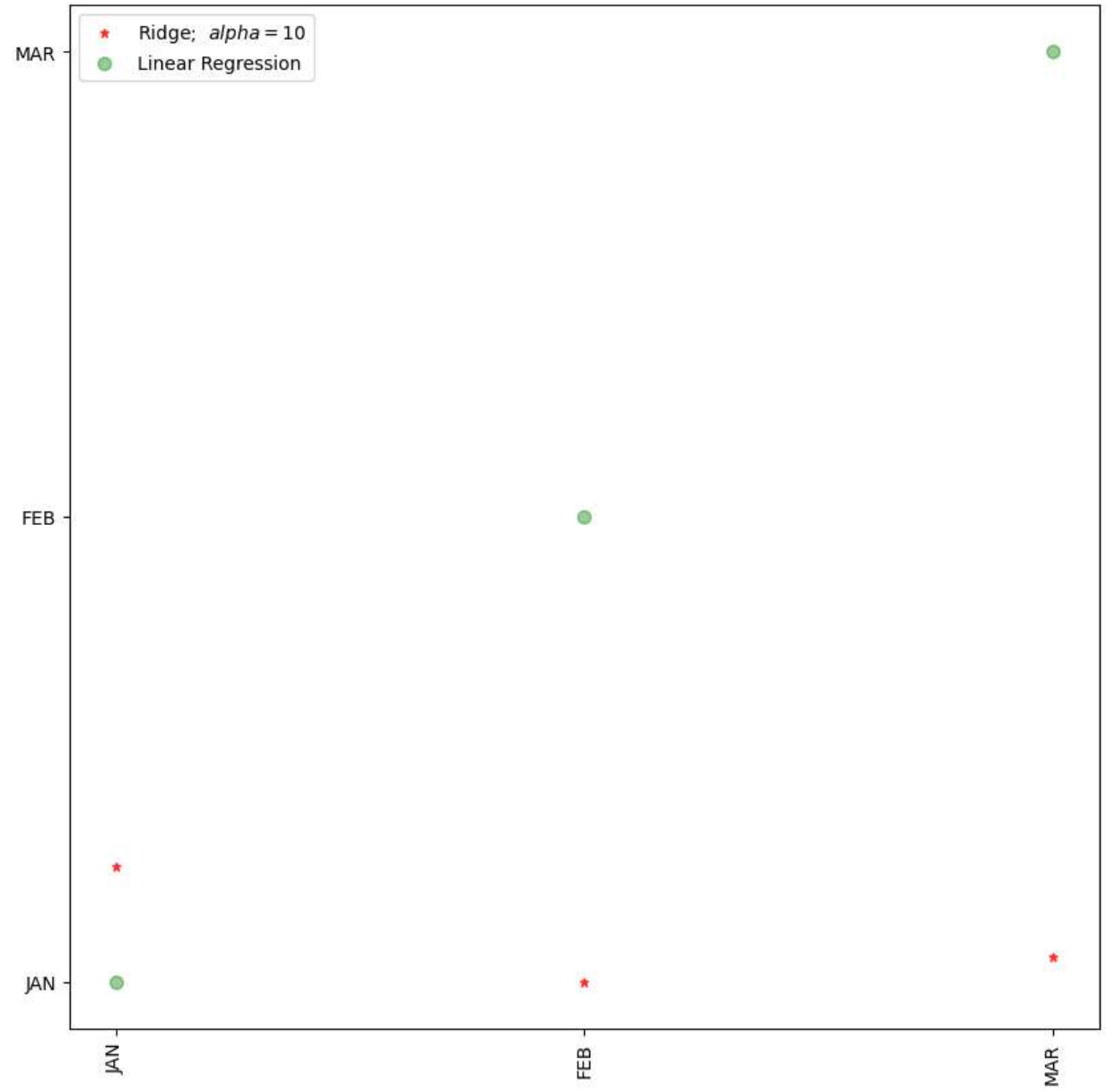
In [46]:
```
x= df[feature].values
y= df[target].values
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=17)
```

In [47]:
```
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("\n Ridge 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 0.05014945209599819
test score for ridge model is 0.05148515553862054
```

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



In [49]:
```
print(ridgeReg.score(x_test,y_test))
```

```
0.05148515553862054
```

LASSO:

In [50]:
```
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("\n Lasso Model \n")
print("train score for lasso model is {}".format(train_score_lasso))
print("test score for lasso model is {}".format(test_score_lasso))
```

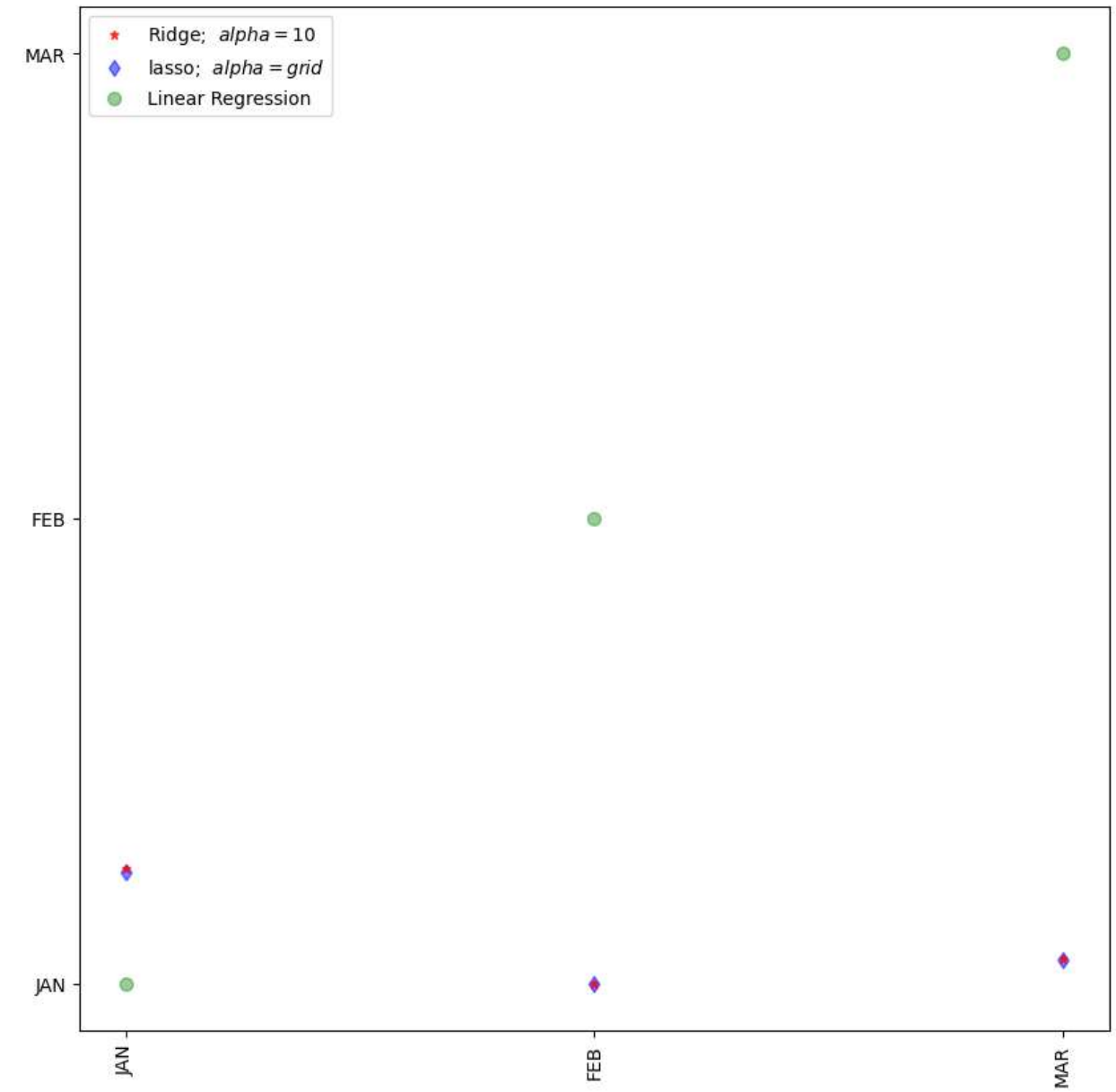
```
 Lasso Model

train score for lasso model is 0.05009768723610697
test score for lasso model is 0.05164202078447555
```

In [51]:
```
from sklearn.linear_model import LassoCV
lasso_cv=LassoCV(alphas=[0.001,0.001,0.01,0.1,1,10]).fit(x_train,y_train)
print("The train score for lasso model is {}".format(lasso_cv.score(x_train,y_train
print("The train score for lasso model is {}".format(lasso_cv.score(x_test,y_test))
```

```
The train score for lasso model is 0.05009768723610697
The train score for lasso model is 0.05164202078447555
```

In [52]:
```
igure(figsize=(10,10))
lot(feature,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5,color=
lot(lasso_cv.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color='blue',l
lot(feature,alpha=0.4,linestyle='none',marker='o',markersize=7,color='green',label='
ticks(rotation=90)
egend()
how()
```



In [53]:
```
print(lassoReg.score(x_test,y_test))
```

```
0.05164202078447555
```

ELASTIC NET:

In [54]:
```
from sklearn.linear_model import ElasticNet
```

In [55]:
```
regr=ElasticNet()
regr.fit(x,y)
print(regr.coef_)
print(regr.intercept_)
```

```
[0.24206174 0.01817624 0.04362447]
12.558042980256765
```

In [56]:
```
y_pred_elastic=regr.predict(x_train)
mean_squared_error=np.mean((y_pred_elastic-y_train)**2)
print(mean_squared_error)
```

```
1797.5088236249962
```

In [57]:
```
print(regr.score(x_test,y_test))
```

```
0.05298141054185812
```

CONCLUSION:Among on accuracy scores of all models that were implemented we can conclude that "Lasso Model" is the best model for the given dataset.