PROBLEM STATEMENT :-

TO PREDICT THE RAIN FALL BASED ON VARIOUS FEATURES OF THE DATASET

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
from sklearn.linear_model import Lasso
from sklearn.linear_model import Ridge
```

In [2]:

```
df=pd.read_csv(r"C:\Users\91955\Desktop\Data Analysis with Python\rainfall in india
df
```

Out[2]:

	SUBDIVISION	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	O
0	ANDAMAN & NICOBAR ISLANDS	1901	49.2	87.1	29.2	2.3	528.8	517.5	365.1	481.1	332.6	38
1	ANDAMAN & NICOBAR ISLANDS	1902	0.0	159.8	12.2	0.0	446.1	537.1	228.9	753.7	666.2	19
2	ANDAMAN & NICOBAR ISLANDS	1903	12.7	144.0	0.0	1.0	235.1	479.9	728.4	326.7	339.0	18
3	ANDAMAN & NICOBAR ISLANDS	1904	9.4	14.7	0.0	202.4	304.5	495.1	502.0	160.1	820.4	22
4	ANDAMAN & NICOBAR ISLANDS	1905	1.3	0.0	3.3	26.9	279.5	628.7	368.7	330.5	297.0	26
...	...	...	...	...	...	...	...	...	...	...	...	
4111	LAKSHADWEEP	2011	5.1	2.8	3.1	85.9	107.2	153.6	350.2	254.0	255.2	11
4112	LAKSHADWEEP	2012	19.2	0.1	1.6	76.8	21.2	327.0	231.5	381.2	179.8	14
4113	LAKSHADWEEP	2013	26.2	34.4	37.5	5.3	88.3	426.2	296.4	154.4	180.0	7
4114	LAKSHADWEEP	2014	53.2	16.1	4.4	14.9	57.4	244.1	116.1	466.1	132.2	16
4115	LAKSHADWEEP	2015	2.2	0.5	3.7	87.1	133.1	296.6	257.5	146.4	160.4	16

4116 rows × 19 columns

In [3]:

```
df.head()
```

Out[3]:

	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	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	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	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	3
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	

In [4]:

```
df.tail()
```

Out[4]:

	SUBDIVISION	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OC
4111	LAKSHADWEEP	2011	5.1	2.8	3.1	85.9	107.2	153.6	350.2	254.0	255.2	117.
4112	LAKSHADWEEP	2012	19.2	0.1	1.6	76.8	21.2	327.0	231.5	381.2	179.8	145.
4113	LAKSHADWEEP	2013	26.2	34.4	37.5	5.3	88.3	426.2	296.4	154.4	180.0	72.
4114	LAKSHADWEEP	2014	53.2	16.1	4.4	14.9	57.4	244.1	116.1	466.1	132.2	169.
4115	LAKSHADWEEP	2015	2.2	0.5	3.7	87.1	133.1	296.6	257.5	146.4	160.4	165.

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.isnull().sum()
```

Out[6]:

```
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 [7]:

```
df.fillna(value = 0,
          inplace = True)
```

In [8]:

```
df.isnull().any()
```

Out[8]:

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

In [9]:

```
df.isnull().sum()
```

Out[9]:

```
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 [10]:

```
1 df.describe()
```

Out[10]:

	YEAR	JAN	FEB	MAR	APR	MAY	
count	4116.000000	4116.000000	4116.000000	4116.000000	4116.000000	4116.000000	4116.0(
mean	1958.218659	18.938897	21.789431	27.319315	43.085520	85.682920	229.9
std	33.140898	33.574242	35.901220	46.936787	67.811512	123.211711	234.7(
min	1901.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0(
25%	1930.000000	0.600000	0.600000	1.000000	3.000000	8.600000	70.0(
50%	1958.000000	6.000000	6.700000	7.800000	15.600000	36.400000	138.6
75%	1987.000000	22.125000	26.800000	31.225000	49.825000	96.825000	304.9
max	2015.000000	583.700000	403.500000	605.600000	595.100000	1168.600000	1609.9(

In [11]:

```
1 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 [12]:

```
df.shape
```

Out[12]:

```
(4116, 19)
```

In [13]:

```
df.columns
```

Out[13]:

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

In [14]:

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

Out[14]:

```
Mar-May
0.0      38
0.1      13
0.3      11
8.3      11
2.7      10
         ..
249.5     1
148.8     1
191.9     1
207.0     1
223.9     1
Name: count, Length: 2262, dtype: int64
```

In [15]:

```
df['ANNUAL'].value_counts()
```

Out[15]:

```
ANNUAL
0.0       26
1024.6     4
790.5      4
770.3      4
1114.2     3
          ..
419.8      1
428.9      1
527.8      1
322.9      1
1642.9     1
Name: count, Length: 3713, dtype: int64
```

In [16]:

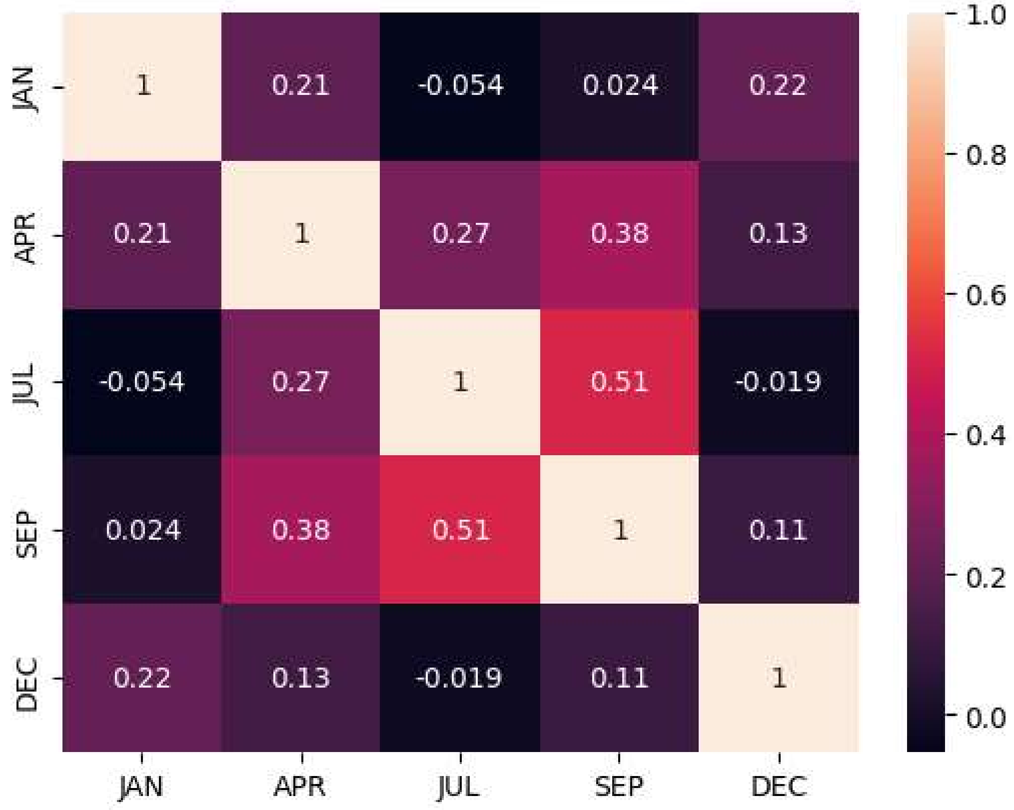
```
df['SUBDIVISION'].value_counts()
```

Out[16]:

```
SUBDIVISION
WEST MADHYA PRADESH                   115
EAST RAJASTHAN                        115
COASTAL KARNATAKA                     115
TAMIL NADU                            115
RAYALSEEMA                            115
TELANGANA                             115
COASTAL ANDHRA PRADESH                115
CHHATTISGARH                          115
VIDARBHA                              115
MATATHWADA                            115
MADHYA MAHARASHTRA                    115
KONKAN & GOA                          115
SAURASHTRA & KUTCH                    115
GUJARAT REGION                        115
EAST MADHYA PRADESH                   115
KERALA                                115
WEST RAJASTHAN                        115
SOUTH INTERIOR KARNATAKA              115
JAMMU & KASHMIR                       115
HIMACHAL PRADESH                      115
PUNJAB                                115
HARYANA DELHI & CHANDIGARH            115
UTTARAKHAND                           115
WEST UTTAR PRADESH                    115
EAST UTTAR PRADESH                    115
BIHAR                                 115
JHARKHAND                             115
ORISSA                                115
GANGETIC WEST BENGAL                  115
SUB HIMALAYAN WEST BENGAL & SIKKIM    115
NAGA MANI MIZO TRIPURA                115
ASSAM & MEGHALAYA                     115
NORTH INTERIOR KARNATAKA              115
LAKSHADWEEP                           114
ANDAMAN & NICOBAR ISLANDS             110
ARUNACHAL PRADESH                      97
Name: count, dtype: int64
```

In [17]:

```
df=df[['JAN','APR','JUL','SEP','DEC']]
sns.heatmap(df.corr(),annot=True)
plt.show()
```



In [18]:

```
x=df[["JUL"]]
y=df["DEC"]
```

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_
```

20.87220728751849

Out[20]:

	coefficient
JUL	-0.005482

In [21]:

```
score=reg.score(X_test,y_test)
print(score)
```

-0.002456456137450269

In [22]:

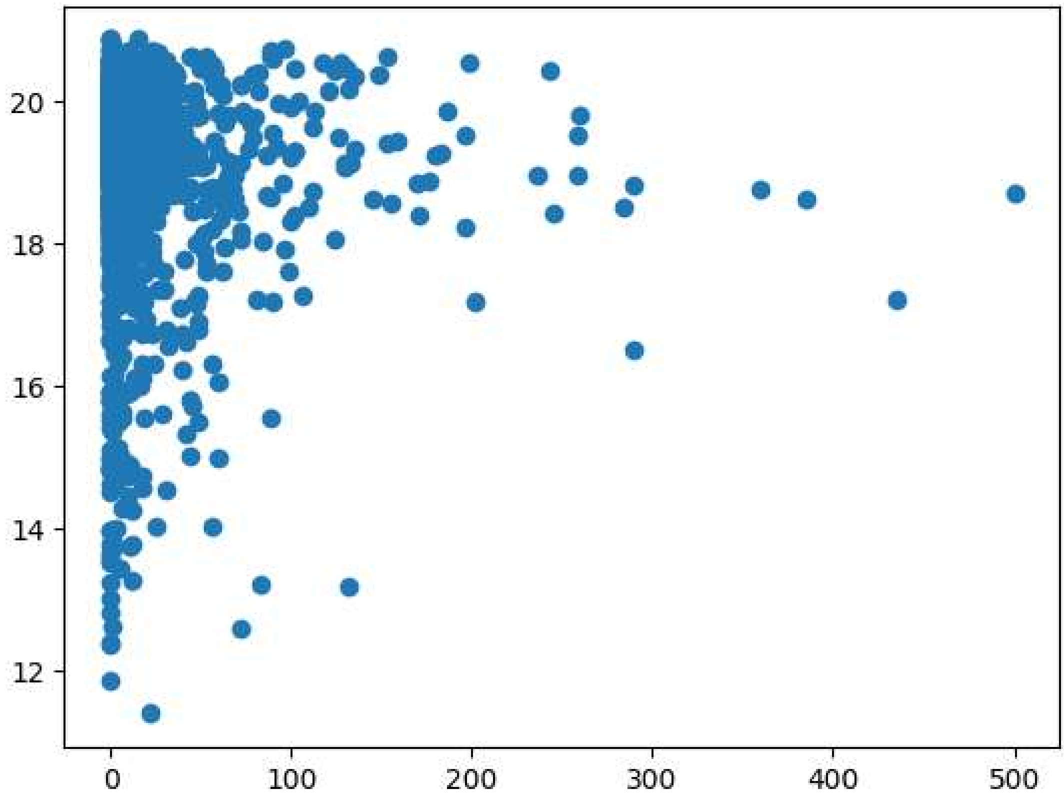
```
predictions=reg.predict(X_test)
```

In [23]:

```
plt.scatter(y_test,predictions)
```

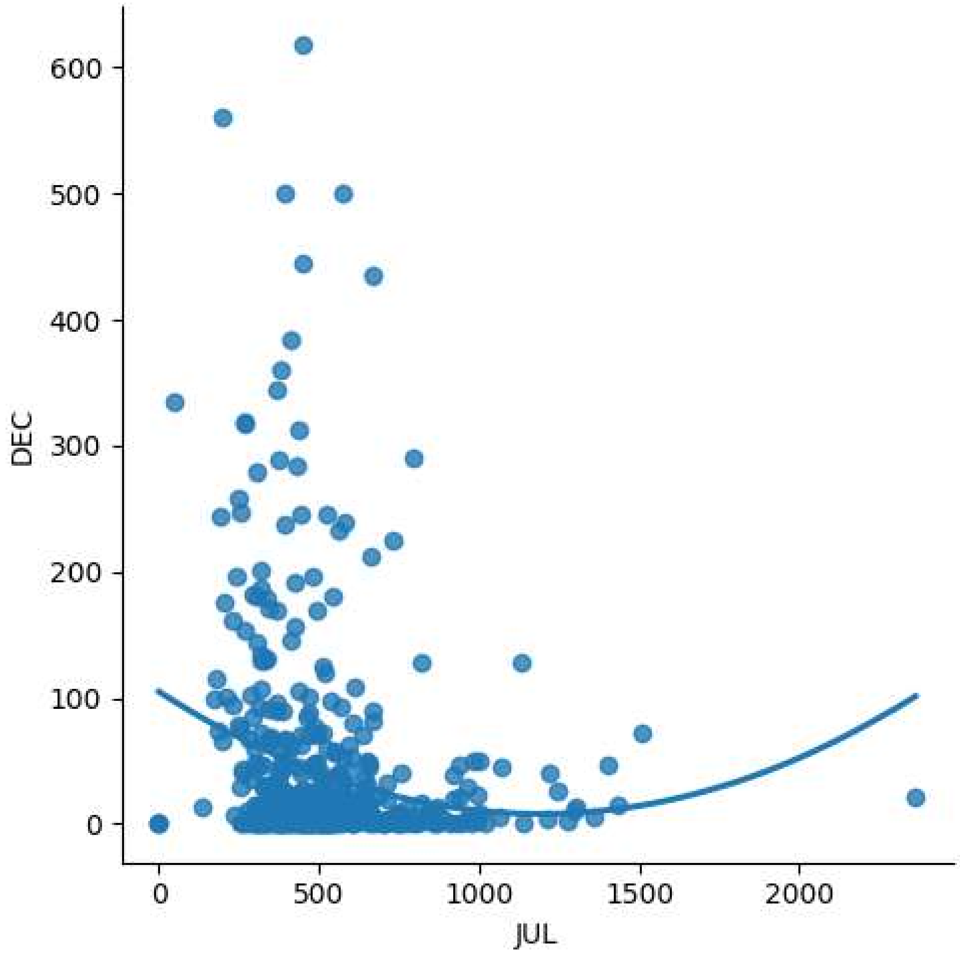
Out[23]:

<matplotlib.collections.PathCollection at 0x22bd1d53a60>



In [24]:

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



In [25]:

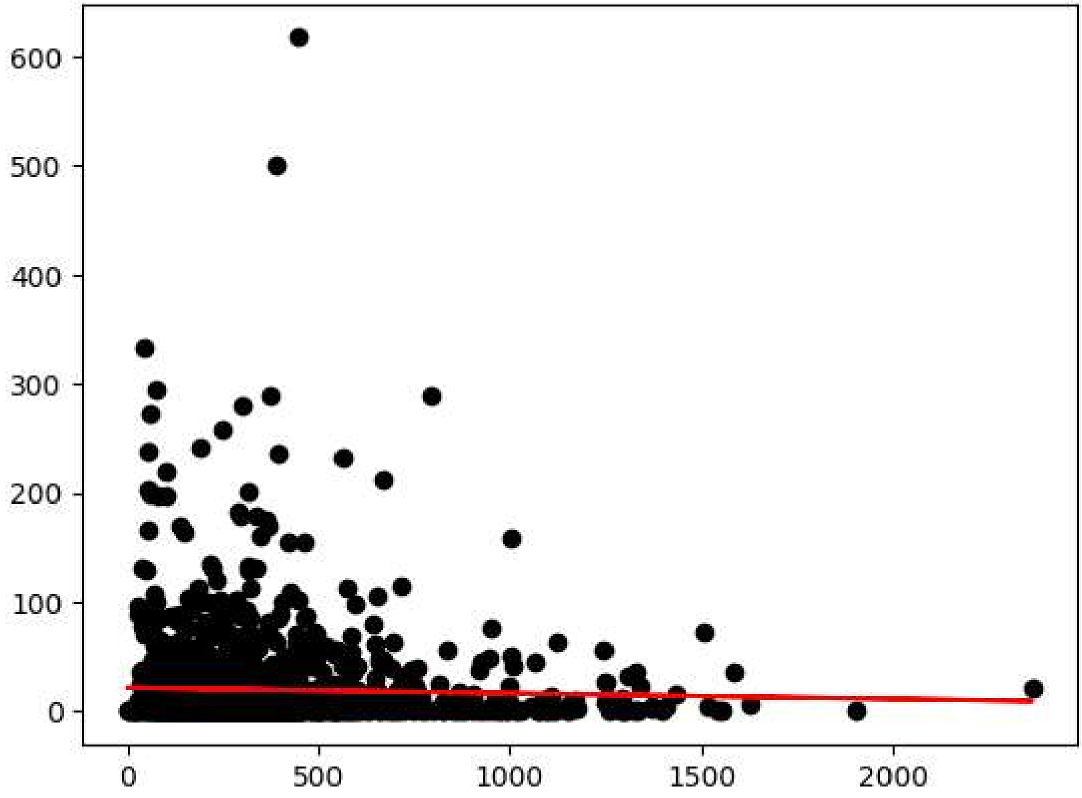
```
X_train,X_test,y_train,y_test=train_test_split(x,y,test_size=0.33)
reg.fit(X_train,y_train)
reg.fit(X_test,y_test)
```

Out[25]:

```
▾ LinearRegression
LinearRegression()
```

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: -0.0001948420411366225
```

Ridge Regression

In [28]:

```
from sklearn.linear_model import Lasso,Ridge
from sklearn.preprocessing import StandardScaler
```

In [29]:

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

In [30]:

```
x=np.array(df['JUL']).reshape(-1,1)
y=np.array(df['DEC']).reshape(-1,2)
```

In [31]:

```
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 [32]:

```
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 [33]:

```
print("Ridge Model:")
print("The train score for ridge model is {}".format(train_score_ridge))
print("the test score for ridge model is {}".format(test_score_ridge))
```

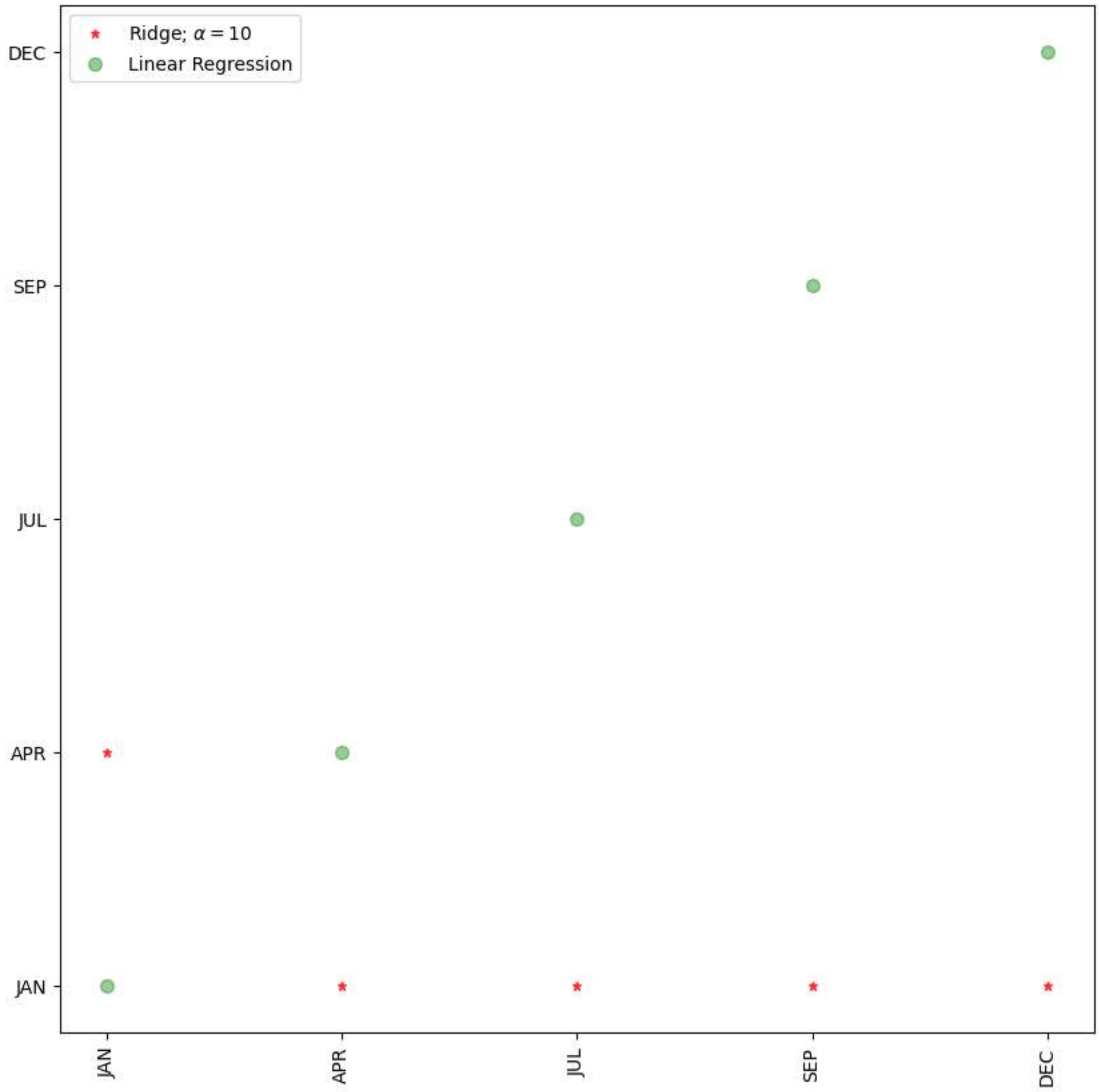
```
Ridge Model:
The train score for ridge model is 0.9999999999895616
the test score for ridge model is 0.9999999999897634
```

In [34]:

```
lr=LinearRegression()
```

In [35]:

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



Lasso Regression

In [36]:

```
print("Lasso Model:")
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))
```

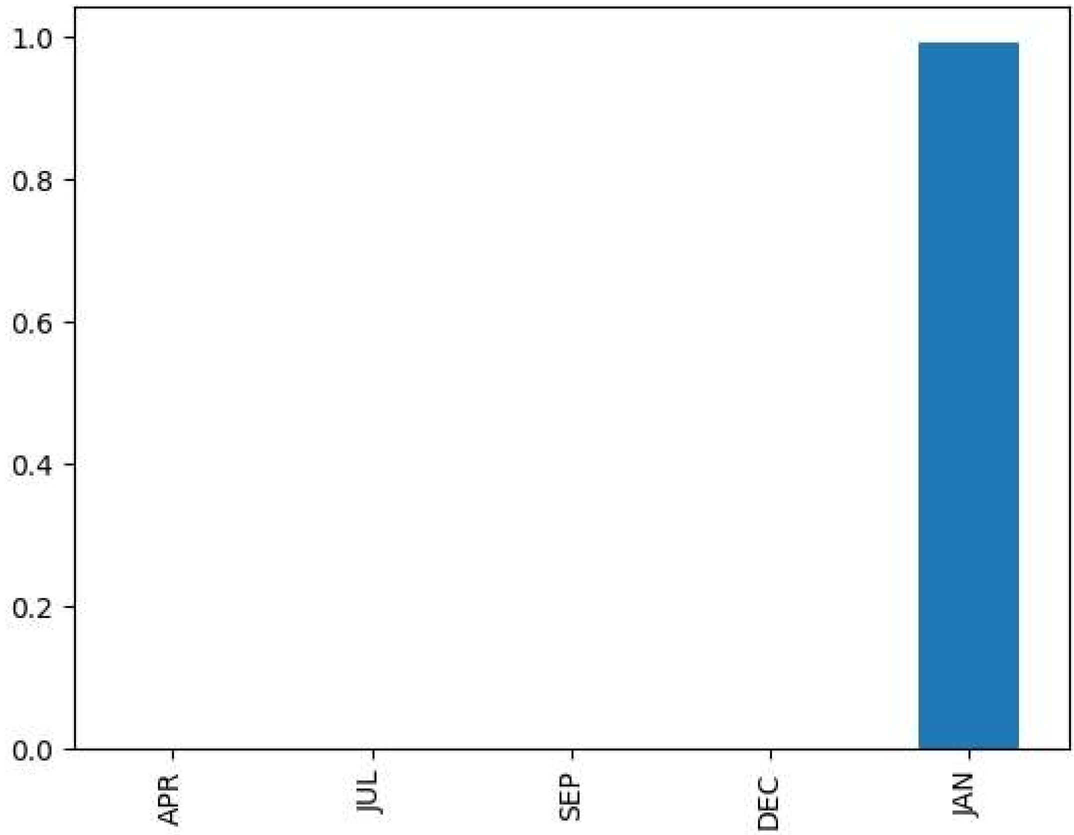
```
Lasso Model:
The train score for ls model is 0.9999207503194595
The test score for ls model is0.9999206588980594
```

In [37]:

```
pd.Series(lasso.coef_,features).sort_values(ascending=True).plot(kind="bar")
```

Out[37]:

<Axes: >



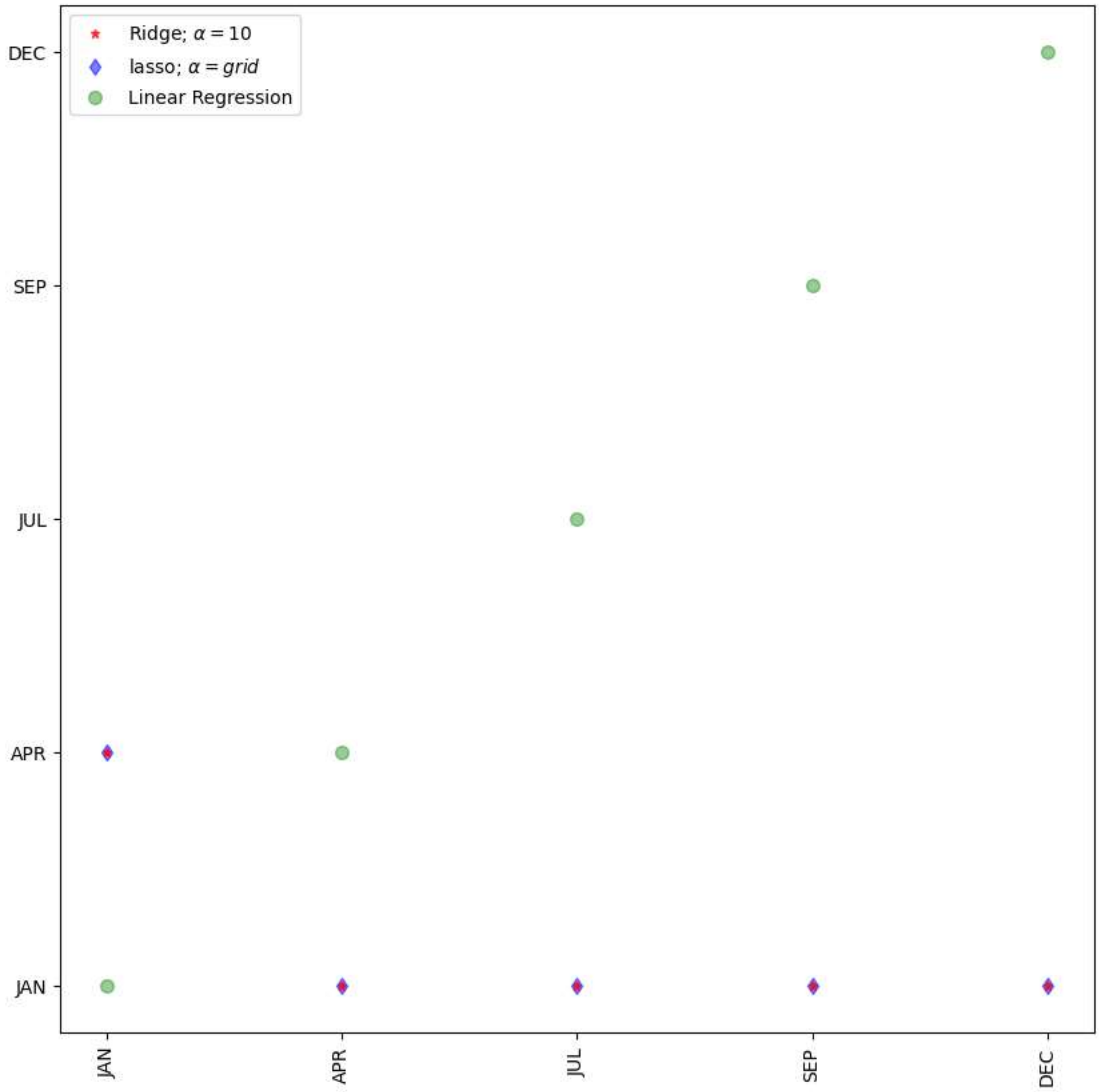
In [38]:

```
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_trair
print(lasso_cv.score(x_train,y_train))
print(lasso_cv.score(x_test,y_test))
```

```
0.9999999999999921
0.9999999999999921
```

In [39]:

```
#plot size
plt.figure(figsize = (10, 10))
#add plot for ridge regression
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5,
#add plot for lasso regression
plt.plot(lasso_cv.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color='bl
#add plot for linear model
plt.plot(features,alpha=0.4,linestyle='none',marker='o',markersize=7,color='green',l
plt.xticks(rotation = 90)
plt.legend()
plt.show()
```



Elastic Net Regression

In [40]:

```
from sklearn.linear_model import ElasticNet
er=ElasticNet()
er.fit(x,y)
print(er.coef_)
print(er.intercept_)
print(er.score(x,y))
```

```
[ 0.99911305  0.         -0.          0.          0.        ]
0.016797905920284496
0.9999992133148984
```

In [41]:

```
y_pred_elastic = er.predict(x_train)
mean_squared_error=np.mean((y_pred_elastic - y_train)**2)
print(mean_squared_error)
```

```
0.0008837872213758883
```

Conclusion:For the given dataset,we have performed linear regression,ridge 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 got highest accuracy.