PROBLEM STATEMENT:HOW TO PREDICT THE BEST ACCURACY BASED ON THE GIVEN DATASET

1.Data Collection

In [1]:
```
#Importing libraries
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
```

In [3]:
```
df=pd.read_csv(r"C:\Users\pavan\Downloads\rainfall in india 1901-2015.csv")
df
```

Out[3]:

	SUBDIVISION	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC	ANNUAL	Jan-Feb	Mar-May	Jun-Sep	Oc De
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	980
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	716
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	690
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	571
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	630
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
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	316
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	167
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	177
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	290
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	555

4116 rows × 19 columns

2.Data Cleaning & Preprocessing

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

Out[4]:

	SUBDIVISION	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC	ANNUAL	Jan-Feb	Mar-May	Jun-Sep	Oct-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	3373.2	136.3	560.3	1696.3	980.3
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	2185.9	716.7
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	690.6
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	571.0
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	630.8

In [5]:
```
df.tail()
```

Out[5]:

	SUBDIVISION	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC	ANNUAL	Jan-Feb	Mar-May	Jun-Sep	Oct-Dec
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	316.6
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	167.1
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	177.6
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	958.5	290.5
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	555.4

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	4109.0
mean	1958.218659	18.957320	21.805325	27.359197	43.127432	85.745417	230.234444	347.214334	290.263497	197.361922	95.5
std	33.140898	33.585371	35.909488	46.959424	67.831168	123.234904	234.710758	269.539667	188.770477	135.408345	99.5
min	1901.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.400000	0.000000	0.000000	0.100000	0.0
25%	1930.000000	0.600000	0.600000	1.000000	3.000000	8.600000	70.350000	175.600000	155.975000	100.525000	14.6
50%	1958.000000	6.000000	6.700000	7.800000	15.700000	36.600000	138.700000	284.800000	259.400000	173.900000	65.2
75%	1987.000000	22.200000	26.800000	31.300000	49.950000	97.200000	305.150000	418.400000	377.800000	265.800000	148.4
max	2015.000000	583.700000	403.500000	605.600000	595.100000	1168.600000	1609.900000	2362.800000	1664.600000	1222.000000	948.3

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
```

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.isna().any()
```

Out[10]:
```
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
```

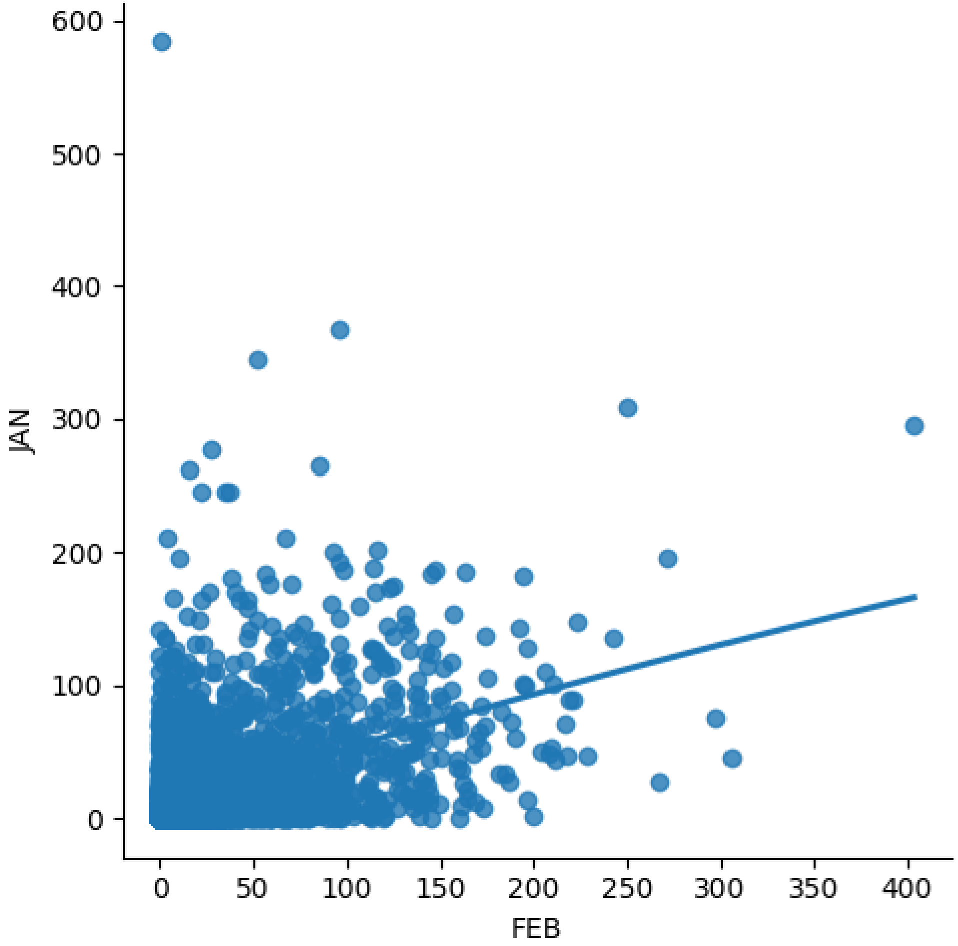
In [13]:
```
df.fillna(method="ffill",inplace=True)
```

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

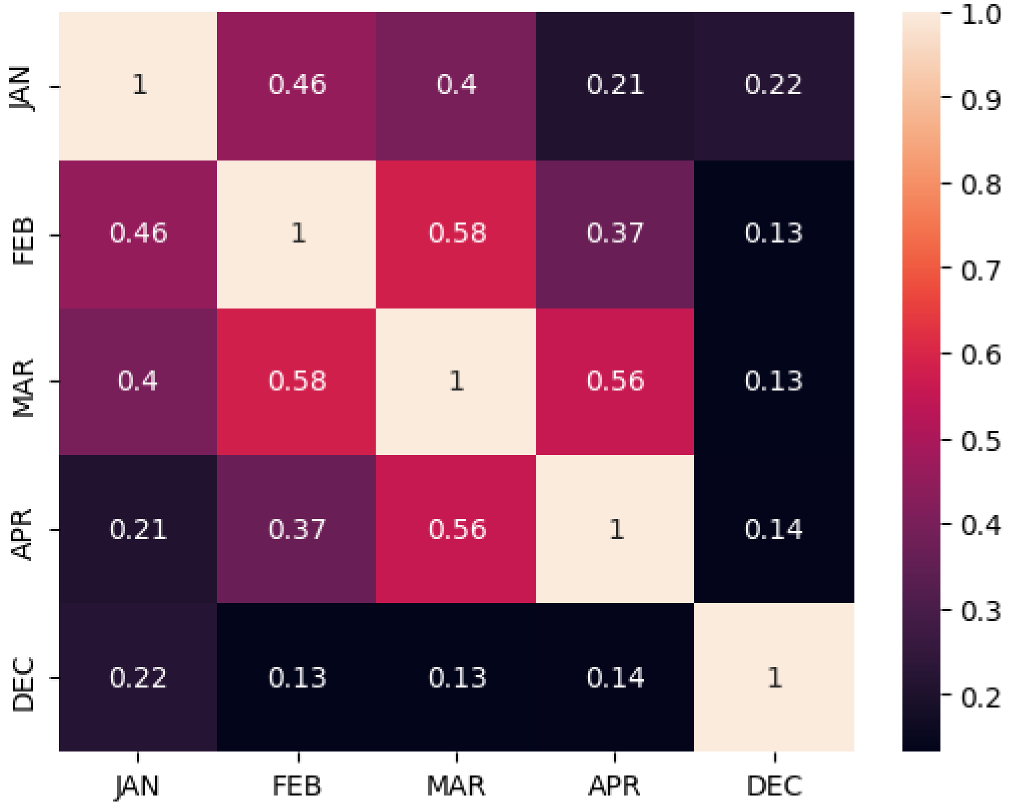
Out[12]:
```
YEAR
1963    36
2002    36
1976    36
1975    36
1974    36
        ..
1915    35
1918    35
1954    35
1955    35
1909    34
Name: count, Length: 115, dtype: int64
```

Data Analysis

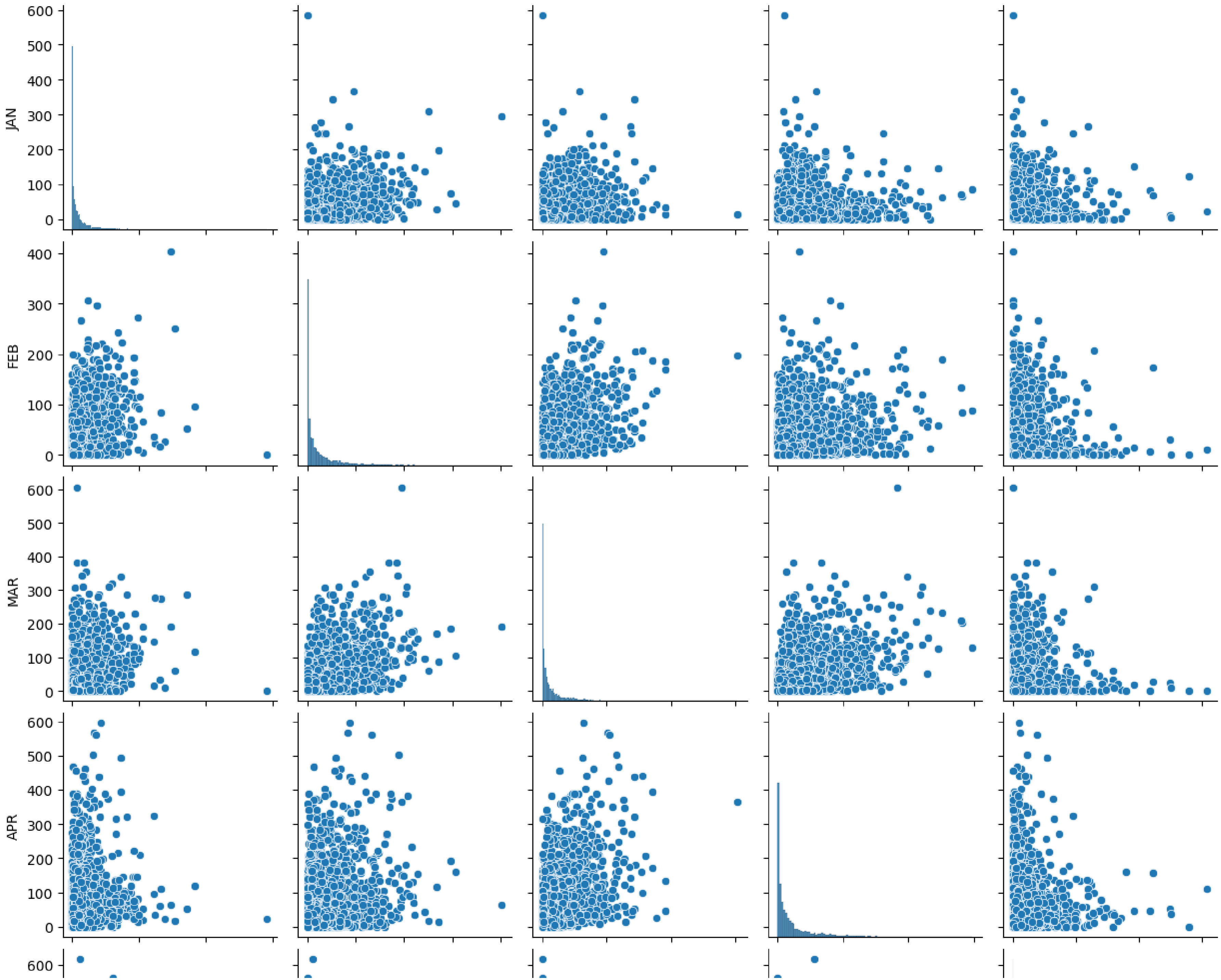
In [14]:
```
sns.lmplot(x='FEB',y='JAN',order=2,data=df,ci=None)
plt.show()
```



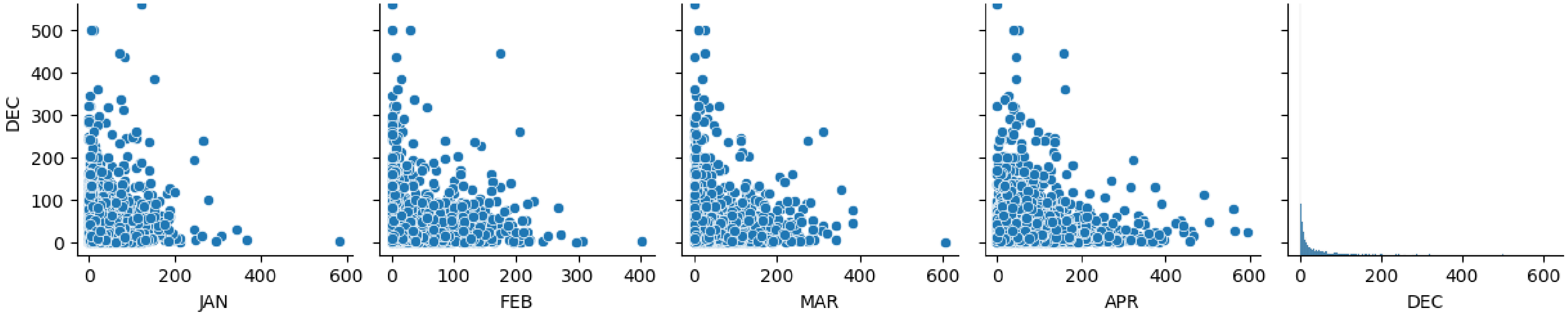
In [15]:
```
df=df[['JAN','FEB','MAR','APR','DEC']]
sns.heatmap(df.corr(),annot=True)
plt.show()
```



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



JAN
FEB
MAR
APR



Training the model

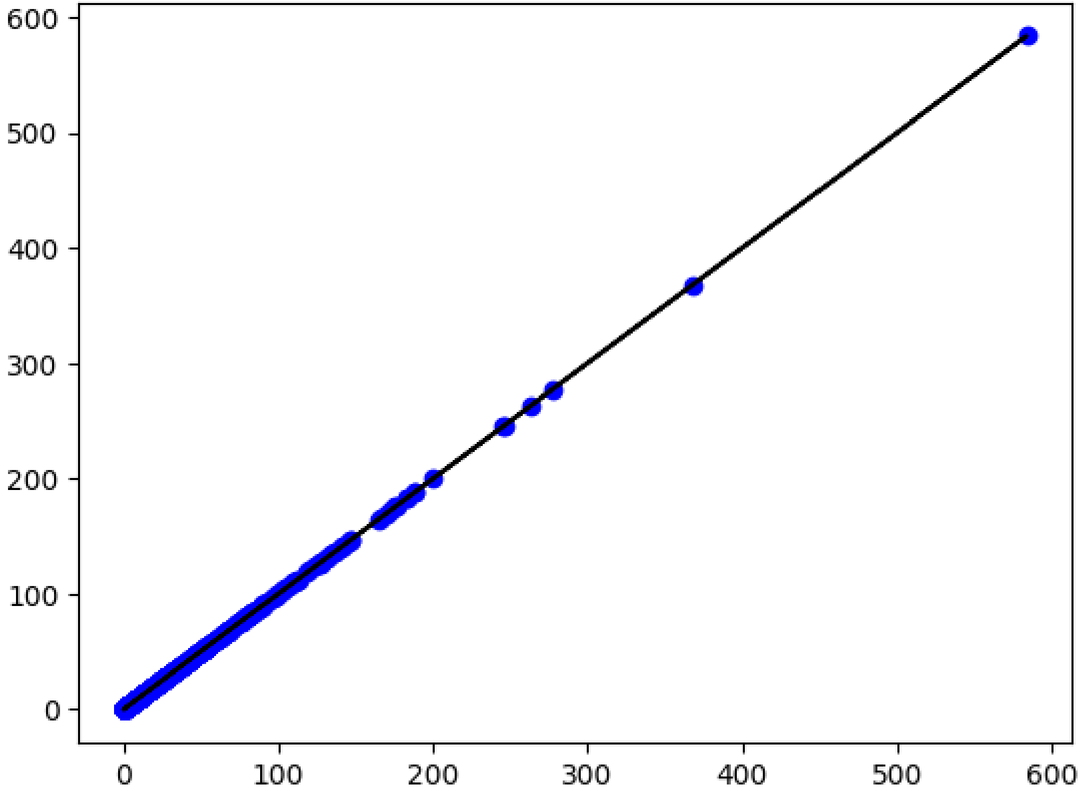
In [17]:
```
x=np.array(df['FEB']).reshape(-1,1)
y=x=np.array(df['JAN']).reshape(-1,1)
```

In [18]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30)
```

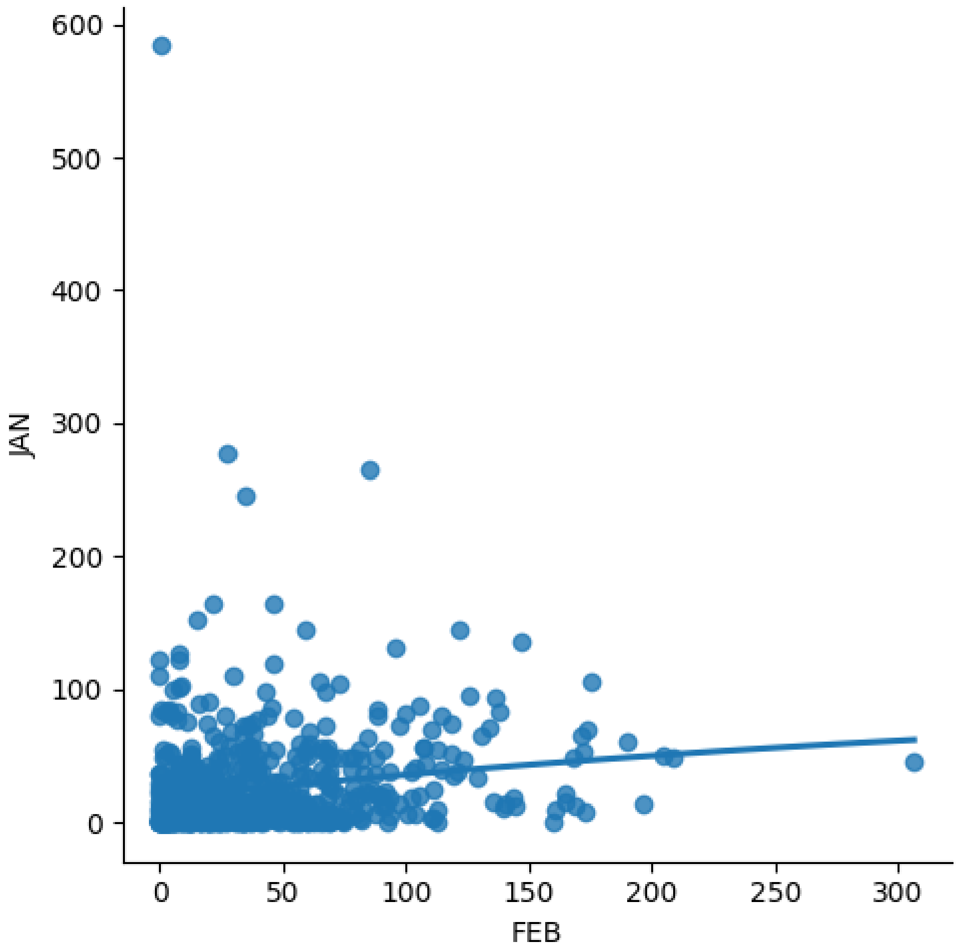
In [19]:
```
lin=LinearRegression()
lin.fit(x_train,y_train)
print(lin.score(x_train,y_train))
```

```
1.0
```

In [20]:
```
y_pred=lin.predict(x_test)
plt.scatter(x_test,y_test,color='blue')
plt.plot(x_test,y_pred,color='black')
plt.show()
```



In [21]:
```
df700=df[:][:700]
sns.lmplot(x='FEB',y='JAN',order=2,ci=None,data=df700)
plt.show()
```



In [22]:
```
df700.fillna(method='ffill',inplace=True)
```

In [23]:
```
x=np.array(df700['FEB']).reshape(-1,1)
y=x=np.array(df700['JAN']).reshape(-1,1)
```

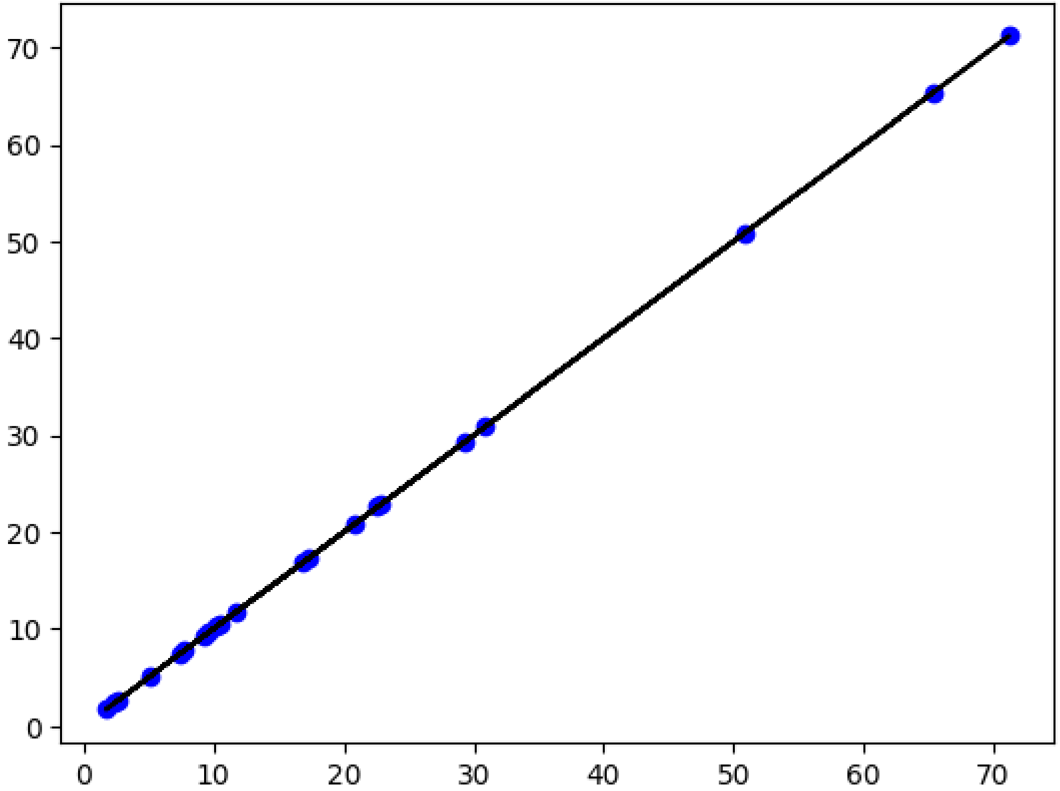
In [24]:
```
df700.dropna(inplace=True)
```

In [25]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.03)
```

In [26]:
```
lr=LinearRegression()
lr.fit(x_train,y_train)
print(lr.score(x_test,y_test))
```

```
1.0
```

In [27]:
```
y_pred=lr.predict(x_test)
plt.scatter(x_test,y_test,color='b')
plt.plot(x_test,y_pred,color='k')
plt.show()
```



In [28]:
```


from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
```

In [29]:
```
lr=LinearRegression()
lr.fit(x_train,y_train)
y_pred=lr.predict(x_test)
r2=r2_score(y_test,y_pred)
print("R2 score:",r2)
```

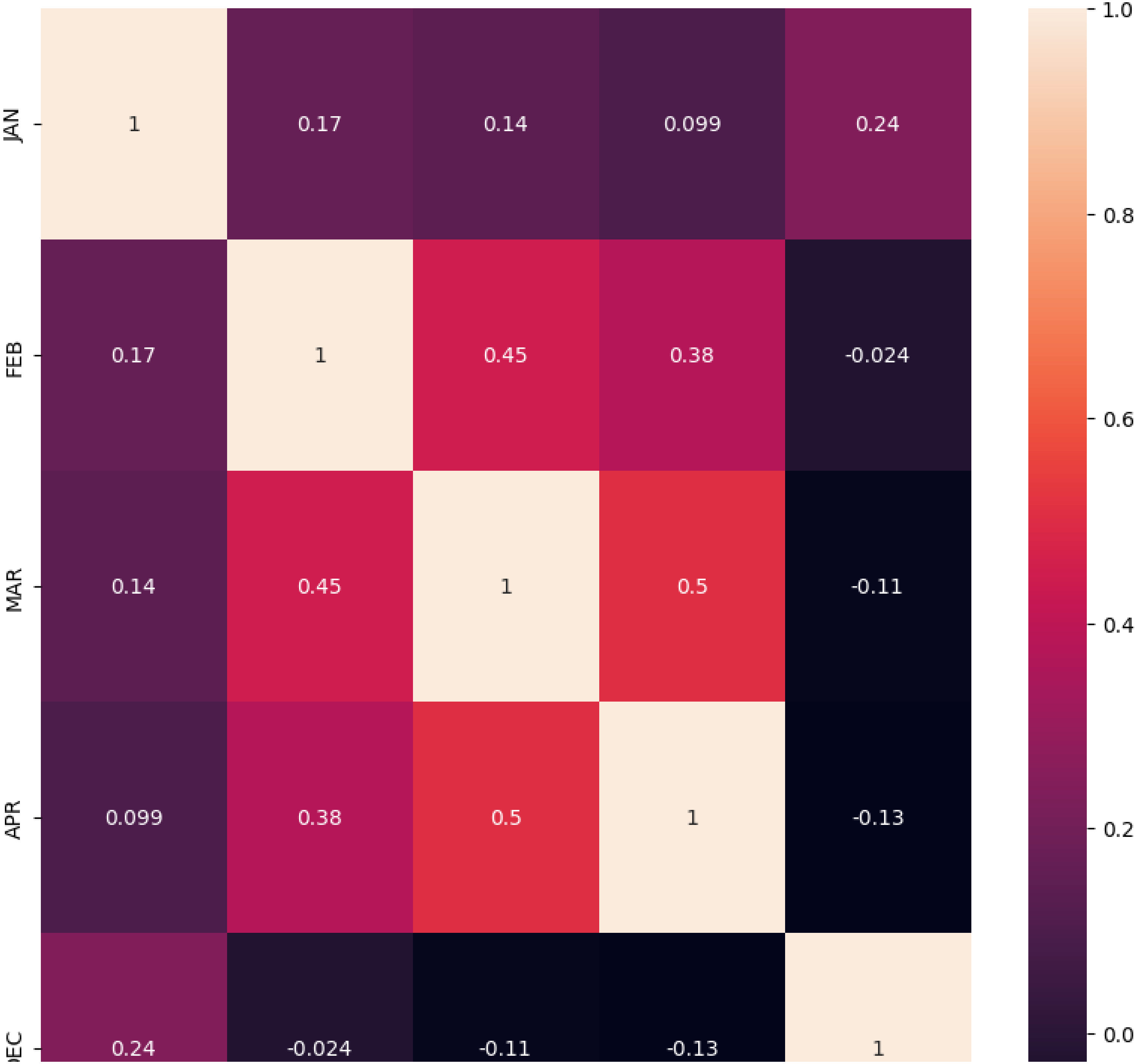
```
R2 score: 1.0
```

The Accuracy is 1.0

Ridge Regression

In [30]:
```
#Importing Libraries
from sklearn.linear_model import Lasso,Ridge
from sklearn.preprocessing import StandardScaler
```

In [31]:
```
plt.figure(figsize=(10,10))
sns.heatmap(df700.corr(),annot=True)
plt.show()
```


JAN
FEB
MAR
APR
DEC
1
0.17
0.14
0.099
0.24
0.17
1
0.45
0.38
-0.024
0.14
0.45
1
0.5
-0.11
0.099
0.38
0.5
1
-0.13
0.24
-0.024
-0.11
-0.13
1
1.0
0.8
0.6
0.4
0.2
0.0

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

In [33]:
```
x=df[features].values
y=df[target].values
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30,random_state=1)
print("The dimension of X_train is {}".format(x_train.shape))
print("The dimension of X_test is {}".format(x_test.shape))
```

```
The dimension of X_train is (2881, 5)
The dimension of X_test is (1235, 5)
```

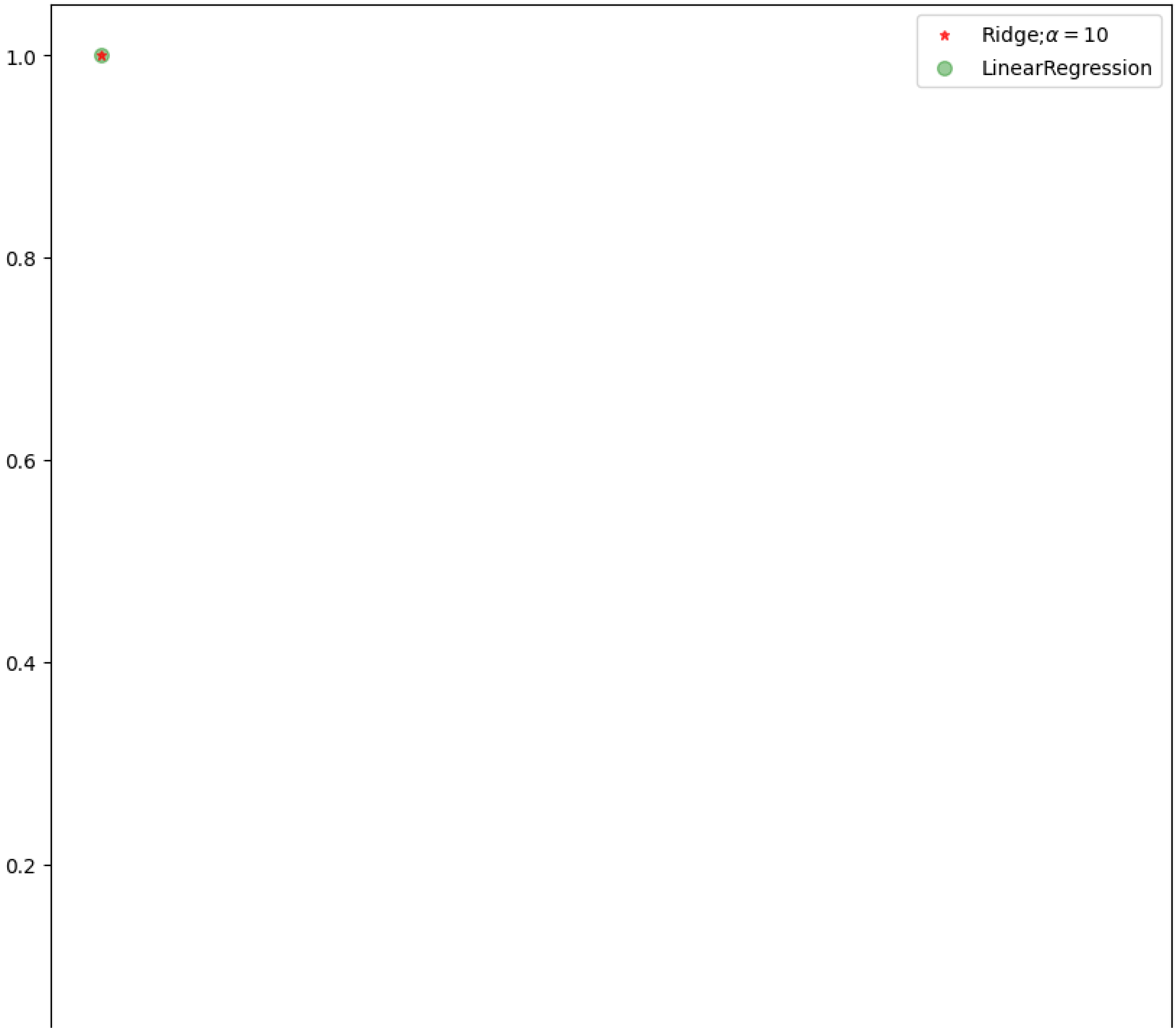
In [34]:
```
lr = LinearRegression()
#Fit model
lr.fit(x_train, y_train)
#predict
actual = y_test
train_score_lr = lr.score(x_train, y_train)
test_score_lr = lr.score(x_test, y_test)
print("\nLinear Regression Model:\n")
print("The train score for lr model is {}".format(train_score_lr))
print("The test score for lr model is {}".format(test_score_lr))
```

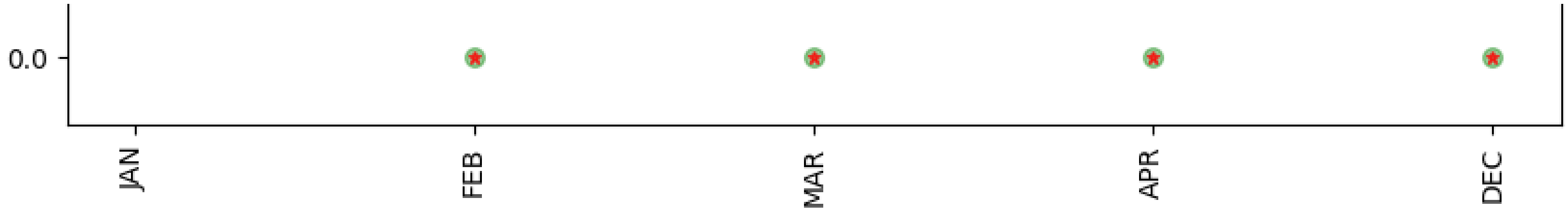
```

Linear Regression Model:

The train score for lr model is 1.0
The test score for lr model is 1.0
```

In [35]:
```
ridgeReg = Ridge(alpha=10)
ridgeReg.fit(x_train,y_train)
#train and test scorefor ridge regression
train_score_ridge = ridgeReg.score(x_train, y_train)
test_score_ridge = ridgeReg.score(x_test, y_test)
print("\nRidge 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 is 0.9999999999856335
The test score for ridge model is 0.9999999999840021
```

In [36]:
```
plt.figure(figsize=(10,10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker="*",markersize=5,color='red',label=r'Ridge;$\alpha
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker="o",markersize=7,color='green',label='LinearRegression')
plt.xticks(rotation=90)
plt.legend()
plt.show()
```


Ridge;$\alpha = 10$
LinearRegression
1.0
0.8
0.6
0.4
0.2



The Accuracy is 0.999

Lasso Regression

In [37]:
```
#Importing libraries
lasso= Lasso(alpha=10)
lasso.fit(x_train,y_train)
#train and test scorefor ridge regression
train_score_ls = lasso.score(x_train, y_train)
test_score_ls= lasso.score(x_test, y_test)
print("\nLasso Model:\n")
print("The train score for lasso model is {}".format(train_score_ls))
print("The test score for lasso model is {}".format(test_score_ls))
```

```
Lasso Model:

The train score for lasso model is 0.9999147271297208
The test score for lasso model is 0.9999147248375002
```

In [38]:
```
plt.figure(figsize=(10,10))
```

Out[38]:
```
<Figure size 1000x1000 with 0 Axes>

<Figure size 1000x1000 with 0 Axes>
```

In [39]:
```
from sklearn.linear_model import LassoCV
```

In [40]:
```
#using the linear cv model
from sklearn.linear_model import RidgeCV
#cross validation
ridge_cv=RidgeCV(alphas =[0.0001,0.001,0.01,0.1,1,10]).fit(x_train,y_train)
#score
print(ridge_cv.score(x_train,y_train))
print(ridge_cv.score(x_test,y_test))
```

```
0.999999999261034
0.9999999993719254
```

In [41]:
```
#using the linear cv model
from sklearn.linear_model import LassoCV
#cross validation
lasso_cv=LassoCV(alphas =[0.0001,0.001,0.01,0.1,1,10]).fit(x_train,y_train)
#score
print(lasso_cv.score(x_train,y_train))
print(lasso_cv.score(x_test,y_test))
```

```
0.9999999999999915
0.9999999999999915
```

The Accuracy is 0.999

Elastic Net

In [42]:
```
from sklearn.linear_model import ElasticNet
```

In [43]:
```
el=ElasticNet()
el.fit(x_train,y_train)
print(el.coef_)
print(el.intercept_)
el.score(x,y)
```

```
[9.99044548e-01 1.38835344e-05 4.58897515e-05 0.00000000e+00
 0.00000000e+00]
0.01656567968369771
```

Out[43]:
```
0.9999991435191248
```

In [44]:
```
y_pred_elastic=el.predict(x_train)
```

In [45]:
```
mean_squared_error=np.mean((y_pred_elastic-y_train)**2)
print(mean_squared_error)
```

```
0.0009226812593703956
```

The accuracy is 0.0009

CONCLUSION

The given dataset is "Rain Fall Prediction".we need to find the bestfit for this dataset.For tht we used different models and got different accuracies.for linear regression we got the accuracy of 1.0.For lasso regression we got the accuracy of 0.999.For Ridge regreesion we got the accuracy of 0.999.For ElasticNet we got the accuracy of 0.0009.

So,the Ridge Regression is the bestfit model for the given dataset "Rainfall Prediction"

In [ ]: