# PROBLEM STATEMENT:-

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

### Importing All The Required Libraries

In [1]:

```
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn import preprocessing,svm
import matplotlib.pyplot as plt
import seaborn as sns
```

# STEP-1:Data Collection

In [2]:

```
df=pd.read_csv(r"C:\Users\thara\Downloads\RainFall.csv")
df
```

Out[2]:

| | STATE_UT_NAME | DISTRICT | JAN | FEB | MAR | APR | MAY | JUN | JUL | A |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | ANDAMAN And NICOBAR ISLANDS | NICOBAR | 107.3 | 57.9 | 65.2 | 117.0 | 358.5 | 295.5 | 285.0 | 27 |
| **1** | ANDAMAN And NICOBAR ISLANDS | SOUTH ANDAMAN | 43.7 | 26.0 | 18.6 | 90.5 | 374.4 | 457.2 | 421.3 | 42 |
| **2** | ANDAMAN And NICOBAR ISLANDS | N & M ANDAMAN | 32.7 | 15.9 | 8.6 | 53.4 | 343.6 | 503.3 | 465.4 | 46 |
| **3** | ARUNACHAL PRADESH | LOHIT | 42.2 | 80.8 | 176.4 | 358.5 | 306.4 | 447.0 | 660.1 | 42 |
| **4** | ARUNACHAL PRADESH | EAST SIANG | 33.3 | 79.5 | 105.9 | 216.5 | 323.0 | 738.3 | 990.9 | 71 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **636** | KERALA | IDUKKI | 13.4 | 22.1 | 43.6 | 150.4 | 232.6 | 651.6 | 788.9 | 52 |
| **637** | KERALA | KASARGOD | 2.3 | 1.0 | 8.4 | 46.9 | 217.6 | 999.6 | 1108.5 | 63 |
| **638** | KERALA | PATHANAMTHITTA | 19.8 | 45.2 | 73.9 | 184.9 | 294.7 | 556.9 | 539.9 | 35 |
| **639** | KERALA | WAYANAD | 4.8 | 8.3 | 17.5 | 83.3 | 174.6 | 698.1 | 1110.4 | 59 |
| **640** | LAKSHADWEEP | LAKSHADWEEP | 20.8 | 14.7 | 11.8 | 48.9 | 171.7 | 330.2 | 287.7 | 21 |

641 rows × 19 columns

# STEP-2:Data Cleaning And Preprocessing

In [3]:

```
df.head()
```

Out[3]:

| | STATE_UT_NAME | DISTRICT | JAN | FEB | MAR | APR | MAY | JUN | JUL | AUG | SEP |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | ANDAMAN And NICOBAR ISLANDS | NICOBAR | 107.3 | 57.9 | 65.2 | 117.0 | 358.5 | 295.5 | 285.0 | 271.9 | 354.8 |
| **1** | ANDAMAN And NICOBAR ISLANDS | SOUTH ANDAMAN | 43.7 | 26.0 | 18.6 | 90.5 | 374.4 | 457.2 | 421.3 | 423.1 | 455.6 |
| **2** | ANDAMAN And NICOBAR ISLANDS | N & M ANDAMAN | 32.7 | 15.9 | 8.6 | 53.4 | 343.6 | 503.3 | 465.4 | 460.9 | 454.8 |
| **3** | ARUNACHAL PRADESH | LOHIT | 42.2 | 80.8 | 176.4 | 358.5 | 306.4 | 447.0 | 660.1 | 427.8 | 313.6 |
| **4** | ARUNACHAL PRADESH | EAST SIANG | 33.3 | 79.5 | 105.9 | 216.5 | 323.0 | 738.3 | 990.9 | 711.2 | 568.0 |

In [4]:

```
df.tail()
```

Out[4]:

| | STATE_UT_NAME | DISTRICT | JAN | FEB | MAR | APR | MAY | JUN | JUL | AU |
|---|---|---|---|---|---|---|---|---|---|---|
| **636** | KERALA | IDUKKI | 13.4 | 22.1 | 43.6 | 150.4 | 232.6 | 651.6 | 788.9 | 527 |
| **637** | KERALA | KASARGOD | 2.3 | 1.0 | 8.4 | 46.9 | 217.6 | 999.6 | 1108.5 | 636 |
| **638** | KERALA | PATHANAMTHITTA | 19.8 | 45.2 | 73.9 | 184.9 | 294.7 | 556.9 | 539.9 | 352 |
| **639** | KERALA | WAYANAD | 4.8 | 8.3 | 17.5 | 83.3 | 174.6 | 698.1 | 1110.4 | 592 |
| **640** | LAKSHADWEEP | LAKSHADWEEP | 20.8 | 14.7 | 11.8 | 48.9 | 171.7 | 330.2 | 287.7 | 217 |

In [5]:

```
df.describe()
```

Out[5]:

| | JAN | FEB | MAR | APR | MAY | JUN | JUL |
|---|---|---|---|---|---|---|---|
| **count** | 641.000000 | 641.000000 | 641.000000 | 641.000000 | 641.000000 | 641.000000 | 641.00000( |
| **mean** | 18.355070 | 20.984399 | 30.034789 | 45.543214 | 81.535101 | 196.007332 | 326.03369‡ |
| **std** | 21.082806 | 27.729596 | 45.451082 | 71.556279 | 111.960390 | 196.556284 | 221.36464: |
| **min** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.900000 | 3.800000 | 11.60000( |
| **25%** | 6.900000 | 7.000000 | 7.000000 | 5.000000 | 12.100000 | 68.800000 | 206.40000( |
| **50%** | 13.300000 | 12.300000 | 12.700000 | 15.100000 | 33.900000 | 131.900000 | 293.70000( |
| **75%** | 19.200000 | 24.100000 | 33.200000 | 48.300000 | 91.900000 | 226.600000 | 374.80000( |
| **max** | 144.500000 | 229.600000 | 367.900000 | 554.400000 | 733.700000 | 1476.200000 | 1820.90000( |

In [6]:

```
df.shape
```

Out[6]:

```
(641, 19)
```

In [7]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 641 entries, 0 to 640
Data columns (total 19 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   STATE_UT_NAME  641 non-null    object 
 1   DISTRICT       641 non-null    object 
 2   JAN            641 non-null    float64
 3   FEB            641 non-null    float64
 4   MAR            641 non-null    float64
 5   APR            641 non-null    float64
 6   MAY            641 non-null    float64
 7   JUN            641 non-null    float64
 8   JUL            641 non-null    float64
 9   AUG            641 non-null    float64
 10  SEP            641 non-null    float64
 11  OCT            641 non-null    float64
 12  NOV            641 non-null    float64
 13  DEC            641 non-null    float64
 14  ANNUAL         641 non-null    float64
 15  Jan-Feb        641 non-null    float64
 16  Mar-May        641 non-null    float64
 17  Jun-Sep        641 non-null    float64
 18  Oct-Dec        641 non-null    float64
dtypes: float64(17), object(2)
memory usage: 95.3+ KB
```

In [8]:

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

Out[8]:

```
STATE_UT_NAME    False
DISTRICT         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 [10]:

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

Out[10]:

```
STATE_UT_NAME    0
DISTRICT         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.columns
```

Out[11]:

```
Index(['STATE_UT_NAME', 'DISTRICT', 'JAN', 'FEB', 'MAR', 'APR', 'MAY', 'JU
N',
       'JUL', 'AUG', 'SEP', 'OCT', 'NOV', 'DEC', 'ANNUAL', 'Jan-Feb',
       'Mar-May', 'Jun-Sep', 'Oct-Dec'],
      dtype='object')
```

In [12]:

```
df.duplicated().sum()
```

Out[12]:

```
0
```

In [13]:

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

Out[13]:

```
ANNUAL
747.1     9
2080.0    4
1336.5    3
1824.8    3
2814.4    3
         ..
1037.6    1
907.2     1
944.5     1
1003.3    1
3253.1    1
Name: count, Length: 591, dtype: int64
```

# Feature Scaling:

## To Split the data into training data and test data

In [14]:

```
x=df[["APR"]]
y=df["JAN"]
```

In [15]:

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=50)
```

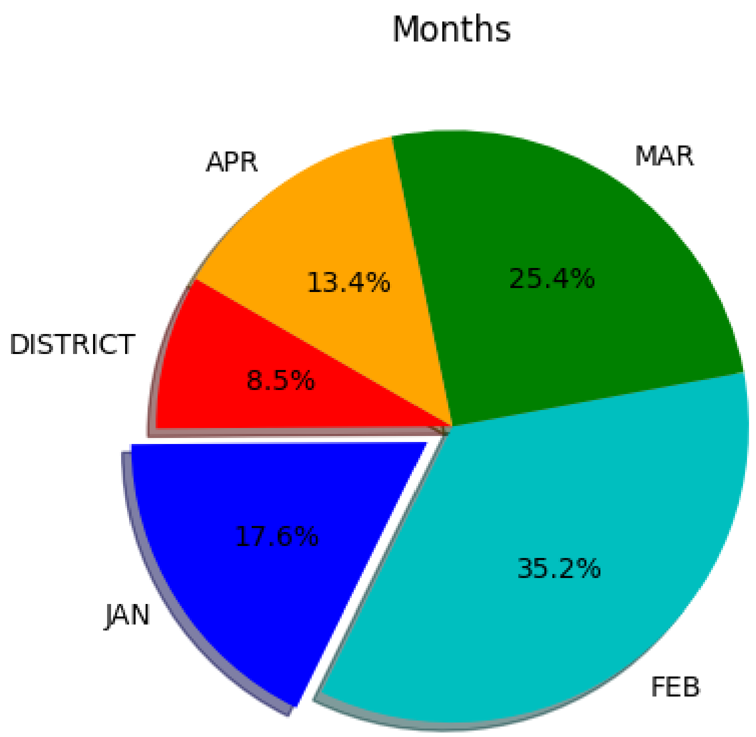
# STEP-3: Data Visualization

In [16]:

```
slice = [12, 25, 50, 36, 19]
activities = ['DISTRICT','JAN', 'FEB', 'MAR', 'APR']
cols = ['r', 'b', 'c', 'g', 'orange']
plt.pie(slice,labels = activities,colors=cols,startangle=150,shadow=True,explode=(0, 0.1
plt.title('Months')
```

Out[16]:

```
Text(0.5, 1.0, 'Months')
```

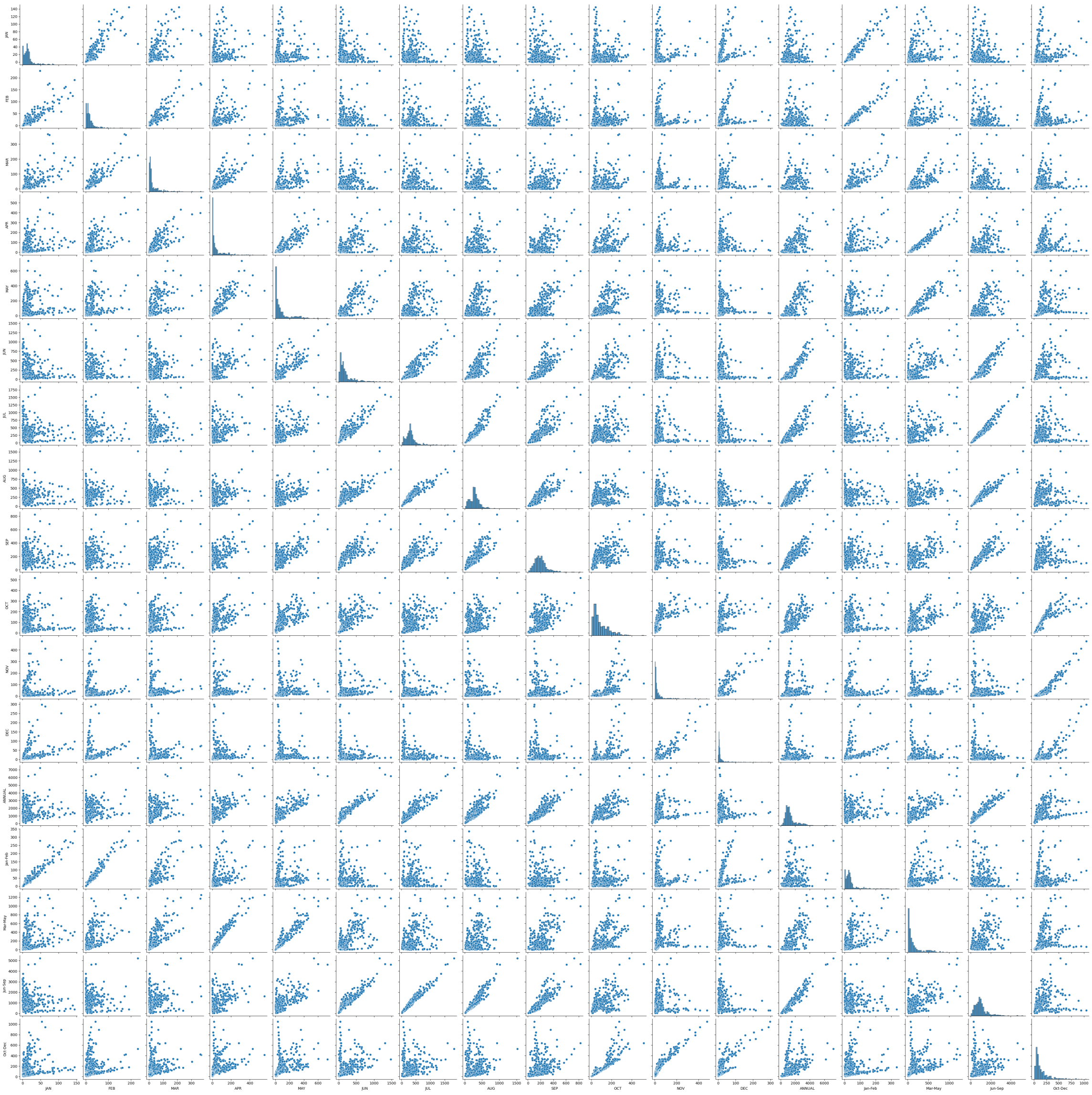


In [17]:

```
sns.pairplot(df)
```

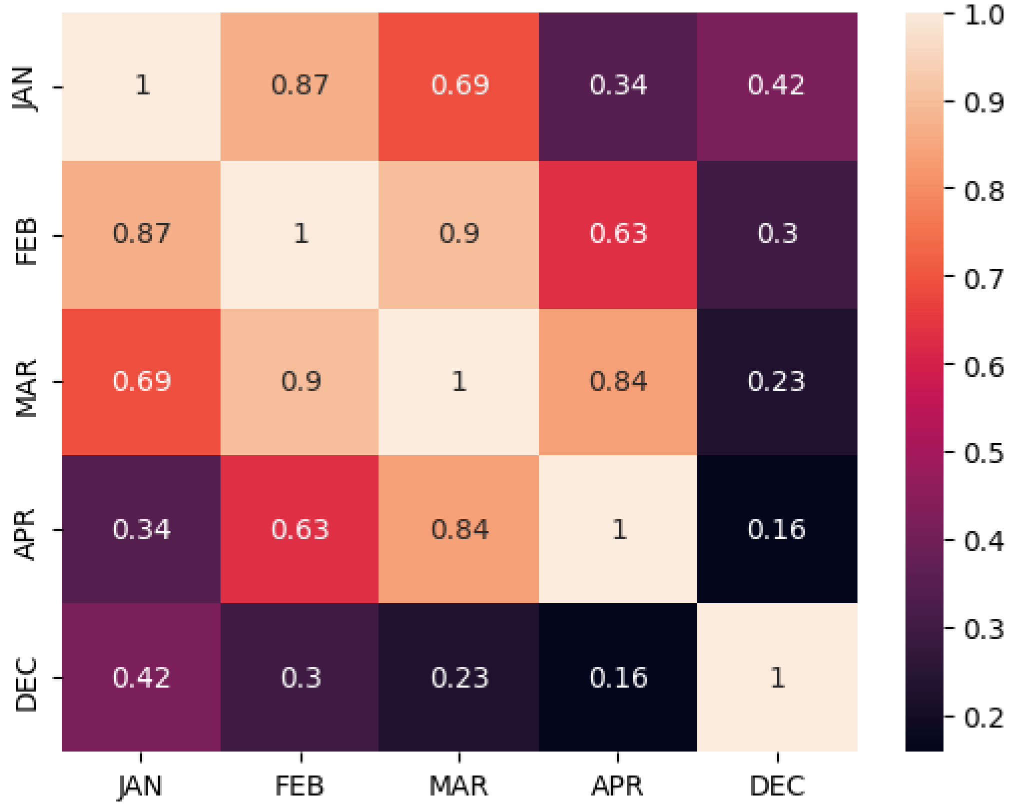
Out[17]:

```
<seaborn.axisgrid.PairGrid at 0x2393a533bb0>
```



In [18]:

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




# STEP-4:- Data Modelling

# Applying LINEAR REGRESSION

In [19]:

```
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_
```

13.76772905796956

Out[19]:

| | coefficient |
|---|---|
| **APR** | 0.08298 |

In [20]:

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

```
0.13760810725167383
```

In [21]:

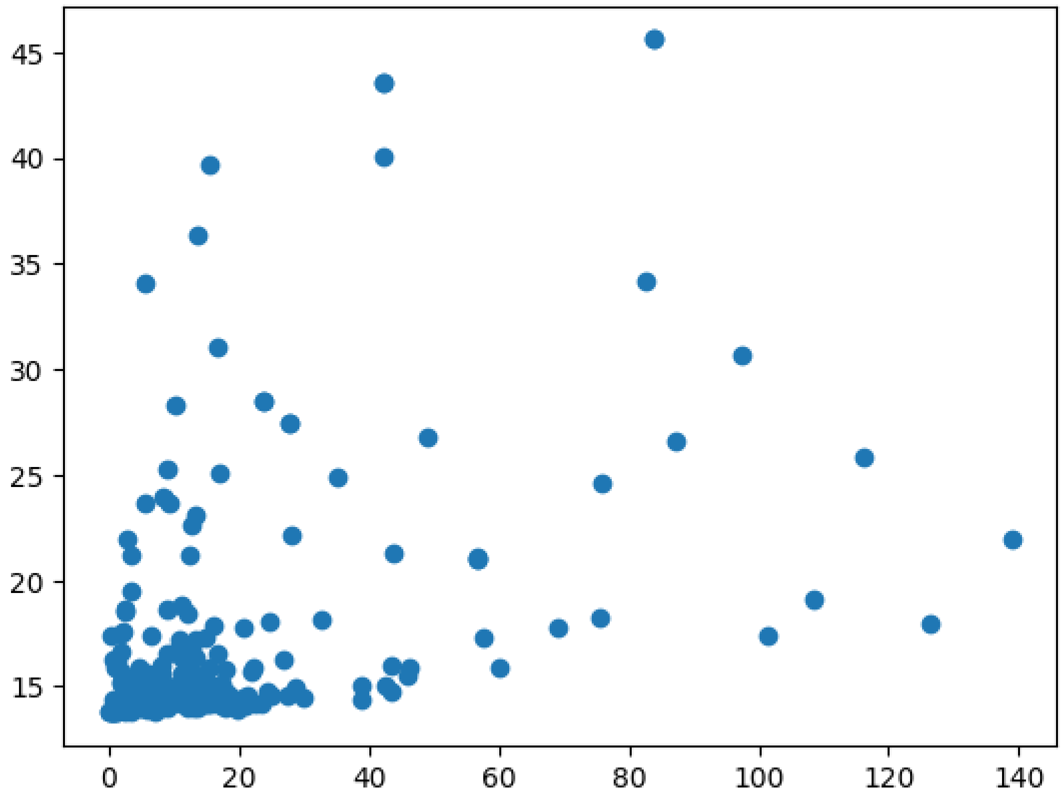
```
predictions=reg.predict(X_test)
```

In [22]:

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

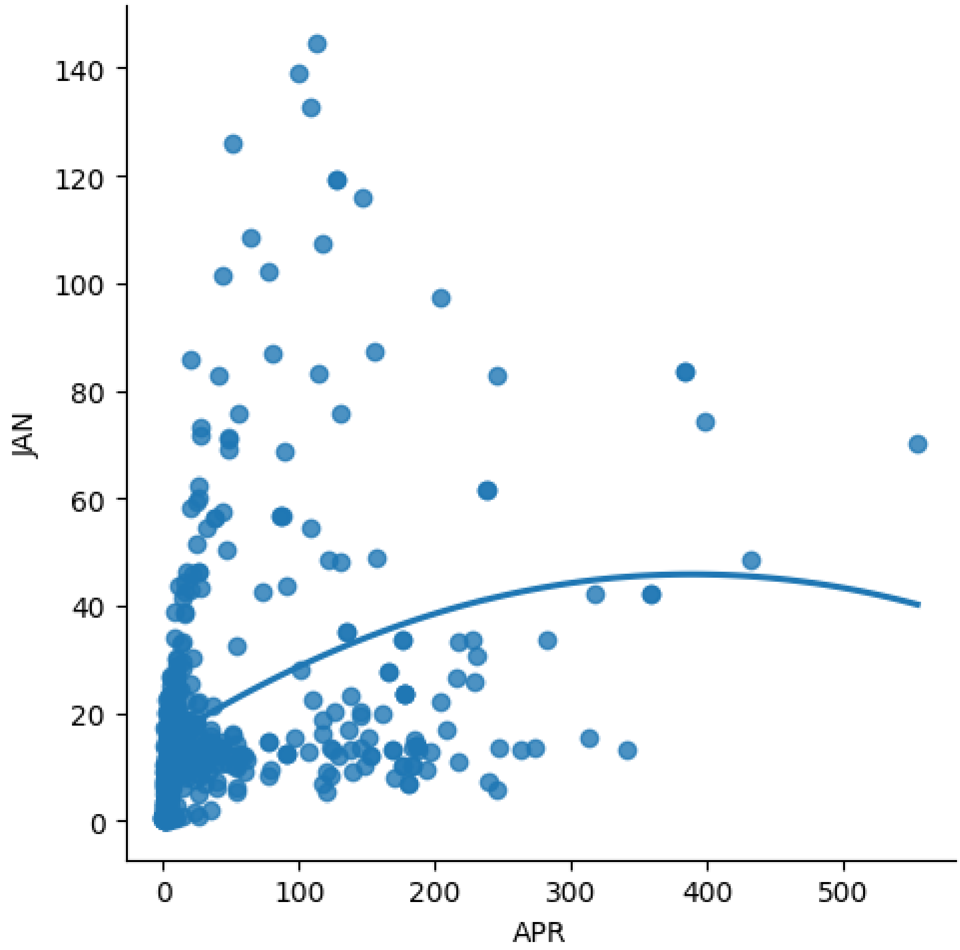
Out[22]:

```
<matplotlib.collections.PathCollection at 0x2394ebfa8c0>
```



In [23]:

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




In [24]:

```
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)
```

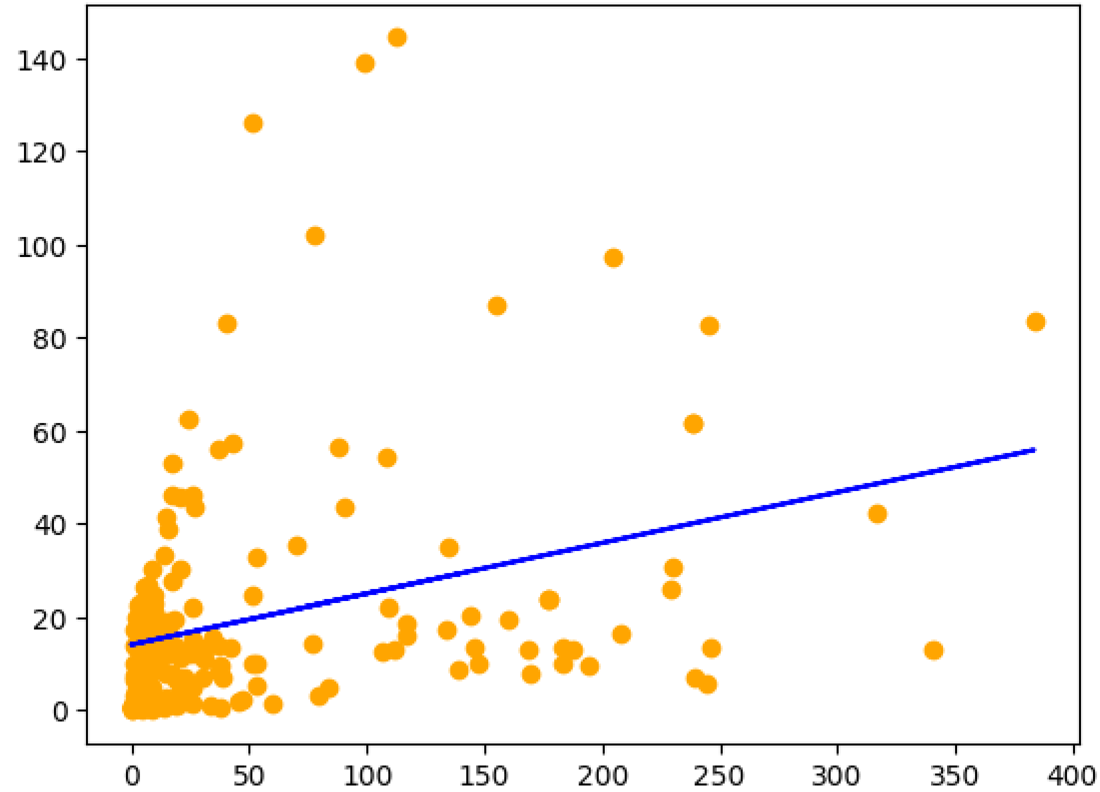
Out[24]:

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 [25]:

```
y_pred=reg.predict(X_test)
plt.scatter(X_test,y_test,color='ORANGE')
plt.plot(X_test,y_pred,color='BLUE')
plt.show()
```




In [26]:

```
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: 0.11486832305419492
```

# RIDGE MODEL

In [27]:

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

In [28]:

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

In [29]:

```
x=np.array(df['APR']).reshape(-1,1)
y=np.array(df['JAN']).reshape(-1,1)
```

In [30]:

```
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 [31]:

```
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 [32]:

```
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))
```

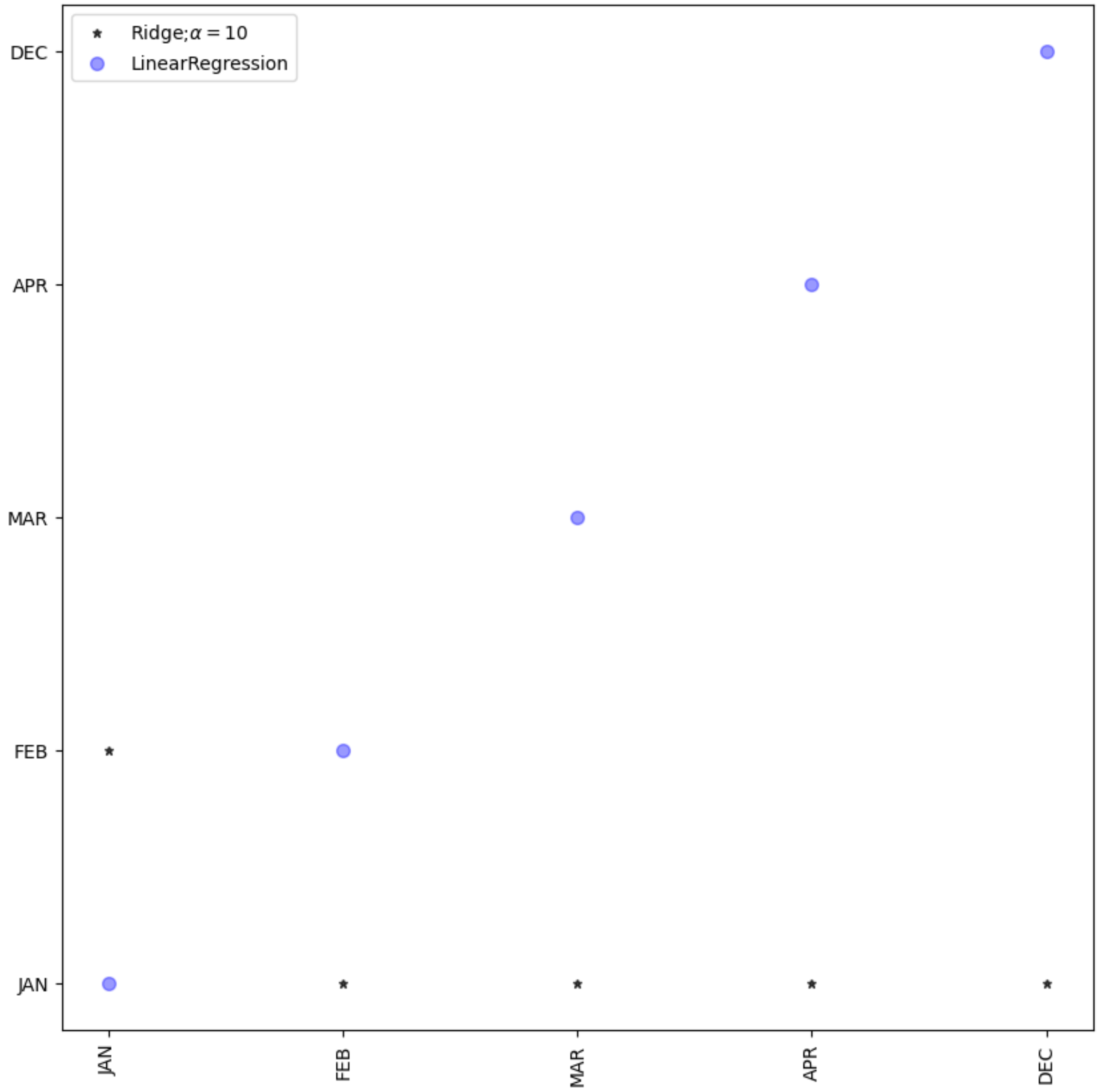
```
 Ridge Model:

the train score for ridge model is0.9999999792491524
the test score for ridge model is0.9999999887465535
```

In [33]:

```
lr=LinearRegression()
```

In [34]:

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



# LASSO MODEL

In [35]:

```
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))
```

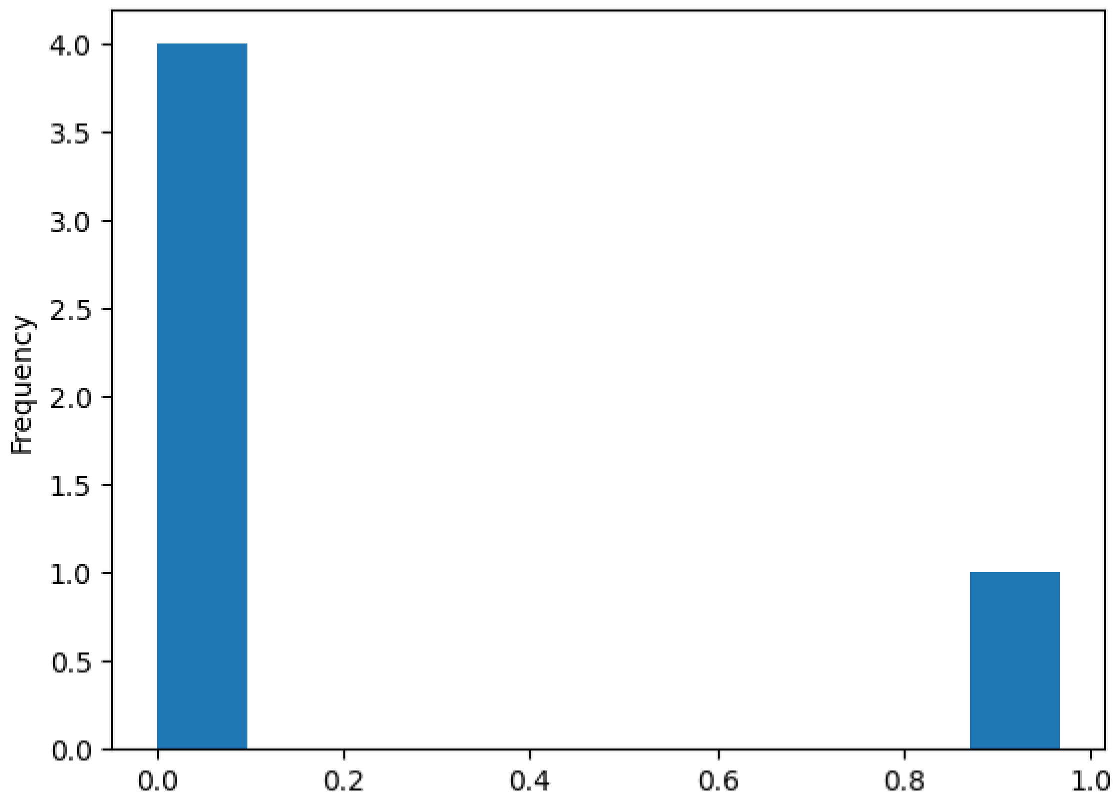
```
 Lasso Model:

The train score for ls model is 0.99912857000705
The test score for ls model is0.9991969731663574
```

In [36]:

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

Out[36]:

```
<Axes: ylabel='Frequency'>
```



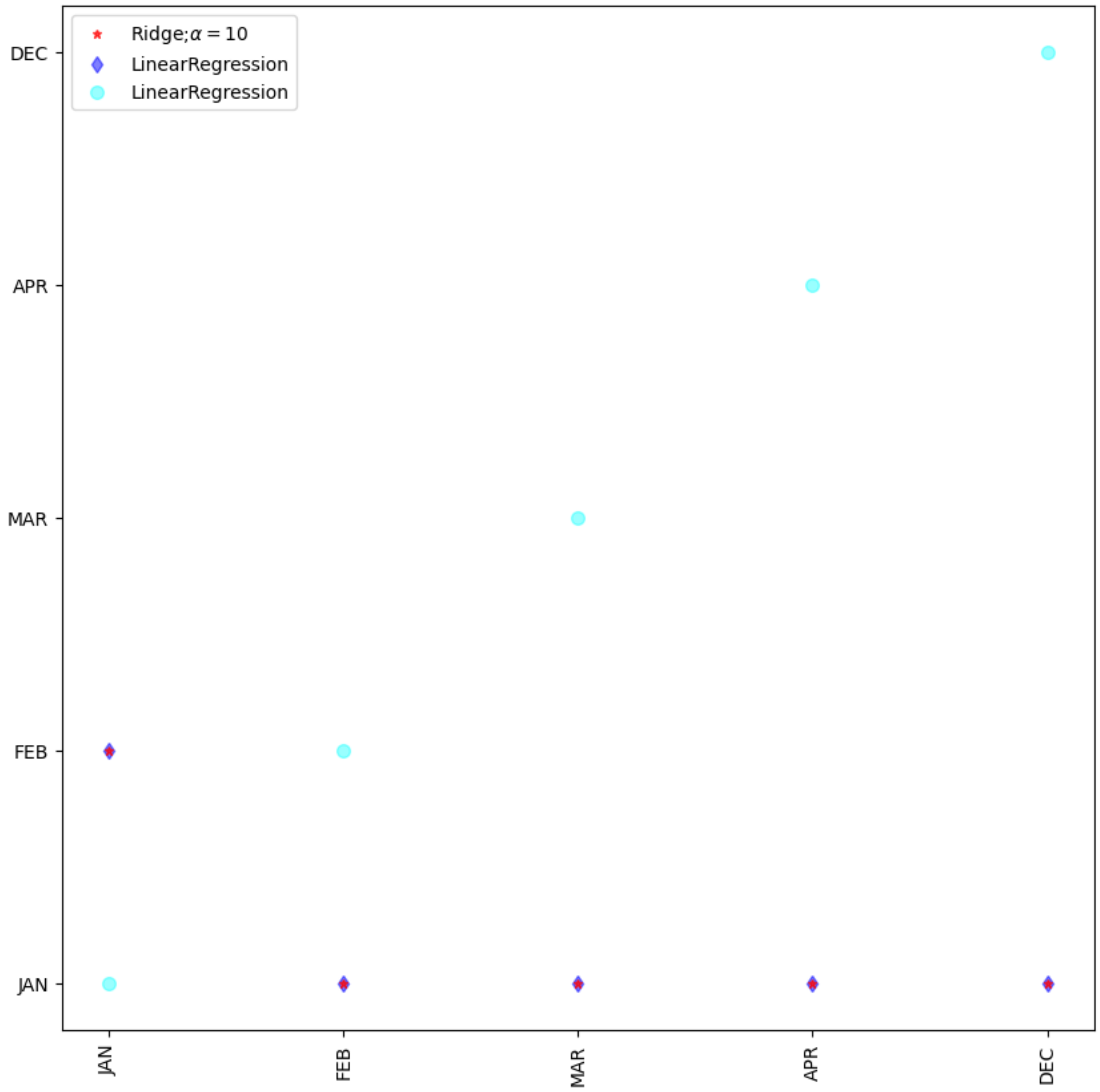
In [37]:

```
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))
```

```
0.9999999999999198
0.9999999999999254
```

In [38]:

```
plt.figure(figsize= (10,10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker="*",markersize=5,colo
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="Aqua",label=
plt.xticks(rotation=90)
plt.xticks(rotation = 90)
plt.legend()
plt.show()
```



# ELASTICNET

In [39]:

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

```
[ 9.93078025e-01  4.17330119e-03  0.00000000e+00 -2.56844715e-04
  4.33064019e-04]
0.04331617537314969
0.9999905490942288
```

In [45]:

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

```
0.004278888506659882
```

# Conclusion:

THE SCORE OF LINEAR REGRESSION IS :- 0.13760810725167383 THE SCORE OF RIDGE MODEL IS :- 0.9999999792491524 THE SCORE OF LASSO MODEL IS :- 0.9999999999999198 THE SCORE OF ELASTIC NET IS :- 0.9999905490942288 *AMONG ALL MODELS LASSO YEILD HIGHEST ACCURACY.SO,WE PREFER LASSO MODEL FOR THIS DATA SET*