

problem statement:-To Predict the Rainfall based on the 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 [3]: df=pd.read_csv(r"C:\Users\anu\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	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
...
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

4116 rows × 19 columns

Data 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.isnull().any()
```

```
Out[6]: 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 [7]: df.fillna(method='ffill',inplace=True)
```

```
In [8]: df.fillna(method='bfill',inplace=True)
```

```
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]: df.describe()
```

Out[10]:

	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	
count	4116.000000	4116.000000	4116.000000	4116.000000	4116.000000	4116.000000	4116.000000	4116.000000	4116.000000	4116.000000	4116.000000	4116.
mean	1958.218659	18.957240	21.823251	27.415379	43.160641	85.788994	230.567979	347.177235	290.239796	197.524781	95.724198	40.
std	33.140898	33.576192	35.922602	47.045473	67.816588	123.220150	234.896056	269.321089	188.785639	135.509037	99.689878	68.
min	1901.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.400000	0.000000	0.000000	0.100000	0.000000	0.
25%	1930.000000	0.600000	0.600000	1.000000	3.000000	8.600000	70.475000	175.900000	155.850000	100.575000	14.600000	0.
50%	1958.000000	6.000000	6.700000	7.900000	15.700000	36.700000	138.900000	284.800000	259.400000	174.000000	65.750000	9.
75%	1987.000000	22.200000	26.800000	31.400000	50.125000	97.400000	306.150000	418.325000	377.800000	266.225000	148.600000	46.
max	2015.000000	583.700000	403.500000	605.600000	595.100000	1168.600000	1609.900000	2362.800000	1664.600000	1222.000000	948.300000	648.

```
In [11]: 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.columns
```

```
Out[12]: 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 [13]: df.shape
```

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

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

```
Out[14]: 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 [15]: df['Jan-Feb'].value_counts()
```

```
Out[15]: 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 [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()
```

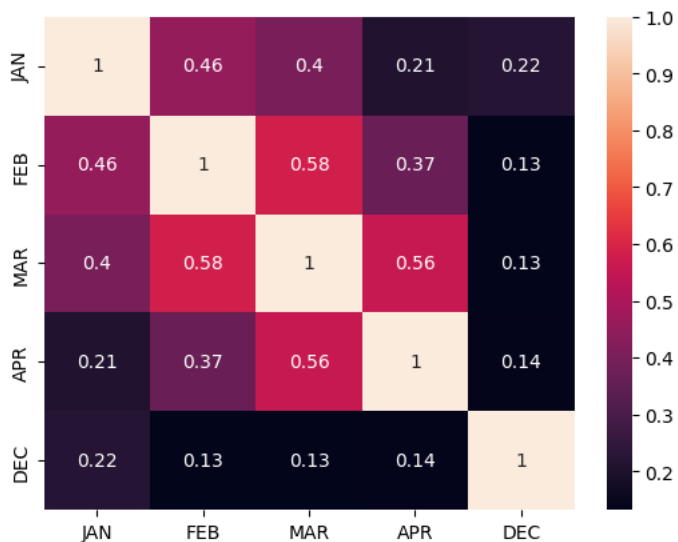
```
Out[17]: Jun-Sep
434.3    4
334.8    4
573.8    4
613.3    4
1082.3   3
..
301.6    1
380.9    1
409.3    1
229.4    1
958.5    1
Name: count, Length: 3683, dtype: int64
```

```
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']]
```

Linear Regression

```
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=42)
```

```
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_
```

[9.65066661]

Out[23]:

	coefficient
FEB	0.442278

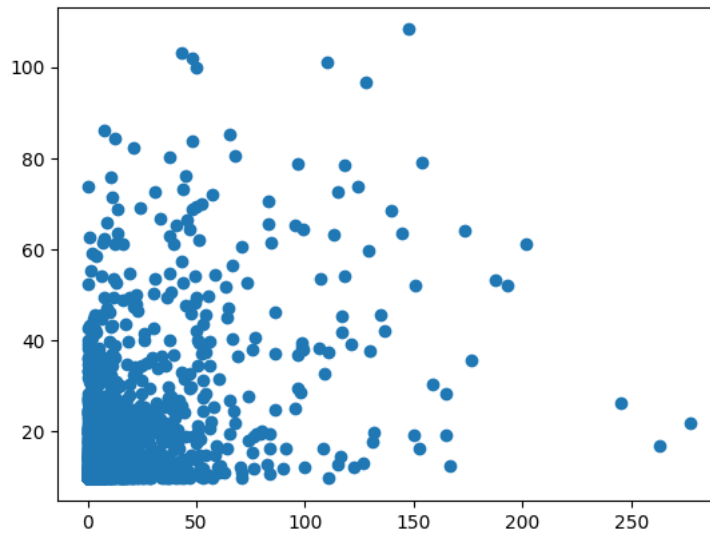
```
In [24]: score=reg.score(x_test,y_test)
print(score)
```

0.1793580786264921

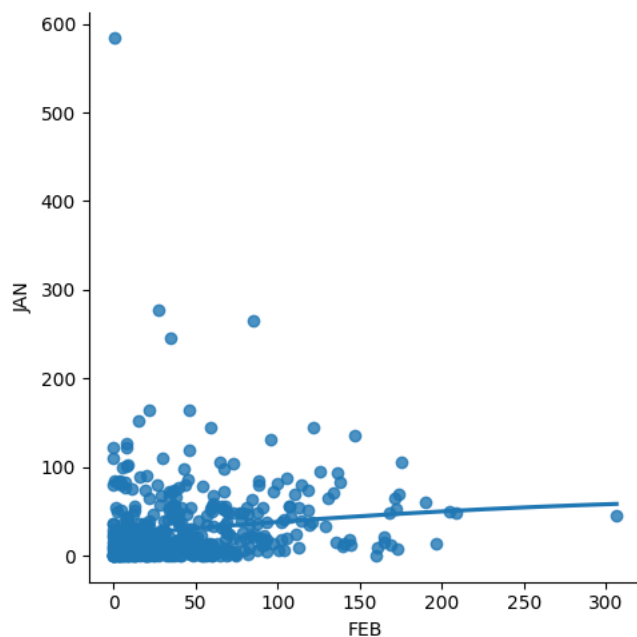
```
In [25]: predictions=reg.predict(x_test)
```

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

Out[26]: <matplotlib.collections.PathCollection at 0xfb00ae3e50>



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

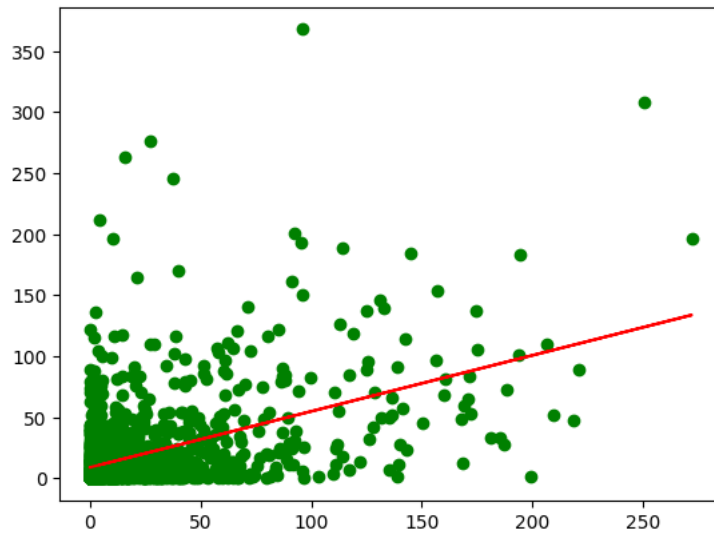


```
In [28]: x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.33)
reg.fit(x_train,y_train)
reg.fit(x_test,y_test)
```

Out[28]:

```
LinearRegression
LinearRegression()
```

```
In [29]: y_pred=reg.predict(x_test)
plt.scatter(x_test,y_test,color='green')
plt.plot(x_test,y_pred,color='red')
plt.show()
```



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

```
In [31]: features=df.columns[0:5]
targets=df.columns[-5]
```

```
In [32]: x=df[features].values
y=df[targets].values
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=17)
```

```
In [33]: 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 [ ]: 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))
```

```
In [35]: lr=LinearRegression()
```

Lasso Model

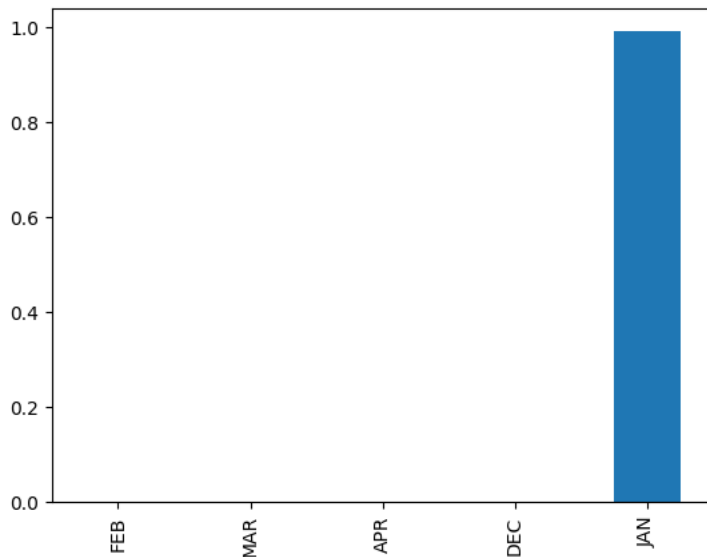
```
In [37]: print("\n lasso Model:\n")
lasso=Lasso(alpha=10)
lasso.fit(x_train,y_train)
train_score_ls=lasso.score(x_train,y_train)
test_score_ls=lasso.score(x_test,y_test)
print(" the train score for ls model is{}".format(train_score_ls))
print(" the test score for ls model is{}".format(test_score_ls))
```

lasso Model:

```
the train score for ls model is0.9999207747038827
the test score for ls model is0.9999206791315255
```

```
In [38]: pd.Series(lasso.coef_, features).sort_values(ascending=True).plot(kind="bar")
```

```
Out[38]: <Axes: >
```



```
In [39]: from sklearn.linear_model import LassoCV
lasso_cv=LassoCV(alphas=[0.0001,0.001,0.01,0.1,1,10],random_state=0).fit(x_train,y_train)
print(lasso_cv.score(x_train,y_train))
print(lasso_cv.score(x_test,y_test))

0.9999999999999921
0.9999999999999921
```

Elastic Net:-

```
In [40]: from sklearn.linear_model import ElasticNet
eln=ElasticNet()
eln.fit(x,y)
print(eln.coef_)
print(eln.intercept_)
print(eln.score(x,y))

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

```
In [41]: y_pred_elastic=eln.predict(x_train)
mean_squared_score=np.mean((y_pred_elastic - y_train)**2)
print(mean_squared_score)

0.0008816302333951295
```

Conclusion

among all models lasso yield highest accuracy.so we prefer Lasso model for this Rain fall dataset

```
In [ ]:
```