Problem Statement

Finding the best fit by performing all the model

Data Collection

In [1]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import preprocessing,svm
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df=pd.read_csv(r"C:\Users\Sushma sree\Downloads\rainfall in india 1901-201
df
```

Out[2]:

	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
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
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
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
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
...	...	...	...	...	...	...	...	...	...	...	...	...
4111	LAKSHADWEEP	2011	5.1	2.8	3.1	85.9	107.2	153.6	350.2	254.0	255.2	117.4
4112	LAKSHADWEEP	2012	19.2	0.1	1.6	76.8	21.2	327.0	231.5	381.2	179.8	145.9
4113	LAKSHADWEEP	2013	26.2	34.4	37.5	5.3	88.3	426.2	296.4	154.4	180.0	72.8
4114	LAKSHADWEEP	2014	53.2	16.1	4.4	14.9	57.4	244.1	116.1	466.1	132.2	169.2
4115	LAKSHADWEEP	2015	2.2	0.5	3.7	87.1	133.1	296.6	257.5	146.4	160.4	165.4

4116 rows × 19 columns

Data Preprocessing

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

Out[3]:

	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

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

Out[4]:

	SUBDIVISION	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	
4111	LAKSHADWEEP	2011	5.1	2.8	3.1	85.9	107.2	153.6	350.2	254.0	255.2	117.4	1
4112	LAKSHADWEEP	2012	19.2	0.1	1.6	76.8	21.2	327.0	231.5	381.2	179.8	145.9	
4113	LAKSHADWEEP	2013	26.2	34.4	37.5	5.3	88.3	426.2	296.4	154.4	180.0	72.8	
4114	LAKSHADWEEP	2014	53.2	16.1	4.4	14.9	57.4	244.1	116.1	466.1	132.2	169.2	
4115	LAKSHADWEEP	2015	2.2	0.5	3.7	87.1	133.1	296.6	257.5	146.4	160.4	165.4	2

In [5]:
```
1 df.describe
```

Out[5]:
```
<bound method NDFrame.describe of                    SUBDIVISION  YEAR   JAN
FEB   MAR    APR    MAY    JUN
0     ANDAMAN & NICOBAR ISLANDS  1901  49.2   87.1  29.2    2.3  528.8  517.5
\
1     ANDAMAN & NICOBAR ISLANDS  1902   0.0  159.8  12.2    0.0  446.1  537.1
2     ANDAMAN & NICOBAR ISLANDS  1903  12.7  144.0   0.0    1.0  235.1  479.9
3     ANDAMAN & NICOBAR ISLANDS  1904   9.4   14.7   0.0  202.4  304.5  495.1
4     ANDAMAN & NICOBAR ISLANDS  1905   1.3    0.0   3.3   26.9  279.5  628.7
...                         ...   ...   ...   ...   ...    ...    ...    ...
4111               LAKSHADWEEP  2011   5.1    2.8   3.1   85.9  107.2  153.6
4112               LAKSHADWEEP  2012  19.2    0.1   1.6   76.8   21.2  327.0
4113               LAKSHADWEEP  2013  26.2   34.4  37.5    5.3   88.3  426.2
4114               LAKSHADWEEP  2014  53.2   16.1   4.4   14.9   57.4  244.1
4115               LAKSHADWEEP  2015   2.2    0.5   3.7   87.1  133.1  296.6

        JUL    AUG    SEP    OCT    NOV    DEC  ANNUAL  Jan-Feb  Mar-May
0     365.1  481.1  332.6  388.5  558.2   33.6  3373.2    136.3    560.3  \
1     228.9  753.7  666.2  197.2  359.0  160.5  3520.7    159.8    458.3
2     728.4  326.7  339.0  181.2  284.4  225.0  2957.4    156.7    236.1
3     502.0  160.1  820.4  222.2  308.7   40.1  3079.6     24.1    506.9
4     368.7  330.5  297.0  260.7   25.4  344.7  2566.7      1.3    309.7
...     ...    ...    ...    ...    ...    ...     ...      ...      ...
4111  350.2  254.0  255.2  117.4  184.3   14.9  1533.7      7.9    196.2
4112  231.5  381.2  179.8  145.9   12.4    8.8  1405.5     19.3     99.6
4113  296.4  154.4  180.0   72.8   78.1   26.7  1426.3     60.6    131.1
4114  116.1  466.1  132.2  169.2   59.0   62.3  1395.0     69.3     76.7
4115  257.5  146.4  160.4  165.4  231.0  159.0  1642.9      2.7    223.9

      Jun-Sep  Oct-Dec
0      1696.3    980.3
1      2185.9    716.7
2      1874.0    690.6
3      1977.6    571.0
4      1624.9    630.8
...       ...      ...
4111   1013.0    316.6
4112   1119.5    167.1
4113   1057.0    177.6
4114    958.5    290.5
4115    860.9    555.4

[4116 rows x 19 columns]>
```

In [6]:
```
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          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 [7]:
```
1 df.isna().any()
```

Out[7]:
```
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 [8]:
```
1 df.isnull().sum()
```

Out[8]:
```
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 [9]:
```
1 df.dropna(inplace=True)
```

In [10]:
```
1 df.isnull().sum()
```

Out[10]:
```
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 [11]:
```
1 df.shape
```

Out[11]:
```
(4090, 19)
```

In [12]:
```
1 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]:
```
1 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]:
```
1 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]:
```
1 df['Jun-Sep'].value_counts()
```

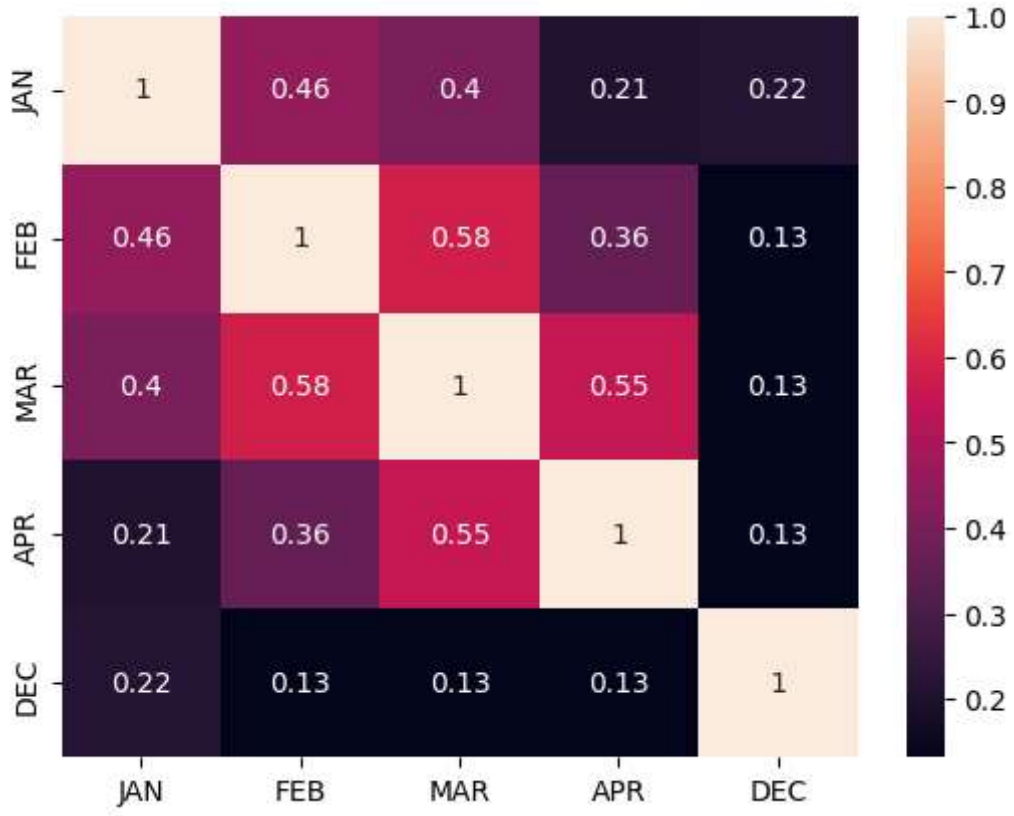
Out[15]:
```
Jun-Sep
334.8    4
434.3    4
573.8    4
613.3    4
403.9    3
        ..
897.7    1
301.6    1
380.9    1
409.3    1
958.5    1
Name: count, Length: 3670, dtype: int64
```

In [16]:
```
1 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]:
```
df.columns
```

Out[18]:
```
Index(['JAN', 'FEB', 'MAR', 'APR', 'DEC'], dtype='object')
```

In [19]:
```
x=df[['FEB']]
y=df['JAN']
```

Linear Regression

In [20]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
```

In [21]:
```
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[21]:
```
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 [22]:
```
lr.intercept_
```

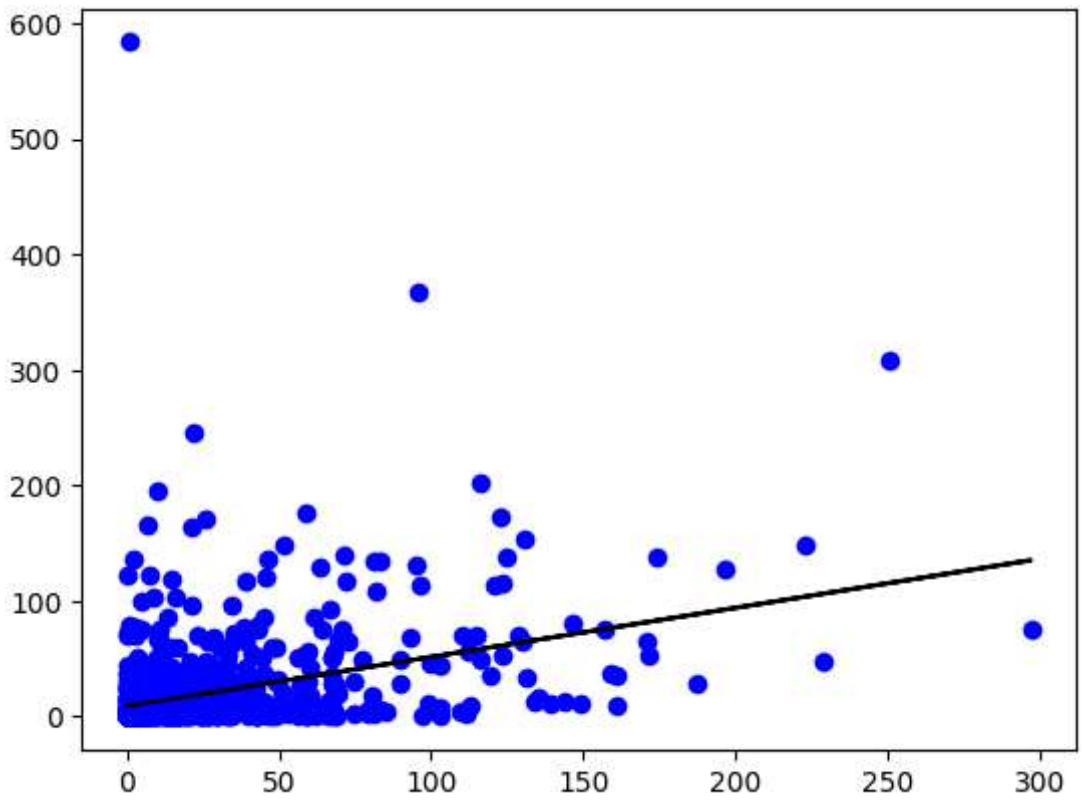
Out[22]:
```
9.152350153272794
```

In [23]:
```
lr.score(x_test,y_test)
```

Out[23]:
```
0.14915515739755592
```

In [24]:
```
y_pred=lr.predict(x_test)
```

In [25]:
```
plt.scatter(x_test,y_test,color='b')
plt.plot(x_test,y_pred,color='k')
plt.show()
```



In [26]:
```
df500=df[:500]
df500
```

Out[26]:

	JAN	FEB	MAR	APR	DEC
0	49.2	87.1	29.2	2.3	33.6
1	0.0	159.8	12.2	0.0	160.5
2	12.7	144.0	0.0	1.0	225.0
3	9.4	14.7	0.0	202.4	40.1
4	1.3	0.0	3.3	26.9	344.7
...	...	...	...	...	...
507	28.4	21.3	63.0	239.8	1.9
508	9.4	41.6	52.1	134.3	0.9
509	13.8	35.9	40.8	82.9	3.9
510	19.1	2.5	94.4	210.8	8.3
511	20.4	29.9	15.6	83.7	6.6

500 rows × 5 columns

In [27]:
```
x=df500[['FEB']]
y=df500['JAN']
```

In [28]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
```

In [29]:
```
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[29]:
```
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 [30]:
```
lr.score(x_test,y_test)
```

Out[30]:
```
-0.058616121876582605
```

In [31]:
```
### The linear regression has the 0.009% of accuracy
```

Ridge Regression

In [32]:
```
from sklearn.linear_model import Ridge,Lasso
from sklearn.preprocessing import StandardScaler
```

In [33]:
```
features=df.columns[:5]
target=df.columns[-5]
```

In [34]:
```
x=np.array(df['JAN']).reshape(-1,1)
y=np.array(df['FEB']).reshape(-1,1)
```

In [35]:
```
x=df[features].values
y=df[target].values
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
```

In [36]:
```
ridgeReg=Ridge(alpha=10)
ridgeReg.fit(x_train,y_train)
train_score_ridge=ridgeReg.score(x_test,y_test)
test_score_ridge=ridgeReg.score(x_test,y_test)
```

In [37]:
```
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 is0.9999999999922101
The test score for ridge model is 0.9999999999922101
```

Conclusion: The train Score for ridge model is 0.99999999998732 The test score for ridge model is 0.99999999998732

Lasso Regression

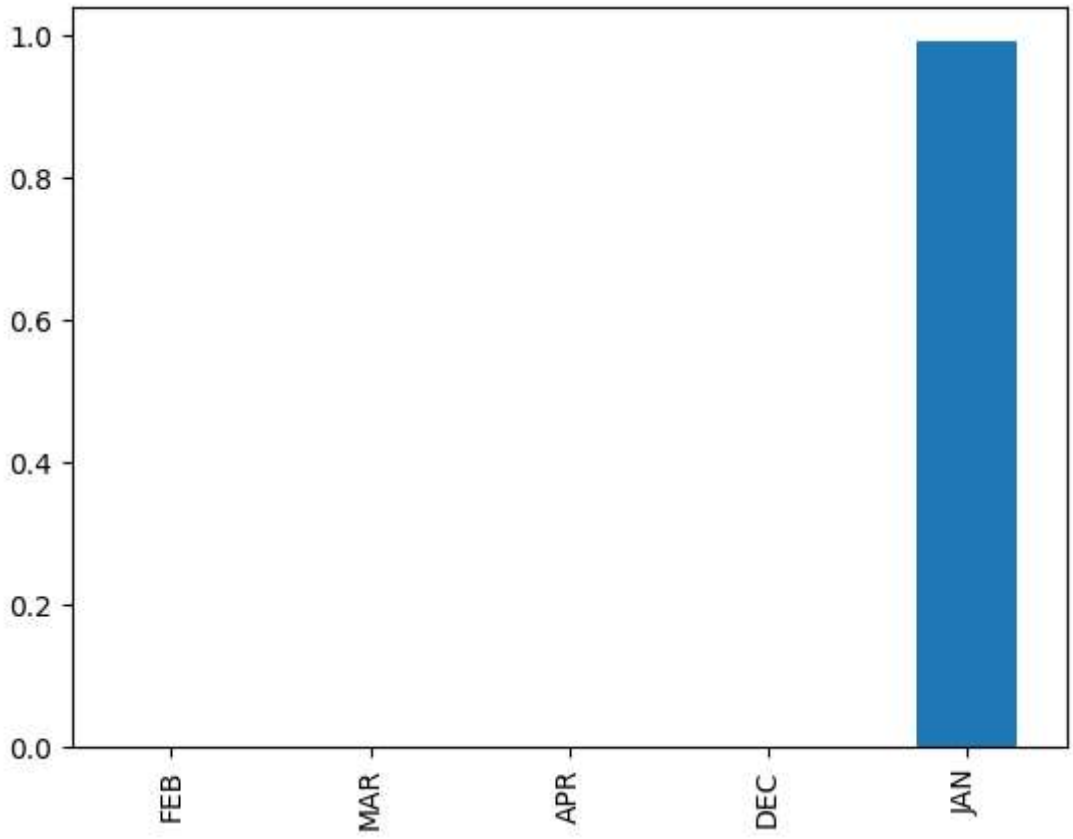
In [40]:
```
print("\n Lasso Model: \n")
lasso=Lasso(alpha=10)
lasso.fit(x_train,y_train)
train_score_ls=lasso.score(x_train,y_train)
test_score_ls=lasso.score(x_test,y_test)
print("The train score for is model is {}".format(train_score_ls))
print("The test score for ls model is {}".format(test_score_ls))
```

```
 Lasso Model:

The train score for is model is 0.9999295366875157
The test score for ls model is 0.9999291499381193
```

In [42]:
```
1 pd.Series(lasso.coef_,features).sort_values(ascending=True).plot(kind="ba
```

Out[42]: <Axes: >



In [44]:
```
1 from sklearn.linear_model import LassoCV
2 lasso_cv=LassoCV(alphas=[0.0001,0.001,0.01,1,10],random_state=0).fit(x_tr
3 print(lasso_cv.score(x_train,y_train))
4 print(lasso_cv.score(x_test,y_test))
```

```
0.999999999999993
0.9999999999999929
```

Elastic Net

In [45]:
```
1 from sklearn.linear_model import ElasticNet
2 eln=ElasticNet()
3 eln.fit(x,y)
4 print(eln.coef_)
5 print(eln.intercept_)
6 print(eln.score(x,y))
```

```
[9.99095428e-01 0.00000000e+00 3.08223758e-05 0.00000000e+00
 0.00000000e+00]
0.01618268382513577
0.9999992110406136
```

In [46]:
```
y_pred_elastic=eln.predict(x_train)
mean_squared_error=np.mean((y_pred_elastic-y_train)**2)
print(mean_squared_error)
```

```
0.000940216989064446
```

Conclusion

The Score of Linear Regression :-0.058616121876582605 The Score of Ridge Model:0.9999999999922101 The score of Lasso Model:0.999999999999993 The Score of Elastic Net :0.000940216989064446

In [ ]:
```

```