District wise rainfall:

PROBLEM STATEMENT:To predict the Rainfall based on 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 [2]:
```
df=pd.read_csv(r"C:\Users\DELL\Desktop\district wise rainfall normal.csv")
df
```

Out[2]:

	STATE_UT_NAME	DISTRICT	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG
0	ANDAMAN And NICOBAR ISLANDS	NICOBAR	107.3	57.9	65.2	117.0	358.5	295.5	285.0	271.9
1	ANDAMAN And NICOBAR ISLANDS	SOUTH ANDAMAN	43.7	26.0	18.6	90.5	374.4	457.2	421.3	423.1
2	ANDAMAN And NICOBAR ISLANDS	N & M ANDAMAN	32.7	15.9	8.6	53.4	343.6	503.3	465.4	460.9
3	ARUNACHAL PRADESH	LOHIT	42.2	80.8	176.4	358.5	306.4	447.0	660.1	427.8
4	ARUNACHAL PRADESH	EAST SIANG	33.3	79.5	105.9	216.5	323.0	738.3	990.9	711.2
...	...	...	...	...	...	...	...	...	...	...
636	KERALA	IDUKKI	13.4	22.1	43.6	150.4	232.6	651.6	788.9	527.3
637	KERALA	KASARGOD	2.3	1.0	8.4	46.9	217.6	999.6	1108.5	636.3
638	KERALA	PATHANAMTHITTA	19.8	45.2	73.9	184.9	294.7	556.9	539.9	352.7
639	KERALA	WAYANAD	4.8	8.3	17.5	83.3	174.6	698.1	1110.4	592.9
640	LAKSHADWEEP	LAKSHADWEEP	20.8	14.7	11.8	48.9	171.7	330.2	287.7	217.5

641 rows × 19 columns

Data cleaning & preprocessing:

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

Out[3]:

	STATE_UT_NAME	DISTRICT	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OC
0	ANDAMAN And NICOBAR ISLANDS	NICOBAR	107.3	57.9	65.2	117.0	358.5	295.5	285.0	271.9	354.8	326.
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	301.
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	276.
3	ARUNACHAL PRADESH	LOHIT	42.2	80.8	176.4	358.5	306.4	447.0	660.1	427.8	313.6	167.
4	ARUNACHAL PRADESH	EAST SIANG	33.3	79.5	105.9	216.5	323.0	738.3	990.9	711.2	568.0	206.

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

Out[4]:

	STATE_UT_NAME	DISTRICT	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	
636	KERALA	IDUKKI	13.4	22.1	43.6	150.4	232.6	651.6	788.9	527.3	3
637	KERALA	KASARGOD	2.3	1.0	8.4	46.9	217.6	999.6	1108.5	636.3	2
638	KERALA	PATHANAMTHITTA	19.8	45.2	73.9	184.9	294.7	556.9	539.9	352.7	2
639	KERALA	WAYANAD	4.8	8.3	17.5	83.3	174.6	698.1	1110.4	592.9	2
640	LAKSHADWEEP	LAKSHADWEEP	20.8	14.7	11.8	48.9	171.7	330.2	287.7	217.5	1

In [5]:
```
df.shape
```

Out[5]: (641, 19)

In [6]:
```
df.describe()
```

Out[6]:

	JAN	FEB	MAR	APR	MAY	JUN	JUL	
count	641.000000	641.000000	641.000000	641.000000	641.000000	641.000000	641.000000	6
mean	18.355070	20.984399	30.034789	45.543214	81.535101	196.007332	326.033697	2
std	21.082806	27.729596	45.451082	71.556279	111.960390	196.556284	221.364643	1
min	0.000000	0.000000	0.000000	0.000000	0.900000	3.800000	11.600000	
25%	6.900000	7.000000	7.000000	5.000000	12.100000	68.800000	206.400000	1
50%	13.300000	12.300000	12.700000	15.100000	33.900000	131.900000	293.700000	2
75%	19.200000	24.100000	33.200000	48.300000	91.900000	226.600000	374.800000	3
max	144.500000	229.600000	367.900000	554.400000	733.700000	1476.200000	1820.900000	15

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.columns
```

Out[8]:
```
Index(['STATE_UT_NAME', 'DISTRICT', 'JAN', 'FEB', 'MAR', 'APR', 'MAY', 'JUN',
       'JUL', 'AUG', 'SEP', 'OCT', 'NOV', 'DEC', 'ANNUAL', 'Jan-Feb',
       'Mar-May', 'Jun-Sep', 'Oct-Dec'],
      dtype='object')
```

In [9]:
```
df.isnull().sum()
```

Out[9]:
```
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
```

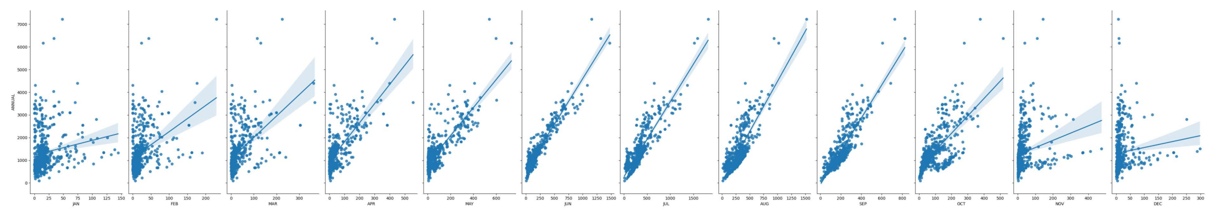
In [10]:
```
df["Jan-Feb"].value_counts()
```

Out[10]:
```
Jan-Feb
32.7     9
18.2     5
21.4     5
0.8      5
17.5     5
        ..
107.7    1
87.0     1
101.0    1
135.2    1
65.0     1
Name: count, Length: 399, dtype: int64
```

Data Visualisation:

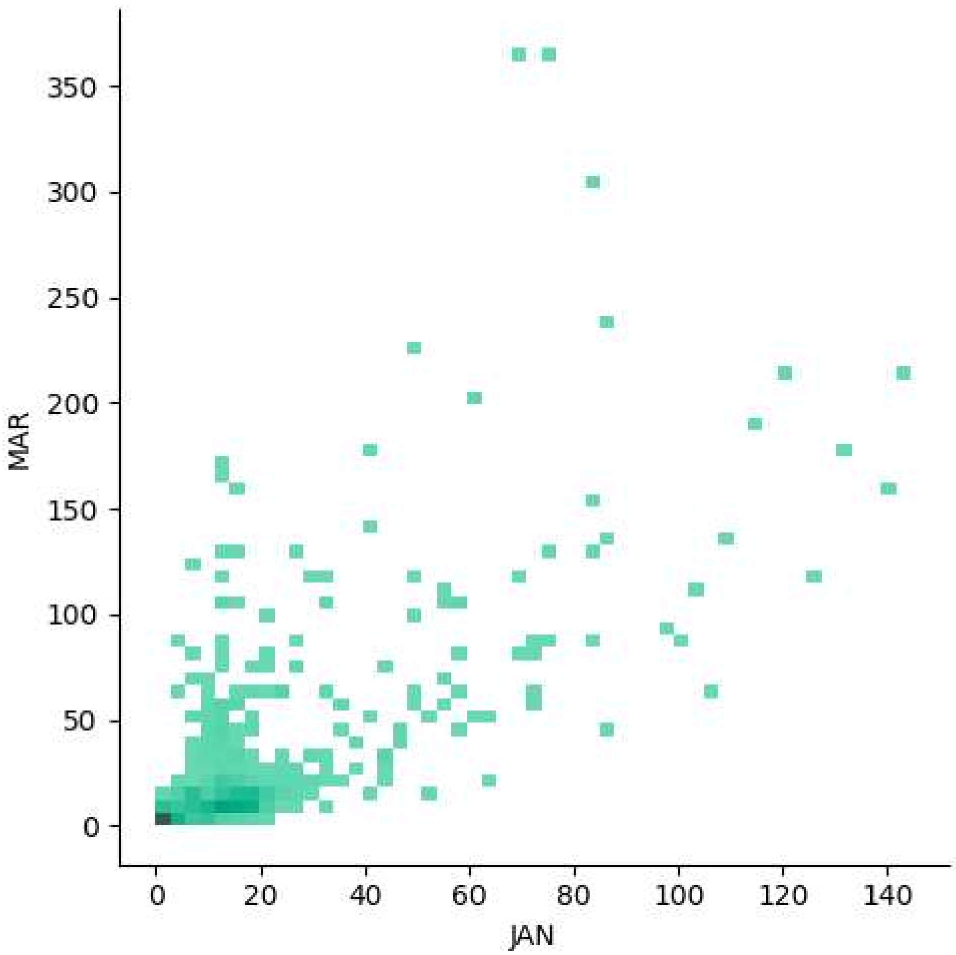
In [11]:
```
sns.pairplot(df,x_vars=['JAN', 'FEB', 'MAR', 'APR', 'MAY', 'JUN', 'JUL',
 'AUG', 'SEP', 'OCT', 'NOV', 'DEC'],y_vars=['ANNUAL'],height=7,aspect=0.5,kind=
```

Out[11]:
```
<seaborn.axisgrid.PairGrid at 0x209840389d0>
```



In [12]:
```
df=df[['JAN','FEB','MAR','APR','DEC']]
sns.heatmap(df.corr(),annot=True)
plt.show()
```

In [13]:
```
sns.displot(x='JAN',y='MAR',data=df,color='aquamarine')
plt.show()
```



In [14]:
```
x=df[['JAN']]
y=df['APR']
```

Linear Regression:

In [15]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=1
```

In [16]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)
regr=LinearRegression()
regr.fit(x_train,y_train)
```

Out[16]:
```
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 [17]:
```
score=regr.score(x_test,y_test)
print(score)
```

```
0.1200065273449159
```

Ridge:

In [18]:
```
from sklearn.linear_model import Lasso,RidgeCV,Ridge
from sklearn.preprocessing import StandardScaler
```

In [19]:
```
feature=df.columns[0:3]
target=df.columns[-1]
```

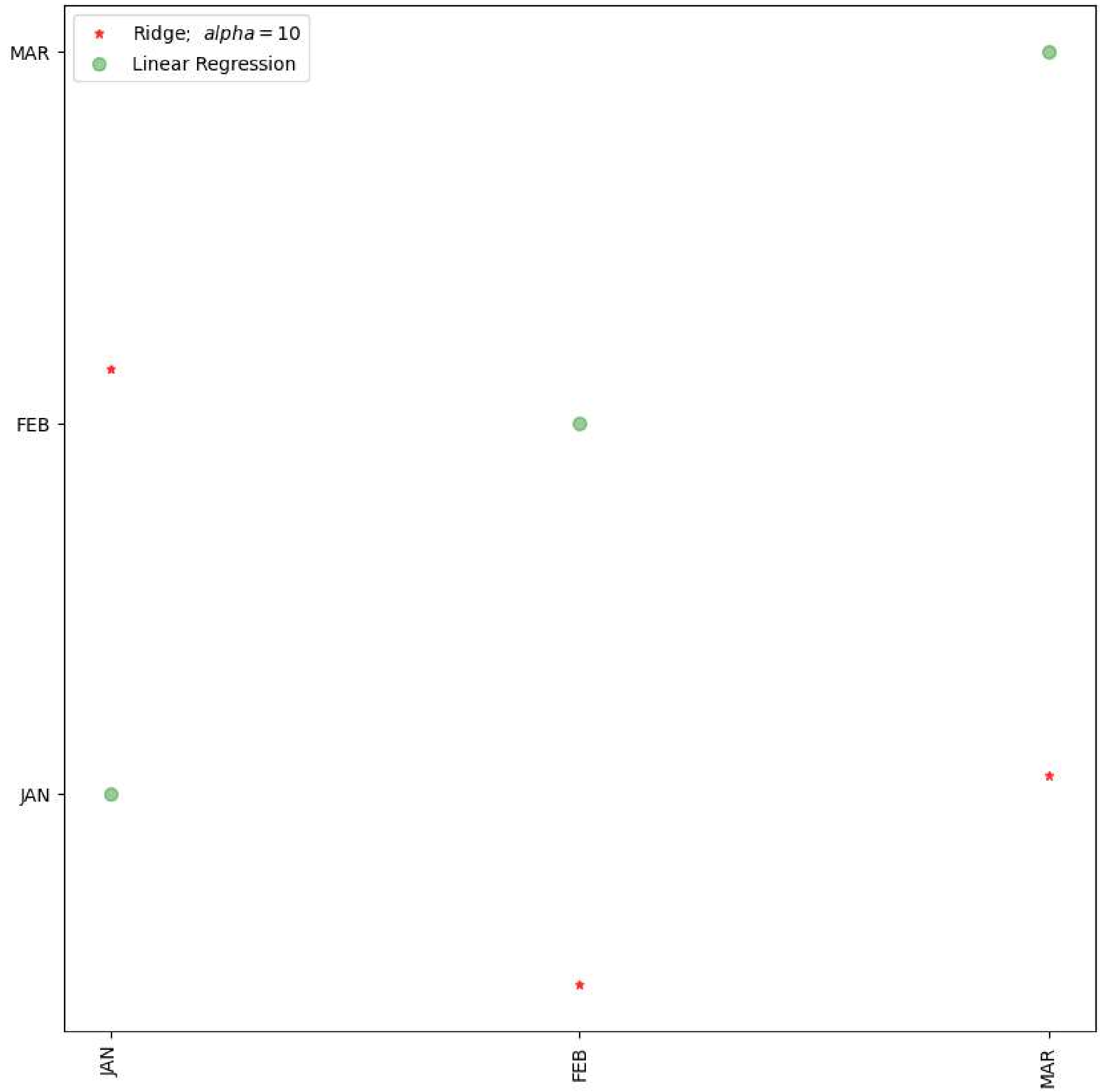
In [20]:
```
x= df[feature].values
y= df[target].values
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=1
```

In [21]:
```
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)
print("\n Ridge Model \n")
print("train score for ridge model is {}".format(train_score_ridge))
print("test score for ridge model is {}".format(test_score_ridge))
```

```
 Ridge Model 

train score for ridge model is 0.18674498377799043
test score for ridge model is 0.19747287933454494
```

In [22]:
```
plt.figure(figsize=(10,10))
plt.plot(feature,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersiz
plt.plot(feature,alpha=0.4,linestyle='none',marker='o',markersize=7,color='gree
plt.xticks(rotation=90)
plt.legend()
plt.show()
```



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

```
0.19747287933454494
```

Lasso:

In [24]:
```
lassoReg=Lasso(alpha=10)
lassoReg.fit(x_train,y_train)
train_score_lasso=lassoReg.score(x_train,y_train)
test_score_lasso=lassoReg.score(x_test,y_test)
print("\n Lasso Model \n")
print("train score for lasso model is {}".format(train_score_lasso))
print("test score for lasso model is {}".format(test_score_lasso))
```

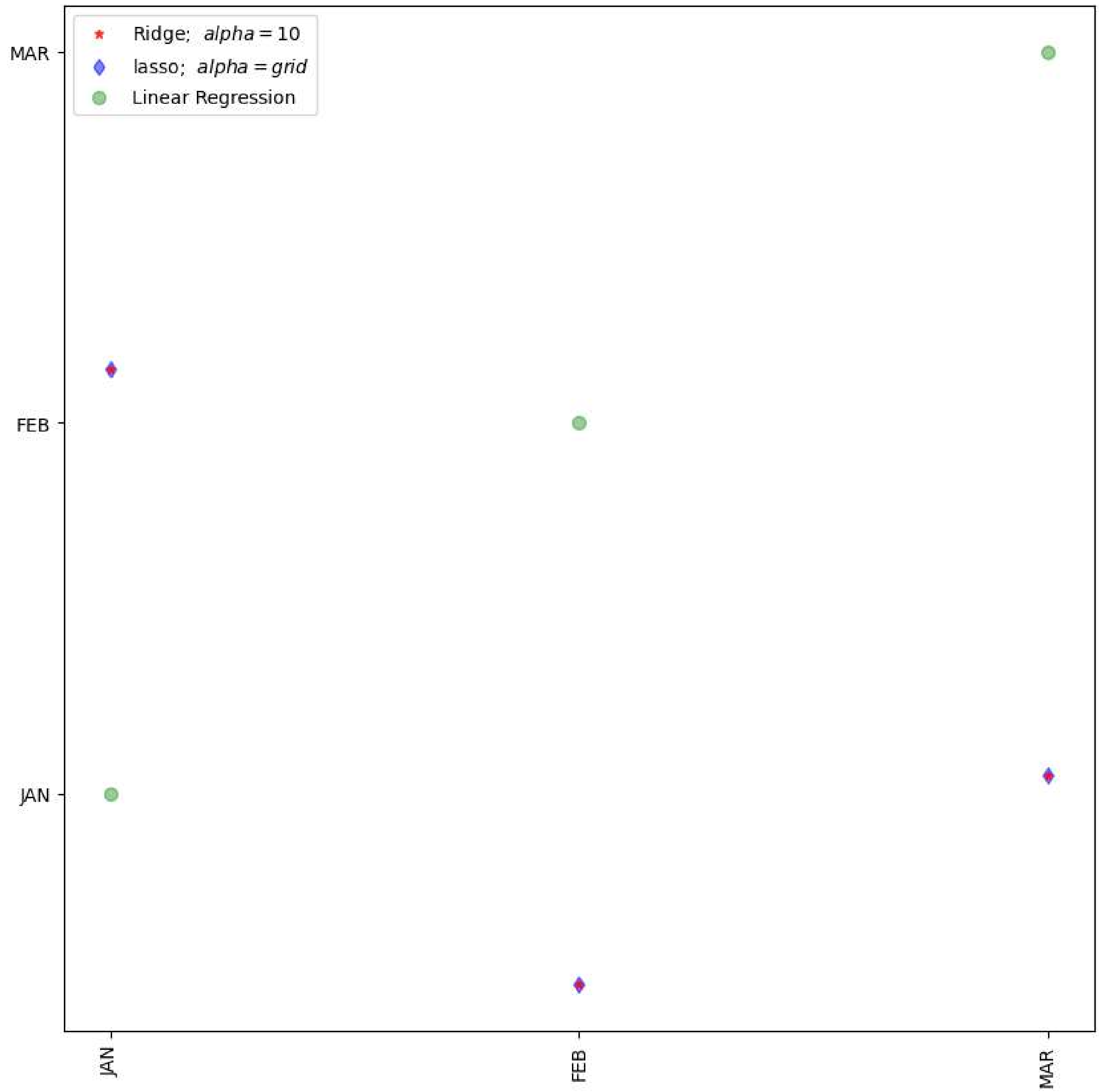
```
 Lasso Model 

train score for lasso model is 0.1833839875827794
test score for lasso model is 0.20227481138935577
```

In [25]:
```
from sklearn.linear_model import LassoCV
lasso_cv=LassoCV(alphas=[0.001,0.001,0.01,0.1,1,10]).fit(x_train,y_train)
print("The train score for lasso model is {}".format(lasso_cv.score(x_train,y_t
print("The train score for lasso model is {}".format(lasso_cv.score(x_test,y_te
```

```
The train score for lasso model is 0.18674499707273973
The train score for lasso model is 0.19746402052680911
```

In [26]:
```
plt.figure(figsize=(10,10))
plt.plot(feature,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersiz
plt.plot(lasso_cv.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,colc
plt.plot(feature,alpha=0.4,linestyle='none',marker='o',markersize=7,color='gree
plt.xticks(rotation=90)
plt.legend()
plt.show()
```



In [27]:
```
print(lassoReg.score(x_test,y_test))
```

0.20227481138935577

Elastic Net:

In [28]:
```
from sklearn.linear_model import ElasticNet
```

In [29]:
```
regr=ElasticNet()
regr.fit(x,y)
print(regr.coef_)
print(regr.intercept_)
```

```
[ 1.06176358 -0.47019381  0.08183614]
6.0709164259899975
```

In [30]:
```
y_pred_elastic=regr.predict(x_train)
mean_squared_error=np.mean((y_pred_elastic-y_train)**2)
print(mean_squared_error)
```

```
825.5143097001077
```

In [31]:
```
print(regr.score(x_test,y_test))
```

```
0.2100200576179274
```

CONCLUSION:Among on accuracy scores of all models that were implemented we can conclude that "Elastic Net Model" is the best model for the given dataset.