657Aass2CM1 (1)

July 18, 2021

1 [CM1] Data Pre-processing and Preparation

Importing all necessary libraries.

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.model_selection import KFold
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn import tree
import scipy
```

[2]:
```
df = pd.read_csv('dkmacovid_train.csv')
```

[3]:
```
df.shape
```

[3]:
```
(1380, 17)
```

[4]:
```
df.head(47)
```

[4]:
```
    Day  State ID                 State      Lat      Long_   Active  \
0     2         1               Alabama  32.3182   -86.9023   162449
1     2         2                Alaska  61.3707  -152.4044    40421
2     2         3               Arizona  33.7298  -111.4312   452222
3     2         4              Arkansas  34.9697   -92.3731    24012
4     2         5            California  36.1162  -119.6816  2362015
5     2         6              Colorado  39.0598  -105.3111   316043
6     2         7           Connecticut  41.5978   -72.7554   174221
7     2         8              Delaware  39.3185   -75.5071    39092
8     2         9  District of Columbia  38.8974   -77.0268     7715
9     2        10               Florida  27.7663   -81.6868  1332943
10    2        11               Georgia  33.0406   -83.6431   674162
```

11	2	12	Hawaii	21.0943	-157.4983	9934
12	2	13	Idaho	44.2405	-114.4788	82102
13	2	15	Indiana	39.8494	-86.2583	167279
14	2	17	Kansas	38.5266	-96.7265	222830
15	2	18	Kentucky	37.6681	-84.6701	233999
16	2	19	Louisiana	31.1695	-91.8678	44075
17	2	20	Maine	44.6939	-69.3819	13449
18	2	21	Maryland	39.0639	-76.8021	267830
19	2	22	Massachusetts	42.2302	-71.5301	110007
20	2	23	Michigan	43.3266	-84.5361	161204
21	2	24	Minnesota	45.6945	-93.9002	14197
22	2	25	Mississippi	32.7416	-89.6787	48174
23	2	26	Missouri	38.4561	-92.2884	404588
24	2	27	Montana	46.9219	-110.4544	4999
25	2	28	Nebraska	41.1254	-98.2681	56888
26	2	29	Nevada	38.3135	-117.0554	225723
27	2	30	New Hampshire	43.4525	-71.5639	6490
28	2	31	New Jersey	40.2989	-74.5210	412610
29	2	32	New Mexico	34.8405	-106.2485	75272
30	2	33	New York	42.1657	-74.9481	869564
31	2	34	North Carolina	35.6301	-79.8064	148057
32	2	35	North Dakota	47.5289	-99.7840	1999
33	2	36	Ohio	40.3888	-82.7649	132015
34	2	37	Oklahoma	35.5653	-96.9289	33687
35	2	38	Oregon	44.5720	-122.0709	108986
36	2	39	Pennsylvania	40.5908	-77.2098	190870
37	2	40	Rhode Island	41.6809	-71.5118	80791
38	2	41	South Carolina	33.8569	-80.9450	154798
39	2	42	South Dakota	44.2998	-99.4388	5733
40	2	45	Utah	40.1500	-111.8624	50073
41	2	46	Vermont	44.0459	-72.7107	2362
42	2	47	Virginia	37.7693	-78.1700	322456
43	2	49	West Virginia	38.4912	-80.9545	26834
44	2	50	Wisconsin	44.2685	-89.6165	65894
45	2	51	Wyoming	42.7560	-107.3025	1098
46	3	1	Alabama	32.3182	-86.9023	164924

	Incident_Rate	Total_Test_Results	Case_Fatality_Ratio	Testing_Rate	\
0	7535.061394	1891468	1.318688	38576.31315	
1	6534.252848	1290349	0.449781	176386.82510	
2	7407.212013	5218721	1.680608	39916.14181	
3	7669.219075	2079788	1.611203	68917.26567	
4	6045.109130	33391442	1.111215	84509.14544	
5	5889.695239	4474747	1.448233	77703.63149	
6	5332.530032	4383361	3.207974	122945.53010	
7	6045.920778	991318	1.579671	101802.69550	
8	4181.231571	911378	2.683927	129136.27930	

9	6308.080782	15950750	1.615697	74266.43692
10	6452.808747	5436988	1.599715	51208.16982
11	1566.596415	822805	1.302917	58112.95089
12	7957.125230	545485	1.018291	30524.07159
13	7769.949254	5769273	1.607754	85696.45462
14	7905.567337	1012506	1.246993	34754.44116
15	6132.275124	3210804	0.984779	71867.48001
16	6781.866437	4214182	2.375069	90651.08069
17	1878.051974	1112457	1.418103	82759.04396
18	4683.856903	5807666	2.107207	96063.07314
19	5573.896740	11046093	3.254195	160262.43300
20	5388.291832	8122575	2.472678	81332.64550
21	7408.852209	5387936	1.301001	95537.01376
22	7401.410346	1347935	2.197233	45291.24718
23	6685.292276	3679665	1.393354	59954.51189
24	7667.073985	792779	1.184956	74176.20872
25	8698.371802	1766104	0.991905	91299.45699
26	7482.315831	2111559	1.365922	68553.63819
27	3324.677082	1032929	1.701102	75966.80471
28	5498.328678	7798225	3.928767	87796.19666
29	6933.278775	1995329	1.743030	95159.35730
30	5200.410352	25706759	3.783169	132144.23310
31	5324.490155	7079384	1.234159	67499.30683
32	12189.428160	1290077	1.410255	169287.66950
33	6114.012199	7819008	1.261696	66891.44588
34	7481.859230	2690373	0.853558	67990.71815
35	2758.540895	2652670	1.282360	62893.20553
36	5159.979438	7675010	2.449211	30829.58437
37	8302.080216	1974498	2.020489	186385.75520
38	6073.710833	3204255	1.721999	62234.08408
39	11284.461020	373946	1.503571	42270.07242
40	8785.330313	2268187	0.459429	70749.11774
41	1232.233261	708246	1.807777	113502.96240
42	4203.083609	4337939	1.426322	50822.20542
43	4984.356752	1552160	1.537049	86608.96679
44	8993.695764	5402674	1.003720	92790.64391
45	7701.478508	501784	0.982658	86699.99084
46	7585.559182	1900070	1.310179	38751.75014

	Resident Population 2020 Census	Population Density 2020 Census	\
0	5,024,279	99.2	
1	733,391	1.3	
2	7,151,502	62.9	
3	3,011,524	57.9	
4	39,538,223	253.7	
5	5,773,714	55.7	
6	3,605,944	744.7	

7	989,948	508
8	689,545	11,280.00
9	21,538,187	401.4
10	10,711,908	185.6
11	1,455,271	226.6
12	1,839,106	22.3
13	6,785,528	189.4
14	2,937,880	35.9
15	4,505,836	114.1
16	4,657,757	107.8
17	1,362,359	44.2
18	6,177,224	636.1
19	7,029,917	901.2
20	10,077,331	178
21	5,706,494	71.7
22	2,961,279	63.1
23	6,154,913	89.5
24	1,084,225	7.4
25	1,961,504	25.5
26	3,104,614	28.3
27	1,377,529	153.8
28	9,288,994	1,263.00
29	2,117,522	17.5
30	20,201,249	428.7
31	10,439,388	214.7
32	779,094	11.3
33	11,799,448	288.8
34	3,959,353	57.7
35	4,237,256	44.1
36	13,002,700	290.6
37	1,097,379	1,061.40
38	5,118,425	170.2
39	886,667	11.7
40	3,271,616	39.7
41	643,077	69.8
42	8,631,393	218.6
43	1,793,716	74.6
44	5,893,718	108.8
45	576,851	5.9
46	5,024,279	99.2

	Density Rank 2020 Census	SexRatio	Confirmed	Deaths	Recovered
0	29	94	True	False	False
1	52	109	True	True	False
2	35	99	True	True	True
3	36	96	True	True	True
4	13	99	True	True	False

5	39	101	True	True	True
6	6	95	True	True	False
7	8	94	True	True	False
8	1	96	True	True	True
9	10	96	True	True	False
10	19	95	True	True	False
11	15	101	True	False	False
12	46	101	True	True	False
13	18	97	True	True	True
14	43	99	True	True	False
15	25	97	True	True	True
16	28	96	False	False	False
17	40	96	True	True	True
18	7	94	True	True	True
19	5	94	True	True	False
20	20	97	True	True	True
21	32	99	True	True	True
22	34	94	True	True	False
23	30	96	True	True	False
24	50	101	True	True	True
25	45	100	True	True	True
26	44	101	True	True	False
27	23	98	True	True	True
28	2	95	True	True	True
29	47	98	True	True	True
30	9	94	True	True	True
31	17	95	True	True	False
32	49	105	True	True	True
33	12	96	True	True	True
34	37	98	True	True	True
35	41	98	True	True	False
36	11	96	True	True	True
37	3	95	False	True	False
38	21	94	True	True	True
39	48	102	True	True	True
40	42	101	True	True	True
41	33	97	True	True	True
42	16	97	True	True	True
43	31	98	True	True	True
44	27	99	True	True	True
45	51	104	True	False	True
46	29	94	True	True	False

1.1 We can see that states with State Id 14, 16, 43, 44 and 48 are missing in the data. That's why the total count of states in the data is 46.

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

[5]:
```
      Day  State ID          State      Lat     Long_  Active  Incident_Rate  \
1375   31        46        Vermont  44.0459  -72.7107    3537    1917.501751
1376   31        47       Virginia  37.7693  -78.1700  457993    5913.864172
1377   31        49  West Virginia  38.4912  -80.9545   21195    6751.734093
1378   31        50      Wisconsin  44.2685  -89.6165   68537   10169.973590
1379   31        51        Wyoming  42.7560 -107.3025    1313    8969.536543

      Total_Test_Results  Case_Fatality_Ratio  Testing_Rate  \
1375              897351             1.454242  143808.78510
1376             5234155             1.280560   61322.04732
1377             1945579             1.672713  108561.35130
1378             6177575             1.086567  106099.52810
1379              634985             1.148097  109714.92450

     Resident Population 2020 Census Population Density 2020 Census  \
1375                         643,077                           69.8
1376                       8,631,393                          218.6
1377                       1,793,716                           74.6
1378                       5,893,718                          108.8
1379                         576,851                            5.9

      Density Rank 2020 Census  SexRatio  Confirmed  Deaths  Recovered
1375                        33        97       True    True       True
1376                        16        97       True    True       True
1377                        31        98       True    True       True
1378                        27        99       True    True       True
1379                        51       104       True   False       True
```

[6]:
```
df.describe(include = 'all')
```

[6]:
```
                Day     State ID           State          Lat        Long_  \
count   1380.000000  1380.000000            1380  1380.000000  1380.000000
unique          NaN          NaN              46          NaN          NaN
top             NaN          NaN  South Carolina          NaN          NaN
freq            NaN          NaN              30          NaN          NaN
mean      16.500000    25.239130             NaN    39.470717   -92.879928
std        8.658579    14.513405             NaN     6.070494    19.632514
min        2.000000     1.000000             NaN    21.094300  -157.498300
25%        9.000000    12.000000             NaN    35.630100  -105.311100
50%       16.500000    25.500000             NaN    39.583950   -88.259400
75%       24.000000    37.000000             NaN    43.326600   -77.209800
max       31.000000    51.000000             NaN    61.370700   -69.381900
```

	Active	Incident_Rate	Total_Test_Results	Case_Fatality_Ratio	\
count	1.380000e+03	1380.000000	1.380000e+03	1380.000000	
unique	NaN	NaN	NaN	NaN	
top	NaN	NaN	NaN	NaN	
freq	NaN	NaN	NaN	NaN	
mean	2.610390e+05	7203.192905	5.271097e+06	1.631757	
std	4.914059e+05	2305.025102	6.991478e+06	0.656702	
min	9.550000e+02	1232.233261	3.739460e+05	0.439598	
25%	2.731600e+04	6042.134459	1.310515e+06	1.246993	
50%	1.005915e+05	7453.675956	2.919566e+06	1.499993	
75%	2.592418e+05	8621.924085	6.093790e+06	1.817013	
max	3.283336e+06	12811.162350	4.227902e+07	3.928767	

	Testing_Rate	Resident Population 2020 Census	\
count	1380.000000	1380	
unique	NaN	46	
top	NaN	5,773,714	
freq	NaN	30	
mean	91763.237514	NaN	
std	40858.185997	NaN	
min	30524.071590	NaN	
25%	67457.197525	NaN	
50%	85438.613770	NaN	
75%	104509.453475	NaN	
max	235733.711200	NaN	

	Population Density 2020 Census	Density Rank 2020 Census	SexRatio	\
count	1380	1380.000000	1380.000000	
unique	46	NaN	NaN	
top	1.3	NaN	NaN	
freq	30	NaN	NaN	
mean	NaN	27.173913	97.760870	
std	NaN	15.378197	3.219219	
min	NaN	1.000000	94.000000	
25%	NaN	13.000000	95.000000	
50%	NaN	28.500000	97.000000	
75%	NaN	41.000000	99.000000	
max	NaN	52.000000	109.000000	

	Confirmed	Deaths	Recovered
count	1380	1380	1380
unique	2	2	2
top	True	True	True
freq	1329	1244	864
mean	NaN	NaN	NaN
std	NaN	NaN	NaN
min	NaN	NaN	NaN

```
25%              NaN     NaN         NaN
50%              NaN     NaN         NaN
75%              NaN     NaN         NaN
max              NaN     NaN         NaN
```

1.1.1 As the columns 'Resident Population 2020 Census' and 'Population Density 2020 Census' have commas in the data , we will be removing it for computation purposes.

[7]:
```
df['Resident Population 2020 Census'] = df['Resident Population 2020 Census'].
 ↪str.replace(',','').astype(int)
df['Population Density 2020 Census'] = df['Population Density 2020 Census'].str.
 ↪replace(',','').astype(float)
```

[8]:
```
for i in set(df.loc[:,"State"]):
    lat = set(df[df.loc[:,"State"]==i].loc[:,"Lat"])
    lon = set(df[df.loc[:,"State"]==i].loc[:,"Long_"])
    print(i,"\t",lat,lon)
print("No of sets ",len(set(df.loc[:,"State"])))
```

```
New Mexico       {34.8405} {-106.2485}
Massachusetts    {42.2302} {-71.5301}
Oklahoma         {35.5653} {-96.9289}
Michigan         {43.3266} {-84.5361}
Pennsylvania     {40.5908} {-77.2098}
Wisconsin        {44.2685} {-89.6165}
Indiana          {39.8494} {-86.2583}
Mississippi      {32.7416} {-89.6787}
Virginia         {37.7693} {-78.17}
Alaska   {61.3707} {-152.4044}
New Jersey       {40.2989} {-74.521}
Ohio     {40.3888} {-82.7649}
Minnesota        {45.6945} {-93.9002}
Oregon   {44.572} {-122.0709}
California       {36.1162} {-119.6816}
Louisiana        {31.1695} {-91.8678}
Arkansas         {34.9697} {-92.3731}
Vermont          {44.0459} {-72.7107}
Delaware         {39.3185} {-75.5071}
Colorado         {39.0598} {-105.3111}
Alabama          {32.3182} {-86.9023}
New Hampshire    {43.4525} {-71.5639}
Georgia          {33.0406} {-83.6431}
Nebraska         {41.1254} {-98.2681}
Hawaii   {21.0943} {-157.4983}
Connecticut      {41.5978} {-72.7554}
Florida          {27.7663} {-81.6868}
Kentucky         {37.6681} {-84.6701}
```

```
West Virginia    {38.4912} {-80.9545}
Kansas   {38.5266} {-96.7265}
Nevada   {38.3135} {-117.0554}
South Dakota     {44.2998} {-99.4388}
North Carolina   {35.6301} {-79.8064}
Utah     {40.15} {-111.8624}
Missouri         {38.4561} {-92.2884}
Idaho    {44.2405} {-114.4788}
Wyoming          {42.756} {-107.3025}
Rhode Island     {41.6809} {-71.5118}
Maryland         {39.0639} {-76.8021}
Maine    {44.6939} {-69.3819}
North Dakota     {47.5289} {-99.784}
District of Columbia     {38.8974} {-77.0268}
New York         {42.1657} {-74.9481}
Arizona          {33.7298} {-111.4312}
Montana          {46.9219} {-110.4544}
South Carolina   {33.8569} {-80.945}
No of sets  46
```

1.1.2 From the above, we can say that every State corresponds to a unique set of co-ordinates represented by Latitude and Longitude. We'll keep the Lat and Long for better data understanding.

1. Check for Null values

[9]:
```
df.isna().sum()
```

[9]:
```
Day                               0
State ID                          0
State                             0
Lat                               0
Long_                             0
Active                            0
Incident_Rate                     0
Total_Test_Results                0
Case_Fatality_Ratio               0
Testing_Rate                      0
Resident Population 2020 Census   0
Population Density 2020 Census    0
Density Rank 2020 Census          0
SexRatio                          0
Confirmed                         0
Deaths                            0
Recovered                         0
dtype: int64
```

No Null values in the dataset as mentioned in the question

2. Checking for negative values

[10]:
```
(df.iloc[:,3:-3]<0).sum()
```

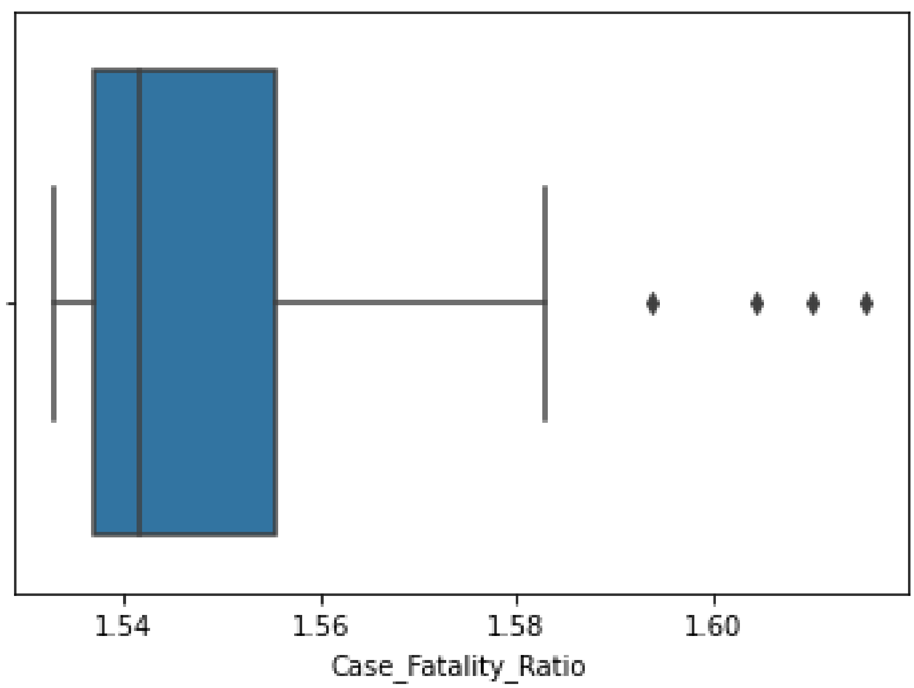
[10]:
```
Lat                                0
Long_                           1380
Active                             0
Incident_Rate                      0
Total_Test_Results                 0
Case_Fatality_Ratio                0
Testing_Rate                       0
Resident Population 2020 Census    0
Population Density 2020 Census     0
Density Rank 2020 Census           0
SexRatio                           0
dtype: int64
```

No negative values in the dataset.

1.2 3. Checking Outliers and Removing them

[11]:
```
df_gstate = df.groupby('State')
z = df_gstate.get_group('Florida')['Case_Fatality_Ratio']
sns.boxplot(x = z)
```

[11]:
```
<AxesSubplot:xlabel='Case_Fatality_Ratio'>
```

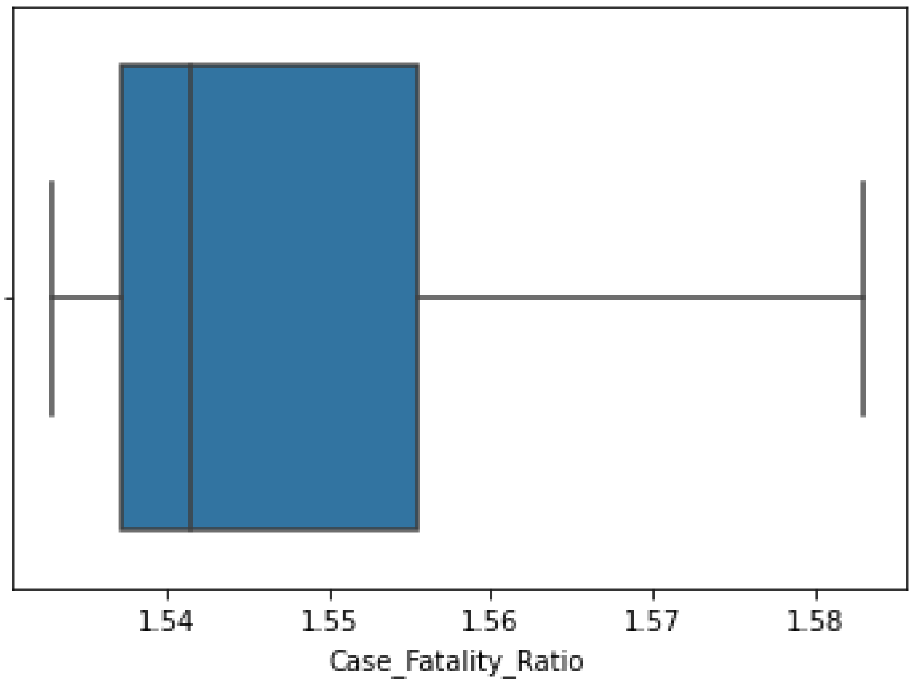


[12]:
```
#The outliers are checked and removed by grouping the dataset according to the␣
↪State
for key,value in df_gstate:
    groups = df_gstate.get_group(key)
    temp = groups.iloc[:,5:10]
    for columns in temp:
        Q1 = np.percentile(temp[columns],25)
        Q3 = np.percentile(temp[columns],75)
        IQR = Q3 - Q1
        right_limit = Q3 + 1.5*IQR
        left_limit = Q1 - 1.5*IQR
        outlier_right_index = groups[groups[columns] > right_limit][columns].
↪index
        outlier_left_index = groups[groups[columns] < left_limit][columns].index
        n_outliers = len(outlier_right_index) + len(outlier_left_index)
        if(n_outliers > 0):
            print(key,columns,n_outliers)
            df.loc[outlier_right_index,columns] = right_limit
            df.loc[outlier_left_index,columns] = left_limit
```

```
Florida Case_Fatality_Ratio 4
Hawaii Case_Fatality_Ratio 6
Indiana Case_Fatality_Ratio 1
Maine Case_Fatality_Ratio 1
Mississippi Case_Fatality_Ratio 1
Montana Active 3
Montana Case_Fatality_Ratio 3
Nebraska Active 1
Nebraska Case_Fatality_Ratio 1
Ohio Active 2
Utah Active 3
```

[13]:
```
z = df_gstate.get_group('Florida')['Case_Fatality_Ratio']
sns.boxplot(x = z)
```

[13]:
```
<AxesSubplot:xlabel='Case_Fatality_Ratio'>
```



1.3 What do you do with ”Day”, ”State” and ”State ID”?

[14]:
```
print(df.State.nunique())
print(df['State ID'].nunique())
print(df.Day.nunique())
```

```
46
46
30
```

The dataset contains data for day 2 to 31 total 30 days of covid data for 46 unique states.

1.4 As State ID and State both are giving a unique identity to the dataset we can remove state column.

[15]:
```
#dropping State Id Column
df = df.drop(columns=['State'])
```

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

[16]:
```
Index(['Day', 'State ID', 'Lat', 'Long_', 'Active', 'Incident_Rate',
       'Total_Test_Results', 'Case_Fatality_Ratio', 'Testing_Rate',
       'Resident Population 2020 Census', 'Population Density 2020 Census',
       'Density Rank 2020 Census', 'SexRatio', 'Confirmed', 'Deaths',
       'Recovered'],
```

```
      dtype='object')
```

1.5 Normalization

[17]:
```python
normalization =␣
↪df[['Lat','Long_','Active','Incident_Rate','Total_Test_Results','Case_Fatality_Ratio',
            'Testing_Rate','Resident Population 2020 Census','Population␣
↪Density 2020 Census',
            'Density Rank 2020 Census','SexRatio']]
normalization.dtypes
```

[17]:
```
Lat                                float64
Long_                              float64
Active                             float64
Incident_Rate                      float64
Total_Test_Results                   int64
Case_Fatality_Ratio                float64
Testing_Rate                       float64
Resident Population 2020 Census      int32
Population Density 2020 Census     float64
Density Rank 2020 Census             int64
SexRatio                             int64
dtype: object
```

[18]:
```python
# Z-score normalization
normalization = (normalization - normalization.mean()) / normalization.std()
```

[19]:
```python
df[['Lat','Long_','Active','Incident_Rate','Total_Test_Results','Case_Fatality_Ratio',
            'Testing_Rate','Resident Population 2020 Census','Population␣
↪Density 2020 Census',
            'Density Rank 2020 Census','SexRatio']] = normalization
df
```

[19]:
```
      Day  State ID       Lat     Long_    Active  Incident_Rate  \
0       2         1 -1.178243  0.304476 -0.200641       0.143976
1       2         2  3.607611 -3.031933 -0.448967      -0.290209
2       2         3 -0.945708 -0.944926  0.389043       0.088511
3       2         4 -0.741458  0.025816 -0.482359       0.202178
4       2         5 -0.552594 -1.365168  4.275448      -0.502417
…    …        …         …         …         …             …
1375   31        46  0.753675  1.027338 -0.524025      -2.293117
1376   31        47 -0.280277  0.749264  0.400787      -0.559356
1377   31        49 -0.161357  0.607433 -0.488091      -0.195859
1378   31        50  0.790345  0.166226 -0.391751       1.287093
1379   31        51  0.541189 -0.734627 -0.528551       0.766301

      Total_Test_Results  Case_Fatality_Ratio  Testing_Rate  \
```

```
0                -0.483393             -0.475230     -1.301745
1                -0.569371             -1.797949      2.071154
2                -0.007491              0.075713     -1.268952
3                -0.456457             -0.029941     -0.559153
4                 4.022089             -0.791062     -0.177543
…                   …                     …            …
1375             -0.625582             -0.268880      1.273810
1376             -0.005284             -0.533271     -0.745045
1377             -0.475653              0.063695      0.411132
1378              0.129655             -0.828583      0.350879
1379             -0.663109             -0.734918      0.439366

      Resident Population 2020 Census  Population Density 2020 Census  \
0                           -0.128579                       -0.217013
1                           -0.754174                       -0.276752
2                            0.181561                       -0.239163
3                           -0.422031                       -0.242214
4                            4.903416                       -0.122735
…                              …                               …
1375                        -0.767341                       -0.234953
1376                         0.397324                       -0.144153
1377                        -0.599582                       -0.232024
1378                        -0.001818                       -0.211155
1379                        -0.776996                       -0.273945

      Density Rank 2020 Census  SexRatio  Confirmed  Deaths  Recovered
0                     0.118745 -1.168255       True   False      False
1                     1.614369  3.491260       True    True      False
2                     0.508908  0.384916       True    True       True
3                     0.573935 -0.546987       True    True       True
4                    -0.921689  0.384916       True    True      False
…                        …         …       …      …        …
1375                  0.378854 -0.236352       True    True       True
1376                 -0.726607 -0.236352       True    True       True
1377                  0.248799  0.074282       True    True       True
1378                 -0.011309  0.384916       True    True       True
1379                  1.549342  1.938088       True   False       True

[1380 rows x 16 columns]
```

[20]:
```python
df.to_csv("cleaned_normalized_coviddata.csv",index = False)
```

2 Summary Report

1) The dataset that is imported goes though a series of preprocessing steps. It is initially checked for NAN and negative values, there were no NAN values in the dataset.

2) The 'Resident Population 2020 Census' and 'Population Density 2020 Census' were of type object because there were commas between numbers and thus the data was stored as string type. The commas were removed and these columns were converted to numerical datatype.

3) The datset is then checked for outliers. The outliers were to be considered based on grouping the dataset by 'State'. This resulted in 25 outliers which were replaced by their upper and lower limit values.

4) The state and stateId represented the same information so the 'State' column was dropped. We have kept the 'Day' and 'State ID' so that the data may be grouped according to state for any future reference in the following CM's. However, these values will not be used for calculation purposes.

5) We used z-score normalization since the columns would be normally distributed with a specified range of values and most of the classifiers calculate the distance between points for classification. Min Max scaler is not used as presence of outlier might affect its values and since the data is generated over a specific population, there might be chances of outliers.