COVID 19 MOVEMENT IN INDIA & BEST ALGORITHM FOR RISK ZONE PREDICTION IN WEST BENGAL

Import all required Libraries

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from sklearn.preprocessing import LabelEncoder, MinMaxScaler, StandardScaler
sns.set_style('white')
```

In [2]:
```
import plotly.express as px
from plotly.offline import init_notebook_mode, iplot
init_notebook_mode(True)
import plotly.offline as pyo
from plotly.offline import plot, iplot, init_notebook_mode,plot_mpl,iplot
pyo.init_notebook_mode()

pyo.init_notebook_mode(connected=True)
import folium
from folium import plugins
%matplotlib inline
%matplotlib inline # for jupyter graphs

# To display the updated plot-size and set the color scheme
plt.rcParams['figure.figsize'] = 8, 6
```

Read Dataset of Covid19 India

In [3]:
```
data_path=r'C:\Pro_Ana\Data C 19 IN UPDATED
df=pd.read_csv('C:\Pro_Ana\Data C 19 IN UPDATED.csv')
df.head(5)
```

Out[3]:

	Sno	Date	Time	State/UnionTerritory	ConfirmedIndianNational	ConfirmedForeignNational	Cured	Death	Confirmed
0	1	30-01-2020	06:00 PM	Kerala	1	0	0	0	1
1	2	31-01-2020	06:00 PM	Kerala	1	0	0	0	1
2	3	01-02-2020	06:00 PM	Kerala	1	0	0	0	1
3	4	02-02-2020	06:00 PM	Kerala	1	0	0	0	1
4	5	03-02-2020	06:00 PM	Kerala	1	0	0	0	1
5	6	04-02-2020	06:00 PM	Kerala	1	0	0	0	1
6	7	05-02-2020	06:00 PM	Kerala	1	0	0	0	1
7	8	06-02-2020	06:00 PM	Kerala	1	0	0	0	1
8	9	07-02-2020	06:00 PM	Kerala	1	0	0	0	1
9	10	08-02-2020	06:00 PM	Kerala	1	0	0	0	1
10	11	09-02-2020	06:00 PM	Kerala	1	0	0	0	1
11	12	10-02-2020	06:00 PM	Kerala	1	0	0	0	1
12	13	11-02-2020	06:00 PM	Kerala	1	0	0	0	1
13	14	12-02-2020	06:00 PM	Kerala	1	0	0	0	1
14	15	13-02-2020	06:00 PM	Kerala	1	0	0	0	1
15	16	14-02-2020	06:00 PM	Kerala	1	0	0	0	1
16	17	15-02-2020	06:00 PM	Kerala	1	0	0	0	1
17	18	16-02-2020	06:00 PM	Kerala	1	0	0	0	1
18	19	17-02-2020	06:00 PM	Kerala	1	0	0	0	1
19	20	18-02-2020	06:00 PM	Kerala	1	0	0	0	1
20	21	19-02-2020	06:00 PM	Kerala	1	0	0	0	1
21	22	20-02-2020	06:00 PM	Kerala	1	0	0	0	1
22	23	21-02-2020	06:00 PM	Kerala	1	0	0	0	1
23	24	22-02-2020	06:00 PM	Kerala	1	0	0	0	1
24	25	23-02-2020	06:00 PM	Kerala	1	0	0	0	1
25	26	24-02-2020	06:00 PM	Kerala	1	0	0	0	1
26	27	25-02-2020	06:00 PM	Kerala	1	0	0	0	1
27	28	26-02-2020	06:00 PM	Kerala	1	0	0	0	1
28	29	27-02-2020	06:00 PM	Kerala	1	0	0	0	1
29	30	28-02-2020	06:00 PM	Kerala	1	0	0	0	1
30	31	29-02-2020	06:00 PM	Kerala	1	0	0	0	1
31	32	01-03-2020	06:00 PM	Kerala	1	0	0	0	1
32	33	02-03-2020	06:00 PM	Kerala	1	0	0	0	1
33	34	03-03-2020	06:00 PM	Kerala	1	0	0	0	1
34	35	04-03-2020	06:00 PM	Kerala	1	0	0	0	1
35	36	05-03-2020	06:00 PM	Kerala	1	0	0	0	1
36	37	06-03-2020	06:00 PM	Kerala	1	0	0	0	1
37	38	07-03-2020	06:00 PM	Kerala	1	0	0	0	1
38	39	08-03-2020	06:00 PM	Kerala	1	0	0	0	1
39	40	09-03-2020	06:00 PM	Kerala	1	0	0	0	1
40	41	10-03-2020	06:00 PM	Kerala	1	0	0	0	1
41	42	11-03-2020	06:00 PM	Kerala	1	0	0	0	1
42	43	12-03-2020	06:00 PM	Kerala	1	0	0	0	1
43	44	13-03-2020	06:00 PM	Kerala	1	0	0	0	1
44	45	14-03-2020	06:00 PM	Kerala	1	0	0	0	1
45	46	15-03-2020	06:00 PM	Kerala	1	0	0	0	1
46	47	16-03-2020	06:00 PM	Kerala	1	0	0	0	1
47	48	17-03-2020	06:00 PM	Kerala	1	0	0	0	1
48	49	18-03-2020	06:00 PM	Kerala	1	0	0	0	1
49	50	19-03-2020	06:00 PM	Kerala	1	0	0	0	1

Preprocessing on dataset

In [4]:
```
# Data type of all column
# We need to convert Date to Object format
df.info()

class: pandas.core.frame.DataFrame'>
RangeIndex: 1085 entries, 0 to 1084
Data columns (total 9 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Sno                     1085 non-null   int64
 1   Date                    1085 non-null   object
 2   Time                    1085 non-null   object
 3   State/UnionTerritory    1085 non-null   object
 4   ConfirmedIndianNational 1085 non-null   object
 5   ConfirmedForeignNational 1085 non-null  object
 6   Cured                   1085 non-null   int64
 7   Death                   1085 non-null   int64
 8   Confirmed               1085 non-null   int64
dtypes: int64(5), object(4)
memory usage: 76.4 KB
```

In [5]:
```
# WHICH COLUMN HAS NULL VALUE FOR EACH COLUMN ?
df.isnull().sum()
```

Out[5]:
```
CHECK NULL VALUE FOR EACH COLUMN
Sno                       0
Date                      0
Time                      0
State/UnionTerritory      0
ConfirmedIndianNational   0
ConfirmedForeignNational  0
Cured                     0
Death                     0
Confirmed                 0
dtype: int64
```

In [6]:
```
# CHECK ANY VALUE IS MISSING OR NOT
df.isnull().sum()
```

Out[6]:
```
Sno                       0
Date                      0
Time                      0
State/UnionTerritory      0
ConfirmedIndianNational   0
ConfirmedForeignNational  0
Cured                     0
Death                     0
Confirmed                 0
dtype: int64
```

In [7]:
```
data_change1_column_head=df.rename({'State/UnionTerritory':'St',
                                    'ConfirmedIndianNational':'CIN',
                                    'ConfirmedForeignNational':'CFN'})

data_change1_column_head1
```

Out[7]:

	Sno	Date	Time	St	CIN	CFN	Cured	Death	Confirmed
0	1	30-01-2020	06:00 PM	Kerala	1	0	0	0	1
1	2	31-01-2020	06:00 PM	Kerala	1	0	0	0	1
2	3	01-02-2020	06:00 PM	Kerala	1	0	0	0	1
3	4	02-02-2020	06:00 PM	Kerala	1	0	0	0	1
4	5	03-02-2020	06:00 PM	Kerala	1	0	0	0	1

In [8]:
```
data_change1_column_count=data_change1.count()
```

In [9]:
```
data_state1=data_change1.St.value_counts()
print(data_state1, data_state1.count)
```

In [10]:
```
Total = data[['St',data_cols0]].
groupby('St').agg(sum).
round(0).
sort_values(ascending=False).
head(35).
style.
background_gradient(cmap='Blues')
```

Out[10]:

St	Total
Maharashtra	2547
Gujarat	1596
Telangana	1392
Karnataka	1017
Andhra Pradesh	1000
Uttar Pradesh	1000
Tamil Nadu	1000
Madhya Pradesh	1000
Rajasthan	1000
Bihar	1000
Odisha	1000
West Bengal	1000
Kerala	1000
Chhattisgarh	1000
Punjab	1000
Himachal Pradesh	1000
Haryana	1000
Jharkhand	1000
Uttarakhand	1000
Jammu and Kashmir Ladakh	1000
Goa	1000
Assam	1000
Chandigarh	1000
Tripura	1000
Puducherry	1000
Andaman and Nicobar Islands	1000
Manipur	1000
Meghalaya	1000
Himachal Pradesh	1000
Nagaland	1000
Sikkim	1000
Chandigarh	1000
Dadra and Nagar Haveli and Daman and Diu	1000
Ladakh	1000
Lakshadweep	1000
Arunachal Pradesh	1000
Mizoram	1000
Cases being reassigned to states	1000
Chhattisgarh	1000
Telangana	1000
Cases being reassigned to states	1000
Jammu and Kashmir	1000
Nagaland	1000
Manipur	1000
Mizoram	1000
Telangana	1000
Dadra and Nagar Haveli	1000
Dadra and Nagar Haveli	1000
Madhya Pradesh	1000
Telangana	1000

A Bar-Plot of per State which is total no of rows of Confirmed Cases

In [11]:
```
head_plot_barchart=df.head(5).plot(x='Date', y=['Cured', 'Death', 'Confirmed'],
                                   kind='bar')
```

Out[11]:

A Graphical Plot with coordinates for showing the Movement

In [12]:
```
# Visualization of Hotel Trivago and Fare
data_state1.sort_head_cost'Latitude'Longitude'].
head(5).
folium.Map(location=[20.593684, 78.96288],
zoom_start=4.).add_to(map).add'Latitude'Longitude'].head(5).add'Confirmed'].
head(5).add_to(map)
```

In [13]: *# create map and display it*

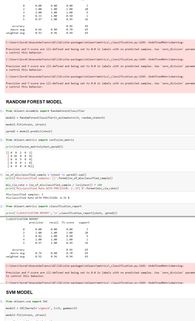
```
data_full = pd.merge(data_cases,data_confirm,on='Date')# merge 2 df
data_full = data_full[['Date','Cured','Death','Confirmed']]

for lat, lon, value, name in zip(data_full['Latitude'], data_full['Longitude'], data_full['Confirmed_Country'],
                                 data_full['Country/Region'.sample(2, random_state=1)
    folium.CircleMarker([lat, lon],
                        radius=value*0.05,
                        popup = ('<b>{}</b>'.format(name)+'<br>'
                                 'Confirmed Cases: {:.1f}'.format(value)+'<br>'),
                        color='red',

                        fill_color='red',
                        fill_opacity=0.7 ).add_to(map)

map
```

Out[13]:



Read WestBengal Covid19 Situation

In [14]:
```
data_path=r'C:\Pro_Ana\Data C 19 IN UPDATED
df=pd.read_csv('C:\Pro_Ana\Data C 19 IN UPDATED.csv')

Series obj.head(n=5)
df.head(5)
```

Out[14]:

	Sno	Date	Time	Cured
0	1	Haryana 2021-2020	100	
1	2	Chandigarh 2021-2020	100	
2	3	Jharkhand 2021-2020	100	
3	4	Karnataka 2021-2020	100	
4	5	Andhra Pradesh 2021-2020	100	
5	6	Uttar Pradesh 2021-2020	100	
6	7	Madhya Pradesh 2021-2020	100	
7	8	Rajasthan 2021-2020	100	
8	9	Bihar 2021-2020	100	
9	10	Odisha 2021-2020	100	
10	11	West Bengal 2021-2020	100	
11	12	Kerala 2021-2020	100	
12	13	Chhattisgarh 2021-2020	100	
13	14	Punjab 2021-2020	100	
14	15	Himachal Pradesh 2021-2020	100	
15	16	Haryana 2021-2020	100	
16	17	Jharkhand 2021-2020	100	
17	18	Uttarakhand 2021-2020	100	
18	19	Goa 2021-2020	100	

In [15]:
```
df.info()

class: pandas.core.frame.DataFrame'>
RangeIndex: 1085 entries, 0 to 1084
Data columns (total 4 columns):
 #   Sno     Non-Null Count  dtype
---  ------  --------------  -----
 0   Sno     1085 non-null   int64
 1   Date    1085 non-null   object
 2   Time    1085 non-null   object
 3   Cured   1085 non-null   int64
dtypes: int64(2), object(2)
memory usage: 34.0 KB
```

In [16]:
```
# df["Time"] = df["Time"].astype(int).astype(int)
# df.info()
```

Preprocessing

In [17]:
```
def map_function(x):
    if x == 0 and x < 100:
        return "Very Low"
    elif x >= 100 and x < 1000:
        return "Low"
    elif x >= 1000 and x < 10000:
        return "Medium"
    elif x >= 10000 and x < 100000:
        return "High"
    elif x >= 100000 and x < 1000000:
        return "Very High"
    else:
        return "Critical"

df["Cured"] = df["Cured"].apply(map_function)

df.head(10)
```

Out[17]:

	Sno	Date	Time	Cured
0	1	Haryana 2021-2020	100	Medium
1	2	Chandigarh 2021-2020	100	Medium
2	3	Jharkhand 2021-2020	100	Medium
3	4	Karnataka 2021-2020	100	Medium
4	5	Andhra Pradesh 2021-2020	100	Medium
5	6	Uttar Pradesh 2021-2020	100	Medium
6	7	Madhya Pradesh 2021-2020	100	Medium
7	8	Rajasthan 2021-2020	100	Medium
8	9	Bihar 2021-2020	100	Medium
9	10	Odisha 2021-2020	100	Medium

In [18]:
```
df["Cured"] = df["Cured"].apply(map_function)
df.head(10)
```

Out[18]:

	Sno	Date	Time	Cured
0	1	Haryana 2021-2020	100	Medium
1	2	Chandigarh 2021-2020	100	Medium
2	3	Jharkhand 2021-2020	100	Medium
3	4	Karnataka 2021-2020	100	Medium
4	5	Andhra Pradesh 2021-2020	100	Medium
5	6	Uttar Pradesh 2021-2020	100	Medium
6	7	Madhya Pradesh 2021-2020	100	Medium
7	8	Rajasthan 2021-2020	100	Medium
8	9	Bihar 2021-2020	100	Medium
9	10	Odisha 2021-2020	100	Medium

In [19]:
```
# Convert Date to Year, Month and Day
df["Year"] = df.Date.dtype.str[1:2]
df["Month"] = df.Date.dtype.str[3:4]
df["Day"] = df.Date.dtype.str[5:6]
df.head(10)
```

Out[19]:

	Sno	Date	Time	Cured	Year	Month	Day
0	1	Haryana 2021-2020	100	Medium	M	2	2020
1	2	Chandigarh 2021-2020	100	Medium	M	2	2020
2	3	Jharkhand 2021-2020	100	Medium	M	2	2020
3	4	Karnataka 2021-2020	100	Medium	M	2	2020
4	5	Andhra Pradesh 2021-2020	100	Medium	M	2	2020
5	6	Uttar Pradesh 2021-2020	100	Medium	M	2	2020
6	7	Madhya Pradesh 2021-2020	100	Medium	M	2	2020
7	8	Rajasthan 2021-2020	100	Medium	M	2	2020
8	9	Bihar 2021-2020	100	Medium	M	2	2020
9	10	Odisha 2021-2020	100	Medium	M	2	2020

In [20]:
```
# Convert Date to Year, Month and Day
df["Year"] = df.Date.dtype.str[1:2]
df["Month"] = df.Date.dtype.str[3:4]
df["Day"] = df.Date.dtype.str[5:6]
df.head(10)
```

Out[20]:

	Sno	Date	Time	Cured	Year	Month	Day
0	1	Haryana 2021-2020	100	Medium	M	2	2020
1	2	Chandigarh 2021-2020	100	Medium	M	2	2020
2	3	Jharkhand 2021-2020	100	Medium	M	2	2020
3	4	Karnataka 2021-2020	100	Medium	M	2	2020
4	5	Andhra Pradesh 2021-2020	100	Medium	M	2	2020
5	6	Uttar Pradesh 2021-2020	100	Medium	M	2	2020
6	7	Madhya Pradesh 2021-2020	100	Medium	M	2	2020
7	8	Rajasthan 2021-2020	100	Medium	M	2	2020
8	9	Bihar 2021-2020	100	Medium	M	2	2020
9	10	Odisha 2021-2020	100	Medium	M	2	2020

In [21]:
```
df.info()
```

Out[21]:
```
Sno     int64
Date    object
Time    object
Cured   object
Year    object
Month   object
Day     object
dtype: object
<class'pandas.core.frame.DataFrame'>
RangeIndex: 1085 entries, 0 to 1084
Data columns (total 7 columns):
 #   Sno     Non-Null Count  Dtype
---  ------  --------------  -----
 0   Sno     1085 non-null   int64
 1   Date    1085 non-null   object
 2   Time    1085 non-null   object
 3   Cured   1085 non-null   object
 4   Year    1085 non-null   object
 5   Month   1085 non-null   object
 6   Day     1085 non-null   object
dtypes: int64(1), object(6)
memory usage: 59.4 KB
```

Split Input-Data, Output-Data, Train-Test Data

In [22]:
```
# Split data into "Train" and "Validation" (70:30)
split_df1 = df[df

Stratified(Sno,"Cured", 0.30)

0   1
1   2
2   3
3   4
4   5
5   6
6   7
7   8
8   9
9   10
10  11
11  12
12  13
13  14
14  15
Name: Sno, Length: 1085, dtype: int32

'Cured'

Sno  Month  Year  Cured  Test
0    M      2020  1      100
1    M      2020  1      100
2    M      2020  1      100
3    M      2020  1      100
4    M      2020  1      100
5    M      2020  1      100
6    M      2020  1      100
7    M      2020  1      100
```

```
   macro avg       0.10      0.20      0.14        69
weighted avg       0.27      0.52      0.36        69
```

```
C:\Users\Surat\Anaconda3\envs\AI\lib\site-packages\sklearn\metrics\_classification.py:1245: UndefinedMetricWarning:

Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter t
o control this behavior.

C:\Users\Surat\Anaconda3\envs\AI\lib\site-packages\sklearn\metrics\_classification.py:1245: UndefinedMetricWarning:

Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter t
o control this behavior.

C:\Users\Surat\Anaconda3\envs\AI\lib\site-packages\sklearn\metrics\_classification.py:1245: UndefinedMetricWarning:

Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter t
o control this behavior.
```

MLPCLASSFIER MODEL

In [36]:
```
from sklearn.neural_network import MLPClassifier

model4 = MLPClassifier(hidden_layer_sizes = (300, 300), max_iter = 200)

model4.fit(xtrain, ytrain)

ypred4 = model4.predict(xtest)
```

In [37]:
```
from sklearn.metrics import confusion_matrix

print(confusion_matrix(ytest,ypred4))
```

```
[[ 0  1  0  2  0]
 [ 0 20  0  0  0]
 [ 0  0  9  0  0]
 [ 0  0  0  1  0]
 [ 0  0  0  0 36]]
```

In [38]:
```
no_of_misclassified_sample = (ytest != ypred4).sum()
print('Misclassified samples: {}'.format(no_of_misclassified_sample))

mis_cla_rate = (no_of_misclassified_sample / len(ytest)) * 100
print('Misclassified Rate WITH PRECISION: {:.2f} %'.format(mis_cla_rate))
```

```
Misclassified samples: 3
Misclassified Rate WITH PRECISION: 4.35 %
```

In [39]:
```
from sklearn.metrics import classification_report

print('CLASSIFICATION REPORT','\n',classification_report(ytest, ypred4))
```

```
CLASSIFICATION REPORT
               precision    recall  f1-score   support

           0       0.00      0.00      0.00         3
           2       0.95      1.00      0.98        20
           3       1.00      1.00      1.00         9
           4       0.33      1.00      0.50         1
           5       1.00      1.00      1.00        36

    accuracy                           0.96        69
   macro avg       0.66      0.80      0.70        69
weighted avg       0.93      0.96      0.94        69
```

```
C:\Users\Surat\Anaconda3\envs\AI\lib\site-packages\sklearn\metrics\_classification.py:1245: UndefinedMetricWarning:

Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter
to control this behavior.

C:\Users\Surat\Anaconda3\envs\AI\lib\site-packages\sklearn\metrics\_classification.py:1245: UndefinedMetricWarning:
```

In [40]:
```
filename = open('Covidzone.pkl', 'wb')
pickle.dump(model4,filename)
filename.close()
```