TASK-6: COVID19 ANALYSIS USING MACHINE LEARNING

In [1]:
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
colors=['#0C68C7','#3A6794','#00FAF3','#FA643C','#C71D12']
```

In [2]:
```
df=pd.read_csv("covidd19.csv")
```

In [3]:
```
# Overviewing the data before modifications
df.head()
```

Out[3]:

	Sno	Date	Time	State/UnionTerritory	ConfirmedIndianNational	ConfirmedForeignNational	Cured
0	1	30-01-2020	6:00 PM	Kerala	1	0	0
1	2	31-01-2020	6:00 PM	Kerala	1	0	0
2	3	01-02-2020	6:00 PM	Kerala	2	0	0
3	4	02-02-2020	6:00 PM	Kerala	3	0	0
4	5	03-02-2020	6:00 PM	Kerala	3	0	0

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

Out[4]:

	Sno	Date	Time	State/UnionTerritory	ConfirmedIndianNational	ConfirmedForeignNational
15549	15550	01-06-2021	8:00 AM	Telangana	-	-
15550	15551	01-06-2021	8:00 AM	Tripura	-	-
15551	15552	01-06-2021	8:00 AM	Uttarakhand	-	-
15552	15553	01-06-2021	8:00 AM	Uttar Pradesh	-	-
15553	15554	01-06-2021	8:00 AM	West Bengal	-	-

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

Out[5]:
```
139986
```

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

Out[6]:
```
<bound method NDFrame.describe of          Sno        Date     Time State/Union
Territory  \
0          1  30-01-2020  6:00 PM               Kerala
1          2  31-01-2020  6:00 PM               Kerala
2          3  01-02-2020  6:00 PM               Kerala
3          4  02-02-2020  6:00 PM               Kerala
4          5  03-02-2020  6:00 PM               Kerala
...      ...         ...      ...                  ...
15549  15550  01-06-2021  8:00 AM            Telangana
15550  15551  01-06-2021  8:00 AM              Tripura
15551  15552  01-06-2021  8:00 AM          Uttarakhand
15552  15553  01-06-2021  8:00 AM        Uttar Pradesh
15553  15554  01-06-2021  8:00 AM          West Bengal

      ConfirmedIndianNational ConfirmedForeignNational    Cured  Deaths  \
0                           1                        0        0       0
1                           1                        0        0       0
2                           2                        0        0       0
3                           3                        0        0       0
4                           3                        0        0       0
...                       ...                      ...      ...     ...
15549                       -                        -   540986    3281
15550                       -                        -    44908     519
15551                       -                        -   294671    6452
15552                       -                        -  1633947   20497
15553                       -                        -  1273788   15541

       Confirmed
0              1
1              1
2              2
3              3
4              3
...          ...
15549     578351
15550      51974
15551     329494
15552    1691488
15553    1376377

[15554 rows x 9 columns]>
```

In [7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15554 entries, 0 to 15553
Data columns (total 9 columns):
 #   Column                   Non-Null Count  Dtype 
---  ------                   --------------  ----- 
 0   Sno                      15554 non-null  int64 
 1   Date                     15554 non-null  object
 2   Time                     15554 non-null  object
 3   State/UnionTerritory     15554 non-null  object
 4   ConfirmedIndianNational  15554 non-null  object
 5   ConfirmedForeignNational 15554 non-null  object
 6   Cured                    15554 non-null  int64 
 7   Deaths                   15554 non-null  int64 
 8   Confirmed                15554 non-null  int64 
dtypes: int64(4), object(5)
memory usage: 1.1+ MB
```

In [8]:
```
# Checking for any null values
df.isnull().sum()
```

Out[8]:
```
Sno                         0
Date                        0
Time                        0
State/UnionTerritory        0
ConfirmedIndianNational     0
ConfirmedForeignNational    0
Cured                       0
Deaths                      0
Confirmed                   0
dtype: int64
```

In [9]:
```
# Dropping the columns which are not going to be used

df.drop(["Sno","Time","ConfirmedIndianNational","ConfirmedForeignNational"],inpla
```

In [10]:
```
#finding the active cases

df['Active_cases']=df['Confirmed']-(df['Cured']+df['Deaths'])
df.head()
```

Out[10]:

	Date	State/UnionTerritory	Cured	Deaths	Confirmed	Active_cases
0	30-01-2020	Kerala	0	0	1	1
1	31-01-2020	Kerala	0	0	1	1
2	01-02-2020	Kerala	0	0	2	2
3	02-02-2020	Kerala	0	0	3	3
4	03-02-2020	Kerala	0	0	3	3

In [11]:

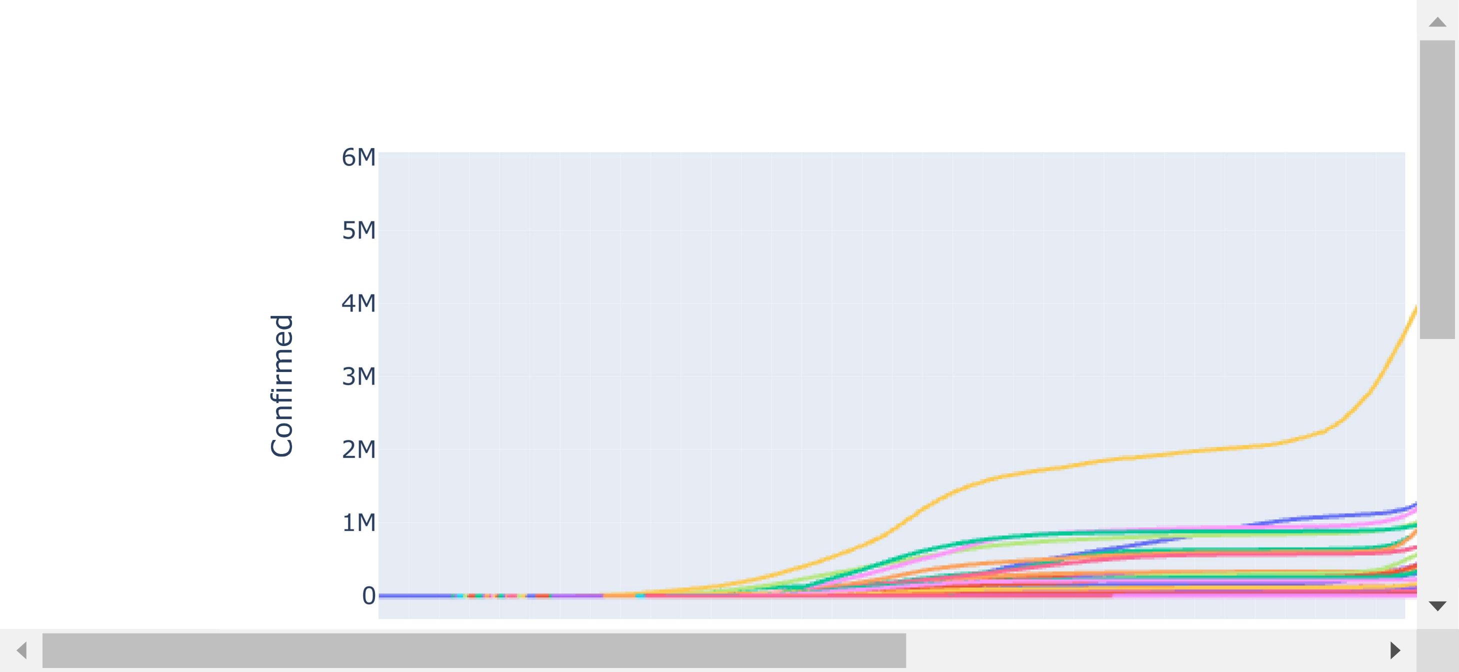
```
df['Confirmed'] = pd.to_numeric(df['Confirmed'], errors='coerce')
df['Confirmed']=df['Confirmed'].fillna(0)
df['Confirmed']=df['Confirmed'].astype('int')

df['Deaths'] = pd.to_numeric(df['Deaths'], errors='coerce')
df['Deaths']=df['Deaths'].fillna(0)
df['Deaths']=df['Deaths'].astype('int')

df['Cured'] = pd.to_numeric(df['Cured'], errors='coerce')
df['Cured']=df['Cured'].fillna(0)
df['Cured']=df['Cured'].astype('int')

df['Deaths'] = pd.to_numeric(df['Deaths'], errors='coerce')
df['Deaths']=df['Deaths'].fillna(0)
df['Deaths']=df['Deaths'].astype('int')

df['Cured'] = pd.to_numeric(df['Cured'], errors='coerce')
df['Cured']=df['Cured'].fillna(0)
df['Cured']=df['Cured'].astype('int')
```

DATA VISUALIZATION

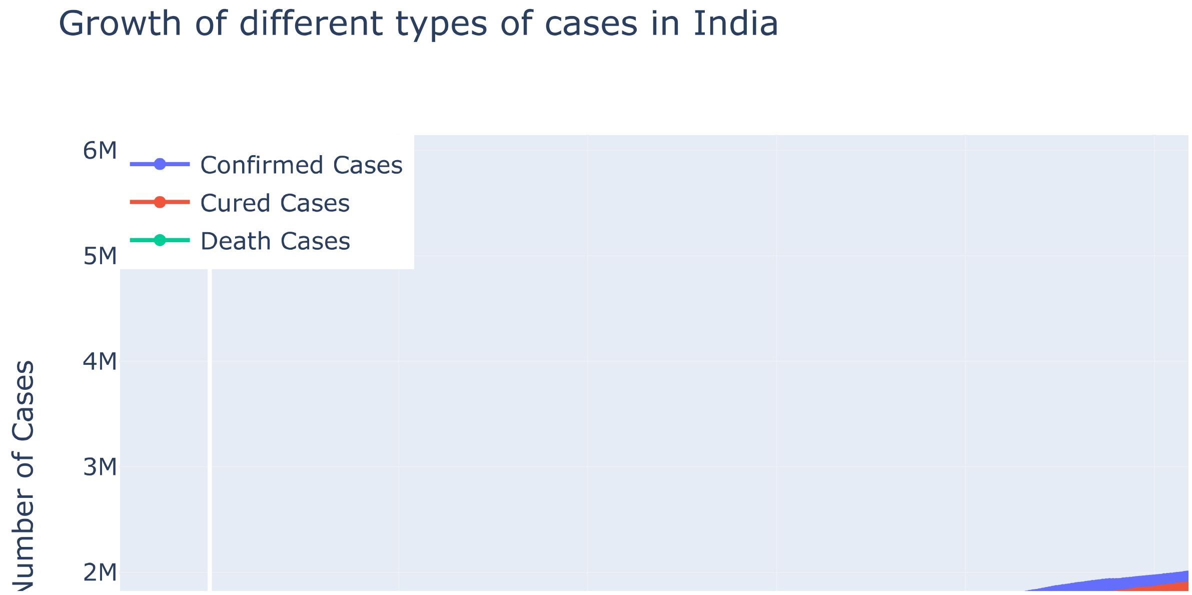
In [12]:

```
# Data Visualization Liraries
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import plotly.express as px
import plotly.offline as pyo
import plotly.graph_objs as go
from plotly.subplots import make_subplots
```

```
In [13]: plt.figure(figsize = (18,10))
         figure = px.line(df, x='Date', y='Confirmed', color='State/UnionTerritory')
         figure.update_xaxes(rangeslider_visible=True)
         pyo.iplot(figure)
```

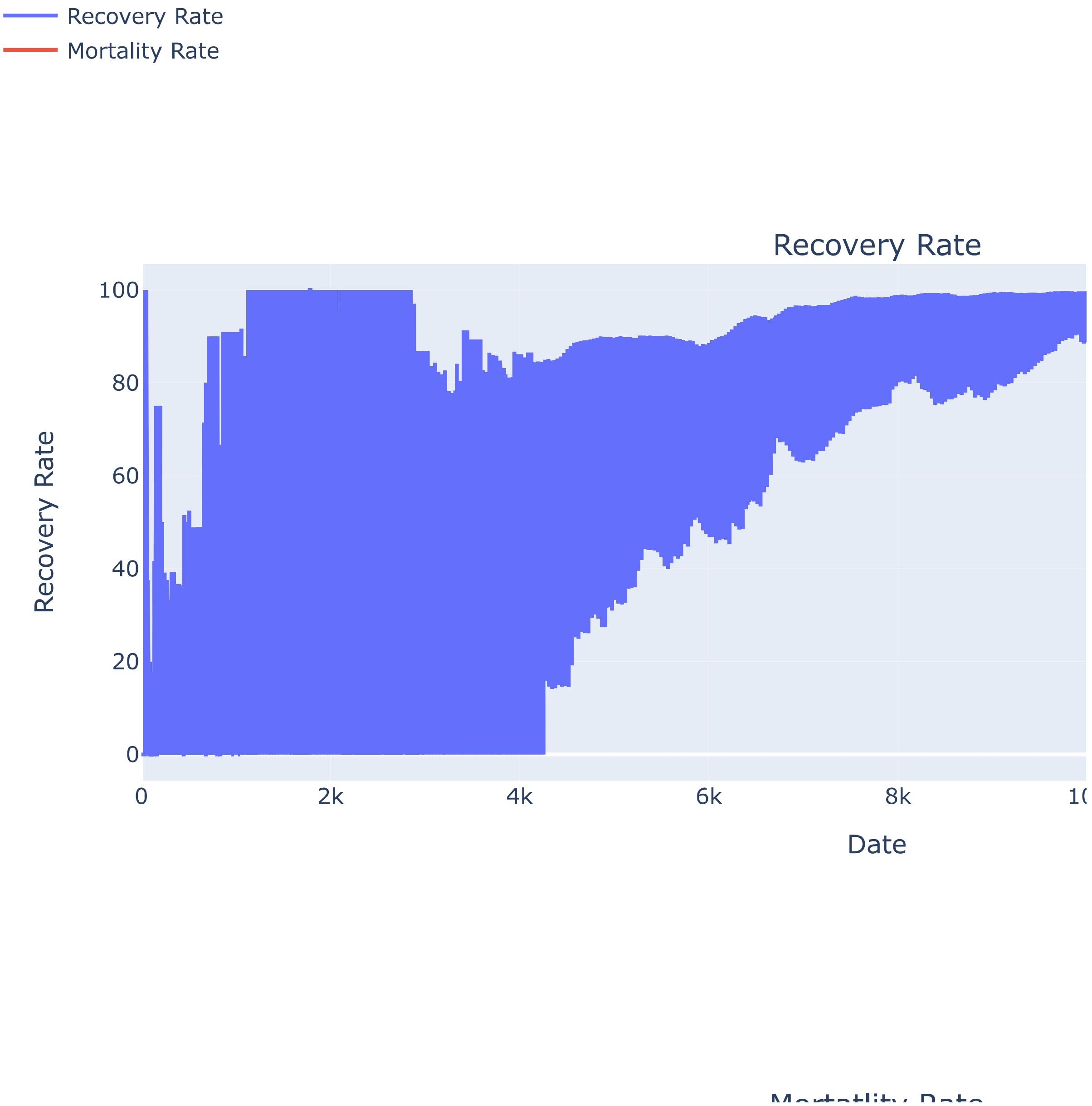


In [14]:
```
fig=go.Figure()
fig.add_trace(go.Scatter(x=df.index, y=df["Confirmed"],
                    mode='lines+markers',
                    name='Confirmed Cases'))
fig.add_trace(go.Scatter(x=df.index, y=df["Cured"],
                    mode='lines+markers',
                    name='Cured Cases'))
fig.add_trace(go.Scatter(x=df.index, y=df["Deaths"],
                    mode='lines+markers',
                    name='Death Cases'))
fig.update_layout(title="Growth of different types of cases in India",
                 xaxis_title="Date",yaxis_title="Number of Cases",legend=dict(x=
fig.show()
```



In [15]:

```
fig = make_subplots(rows=2, cols=1,
                   subplot_titles=("Recovery Rate", "Mortatlity Rate"))
fig.add_trace(
    go.Scatter(x=df.index, y=(df["Cured"]/df["Confirmed"])*100,
              name="Recovery Rate"),
    row=1, col=1
)
fig.add_trace(
    go.Scatter(x=df.index, y=(df["Deaths"]/df["Confirmed"])*100,
              name="Mortality Rate"),
    row=2, col=1
)
fig.update_layout(height=1000,legend=dict(x=-0.1,y=1.2,traceorder="normal"))
fig.update_xaxes(title_text="Date", row=1, col=1)
fig.update_yaxes(title_text="Recovery Rate", row=1, col=1)
fig.update_xaxes(title_text="Date", row=1, col=2)
fig.update_yaxes(title_text="Mortality Rate", row=1, col=2)
fig.show()
```



MACHINE LEARNING MODELS

1. SARIMAX

In [16]:

```
# test size
test_size = 30
train_size= len(df) - test_size
train = df[['Confirmed']].iloc[:train_size]
test = df[['Confirmed']].iloc[train_size:]
# train and test

print(train.shape)
print(test.shape)
```

```
(15524, 1)
(30, 1)
```

In [17]:

```
# exons variables
exons=df[['Cured','Deaths']]

full_data = df['Confirmed']
```

In [*]:
```
import statsmodels.api as sm
SARIMAX__model = sm.tsa.statespace.SARIMAX(full_data.values,exons=exons,
                                            order=(1,0,1),
                                            seasonal_order=(1,0,1,7),
                                         enforce_stationarity=False,
                                         enforce_invertibility=False,)
SARIMAX__model = SARIMAX__model.fit(maxiter=1000)

days=60
prediction = SARIMAX__model.get_forecast(steps=days)
pred_date = prediction.summary_frame(alpha=0.05).set_index(pd.date_range(start='2
```

In [*]:
```
fig = go.Figure()

fig.add_trace(go.Scatter(
    name="Actual",
     x=df['Date'], y=df["Confirmed"]))

fig.add_trace(go.Scatter(
    name="prediction",mode="lines",
     x=pred_date.index, y=pred_date['mean']))

fig.add_trace(go.Scatter(
    name="lowerbound",mode="lines",
        line=dict(width=0),fillcolor='rgba(68, 68, 68, 0.3)',
        fill='tonexty',showlegend=False,
     x=pred_date.index, y=pred_date['mean_ci_lower']))

fig.add_trace(go.Scatter(name="upperbound",mode="lines",
        line=dict(width=0),fillcolor='rgba(68, 68, 68, 0.3)',fill='tonexty',showl
                         x=pred_date.index, y=pred_date['mean_ci_upper']))
```

2. SCIPY ¶

In [*]:
```
median_states=df[df['State/UnionTerritory'].isin(['Kerala','Tamil Nadu','Delhi','
median_states.groupby(by=['State/UnionTerritory']).median().style.bar(['Confirmed
```

In [*]:
```
from scipy.stats import norm
with plt.xkcd():
    fig=plt.figure(figsize=(15,8))
    ax=sns.kdeplot(data=median_states[median_states['State/UnionTerritory'].isin(
    ax.axvline(median_states[median_states['State/UnionTerritory']=='Kerala']['De
    ax.axvline(median_states[median_states['State/UnionTerritory']=='Tamil Nadu']
    ax.axvline(median_states[median_states['State/UnionTerritory']=='Maharashtra'
```

In [ ]:

In [ ]:

In [ ]:

In [ ]: