# CMPE 257 - Machine Learning

## Project Proposal

COVID-19 Forecaster


### Team 8

Nhat Trinh [011227645]
Suhas Byrapuneni [016118596]
Venkata Sai Sri Batchu [016118557]
Rutik Sanjay Sangle [016007589]


GitHub Repository Link:
https://github.com/rutiksangle3436/CMPE257-Machine_Learning

Google Colab Link:
https://colab.research.google.com/drive/13T6_5sz46ejlapYEGo5OnbOQgm5WmqTL?usp=sharing

# Introduction

## Data

Our group will build the project using the COVID-19 dataset provided by WHO. Specifically, the dataset is named “Daily cases and deaths by date reported to WHO”. The dataset is provided as a CSV file. The dataset includes almost every country in the world that has reported COVID deaths since the start of the 2020 calendar year. Each row in the dataset has the date, country code, country name, assigned WHO region, new deaths, new cases, cumulative deaths, and cumulative cases.

## Problem

The problem our group will be trying to solve is finding where and when the next COVID outbreak will happen by looking at historic daily data from the beginning of the pandemic and learning from it. A COVID outbreak can be characterized as an abnormal change of upwards slope in the daily COVID cases/deaths graph. Even though the cause of COVID-19 transmission can be multi-faceted, such as transmission through touch, proximity, city planning, region, etc. Our group believes it can be largely tied to seasonal and temperature changes that cause the uptick in COVID cases and/or deaths.

## Potential Methods

Since our dataset is labeled, Our team will potentially try to use a supervised learning method; namely, the regression method can understand the relationship between dependent and independent variables. Specifically, our group will try to use an autoregression model utilizing Poisson distribution, called Poisson Autoregression (PAR).

# Preprocessing & Initial Findings

## Preprocessing

The initial data analysis was carried out by checking for the types of data values and checking if there are any missing values. The type of data values is suitable for our solution but there were a few missing values in the column ‘Country_code’. After some more investigation, we found out that the country code for Namibia was missing. By using the fillna() method of pandas, the missing values were replaced.

## Initial Findings

For initial findings, we made some plots using Plotly.express library. The plots describe some basic information like the Top 10 countries with the highest number of cases. Along with that, we used the choropleth plot for plotting the world map which shows the number of total cases in every country. Another plot we made is of the top 5 countries by the number of cases and showing the growth in cases on a time frame. This plot gives us the idea that there were 2 times when there was a sudden increase in cases. First in Jan 2021 and second around Jan 2022. This gives us an idea that the month of January is seeing a major rise in cases. The last plot we made is of the top cases in a single day. This plot gave us the idea that the United States of America has seen the most cases in a single, which is around 5.5 million cases.

## Importing required Libraries

In [1]:
```
!pip install pandasql
!pip install iso3166

import pandas as pd
import numpy as np
from sklearn import datasets
import matplotlib.pyplot as plt
```

```
Requirement already satisfied: pandasql in c:\users\rutik sangle\anaconda3\lib\site-packages (0.7.3)
Requirement already satisfied: sqlalchemy in c:\users\rutik sangle\anaconda3\lib\site-packages (from pandasql) (1.4.39)
Requirement already satisfied: numpy in c:\users\rutik sangle\anaconda3\lib\site-packages (from pandasql) (1.21.5)
Requirement already satisfied: pandas in c:\users\rutik sangle\anaconda3\lib\site-packages (from pandasql) (1.4.4)
Requirement already satisfied: pytz>=2020.1 in c:\users\rutik sangle\anaconda3\lib\site-packages (from pandas->pandasql) (2022.
1)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\users\rutik sangle\anaconda3\lib\site-packages (from pandas->pandas
ql) (2.8.2)
Requirement already satisfied: greenlet!=0.4.17 in c:\users\rutik sangle\anaconda3\lib\site-packages (from sqlalchemy->pandasq
l) (1.1.1)
Requirement already satisfied: six>=1.5 in c:\users\rutik sangle\anaconda3\lib\site-packages (from python-dateutil>=2.8.1->pand
as->pandasql) (1.16.0)
Requirement already satisfied: iso3166 in c:\users\rutik sangle\anaconda3\lib\site-packages (2.1.1)
```

## Importing the dataset using csv file

In [2]:
```
covid = pd.read_csv("WHO-COVID-19-global-data.csv")
covid.head()
```

Out[2]:

| | Date_reported | Country_code | Country | WHO_region | New_cases | Cumulative_cases | New_deaths | Cumulative_deaths |
|---|---|---|---|---|---|---|---|---|
| 0 | 2020-01-03 | AF | Afghanistan | EMRO | 0 | 0 | 0 | 0 |
| 1 | 2020-01-04 | AF | Afghanistan | EMRO | 0 | 0 | 0 | 0 |
| 2 | 2020-01-05 | AF | Afghanistan | EMRO | 0 | 0 | 0 | 0 |
| 3 | 2020-01-06 | AF | Afghanistan | EMRO | 0 | 0 | 0 | 0 |
| 4 | 2020-01-07 | AF | Afghanistan | EMRO | 0 | 0 | 0 | 0 |

### Checking for missing values

In [3]:
```
missing_props = covid.isna().sum()
missing_props
```

Out[3]:
```
Date_reported          0
Country_code        1030
Country                0
WHO_region             0
New_cases              0
Cumulative_cases       0
New_deaths             0
Cumulative_deaths      0
dtype: int64
```

In [4]:
```
print(covid[covid['Country_code'].isnull()])
```

```
       Date_reported Country_code  Country WHO_region  New_cases  \
147290    2020-01-03          NaN  Namibia       AFRO          0
147291    2020-01-04          NaN  Namibia       AFRO          0
147292    2020-01-05          NaN  Namibia       AFRO          0
147293    2020-01-06          NaN  Namibia       AFRO          0
147294    2020-01-07          NaN  Namibia       AFRO          0
...              ...          ...      ...        ...        ...
148315    2022-10-24          NaN  Namibia       AFRO          0
148316    2022-10-25          NaN  Namibia       AFRO          0
148317    2022-10-26          NaN  Namibia       AFRO          0
148318    2022-10-27          NaN  Namibia       AFRO          0
148319    2022-10-28          NaN  Namibia       AFRO          0

        Cumulative_cases  New_deaths  Cumulative_deaths
147290                 0           0                  0
147291                 0           0                  0
147292                 0           0                  0
147293                 0           0                  0
147294                 0           0                  0
...                  ...         ...                ...
148315            169891           0               4080
148316            169891           0               4080
148317            169891           0               4080
148318            169891           0               4080
148319            169891           0               4080

[1030 rows x 8 columns]
```

**Using SQL queries with pandas to confirm the missing values**

In [5]:
```
from pandasql import sqldf
pysqldf = lambda q: sqldf(q, globals())

q = """SELECT count(*)
       FROM covid
       where Country = "Namibia"
       ;"""

missing = pysqldf(q)
missing
```

Out[5]:

| | count(*) |
|---|---|
| 0 | 1030 |

**Using fillna() to fill the missing values with the appropriate country code**

In [6]:
```
covid['Country_code'] = covid['Country_code'].fillna('NA')
```

In [7]:
```
missing_props = covid.isna().sum()
missing_props
```

Out[7]:
```
Date_reported        0
Country_code         0
Country              0
WHO_region           0
New_cases            0
Cumulative_cases     0
New_deaths           0
Cumulative_deaths    0
dtype: int64
```

## Top 10 Countries by highest number of cases

In [8]:
```
TopCases = covid.groupby(['Country'])['Cumulative_cases'].max().reset_index()
WorldMap = TopCases # Making a copy for later use
```

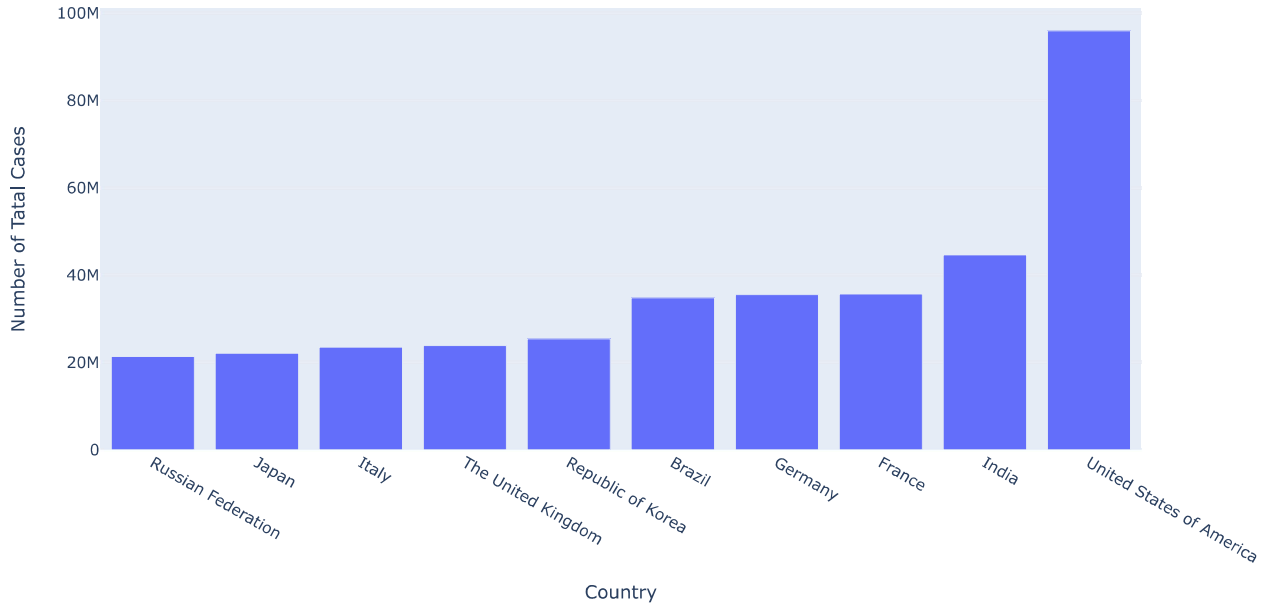
In [9]:
```
TopCases.head(20)
```

Out[9]:

| | Country | Cumulative_cases |
|---|---|---|
| 0 | Afghanistan | 202537 |
| 1 | Albania | 331723 |
| 2 | Algeria | 270817 |
| 3 | American Samoa | 8257 |
| 4 | Andorra | 46535 |
| 5 | Angola | 103131 |
| 6 | Anguilla | 3866 |
| 7 | Antigua and Barbuda | 9106 |
| 8 | Argentina | 9717546 |
| 9 | Armenia | 445100 |
| 10 | Aruba | 43334 |
| 11 | Australia | 10332884 |
| 12 | Austria | 5421306 |
| 13 | Azerbaijan | 823100 |
| 14 | Bahamas | 37369 |
| 15 | Bahrain | 688646 |
| 16 | Bangladesh | 2034968 |
| 17 | Barbados | 103014 |
| 18 | Belarus | 994037 |
| 19 | Belgium | 4612239 |

In [10]:
```
TopCases = TopCases.sort_values('Cumulative_cases')
TopCases = TopCases[-10:]
```

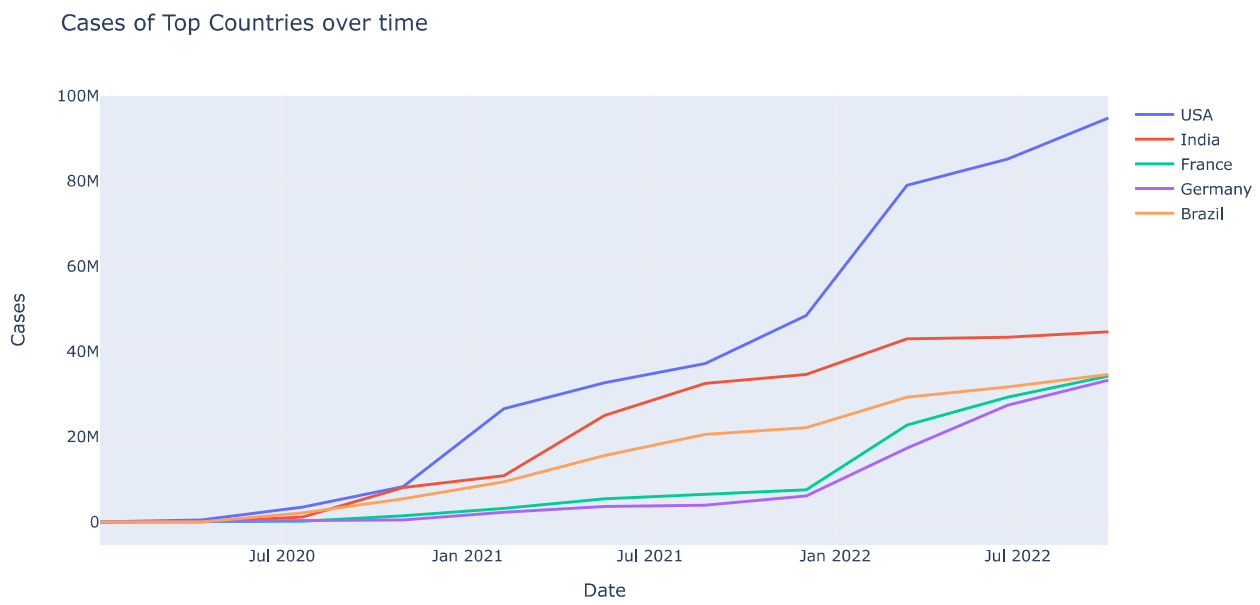
In [11]:
```
import plotly.express as px
fig = px.bar(TopCases, x='Country', y='Cumulative_cases', labels={'Country':'Country','Cumulative_cases':'Number of Tatal Cases'
fig.show()
```




## Comparison of Top 5 Countries(by cases) over the time period

In [12]:
```
USA = covid.loc[covid['Country'] == "United States of America"]
USA = USA.iloc[::100]
USA = USA[["Date_reported", "Cumulative_cases"]]

IND = covid.loc[covid['Country'] == "India"]
IND = IND.iloc[::100]
IND = IND[["Date_reported", "Cumulative_cases"]]

FRA = covid.loc[covid['Country'] == "France"]
FRA = FRA.iloc[::100]
FRA = FRA[["Date_reported", "Cumulative_cases"]]

GER = covid.loc[covid['Country'] == "Germany"]
GER = GER.iloc[::100]
GER = GER[["Date_reported", "Cumulative_cases"]]

BRA = covid.loc[covid['Country'] == "Brazil"]
BRA = BRA.iloc[::100]
BRA = BRA[["Date_reported", "Cumulative_cases"]]
```

In [13]:
```
import plotly.graph_objects as go

fig = go.Figure()
fig.add_trace(go.Scatter(x=USA["Date_reported"], y=USA["Cumulative_cases"], name="USA", mode="lines"))
fig.add_trace(go.Scatter(x=IND["Date_reported"], y=IND["Cumulative_cases"], name="India", mode="lines"))
fig.add_trace(go.Scatter(x=FRA["Date_reported"], y=FRA["Cumulative_cases"], name="France", mode="lines"))
fig.add_trace(go.Scatter(x=GER["Date_reported"], y=GER["Cumulative_cases"], name="Germany", mode="lines"))
fig.add_trace(go.Scatter(x=BRA["Date_reported"], y=BRA["Cumulative_cases"], name="Brazil", mode="lines"))
fig.update_layout(
    title="Cases of Top Countries over time", xaxis_title="Date", yaxis_title="Cases"
)
fig.show()
```



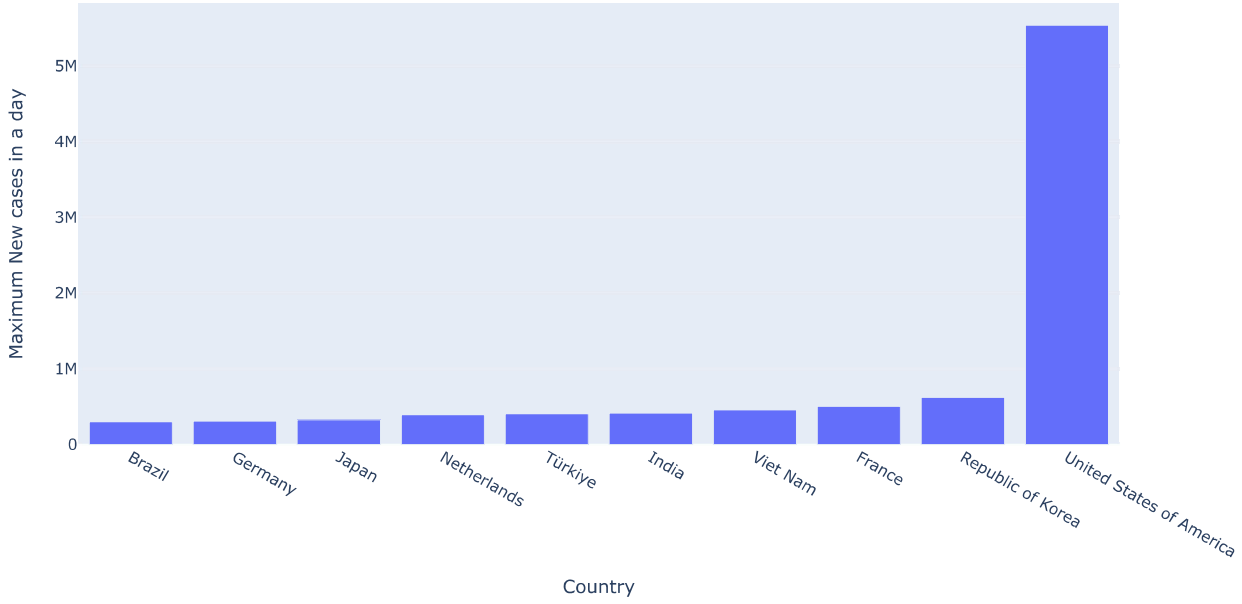

## Maximum New Cases in a day

In [14]:
```
from pandasql import sqldf
pysqldf = lambda q: sqldf(q, globals())

q = """SELECT Date_reported, Country, max(New_cases) as New_cases
       FROM covid
       group by Country
       ;"""

TopDaily = pysqldf(q)
TopDaily
TopDaily = TopDaily.sort_values('New_cases')
TopDaily = TopDaily[-10:]
TopDaily
```

Out[14]:

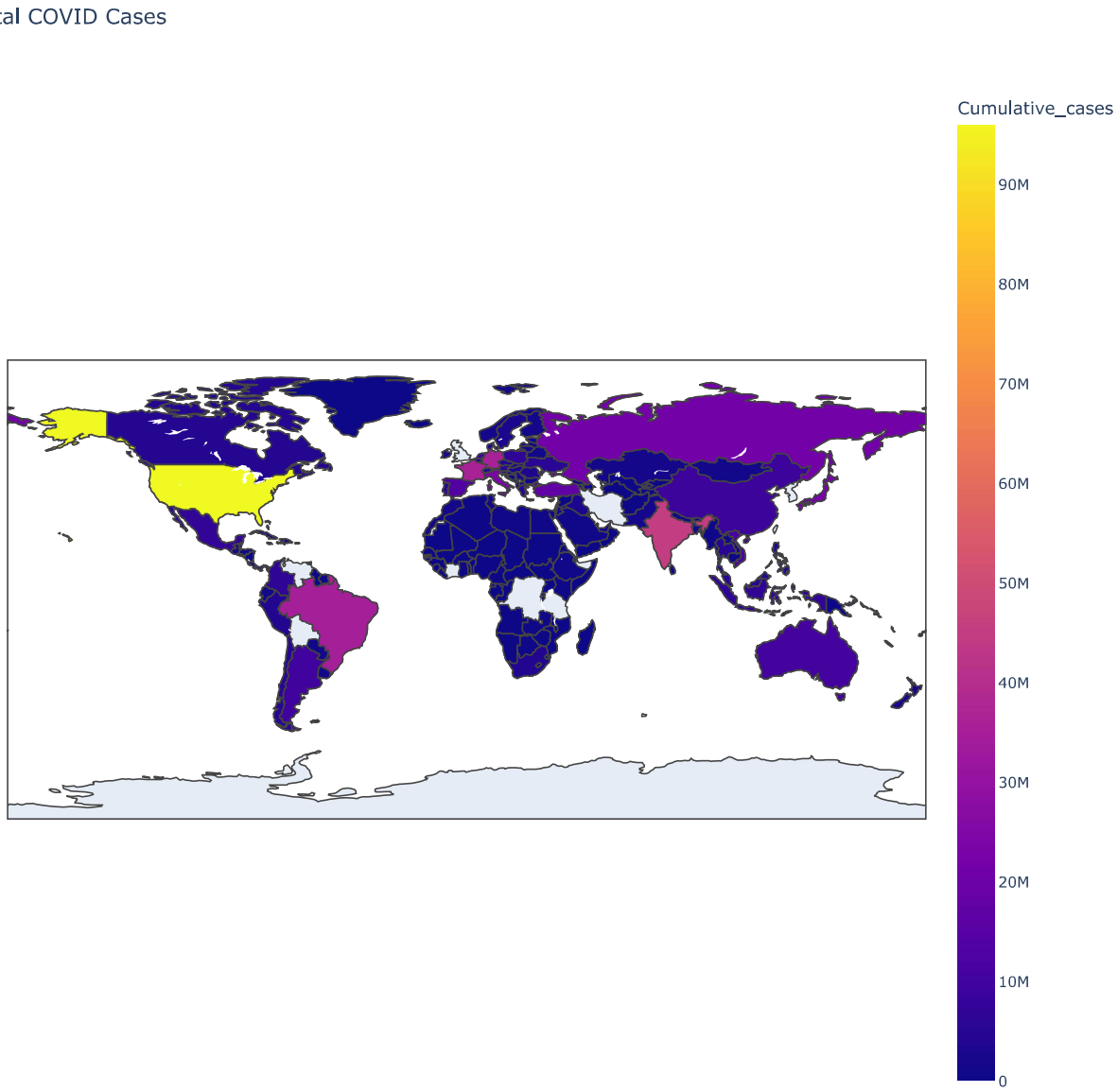
| | Date_reported | Country | New_cases |
|---|---|---|---|
| **28** | 2022-02-05 | Brazil | 298408 |
| **78** | 2022-03-24 | Germany | 307922 |
| **105** | 2022-08-29 | Japan | 326090 |
| **146** | 2022-02-09 | Netherlands | 391563 |
| **220** | 2022-08-01 | Türkiye | 406322 |
| **96** | 2021-05-07 | India | 414188 |
| **231** | 2022-03-13 | Viet Nam | 454212 |
| **72** | 2022-01-26 | France | 500563 |
| **170** | 2022-03-17 | Republic of Korea | 621328 |
| **226** | 2022-01-21 | United States of America | 5535129 |

In [15]:
```
fig = px.bar(TopDaily, x='Country', y='New_cases', hover_data=['Country', 'Date_reported', 'New_cases'], labels={'Country':'Coun
fig.show()
```




## World map showing number of total cases in every country

In [16]:
```
from iso3166 import countries

def findCountryOfficialName(country_name):
    try:
        return countries.get(country_name)[2]
    except:
        return ("Unknown")

country_names =[]
for country in WorldMap['Country']:
    country_names.append(findCountryOfficialName(country))

WorldMap['Country'] = country_names


plot1 = px.choropleth(WorldMap, locations='Country',
                    color="Cumulative_cases",
                    title="Total COVID Cases",
                    height=1000, width=1000,
                    color_continuous_scale=px.colors.sequential.Plasma)
plot1.show()
```




In [ ]: