VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

TEAM MEMBERS

ABHISHEK KUMAR-18BIT0348(abhishek.kumar2018e@vitstudent.ac.in)
DEVANSHU RANJAN-18BIT0353(devanshu.ranjan2018@vitstudent.ac.in)
SUDHANSHU BHATIA-18BIT0391(sudhanshu.bhatia2018@vitstudent.ac.in)

FACULTY
Dr. DURAI RAJ VINCENT P.M
Associate Professor Grade 2
School of Information Technology and Engineering
Vellore Institute of Technology
Vellore, India

TITLE:-

Prediction of Covid-19 Status in future (For Next Days) Using Machine Learning.

Abstract :

The spread of COVID-19 in the whole world has put the humanity at risk. The resources of some of the largest economies are stressed out due to the large infectivity and transmissibility of this disease. Due to the growing magnitude of number of cases and its subsequent stress on the administration and health professionals, some prediction methods would be required to predict the number of cases in future. In this paper, we have used data-driven estimation methods like long short-term memory (LSTM) and curve fitting for prediction of the number of COVID-19 cases in India 30 days ahead and effect of preventive measures like social isolation and lockdown on the spread of COVID-19. The prediction of various parameters (number of positive cases, number of recovered cases, etc.) obtained by the proposed method is accurate within a certain range and will be a beneficial tool for administrators and health officials.

As India is suffering from huge Pandemic now a days and we don't know how and when we all get free from this .Huge number of cases are coming out every day and it is increasing day by day and we even don't know whether the cases will increase more in future or declination will take place .So this huge no of increasing COVID-19 cases somehow affecting our mental health.

So we need a strong model that predicts what will be the condition of our country in future and how will be the condition compared from other countries.

Description of the Project :

Most of the prior research and media coverage focused on the number of infections in the entire country. However, given the size and diversity of India, it is important to look at the spread of the disease in each state separately, wherein the situations are quite different. In this paper, we aim to analyze data on the number of infected people in each Indian state (restricted to only those states with enough data for prediction) and predict the number of infections for that state in the next 30 days. We hope that such statewise predictions would help the state governments better channelize their limited health care resources.

Literature Survey:

Author	Title	Source	Findings
Romney B Duffey, Enrico Dio	Analysing Recovery From Pandemics by Learning Theory: The Case of CoVid-19	https:// ieeexplore.ieee.org/ document/9113279	Here is the method for predicting the recovery time from infectious diseases outbreaks such as the recent CoVid-19 virus.
Minii Jain, Rohit Kumar,Pranav Kataria	Modelling Logistic Growth Model for COVID-19 Pandemic in India	https:// ieeexplore.ieee.org/ document/9138049	this early analysis of growth dynamics for infectious diseases, like COVID-19, is needed to dissect the crucial driving factors that result in rapid disease transmission, it refines the measures taken to control the pandemic and improve disease forecast.
Ingid Burbey, Thomas L. Martin	Predicting future locations using prediction-by-partial-match	https://dl.acm.org/ doi/ 10.1145/1410012.1 410014	Here they implemented the Prediction-by-Partial-Match data compression algorithm as a predictor of future locations. Positioning were done using IEEE 802.11 wireless access logs.

Burb L Brien	Explore scientific, technical, and medical research on ScienceDirect	https:// www.sciencedirect.com	We gone through analyzing the existing data of India epidemic situation, the corresponding model is established, and then the simulation is carried out.
Himanshu Kumar,Bijaya ketan panigrahi	Prediction and analysis of COVID-19 positive cases using deep learning models: A descriptive case study of India.	https:// www.sciencedirect.com/science/article/pii/S096007792030415X	we have proposed deep learning models for predicting the number of COVID-19 positive cases in Indian states.
Jhon F Kenn	COVID-19 India Datasets by DataMeet	http:// projects.datameet.org/covid19/	Just Gone through all dataset about how much and what parameters we need to select in order to make our Projects.
Amit Bhatti, Anurag Jagetia	Prediction of COVID-19 Outbreak in India adopting Bhilwara Model of Containment	https:// ieeexplore.ieee.org/document/9138060	This paper has described the Bhilwara model and compare the model with India COVID-19 outbreak lockdown along with a prediction for a reduction in the number of upcoming

			cases with its implementation.
Piyush Raj, Siddharth Singh, Raman kumar, Rishu Chaujar	Prediction and forecast for COVID-19 Outbreak in India based on Enhanced Epidemiological Models.	https:// ieeexplore.ieee.org/ document/9183126	This paper has analyzed the COVID-19 outbreak in India. In this work, the difference between number of actual reported confirmed cases and approximate number of actual cases, due to insufficient number of tests being conducted,

Munish Kumar, Surbhi gupta, Krishan Kumar, Monica Sachdeva	SPREADING OF COVID-19 IN INDIA, ITALY, JAPAN, SPAIN, UK, US: A Prediction Using ARIMA and LSTM Model.	https://dl.acm.org/ doi/ 10.1145/3411760	In this article, the authors have presented two data-driven estimation techniques, namely, Auto- Regressive Integrated Moving Average (ARIMA) and Long Short-Term Memory (LSTM) for prediction of the cumulative number of COVID-19 cases and the cumulative number of deaths due to COVID-19.

Rajan Gupta, Gaurav Pandey, Poonam Chaudhary	Machine Learning Models for Government to Predict COVID-19 Outbreak.	https://dl.acm.org/doi/10.1145/3411761	In this paper, outbreak of this disease has been analysed and trained for Indian region till 10th May, 2020, and testing has been done for the number of cases for the next three weeks. Machine learning models such as SEIR model and Regression model have been used for predictions based on the data collected from the official portal of the Government of India
Bowen Bang, Yanjing Sun, Trung Q, Duong	Risk-Aware Identification of Highly Suspected COVID-19 Cases in Social IoT: A Joint Graph Theory and Reinforcement Learning Approach	https://ieeexplore.ieee.org/document/9121230	The outbreak of the coronavirus disease 2019 (COVID-19) has rapidly become a pandemic, which calls for prompt action in identifying suspected cases at an early stage through risk prediction.
R sujath, Jyothir Moy Chatterjee	A machine learning forecasting model for COVID-19 pandemic in India.	https://link.springer.com/article/10.1007/s00477-020-01827-s	Information and communication technology help in the decision- making process based on the past data with the data analytics and

			data mining perspective.

Outcome of the Project :

Here a comparison of Random Forest and Multiple Linear Regression model is performed where the score of the model R2 tends to be 0.99 and 1.0 which indicates a strong prediction model to forecast the next coming days active cases. Using the Multiple Linear Regression model as on September month, the forecast value of 52,290 active cases are predicted towards the next month of 15th October in India and 9,358 active cases in Odisha if situation continues like this way.
At the end of the our Project,we are trying to compare the best accurrate result predicted by these two models.

Complete Code and Complete Analysis :

Part 1: Analysing the present condition in India

1.1 Reading the Datasets

```
#Learn how to read a .xls file by creating a dataframe using pandas
# Reading the datasets
df= pd.read_excel('/content/Covid cases in India.xlsx')
df_india = df.copy()
df
```

```
[ ] df= pd.read_excel('/Covid in India.xlsx')
    df_india = df.copy()
    df
```

	S. No.	Name of State / UT	Total Confirmed cases (Indian National)	Total Confirmed cases (Foreign National)	Cured	Death
0	1	Andhra Pradesh	693484	115	629211	5828
1	2	Bihar	182906	25	169625	904
2	3	Chhattisgarh	113602	13	81718	957
3	4	Delhi	279715	221	247446	5361
4	5	Gujarat	137394	171	117331	3453
5	6	Haryana	128599	27	112877	1382
6	7	Himachal Pradesh	14976	4	11370	181
7	8	Karnataka	601767	205	485268	8864
8	9	Kerala	196107	1121	128220	743
9	10	Madhya Pradesh	128047	148	104734	2316
10	11	Maharashtra	1384446	2714	1088322	36662
11	12	Manipur	10983	2	8460	67
12	13	Mizoram	2017	2	1597	0
13	14	Odisha	219119	121	185700	895
14	15	Puducherry	27544	56	22074	521
15	16	Punjab	113886	271	93666	3406
16	17	Rajasthan	135292	96	113225	1486

```
# Coordinates of India States and Union Territories
India_coord = pd.read_excel('/content/Indian Coordinates.xlsx')

#Day by day data of India, Korea, Italy and Wuhan
dbd_India = pd.read_excel('/content/per_day_cases.xlsx',parse_dates=True,
sheet_name='India')
dbd_Italy = pd.read_excel('/content/per_day_cases.xlsx',parse_dates=True,
sheet_name="Italy")
dbd_Korea = pd.read_excel('/content/per_day_cases.xlsx',parse_dates=True,
sheet_name="Korea")
dbd_Wuhan = pd.read_excel('/content/per_day_cases.xlsx',parse_dates=True,
sheet_name="Wuhan")
```

[ ]

6	7	Himachal Pradesh	14976	4	11370	181
7	8	Karnataka	601767	205	485268	8864
8	9	Kerala	196107	1121	128220	743
9	10	Madhya Pradesh	128047	148	104734	2316
10	11	Maharashtra	1384446	2714	1088322	36662
11	12	Manipur	10983	2	8460	67
12	13	Mizoram	2017	2	1597	0
13	14	Odisha	219119	121	185700	895
14	15	Puducherry	27544	56	22074	521
15	16	Punjab	113886	271	93666	3406
16	17	Rajasthan	135292	96	113225	1486
17	18	Tamil Nadu	597602	128	541819	9520
18	19	Telengana	193600	78	163407	1135
19	20	Chandigarh	11938	49	9813	162
20	21	Jammu and Kashmir	75070	35	56872	1181
21	22	Ladakh	4269	4	3147	58
22	23	Uttar Pradesh	399082	181	342415	5784
23	24	Uttarakhand	49000	42	39035	611
24	25	West Bengal	257049	78	225759	4958

1.2 Analysing COVID19 Cases in India

```
#Learn how to play around with the dataframe and create a new attrib
ute of 'Total Case'
#Total case is the total number of confirmed cases (Indian National
+ Foreign National)

df.drop(['S. No.'],axis=1,inplace=True)
df['Total cases'] = df['Total Confirmed cases (Indian National)'] +
df['Total Confirmed cases ( Foreign National )']
total_cases = df['Total cases'].sum()
print('Total number of confirmed COVID 2019 cases across India till
date (22nd March, 2020):', total_cases)

#Learn how to highlight your dataframe
df.style.background_gradient(cmap='Greens')
```

```
[ ] df.style.background_gradient(cmap='Greens')
```

	S. No.	Name of State / UT	Total Confirmed cases (Indian National)	Total Confirmed cases (Foreign National)	Cured	Death
0	1	Andhra Pradesh	693484	115	629211	5828
1	2	Bihar	182906	25	169625	904
2	3	Chhattisgarh	113602	13	81718	957
3	4	Delhi	279715	221	247446	5361
4	5	Gujarat	137394	171	117331	3453
5	6	Haryana	128599	27	112877	1382
6	7	Himachal Pradesh	14976	4	11370	181
7	8	Karnataka	601767	205	485268	8864
8	9	Kerala	196107	1121	128220	743
9	10	Madhya Pradesh	128047	148	104734	2316
10	11	Maharashtra	1384446	2714	1088322	36662
11	12	Manipur	10983	2	8460	67
12	13	Mizoram	2017	2	1597	0
13	14	Odisha	219119	121	185700	895
14	15	Puducherry	27544	56	22074	521
15	16	Punjab	113886	271	93666	3406
16	17	Rajasthan	135292	96	113225	1486
17	18	Tamil Nadu	597602	128	541819	9520
18	19	Telengana	193600	78	163407	1135
19	20	Chandigarh	11938	49	9813	162
20	21	Jammu and Kashmir	75070	35	56872	1181
21	22	Ladakh	4269	4	3147	58
22	23	Uttar Pradesh	399082	181	342415	5784
23	24	Uttarakhand	49000	42	39035	611
24	25	West Bengal	257049	78	225759	4958

1.3 Number of Active COVID-19 cases in affected State/Union Territories

```
#Total Active  is the Total cases - (Number of death + Cured)
df['Total Active'] = df['Total cases'] - (df['Death'] + df['Cured'])
total_active = df['Total Active'].sum()
print('Total number of active COVID 2019 cases across India:', total
_active)
Tot_Cases = df.groupby('Name of State / UT')['Total Active
'].sum().sort_values(ascending=False).to_frame()
Tot_Cases.style.background_gradient(cmap='Reds')
```

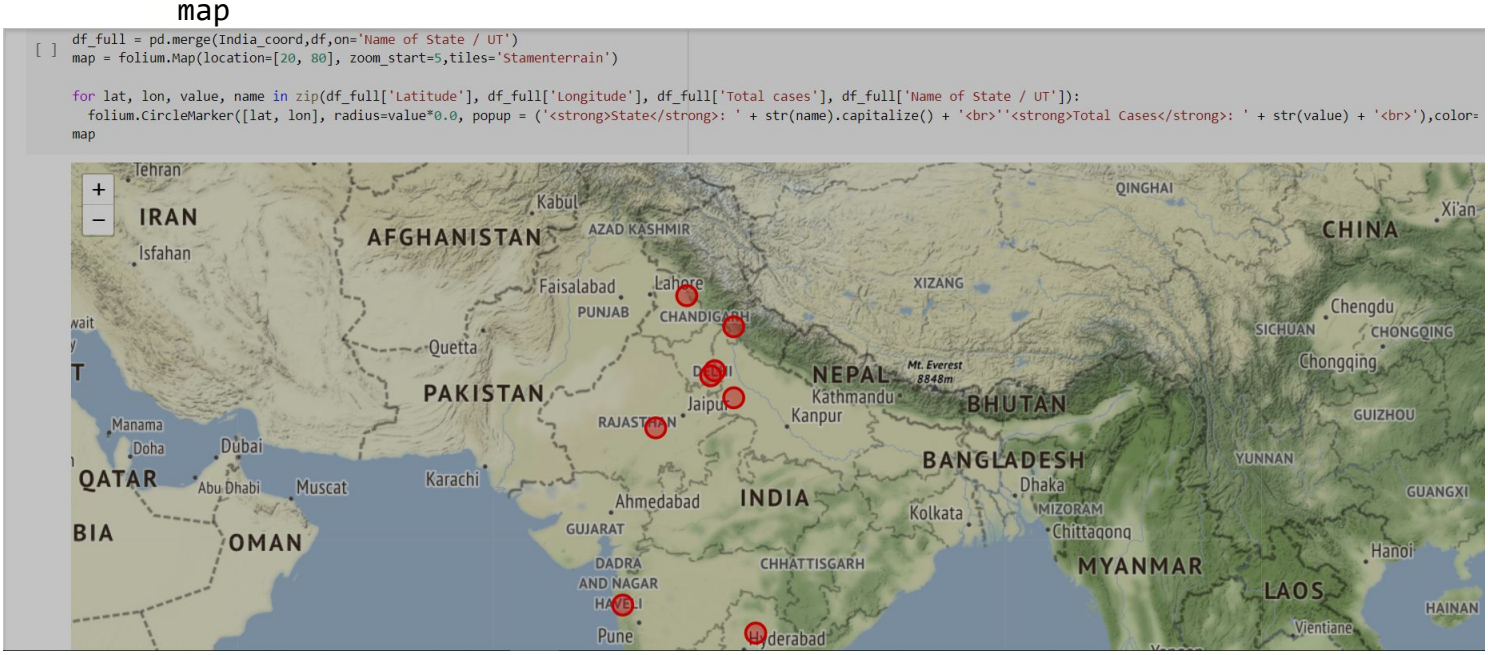
```
[ ] df.drop(['S. No.'],axis=1,inplace=True)
    df['Total cases'] = df['Total Confirmed cases (Indian National)'] + df['Total Confirmed cases ( Foreign National )']
    total_cases = df['Total cases'].sum()
    print('Total number of confirmed COVID 2019 cases across India till date (1st September, 2020):', total_cases)

    Total number of confirmed COVID 2019 cases across India till date (1st September, 2020): 5963401

[ ] df['Total Active'] = df['Total cases'] - (df['Death'] + df['Cured'])
    total_active = df['Total Active'].sum()
    print('Total number of active COVID 2019 cases across India:', total_active)
    Tot_Cases = df.groupby('Name of State / UT')['Total Active'].sum().sort_values(ascending=False).to_frame()
    Tot_Cases.style.background_gradient(cmap='Reds')

    Total number of active COVID 2019 cases across India: 883855
```

Name of State / UT	Total Active
Maharashtra	262176
Karnataka	107840
Kerala	68265
Andhra Pradesh	58560
Uttar Pradesh	51064
Tamil Nadu	46391
Odisha	32645
Chhattisgarh	30940
Telengana	29136
Delhi	27129
West Bengal	26410
Madhya Pradesh	21145
Rajasthan	20677
Punjab	17085

Name of State / UT	Total Active
Maharashtra	262176
Karnataka	107840
Kerala	68265
Andhra Pradesh	58560
Uttar Pradesh	51064
Tamil Nadu	46391
Odisha	32645
Chhattisgarh	30940
Telengana	29136
Delhi	27129
West Bengal	26410
Madhya Pradesh	21145
Rajasthan	20677
Punjab	17085
Jammu and Kashmir	17052
Gujarat	16781
Haryana	14367
Bihar	12402
Uttarakhand	9396
Puducherry	5005
Himachal Pradesh	3429
Manipur	2458
Chandigarh	2012
Ladakh	1068
Mizoram	422

1.4 Visualising the spread geographically

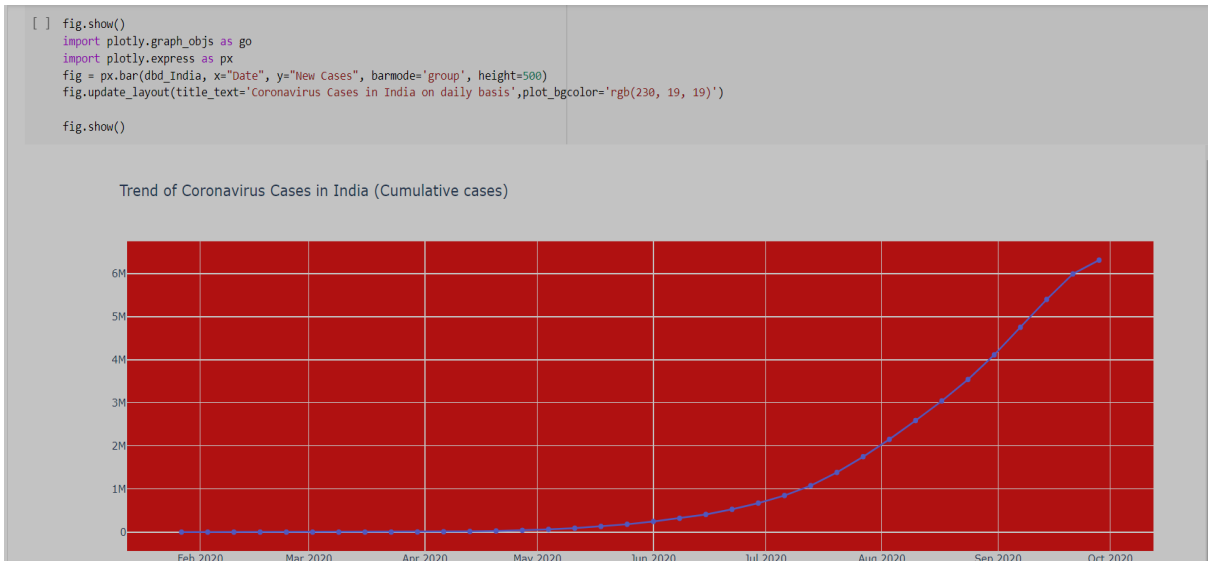
```
# Learn how to use folium to create a zoomable map
df_full = pd.merge(India_coord,df,on='Name of State / UT')
map = folium.Map(location=[20, 70], zoom_start=4,tiles='Stamenterrai
n')

for lat, lon, value, name in zip(df_full['Latitude'], df_full['Longi
tude'], df_full['Total cases'], df_full['Name of State / UT']):
  folium.CircleMarker([lat, lon], radius=value*0.8, popup = ('<stron
  g>State</strong>: ' + str(name).capitalize() + '<br>''<strong>Tota
  l Cases</strong>: ' + str(value) + '<br>'),color='red',fill_color=
  'red',fill_opacity=0.3 ).add_to(map)
  map
```

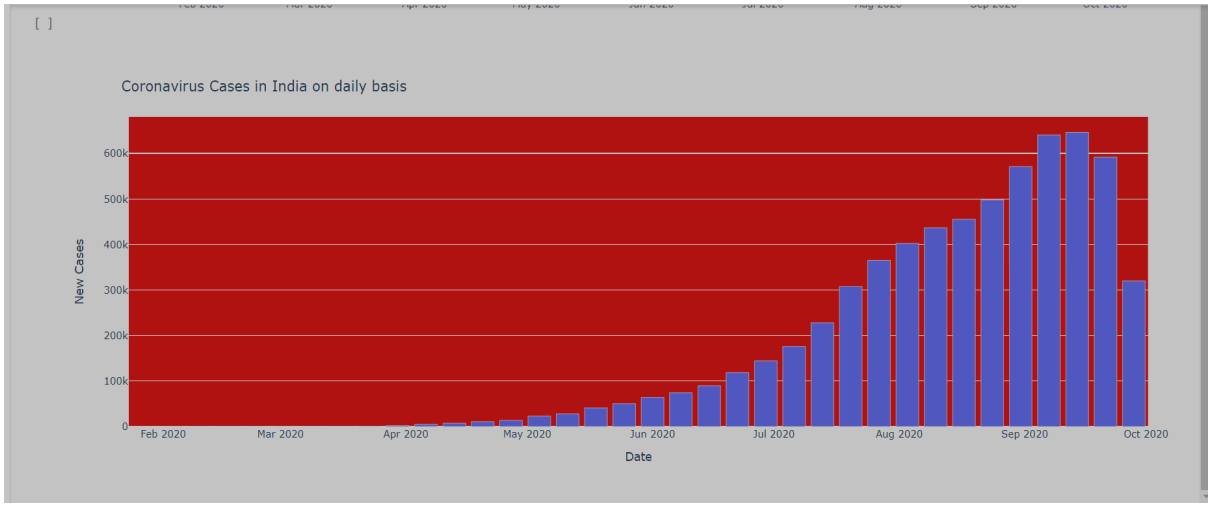


1.5 How the Coronavirus cases are rising?

```
#This cell's code is required when you are working with plotly on colab
import plotly
plotly.io.renderers.default = 'colab'

#Learn how to create interactive graphs using plotly
# import plotly.graph_objects as go
# Rise of COVID-19 cases in India
fig = go.Figure()
fig.add_trace(go.Scatter(x=dbd_India['Date'], y = dbd_India['Total Cases']
, mode='lines+markers',name='Total Cases'))
fig.update_layout(title_text='Trend of Coronavirus Cases in India (Cumulat
ive cases)',plot_bgcolor='rgb(230, 230, 230)')
fig.show()
```

```
# New COVID-19 cases reported daily in India

import plotly.express as px
fig = px.bar(dbd_India, x="Date", y="New Cases", barmode='group', height=4
00)
fig.update_layout(title_text='Coronavirus Cases in India on daily basis',p
lot_bgcolor='rgb(230, 230, 230)')

fig.show()
```



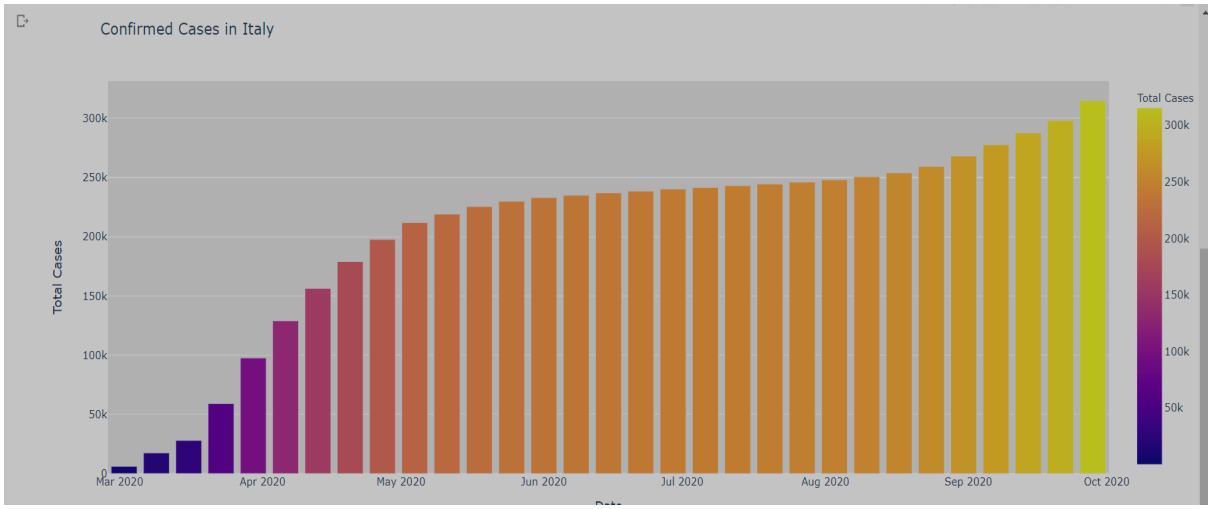


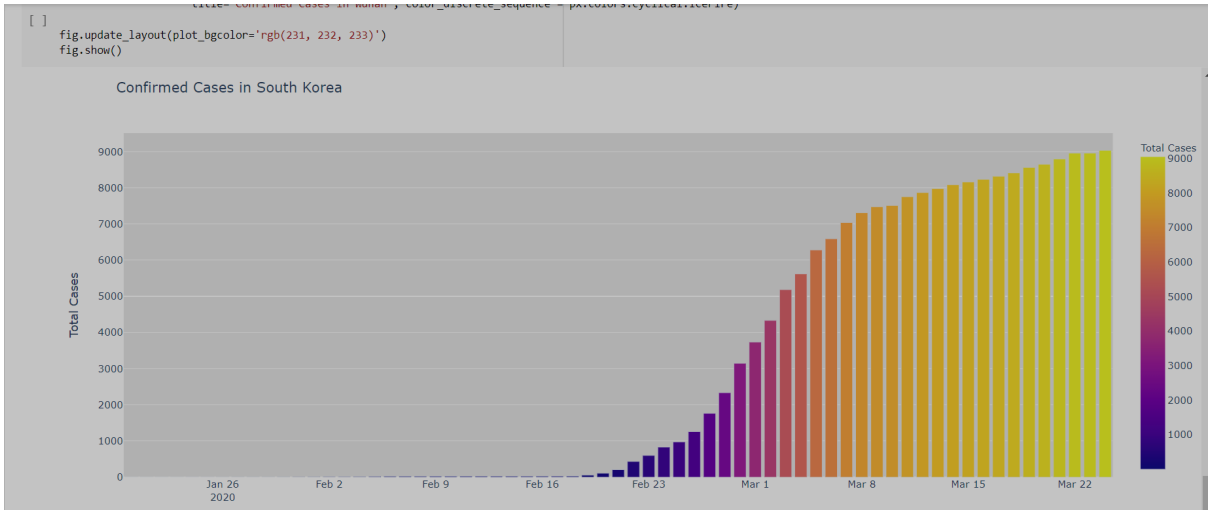
Part 2: Is the trend similar to Italy/ S.Korea/ Wuhan?

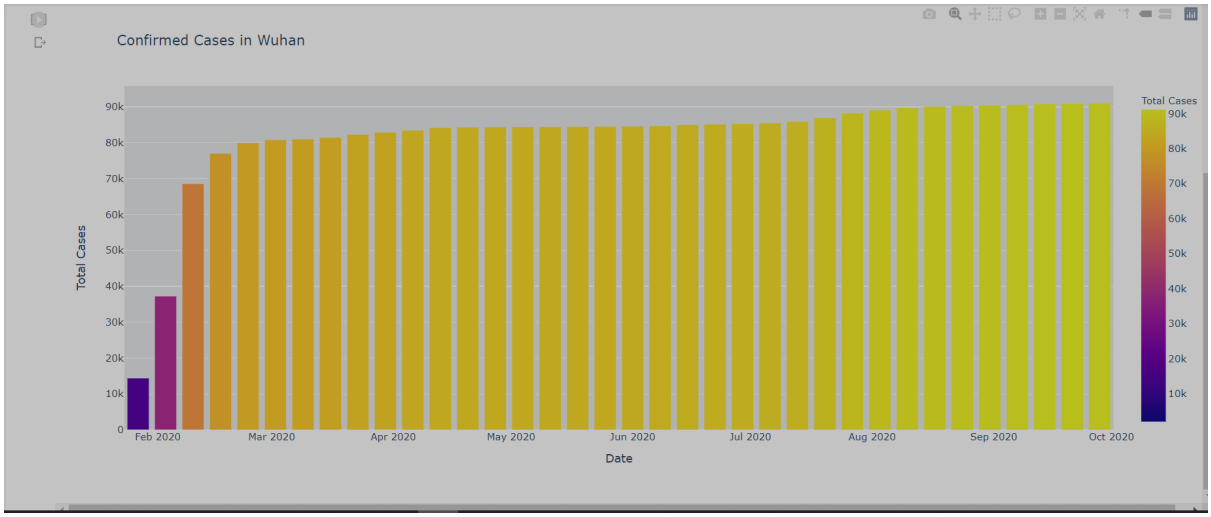
2.1 Cumulative cases in India, Italy, S.Korea, and Wuhan

```
# import plotly.express as px
fig = px.bar(dbd_India, x="Date", y="Total Cases", color='Total Cases', or
```

```
ientation='v', height=600,
       title='Confirmed Cases in India', color_discrete_sequence = px.color
s.cyclical.IceFire)

'''Colour Scale for plotly
https://plot.ly/python/builtin-colorscales/
'''

fig.update_layout(plot_bgcolor='rgb(230, 230, 230)')
fig.show()

fig = px.bar(dbd_Italy, x="Date", y="Total Cases", color='Total Cases', or
ientation='v', height=600,
       title='Confirmed Cases in Italy', color_discrete_sequence = px.color
s.cyclical.IceFire)

fig.update_layout(plot_bgcolor='rgb(230, 230, 230)')
fig.show()

fig = px.bar(dbd_Korea, x="Date", y="Total Cases", color='Total Cases', or
ientation='v', height=600,
       title='Confirmed Cases in South Korea', color_discrete_sequence = px
.colors.cyclical.IceFire)

fig.update_layout(plot_bgcolor='rgb(230, 230, 230)')
fig.show()
fig = px.bar(dbd_Wuhan, x="Date", y="Total Cases", color='Total Cases', or
ientation='v', height=600,
       title='Confirmed Cases in Wuhan', color_discrete_sequence = px.color
s.cyclical.IceFire)

fig.update_layout(plot_bgcolor='rgb(230, 230, 230)')
fig.show()
```


Confirmed Cases in Italy
Total Cases
300k
250k
200k
150k
100k
50k
0
Mar 2020
Apr 2020
May 2020
Jun 2020
Jul 2020
Aug 2020
Sep 2020
Oct 2020


[ ]
fig.update_layout(plot_bgcolor='rgb(231, 232, 233)')
fig.show()
Confirmed Cases in South Korea
Total Cases
9000
8000
7000
6000
5000
4000
3000
2000
1000
0
Jan 26
2020
Feb 2
Feb 9
Feb 16
Feb 23
Mar 1
Mar 8
Mar 15
Mar 22



Part 3: Exploring World wide data

```
df = pd.read_csv('/content/
covid_19_clean_complete.csv',parse_dates=['Date'])
df.rename(columns={'ObservationDate':'Date', 'Country/Region':'Country'},
inplace=True)

df_confirmed = pd.read_csv("/content/
time_series_covid19_confirmed_global.csv")
df_recovered = pd.read_csv("/content/
time_series_covid19_recovered_global.csv")
df_deaths = pd.read_csv("/content/time_series_covid19_deaths_global.csv")

df_confirmed.rename(columns={'Country/Region':'Country'}, inplace=True)
df_recovered.rename(columns={'Country/Region':'Country'}, inplace=True)
df_deaths.rename(columns={'Country/Region':'Country'}, inplace=True)
```

```
[ ] df = pd.read_csv('/covid_19_clean_complete.csv',parse_dates=['Date'])
    df.rename(columns={'ObservationDate':'Date', 'Country/Region':'Country'}, inplace=True)

    df_confirmed = pd.read_csv("/time_series_covid19_confirmed_global.csv")
    df_recovered = pd.read_csv("/time_series_covid19_recovered_global.csv")
    df_deaths = pd.read_csv("/time_series_covid19_deaths_global.csv")

    df_confirmed.rename(columns={'Country/Region':'Country'}, inplace=True)
    df_recovered.rename(columns={'Country/Region':'Country'}, inplace=True)
    df_deaths.rename(columns={'Country/Region':'Country'}, inplace=True)
```

```
[ ] df_deaths.head()
```

	Province/State	Country	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	1/28/20	1/29/20	1/30/20	1/31/20	2/1/20	2/2/20	2/3/20	2/4/20	2/5/20	2/6/
0	NaN	Afghanistan	33.93911	67.709953	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
1	NaN	Albania	41.15330	20.168300	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
2	NaN	Algeria	28.03390	1.659600	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
3	NaN	Andorra	42.50630	1.521800	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
4	NaN	Angola	-11.20270	17.873900	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

5 rows × 257 columns

```
# Check for India's data
df.query('Country=="India"').groupby("Date")[['Confirmed', 'Deaths', 'Reco
vered']].sum().reset_index()
```

```
[ ] df.query('Country=="India"').groupby("Date")[['Confirmed', 'Deaths', 'Recovered']].sum().reset_index()
```

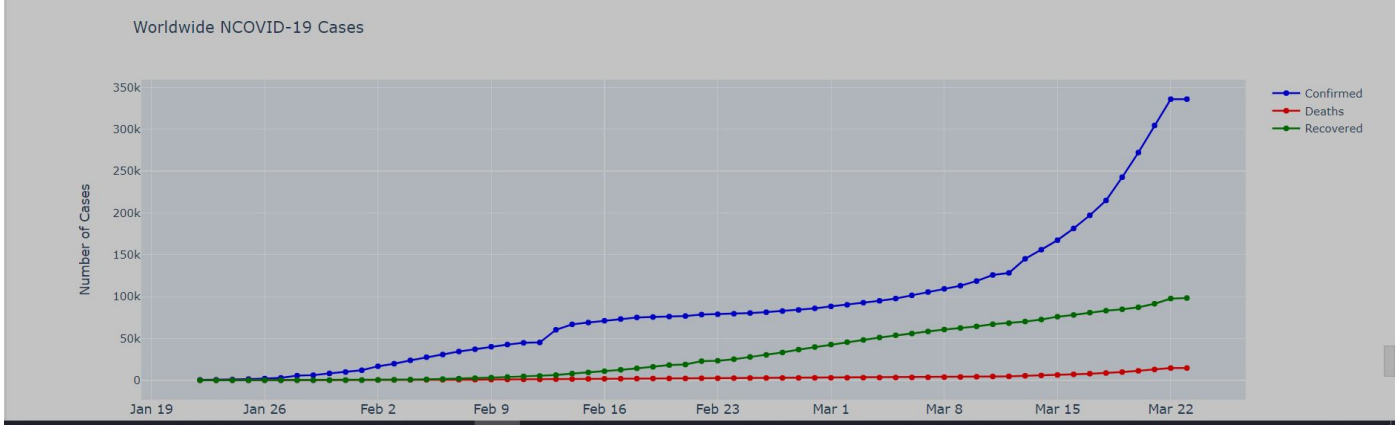
	Date	Confirmed	Deaths	Recovered
0	2020-01-22	0.0	0.0	0.0
1	2020-01-23	0.0	0.0	0.0
2	2020-01-24	0.0	0.0	0.0
3	2020-01-25	0.0	0.0	0.0
4	2020-01-26	0.0	0.0	0.0
...	...	...	...	...
57	2020-03-19	194.0	4.0	15.0
58	2020-03-20	244.0	5.0	20.0
59	2020-03-21	330.0	4.0	23.0
60	2020-03-22	396.0	7.0	27.0
61	2020-03-23	396.0	7.0	27.0

62 rows × 4 columns

3.1 Visualizing: Worldwide NCOVID-19 cases

```
confirmed = df.groupby('Date').sum()['Confirmed'].reset_index()
deaths = df.groupby('Date').sum()['Deaths'].reset_index()
recovered = df.groupby('Date').sum()['Recovered'].reset_index()
fig = go.Figure()

#Plotting datewise confirmed cases
fig.add_trace(go.Scatter(x=confirmed['Date'], y=confirmed['Confirmed'], mo
de='lines+markers', name='Confirmed',line=dict(color='blue', width=2)))
fig.add_trace(go.Scatter(x=deaths['Date'], y=deaths['Deaths'], mode='lines
+markers', name='Deaths', line=dict(color='Red', width=2)))
fig.add_trace(go.Scatter(x=recovered['Date'], y=recovered['Recovered'], mo
de='lines+markers', name='Recovered', line=dict(color='Green', width=2)))
fig.update_layout(title='Worldwide NCOVID-19 Cases', xaxis_tickfont_size
=14,yaxis=dict(title='Number of Cases'))

fig.show()
```

```
[ ]
fig = go.Figure()

fig.add_trace(go.Scatter(x=confirmed['Date'], y=confirmed['Confirmed'], mode='lines+markers', name='Confirmed',line=dict(color='blue', width=2)))
fig.add_trace(go.Scatter(x=deaths['Date'], y=deaths['Deaths'], mode='lines+markers', name='Deaths', line=dict(color='Red', width=2)))
fig.add_trace(go.Scatter(x=recovered['Date'], y=recovered['Recovered'], mode='lines+markers', name='Recovered', line=dict(color='Green', width=2)))
fig.update_layout(title='Worldwide NCOVID-19 Cases', xaxis_tickfont_size=14,yaxis=dict(title='Number of Cases'))
fig.show()
```



Linear regression Algorithm:

```
from sklearn.model_selection import train_test_split

#converting string date to date-time
import datetime as dt
india_case['date'] = pd.to_datetime(india_case['date'])
india_case.head()
```

	iso_code	location	date	total_cases	new_cases	total_deaths	new_deaths	total_cases_
8379	IND	India	2019-12-31	0	0	0	0	
8380	IND	India	2020-01-01	0	0	0	0	
8381	IND	India	2020-01-02	0	0	0	0	
8382	IND	India	2020-01-03	0	0	0	0	
8383	IND	India	2020-01-04	0	0	0	0	

```
india_case.head()
```

	iso_code	location	date	total_cases	new_cases	total_deaths	new_deaths	total_cases_
8379	IND	India	2019-12-31	0	0	0	0	
8380	IND	India	2020-01-01	0	0	0	0	
8381	IND	India	2020-01-02	0	0	0	0	
8382	IND	India	2020-01-03	0	0	0	0	
8383	IND	India	2020-01-04	0	0	0	0	

```
#converting date-time to ordinal
india_case['date']=india_case['date'].map(dt.datetime.toordinal)
india_case.head()
```

	iso_code	location	date	total_cases	new_cases	total_deaths	new_deaths	total_case
8379	IND	India	737424	0	0	0	0	
8380	IND	India	737425	0	0	0	0	
8381	IND	India	737426	0	0	0	0	
8382	IND	India	737427	0	0	0	0	
8383	IND	India	737428	0	0	0	0	

```
#getting dependent variable and inpedent variable
x=india_case['date']
y=india_case['total_cases']


x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)


from sklearn.linear_model import LinearRegression
lr = LinearRegression()


import numpy as np
lr.fit(np.array(x_train).reshape(-1,1),np.array(y_train).reshape(-
1,1))

LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
normalize=False)

india_case.tail()
```

	iso_code	location	date	total_cases	new_cases	total_deaths	new_deaths	total_case
8519	IND	India	737565	106750	5611	3303	140	
8520	IND	India	737566	112359	5609	3435	132	
8521	IND	India	737567	118447	6088	3583	148	
8522	IND	India	737568	125101	6654	3720	137	
8523	IND	India	737569	131868	6767	3867	147	

```
y_pred=lr.predict(np.array(x_test).reshape(-1,1))
```

```
from sklearn.metrics import mean_squared_error
```

```
mean_squared_error(x_test,y_pred)
```

```
OUTPUT: 524620551377.64185
```

```
lr.predict(np.array([[737573]]))
```

```
OUTPUT: array([[49967.36422545]])
```

RANDOM FOREST ALGORITHM:

```
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

Select all rows and column 1 from dataset to x and all rows and column 2 as y

```
x = data.iloc[:, 1:2].values
print(x)
y = data.iloc[:, 2].values
```

```
# Fitting Random Forest Regression to the dataset
# import the regressor
from sklearn.ensemble import RandomForestRegressor

 # create regressor object
regressor = RandomForestRegressor(n_estimators = 100, random_state = 0)

# fit the regressor with x and y data
regressor.fit(x, y)
```

```
Y_pred = regressor.predict(np.array([6.5]).reshape(1, 1))  # test the
output by changing values



# Visualising the Random Forest Regression results

# arange for creating a range of values
# from min value of x to max
# value of x with a difference of 0.01
# between two consecutive values
X_grid = np.arange(min(x), max(x), 0.01)

# reshape for reshaping the data into a len(X_grid)*1 array,
# i.e. to make a column out of the X_grid value
X_grid = X_grid.reshape((len(X_grid), 1))

# Scatter plot for original data
plt.scatter(x, y, color = 'blue')

# plot predicted data
plt.plot(X_grid, regressor.predict(X_grid),
          color = 'green')
plt.title('Random Forest Regression')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()



from sklearn.metrics import mean_squared_error

mean_squared_error(x_test,y_pred)

OUTPUT: 5246.64185

lr.predict(np.array([[737573]]))


OUTPUT: array([[1344252.36422545]])
```

Result:

If the dataset contains features some of which are Categorical Variables and some of the others are continuous variable Decision Tree is better than Linear Regression,since Trees can accurately divide the data based on Categorical Variables.

In Terms of Accuracy random forest model has produced the best accurate result then Linear regression model.

CONCLUSION :

Our environment is under the control of the COVID-19 virus. This article intended to employ the machine learning models for pandemic analysis through a dataset from Johns Hopkins. In conclusion, the method of Polynomial Regression (PR) generated a minimum Root Mean Square Error (RMSE) amount over other methods in projecting the COVID-19 transmission. However, if the spread mimics the prognosticated trend of the PR model, then it would lead to extensive loss of lives as it presents the incredible growth of the transmission globally. As perceived in China, the increased case of COVID-19 can be degraded by lessening the number of sensitive individuals from infected people. This new normal is obtainable by becoming unsocial and supporting the lockdown regulation with control.These models acquired remarkable accuracy in COVID-19 recognition. Bearing in mind these projected active results, the current estimated for COVID-19 containment needs to be reinforced or updated. Our framework could assist and protect healthcare professionals, government officials in making plans appropriate to cope with the influx of future COVID-19 patients.

References:

[1].Barstugan M, Ozkaya U, Ozturk S (2020) Coronavirus (covid-19) classification using ct images by machine learning methods. arXiv preprint arXiv:2003.09424

[2].Billio M, Casarin R, Rossini L (2019) Bayesian nonparametric sparse VAR models. J Econ 212(1):97–115

[3].Cui H, Singh VP (2017) Application of minimum relative entropy theory for streamflow forecasting. Stoch Env Res Risk Assess 31(3):587–608

[4].Elmousalami HH, Hassanien AE (2020) Day level forecasting for Coronavirus Disease (COVID-19) spread: analysis, modeling and recommendations. arXiv preprint arXiv:2003.07778

[5].Ezzat D, Ella HA (2020) GSA-DenseNet121-COVID-19: a hybrid deep learning architecture for the diagnosis of COVID-19 disease based on gravitational search optimization algorithm. arXiv preprint arXiv:2004.05084

[6].Gautheir TD (2001) Detecting trends using Spearman's rank correlation coefficient. Environ Forens 2(4):359–362

[7].Hajirahimi Z, Khashei M (2019) Hybrid structures in time series modeling and forecasting: A review. Eng Appl Artif Intell 86:83–106

[8].Mu Y, Liu X, Wang L (2018) A Pearson's correlation coefficient-based decision tree and its parallel implementation. Inf Sci 435:40–58

[9].Navares R, Díaz J, Linares C, Aznarte JL (2018) Comparing ARIMA and computational intelligence methods to forecast daily hospital admissions due to circulatory and respiratory causes in Madrid. Stoch Env Res Risk Assess 32(10):2849–2859

[10].Portet S (2020) A primer on the model selection using the Akaike information criterion. Infect Dis Modell 5:111–128

[11].Rezaee MJ, Yousefi S, Eshkevari M, Valipour M, Saberi M (2020) Risk analysis of health, safety and environment in chemical industry integrating linguistic FMEA, fuzzy inference system and fuzzy DEA. Stoch Env Res Risk Assess 34(1):201–218