A Project Report
on

ANALYSIS AND PREDICTION OF COVID-19 USING TIME SERIES FORECASTING

Submitted in partial fulfillment of the requirements for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SICENCE AND ENGINEERING

by

17WH1A05A3	**Ms. A.PADMASREE**
17WH1A0571	**Ms. T.KAVYA**
17WH1A0587	**Ms. K.SANTHOSHI**

under the esteemed guidance of

Dr. K.Srinivasa Reddy
Professor and HoD

Department of Computer Science and Engineering
BVRIT HYDERABAD
College of Engineering for Women
(NBA Accredited – EEE, ECE, CSE and IT)
(Approved by AICTE, New Delhi and Affiliated to JNTUH, Hyderabad)
Bachupally, Hyderabad – 500090

May, 2021

DECLARATION

We hereby declare that the work presented in this project entitled **“ANALYSIS AND PREDICTION OF COVID-19 USING TIME SERIES FORECASTING”** submitted towards completion of Project Work in IV year of B.Tech., CSE at ‘BVRIT HYDERABAD College of Engineering for Women’, Hyderabad is an authentic record of our original work carried out under the guidance of Dr. K.Srinivasa Reddy, Professor and HoD, Department of CSE.

Padmasree 30/5/21

Sign. with date:
Ms. A.PADMASREE
(17WH1A05A3)

Kavya 30/5/2021

Sign. with date:
Ms. T.KAVYA
(17WH1A0571)

Santhoshi 30/5/2021

Sign. with date:
Ms. K.SANTHOSHI
(17WH1A0587)

BVRIT HYDERABAD
College of Engineering for Women
(NBA Accredited – EEE, ECE, CSE and IT)
(Approved by AICTE, New Delhi and Affiliated to JNTUH, Hyderabad)
Bachupally, Hyderabad – 500090

Department of Computer Science and Engineering

Certificate

This is to certify that the Project Work report on "**ANALYSIS AND PREDICTION OF COVID-19 USING TIME SERIES FORECASTING**" is a bonafide work carried out by Ms. A.PADMASREE (17WH1A05A3) ; Ms. T.KAVYA (17WH1A0571) ; Ms. K.SANTHOSHI (17WH1A0587) in the partial fulfillment for the award of B.Tech. degree in **Computer Science and Engineering, BVRIT HYDERABAD College of Engineering for Women, Bachupally, Hyderabad**, affiliated to Jawaharlal Nehru Technological University Hyderabad, Hyderabad under my guidance and supervision.

The results embodied in the project work have not been submitted to any other University or Institute for the award of any degree or diploma.

Head of the Department	**Guide**
Dr. K.Srinivasa Reddy	**Dr. K.Srinivasa Reddy**
Professor and HoD,	**Professor and HoD,**
Department of CSE	**Department of CSE**

External Examiner

Acknowledgements

We would like to express our sincere thanks to **Dr. K V N Sunitha, Principal**, **BVRIT HYDERABAD College of Engineering for Women**, for providing the working facilities in the college.

Our sincere thanks and gratitude to our **Dr. K.Srinivasa Reddy, Professor**, Department of CSE, **BVRIT HYDERABAD College of Engineering for Women** for all the timely support and valuable suggestions during the period of our project.

We are extremely thankful and indebted to our internal guide, **Dr. K.Srinivasa Reddy, Professor**, Department of CSE**, BVRIT HYDERABAD College of Engineering for Women** for his constant guidance, encouragement and moral support throughout the project.

Finally, we would also like to thank our Project Coordinator, all the faculty and staff of **CSE** Department who helped us directly or indirectly, parents and friends for their cooperation in completing the project work.

Ms. A.PADMASREE

(17WH1A05A3)

Ms. T.KAVYA

(17WH1A0571)

Ms. K.SANTHOSHI

(17WH1A0587)

Contents

ABSTRACT

COVID-19 pandemic is one of the prevalent challenges mankind has ever faced and there is a lot of uncertainty prevailing over the future with respect to it. In this situation, machine learning algorithms can be useful for real-time analysis and prediction of the trends of infections. The objective of our project is to analyze the COVID-19 and vaccination trend globally and individually, and forecast the trend of the pandemic in the near future. Considering the variation of the scenario with time, it has been observed to analyze the data with the time series analysis models. This analysis has been conducted using six time series forecasting methods, viz. AR, MA, ARIMA, Holt's Linear Trend, Holt's Winter Seasonal and FB Prophet. It is of utmost importance to identify the future infected cases, the virus spread rate and time required to vaccinate the total population for advance preparation in the healthcare services to avoid deaths, and social entities to tackle this pandemic across the world. The datasets are taken from Coronavirus COVID-19 Global Cases by the Center for Systems Science and Engineering (CSSE) at Johns Hopkins University (JHU), Dataset on Novel Corona Virus Disease 2019 in India and COVID-19 World Vaccination Progress.

LIST OF FIGURES

1. INTRODUCTION

There exist many devastating epidemics that affected not only human health but also the economic condition of the countries, viz. American polio epidemic (1916), Spanish Flu (1918 – 1920), Asian Flu (1957 -1958), AIDS pandemic and epidemic (1918-present day), H1N1 Swine Flu pandemic (2009-2010), West African Ebola epidemic (2014-2016), Zika Virus epidemic (2015-present day), etc.. But now, the whole world is facing the most dangerous disease COVID-19 due to Coronavirus. The mortality rate due to Coronavirus is different in different countries. This virus is impacting people of all the ages, but it would be more severe for those people who are older and already affected by some diseases like asthma, diabetes, heart disease, etc. This virus is newly developed and discovered for the first time in 2019. This virus is unique, that’s why it will take time to develop its remedies. The symptoms of this disease are sore throat, cough, headaches, etc. Coronavirus enters the body of the human through the mouth, nose and eyes, so we should avoid touching these body parts. Social distancing, covering the mouth and wearing gloves are made mandatory to reduce the contagious pandemic of COVID-19.
Currently, COVID-19 is a green research topic as the whole world is suffering and struggling in this time due to this pandemic disease. Prediction of the future possibilities is very important to support several social entities and organizations, viz. hospitals, pharmaceuticals, NGOs, Government bodies, etc., for their readiness to combat.

1.1 Objectives

In this project work, python and its libraries are applied for the exploratory data analysis of COVID-19 cases and vaccination data. Considering the variation of the scenario with time, it has been observed to analyze the data with the time series analysis (TSA) in order to forecast the future effect of Coronavirus globally or individually. This analysis has been conducted using six forecast methods, viz. Autoregressive (AR), Moving Averages (MA), Autoregressive Integrated Moving Average (ARIMA), Holt’s Linear Trend, Holt’s Winter Seasonal and Facebook Prophet.

1.2 Methodology

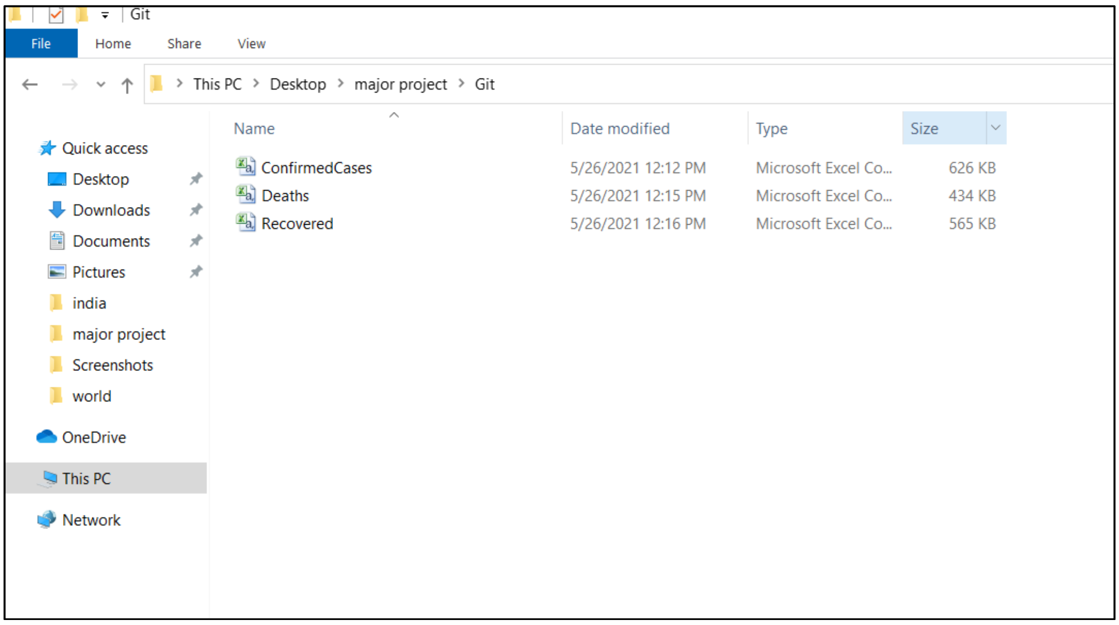
To predict the cases for future forecasting, large collection of data is required. The data is taken from Covid-19 dataset. In this section, the methodology followed is discussed in detail.

1.2.1 Dataset

Proper and large dataset is required for the analysis and forecasting during the training and the testing phase. The datasets are taken from Coronavirus COVID-19 Global Cases by the Center for Systems Science and Engineering (CSSE) at Johns Hopkins University (JHU), Dataset on Novel Corona Virus Disease 2019 in India and COVID-19 World Vaccination Progress.

Using Coronavirus COVID-19 Global Cases by the Center for Systems Science and Engineering (CSSE) at Johns Hopkins University (JHU)

This folder contains daily time series summary tables from 22-01-2020, which includes confirmed, deaths and recovered cases of 276 provinces. The tables are ConfirmedCases.csv, Deaths.csv and Recovered.csv.

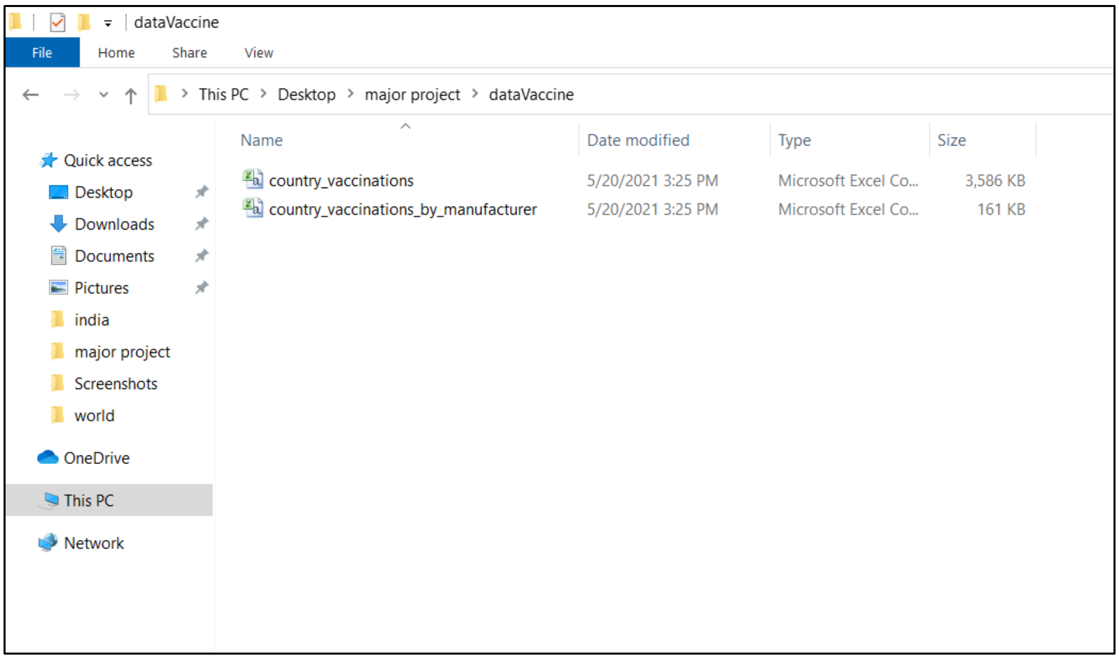


Name	Date modified	Type	Size
ConfirmedCases	5/26/2021 12:12 PM	Microsoft Excel Co...	626 KB
Deaths	5/26/2021 12:15 PM	Microsoft Excel Co...	434 KB
Recovered	5/26/2021 12:16 PM	Microsoft Excel Co...	565 KB

Fig 1.2.1.1: Global Cases Data

Using COVID-19 World Vaccination Progress

This folder contains daily time series summary tables from 02-12-2020, which includes vaccination details like total vaccinations, people vaccinated, daily vaccinations, etc. of 18764 provinces.

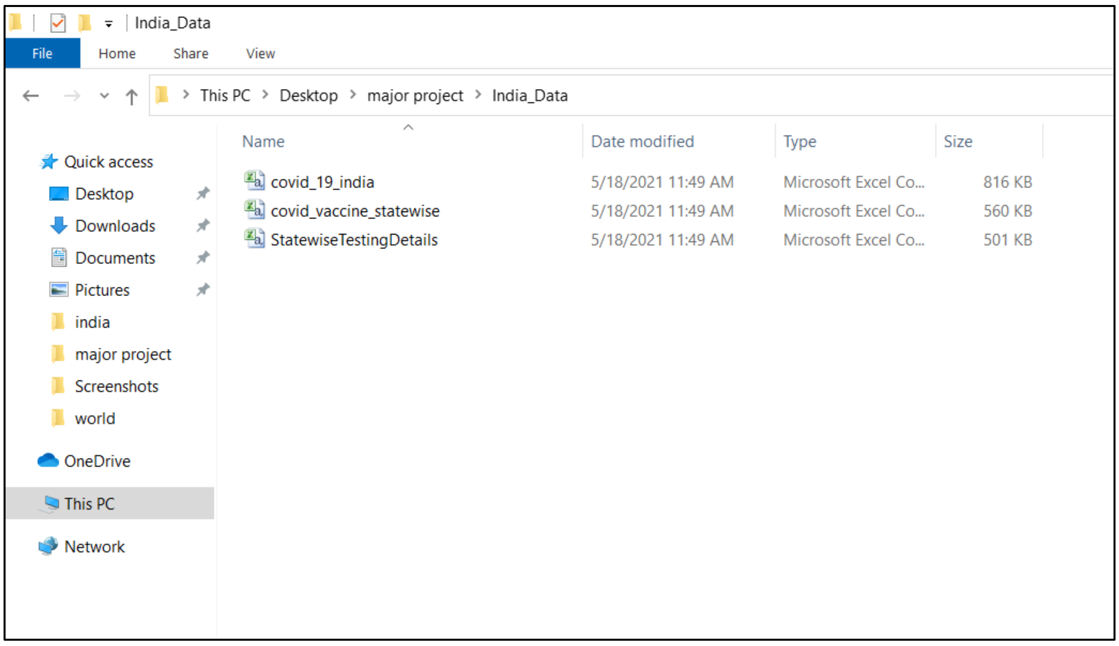


Name	Date modified	Type	Size
country_vaccinations	5/20/2021 3:25 PM	Microsoft Excel Co...	3,586 KB
country_vaccinations_by_manufacturer	5/20/2021 3:25 PM	Microsoft Excel Co...	161 KB

Fig 1.2.1.2: Vaccination Data

Using Dataset on Novel Corona Virus Disease 2019 in India

This folder contains daily time series summary tables from 30-01-2020, which includes details like confirmed, deaths and recovered cases, total samples collected, and vaccination details of 14978 provinces.



Name	Date modified	Type	Size
covid_19_india	5/18/2021 11:49 AM	Microsoft Excel Co...	816 KB
covid_vaccine_statewise	5/18/2021 11:49 AM	Microsoft Excel Co...	560 KB
StatewiseTestingDetails	5/18/2021 11:49 AM	Microsoft Excel Co...	501 KB

Fig 1.2.1.3: State-wise Data(India)

1.2.2 Proposed TSA Models

1.2.2.1 Auto Regressive Model

In a multiple regression model, we forecast the variable of interest using a linear combination of predictors. In an autoregression model, we forecast the variable of interest using a linear combination of past values of the variable. The term autoregression indicates that it is a regression of the variable against itself. Autoregressive models are remarkably flexible at handling a wide range of different time series patterns.

Thus, an autoregressive model of order p can be written as

$$yt=c+\phi 1yt-1+\phi 2yt-2+\cdots+\phi pyt-p+\varepsilon t,$$

where εt is white noise. This is like a multiple regression but with lagged values of yt as predictors. We refer to this as an AR(**p**) model, an autoregressive model of order p.

1.2.2.2 Moving Average Model

Rather than using past values of the forecast variable in a regression, a moving average model uses past forecast errors in a regression-like model.

$$y_t=c+\varepsilon_t+\theta_1\varepsilon_{t-1}+\theta_2\varepsilon_{t-2}+\cdots+\theta_q\varepsilon_{t-q},$$

where ε_t is white noise. We refer to this as an MA(**q**) model, a moving average model of order qq. Of course, we do not observe the values of ε_t, so it is not really a regression in the usual sense.

1.2.2.3 ARIMA Model

ARIMA, short for 'Auto Regressive Integrated Moving Average' is actually a class of models that 'explains' a given time series based on its own past values, that is, its own lags and the lagged forecast errors, so that equation can be used to forecast future values.

An ARIMA model is characterized by 3 terms: p, d, q where,

p is the order of the AR term

q is the order of the MA term

d is the number of differencing required to make the time series stationary

An ARIMA model is one where the time series was differenced at least once to make it stationary and you combine the AR and the MA terms. So the equation becomes:

Predicted Yt = Constant + Linear combination Lags of Y (upto p lags) + Linear Combination of Lagged forecast errors (upto q lags)

1.2.2.4 Holts' Linear Model

Holt extended simple exponential smoothing to allow the forecasting of data with a trend. This method involves a forecast equation and two smoothing equations (one for the level and one for the trend):

Forecast equation - $y_{t+h|t}=\ell_t+hb_t$

Level equation$\ell_t=\alpha y_t+(1-\alpha)(\ell_{t-1}+b_{t-1})$

Trend equation$b_t=\beta^*(\ell_t-\ell_{t-1})+(1-\beta^*)b_{t-1}$,

where ℓ_t denotes an estimate of the level of the series at time t, b_t denotes an estimate of the trend (slope) of the series at time t, α is the smoothing parameter for the level, $0\leq\alpha\leq1$, and β^* is the smoothing parameter for the trend, $0\leq\beta^*\leq1$.

1.2.2.5 Holt’s Winter Model

Holt-Winters is a model of time series behavior. Forecasting always requires a model, and Holt-Winters is a way to model three aspects of the time series: a typical value (average), a slope (trend) over time, and a cyclical repeating pattern (seasonality). Holt-Winters uses exponential smoothing to encode lots of values from the past and use them to predict “typical” values for the present and future

The component form for the additive method is:

$yt+h|t=\ell t+hbt+st+h-m(k+1)$

$\ell t=\alpha(yt-st-m)+(1-\alpha)(\ell t-1+bt-1)$

$bt=\beta*(\ell t-\ell t-1)+(1-\beta*)bt-1$

$st=\gamma(yt-\ell t-1-bt-1)+(1-\gamma)st-m$

1.2.2.6 Facebook Prophet Model

Prophet is an open source library published by Facebook that is based on decomposable (trend+seasonality+holidays) models. It provides us with the ability to make time series predictions with good accuracy using simple intuitive parameters and has support for including impact of custom seasonality and holidays!

We use a decomposable time series model with three main model components: trend, seasonality, and holidays. They are combined in the following equation:

$$y(t) = g(t) + s(t) + h(t) + \epsilon_t$$

- **g(t)**: piecewise linear or logistic growth curve for modelling non-periodic changes in time series
- **s(t)**: periodic changes (e.g. weekly/yearly seasonality)
- **h(t)**: effects of holidays (user provided) with irregular schedules
- **ε_t**: error term accounts for any unusual changes not accommodated by the model

1.2.3 Accuracy Metrics

1.2.3.1 Root Mean Square Error

In RMSE, the errors are squared before they are averaged. This basically implies that RMSE assigns a higher weight to larger errors. This indicates that RMSE is much more useful when large errors are present and they drastically affect the model's performance. It avoids taking the absolute value of the error and this trait is useful in many mathematical calculations. In this metric also, lower the value, better is the performance of the model.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (Predicted_i - Actual_i)^2}{N}}$$

RMSE

Fig 1.2.3.1: RMSE

1.2.3.2 Mean Absolute Error

Mean Absolute Error (MAE) is another loss function used for regression models. MAE is the sum of absolute differences between our target and predicted variables. So it measures the average magnitude of errors in a set of predictions, without considering their directions.

$$MSE = \frac{1}{n} \Sigma \underbrace{\left(y - \widehat{y} \right)^2}_{\text{The square of the difference between actual and predicted}}$$

Fig 1.2.3.2: MAE

1.2.3.3 R2 Score

Coefficient of determination also called as R2 score is used to evaluate the performance of a linear regression model. It is the amount of the variation in the output dependent attribute which is predictable from the input independent variable(s).

$$\boxed{R^2 = 1 - \frac{SS_{RES}}{SS_{TOT}} = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \overline{y})^2}}$$

Fig 1.2.3.3: R2 Score

1.3 Organization of the Project

The project is taking Covid-19 cases and vaccination data as input. Different models like AR, MA, ARIMA, Holt’s linear trend, Holt’s winter seasonal and FB Prophet methods are applied on the input to make future forecasting.

We have three modules in our project.

- World-wide Analysis and Prediction
- Vaccination Analysis and Prediction of Percentages
- India Analysis and Prediction

2. THEORETICAL ANALYSIS OF THE PROPOSED PROJECT

2.1 Requirements Gathering

2.1.1 Software Requirements

Programming Language	: Python 3.6
Dataset	: COVID-19 Cases and Vaccination dataset
Packages	: Numpy, Pandas, Matplotlib, Scikit-learn, IPython, Wget
Tool	: Colab Notebook

2.1.2 Hardware Requirements

Operating System	: Windows 10
Processor	: Intel Core i3-2348M
CPU Speed	: 2.30 GHz
Memory	: 4 GB (RAM)

2.2 Technologies Description

Python

Python is an interpreted high-level programming language for general-purpose programming.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

- Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Wget

Wget (or just Wget, formerly Geturl, also written as its package name, wget) is a computer program that retrieves content from web servers. It is part of the GNU Project. Its name derives from World Wide Web and get. It supports downloading via HTTP, HTTPS, and FTP. Its features include recursive download, conversion of links for offline viewing of local HTML, and support for proxies.

Numpy

Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Pandas

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze.

Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

Scikit – learn

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use. The library is built upon the SciPy (Scientific Python) that must be installed before you can use scikit-learn. This stack that includes:

- **NumPy**: Base n-dimensional array package
- **SciPy**: Fundamental library for scientific computing

- **Matplotlib**: Comprehensive 2D/3D plotting
- **IPython**: Enhanced interactive console
- **Pandas**: Data structures and analysis

plotly

The plotly Python library is an interactive, open-source plotting library that supports over 40 unique chart types covering a wide range of statistical, financial, geographic, scientific, and 3-dimensional use-cases. Plotly's Python graphing library makes interactive, publication-quality graphs. Examples of how to make line plots, scatter plots, area charts, bar charts, error bars, box plots, histograms, heatmaps, subplots, multiple-axes, polar charts, and bubble charts.

GeoPandas

GeoPandas is an open source project to make working with geospatial data in python easier. GeoPandas extends the datatypes used by pandas to allow spatial operations on geometric types. Geometric operations are performed by shapely. Geopandas further depends on fiona for file access and matplotlib for plotting.The goal of GeoPandas is to make working with geospatial data in python easier.

Seaborn

Seaborn is a library for making statistical graphics in Python. It builds on top of matplotlib and integrates closely with pandas data structures. Seaborn helps you explore and understand your data. Its plotting functions operate on dataframes and arrays containing whole datasets and internally perform the necessary semantic mapping and statistical aggregation to produce informative plots.

Colab Notebook

Colaboratory, or “Colab” for short, is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education. More technically, Colab is a hosted Jupyter notebook service that requires no setup to use, while providing free access to computing resources including GPUs.

3. DESIGN

3.1 Introduction

Design is the first step in the development phase for any engineered product or system. The designer’s goal is to produce a model or representation of an entity that will later be built. Beginning, once system requirement have been specified and analyzed, system design is the first of the three technical activities -design, code and test that is required to build and verify software.

During design, progressive refinement of data structure, program structure, and procedural details are developed reviewed and documented. System design can be viewed from either technical or project management perspective. The core design skills such as identifying and communicating needs, visualizing data, prototyping, building tools, and doing research all play important roles in the core machine learning process. From the technical point of view, design is comprised of four activities – architectural design, data structure design, interface design and procedural design.

3.2 Architecture Diagram

The architecture diagram is a schematic representation of a collection of ideas that are part of an architecture including its principles, elements and materials. Architecture diagram will support designers and engineers in visualizing a system or application's high-level, overarching layout to ensure the framework addresses the needs of their customers. The machine learning architecture defines the various layers involved in the machine learning cycle and involves the major steps being carried out in the transformation of raw data into training data sets capable for enabling the decision making of a system.

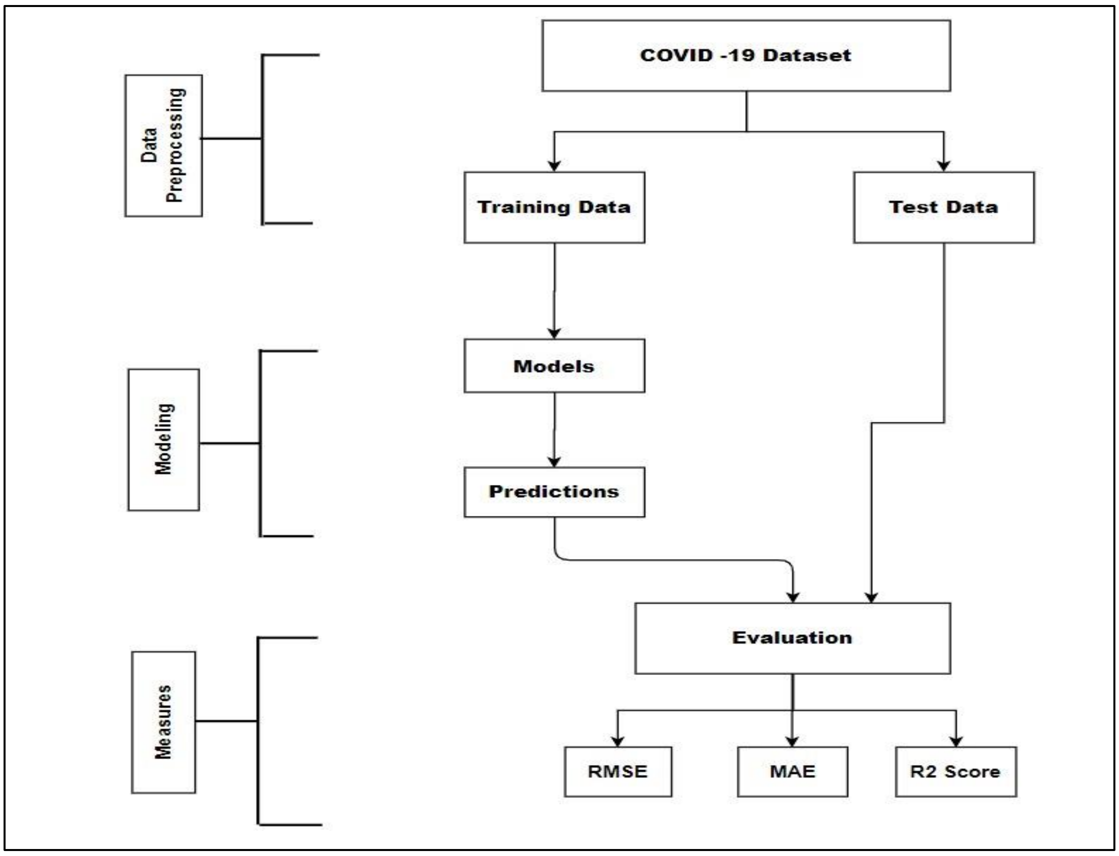


Fig 3.2: Architecture Diagram

3.3 UML Diagrams

3.3.1 Use Case Diagram

To model a system, the most important aspect is to capture the dynamic behavior. Dynamic behavior means the behavior of the system when it is running/operating.

Only static behavior is not sufficient to model a system rather dynamic behavior is more important than static behavior. In UML, there are five diagrams available to model the dynamic nature and use case diagram is one of them. Now as we have to discuss that the use case diagram is dynamic in nature, there should be some internal or external factors for making the interaction.

These internal and external agents are known as actors. Use case diagrams consist of actors, use cases and their relationships. The diagram is used to model the system/subsystem of an application. A single use case diagram captures a particular functionality of a system.

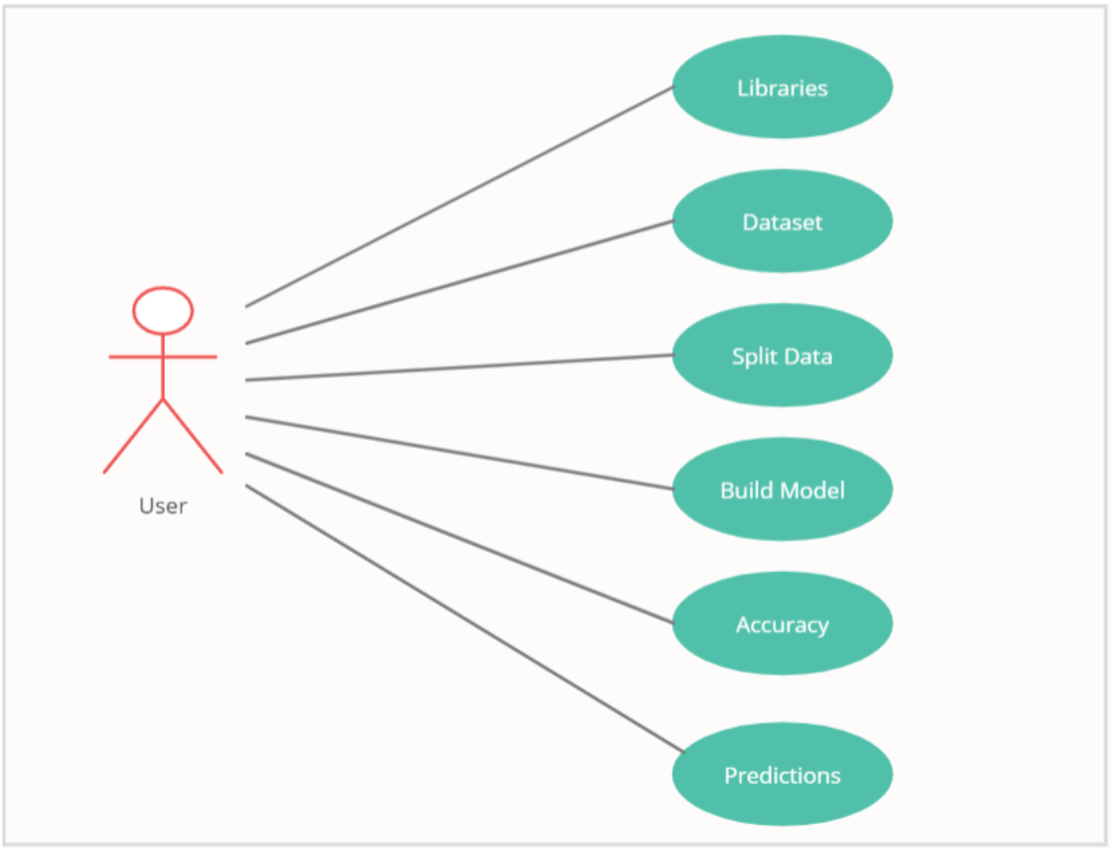


Fig 3.3.1: Use Case Diagram

3.3.2 Sequence Diagram

Sequence Diagrams Represent the objects participating the interaction horizontally and time vertically. A Use Case is a kind of behavioral classifier that represents a declaration of an offered behavior. Each use case specifies some behavior, possibly including variants that the subject can perform in collaboration with one or more actors. Use cases define the offered behavior of the subject without reference to its internal structure. These behaviors, involving interactions between the actor and the subject, may result in changes to the state of the subject and communications with its environment. A use case can include possible variations of its basic behavior, including exceptional behavior and error handling.

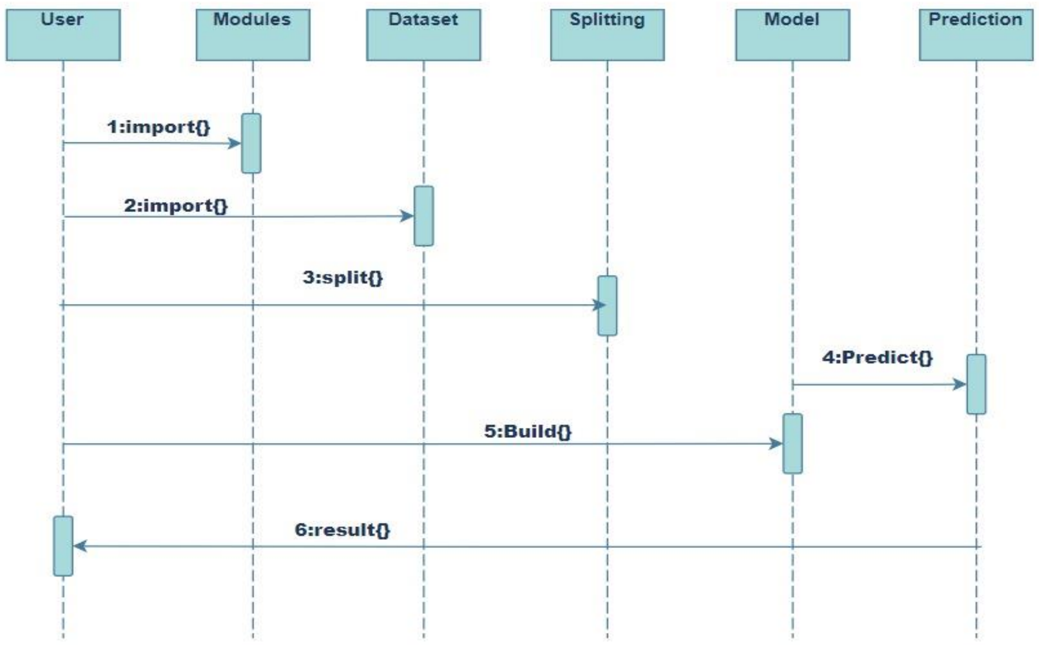


Fig 3.3.2: Sequence Diagram

3.3.3 Activity Diagram

Activity diagrams are graphical representations of Workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

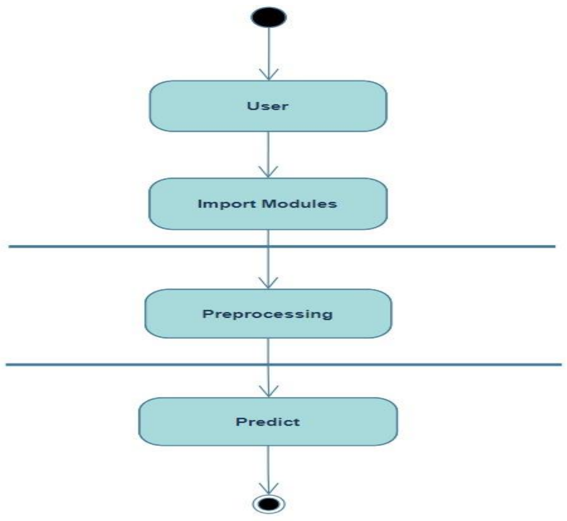


Fig 3.3.3: Activity Diagram

3.3.4 Collaboration Diagram

A collaboration diagram resembles a flowchart that portrays the roles, functionality and behavior of individual objects as well as the overall operation of the system in real time. Objects are shown as rectangles with naming labels inside. The relationships between the objects are shown as lines connecting the rectangles.

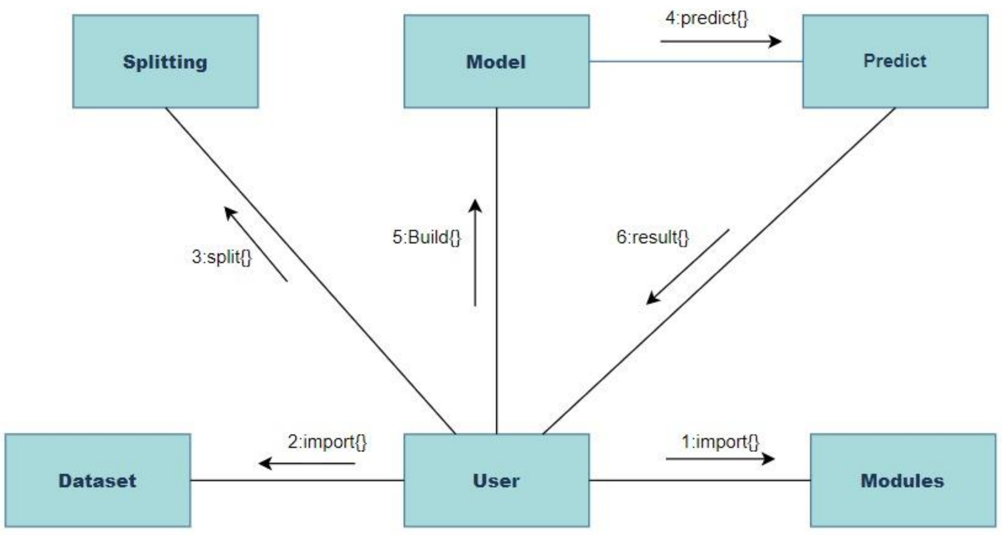


Fig 3.3.4:Collabration Diagram

3.3.5 Class Diagram

The class diagram is the main building block of object-oriented modeling. It is used for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code.The classes in a class diagram represent both the main elements, interactions in the application, and the classes to be programmed.

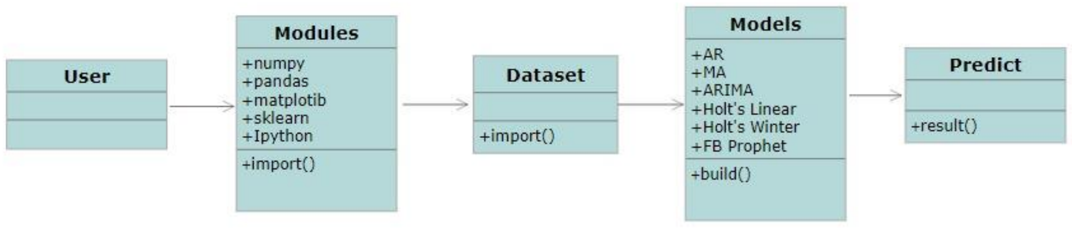


Fig 3.3.5: Class Diagram

4. IMPLEMENTATION

4.1 Coding

WorldAnalysis.py

```
! pip install wget
# Data Processing
import numpy as np
import pandas as pd
import wget
from datetime import datetime, timedelta
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.svm import SVR
from statsmodels.tsa.api import Holt,SimpleExpSmoothing,ExponentialSmoothing
from sklearn.metrics import mean_squared_error as mse
from sklearn.metrics import r2_score
import statsmodels.api as sm
from sklearn.metrics import mean_absolute_error as mae
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objs as go
import plotly.figure_factory as ff
import folium
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
from IPython.display import Javascript
from IPython.core.display import display, HTML
pd.set_option('display.max_columns', 100)
pd.set_option('display.max_rows', 100)
print("Setup Complete")
```

```
yesterday = datetime.today() - timedelta(days=2)
yesterday = yesterday.strftime('%m-%d-%Y')
yesterday
urls = ['https://raw.githubusercontent.com/CSSEGISandData/COVID-
19/master/csse_covid_19_data/csse_covid_19_time_series/time_series_covid19_confi
rmed_global.csv',
    'https://raw.githubusercontent.com/CSSEGISandData/COVID-
19/master/csse_covid_19_data/csse_covid_19_time_series/time_series_covid19_deat
hs_global.csv','https://raw.githubusercontent.com/CSSEGISandData/COVID19/maste
r/csse_covid_19_data/csse_covid_19_time_series/time_series_covid19_recovered_gl
obal.csv',f'https://raw.githubusercontent.com/CSSEGISandData/COVID19/master/css
e_covid_19_data/csse_covid_19_daily_reports/{yesterday}.csv']
[wget.download(url) for url in urls]
confirmed_df = pd.read_csv('time_series_covid19_confirmed_global.csv')
death_df = pd.read_csv('time_series_covid19_deaths_global.csv')
recovered_df = pd.read_csv('time_series_covid19_recovered_global.csv')
df = pd.read_csv(f'{yesterday}.csv')
df.tail()
dates = confirmed_df.columns[4:]
confirmed_df_long = confirmed_df.melt(id_vars=['Province/State', 'Country/Region',]
,value_vars=dates, var_name='Date', value_name='Confirmed')
death_df_long = death_df.melt(id_vars=['Province/State', 'Country/Region',],
value_vars=dates, var_name='Date', value_name='Deaths')
recovered_df_long = recovered_df.melt( id_vars=['Province/State', 'Country/Region',]
, value_vars=dates,   var_name='Date', value_name='Recovered')
full_table = confirmed_df_long.merge(right=death_df_long, how='left')
full_table = full_table.merge(right=recovered_df_long, how='left')
full_table.head()
full_table['Date'] = pd.to_datetime(full_table['Date'])
full_table.isna().sum()
full_table['Recovered'] = full_table['Recovered'].fillna(0).astype(int)
ship_rows = full_table['Province/State'].str.contains('Grand Princess') | full_table['Pro
vince/State'].str.contains('Diamond Princess') | full_table['Country/Region'].str.contain
s('Diamond Princess') | full_table['Country/Region'].str.contains('MS Zaandam')
```

```
ship_df = full_table[ship_rows]
ship_df.head()
full_table = full_table[~(ship_rows)]
full_table['Active'] = full_table['Confirmed'] - full_table['Deaths'] -
 full_table['Recovered']
full_table.head()
full_grouped = full_table.groupby(['Date', 'Country/Region'])['Confirmed','Deaths', 'R
ecovered', 'Active'].sum().reset_index()
full_grouped
temp = full_grouped.groupby(['Country/Region', 'Date', ])['Confirmed', 'Deaths', 'Reco
vered']
temp = temp.sum().diff().reset_index()
mask = temp['Country/Region'] != temp['Country/Region'].shift(1)
temp.loc[mask, 'Confirmed'] = np.nan
temp.loc[mask, 'Deaths'] = np.nan
temp.loc[mask, 'Recovered'] = np.nan
temp.columns = ['Country/Region', 'Date', 'New cases', 'New deaths', 'New recovered']
# merging new values
full_grouped = pd.merge(full_grouped, temp, on=['Country/Region', 'Date'])
full_grouped = full_grouped.fillna(0)
# fixing data types
cols = ['New cases', 'New deaths', 'New recovered']
full_grouped[cols] = full_grouped[cols].astype('int')
full_grouped['New cases'] = full_grouped['New cases'].apply(lambda x: 0 if x<0 else
x)
full_grouped.tail()

#World-wide cases
temp = full_grouped.groupby('Date')['Confirmed', 'Deaths', 'Recovered', 'Active'].sum
().reset_index()
temp = temp[temp['Date']==max(temp['Date'])].reset_index(drop=True)
world_cases = temp.to_numpy()
temp.style.background_gradient(cmap='Pastel1')
```

```
temp = full_grouped.groupby(['Country/Region'])['Confirmed', 'Deaths', 'Recovered', '
Active', 'New cases', 'New deaths', 'New recovered'].max()
#Sorting by taking confirmed cases
temp.sort_values('Confirmed', ascending=False).style.bar(subset=['Confirmed', 'Death
s', 'Recovered', 'Active', 'New cases', 'New deaths', 'New recovered'],align = 'left', colo
r='#d65f5f')
temp = full_grouped.groupby('Date')['Confirmed', 'Deaths', 'Recovered', 'Active'].sum
().reset_index()
temp = temp.melt(id_vars="Date", value_vars=['Confirmed', 'Deaths', 'Recovered', 'A
ctive'], var_name='Case', value_name='Count')
print(temp)
fig = px.line(temp, x="Date", y="Count", color='Case',title='Cases over time')
fig
# adding two more columns
temp = full_grouped
temp['Mortality Rate'] = round(temp['Deaths']/temp['Confirmed'], 3)
temp['Recovery Rate'] = round(temp['Recovered']/temp['Confirmed'], 3)
temp.groupby(['Country/Region'])['Recovery Rate','Mortality Rate'].max().sort_values
('Recovery Rate', ascending=False).style.background_gradient(cmap='Reds')
temp = full_grouped.groupby('Date').sum().reset_index()
temp['Mortality Rate'] = round(temp['Deaths']/temp['Confirmed'], 3)
temp['Recovery Rate'] = round(temp['Recovered']/temp['Confirmed'], 3)
temp = temp.melt(id_vars='Date', value_vars=['Mortality Rate', 'Recovery Rate'],var_
name='Ratio', value_name='Value')
print(temp)
fig = px.line(temp, x="Date", y="Value", color='Ratio',title='Recovery and Mortality
Rate Over The Time')
fig
temp = full_grouped.groupby('Date')['New cases', 'New deaths', 'New recovered'].sum
().reset_index()
temp = temp.melt(id_vars="Date", value_vars=['New cases', 'New deaths', 'New recov
ered'],var_name='Case', value_name='Count')
print(temp)
fig = px.line(temp, x="Date", y="Count", color='Case',title='Daily Cases')
```

```
fig
country_grouped = df.groupby('Country_Region')['Confirmed', 'Deaths', 'Recovered', '
Active'].sum().reset_index()
country_grouped['Active'] = country_grouped['Active'].astype(int)
country_grouped = country_grouped.sort_values('Confirmed', ascending=False)
others_series = pd.Series(np.sum(country_grouped[10:]))
country_grouped_others = country_grouped[:10] #First ten countries
country_grouped_others = country_grouped_others.append(others_series, ignore_inde
x=True)
country_grouped_others.iloc[10,0] = 'Rest of the World'
fig = px.choropleth(country_grouped, locations="Country_Region",locationmode='co
untry names', color="Confirmed",hover_name="Country_Region", range_color=[1,70
0000],color_continuous_scale="aggrnyl",title='Countries with Confirmed Cases')
fig
fig = px.pie(country_grouped_others, values='Confirmed', names='Country_Region', t
itle='Confirmed Cases')
fig.update_traces(textinfo='percent+label')
fig.show()
fig = px.bar(country_grouped.head(20).sort_values('Confirmed', ascending=True),x="
Confirmed", y="Country_Region",title='Top 20 Countries Confirmed Cases',text='Co
nfirmed', orientation='h',width=700, height=700)
fig.update_traces(opacity=0.6)
fig
# Deaths
fig = px.choropleth(country_grouped[country_grouped['Deaths']>0],locations="Count
ry_Region", locationmode='country names',color="Deaths", hover_name
="Country_Region",range_color=[1,50000], color_continuous_scale="agsunset",title
='Countries with Deaths Reported')
fig
fig = px.pie(country_grouped_others.sort_values('Deaths', ascending=False), values='
Deaths', names='Country_Region', title='Total Deaths')
fig.update_traces(textinfo='percent+label')
fig.show()
```

```
# import Packages for Prediction
!pip install pmdarima
from pmdarima.arima import auto_arima
from sklearn.metrics import mean_squared_error
from datetime import timedelta
from fbprophet import Prophet
full_grouped = full_grouped.rename(columns = {'Date': 'ds'})
df_group = full_grouped.groupby(by = 'ds')['Confirmed', 'Deaths', 'Recovered', 'Active
'].sum()
df_group.index = pd.to_datetime(df_group.index)
df_group = df_group.asfreq(freq = '1D')
df_group = df_group.sort_index(ascending = True)
df_group = df_group.fillna(value = 0)
df_group = df_group.rename(columns = {'Date': 'ds'})
display(df_group.tail())
display(df_group.head())

#AR Model
model_train = df_group.iloc[:int(df_group.shape[0]*0.95)]
valid = df_group.iloc[int(df_group.shape[0]*0.95):]
y_pred = valid.copy()
print(model_train,valid)
model_ar = auto_arima(model_train["Confirmed"],start_q=0,max_q=0,start_p =0, ma
x_p = 7,stepwise= False,method = 'cg',trace=True, error_action='ignore')
model_ar.fit(model_train["Confirmed"])
sample_len = len(valid)#to store the length of the sample
prediction_ar = model_ar.predict(sample_len)
print(sample_len)
y_pred["AR Model Prediction Confirmed" ]=prediction_ar
#RMSE
rmse_scores = [] #for storing rmse values of all the models
ar_rmse_score = (np.sqrt(mse(y_pred["Confirmed"],y_pred["AR Model Prediction C
onfirmed"])))/sample_len
rmse_scores.append(["AR RMSE Score:",ar_rmse_score])
```

```
print("Root Mean Square Error for AR Model: ",ar_rmse_score)
print(rmse_scores)
#MAE Score
mae_scores = []
ar_mae_score = mae(y_pred["Confirmed"],y_pred["AR Model Prediction Confirmed"
])/sample_len
mae_scores.append(["AR MAE Score:", ar_mae_score])
print("AR- MAE Score:", ar_mae_score)
print(mae_scores)
#R2 Score
r2_scores = []
ar_r2_score = r2_score(y_pred["Confirmed"],y_pred["AR Model Prediction Confirme
d"])
r2_scores.append(["AR R2 Score:", ar_r2_score])
print("AR- R2 Score:", ar_r2_score)
print(r2_scores)
fig=go.Figure()
fig.add_trace(go.Scatter(x=model_train.index, y=model_train["Confirmed"],
mode='lines+markers',name="Train Data for Confirmed Cases"))
fig.add_trace(go.Scatter(x=valid.index, y=valid["Confirmed"],
mode='lines+markers',name="Validation Data for Confirmed Cases",))
fig.add_trace(go.Scatter(x=valid.index, y=y_pred["AR Model Prediction Confirmed"]
,mode='lines',name="Prediction of Confirmed Cases",line=dict(color='black', dash='d
ot')))
fig.update_layout(title="Confirmed Cases AR Model Prediction",xaxis_title="Date",y
axis_title="Confirmed Cases")
fig.show()
AR_model_new_prediction=[]
new_date=[]
last_date = df_group.index[-1]
model = model_ar.predict(len(valid)+12)
for i in range(2,12):
    new_date.append(last_date+timedelta(days=i))
    print(model[len(valid)+i])
    AR_model_new_prediction.append(model[len(valid)+i])
pd.options.display.float_format = '{:.3f}'.format
model_predictions=pd.DataFrame(zip(new_date,AR_model_new_prediction),column
s=['Dates', 'AR_model_new_prediction'])
print(model_predictions)
#MA Model
model_train = df_group.iloc[:int(df_group.shape[0]*0.95)]
valid = df_group.iloc[int(df_group.shape[0]*0.95):]
y_pred = valid.copy()
model_ma= auto_arima(model_train["Confirmed"],trace=True, error_action='ignore',
method='nm', start_p=0,max_p=0,start_q=0,max_q=5)
model_ma.fit(model_train["Confirmed"])
prediction_ma=model_ma.predict(len(valid))
y_pred["MA Model Prediction"]=prediction_ma
#RMSE
ma_rmse_score = (np.sqrt(mse(valid["Confirmed"],prediction_ma)))/sample_len
rmse_scores.append(["MA RMSE Score:",ma_rmse_score])
print("Root Mean Square Error for MA Model: ",ma_rmse_score)
print(rmse_scores)
#MAE Score
ma_mae_score = mae(valid["Confirmed"],prediction_ma)/sample_len
mae_scores.append(["MA MAE Score:", ma_mae_score])
print("MA -MAE Score:", ma_mae_score)
print(mae_scores)
#R2 Score
ma_r2_score = r2_score(valid["Confirmed"],prediction_ma)
r2_scores.append(["MA R2 Score:", ma_r2_score])
print(" MA - R2 Score:", ma_r2_score)
print(r2_scores)
fig=go.Figure()
fig.add_trace(go.Scatter(x=model_train.index, y=model_train["Confirmed"],
mode='lines+markers',name="Train Data for Confirmed Cases"))
fig.add_trace(go.Scatter(x=valid.index, y=valid["Confirmed"],
mode='lines+markers',name="Validation Data for Confirmed Cases",))
```

```
fig.add_trace(go.Scatter(x=valid.index, y=y_pred["MA Model Prediction"],
mode='lines',name="Prediction for Confirmed Cases",line=dict(color='black',
dash='dot')))
fig.update_layout(title="Confirmed Cases MA Model Prediction",xaxis_title="Date",
yaxis_title="Confirmed Cases")
fig.show()
MA_model_new_prediction=[]
for i in range(2, 12):
    MA_model_new_prediction.append(model_ma.predict(len(valid)+i)[-1])
model_predictions["MA Model Prediction"]=MA_model_new_prediction
print(model_predictions)
#Holt's Linear Trend Model
model_train=df_group.iloc[:int(df_group.shape[0]*0.95)]
valid=df_group.iloc[int(df_group.shape[0]*0.95):]
y_pred=valid.copy()
holt=Holt(np.asarray(model_train["Confirmed"])).fit(smoothing_level=0.2, smoothin
g_slope=0.2)
y_pred["Holt"]=holt.forecast(len(valid))
#RMSE
hl_rmse_score = np.sqrt(mse(y_pred["Confirmed"],y_pred["Holt"]))/sample_len
rmse_scores.append(["Holt's Linear RMSE Score:",hl_rmse_score])
print("Root Mean Square Error for Holt's Linear Model: ",hl_rmse_score)
print(rmse_scores)
#MAE Score
hl_mae_score = mae(y_pred["Confirmed"],y_pred["Holt"])/sample_len
mae_scores.append(["Holt's Linear MAE Score", hl_mae_score])
print("olt's Linear MAE Score:", hl_mae_score)
print(mae_scores)
#R2 Score
hl_r2_score = r2_score(y_pred["Confirmed"],y_pred["Holt"])
r2_scores.append(["Holt's Linear R2 Score", hl_r2_score])
print("Holt's Linear R2 Score:", hl_r2_score)
print(r2_scores)
fig=go.Figure()
```

```
fig.add_trace(go.Scatter(x=model_train.index, y=model_train["Confirmed"],
mode='lines+markers',name="Train Data for Confirmed Cases"))
fig.add_trace(go.Scatter(x=valid.index, y=valid["Confirmed"],
mode='lines+markers',name="Validation Data for Confirmed Cases",))
fig.add_trace(go.Scatter(x=valid.index, y=y_pred["Holt"],
mode='lines',name="Prediction of Confirmed Cases",line=dict(color='black',
dash='dot')))
fig.update_layout(title="Confirmed Cases Holt's Linear Model Prediction",
xaxis_title="Date",yaxis_title="Confirmed Cases",legend=dict(x=0,y=1,traceorder="n
ormal"))
fig.show()
holt_new_prediction=[]
for i in range(2,12):
    holt_new_prediction.append(holt.forecast((len(valid)+i))[-1])
model_predictions["Holt's Linear Model Prediction"]=holt_new_prediction
model_predictions.head()
#Holt's Winter Model
model_train=df_group.iloc[:int(df_group.shape[0]*0.95)]
valid=df_group.iloc[int(df_group.shape[0]*0.95):]
y_pred=valid.copy()
es=ExponentialSmoothing(np.asarray(model_train['Confirmed']),seasonal_periods=11
, trend='mul', seasonal='mul').fit()
y_pred["Holt's Winter Model"]=es.forecast(len(valid))
#RMSE
hw_rmse_score = np.sqrt(mse(y_pred["Confirmed"],y_pred["Holt's Winter Model"]))
/sample_len
rmse_scores.append(["Holt's Winter RMSE Score:",hw_rmse_score])
print("Root Mean Square Error for Holt's Winter Model: ",hw_rmse_score)
print(rmse_scores)
#MAE Score
hw_mae_score = mae(y_pred["Confirmed"],y_pred["Holt's Winter Model"])/sample_
len
mae_scores.append(["Holt's Winter MAE Score", hw_mae_score])
print("Holt's Winter MAE Score:", hw_mae_score)
print(mae_scores)
#R2 Score
hw_r2_score = r2_score(y_pred["Confirmed"],y_pred["Holt's Winter Model"])
r2_scores.append(["Holt's Winter R2 Score", hw_r2_score])
print("Holt's Winter R2 Score:", hw_r2_score)
print(r2_scores)
fig=go.Figure()
fig.add_trace(go.Scatter(x=model_train.index, y=model_train["Confirmed"],
mode='lines+markers',name="Train Data for Confirmed Cases"))
fig.add_trace(go.Scatter(x=valid.index, y=valid["Confirmed"],mode='lines+markers',
name="Validation Data for Confirmed Cases",))
fig.add_trace(go.Scatter(x=valid.index, y=y_pred["Holt\'s Winter Model"],
mode='lines',name="Prediction of Confirmed Cases",line=dict(color='black',
dash='dot')))
fig.update_layout(title="Confirmed Cases Holt's Winter Model Prediction",
xaxis_title="Date",yaxis_title="Confirmed Cases",legend=dict(x=0,y=1,traceorder="n
ormal"))
fig.show()
holt_winter_new_prediction=[]
for i in range(2,12):
    holt_winter_new_prediction.append(es.forecast((len(valid)+i))[-1])
model_predictions["Holt's Winter Model Prediction"]=holt_winter_new_prediction
model_predictions.head()
#ARIMA Model
model_train = df_group.iloc[:int(df_group.shape[0]*0.95)]
valid = df_group.iloc[int(df_group.shape[0]*0.95):]
y_pred = valid.copy()
print(model_train)
model_arima= auto_arima(model_train["Confirmed"],method='cg',trace=True,stepwis
e=False,start_p=0,start_q=0,max_p=2,max_q=2)
model_arima.fit(model_train["Confirmed"])
prediction_arima=model_arima.predict(len(valid))
y_pred["ARIMA Model Prediction"]=prediction_arima
#RMSE
```

```
arima_rmse_score = np.sqrt(mse(valid["Confirmed"],prediction_arima))/sample_len
rmse_scores.append(["ARIMA RMSE Score:",arima_rmse_score])
print("Root Mean Square Error for ARIMA Model: ",arima_rmse_score)
print(rmse_scores)
#MAE Score
arima_mae_score = mae(valid["Confirmed"],prediction_arima)/sample_len
mae_scores.append(["ARIMA MAE Score", arima_mae_score])
print("ARIMA MAE Score:", arima_mae_score)
print(mae_scores)
#R2 Score
arima_r2_score = r2_score(valid["Confirmed"],prediction_arima)
r2_scores.append(["ARIMA R2 Score", arima_r2_score])
print("ARIMA R2 Score:", arima_r2_score)
print(r2_scores)
fig=go.Figure()
fig.add_trace(go.Scatter(x=model_train.index, y=model_train["Confirmed"],
mode='lines+markers',name="Train Data for Confirmed Cases"))
fig.add_trace(go.Scatter(x=valid.index, y=valid["Confirmed"],mode='lines+markers',
name="Validation Data for Confirmed Cases",))
fig.add_trace(go.Scatter(x=valid.index, y=y_pred["ARIMA Model Prediction"],mode
='lines',name="Prediction for Confirmed Cases",line=dict(color='black', dash='dot')))
fig.update_layout(title="Confirmed Cases ARIMA Model Prediction",xaxis_title="Da
te",yaxis_title="Confirmed Cases")
fig.show()
ARIMA_model_new_prediction=[]
for i in range(2,12):
    ARIMA_model_new_prediction.append(model_arima.predict(len(valid)+i)[-1])
model_predictions["ARIMA Model Prediction"]=ARIMA_model_new_prediction
model_predictions.head()
#Fb prophet
df_prophet = df_group[['Confirmed']]
df_prophet = df_prophet.reset_index()
df_prophet = df_prophet.rename(columns = {'ds': 'ds', 'Confirmed': 'y'})
df_prophet['ds'] = pd.to_datetime(df_prophet['ds'])
```

```
m = Prophet(growth='linear', n_changepoints=30, changepoint_range=0.85)
m.fit(df_prophet)
future = m.make_future_dataframe(periods = 37)
forecast = m.predict(future)
fbprophet_score = np.sqrt(mean_squared_error(df_group["Confirmed"],forecast['yhat'
].head(df_group.shape[0])))
#RMSE
fbprophet_rmse_score = np.sqrt(mse(df_group["Confirmed"],forecast['yhat'].head(df_
group.shape[0])))/sample_len
rmse_scores.append(["FB Prophet RMSE Score:",fbprophet_rmse_score])
print("Root Mean Square Error for FB Prophet Model: ",fbprophet_rmse_score)
print(rmse_scores)
#MAE Score
fbprophet_mae_score = mae(df_group["Confirmed"],forecast['yhat'].head(df_group.s
hape[0]))/sample_len
mae_scores.append(["FB Prophet MAE Score", fbprophet_mae_score])
print("FB Prophet MAE Score:", fbprophet_mae_score)
print(mae_scores)
#R2 Score
fbprophet_r2_score = r2_score(df_group["Confirmed"],forecast['yhat'].head(df_group
.shape[0]))
r2_scores.append(["FB Prophet R2 Score", fbprophet_r2_score])
print("FB Prophet R2 Score:", fbprophet_r2_score)
print(r2_scores)
figure = m.plot(forecast, xlabel = 'Date', ylabel = 'Confirmed Cases')
figure2 = m.plot_components(forecast)
model_predictions["Prophet's Prediction"]=list(forecast["yhat"].tail(10))
model_predictions.head()
#Summary
rmse_summary = pd.DataFrame(rmse_scores,columns=["Model Name","Root Mean
Squared Error"]).sort_values(["Root Mean Squared Error"])
mae_summary = pd.DataFrame(mae_scores,columns=["Model Name","Mean Absolut
e Error"]).sort_values(["Mean Absolute Error"])
```

```
r2_score_summary = pd.DataFrame(r2_scores,columns=["Model Name","R2 Score"]
).sort_values(["R2 Score"],ascending = False)
print(rmse_summary,"\n\n",mae_summary,"\n\n",r2_score_summary)
```

India_Analysis_Predictions.py

```
!pip install chart_studio
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import plotly
import plotly.express as px
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.svm import SVR
from statsmodels.tsa.api import Holt,SimpleExpSmoothing,ExponentialSmoothing
from sklearn.metrics import mean_squared_error as mse
from sklearn.metrics import r2_score
import statsmodels.api as sm
from sklearn.metrics import mean_absolute_error as mae
!pip install pmdarima
from pmdarima.arima import auto_arima
from sklearn.metrics import mean_squared_error
from datetime import timedelta
from fbprophet import Prophet
import plotly.graph_objects as go
import cufflinks as cf
import plotly.offline as pyo
from plotly.offline import init_notebook_mode,plot,iplot
import folium
from folium import plugins
plt.rcParams['figure.figsize'] = 10, 12
```

```
import warnings
warnings.filterwarnings('ignore')
from google.colab import drive
drive.mount("/content/drive")
df= pd.read_csv("/content/drive/MyDrive/Covid-19-Project/Major-Project-
Datasets/covid_19_india.csv")
df.tail()
df.isnull().sum()
df.shape
df.drop(['ConfirmedIndianNational','ConfirmedForeignNational'],axis=1,inplace=Tru
e)
df['Date'] = pd.to_datetime(df['Date'],dayfirst = True)
df_confirmed_india=df.groupby('Date')['Confirmed'].sum().reset_index()
df_cured_india=df.groupby('Date')['Cured'].sum().reset_index()
df_death_india=df.groupby('Date')['Deaths'].sum().reset_index()
print("The Confirmed Cases are",df_confirmed_india.Confirmed.max())
print("The Recovered Cases are",df_cured_india.Cured.max())
print("The Deaths Cases are",df_death_india.Deaths.max())
df.shape
fig = go.Figure()
fig.add_trace(go.Scatter(x=df_confirmed_india['Date'], y =
df_confirmed_india['Confirmed'], mode='lines+markers',name='Total Cases'))
fig.update_layout(title_text='Confirmed Cases in
India',xaxis_title="Date",yaxis_title="Confirmed Cases",plot_bgcolor='rgb(230, 230,
230)')
fig.show()
fig = go.Figure()
fig.add_trace(go.Scatter(x=df_cured_india['Date'], y = df_cured_india['Cured'],
mode='lines+markers',name='Total Cases'))
fig.update_layout(title_text='Recovered Cases in
India',xaxis_title="Date",yaxis_title="Recovered Cases",plot_bgcolor='rgb(230, 230,
230)')
fig.show()
fig = go.Figure()
```

```
fig.add_trace(go.Scatter(x=df_death_india['Date'], y = df_death_india['Deaths'],
mode='lines+markers',name='Total Deaths',line=dict(color='Red', width=2)))
fig.update_layout(title_text='Death Cases in
India',xaxis_title="Date",yaxis_title="Death Cases",plot_bgcolor='rgb(230, 230,
230)')
fig.update_layout(title_text='Coronavirus Deaths in India on daily basis')
fig.show()
fig = go.Figure()
fig.add_trace(go.Scatter(x=df_confirmed_india['Date'],
y=df_confirmed_india['Confirmed'], mode='lines+markers',
name='Confirmed',line=dict(color='blue', width=2)))
fig.add_trace(go.Scatter(x=df_death_india['Date'], y=df_death_india['Deaths'],
mode='lines+markers', name='Deaths', line=dict(color='Red', width=2)))
fig.add_trace(go.Scatter(x=df_cured_india['Date'], y=df_cured_india['Cured'],
mode='lines+markers', name='Recovered', line=dict(color='Green', width=2)))
fig.update_layout(title='India COVID-19 Cases',
xaxis_tickfont_size=14,yaxis=dict(title='Number of Cases'))
fig.show()
state_cases =
df.groupby('State/UnionTerritory')['Confirmed','Deaths','Cured'].max().reset_index()
state_cases['Active'] = state_cases['Confirmed'] - (state_cases['Deaths']+
state_cases['Cured'])
state_cases["Death Rate (per 100)"] =
np.round(100*state_cases["Deaths"]/state_cases["Confirmed"],2)
state_cases["Cure Rate (per 100)"] =
np.round(100*state_cases["Cured"]/state_cases["Confirmed"],2)
state_cases.head()
!pip install --upgrade plotly
px.treemap(state_cases,path=["State/UnionTerritory"],values="Deaths",title="Overall
States  Comparision of deaths")
px.treemap(state_cases,path=["State/UnionTerritory"],values="Confirmed",title="Ove
rall States Comparision of Confirmed")
px.treemap(state_cases,path=["State/UnionTerritory"],values="Cured",title="Overall
States Comparision of Recovered cases")
```

```
top_10=state_cases.groupby('State/UnionTerritory')['Confirmed'].sum().sort_values(a
scending=False).reset_index()
trace = go.Table(domain=dict(x=[0, 0.52],y=[0, 1.0]),
header=dict(values=["State/UnionTerritory","Confirmed Cases"],fill = dict(color =
'#119DFF'),
font = dict(color = 'white', size = 14),align = ['center'],height = 30),
cells=dict(values=[top_10['State/UnionTerritory'].head(10),top_10['Confirmed'].head(
10)],
 fill = dict(color = ['#25FEFD', 'white']),align = ['center'],height=20))
trace1 =
go.Bar(x=top_10['State/UnionTerritory'].head(10),y=top_10['Confirmed'].head(10),
marker=dict(color='blue'),opacity=0.60)
layout = dict(width=1000,height=430,title='Top-10 States with Highest No. of Cases',
xaxis1=dict(**dict(domain=[0.6, 1.0])),)fig1 = dict(data=[trace, trace1],
layout=layout)
iplot(fig1)
state_wise_cases=state_cases.sort_values('Confirmed', ascending=
False).fillna(0).style\
.background_gradient(cmap='Blues',subset=["Confirmed"])\.background_gradient(cm
ap='Reds',subset=["Deaths"])\.background_gradient(cmap='Greens',subset=["Cured"]
)\.background_gradient(cmap='Oranges',subset=["Active"])\.background_gradient(cm
ap='RdYlBu',subset=["Death Rate (per 100)"])\
.background_gradient(cmap='Accent',subset=["Cure Rate (per 100)"])
state_wise_cases
df_testing=pd.read_csv("/content/drive/MyDrive/Covid-19-Project/Major-Project
Datasets/StatewiseTestingDetails.csv")
from google.colab import drive
df_testing.head()
df_sample=df_testing.groupby('State')['TotalSamples'].sum().sort_values(ascending=
False).reset_index()
df_sample.head()
px.bar(df_sample.head(10),x='State',y='TotalSamples',width=700,
height=500,title='Total Samples Collected in Top 10 States')
```

```
df_sample_positive=df_testing.groupby('State')['Positive'].sum().sort_values(ascendin
g=False).reset_index()
df_sample_positive.head()
px.bar(df_sample_positive.head(10),x='State',y='Positive',width=700,
height=500,title='Total Positive Cases in Top 10 States')
df_vaccine=pd.read_csv('/content/drive/MyDrive/Covid-19-Project/Major-Project-
Datasets/covid_vaccine_statewise.csv')
df_vaccine.head()
df_vaccine.columns
male = df_vaccine["Male(Individuals Vaccinated)"].sum()
female = df_vaccine["Female(Individuals Vaccinated)"].sum()
trans = df_vaccine["Transgender(Individuals Vaccinated)"].sum()
px.pie(names=["Male Vaccinated","Female Vaccinated","Trans
Gender"],values=[male,female,trans],title="Vaccinated ratio for Covid19")
Covaxin = df_vaccine["Total Covaxin Administered"].sum()
Covishield = df_vaccine["Total CoviShield Administered"].sum()
px.pie(names=["Covaxin Vaccinated","Covishield
Vaccinated"],values=[Covaxin,Covishield],title="Covaxin and Covishield
Vaccination")
Doses = df_vaccine["Total Doses Administered"].sum()
Vaccinated = df_vaccine["Total Individuals Vaccinated"].sum()
px.pie(names=["Doses Administered","People
Vaccinated"],values=[Doses,Vaccinated],title="Doses administered vs People
Vaccinated")
df_vaccine['State'].unique()
df_Telangana=df_vaccine[df_vaccine['State']=='Telangana']
df_Maharashtra=df_vaccine[df_vaccine['State']=='Maharashtra']
df_Andhra_Pradesh=df_vaccine[df_vaccine['State']=='Andhra Pradesh']
Doses = df_Telangana["Total Doses Administered"].sum()
Vaccinated = df_Telangana["Total Individuals Vaccinated"].sum()
px.pie(names=["Doses Administered","People
Vaccinated"],values=[Doses,Vaccinated],title="Doses administered vs People
Vaccinated in Telangana")
Doses = df_Maharashtra["Total Doses Administered"].sum()
```

```
Vaccinated = df_Maharashtra["Total Individuals Vaccinated"].sum()
px.pie(names=["Doses Administered","People
Vaccinated"],values=[Doses,Vaccinated],title="Doses administered vs People
Vaccinated in Maharastra")
Doses = df_Andhra_Pradesh["Total Doses Administered"].sum()
Vaccinated = df_Andhra_Pradesh["Total Individuals Vaccinated"].sum()
px.pie(names=["Doses Administered","People
Vaccinated"],values=[Doses,Vaccinated],title="Doses administered vs People
Vaccinated in Andhra Pradesh")
df = df.rename(columns = {'Date': 'ds'})
df_group = df.groupby(by = 'ds')['Confirmed', 'Deaths', 'Cured'].sum()
df_group.index = pd.to_datetime(df_group.index)
df_group = df_group.asfreq(freq = '1D')
df_group = df_group.sort_index(ascending = True)
df_group = df_group.fillna(value = 0)
df_group = df_group.rename(columns = {'Date': 'ds'})
display(df_group.tail())
#display(df_group.head())
"""## FB Prophet"""
df_prophet = df_group[['Confirmed']]
df_prophet = df_prophet.reset_index()
df_prophet = df_prophet.rename(columns = {'ds': 'ds', 'Confirmed': 'y'})
df_prophet['ds'] = pd.to_datetime(df_prophet['ds'])
m = Prophet(growth='linear', n_changepoints=30,
changepoint_range=0.85)m.fit(df_prophet)
future = m.make_future_dataframe(periods = 20)
forecast = m.predict(future)
fbprophet_score=np.sqrt(mean_squared_error(df_group["Confirmed"],forecast['yhat'].
head(df_group.shape[0])))
fbprophet_rmse_score=np.sqrt(mse(df_group["Confirmed"],forecast['yhat'].head(df_g
roup.shape[0])))/valid_len
rmse_scores.append(["FB Prophet RMSE Score:",fbprophet_rmse_score])
print("Root Mean Square Error for FB Prophet Model: ",fbprophet_rmse_score)
print(rmse_scores)
```

```
figure = m.plot(forecast, xlabel = 'Date', ylabel = 'Confirmed Cases')
figure2 = m.plot_components(forecast)
#Facebook Prophet Predictions for next 10 days
model_predictions["Prophet's Prediction"]=list(forecast["yhat"].tail(10))
model_predictions.head()
```

Vaccine_Analysis.py

```
#importing all the required libraries
!pip install geopandas
import pandas as pd
import numpy as np
import plotly.graph_objects as go
from matplotlib import pyplot as plt
import warnings
import plotly.express as px
import seaborn as sns
import geopandas
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot
warnings.filterwarnings('ignore')
sns.set(rc={'figure.figsize':(20,20)})
pip install --upgrade plotly
#Mounting the drive
from google.colab import drive
drive.mount("/content/drive")
#Loading the dataset
df =pd.read_csv("/content/drive/MyDrive/Dataset/vaccine/country_vaccinations.csv")
new_df = df.groupby(["country",'iso_code','vaccines'])['total_vaccinations',
'people_vaccinated',
'people_fully_vaccinated','daily_vaccinations','total_vaccinations_per_hundred','peopl
e_vaccinated_per_hundred',’people_fully_vaccinated_per_hundred’,'daily_vaccinatio
ns_per_million'].max().reset_index()
##Analysis and Visualisation
##Total Vaccinations
data = (new_df[['country','total_vaccinations']].nlargest(20,'total_vaccinations'))
```

```
px.bar(data, x = 'country',y = 'total_vaccinations',labels={'country':'Country',
'total_vaccinations' :"Total Vaccinations"},title="Number of total vaccinations
according to countries")
df['date'] = pd.to_datetime(df['date'])
df = df.sort_values('date', ascending=True)
df['date'] = df['date'].dt.strftime('%m-%d-%Y')
tdf = df.copy()
for iso_code in tdf['iso_code'].unique():
    tdf.loc[tdf['iso_code'] == iso_code, :] = tdf.loc[tdf['iso_code'] == iso_code,
:].fillna(method='ffill').fillna(0)
fig = px.choropleth(
    tdf, locations="iso_code",color="total_vaccinations",  hover_name="country",
animation_frame="date",orders={"frame": list(sorted(tdf['date'].unique()))},
color_continuous_scale= 'viridis',projection="natural earth",
range_color=[0,3000000],
title='<span style="font-size:36px; font-family:Times New Roman">Number of total
vaccinations per country</span>',)
fig.show()
fig = px.treemap(new_df,names = 'country',values = 'total_vaccinations',path =
['vaccines','country'],title="Total vaccinations per country grouped by vaccines",
   color_discrete_sequence =px.colors.qualitative.Set1)
fig.show()
trace = go.Choropleth(locations = new_df['country'],locationmode='country names',z
= new_df['total_vaccinations'],text =new_df['country'],autocolorscale
=False,reversescale = True,
colorscale = 'viridis', colorbar = dict(title = 'Total vaccinations'))
data = [trace]
layout = go.Layout(title = 'Total vaccinations per country',geo = dict(projection =
dict(           type = 'natural earth' )))
fig = dict( data=data, layout=layout )
iplot(fig)
##People Vaccinated
data = new_df[['country','people_vaccinated']].nlargest(20,'people_vaccinated')
```

```
fig = px.bar(data, x = 'country',y = 'people_vaccinated',labels={'country': 'Country',
'people_vaccinated':"People Vaccinated"},title="Number of people vaccinated
according to countries",)
fig.show()
fig = px.choropleth(tdf,locations="iso_code", color="people_vaccinated",
hover_name="country", animation_frame="date",orders={"frame":
list(sorted(tdf['date'].unique()))},color_continuous_scale= 'viridis',projection="natural
earth”,     range_color=[0,5000000],title='<span style="font-size:36px; font-
family:Times New Roman">Number of people vaccinated per country</span>')
fig.show()
fig = px.treemap(new_df,names = 'country',values = 'people_vaccinated',path =
['vaccines','country'],title="People vaccinated per country grouped by vaccines",
color_discrete_sequence =px.colors.qualitative.Set1)
fig.show()
trace = go.Choropleth(locations = new_df['country'], locationmode='country names',z
= new_df['people_vaccinated'],text =new_df['country'],autocolorscale
=False,reversescale = True,
colorscale = 'viridis',colorbar = dict(title=’people_vaccinated”))
data = [trace]
layout = go.Layout(title = 'People vaccinated per country',geo = dict(projection =
dict(type = 'natural earth' )))
fig = dict( data=data, layout=layout)
iplot(fig)
##People Fully Vaccinated
Data=new_df[['country','people_fully_vaccinated']].nlargest(20,'people_fully_vaccina
ted')
fig = px.bar(data, x = 'country',y = 'people_fully_vaccinated',labels={'country':
'Country', 'people_fully_vaccinated':"People Fully Vaccinated"},title="Number of
people fully vaccinated according to countries",)
fig.show()
fig = px.choropleth(tdf,locations="iso_code",color="people_fully_vaccinated",
hover_name="country",animation_frame="date",orders={"frame":
list(sorted(tdf['date'].unique()))},color_continuous_scale= 'viridis',projection="natural
earth",range_color=[0,5000000],title='<span style="font-size:36px; font-family:Times
New Roman">Number of people fully vaccinated per country</span>')
fig.show()
fig = px.treemap(new_df,names = 'country',values = 'people_fully_vaccinated',path =
['vaccines','country'],title="People fully vaccinated per country grouped by vaccines",
color_discrete_sequence =px.colors.qualitative.Set1)
fig.show()
trace = go.Choropleth(locations = new_df['country'],locationmode='country names',z
= new_df['people_fully_vaccinated'],text =new_df['country'],autocolorscale
=False,reversescale = True,colorscale = 'viridis',colorbar = dict(title = 'People fully
vaccinated'))
data = [trace]
layout = go.Layout(title = 'People fully vaccinated per country',geo = dict(projection
= dict(type ='natural earth')))
fig = dict( data=data, layout=layout )
iplot(fig)
##Daily Vaccinations
#Representing the top 20 countries having highest daily vaccinations
data = new_df[['country','daily_vaccinations']].nlargest(20,'daily_vaccinations')
fig = px.bar(data, x = 'country',y = 'daily_vaccinations',labels={'country':
'Country','daily_vaccinations': "Daily Vaccinations"},title="Number of daily
vaccinations according to countries”)
fig.show()
fig = px.choropleth(tdf,locations="iso_code", color="daily_vaccinations",
hover_name = “country”,animation_frame="date",orders={"frame":
list(sorted(tdf['date'].unique()))},     color_continuous_scale=
'viridis',projection="natural earth", range_color=[0,1000000],
title='<span style="font-size:36px; font-family:Times New Roman">Number of daily
vaccinations per country</span>')
fig.show()
fig=px.treemap(new_df,names='country',values='daily_vaccinations',path=['vaccines','
country'] ,title="Daily vaccinations per country grouped by
vaccines",color_discrete_sequence  =px.colors.qualitative.Set1)
fig.show()
```

```
trace = go.Choropleth(locations = new_df['country'],locationmode='country names',z
= new_df['daily_vaccinations'],text =new_df['country'],autocolorscale
=False,reversescale = True,
colorscale = 'viridis',colorbar = dict(title = 'Daily vaccinations'))
data = [trace]
layout = go.Layout(title = 'Daily vaccinations per country', geo = dict(projection =
dict(type = 'natural earth')))
fig = dict( data=data, layout=layout )
iplot(fig)
##Density-wise Analysis and Visualisation
##Total Vaccinations Per Hundred
#Representing the top 20 countries having highest people vaccinated
data = new_df[['country','total_vaccinations_per_hundred']].nlargest(20,
'total_vaccinations_per_hundred')
fig = px.bar(data, x = 'country',y = 'total_vaccinations_per_hundred',labels={'country':
'Country','total_vaccinations_per_hundred':"Total vaccinations per
hundred"},title="Number of total vaccinations per hundred according to countries")
fig.show()
fig = px.choropleth(tdf,
locations="iso_code",color="total_vaccinations_per_hundred",
hover_name="country",animation_frame="date",orders={"frame": list(sorted
(tdf['date']. unique()))},color_continuous_scale= 'viridis',projection="natural
earth",range_color=[0,100],
title='<span style="font-size:36px; font-family:Times New Roman">Number of total
vaccinations per hundred per country</span>')
fig.show()
fig = px.treemap(new_df,names = 'country',values =
'total_vaccinations_per_hundred',path = ['vaccines','country'],title="Total vaccinations
per hundred per country grouped by vaccines",
color_discrete_sequence =px.colors.qualitative.Set1)
fig.show()
trace = go.Choropleth(locations = new_df['country'],locationmode='country names',z
= new_df['total_vaccinations_per_hundred'],text =new_df['country'],reversescale =
True,colorscale = 'viridis',colorbar = dict(title = 'Total vaccinations per hundred'))
```

```
data = [trace]
layout = go.Layout(title = 'Total vaccinations per hundred per country',geo =
dict(projection = dict(type = 'natural earth')))
fig = dict( data=data, layout=layout )
iplot(fig)
##People Vaccinated Per Hundred
#Representing the top 20 countries having highest people vaccinated
data =new_df[['country','people_vaccinated_per_hundred']].nlargest(20,
'people_vaccinated_per_hundred')
fig = px.bar(data, x = 'country',y =
'people_vaccinated_per_hundred',labels={'country':
'Country','people_vaccinated_per_hundred':"People vaccinated per
hundred"},title="Number of people vaccinated per hundred according to countries")
fig.show()
fig = px.choropleth(tdf, locations="iso_code",color="
people_vaccinated_per_hundred",
hover_name="country",animation_frame="date",orders={"frame":
list(sorted(tdf['date'].unique()))},color_continuous_scale= 'viridis',projection="natural
earth",range_color=[0,70],title='<span style="font-size:36px; font-family:Times New
Roman"> Number of people vaccinated per hundred per country</span>')
fig.show()
fig = px.treemap(new_df,names = 'country',values =
'people_vaccinated_per_hundred',path = ['vaccines','country'],title="People vaccinated
per hundred per country grouped by vaccines",
color_discrete_sequence =px.colors.qualitative.Set1)
fig.show()
trace = go.Choropleth(locations = new_df['country'],locationmode='country names',z
= new_df['people_vaccinated_per_hundred'],text =new_df['country'],autocolorscale
=False,
reversescale = True,colorscale = 'viridis',colorbar = dict(title = 'People vaccinated per
hundred'))
 data = [trace]
layout = go.Layout(title = 'People vaccinated per hundred per country',geo =
dict(projection = dict(type = 'natural earth')))
```

```
fig = dict( data=data, layout=layout )
iplot(fig)
##People Fully Vaccinated Per Hundred
#Representing the top 20 countries having highest people vaccinated
data = new_df[['country','people_fully_vaccinated_per_hundred']].nlargest(20,
'people_fully_vaccinated_per_hundred')
fig = px.bar(data, x = 'country',y =
'people_fully_vaccinated_per_hundred',labels={'country':
'Country','people_fully_vaccinated_per_hundred':"People fully vaccinated per
hundred"}, title="Number of people fully vaccinated per hundred per country
according to countries",)
fig.show()
fig=px.choropleth(tdf,locations="iso_code",color="people_fully_vaccinated_per_hun
dred",
hover_name="country",animation_frame="date",orders={"frame":
list(sorted(tdf['date'].unique() ))},color_continuous_scale=
'viridis',projection="natural earth",range_color=[0,50],title='<span style="font-
size:36px; font-family:Times New Roman">Number of people fully vaccinated per
hundred per country</span>')
fig.show()
fig = px.treemap(new_df,names = 'country',values =
'people_fully_vaccinated_per_hundred',path = ['vaccines','country'],title="People fully
vaccinated per hundred per country grouped by vaccines",color_discrete_sequence
=px.colors.qualitative.Set1)
fig.show()
trace = go.Choropleth(locations = new_df['country'],locationmode='country names',z
= new_df['people_fully_vaccinated_per_hundred'],text
=new_df['country'],reversescale = True,
colorscale = 'viridis',colorbar = dict(title = 'People fully vaccinated per hundred'))
data = [trace]
layout = go.Layout(title = 'People fully vaccinated per hundred per country',geo =
dict(projection = dict(type = 'natural earth')))
fig = dict( data=data, layout=layout )
iplot(fig)
```

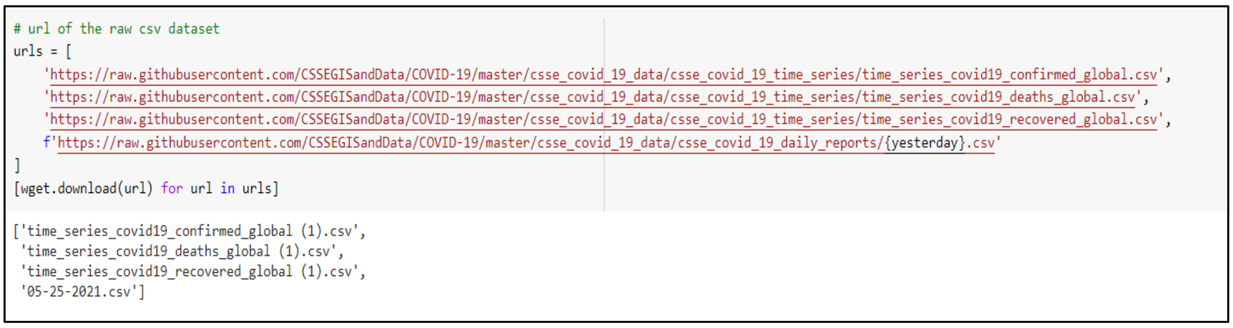
```
##Daily Vaccinations Per Million
#Representing the top 20 countries having highest people vaccinated
data = new_df[['country','daily_vaccinations_per_million']].nlargest(20,
'daily_vaccinations_per_million')
fig = px.bar(data, x = 'country',y = 'daily_vaccinations_per_million',labels={'country':
'Country','
daily_vaccinations_per_million':"Daily vaccinations per million"},title="Number of
daily vaccinations per million according to countries",)
fig.show()
fig = px.choropleth(tdf,locations="iso_code",color="daily_vaccinations_per_million",
hover_name="country",animation_frame="date",orders={"frame":
list(sorted(tdf['date'].unique() ))},color_continuous_scale=
'viridis',projection="natural earth", range_color=[0,10000],   title='< span style=
"font-size:36px; font-family:Times New Roman">Number of daily vaccinations per
million per country</span>')
fig.show()
fig = px.treemap(new_df,names = 'country',values =
'daily_vaccinations_per_million',path = ['vaccines','country'],title="Daily vaccinations
per million per country grouped by vaccines",
color_discrete_sequence =px.colors.qualitative.Set1)
fig.show()
trace = go.Choropleth(locations = new_df['country'],locationmode='country names',z
= new_df['daily_vaccinations_per_million'],text =new_df['country'],autocolorscale
=False,
reversescale = True,colorscale = 'viridis',colorbar = dict(title = 'Daily vaccinations per
million'))
data = [trace]
layout = go.Layout(title = 'Daily vaccinations per million per country',geo =
dict(projection = dict(type = 'natural earth')))
fig = dict( data=data, layout=layout )
iplot(fig)
```

```
##Data Preprocessing For Future Forecastingdf = df[['date', 'country','iso_code',
'daily_vaccinations','people_vaccinated_per_hundred','people_fully_vaccinated_per_h
undred','vaccines',]]
df.head()
print(df.isnull().sum())
#Loading continents dataset
continents = pd.read_csv('/content/drive/MyDrive/Dataset/vaccine/continents2.csv')
continents.head()
#Merging the two datasets
data = df.merge(continents[['alpha-3', 'region', 'sub-region']],how='left',left_on =
'iso_code',right_on = 'alpha-3',).drop(columns=['alpha-3'])
data.head()
data.columns = ['date', 'country', 'iso_code', 'daily_vaccinations',
'people_vaccinated_per_hundred', 'people_fully_vaccinated_per_hundred','vaccines',
'continent', 'region']
data.loc[data.continent == 'Oceania', 'continent'] = 'Australia'
data.continent.value_counts()
data.isnull().sum()
data.vaccines.value_counts()
df = data[['iso_code', 'vaccines']].drop_duplicates()
df.shape
map_plot_df = pd.concat([pd.Series(row['iso_code'], row['vaccines'].split(',')) for _,
row in df.iterrows()]).reset_index()
#rename columns
map_plot_df.columns = ['vaccine', 'iso_code']
map_plot_df['vaccine'] = map_plot_df['vaccine'].str.strip(' '
map_plot_df.vaccine.value_counts()
##Future Forecasting Of Vaccination Percentage Using FBProphet
from fbprophet import Prophet
def fbprophet_predict_and_plot(data, region_column, region, vaccination_metric,
min_date = '2020-12-13', max_date='2021-5-17', future_days=10, plot=True):
    df_data = data[(data[region_column] == region)]
    df_data = df_data[['date', vaccination_metric]]
    df_data.columns = ['ds', 'y']
    model = Prophet(interval_width = 0.90,changepoints=None, n_changepoints=10,
changepoint_range=0.8,)#yearly_seasonality=True, daily_seasonality=True,)
    model.fit(df_data)
    future = model.make_future_dataframe(periods=future_days)
    forecast = model.predict(future)
    if plot:
        fig = plot_plotly(model, forecast)
        fig.show()
    else:
        return forecast
countries = ['India','China','United Kingdom','United States','Russia','Spain']
print(countries)
vaccination_summary = []
reqdate = '2021-07-01'
for country in countries:
    #if there are no people_vaccinated_per_hundred values for country, skip country
    if data[(data['people_fully_vaccinated_per_hundred'].notnull()) & (data['country']
== country)].shape[0] > 2:
        min_date = data[(data['country'] == country)].date.min()
        forecast = fbprophet_predict_and_plot(data = data,region_column =
'country',region = country,vaccination_metric =
'people_fully_vaccinated_per_hundred',
                                             min_date = min_date,future_days = 50,plot = False,)
        vaccination_summary.append([country,round(float(forecast.loc[forecast.ds ==
reqdate].yhat),2) ])
print('Predicted percentage of people fully vaccinated until',reqdate) print("Country",
"%",)
vs_df = pd.DataFrame(vaccination_summary,columns = ("Country","Percentage of
people fully vaccinated"))
vs_df.head()
```

4.2 TEST CASES

4.2.1 INPUT SCREENSHOTS

4.2.1.1 World Analysis and Prediction

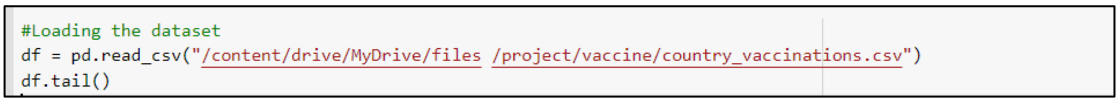


```
# url of the raw csv dataset
urls = [
    'https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_time_series/time_series_covid19_confirmed_global.csv',
    'https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_time_series/time_series_covid19_deaths_global.csv',
    'https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_time_series/time_series_covid19_recovered_global.csv',
    f'https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_daily_reports/{yesterday}.csv'
]
[wget.download(url) for url in urls]
```

```
['time_series_covid19_confirmed_global (1).csv',
 'time_series_covid19_deaths_global (1).csv',
 'time_series_covid19_recovered_global (1).csv',
 '05-25-2021.csv']
```

Fig 4.2.1.1: Loading Global Cases Dataset

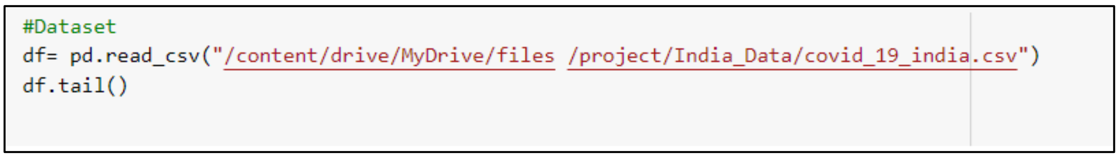
4.2.1.2 Vaccination Analysis



```
#Loading the dataset
df = pd.read_csv("/content/drive/MyDrive/files /project/vaccine/country_vaccinations.csv")
df.tail()
```

Fig 4.2.1.2: Loading Vaccination Dataset

4.2.1.3 India Analysis and Prediction

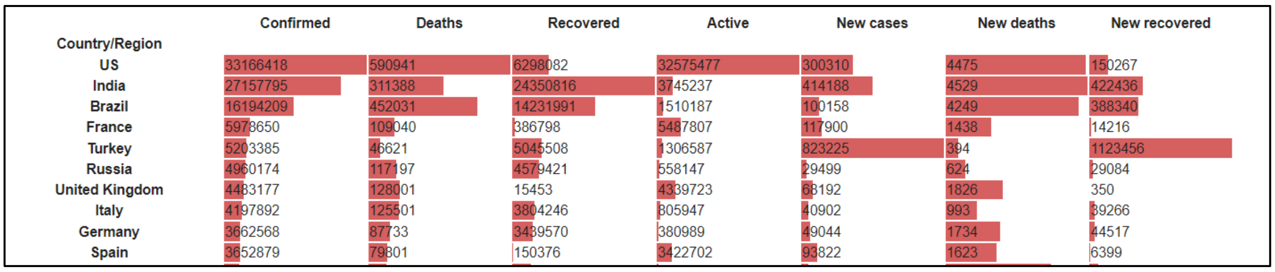


```
#Dataset
df= pd.read_csv("/content/drive/MyDrive/files /project/India_Data/covid_19_india.csv")
df.tail()
```

Fig 4.2.1.3: Loading State-wise India Dataset

4.2.2 OUTPUT SCREENSHOTS

4.2.2.1 World Analysis and Prediction



Country/Region	Confirmed	Deaths	Recovered	Active	New cases	New deaths	New recovered
US	33166418	590941	6298082	32575477	300310	4475	150267
India	27157795	311388	24350816	3745237	414188	4529	422436
Brazil	16194209	452031	14231991	1510187	100158	4249	388340
France	5978650	109040	386798	5487807	117900	1438	14216
Turkey	5203385	46621	5045508	1306587	823225	394	1123456
Russia	4960174	117197	4579421	558147	29499	624	29084
United Kingdom	4483177	128001	15453	4339723	68192	1826	350
Italy	4197892	125501	3804246	805947	40902	993	39266
Germany	3662568	87733	3439570	380989	49044	1734	44517
Spain	3652879	79801	150376	3422702	93822	1623	6399

Fig 4.2.2.1.1: Country Wise Cases

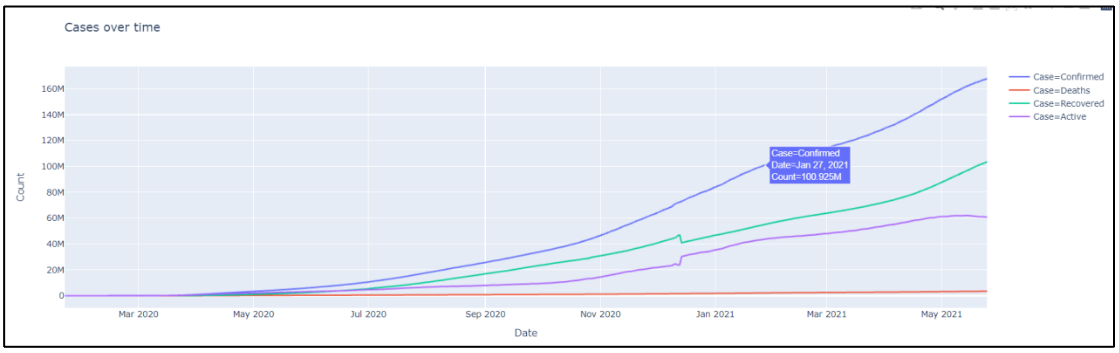


Fig 4.2.2.1.2: World Cases Progress over Time

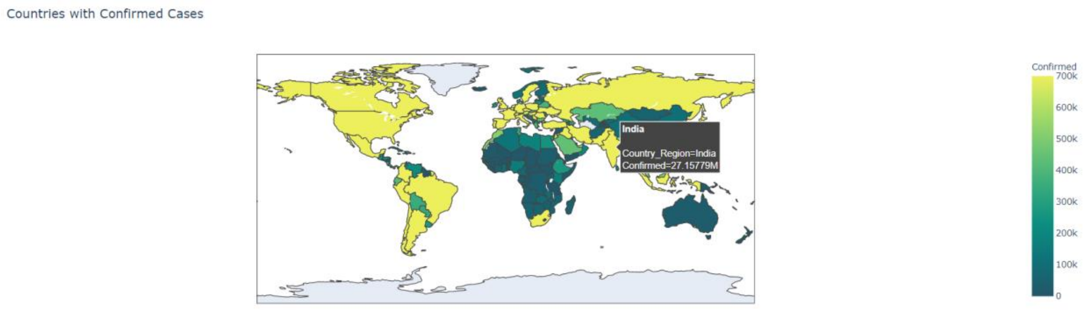
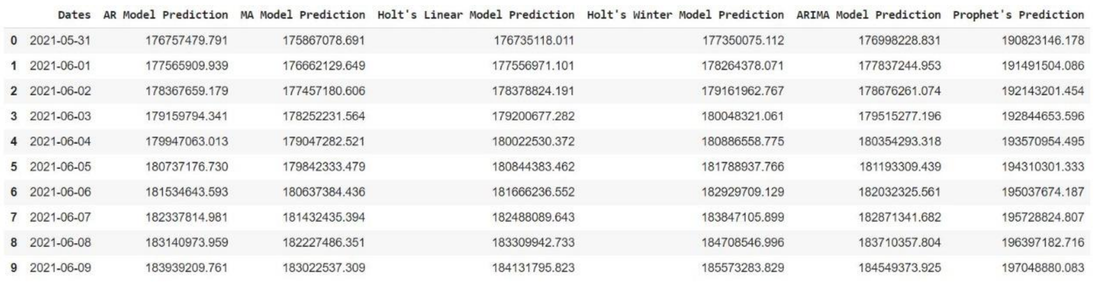
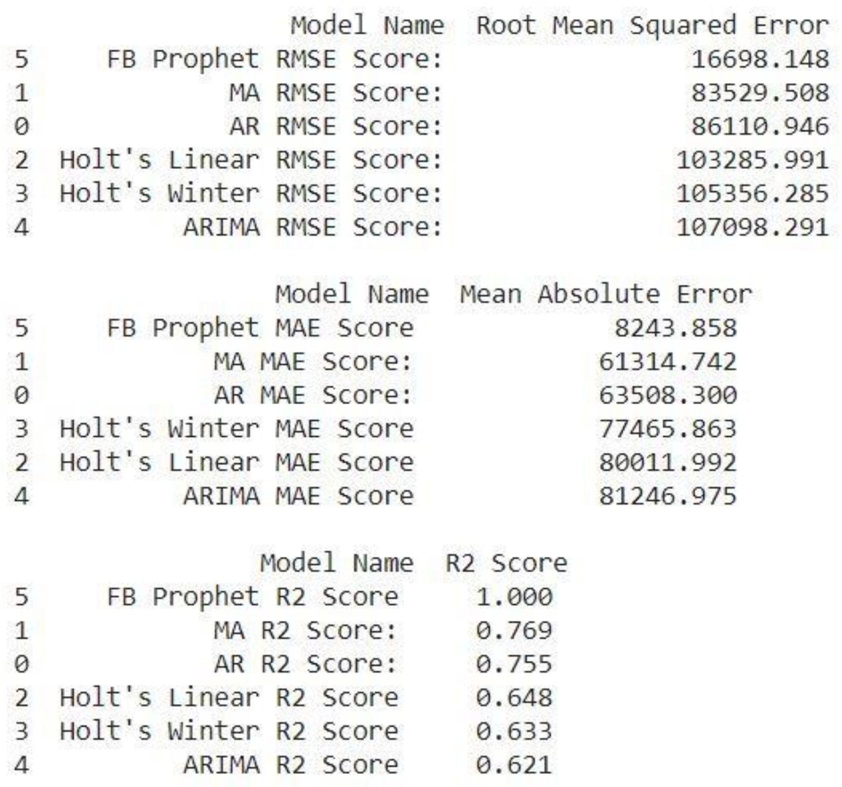


Fig 4.2.2.1.3: Country Wise Representation of Confirmed Cases



	Dates	AR Model Prediction	MA Model Prediction	Holt's Linear Model Prediction	Holt's Winter Model Prediction	ARIMA Model Prediction	Prophet's Prediction
0	2021-05-31	176757479.791	175867078.691	176735118.011	177350075.112	176998228.831	190823146.178
1	2021-06-01	177565909.939	176662129.649	177556971.101	178264378.071	177837244.953	191491504.086
2	2021-06-02	178367659.179	177457180.606	178378824.191	179161962.767	178676261.074	192143201.454
3	2021-06-03	179159794.341	178252231.564	179200677.282	180048321.061	179515277.196	192844653.596
4	2021-06-04	179947063.013	179047282.521	180022530.372	180886558.775	180354293.318	193570954.495
5	2021-06-05	180737176.730	179842333.479	180844383.462	181788937.766	181193309.439	194310301.333
6	2021-06-06	181534643.593	180637384.436	181666236.552	182929709.129	182032325.561	195037674.187
7	2021-06-07	182337814.981	181432435.394	182488089.643	183847105.899	182871341.682	195728824.807
8	2021-06-08	183140973.959	182227486.351	183309942.733	184708546.996	183710357.804	196397182.716
9	2021-06-09	183939209.761	183022537.309	184131795.823	185573283.829	184549373.925	197048880.083

Fig 4.2.2.1.4: World Wide Cases Predictions



```
                 Model Name  Root Mean Squared Error
5     FB Prophet RMSE Score:                16698.148
1             MA RMSE Score:                83529.508
0             AR RMSE Score:                86110.946
2  Holt's Linear RMSE Score:               103285.991
3  Holt's Winter RMSE Score:               105356.285
4          ARIMA RMSE Score:               107098.291

                Model Name  Mean Absolute Error
5     FB Prophet MAE Score            8243.858
1            MA MAE Score:           61314.742
0            AR MAE Score:           63508.300
3  Holt's Winter MAE Score           77465.863
2  Holt's Linear MAE Score           80011.992
4          ARIMA MAE Score           81246.975

               Model Name  R2 Score
5     FB Prophet R2 Score     1.000
1            MA R2 Score:     0.769
0            AR R2 Score:     0.755
2  Holt's Linear R2 Score     0.648
3  Holt's Winter R2 Score     0.633
4          ARIMA R2 Score     0.621
```

Fig 4.2.2.1.5: Accuracy Metrics

4.2.2.2 Vaccination Analysis

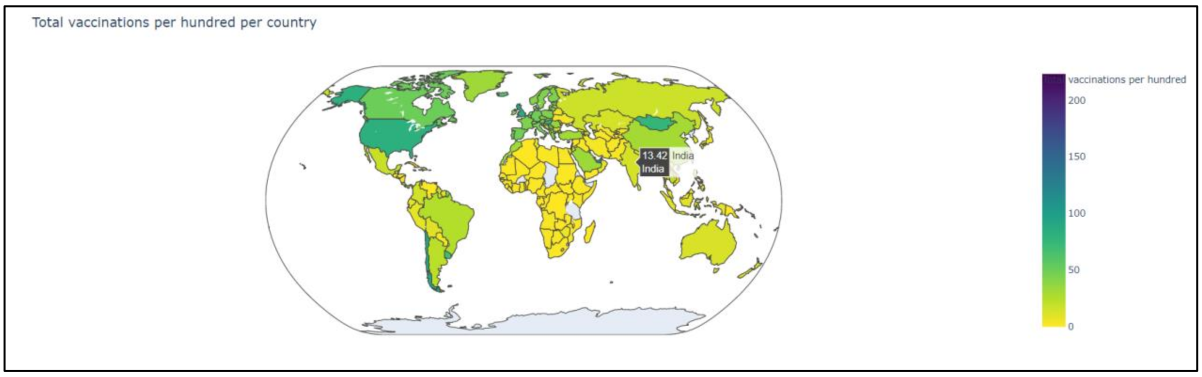


Fig 4.2.2.2.1: Total Vaccinations per Hundred

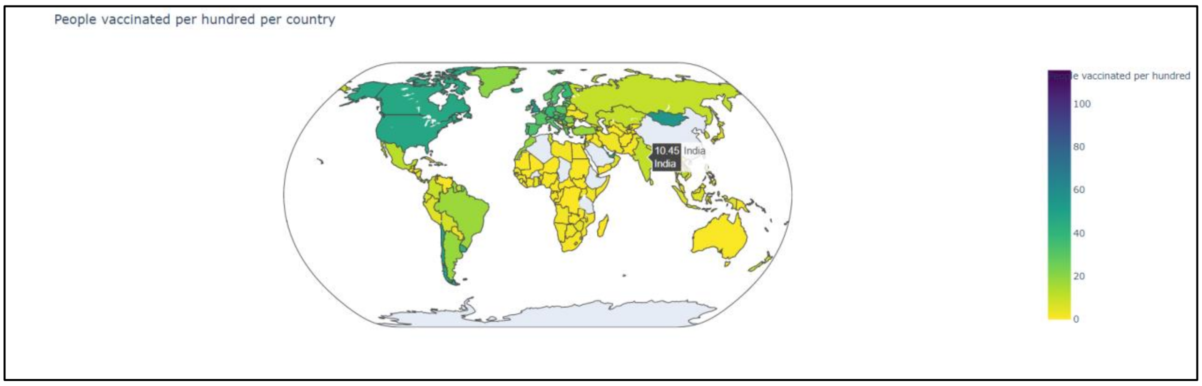


Fig 4.2.2.2.2: People Vaccinated per Hundred

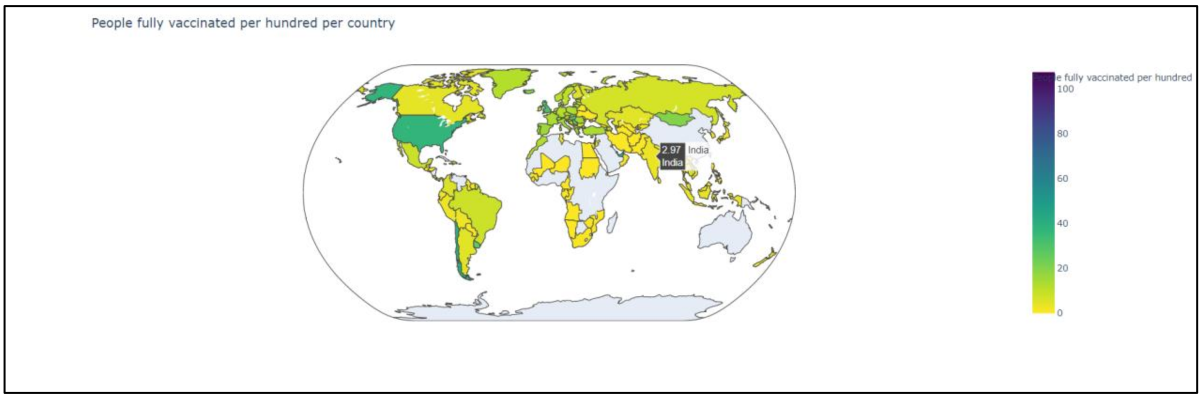
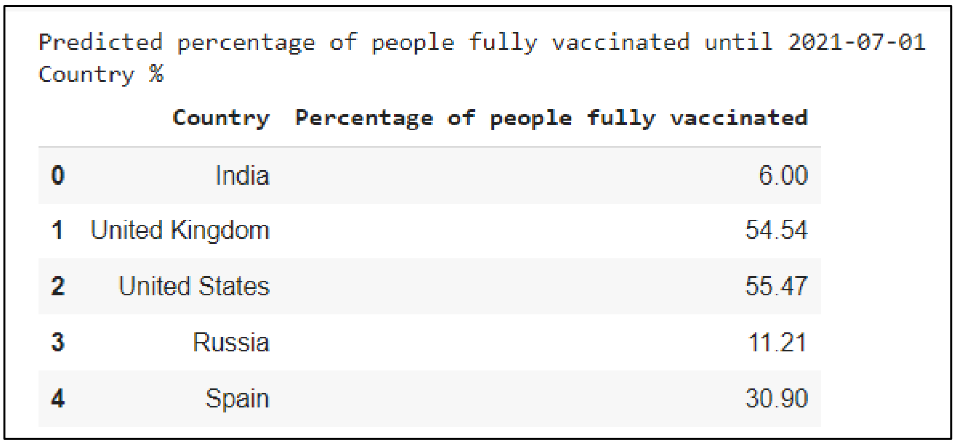

People fully vaccinated per hundred per country
e fully vaccinated per hundred
100
80
60
40
20
0
2.97 India
India

Fig 4.2.2.2.3: People Fully Vaccinated per Hundred



```
Predicted percentage of people fully vaccinated until 2021-07-01
Country %
```

	Country	Percentage of people fully vaccinated
0	India	6.00
1	United Kingdom	54.54
2	United States	55.47
3	Russia	11.21
4	Spain	30.90

Fig 4.2.2.2.4: Prediction of Vaccination Percentage

4.2.2.3 India Analysis and Prediction

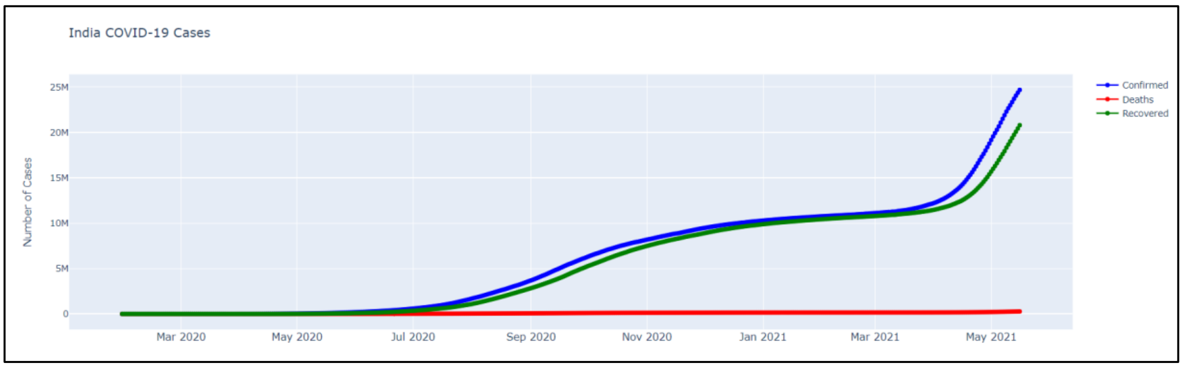
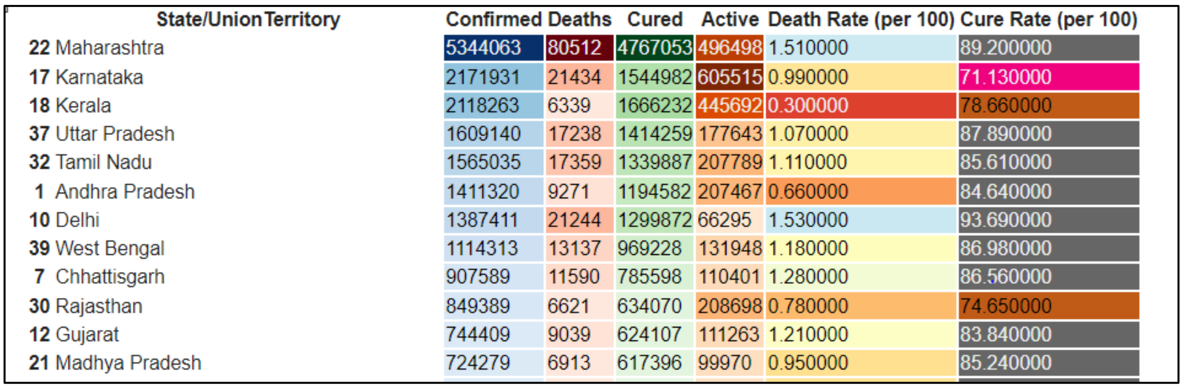


Fig 4.2.2.3.1: Progress of Cases over Time in India



	State/Union Territory	Confirmed	Deaths	Cured	Active	Death Rate (per 100)	Cure Rate (per 100)
22	Maharashtra	5344063	80512	4767053	496498	1.510000	89.200000
17	Karnataka	2171931	21434	1544982	605515	0.990000	71.130000
18	Kerala	2118263	6339	1666232	445692	0.300000	78.660000
37	Uttar Pradesh	1609140	17238	1414259	177643	1.070000	87.890000
32	Tamil Nadu	1565035	17359	1339887	207789	1.110000	85.610000
1	Andhra Pradesh	1411320	9271	1194582	207467	0.660000	84.640000
10	Delhi	1387411	21244	1299872	66295	1.530000	93.690000
39	West Bengal	1114313	13137	969228	131948	1.180000	86.980000
7	Chhattisgarh	907589	11590	785598	110401	1.280000	86.560000
30	Rajasthan	849389	6621	634070	208698	0.780000	74.650000
12	Gujarat	744409	9039	624107	111263	1.210000	83.840000
21	Madhya Pradesh	724279	6913	617396	99970	0.950000	85.240000

Fig 4.2.2.3.2: State Wise Cases in India

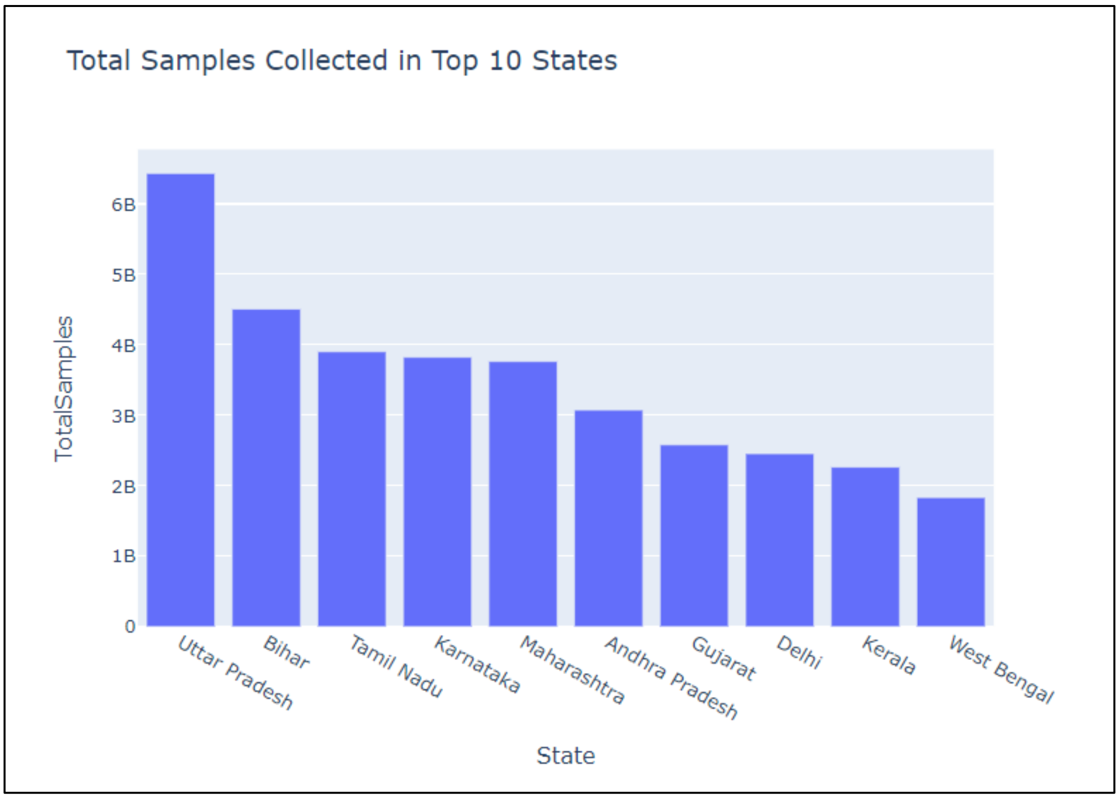
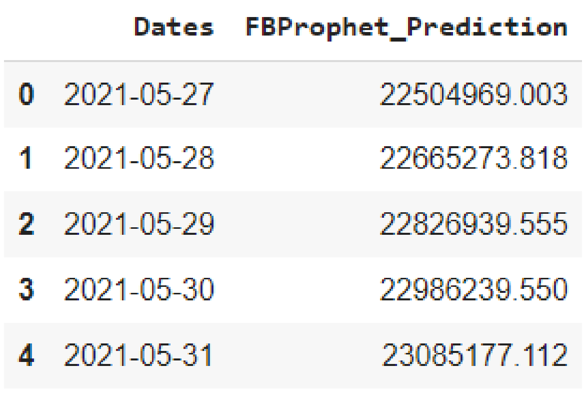


Fig 4.2.2.3.3: Total Samples Collected in Top 10 States



	Dates	FBProphet_Prediction
0	2021-05-27	22504969.003
1	2021-05-28	22665273.818
2	2021-05-29	22826939.555
3	2021-05-30	22986239.550
4	2021-05-31	23085177.112

Fig 4.2.2.3.4: Cases Predictions in India

5. CONCLUSION AND FUTURE SCOPE

The first case of the coronavirus was found in the Wuhan city of China in the seafood market. Slowly this virus spread to more than 200 countries. The findings of this project work are helpful for future prediction of the confirmed cases of the COVID-19 and vaccination percentages for different countries which is helpful for the government entities and other healthcare organizations to mitigate the problem of the outbreak. In this work, worldwide data is used and tested for six most famous models to forecast the future scenarios. The dataset may be increasing with time. The codes developed in this work just need a re-run in the future to see the future value of the cases anytime. FB Prophet and Holt's Winter Seasonal Trend models have better performance compared to other models on the scale of MAE, RMSE and R2 Score error metrics. The trend analysis shows rapid growth in the infected cases and prediction study shows great rise in the expected active, recovered and death cases worldwide. In future, as the data points will be very high for COVIsD-19 dataset, any other model may also come up with the great insights.

The future enhancements of this project are:

- To create a User Interface for efficient visualizations and Mobile Application to view the result
- To create a User Interface for efficient visualizations
- To improve the accuracy of the models
- To generate a model that correlates the vaccination and confirmed cases progress over time
- To predict the duration of COVID-19 and time taken to vaccinate the total population of different countries.

6. REFERENCES

[1]. S. Maurya and S. Singh, "Time Series Analysis of the Covid-19 Datasets," 2020 IEEE International Conference for Innovation in Technology (INOCON), Bangluru, India, 2020

[2]. S. Shaikh, J. Gala, A. Jain, S. Advani, S. Jaidhara and M. Roja Edinburgh, "Analysis and Prediction of COVID-19 using Regression Models and Time Series Forecasting," 2021 11th International Conference on Cloud Computing, Data Science & Engineering (Confluence), Noida, India, 2021

[3] O. Sarkar, M. F. Ahamed and P. Chowdhury, "Forecasting & Severity Analysis of COVID-19 Using Machine Learning Approach with Advanced Data Visualization," 2020 23rd International Conference on Computer and Information Technology (ICCIT), DHAKA, Bangladesh, 2020

[4]. G. R. Shinde, A.B. Kalamkar, P.N. Mahalle, N. Dey, J. Chaki and A.E. Hassanien, "Forecasting Models for Coronavirus Disease (COVID-19): A Survey of the State-of-the-Art", SN Computer Science, vol. 1, no. 4, pp. 1-15, 2020.

[5]. N. Darapaneni, P. Jain, R. Khattar, M. Chawla, R. Vaish and A. R. Paduri, "Analysis and Prediction of COVID-19 Pandemic in India," 2020 2nd International Conference on Advances in Computing, Communication Control and Networking