FORECASTING COVID-19 USING ARIMA MODEL

Project
For
SWE2009 Data Mining Techniques

Submitted by

Ashish M (20MIS0050)

M Charan Deepu (20MIS0071)

Vamsi Krishna (20MIS0120)

Kislay Kumar (20MIS0132)

M S S Harshith (20MIS0133)

To
Dr. Neelu Khare
In
G2+TG2 SLOT

School of Advanced Sciences
VIT University, Vellore
Tamil Nadu - 632 014

April 2023

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

SCHOOL OF ADVANCED SCIENCES

Project
Data Mining Techniques (SWE2009)

It is certified that the project entitle "FORECASTING COVID-19 USING ARIMA MODEL" is the bonafide work for Project component of Data Mining

by the following students

Ashish M (20MIS0050)

M Charan Deepu (20MIS0071)

Vamsi Krishna (20MIS0120)

Kislay Kumar (20MIS0132)

M S S Harshith (20MIS0133)

of M.Tech Integrated Software Engineering branch under my supervision in G2+TG2 slot during the Winter Semester -2022-23 at V.I.T. University, Vellore-632 014.

Faculty Signature:

Date:

Table of contents

Individual contribution

M Charan Deepu
IDEA GENERATION OF THE PROJECT Requirement Analysis, LITERATURE REVIEW ANALYSIS, BACKGROUND INFORMATION.

Vamsi Krishna
ARCHITECTURAL DESIGNING of the project by implementing the suitable pattern, DATA PREPROCESSING and DESCRIPTION.

Kislay Kumar
DEVELOPING THE PSEUDO CODE/FLOW of proposed methodology by exploring the logic behind the scene.

MSS Harshith
DEVELOPING THE APPROPRIATE CODE from pseudocode by exploring ML programming tutorials and e-guides with basic UNIT TESTS timely.

Ashish M
TESTING THE SUITABLE CODE to validate the expected accuracy over the long behavioral span and ANALYSIS over results: LEAD TO FUTURE SCOPE.

1. ABSTRACT

One of the most widespread issues in the world right now is COVID-19, and many nations struggle to deal with its effects. A good tool for improving manual analysis identification of COVID19 cases per day, fatalities per day, and the number of patients cured per day is provided by data mining approaches.
COVID-19 epidemics have a negative effect on the nation's economy in addition to harming people's quality of life. The World Health Organization labelled it a global health emergency on January 30, 2020. (WHO). More than 3 million people have contracted this virus by April 28, 2020, and there was no vaccination to stop it. The WHO issued certain safety regulations, but they were merely preventative measures.

Fig : 1.1

In the fight against this epidemic, information technology can be useful, with a concentration on disciplines like data science and machine learning. It is crucial to establish early warning systems that allow one to predict how much a sickness would harm society and, based on that information, the government's actions won't have an impact on the country's economy.
Here, we provide technique for predicting upcoming cases based on historical information. Two strategies, one for predicting the likelihood of contracting the infection and the other for forecasting the number of positive cases, are described using machine learning techniques. Different algorithms were tested, and we cover the algorithm that produced the most accurate results.
It is discussed how to forecast confirmed cases for different Indian states using autoregressive integrated moving average time series(ARIMA).
We performed novel applications with these models, forecasting the number of infected cases (confirmed cases and similarly the number of deaths and recovery), along with the corresponding 90% prediction interval to estimate uncertainty around pointwise forecasts. In this work, we considered two different statistical/time series models that are easily accessible from the "forecast" package. We frequently employ TIME SERIES methods ARIMA Model to study and forecast COVID-19's present and future developments in India.

2. INTRODUCTION

The globe has been frequently threatened by pandemics over the past few years, thus the effects of these pandemics have always had a significant impact on it. The present devastating pandemic is also currently running its course throughout the world. Not only are economies collapsing, but the general well-being and morals of the severely affected countries are also at risk. Understanding the covid-19 virus's natural

progression is crucial for making precise and effective forecasts. New understandings regarding the validity of the designed and validated prediction models are revealed as the COVID-19 pandemic develops and more data is gathered; nonetheless, projecting the pandemic's future requires open reporting and multiple model evaluations.

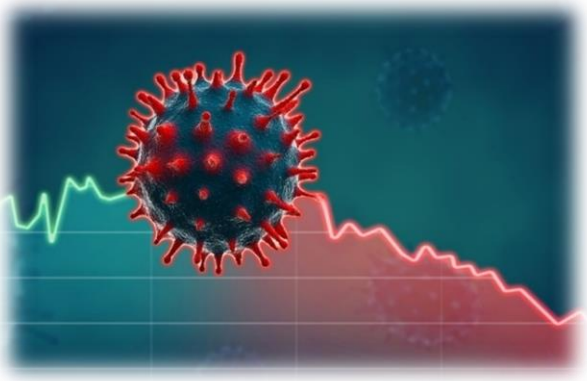


Fig : 2.1

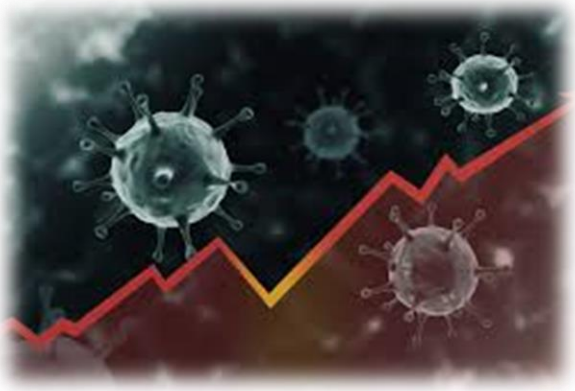


Fig : 2.2
Shows the rapid growth of Covid

Despite these ongoing difficulties, forecasting is still essential to better comprehend the situation and development of COVID-19 and to get ready for the ongoing pandemic's future. Making reliable estimates and using the right models could improve public health decision-making, for instance by influencing how diseases spread. A disease typically advances as a result of infection exposure. 5 hosts are created as a result of this exposure to infection. To accurately and efficiently anticipate the transmission of the virus, we are employing time series analysis together with the model indicated in the abstract.

ML is capable of detecting disease and virus infections more accurately so that patients' disease can be diagnosed at an early stage, the dangerous stages of diseases can be avoided, and there can be fewer patients. In the same manner, ML can be used to automate the task of predicting COVID-19 infection and help forecast future infection tallies of COVID-19.

The objective of this project is to implement the best-performing ML model for predicting and forecasting COVID-19,such that these results can be used to take corrective measures by different government bodies, can make it easier to fight against infectious disease such as COVID-19.

3. LITERATURE REVIEW

1) Machine learning approach for confirmation of COVID-19 cases: positive, negative, death and release
Dutta and Bandyopadhyay SK., Jan 21st, 2020
http://www.iberoamericanjm.periodikos.com.br/article/10.5281/zenodo.3822623/pdf/iberoamericanjm-2-3-172.pdf

2) Prediction and analysis of COVID-19 positive cases using deep learning models: a descriptive case study of India
Arora P, Kumar H, Panigrahi BK., April 16th , 2020
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7298499/

3) A systematic study of covid-19 prediction models of India
Ameet Yadav and chavi rani
https://assets.researchsquare.com/files/rs-2216354/v1/71062b5b-e65f-4a0a-af2a-acb075cb4420.pdf?c=1668790101

4)Short-term forecasting COVID-19 cumulative confirmed cases: perspectives for Brazil
M.H.D.M. Ribeiro,V.C. Mariani,L.D.S. Coelho. 18th APR 2020
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7252162/

5) Time series forecasting of Covid-19 using deep learning models: India USA comparative case study.
Sourabh Shastri Kuljeet Singh SachinKumar Paramjit Kour Vibhakar Mansotra
https://www.sciencedirect.com/science/article/pii/S0960077920306238

6) COVID-19: Forecasting confirmed cases and deaths with a simple time series model
Fotios Petropoulos Spyros Makridakis Neophytos Stylianou
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7717777/

7) Prediction of COVID-19 Trend in India and Its Four Worst Affected States
Punam Bedi
https://link.springer.com/article/10.1007/s42979-021-00598-5

8) COVID-19 Pandemic Forecasting:A Hybrid Approach
Mohamed Hamdy
https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0256971#:~:text=The%20hybrid%20model%20is%20proposed,%2C%20Infectious%2C%20Recovered)%20model.

9) Forecasting COVID19 confirmed cases, deaths and recoveries: Revisiting established time series modeling through novel applications for the USA and Italy
Emrah Gecili , Rhonda D. ,Szczesniak
https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0244173

10) System for Forecasting COVID19 Cases Using Time-Series and Neural Networks Models
Mostafa Abotaleb and Tatiana Makarovsk ikh
https://www.mdpi.com/2673-4591/5/1/46

They used LSTM, GRU for approaching the confirmation. Here the accuracy levels are:

- Confirmed cases: 87%
- Negative cases: 67.8%
- Deceased cases: 62%
- Released cases: 40.5% **[1]**

The model is Deep learning: LSTM, RNN and the accuracy are
Model was highly accurate for short-term predictions (1–3 days) ahead.
MAPE range <3% and weekly forecast 4%-8% **[2]**

The systematic literature review (SLR) methodology was used for this study. SLR is a methodical mapping study that organises, synthesises, and summarises previously published papers on a certain topic and identifies the gaps between earlier studies and more recent studies based on specific research questions (Kitchenham et al., 2010). Research developments and trends in that field may be influenced by the identified research gaps. **[3]**

This paper discuss About new algorithm for considering short term cases and the mostly ARIMA , random forest, SVR, are used and the new algorithms they have discussed is CUBIST, RIDGE and stacking-ensemble learning The dataset was collected from an application programming interface that retrieves the daily data about COVID-19 cases which are publicly available. Based on the performance metrics, SVR, and stacking-ensemble learning outperformed other models. **[4]**

In this paper the authors used just Deep Learning Models for comparing case study in between India and Usa Convolution LSTM based model outperform other models with high accuracy and very less error. **[5]**

This paper discussed about Exponential smoothing model which is used for predicting confirmed cases and death cases in simple time. The proposed time series model has shown good levels of accuracy and uncertainty, especially as more data were accumulated. **[6]**

This paper predicts Covid trend in India Using Modified SEIRD and LSTM Models. For predicting the trend and peak of COVID-19 in India and its four worst-affected states. The modified SEIRD model is based on the SEIRD model. **[7]**

This paper discuss about hybrid approach for forecasting covid-19 by using CNN,LSTM. Hybrid model was developed on a timeseries data set to forecast the number of confirmed cases of COVID-19. **[8]**

The alogirthms are ARIMA and TBATS Model. The ARIMA model was the most consistent model across our different applications and could be preferable over the other models if one must pick a single model for application. **[9]**

Time-series models; ARIMA; BATS; TBATS; Holt's linear trend; NNAR; This algorithm is extensible and various modules can be in it connected to the providing to the construction forecasts by various methods. **[10]**

OTHER REFERENCES

⇒ Palash Ghosh , Rik Ghosh , Bibhas Chakraborty. COVID-19 in India: Statewise Analysis and Prediction.

⇒ Bedi, P., Dhiman, S., Gole, P. et al. Prediction of COVID-19 Trend in India and Its Four Worst-Affected States Using Modified SEIRD and LSTM Models. SN COMPUT. SCI. 2, 224 (2021).

⇒ Sherry Mangla, Ashok Kumar Pathak, Mohd Arshad, Ubydul Haque, Short-term forecasting of the COVID-19 outbreak in India, International Health, 2021.

⇒ Short term prediction model using daily incidence data Naveed N. Merchant Hongwei Zhao 16-oct-2020

⇒ Forecasting the spread of COVID-19 under different reopening strategies Meng Liu, Raphael Thomadsen 23nov2020

⇒ COVID-19: Forecasting confirmed cases and deaths with a simple time series model FotiosPetr opoulos, Sp yrosMakri dakis, Neo phytosStyli anou, 4 dec2020

⇒ Time series forecasting of Covid-19 using deep learning models: India USA comparative case study. SourabhSh astriKuljee tSinghSach inKumarPa ramjitKour Vibhakar Mansotra, 20aug2020.

Analysing these papers have led to a creative thought process for proposed methodology ,which forecasts over major accuracy in time and data scalability.

4. BACKGROUND

❖ **Why COVID-19 Forecasting Is Important?**

Health unions responds to a pandemic of coronavirus disease 2019 (COVID-19) caused by a new coronavirus, SARS-CoV-2, that is spreading from person to person. The federal government works closely with state, tribal, local, and territorial health departments, and other public health partners, to respond to this situation. Forecasts of deaths, hospitalizations, and cases will help inform public health decision making by projecting the likely impact of the COVID-19 pandemic.

❖ **Bringing Together Forecasts for COVID-19 Deaths, Hospitalizations, and Cases in the India.**

Health Unions works with partners to bring together weekly forecasts based on statistical or mathematical models that aim to predict:
National and state numbers of new and total COVID-19 deaths per week for the next 4 weeks.
National and state numbers of new COVID-19 hospitalizations per day for the next 4 weeks.
National, state, and county numbers of new COVID-19 cases per week for the next 4 weeks.

Fig : 4.1 Peak Stages

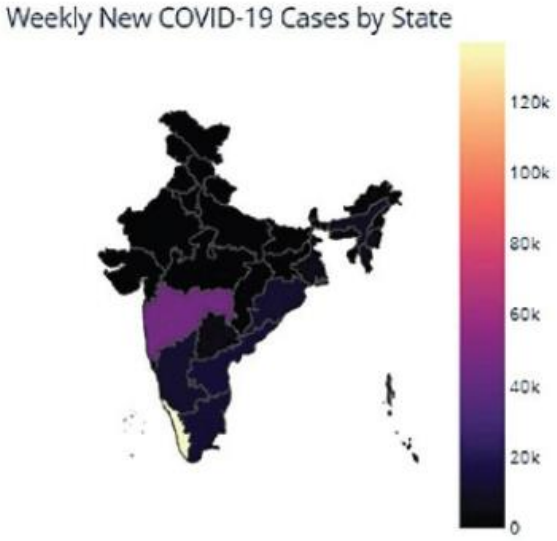


Fig : 4.2

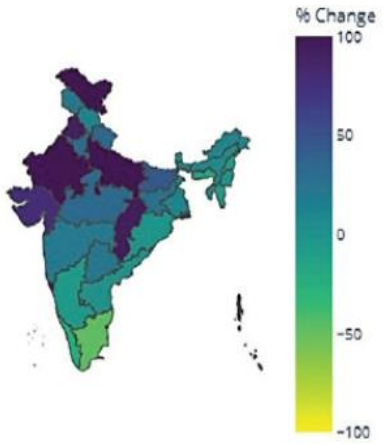


Fig : 4.3 Change Rate

Forecasting teams predict numbers of deaths, hospitalizations, and cases using different types of data (e.g., COVID-19 data, demographic data, mobility data), methods, and estimates of the impacts of interventions (e.g., social distancing, use of face coverings). These forecasts are developed independently and shared publicly. It is important to bring these forecasts together to help understand how they compare with each other and how much uncertainty there is about what may happen in the near future.

Therefore,

- For forecasting through ML, time series analysis may be used, which is an important part of ML.
- It is a univariate type of regression in which the target feature (dependent feature) is forecast using only one input feature (independent feature), which is time.
- It is used to forecast future event values, and it has an important role for forecasting the existence of respiratory diseases such as COVID-19.
- Positive cases are increasing daily, so it is necessary to forecast whether the ratio by which the number is increasing is continuing based on prior observations.
- It is helpful for the government, because based on the forecast, it can plan for resources to control the spread of disease and act for the future so that the growth rate of the infection decreases without affecting more people.
- Forecasts depend completely on past trends, so forecast values cannot be guaranteed. However, this forecasted approximation of events may help authorities to assess forthcoming resource planning to compete with any pandemic situation such as COVID-19.
- We used the most widely used forecasting method, called the ARIMA model for time series forecasting. ARIMA is used for time series data to predict future trends.

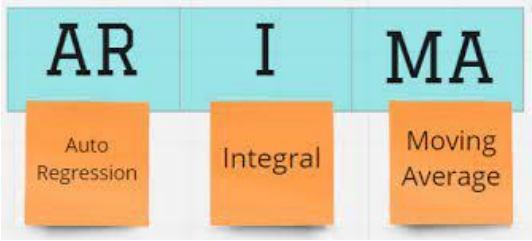


Fig : 4.4

- ARIMA is a form of univariate regression analysis that predicts future values based on differences between values rather than actual values.

5. METHODOLOGY (ARIIMA MODEL)

An autoregressive integrated moving average model is a form of regression analysis that gauges the strength of one dependent variable relative to other changing variables. The model's goal is to predict future by examining the differences between values in the series instead of through actual values.

ARIMA(p,d,q) are one of the most popular econometrics models used to predict time series data such as stock prices, demand forecasting, and even the spread of infectious diseases. An ARIMA model is basically an ARMA model fitted on d-th order differenced time series such that the final differenced time series is stationary.

Here, 'p' is the order of the 'Auto Regressive' (AR) term. It refers to the number of lags of Y to be used as predictors, d is the number of differencing required to make the time series stationary and 'q' is the order of the 'Moving Average' (MA) term. It refers to the number of lagged forecast errors that should go into the ARIMA Model.

A stationary time series is one whose statistical properties such as mean, variance, autocorrelation, etc. are all constant over time. A stationarized series is relatively easy to predict that its statistical properties will be the same in the future as they have been in the past!

An ARIMA model can be understood by outlining each of its components as follows:

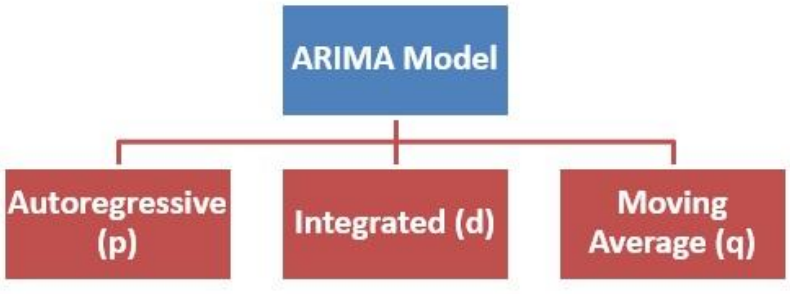


Fig : 5.1

Autoregression (AR): Refers to a model that shows a changing variable that regresses on its own lagged, or prior, values.AR(p) is a regression model with lagged values of y, until p-th time in the past, as predictors. Here, p = the number of lagged observations in the model, ε is white noise at time t, c is a constant and φs are parameters.

$$\hat{y}_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \ldots + \phi_p y_{t-p} + \epsilon_t$$

Integrated (I): Represents the differencing of raw observations to allow the time series to become stationary (i.e., data values are replaced by the difference between the data values and the previous values). The difference is taken d times until the original series becomes stationary. A stationary time series is one whose properties do not depend on the time at which the series is observed.

$By_t = y_{t-1}$ where B is called a backshift operator

Thus, a first order difference is written as

$y'_t = y_t - y_{t-1} = (1 - B)y_t$

In general, a *d* th-order difference can be written as

$y'_t = (1 - B)^d y_t$

Moving average (MA): Incorporates the dependency between an observation and a residual error from a moving average model applied to lagged observations. A moving average model uses a regression-like model on past forecast errors. Here, ε is white noise at time t, c is a constant, and θs are parameters.

$$\hat{y}_t = c + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \ldots + \theta_q \epsilon_{t-q}$$

Combining all of the three types of models above gives the resulting ARIMA(p,d,q) model.

$$\hat{y}'_t = c + \phi_1 y'_{t-1} + \phi_2 y'_{t-2} + \ldots + \phi_p y'_{t-p} + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \ldots + \theta_q \epsilon_{t-q} + \epsilon_t$$

- In general,
  There are 2 types of ARIMA models:
  - Non-Seasonal ARIMA model.
  - Seasonal ARIMA model.

Current forecasting is addressed through ARIMA model(nonseasonal),the future scope of this project tends to work with SARIMA model(Seasonal)

a. ARCHITECTURE

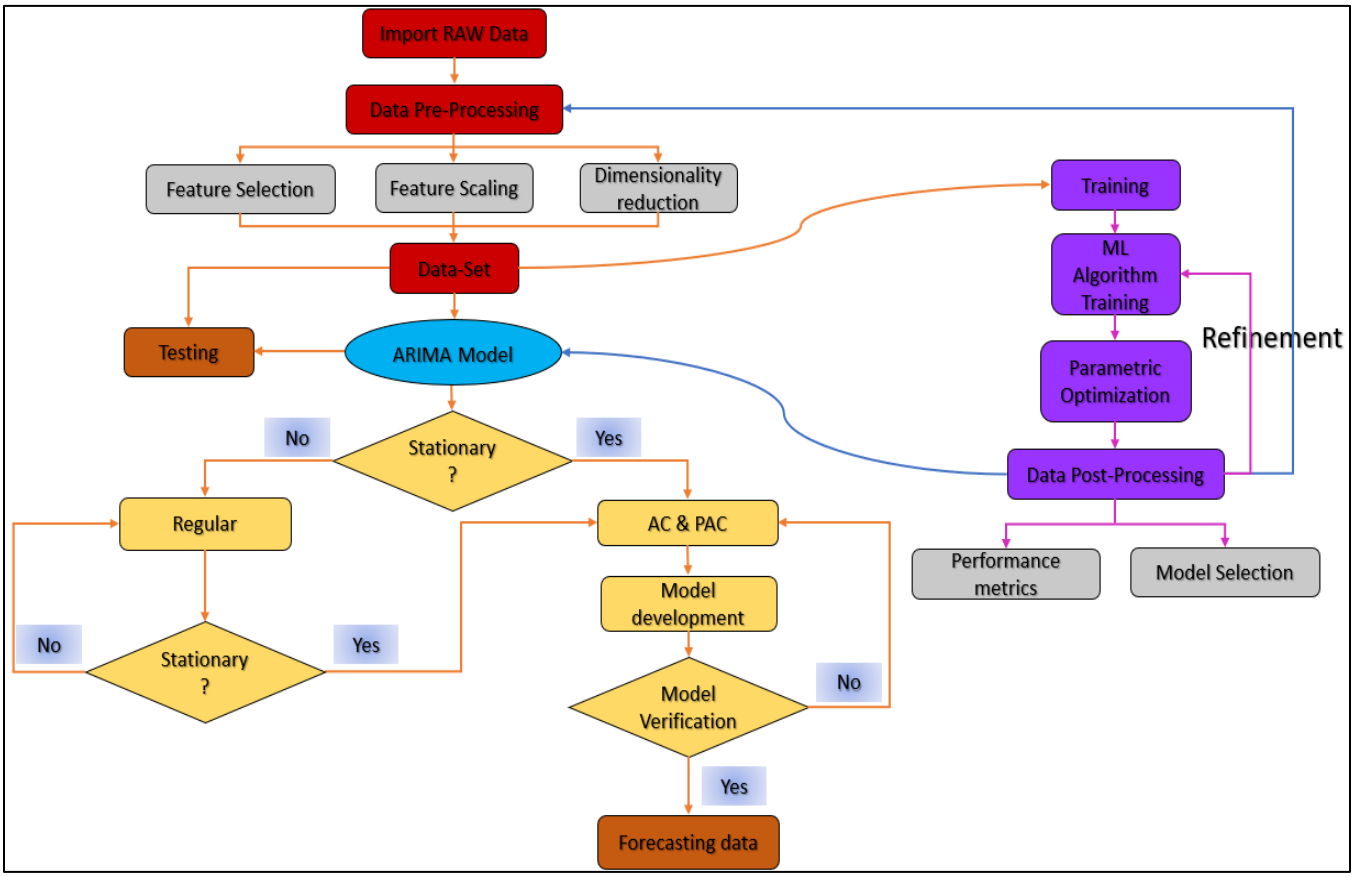


Fig 5.2
Depicts the overall process involved in the proposed methadology

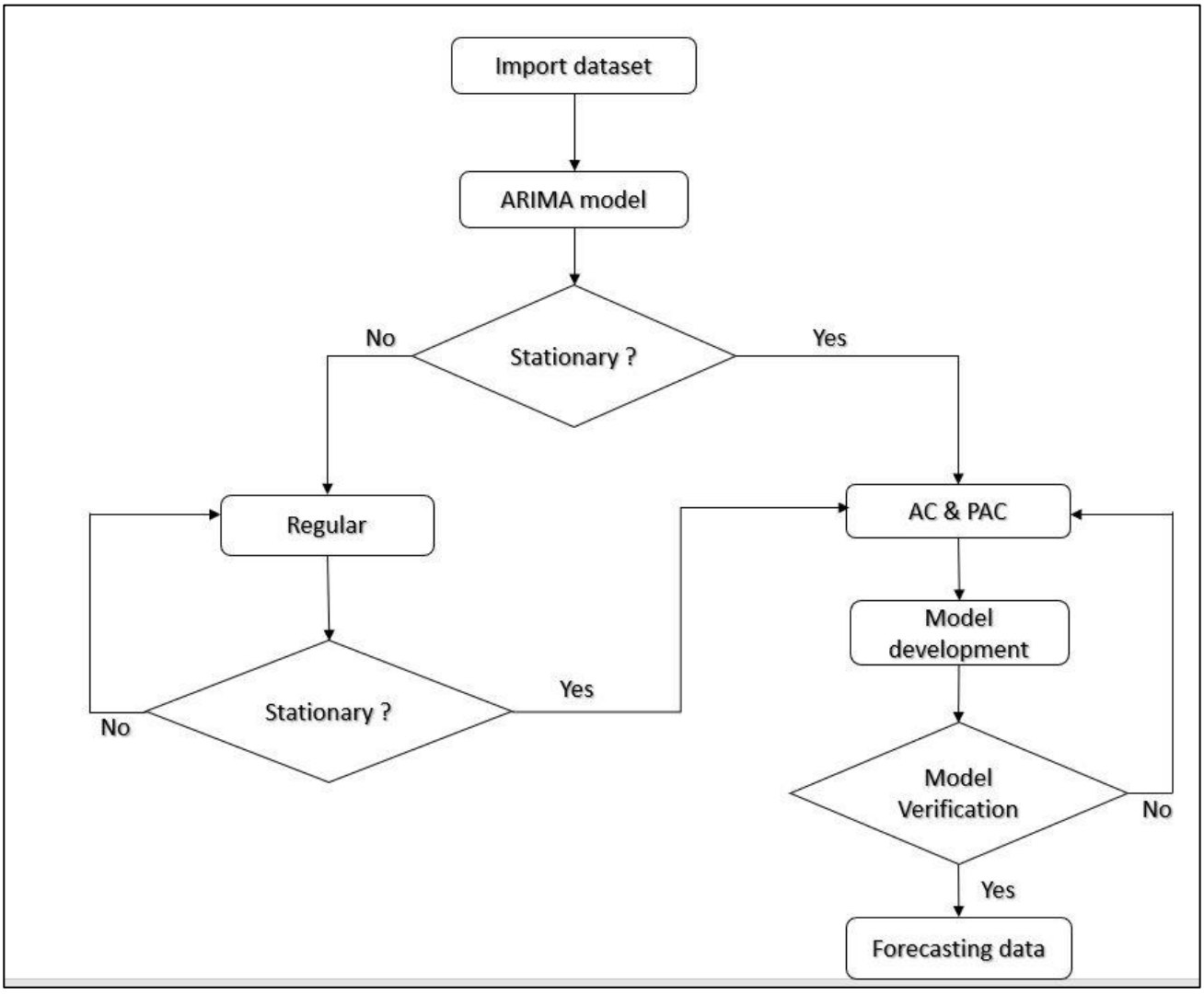


Fig 5.3

Focus on the core process of proposed methodology.

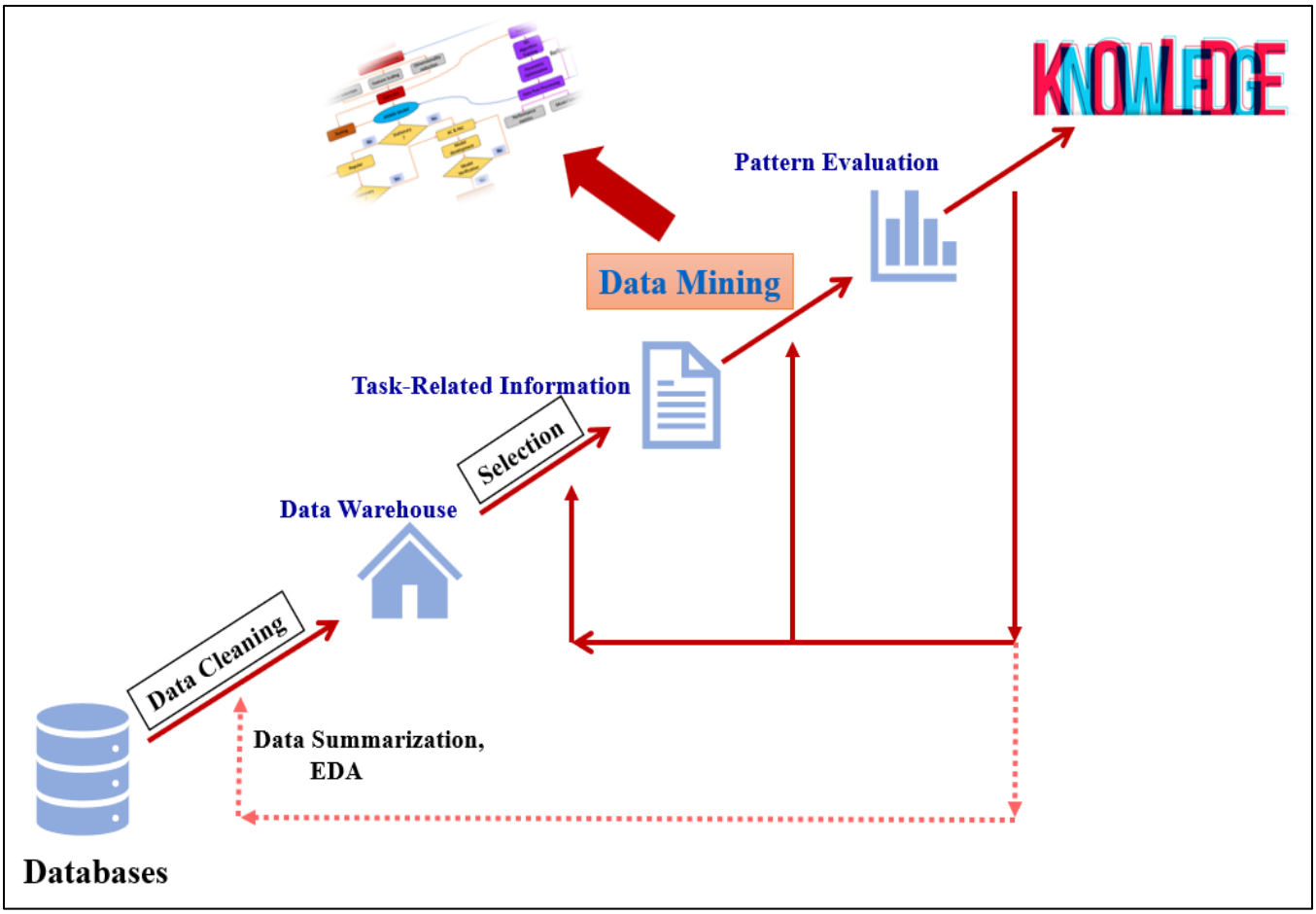


Fig 5.4

Flow of overall process from initial state to achieved state over results.
(proper KDD process: maintained to achieve suitable knowledge)

b. STEPS INVOLVED

1) Import the pre-processed dataset for future forecast.

2) Create the function over rolling average and deviation in order to check P value for the respective selected metric over single region (India),since each country have their own influence over it, then set the relative metric and modify DATE: index field.

//MAKE THE DATA STATIONARY

3) Create test stationary to attain visionary, ADF, P value check.

a) If big variance is observed over the flatten layer with the deviation in rolling mean, then suppress it by log scale.

b) Once the variance gets suppress, P value moves less than 0.05 i.e its ready to apply window method.

c) Subtract the window size mean from log scale data to get data stationary (efficient – mostly constant.

d) Alternative: Period method can be used to stable the data, but this is not preferable mostly. Since P value gets higher.

Once the data gets stationary, try to fit the model by finding p ,d ,q values.

4) Plot ACF(Autocorrelation) and PACF(Partial Autocorrelation) curves in order to find best P,D,Q values.

5) Fit/Predict (PDQ) in ARIMA Model with 80% training and 20% testing data.

6) Accuracy of prediction is plotted via residual score analysis:~0.0…values tested to be accurate and efficient else go through other combination of p,d,q to attain acceptable residual score.

7) Once the above condition is satisfied, Best fit of P,D,Q gives rise to mean square error, residual score, Visual prediction effectively.

8) It's good to determine the behaviour over long span attaining steep residual score and prediction accurately.

9) From the Forecasting results of attained graphs the prediction rate over the given span of specified metric is calculated .

10) Once the model have attained efficiency, it can be upgraded to the idea of SARIMA to remove the seasonality data patterns arised.

c.NOVELTY IN OUR PROJECT

Many projects have been worked over the COVID19 forecasting, but no one have attained the proper accuracy levels for the first training set over the long span and we have achieved that in fact, by making the data stationary using **window size mean technique** via log scale technique alternatively :Period technique have also been used ,this **made us more close towards the accuracy** over prediction in ARIMA Model, the suitable experience of it can also lead us to develop the Model: Forecasting to alter the seasonal data that arises in between, as a **Future scope of the Project**.

Suitable differences over the mentioned novelty :analyzed as shown in **Fig 5.5**, **Fig 5.6** depicts future scope of project

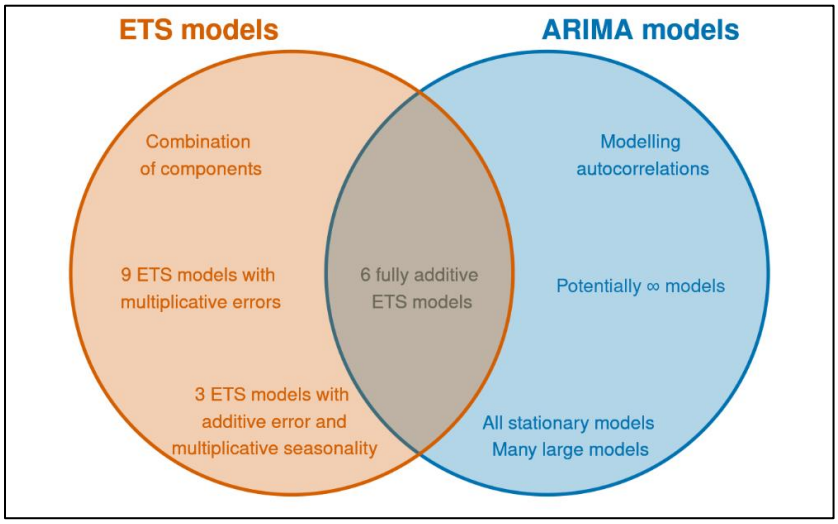


Fig : 5.5

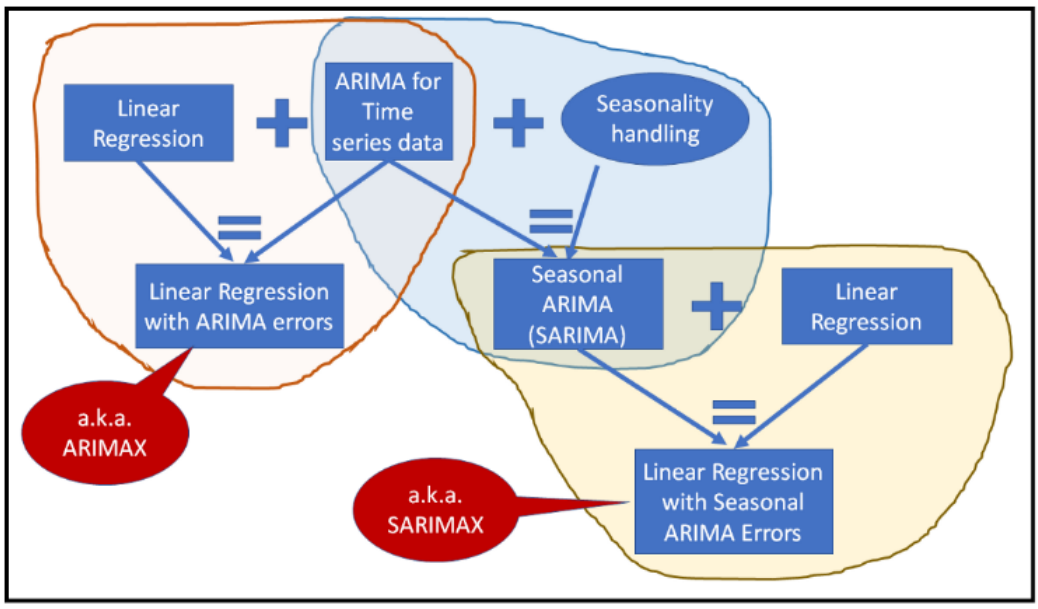


Fig : 5.6 Future Scope

Model	Description	Strengths	Limitations	Applicability to COVID-19 Forecasting	Accuracy	Metrics Used
ARIMA	ARIMA is a time series model that combines autoregression, differencing, and moving average components to effectively capture linear trends and patterns in time series data	Can capture both trend and seasonality in data. Relatively simple to implement. Can handle seasonal patterns effectively.	Requires manual selection of model orders (p, d, q, P, D, Q) which can be challenging.	Applicable for univariate time series data, handles missing data and irregular time intervals effectively. Can be extended to SARIMA model for seasonal variance.	High	ACF, PACF, MAPE, Forecast Error, Root Mean Squared Error (RMSE), Mean Squared Error (MSE)
LSTM	Long Short-Term Memory neural network, a type of recurrent neural network (RNN), that can capture long-term dependencies in time series data.	Can capture complex temporal patterns in data. Can handle non-linear relationships.	May require more data for training compared to traditional models. Can be computationally expensive.	Suitable for data with complex and non-linear patterns, such as COVID-19 data with changing dynamics over time.	Moderate to High	Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Percentage Error (MAPE)

Random Forest	Ensemble learning method that uses decision trees to capture non-linear relationships in data.	Can handle non-linear relationships and interactions in data. Can handle large feature sets.	May overfit with noisy data. May require careful hyperparameter tuning.	Suitable for data with complex interactions and non-linearities, such as COVID-19 data with multiple influencing factors.	Moderate to High	Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Percentage Error (MAPE)
Prophet	Time series forecasting model developed by Facebook that uses additive modeling to capture trend, seasonality, and holiday effects.	Robust to missing data and outliers. Automatically handles trend, seasonality, and holiday effects.	May not perform well for long-term forecasts. Limited flexibility in model customization.	Suitable for data with multiple seasonality or holiday effects, which may be relevant for COVID-19 forecasting during specific time periods, such as holidays or seasonal changes.	Moderate	Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Percentage Error (MAPE)

Table 5.7 : Depicts the comparision between proposed model and other models

6. EXPERIMENTATION

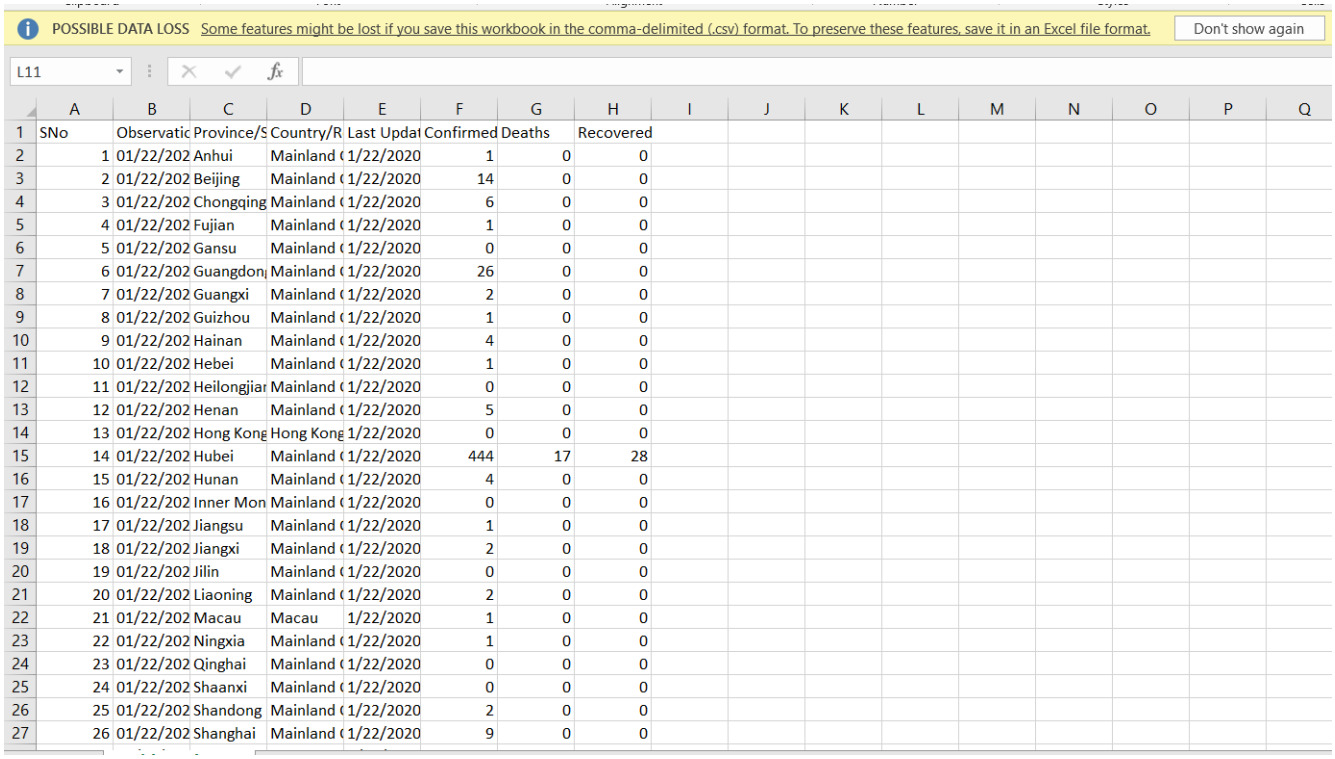
a. SOFTWARE USED :

Google Collab

Colaboratory, sometimes known as "Colab," is a Google Research product. Colab is particularly well suited to machine learning, data analysis, and education. It enables anyone to create and execute arbitrary Python code through the browser. Technically speaking, Colab is a hosted Jupyter notebook service that offers free access to computer resources, including GPUs, and requires no setup to use.

b. DATASET DESCRIPTION

Link: https://drive.google.com/file/d/1QTv4Nk7-gNuLn0LIu1TvDqlUIS4YoKRU/view?usp=sharing



POSSIBLE DATA LOSS Some features might be lost if you save this workbook in the comma-delimited (.csv) format. To preserve these features, save it in an Excel file format. Don't show again

L11

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	SNo	Observatic	Province/S	Country/R	Last Updat	Confirmed	Deaths	Recovered									
2	1	01/22/202	Anhui	Mainland (	1/22/2020	1	0	0									
3	2	01/22/202	Beijing	Mainland (	1/22/2020	14	0	0									
4	3	01/22/202	Chongqing	Mainland (	1/22/2020	6	0	0									
5	4	01/22/202	Fujian	Mainland (	1/22/2020	1	0	0									
6	5	01/22/202	Gansu	Mainland (	1/22/2020	0	0	0									
7	6	01/22/202	Guangdon	Mainland (	1/22/2020	26	0	0									
8	7	01/22/202	Guangxi	Mainland (	1/22/2020	2	0	0									
9	8	01/22/202	Guizhou	Mainland (	1/22/2020	1	0	0									
10	9	01/22/202	Hainan	Mainland (	1/22/2020	4	0	0									
11	10	01/22/202	Hebei	Mainland (	1/22/2020	1	0	0									
12	11	01/22/202	Heilongjiar	Mainland (	1/22/2020	0	0	0									
13	12	01/22/202	Henan	Mainland (	1/22/2020	5	0	0									
14	13	01/22/202	Hong Kong	Hong Kong	1/22/2020	0	0	0									
15	14	01/22/202	Hubei	Mainland (	1/22/2020	444	17	28									
16	15	01/22/202	Hunan	Mainland (	1/22/2020	4	0	0									
17	16	01/22/202	Inner Mon	Mainland (	1/22/2020	0	0	0									
18	17	01/22/202	Jiangsu	Mainland (	1/22/2020	1	0	0									
19	18	01/22/202	Jiangxi	Mainland (	1/22/2020	2	0	0									
20	19	01/22/202	Jilin	Mainland (	1/22/2020	0	0	0									
21	20	01/22/202	Liaoning	Mainland (	1/22/2020	2	0	0									
22	21	01/22/202	Macau	Macau	1/22/2020	1	0	0									
23	22	01/22/202	Ningxia	Mainland (	1/22/2020	1	0	0									
24	23	01/22/202	Qinghai	Mainland (	1/22/2020	0	0	0									
25	24	01/22/202	Shaanxi	Mainland (	1/22/2020	0	0	0									
26	25	01/22/202	Shandong	Mainland (	1/22/2020	2	0	0									
27	26	01/22/202	Shanghai	Mainland (	1/22/2020	9	0	0									

	A	B	C	D	E	F	G	H	I	J	K	L
132379	132378	10/13/202	New Hamp	US	########	9279	456	0				
132380	132379	10/13/202	New Jerse	US	########	221935	16182	0				
132381	132380	10/13/202	New Mexi	US	########	33713	918	0				
132382	132381	10/13/202	New South	Australia	########	4310	53	0				
132383	132382	10/13/202	New York	US	########	480129	33490	0				
132384	132383	10/13/202	Newfound	Canada	########	283	4	271				
132385	132384	10/13/202	Niedersacl	Germany	########	23560	702	19729				
132386	132385	10/13/202	Niigata	Japan	########	179	0	170				
132387	132386	10/13/202	Ningxia	Mainland (	########	75	0	75				
132388	132387	10/13/202	Nizhny No	Russia	########	35350	624	29714				
132389	132388	10/13/202	Noord-Bra	Netherlan	########	26906	1589	0				
132390	132389	10/13/202	Noord-Ho	Netherlan	########	37834	906	0				
132391	132390	10/13/202	Nordrhein	Germany	########	82356	1923	67816				
132392	132391	10/13/202	Norrbotte	Sweden	########	1892	88	0				
132393	132392	10/13/202	Norte de S	Colombia	########	17987	980	15477				
132394	132393	10/13/202	North Car	US	########	234481	3816	0				
132395	132394	10/13/202	North Dak	US	########	28244	357	0				
132396	132395	10/13/202	North Oss	Russia	########	6298	70	4947				
132397	132396	10/13/202	Northern I	UK	########	21898	598	0				
132398	132397	10/13/202	Northern M	US	########	77	2	0				
132399	132398	10/13/202	Northern T	Australia	########	33	0	33				
132400	132399	10/13/202	Northwest	Canada	########	5	0	5				
132401	132400	10/13/202	Nova Scot	Canada	########	1092	65	1023				
132402	132401	10/13/202	Novgorod	Russia	########	6011	86	4006				
132403	132402	10/13/202	Novosibirs	Russia	########	14319	501	12603				
132404	132403	10/13/202	Nuble	Chile	########	6856	135	6406				
132405	132404	10/13/202	Nuevo Lec	Mexico	########	45340	3379	38212				

covid 19 data(2)

L11

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
207244	207243	01/20/202	Ceara	Brazil	########	356985	10243	279170						
207245	207244	01/20/202	Cesar	Colombia	########	36777	1050	34209						
207246	207245	01/20/202	Ceuta	Spain	########	3560	67	163						
207247	207246	01/20/202	Chandigarl	India	########	20623	330	20122						
207248	207247	01/20/202	Channel Is	UK	########	3398	78	3107						
207249	207248	01/20/202	Chechen R	Russia	########	10354	102	7936						
207250	207249	01/20/202	Chelyabins	Russia	########	40652	675	27907						
207251	207250	01/20/202	Cherkasy (	Ukraine	########	44330	500	36307						
207252	207251	01/20/202	Chernihiv (	Ukraine	########	32627	519	21796						
207253	207252	01/20/202	Chernivtsi	Ukraine	########	43374	841	34632						
207254	207253	01/20/202	Chhattisga	India	########	294949	3585	285566						
207255	207254	01/20/202	Chiapas	Mexico	########	8869	1273	0						
207256	207255	01/20/202	Chiba	Japan	########	18574	189	11967						
207257	207256	01/20/202	Chihuahua	Mexico	########	38795	4739	0						
207258	207257	01/20/202	Choco	Colombia	########	5536	172	5076						
207259	207258	01/20/202	Chongqing	Mainland (	########	591	6	584						
207260	207259	01/20/202	Chukotka /	Russia	########	592	4	527						
207261	207260	01/20/202	Chuvashia	Russia	########	18399	747	16501						
207262	207261	01/20/202	Ciudad de	Mexico	########	424666	20028	0						
207263	207262	01/20/202	Coahuila	Mexico	########	56193	4828	0						
207264	207263	01/20/202	Colima	Mexico	########	8770	849	0						
207265	207264	01/20/202	Colorado	US	########	379227	5422	0						
207266	207265	01/20/202	Connectici	US	########	234134	6726	0						
207267	207266	01/20/202	Coquimbo	Chile	########	15683	322	14825						
207268	207267	01/20/202	Cordoba	Colombia	########	31254	1695	28931						
207269	207268	01/20/202	Crimea Re	Ukraine	########	30205	671	24282						
207270	207269	01/20/202	Cundinam	Colombia	########	85809	2134	77691						

covid 19 data(2)

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
306404	306403	05/29/202	Wales	UK	########	212672	5569	0						
306405	306404	05/29/202	Wallis and	France	########	445	7	438						
306406	306405	05/29/202	Walloon B	Belgium	########	40604	0	0						
306407	306406	05/29/202	Washingto	US	########	435849	5765	0						
306408	306407	05/29/202	West Beng	India	########	1354956	15268	1237290						
306409	306408	05/29/202	West Flan	Belgium	########	95724	0	0						
306410	306409	05/29/202	West Virgi	US	########	161287	2792	0						
306411	306410	05/29/202	Western A	Australia	########	1017	9	1006						
306412	306411	05/29/202	Wisconsin	US	########	674003	7830	0						
306413	306412	05/29/202	Wyoming	US	########	60144	719	0						
306414	306413	05/29/202	Xinjiang	Mainland (	########	980	3	977						
306415	306414	05/29/202	Yamagata	Japan	########	1963	42	1692						
306416	306415	05/29/202	Yamaguch	Japan	########	2903	60	2330						
306417	306416	05/29/202	Yamalo-N	Russia	########	39063	419	37848						
306418	306417	05/29/202	Yamanash	Japan	########	1541	19	1390						
306419	306418	05/29/202	Yaroslavl (	Russia	########	40903	605	38968						
306420	306419	05/29/202	Yucatan	Mexico	########	39748	3917	0						
306421	306420	05/29/202	Yukon	Canada	########	84	2	82						
306422	306421	05/29/202	Yunnan	Mainland (	########	352	2	331						
306423	306422	05/29/202	Zabaykalsk	Russia	########	43126	669	41650						
306424	306423	05/29/202	Zacatecas	Mexico	########	30758	2797	0						
306425	306424	05/29/202	Zakarpatti	Ukraine	########	61611	1586	58882						
306426	306425	05/29/202	Zaporizhia	Ukraine	########	102641	2335	95289						
306427	306426	05/29/202	Zeeland	Netherlan	########	29147	245	0						
306428	306427	05/29/202	Zhejiang	Mainland (	########	1364	1	1324						
306429	306428	05/29/202	Zhytomyr (	Ukraine	########	87550	1738	83790						
306430	306429	05/29/202	Zuid-Holla	Netherlan	########	391559	4252	0						

This dataset is from all over country which have confirmed, death and recovered cases.

Attributes:Datatypes::

```
 #   Column           Non-Null Count   Dtype
---  ------           --------------   -----
 0   SNo              306429 non-null  int64
 1   ObservationDate  306429 non-null  object
 2   Province/State   228329 non-null  object
 3   Country/Region   306429 non-null  object
 4   Last Update      306429 non-null  object
 5   Confirmed        306429 non-null  int64
 6   Deaths           306429 non-null  int64
 7   Recovered        306429 non-null  int64
dtypes: int64(4), object(4)
memory usage: 18.7+ MB
```

- The all operation which we are being performed in google colab by taking constraint region = “India”.
- Time-Series data is concerned here to predict the future occurrence.
- Data pre-processing is performed over the given Data Set before applying it to the given Data Model(Arima Model).

Preprocessing of the data is as follows://Concerning India: region to decrease the scalability: involves Exploratory Data Analysis, Data Cleaning, Data Summarization :most required to pass through proposed Data Model.

1.Importing Datasets to google colab

1.1 Importing Libraires

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

1.2 Reading of data from csv file:

```
df_state = pd.read_csv('/content/StatewiseTestingDetails.csv')
df_covid = pd.read_csv('/content/covid_19_india.csv')
df_vacc = pd.read_csv('/content/covid_vaccine_statewise.csv')
```

2.EDA (Exploratory data Analysis)

2.1 Checking of unique and non-unique data values

```
df_state['State'].unique(), df_state['State'].nunique()
```

(array(['India', 'Andaman and Nicobar Islands', 'Andhra Pradesh', 'Arunachal Pradesh', 'Assam', 'Bihar', 'Chandigarh', 'Chhattisgarh', 'Dadra and Nagar Haveli and Daman and Diu', 'Delhi', 'Goa', 'Gujarat', 'Haryana', 'Himachal Pradesh', 'Jammu and Kashmir', 'Jharkhand', 'Karnataka', 'Kerala', 'Ladakh', 'Lakshadweep', 'Madhya Pradesh', 'Maharashtra', 'Manipur', 'Meghalaya', 'Mizoram', 'Nagaland', 'Odisha', 'Puducherry', 'Punjab', 'Rajasthan', 'Sikkim', 'Tamil Nadu', 'Telangana', 'Tripura', 'Uttar Pradesh', 'Uttarakhand', 'West Bengal'], dtype=object), 37)

```
df_covid['State/UnionTerritory'].unique(), df_covid['State/UnionTerritory'].nunique()
```

(array(['Kerala', 'Telengana', 'Delhi', 'Rajasthan', 'Uttar Pradesh', 'Haryana', 'Ladakh', 'Tamil Nadu', 'Karnataka', 'Maharashtra', 'Punjab', 'Jammu and Kashmir', 'Andhra Pradesh', 'Uttarakhand', 'Odisha', 'Puducherry', 'West Bengal', 'Chhattisgarh', 'Chandigarh', 'Gujarat', 'Himachal Pradesh', 'Madhya Pradesh', 'Bihar', 'Manipur', 'Mizoram', 'Andaman and Nicobar Islands', 'Goa', 'Unassigned', 'Assam', 'Jharkhand', 'Arunachal Pradesh', 'Tripura', 'Nagaland', 'Meghalaya', 'Dadra and Nagar Haveli and Daman and Diu', 'Cases being reassigned to states', 'Sikkim', 'Daman & Diu', 'Lakshadweep', 'Telangana', 'Dadra and Nagar Haveli', 'Bihar****', 'Madhya Pradesh***', 'Himanchal Pradesh', 'Karanataka', 'Maharashtra***'], dtype=object), 46)

//zip():referred for pairing together.

```
dict(zip(df_vacc['State'].unique(), df_vacc['State'].unique()))
```

{'India': 'India', 'Andaman and Nicobar Islands': 'Andaman and Nicobar Islands', 'Andhra Pradesh': 'Andhra Pradesh', 'Arunachal Pradesh': 'Arunachal Pradesh', 'Assam': 'Assam', 'Bihar': 'Bihar',

'Chandigarh': 'Chandigarh', 'Chhattisgarh': 'Chhattisgarh', 'Dadra and Nagar Haveli and Daman and Diu': 'Dadra and Nagar Haveli and Daman and Diu', 'Delhi': 'Delhi', 'Goa': 'Goa', 'Gujarat': 'Gujarat', 'Haryana': 'Haryana', 'Himachal Pradesh': 'Himachal Pradesh', 'Jammu and Kashmir': 'Jammu and Kashmir', 'Jharkhand': 'Jharkhand', 'Karnataka': 'Karnataka', 'Kerala': 'Kerala', 'Ladakh': 'Ladakh', 'Lakshadweep': 'Lakshadweep', 'Madhya Pradesh': 'Madhya Pradesh', 'Maharashtra': 'Maharashtra', 'Manipur': 'Manipur', 'Meghalaya': 'Meghalaya', 'Mizoram': 'Mizoram',
...
'Telangana': 'Telangana', 'Tripura': 'Tripura', 'Uttar Pradesh': 'Uttar Pradesh', 'Uttarakhand': 'Uttarakhand', 'West Bengal': 'West Bengal'}

```
df_covid['State/UnionTerritory'].unique(), df_covid['State/UnionTerritory'].nunique()
```

(array(['Kerala', 'Telengana', 'Delhi', 'Rajasthan', 'Uttar Pradesh', 'Haryana', 'Ladakh', 'Tamil Nadu', 'Karnataka', 'Maharashtra', 'Punjab', 'Jammu and Kashmir', 'Andhra Pradesh', 'Uttarakhand', 'Odisha', 'Puducherry', 'West Bengal', 'Chhattisgarh', 'Chandigarh', 'Gujarat', 'Himachal Pradesh', 'Madhya Pradesh', 'Bihar', 'Manipur', 'Mizoram', 'Andaman and Nicobar Islands', 'Goa', 'Unassigned', 'Assam', 'Jharkhand', 'Arunachal Pradesh', 'Tripura', 'Nagaland', 'Meghalaya', 'Dadra and Nagar Haveli and Daman and Diu', 'Cases being reassigned to states', 'Sikkim', 'Daman & Diu', 'Lakshadweep', 'Telangana', 'Dadra and Nagar Haveli', 'Bihar****', 'Madhya Pradesh***', 'Himanchal Pradesh', 'Karanataka', 'Maharashtra***'], dtype=object), 46)

2.2 Data dictionary correction(Tuple convention updation)

```
state_correction_dict = {
  'Bihar****': 'Bihar',
  'Dadra and Nagar Haveli': 'Dadra and Nagar Haveli and Daman and Diu',
  'Madhya Pradesh***': 'Madhya Pradesh',
  'Maharashtra***': 'Maharashtra','Himachal Pradesh':'Himanchal Pradesh','Karnataka':'Karanataka',
'Telengana':'Telangana','Dadra and Nagar Haveli and Daman and Diu':'Daman & Diu'}
```

```
def state_correction(state):
  try:
    return state_correction_dict[state]
  except:
    return state
```

```
df_covid['State/UnionTerritory'] = df_covid['State/UnionTerritory'].apply(state_correction)
```

```
df_vacc['Updated On'] = pd.to_datetime(df_vacc['Updated On'])
df_state['Date'] = pd.to_datetime(df_state['Date'])
df_covid['Date'] = pd.to_datetime(df_covid['Date'])
```

2.3 Data schema (no of columns and no of tuples in data set)

```
df_state.columns, df_state.shape
```
(Index(['Date', 'State', 'TotalSamples', 'Negative', 'Positive'], dtype='object'), (16336, 5))

```
df_covid.columns, df_covid.shape
```
(Index(['Sno', 'Date', 'Time', 'State/UnionTerritory', 'ConfirmedIndianNational', 'ConfirmedForeignNational', 'Cured', 'Deaths', 'Confirmed'], dtype='object'), (18110, 9))

2.4 Merging of Datasets over common attributes//Data Integration.

```
df = df_covid.merge(df_state, left_on = ["State/UnionTerritory","Date"], right_on=["State","Date"], how='left')
```

3.DATA CLEANING

3.1 Get the info of nonnull Count over the DataSet

```
df.info()
```

```
df
```

	Sno	Date	Time	State/UnionTerritory	ConfirmedIndianNational	ConfirmedForeignNational	Cured	Deaths	Confirmed	State	TotalSamples	Negative	Positive
0	1	2020-01-30	6:00 PM	Kerala	1	0	0	0	1	NaN	NaN	NaN	NaN
1	2	2020-01-31	6:00 PM	Kerala	1	0	0	0	1	NaN	NaN	NaN	NaN
2	3	2020-02-01	6:00 PM	Kerala	2	0	0	0	2	NaN	NaN	NaN	NaN
3	4	2020-02-02	6:00 PM	Kerala	3	0	0	0	3	NaN	NaN	NaN	NaN
4	5	2020-02-03	6:00 PM	Kerala	3	0	0	0	3	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...
18106	18106	2021-08-11	8:00 AM	Telangana	-	-	638410	3831	650353	NaN	NaN	NaN	NaN
18107	18107	2021-08-11	8:00 AM	Tripura	-	-	77811	773	80660	NaN	NaN	NaN	NaN
18108	18108	2021-08-11	8:00 AM	Uttarakhand	-	-	334650	7368	342462	NaN	NaN	NaN	NaN
18109	18109	2021-08-11	8:00 AM	Uttar Pradesh	-	-	1685492	22775	1708812	NaN	NaN	NaN	NaN
18110	18110	2021-08-11	8:00 AM	West Bengal	-	-	1506532	18252	1534999	NaN	NaN	NaN	NaN

18111 rows × 13 columns

```
df.shape
```
(18111, 13)

3.2 head():returns the specified no of rows.

```
df_head = df.head(10)
```

```
df_head
```

	Sno	Date	Time	State/UnionTerritory	ConfirmedIndianNational	ConfirmedForeignNational	Cured	Deaths	Confirmed	State	TotalSamples	Negative	Positive
0	1	2020-01-30	6:00 PM	Kerala	1	0	0	0	1	NaN	NaN	NaN	NaN
1	2	2020-01-31	6:00 PM	Kerala	1	0	0	0	1	NaN	NaN	NaN	NaN
2	3	2020-02-01	6:00 PM	Kerala	2	0	0	0	2	NaN	NaN	NaN	NaN
3	4	2020-02-02	6:00 PM	Kerala	3	0	0	0	3	NaN	NaN	NaN	NaN
4	5	2020-02-03	6:00 PM	Kerala	3	0	0	0	3	NaN	NaN	NaN	NaN
5	6	2020-02-04	6:00 PM	Kerala	3	0	0	0	3	NaN	NaN	NaN	NaN
6	7	2020-02-05	6:00 PM	Kerala	3	0	0	0	3	NaN	NaN	NaN	NaN
7	8	2020-02-06	6:00 PM	Kerala	3	0	0	0	3	NaN	NaN	NaN	NaN
8	9	2020-02-07	6:00 PM	Kerala	3	0	0	0	3	NaN	NaN	NaN	NaN
9	10	2020-02-08	6:00 PM	Kerala	3	0	0	0	3	NaN	NaN	NaN	NaN

3.3 Feature Selection(Exclude and Include:Reduction of Attributes)

```
num=df.select_dtypes(exclude=['object']).columns #
num
```

```
Index(['Sno', 'Date', 'Cured', 'Deaths', 'Confirmed', 'TotalSamples', 'Positive'],
      dtype='object')
```

```
strings=df.select_dtypes(include=['object']).columns
strings
```

```
Index(['Time', 'State/UnionTerritory', 'ConfirmedIndianNational', 'ConfirmedForeignNational', 'State',
       'Negative'],
      dtype='object')
```

3.4 Groupby(Grouping rows of same value)

```
df1=df.groupby(['State/UnionTerritory'])['Confirmed'].rank()
df1
```

```
0          1.5
1          1.5
2          3.0
3         21.0
4         21.0
         ...
18106    528.0
18107    492.0
18108    515.0
18109    527.0
18110    512.0
Name: Confirmed, Length: 18111, dtype: float64
```

3.5 Missing value Analysis

```
df.isna()
```

	Sno	Date	Time	State/UnionTerritory	ConfirmedIndianNational	ConfirmedForeignNational	Cured	Deaths	Confirmed	State	TotalSamples	Negative	Positive
0	False	False	False	False	False	False	False	False	False	True	True	True	True
1	False	False	False	False	False	False	False	False	False	True	True	True	True
2	False	False	False	False	False	False	False	False	False	True	True	True	True
3	False	False	False	False	False	False	False	False	False	True	True	True	True
4	False	False	False	False	False	False	False	False	False	True	True	True	True
...	...	...	...	...	...	...	...	...	...	...	...	...	...
18106	False	False	False	False	False	False	False	False	False	True	True	True	True
18107	False	False	False	False	False	False	False	False	False	True	True	True	True
18108	False	False	False	False	False	False	False	False	False	True	True	True	True
18109	False	False	False	False	False	False	False	False	False	True	True	True	True
18110	False	False	False	False	False	False	False	False	False	True	True	True	True

18111 rows × 13 columns

4.Data Summarization:Graphic Displays of Basic Statistical Descriptions

4.1 Histogram

```
import matplotlib.pyplot as plt
plt.hist(df['Cured'], bins=15 )
plt.show()
```

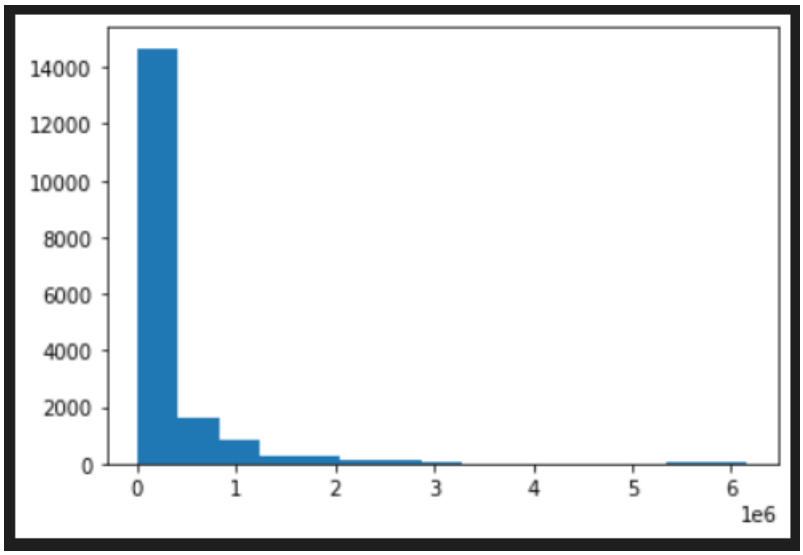


Fig : 6.1

4.2 Scatterplot

```
sns.scatterplot(x=df["Positive"],y= df["Negative"])
```

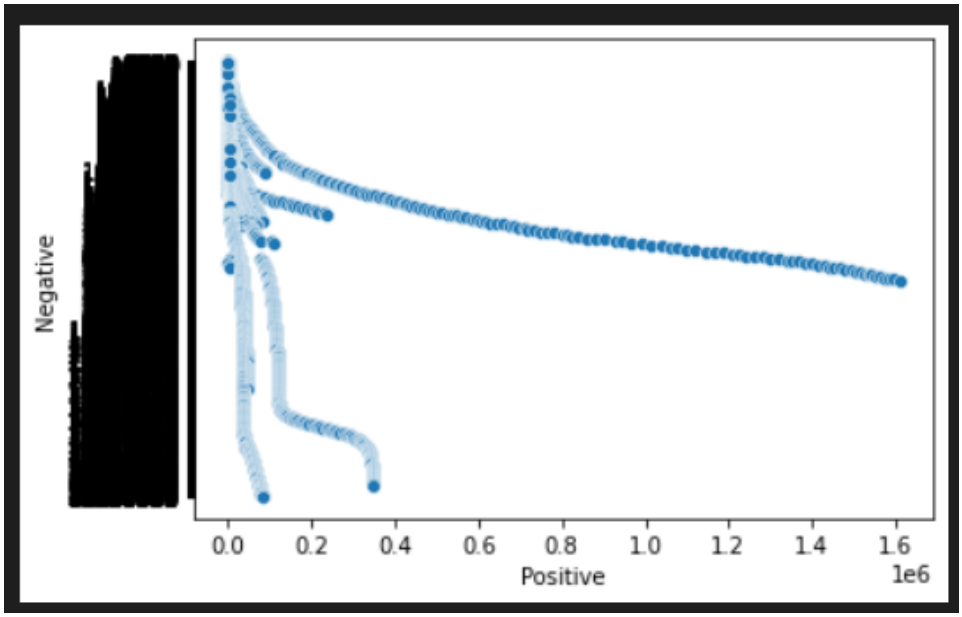


Fig : 6.2

4.3 Central Tendency Measures (Data Description)

```
df_head.describe()
```

	Sno	Cured	Deaths	Confirmed	TotalSamples	Positive
count	10.00000	10.0	10.0	10.000000	0.0	0.0
mean	5.50000	0.0	0.0	2.500000	NaN	NaN
std	3.02765	0.0	0.0	0.849837	NaN	NaN
min	1.00000	0.0	0.0	1.000000	NaN	NaN
25%	3.25000	0.0	0.0	2.250000	NaN	NaN
50%	5.50000	0.0	0.0	3.000000	NaN	NaN
75%	7.75000	0.0	0.0	3.000000	NaN	NaN
max	10.00000	0.0	0.0	3.000000	NaN	NaN

4.4 Refined Dataset & its Commands:

```
plt.show()
df['Day'],df['Month'],df['Year']=df['Date'].dt.day,df['Date'].dt.month,df['Date'].dt.year
df
```

	Sno	Date	Time	State/UnionTerritory	ConfirmedIndianNational	ConfirmedForeignNational	Cured	Deaths	Confirmed	State	TotalSamples	Negative	Positive	Day	Month	Year
0	1	2020-01-30	6:00 PM	Kerala	1	0	0	0	1	NaN	NaN	NaN	NaN	30	1	2020
1	2	2020-01-31	6:00 PM	Kerala	1	0	0	0	1	NaN	NaN	NaN	NaN	31	1	2020
2	3	2020-02-01	6:00 PM	Kerala	2	0	0	0	2	NaN	NaN	NaN	NaN	1	2	2020
3	4	2020-02-02	6:00 PM	Kerala	3	0	0	0	3	NaN	NaN	NaN	NaN	2	2	2020
4	5	2020-02-03	6:00 PM	Kerala	3	0	0	0	3	NaN	NaN	NaN	NaN	3	2	2020
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
18106	18106	2021-08-11	8:00 AM	Telangana	-	-	638410	3831	650353	NaN	NaN	NaN	NaN	11	8	2021
18107	18107	2021-08-11	8:00 AM	Tripura	-	-	77811	773	80660	NaN	NaN	NaN	NaN	11	8	2021
18108	18108	2021-08-11	8:00 AM	Uttarakhand	-	-	334650	7368	342462	NaN	NaN	NaN	NaN	11	8	2021
18109	18109	2021-08-11	8:00 AM	Uttar Pradesh	-	-	1685492	22775	1708812	NaN	NaN	NaN	NaN	11	8	2021
18110	18110	2021-08-11	8:00 AM	West Bengal	-	-	1506532	18252	1534999	NaN	NaN	NaN	NaN	11	8	2021

18111 rows × 16 columns

```
df3=df[df['Year']==2020] # slicing dataframe on basis of Year 2020
df3
```

	Sno	Date	Time	State/UnionTerritory	ConfirmedIndianNational	ConfirmedForeignNational	Cured	Deaths	Confirmed	State	TotalSamples	Negative	Positive	Day	Month	Yea
0	1	2020-01-30	6:00 PM	Kerala	1	0	0	0	1	NaN	NaN	NaN	NaN	30	1	202(
1	2	2020-01-31	6:00 PM	Kerala	1	0	0	0	1	NaN	NaN	NaN	NaN	31	1	202(
2	3	2020-02-01	6:00 PM	Kerala	2	0	0	0	2	NaN	NaN	NaN	NaN	1	2	202(
3	4	2020-02-02	6:00 PM	Kerala	3	0	0	0	3	NaN	NaN	NaN	NaN	2	2	202(
4	5	2020-02-03	6:00 PM	Kerala	3	0	0	0	3	NaN	NaN	NaN	NaN	3	2	202(
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	..
10078	10078	2020-12-31	8:00 AM	Telangana	-	-	278839	1541	286354	Telangana	6882694.0	NaN	NaN	31	12	202(
10079	10079	2020-12-31	8:00 AM	Tripura	-	-	32751	385	33264	Tripura	578723.0	545462	33261.0	31	12	202(
10080	10080	2020-12-31	8:00 AM	Uttarakhand	-	-	84149	1504	90616	Uttarakhand	1777371.0	1686451	NaN	31	12	202(
10081	10081	2020-12-31	8:00 AM	Uttar Pradesh	-	-	562459	8352	584966	Uttar Pradesh	23943169.0	NaN	NaN	31	12	202(
10082	10082	2020-12-31	8:00 AM	West Bengal	-	-	528829	9683	550893	West Bengal	7110430.0	NaN	NaN	31	12	202(

10083 rows × 16 columns

```
df4=df[df['Year']==2021] # slicing dataframe on basis of Year 2021
df4
```

	Sno	Date	Time	State/UnionTerritory	ConfirmedIndianNational	ConfirmedForeignNational	Cured	Deaths	Confirmed	State	TotalSamples	Negative	Positive	Day	Month	Year
10083	10083	2021-01-01	8:00 AM	Andhra Pradesh	-	-	871916	7108	882286	Andhra Pradesh	11884085.0	11001473	NaN	1	1	2021
10084	10084	2021-01-01	8:00 AM	Andaman and Nicobar Islands	-	-	4826	62	4945	Andaman and Nicobar Islands	182631.0	NaN	4946.0	1	1	2021
10085	10085	2021-01-01	8:00 AM	Arunachal Pradesh	-	-	16564	56	16719	Arunachal Pradesh	378364.0	NaN	NaN	1	1	2021
10086	10086	2021-01-01	8:00 AM	Assam	-	-	211910	1045	216211	Assam	6014286.0	NaN	NaN	1	1	2021
10087	10087	2021-01-01	8:00 AM	Bihar	-	-	245476	1397	251743	Bihar	18442165.0	NaN	NaN	1	1	2021
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
18106	18106	2021-08-11	8:00 AM	Telangana	-	-	638410	3831	650353	NaN	NaN	NaN	NaN	11	8	2021
18107	18107	2021-08-11	8:00 AM	Tripura	-	-	77811	773	80660	NaN	NaN	NaN	NaN	11	8	2021
18108	18108	2021-08-11	8:00 AM	Uttarakhand	-	-	334650	7368	342462	NaN	NaN	NaN	NaN	11	8	2021
18109	18109	2021-08-11	8:00 AM	Uttar Pradesh	-	-	1685492	22775	1708812	NaN	NaN	NaN	NaN	11	8	2021
18110	18110	2021-08-11	8:00 AM	West Bengal	-	-	1506532	18252	1534999	NaN	NaN	NaN	NaN	11	8	2021

8028 rows × 16 columns

```
df5=df3.groupby(['State/UnionTerritory'])['Confirmed','Deaths','Cured'].max() # finding max value of columns
```

<ipython-input-32-b88c6b8f6287>:1: FutureWarning: Indexing with multiple keys (implicitly converted to a tuple of keys) will be deprecated, use a list instead.
df5=df3.groupby(['State/UnionTerritory'])['Confirmed','Deaths','Cured'].max() # finding max value of columns

```
df5.reset_index(inplace=True)
df5.columns #gives list of columns in dataframe
```

Index(['index', 'State/UnionTerritory', 'Confirmed', 'Deaths', 'Cured'], dtype='object')

```
df5=df5.drop(labels=[5,34],axis=0)
df5
```

	index	State/UnionTerritory	Confirmed	Deaths	Cured
0	0	Andaman and Nicobar Islands	4941	62	4820
1	1	Andhra Pradesh	881948	7104	871588
2	2	Arunachal Pradesh	16711	56	16549
3	3	Assam	216139	1043	211838
4	4	Bihar	251348	1393	245156
6	7	Chhattisgarh	278540	3350	263251
7	8	Daman & Diu	3375	2	3364
8	9	Delhi	624795	10523	608434
9	10	Goa	50981	737	49313
10	11	Gujarat	244258	4302	229977
11	12	Haryana	262054	2899	255356
12	13	Himanchal Pradesh	55114	931	51387
13	14	Jammu and Kashmir	120744	1880	115830
14	15	Jharkhand	114873	1027	112206
15	16	Karanataka	918544	12081	894834
16	17	Kerala	755718	3042	687104
17	18	Ladakh	9447	127	9132
18	19	Lakshadweep	0	0	0
19	20	Madhya Pradesh	240947	3595	227965
20	21	Maharashtra	1928603	49463	1824934
21	22	Manipur	28137	354	26601
22	23	Meghalaya	13408	139	13085
23	24	Mizoram	4204	8	4091
24	25	Nagaland	11921	79	11624
25	26	Odisha	329306	1871	325103
26	27	Puducherry	38096	633	37100
27	28	Punjab	166239	5331	157043
28	29	Rajasthan	307554	2689	295030
29	30	Sikkim	5877	127	5218
30	31	Tamil Nadu	817077	12109	796353
31	32	Telangana	286354	1541	278839
32	33	Tripura	33264	385	32751
33	35	Uttar Pradesh	584966	8352	562459
35	37	West Bengal	550893	9683	528829

4.5 Bar plot

```
plt.bar(df5['State/UnionTerritory'],df5['Confirmed'])
plt.xticks(rotation = 90)
plt.title("State/union territory wise confirmed cases in 2020")
```

Text(0.5, 1.0, 'State/union territory wise confirmed cases in 2020')

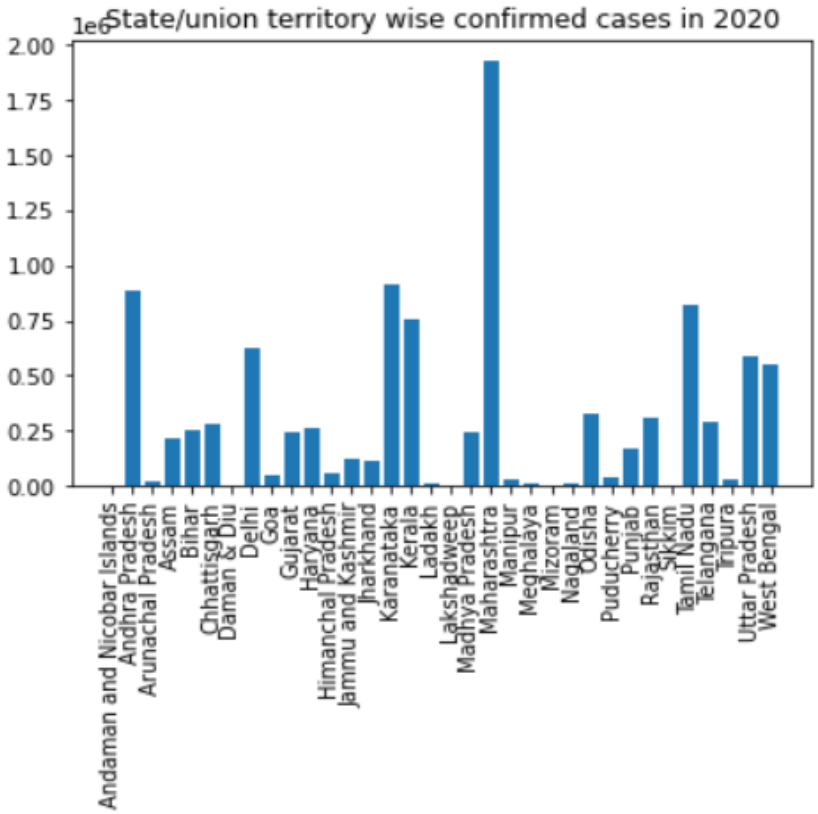


Fig 6.3

4.6 Subplots

```
fig, axs = plt.subplots(3, sharex=True, sharey=True)
fig.suptitle("State/union territory wise confirmed cases,deaths and cured cases in 2020")
axs[0].plot(df5['State/UnionTerritory'],df5['Confirmed'])
axs[1].plot(df5['State/UnionTerritory'],df5['Deaths'])
axs[2].plot(df5['State/UnionTerritory'],df5['Cured'])
plt.xticks(rotation = 90)
```

([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24,
...
30, 31, 32, 33]

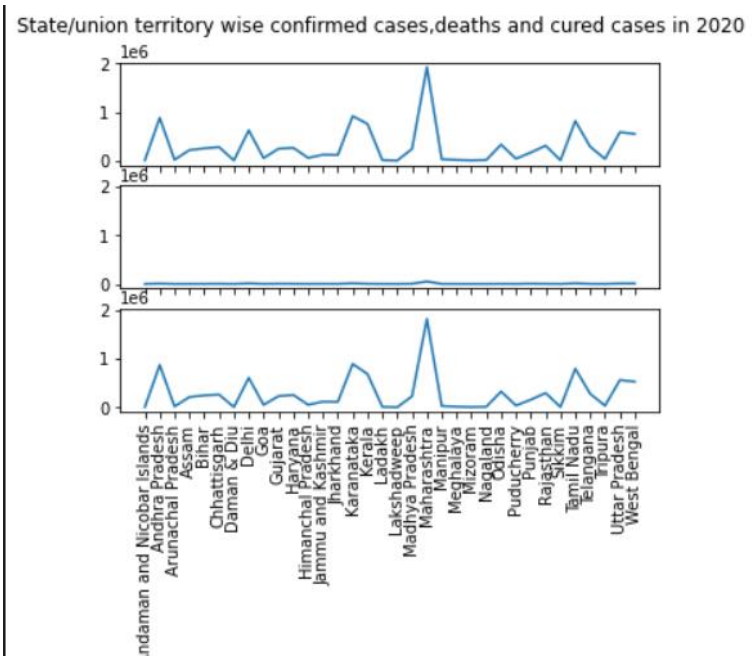


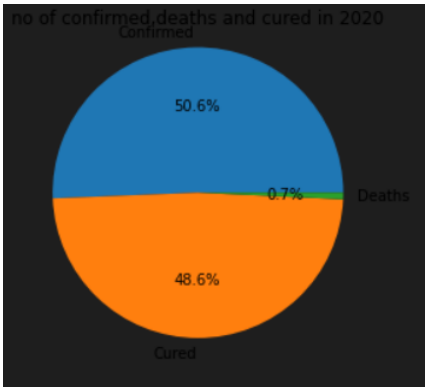
Fig 6.4

```
data=df5['Confirmed'].sum(),df5['Cured'].sum(),df5['Deaths'].sum()
data
```

(10156376, 9757164, 146918)

4.7 Pie Chart

```
my_labels = 'Confirmed','Cured','Deaths'
plt.pie(data,labels=my_labels,autopct='%1.1f%%')
plt.title('no of confirmed,deaths and cured in 2020')
plt.axis('equal')
plt.show()
```



Blue :
Confirmed : 50.6%

Middle :
Death : 0.7%

Orange :
Cured : 40.6%

Fig : 6.5

7. RESULTS & DISCUSSIONS

a. CORE MODULES:

```
#1
def test_stationarity(timeseries):
  import matplotlib.pyplot as plt
  rolmean = timeseries.rolling(window=5).mean()
  rolstd = timeseries.rolling(window=5).std()
  #Visual Inspection#
  orig = plt.plot(timeseries , label='Original')
  mean = plt.plot(rolmean, label='Rolling mean')
  std = plt.plot(rolstd, label='Rolling std')

  plt.legend(loc='best')
  plt.title('Timeseries data with rolling mean and std. dev.')
  plt.show()
  #Stationarity check
  from statsmodels.tsa.stattools import adfuller

  dftest = adfuller(timeseries)
  dfoutput = pd.Series(dftest[0:4], index = ['The test statistic','Mackinnon approximate p-
value','#usedLags','NOBS'])

  print(dfoutput)
```

```
#2
file_location='/content/covid_19_data.csv'

import pandas as pd

covid_19_dataset = pd.read_csv(file_location)

covid_19_dataset.head()
```

Output:

	SNo	ObservationDate	Province/State	Country/Region	Last Update	Confirmed	Deaths	Recovered
0	1	01/22/2020	Anhui	Mainland China	1/22/2020 17:00	1.0	0.0	0.0
1	2	01/22/2020	Beijing	Mainland China	1/22/2020 17:00	14.0	0.0	0.0
2	3	01/22/2020	Chongqing	Mainland China	1/22/2020 17:00	6.0	0.0	0.0
3	4	01/22/2020	Fujian	Mainland China	1/22/2020 17:00	1.0	0.0	0.0
4	5	01/22/2020	Gansu	Mainland China	1/22/2020 17:00	0.0	0.0	0.0

```
#3
#changing the datatype of conf_cases fromobject => datetime
covid_19_dataset['ObservationDate']=pd.to_datetime(covid_19_dataset['ObservationDate'], infer_dateti
me_format=True)

indexed_covid_19_dataset = covid_19_dataset.set_index(['ObservationDate'])

indexed_covid_19_dataset.head()
```

Output:

ObservationDate	SNo	Province/State	Country/Region	Last Update	Confirmed	Deaths	Recovered
2020-01-22	1	Anhui	Mainland China	1/22/2020 17:00	1	0	0
2020-01-22	2	Beijing	Mainland China	1/22/2020 17:00	14	0	0
2020-01-22	3	Chongqing	Mainland China	1/22/2020 17:00	6	0	0
2020-01-22	4	Fujian	Mainland China	1/22/2020 17:00	1	0	0
2020-01-22	5	Gansu	Mainland China	1/22/2020 17:00	0	0	0

```
#4
indexed_covid_19_dataset.head()
covid_data_india = indexed_covid_19_dataset['Country/Region'] == 'India'

covid_india_dataset = indexed_covid_19_dataset[covid_data_india]
#Making data as uni Variate
covid_india_confirmed_case_dataset = covid_india_dataset['Confirmed']

covid_india_confirmed_agg_dataset = covid_india_confirmed_case_dataset.groupby(['ObservationDate'
]).sum()

test_stationarity(covid_india_confirmed_agg_dataset)
```

Output: **Fig : 7.1**

```
import numpy as np

covid_india_confirmed_agg_dataset_log_scaled = np.log(covid_india_confirmed_agg_dataset)

test_stationarity(covid_india_confirmed_agg_dataset_log_scaled)
```

Output: **Fig : 7.2**

```
ma = covid_india_confirmed_agg_dataset_log_scaled.rolling(window=5).mean()
```

```
covid_india_confirmed_agg_dataset_log_scaled_minus_ma = covid_india_confirmed_agg_dataset_log_scaled - ma
covid_india_confirmed_agg_dataset_log_scaled_minus_ma.dropna(inplace=True)
test_stationarity(covid_india_confirmed_agg_dataset_log_scaled_minus_ma)
```

Output: **Fig 7.3**

```
covid_india_confirmed_agg_dataset_log_scaled_ps = covid_india_confirmed_agg_dataset_log_scaled.diff(periods=5)

covid_india_confirmed_agg_dataset_log_scaled_ps.dropna(inplace=True)
test_stationarity(covid_india_confirmed_agg_dataset_log_scaled_ps)
```

Output: **Fig : 7.4**

```
from statsmodels.tsa.stattools import acf
from statsmodels.tsa.stattools import pacf

from statsmodels.graphics.tsaplots import plot_acf
from statsmodels.graphics.tsaplots import plot_pacf

import matplotlib.pyplot as plt

lag_acf = acf(covid_india_confirmed_agg_dataset_log_scaled_minus_ma, nlags = 32)
lag_pacf = pacf(covid_india_confirmed_agg_dataset_log_scaled_minus_ma, nlags = 16)

fig, ax = plt.subplots(1,2,figsize=(20,5))

plot_acf(lag_acf, ax=ax[0])
plot_pacf(lag_pacf,lags=7, ax=ax[1])

plt.show
```

Output: **Fig : 7.5**

```
def predict(timeseries,p,d,q):

  from statsmodels.tsa.arima.model import ARIMA

  from sklearn.model_selection import train_test_split

  timeseries.dropna(inplace=True)

  train, test = train_test_split(timeseries, test_size = 0.20, shuffle=False)

  #ARIMA model
```

```
 model_arima = ARIMA(train, order=(p,d,q))

 model_arima_fit = model_arima.fit()

 predictions = model_arima_fit.predict(start='2021-02-22', end='2021-05-29')
#Test
 from sklearn.metrics import mean_squared_error

 error= mean_squared_error(test, predictions)

 print('Test MSE %.5f' % error)
#end tests
 predict = np.exp(predictions)
 test_set = np.exp(test)

 plt.plot(test_set)
 plt.plot(predict, color='red')
 plt.show()

 from pandas import DataFrame

 residual = DataFrame(model_arima_fit.resid)

 residual.plot(kind='kde')
predict(covid_india_confirmed_agg_dataset_log_scaled_minus_ma,40,5,3)
```

Output: **Fig : 7.6, 7.7**

b.GRAPHS : **(Output Results)**

i).After passing the covid19_india_confirmed data set to the test_Stationarity function the Data is visualised through the below plots.

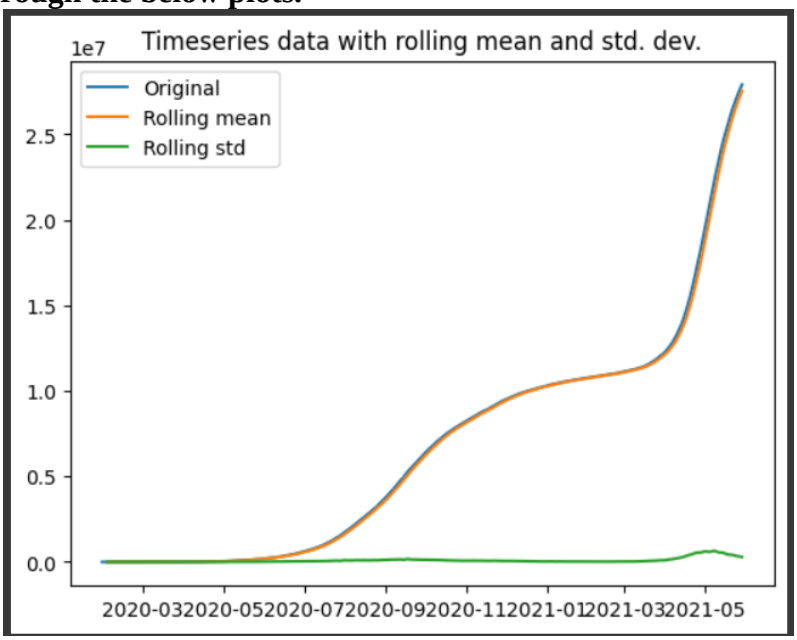


Fig : 7.1

Here, the Rolling mean and Original data are almost same and the standard deviation is constant. But, the Rolling mean still contains the trianed part, which is removed in next step.

ii).To remove the variance log scale is applied to the data.

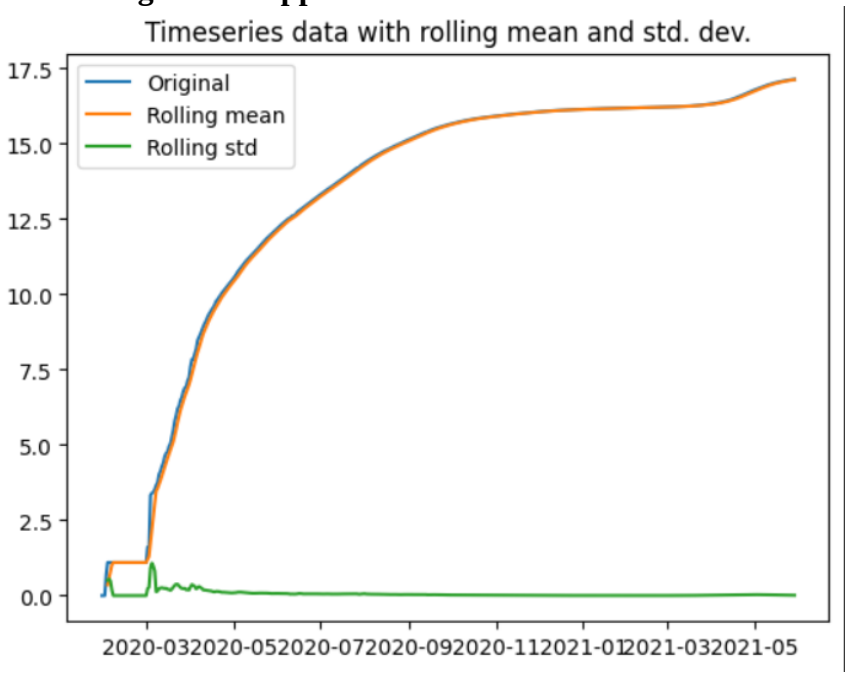


Fig : 7.2

The variance is reduced and the rolling mean and original data are also same but not constant. The Rolling standard is constant, but the Mackinnon approximate p-value is not at acceptable level.

iii.Removing Trained part:

Method 1:
Moving average with window size 5 is subtracted from the log scale data.

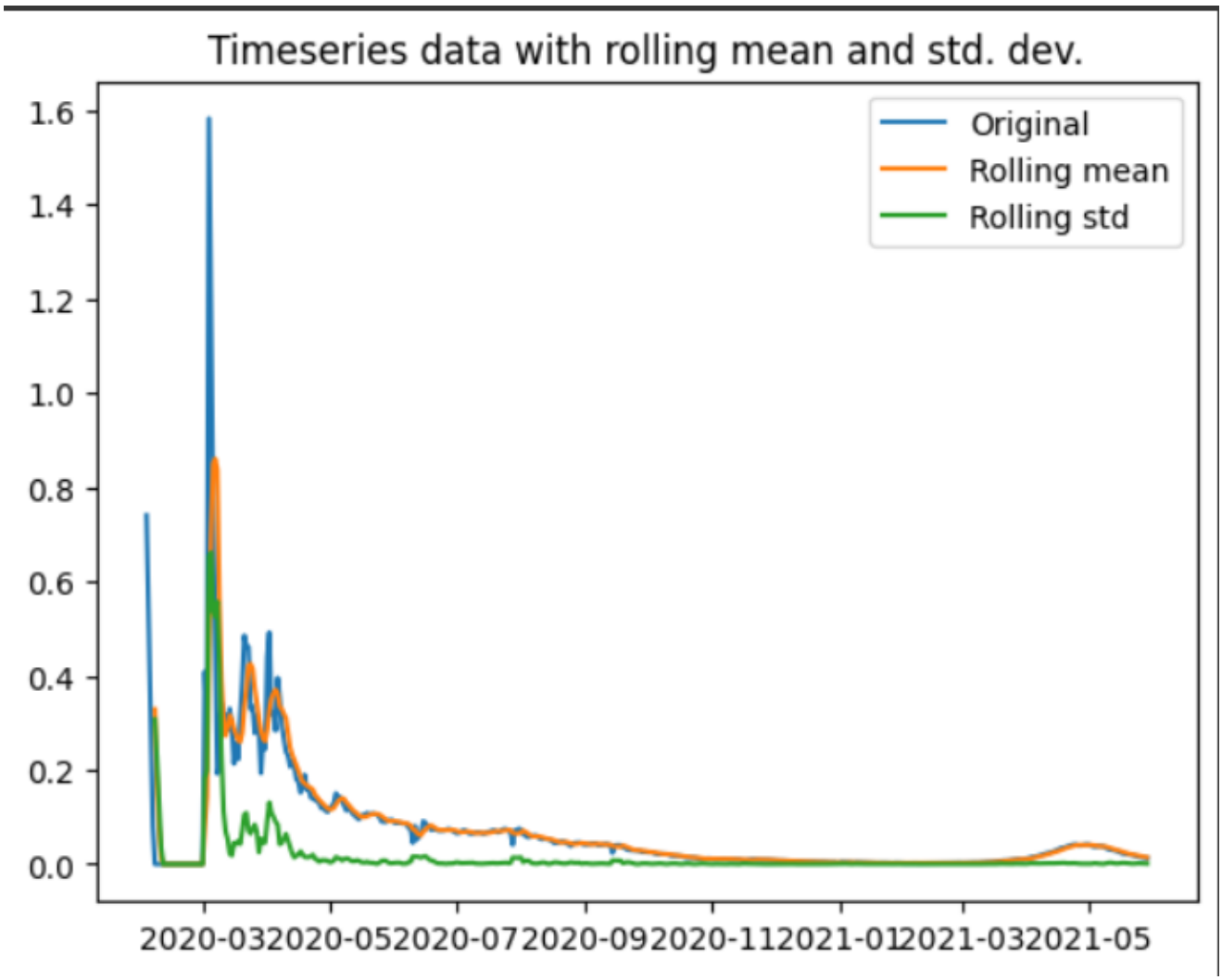


Fig : 7.3

Rolling mean is almost constant as the time proceeds, at starting position due to lack of previous data the anomalies occur. The observed Mackinnon approximate p-value is also at the acceptable level.

Method 2:
Using Period shift

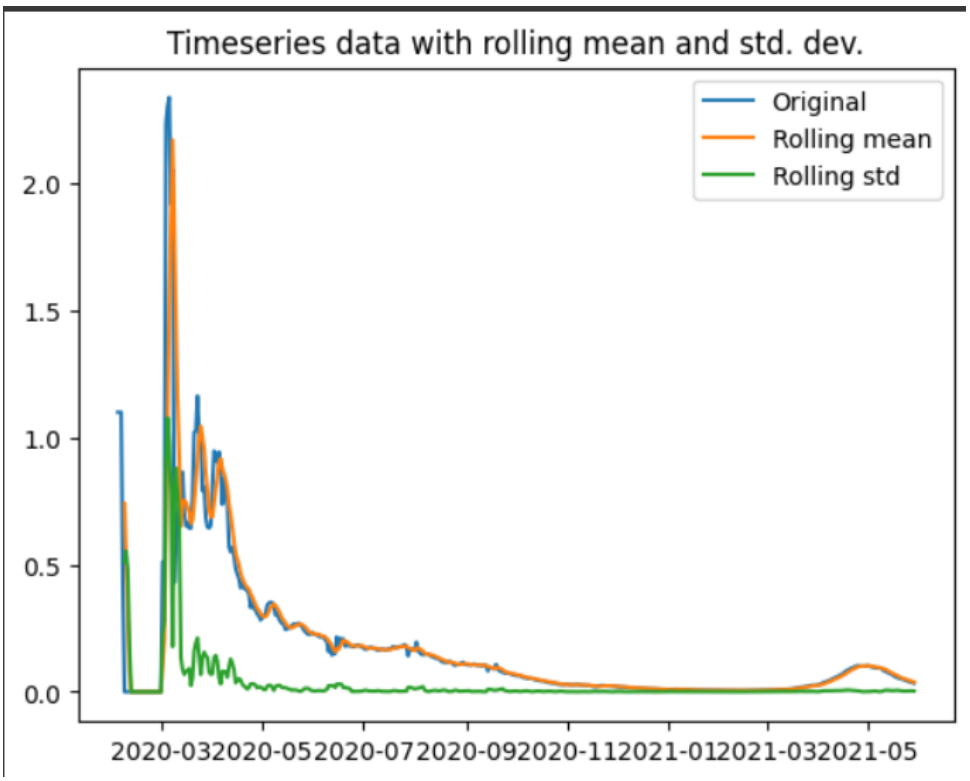


Fig : 7.4

Rolling mean is almost constant as the time proceeds. The Mackinnon approximate p-value is higher than the acceptable level, so method 1 is chosen.

iv) Auto & Partial correlation(ACF,PACF)

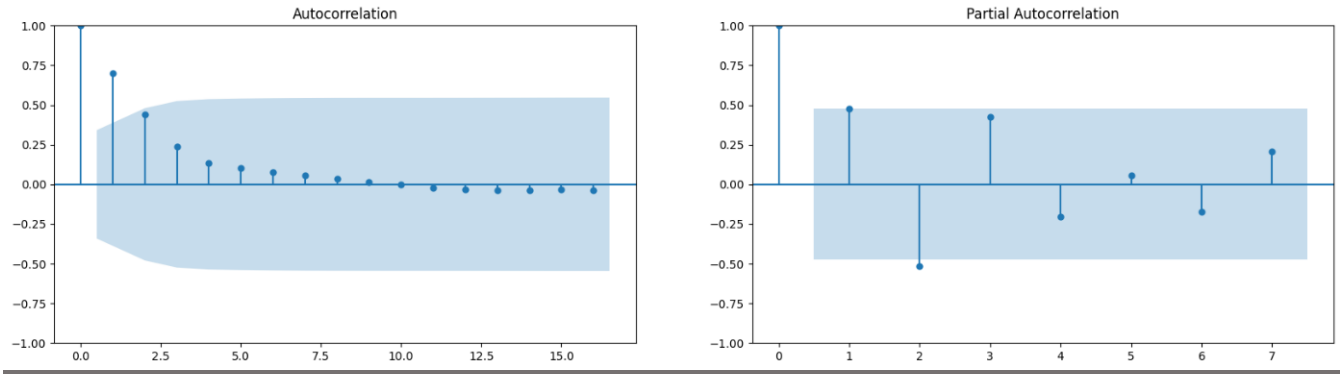


Fig : 7.5

For the ARIMA model the P & Q values are must, which are derived from the Auto & Partial correlation.

1) In the Autocorrelation plot the graph meets the x-axis at 10. So, the q (Moving average) starts form 10.

2) In the Partial Autocorrelation plot the graph meets the negative x-axis at 2. So, the p (Auto regressive lag) starts form 2.

After experimenting with various combinations, best results are obtained at p = 40 , q =5, d=3.

v) ARIMA model representation for the confirmed cases using the above parameters.(over long behaviour: time period)

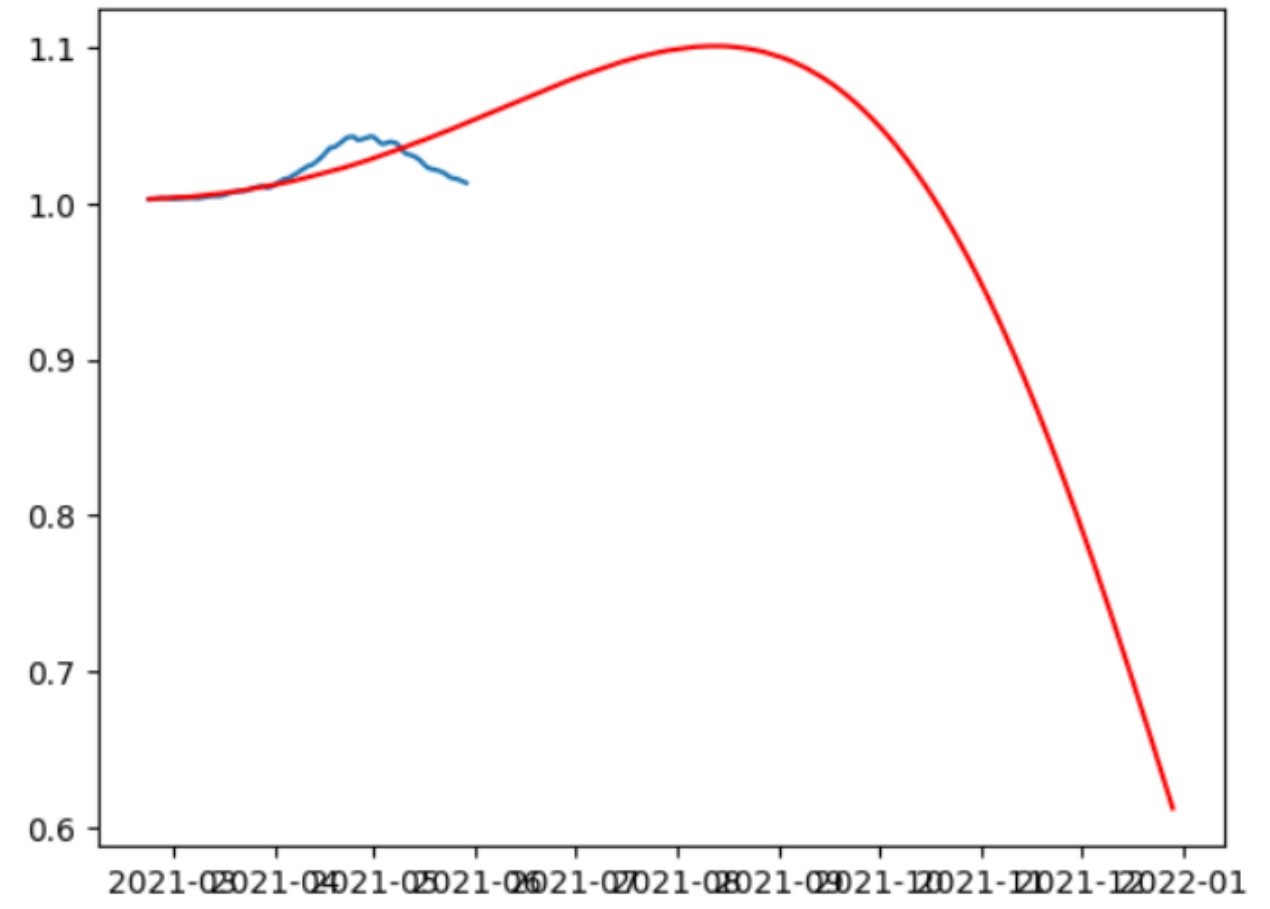


Fig : 7.6

The red line represents the actual data, and the blue line represents the predicted data for the time period.

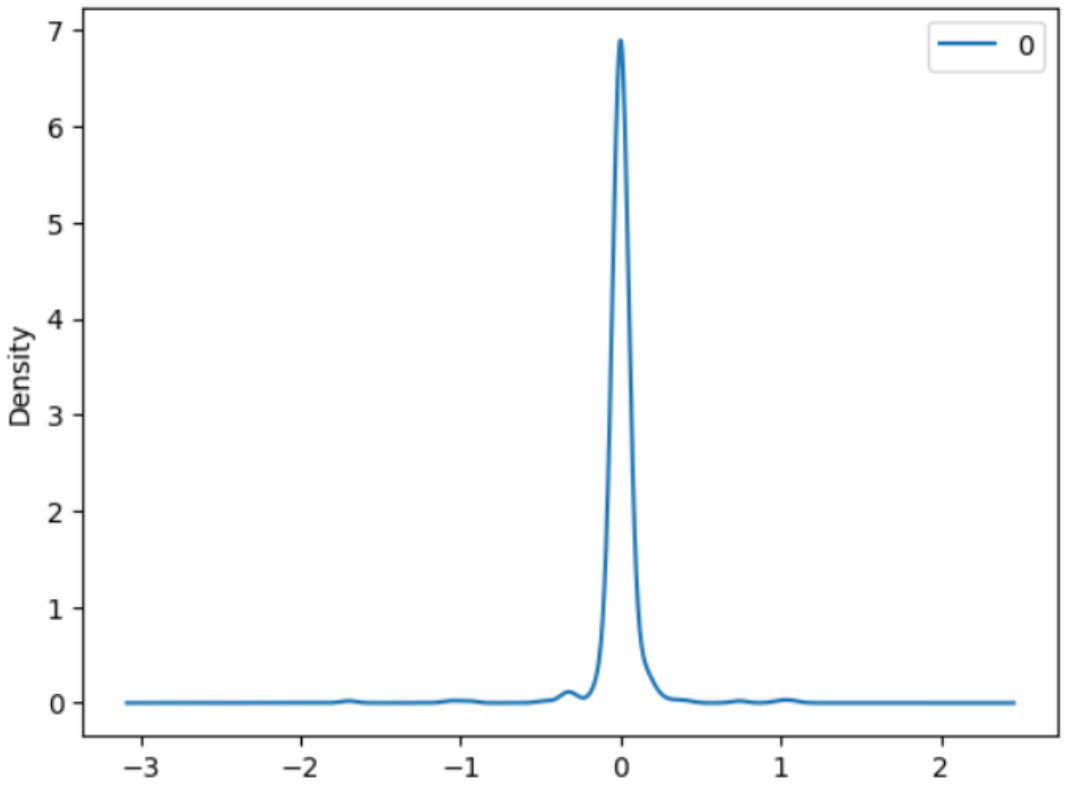


Fig : 7.7

The peak is observed at 0 and the relative points near it are also constant, suggesting that the Residual score is good. By this we can infer that our model has done a good at predicting the timeseries forecast.

8. CONCLUSION

- ARIMA (Auto Regressive Integrated Moving Average) uses it's best fit via

    - p (auto regressive lag)
    - q (Moving average/ Error part)
    - d (Order of differentiation – which will effect from the values of p and q)

    i.e If p or q is high change d value, in order to attain best plot with efficient mean square error, steep residual score & effective visualisation for the future prediction.

- Therefore, by attaining this Model overall pattern evaluation can be done effectively in order to bring out the required Knowledge: Forecasting time series, from the given Data i;e an Efficient KDD(Knowledge Data Discovery) process is carried out throughout the project as referred in **Fig 5.3.**

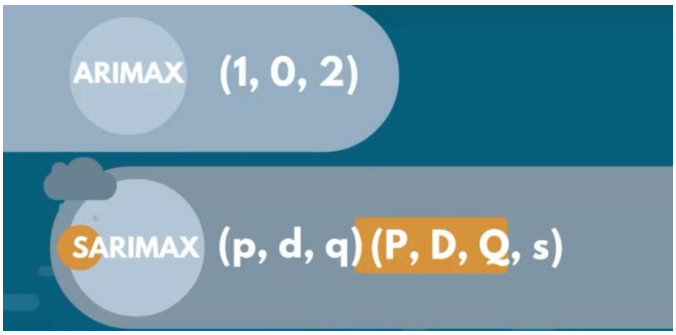
- **Future Scope**

To address the seasonality differentiation the data model can be upgraded using SARIMAX to remove the seasonal data.

P – New Auto Regressive lag

D – New Differential Value

Q – New Moving Average



S – Differential Parameter for Seasonal Data

Fig : 8.1

9.ACKNOWLEDGEMENT

In the accomplishment of this project successfully, many people have best owned upon us their blessing and the heart pledged support, this time we're utilizing to thank all the people who have been concerned with this project.

Then we would like to thank our faculty **Prof. Neelu Khare**, whose valuable guidance has been the ones that helped us patch this project and make it full proof success. Her suggestions and her instructions have served as the major contributor towards the completion of the project. Then we would like to thank our Seniors who have helped us with their valuable suggestions and guidance has been very helpful in various phases of the completion of this project.

Last but not the least we would like to thank our team members who worked sincerely in completion of this project with a successful and fruitful outcome.

THANK YOU